MASSEY UNIVERSITY

LOGO PROGRAMMING: INSTRUCTIONAL METHODS AND PROBLEM SOLVING

by

WING KEE AU

A THESIS

PRESENTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

EDUCATION

FACULTY OF EDUCATION

MASSEY UNIVERSITY

PALMERSTON NORTH

NEW ZEALAND

DECEMBER, 1992

Abstract

This study was conducted to examine the effects of the learning of programming on the problem solving abilities of primary school children. Two programming languages were used: LOGO and BASIC. The aim of the study was threefold. First, the study compared the two programming languages in the development of problem solving skills. Second, this study compared the effectiveness of two different instructional methods in the teaching of LOGO programming: process-oriented and content-oriented approaches. The third aim of this study was to examine the social interactions among the learners who engaged in LOGO and BASIC programming.

The sample for the study comprised 73 subjects drawn from a primary school in Palmerston North, New Zealand. Subjects were screened initially on their background knowledge in programming to ensure that they did not possess any substantial knowledge in programming before participating in the study. The subjects were then randomly assigned to four groups: LOGO process-oriented, LOGO content-oriented, BASIC, and control. These groups of subjects were then pre-tested on a number of problem solving measures: Rule-naming task, Tower of Hanoi, Torrance Test of Creative Thinking, Object Assembly, Block Design, Picture Assembly, and PAT Mathematics. The intervention phase in the form of learning programming of either LOGO or BASIC then took place for the three experimental groups. During the intervention, observations on the social interactions of teachers and students in the learning environment were also made. At the end of the 20 week intervention, subjects were then post-tested on their problem solving skills.

The findings revealed that students who learned LOGO programming were able to demonstrate transfer of problem solving skills to a near-transfer context but not to a far-transfer context when compared to students who learned BASIC. Also, students who learned LOGO programming using a process-oriented approach demonstrated better transfer of problem solving skills to a near-transfer context with complicated problems than did students who learned LOGO programming using a content-oriented approach. Classroom observation during the intervention phase also showed that there were more substantive verbal and non-verbal interactions among

students who learned LOGO compared with students who learned BASIC. Also, students in the process-oriented group were involved in more classroom interactions than students in the content-oriented group.

The main conclusion from this study is that LOGO programming could be used to facilitate the development of problem solving skills among students. In particular, the process-oriented approach, which focuses on the processes of problem solving, could be used to assist students further in the development and transfer of problem solving skills. As well, LOGO programming could also facilitate more social interactions among the students, especially if the instructional method provides such an emphasis.

This dissertation is dedicated to

my parents and my wife, Ching Hung,

whose understanding and support provided me with

the encouragement and determination to complete this work.

Acknowledgments

This thesis has been accomplished through the assistance of a number of mentors, colleagues, and friends. They cannot be held accountable for its deficiencies, but if merit resides in its pages much is undoubtedly due to them.

Thanks must go to my chief supervisor, Dr Kenneth Ryba, who provided me with excellent professional supervision and inspiration throughout the duration of this research. I am especially grateful to Professor Ray Adams, who has continued to supervise my thesis even after his retirement, for his wise counsel and critical appraisal of this dissertation. Special thanks must go to Associate Professor Don McAlpine, whose support during Dr Ryba's sabbatical leave has been invaluable. Over the years, these three mentors have provided me with excellent models of critical and logical analysis, as well as meticulous scholarship, examples which I have striven vainly to emulate. I am particularly indebted to Associate Professor Lorna Chan of University of Newcastle, Australia, who has given me much appreciated advice and guidance regarding statistical analyses. Professor David Battersby of Charles Sturt University, Australia, has offered valuable critical and constructive comments in the final editing of this thesis. Thanks must also be offered to Dr. Kerry Chamberlain of Massey University, who assisted in the initial stages of this investigation.

In addition, a number of colleagues in Australia, Canada, Hong Kong, New Zealand and the United States responded with information and encouragement regarding this study. I extend my gratitude for their efforts, interests and support to the following: Dr Patricia Babbs, Dr John Borkowski, Dr John Burton, Dr James Chapman, Dr Douglas Clements, Dr Geoff Cumming, Dr Diane Cuneo, Ms Teresa Doyle, Dr Catherine Emihovich, Dr Greta Fein, Dr Joan Gallini, Dr Henry Gorman Jr, Dr Mark Grabe, Dr Kay Irwin, Dr Judith Kull, Dr Kwok Wing Lai, Dr David Lancy, Dr John Leung, Dr Marcia Linn, Dr Allan McAllister, Dr Anne McDougall, Dr Annemarie Palincsar, Professor Seymour Papert, Dr Robert Seidman, Mr Chris Watson, Dr Peter Williamson, Dr Sylvia Weir, Ms Sandra Wills, and Dr Susan Zelman.

I am extremely grateful to the principal, Mr. Richard Bullock, and the staff of the Hokowhitu Primary School for their assistance and support during the course of this investigation. My sincere thanks must also go to the students who participated in this research, and to their parents who provided so much support during this study.

This research would not have been possible without support and funding from a number of organizations, and appreciation must be extended to them. Firstly, IBM (NZ) Corporation supplied all the computing equipment and software for this study as well as funding for the employment of the necessary personnel. Special thanks must go to Ms Pamela Yates, the Education and Marketing Manager of IBM (NZ), and the technical staff who were always there to overcome various crises during this study. Secondly, the Education Department of Massey University provided the funds for the purchase of necessary testing instruments. Thirdly, Brightway Ltd donated some of the testing equipment for this study.

I am indebted to Bill Anderson, Ron Henderson, and Jane Horton, who helped to teach the children in this study, and provided constructive suggestions about the design of the teaching modules used in this investigation. A number of graduate students at Massey University helped to conduct the testing of students with much cheerful competency. I am beholden to: Teresa Ball, Karolle Galtema, Ron Henderson, Bev Hong, and Lois Wilkinson.

Appreciation is also due to Associate Professor Phil Moore, who has allowed me time to complete writing this thesis at the University of Newcastle in Australia, as well as for his support and friendship.

Finally, to my parents and my wife are due my acknowledgment and gratitude for the emotional support, encouragement, and understanding through the inevitable stresses associated with this doctoral investigation.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION AND OVERVIEW

Modern society is undergoing many profound technological changes. Among these changes are the invention and rapid development of micro-computers and associated technologies. The world of computers has expanded markedly since the invention of the microchip. Computers are now used for such diverse tasks as banking, guiding rockets, controlling assembly line robots and even home computer games.

While computers have been used with such diversity and versatility, it is only recently that decreasing production costs and increasing capabilities have meant that computers have become economically feasible in schools and widely available for education.

With the increasing dependence of society on computer technology, the need to prepare students for life in an information-based society has made literacy in computer technology a necessity. The workforce is changing owing to the introduction of computer technologies and schools are being urged to educate for this change.

Furthermore, it has been suggested that computers will play an increasingly important role in human learning (eg Taylor, 1980; Papert, 1980; D'Ignazio, 1991; Thornburg, 1991). This prediction has, to a certain extent, been supported by the ever increasing number of computers used in most educational institutions ranging from primary to tertiary and in all disciplines. However, many important questions need to be answered concerning the application of computers in education before educational administrators can decide how microcomputers should best be integrated into the school system. One of the central questions is "What effects do computers have on the learner and the learning process?"

Over the last ten to fifteen years, a great deal of attention has been directed at the application of computers in education and their effects on students and the learning process. In many countries, notably the United States of America, Canada, the United Kingdom, New Zealand, and Australia, governments have developed comprehensive policies and support programmes for introducing and integrating computers into the school curriculum. For instance, in New Zealand, the Department of Education set up various support agencies such as the Computer Courseware Development Unit and later the Computer Education Development Unit to assist schools to integrate computers into the school curriculum although more recent reforms in the late 1980's have prompted the demise of this unit. Most pre-service teacher training courses in New Zealand have included educational computing as a core component.

Whether the introduction and integration of computers into the schools have substantially modified education practices in any fundamental sense is open to question. What is not in doubt is that the vastly increased use of computers for both formal and informal education has begun to provide access to a new and hitherto unavailable mass of experience and data which could enrich educational thought and practice. During the 1980's, one of the important research issues in education has been the examination of the relationship between the use of computer and the development of problem solving skills.

It has been asserted that one of the primary missions of educational institutions is to impart knowledge and to teach cognitive skills (eg Dewey, 1933; Kozmetsky, 1980; Chipman & Segal, 1985; Rowe, 1988a; Yates & Moursand, 1988). While one of the most important cognitive skills is the ability to solve problems, particularly in mathematics, reading and other substantive domains (cf Frederiksen, 1984), perhaps more important, is the development of some general problem solving skills such as planning, analysis, monitoring and evaluation, that can be applied in more than one context.

The importance of the development of problem solving skills in modern education is reflected in the increasing emphasis on the incorporation of elements of problem solving in the school curricula in many countries (Kuhn, 1990). For example, the National Council of Teachers of Mathematics in the United States

argues that problem solving must be the focus of school mathematics in the 1980's (Dolan & Williamson, 1983). In Australia, the curriculum documents of various disciplines such as mathematics, history, and science, have been extensively rewritten during the 1980's so that their aims and objectives would reflect the importance of problem solving in the teaching of these disciplines in schools (eg New South Wales Department of Education, 1986). Apart from the incorporation of a problem solving approach into various subject areas, there are also many independent courses on problem solving offered at the primary, secondary, and tertiary levels in countries such as Australia, Britain, Canada, New Zealand etc., which are dedicated to the teaching of general problem solving skills, independent of the traditional discipline demarcation (cf de Bono, 1985; Chance, 1986; Fisher, 1987).

This concern over the development of problem solving skills in education has also been having a significant influence in educational computing. Traditionally, computers have been used in schools either as an object of instruction, or as an instructional tool based upon the use of programmed instruction and mastery learning. The former type of usage emphasizes the learning of the operations and structures of computers, and embraces different types of school courses such as computer awareness and computer literacy. The latter type of usage, often labelled as computer assisted instruction (CAI), has its focus on the teaching of various curricular content with the computers acting as an instructional tool. However, recently, there has been a significant shift in emphasis within the applications of computers in education. Increasingly, educators have been advocating the need for students to use computers as a tool in their learning, with applications such as word processing, database management, spreadsheet, music, adventure games, telecommunication, graphics packages, and multimedia. In particular, educators have been concerned with how computers could be used to assist the development of problem solving skills of students by focusing on the processes of learning and thinking themselves.

One major impetus for this shift has come from the work of Professor Seymour Papert and his colleagues at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. They have focussed on the use of the

computer programming language LOGO. In his seminal work, "Mindstorms: children, computers, and powerful ideas", Papert (1980) has strongly criticized the traditional applications of computers in education, particularly computer assisted instruction. He argues that this usage, in effect, allows the computer to control the learners, and as a result, stifles the learners' ability to develop their thinking. Instead, Papert suggests that computers should be used in the construction of a learning environment within which children would be able to develop a sense of control over their learning, and at the same time, develop some general problem solving skills. He contends that the use of the LOGO language could be used to create such an environment.

Such arguments, together with other developments of microcomputer technologies which have made computers much more versatile and user friendly, have prompted many educators to re-examine the role of computers in education with the common concern of using computers to assist students to develop their intellectual capabilities. For example, in 1987, the then Minister of Education of New Zealand, the Hon Russell Marshall (1987) commented,

> *The role of the computer as an empowering tool for learners in all disciplines, at all levels of the education system, as an extender of the physical and intellectual capabilities of the user, as a 'machine to think with', seems to be coming to the fore. (p. 63)*

This reflects on the current debates within educational computing which focus on the use of computers in the facilitation of development of general problem solving skills among students (cf Au, Horton, & Ryba, 1987; Au, 1992b). Recently, a number of researchers have found that some computer applications can be used to enhance students' general problem solving skills (cf King, 1989). Against this, other studies have found contradicting results (cf Roblyer, Castine & King, 1988). Consequently, there has been an increasing call for more research to be conducted to examine the development of problem solving skills when students learn how to use computers (eg Yates & Moursand, 1988), and in particular, computer programming (eg Au, Horton & Ryba, 1987; Brady, O'Donoghue & Bajpai, 1989).

Behind this debate lies two related arguments. First, problem solving is essential to the construction of computer programs (Grogono & Nelson, 1982); second, the semantics of problem solving can be learnt in conjunction with the syntax of a programming language (Dyck & Mayer, 1989). Based on these two arguments, it has been suggested that it is possible that a learner's skills in problem solving might be improved through the learning of computer programming.

These assertions, together with the fact that microcomputers have been implemented in New Zealand schools at an increasing pace during the last decade, have emphasised the need for methodical investigations into the different ways and effects of using computers in New Zealand classrooms. However, much remains to be answered concerning the integration of microcomputers into the existing school curriculum or the possible cognitive effects that certain computer applications might have on the learner.

Among the various related issues yet to be addressed is the role of teachers and the possible changes in the nature of their work in a computer environment. This issue is directly related to the implementation of the type of learning environment that might be conducive to the development of problem solving skills of the learners. Inquiries into this area would allow educational practice to be in a position to respond to whatever opportunities are presented by the new technology (Conabere & Anderson, 1985).

The most prominent manifestation of this issue is in the debate concerning the use of LOGO. On one hand, it has been suggested (very much in accord with Piagetian educational philosophy), that because children learn best by free exploration, they should be allowed to explore their own learning without undue intervention from teachers while learning to program with LOGO. On the other hand, it has been advanced by many theorists, such as the metacognitive psychologists (eg Flavell, 1976; Brown, 1982), that some form of support structures in teacher intervention and explicit instructions in problem solving need be in place so that children's cognitive development would be optimized. This has led some computing educators calling for a more structured learning environment for the learners. So far, this debate has not been resolved.

Purpose of the study

The present study represents an attempt to address some of these concerns. The purpose of this study is threefold: first, to examine some of the effects that the learning of programming might have on the development of the problem solving skills of learners; second, to evaluate the effects of different instructional methods within a programming environment; and third, to examine the kinds of social interactions within a programming-learning environment.

Two programming languages were chosen for this study: LOGO and BASIC. Both languages were purportedly easy to learn even by relatively young children, and are widely used in both primary and secondary schools. LOGO was chosen because it has been suggested by many researchers that it could be used to facilitate the development of problem solving skills among students (cf Papert, 1980; Clements & Gullo, 1984). BASIC was chosen because its availability with most microcomputers in schools and that it was widely used in both primary and secondary schools.

Two different instructional methods in relation to the teaching of computer programming were examined in this study: a process-oriented approach and a content-oriented approach. Traditionally, programming has been taught with a content-oriented approach with the emphasis on the syntaxes of a programming language only (Au, Horton & Ryba, 1987). Recent research in metacognition suggests that a learner's problem solving ability might be enhanced when adopting a more process-oriented approach which provides instructions on the executive processes used in solving problems (cf Campione, Brown & Ferrara, 1983; Baker & Brown, 1984). The use of both approaches in the present study provided opportunity for comparison.

It is the case that current research into the effects of learning of programming seems to focus predominantly on the end product of cognitive development. It has been argued that as a result, what happened during the learning of programming has not been properly investigated. This argument has led to the criticism that current research is overly "technocentric" (Salomon & Gardner, 1986; Papert, 1987). It is also the case that much current research on the social context of learning programming is mainly of an anecdotal nature. Consequently, this study also

attempted to examine systematically the different kinds of social interactions within a programming-learning environment. In particular, two different types of interactions were observed: (i) the interactions of individual students with the teacher; and (ii) interactions among students.

It was anticipated that examining (i) the processes of learning and interaction, and (ii) the potential changes in the problem solving skills of the learners, would allow the present project to obtain more conclusive results by triangulating the data acquired.

The report of this study is presented in the following way. Chapter 1 provides an introduction and overview. Chapter Two reviews the use of computers in education. Chapter Three explains the LOGO programming language, its characteristics and theoretical foundation, and examines some of the claims made about the language. Chapter Four considers research with LOGO and its implications. Chapter Five examines the relationship between problem solving, metacognition and LOGO programming. Chapter Six describes the data collection and analysis procedures used in this study in some detail. Chapter Seven reports on the findings resulting from data analysis. Chapter Eight evaluates the results of the study, makes an assessment of the general relevance of the work accomplished and, as well, ventures into the realm of conjecture and prediction.

CHAPTER TWO

REVIEW OF COMPUTER APPLICATIONS IN EDUCATION

This chapter reviews the general applications of computers in education. The first section outlines the traditional ways that computers have been used in schools, and includes a discussion of the advantages and limitations of using computers as instructional tools. This is followed by a review of the more recent applications of computers in educational settings, and how computers should be used as learning tools for students. The last section of the chapter then considers how computer programming might be used to accomplish best results, in particular, as an extension of the learner's intelligence, or as a device to aid the thinking process.

Overview

The history of computers is relatively short but even so their impact on society has been enormous. Nowadays, almost every facet of life has been influenced by this technology. For instance, twenty years ago, no one could have predicted that computers would have become so common place in the world of education. Most secondary schools and many primary schools in New Zealand and other parts of the world are now using computers in their teaching and administration.

Nonetheless, there has been some resistance - fuelled by, for instance, the conservative nature of educational institutions, the fears of teachers, the lack of funds, and the lack of success of past educational innovations such as programmed instruction, television etc. (Miles & Huberman, 1984; Saloman & Gardner, 1986; Thornburg, 1991).

To obtain a better understanding of the impact of computer technology on education, it is useful to review briefly the history of the use of computers in education, especially in schools.

Historical Review

From the earliest times when computers were commercially available, there were educational applications of computers. The earliest documented use of computers in education refers to the use of mainframe computers in the late 1950's (Oliver, 1986). In those early days, a large number of mainframe computers were to be found in some engineering and research departments of universities and tertiary institutions and they were used mainly for research and development programmes. By the mid-1960's, most colleges and universities and many larger schools used computer systems for administrative purposes (Bramble, Mason & Berg, 1985).

Learning about computers

During those years, one of the initial applications was to use educational computing as *an object of instruction*. Many courses were conducted so that learners could learn about the computers.

One aspect stressed the explanation of computer structures and related electronic circuits, or in other words, the machine-level operation of the computer. In essence, the major aim was to promote learning about computer hardware.

Another aspect emphasised software. This became popular after an important development in 1963 at Dartmouth College in the USA when a high-level computing language for "teaching" a computer to do specific tasks was invented (Dennis & Kanksy, 1984). The language was called BASIC (for Beginners All-purpose Symbolic Instruction Code). The word beginners in the name of this language was significant, since one of its alleged merits was the ease with which it could be learned. This feature has caused BASIC to become overwhelmingly the most common language used at the pre-tertiary level in the teaching of programming. Since then, BASIC has become the resident language of most microcomputers as it was deemed to be particularly suited for beginners. Consequently, BASIC programming has often become part of the computer literacy programs offered in schools.

Many other programming languages have been taught since then. They include FORTRAN, PASCAL, and PROLOG. These languages were initially taught

at the tertiary level as they were only available on the mainframe computers. However, since the rapid development of microcomputer technology in the 1970's and in particular, the 1980's, they have often been part of the computer literacy and computer studies courses at the secondary as well as the tertiary levels.

Learning from the computers

During the 1960's, one particular group of educators tried to integrate computers into the existing school curriculum in addition to the computer awareness courses. Their major aim was to program computers so that learners could learn from the computers, in the same way that learners learn from teachers.

When computers were first introduced into the educational setting for general teaching and learning purposes, programs were based mainly on behavioural theories. In particular, they were often modelled on "programmed instruction" (Taylor, 1980; Coburn, Kelman, Roberts, Snyder, Watt & Weiner, 1982; Maddison, 1983). This usage of computers has often been labelled Computer Assisted Instruction (CAI), or Computer Assisted Learning (CAL), probably because the ancillary tasks performed were similar to those that could be performed by (ideally) competent teaching assistants.

The focus of this strand of educational computing was on the teaching of various kinds of curriculum content with computers acting as tutors to the learners (Taylor, 1980). With this type of usage, the program design often entailed a rigorous logical analysis of the subject matter, combined with a careful study of possible learning strategies. The results were then implemented through structured systems of programmed instruction that i) provided immediate feedback to the learners, ii) branched to the appropriate next lesson, and iii) kept meticulous records of student progress.

Back in the late 1950's and 1960's, significant potential was seen in CAI . As a result, many projects, for instance, PLATO and TICCIT, were undertaken at a number of tertiary institutions using mainframe computers to examine how computers might be used to enhance instruction (cf Bramble, Mason, & Berg, 1985; Oliver, 1986).

A number of benefits similar to those claimed for programmed instruction, were often ascribed to good CAI software. For example:

1. One-to-one interaction - computers could provide individual attention to the learners in a way not possible in traditional classrooms. It was often argued that one-to-one instruction provided by CAI, would result in (i) superior learning compared with the conventional classroom where students shared the teacher, and (ii) better interactive learning (compared with normally higher teacher-student ratios) (Telfer & Probert, 1986; Lockard, Abrams & Many, 1990).
2. Immediate feedback - as a result of one-to-one interaction, the computer was able to provide immediate feedback to the learner and at a rate more frequent and desirable than was possible in a traditional classroom. Moreover, feedback could be tailored to reinforce learning. However, it should be noted that some researchers had suggested that immediate feedback might not be desirable for achieving optimal retention of learning (Rankin & Trapper, 1978).
3. Small logical units - as in programmed instruction, tasks could be broken down into small units hence providing opportunities for the learners to assimilate the sequences of a complex task in a hierarchical, logical step-by-step manner (Soulier, 1988).
4. Individual rate of learning - students were able to work at their own speed and difficulty levels. Often the rate of learning could be either controlled by the learners or adjusted by computer program based on the responses of the learners (Hofmeister, 1984; Mandell & Mandell, 1989).
5. Overlearning - the computer could be programmed to drill students in facts and lessons could be repeated until mastery learning was attained. Even after mastery, learning could be repeated so that overlearning which could lead to better retention over time, could be accomplished (Ryba, 1980). The unlimited patience of the computers was ideally suited for this purpose (Kinzer, Sherwood & Bransford, 1986) - particularly for students with learning difficulties (Au & Bruce, 1990).

6. Freeing up teachers' time - while the computers were doing most of the time-consuming tasks required of teachers, the teachers would then be able to devote their attention and time to other aspects of their teaching duties presumably to the benefit of the students.
7. Comprehensive student records - CAI could provide the means for keeping progressive records of student progress. A full profile might be kept of the student's strengths and weaknesses, allowing for individual remediation (or extension) programmes to be devised. The computer might also record the time taken by students to answer each question and/or complete each segment, thus helping to identify the areas that hinder student progress. Such an accurate picture of the progress of pupils through various instructional materials could help educators to formulate new curricula or remedial programmes (Hofmeister, 1984; Bitter & Camuse, 1988).

However, owing to the then limitations of mainframe computers such as their enormous cost, small memory and slow processing speed, the exposure of CAI within schools was rather limited. Indeed, after having examined closely the early effects of CAI by reviewing over 20 studies in computer-based methods, Jamison, Suppes and Wells (1974) indicated that cost-effectiveness ratios made CAI less advantageous than traditional instruction. They also concluded from the review that no widely applicable statement could be made about CAI's effectiveness since effects on learning seemed to vary with student level and instructional mode. Similarly, in a study of factors that prevented more pervasive utilization of CAI, Anastasio and Morgan (1972) found that the lack of evidence of CAI effectiveness to be among the most critical. Additional years of research efforts since then however have failed to resolve the dilemma (Avner, Moore & Smith, 1980).

With the advent of microcomputers in the late 1970's, and their decreasing cost since then, most schools were able to purchase computers. CAI was thus "re-discovered" by teachers in both primary and secondary schools.

Moreover, the further and continual development of the microcomputer, its memory capacity, versatility and processing ability, as well as further advancement of peripheral hardware such as the graphic tablet, the mouse, the voice synthesizer,

the touch sensitive screen, highly realistic graphics, and interfacing with video tapes and disks, served to overcome some of the shortcomings that were apparent when CAI was confined to mainframe use. Such developments have helped CAI to become powerful systems of instruction with the result that interest in CAI has been reawakened (Tobias, 1985).

CAI thus became increasingly popular in schools. It has been estimated that 90 percent of the schools in the U.S.A. now use computers in instruction and the use of CAI predominates (Niemiec & Walberg, 1987).

Some of the most common forms of CAI include: drill and practice, tutorial, simulation and dialogue (Coburn *et al*, 1982; Niemiec & Walberg, 1987). Drill and practice perhaps represents the most primitive form of CAI. Put simply, this is where the computer is used to take learners through a series of exercises. The learners can practise by responding to questions posed by the computer with the computer then analysing the answers and providing appropriate feedback to the learners. Drill and practice have developed over time into other modes such as tutorial, simulation and dialogue (Probert, 1985; Vockell & Schwartz, 1988). Tutorial software uses explanations, descriptions, illustrations and problems for concept development. Questioning and prompts are used widely in much the same way as a tutor would use them to help students gain an understanding of the subject material. Simulation CAI software is used to simulate real or imaginary situations, usually inviting students to make decisions based on information given. Many of the simulation software also incorporate games format to enhance motivation. Dialogue software are designed in such a way that it can be used to conduct the dialogue between the computer and the user. Many of the commonly available software consist of a mixture of these modes of computer assisted instruction.

In these modes, the computers have been employed as instructional systems, intended to facilitate the attainment of curriculum goals - a process not too different in principle from traditional classroom teaching. However, when properly deployed, it is far more flexible than any book- or material-based programmed instruction. For one thing, the material can be presented interactively and dynamic graphics and other sophisticated teaching aids such as video tapes and video discs can be integrated with the computers. For another, within CAI, student performance

histories can be collected, stored, and subsequently used for evaluating the materials and as a basis for routing a student through the materials. Moreover, CAI can be designed to move the student at a variety of speeds and be interrupted more or less at the student's convenience.

Although huge resources were invested in the development of instructional software, research findings in the effectiveness of CAI in the classroom have been inconclusive (c.f. Kulik, Kulik & Bangert-Downes, 1985; Kulik & Kulik, 1987; Roblyer, Castine & King, 1988). Given the relative high cost of implementation, CAI failed to make significant impact within the educational scene (Coburn *et al*, 1982; Maddison, 1983). Numerous criticisms have been levelled at the CAI approach and its limitations, such as, the lack of high quality software, the high cost of producing suitable software, and most important of all, because the machine controls the learner, optimal learning does not take place.

As a result of these criticisms and with the rapid development of microcomputer technology over the past decade, there has been a significant shift within educational computing - a shift in focus from learning from the computers to learning with the computers. This reflects a wider and deeper pedagogical concern - "The focus of attention has shifted from teaching to learning" (Renwick, 1985:3).

It has long been a matter of contention over whether students learn better when they are taught with traditional methods that emphasise teaching or whether they learn better when they are free to pursue and control their own learning. The shift in the role of computers in education is a response to that debate. It is also a response to another debate over whether focus should be placed on the process of learning as well as on the content. It has often been argued that by placing the emphasis on the process of learning, it would be easier for transfer of learning to occur in other contexts (Papert, 1980; Nolan & Ryba, 1984).

Both issues are important in the context of the present study and will be discussed in later sections. However, in order to have a better understanding of why such a shift has occurred, it is important to examine and to evaluate some of criticisms levelled at computer assisted instruction and hence its educational value either as an adjunct to teaching or as a stand-alone system of instruction. There have been four main criticisms, they are: CAI duhumanizes the educative process,

poor quality of CAI software, high cost of implementing CAI in schools, and unsound educational values. Discussion of these criticisms follows:

1. CAI dehumanizes the educative process - CAI, as a form of individualised instruction, isolates students from human interaction. The presumption here is that students will communicate with computers only but not with the teacher or the other students thus eventually making the classroom and the teacher redundant.

   This view seems to stem from a lack of understanding of the applications of computers in education rather than empirical evidence and practical considerations. Research evidence to date does not suggest that CAI, and indeed the use of computers in education in general would alienate the students from social and human interaction (cf Hawkins, 1983). Given the benefits of CAI such as individual attention, unlimited patience and detailed records of students' progress, the classroom teacher would then be able to devote more time to the personal human considerations and hence (theoretically) would facilitate learning (Liao, 1992).

   Moreover, the extent of social interaction when using CAI depends very much on what and how CAI programs are used in the classroom. First, the fact that the screens of the computers tend to make student work more public, tends to support the argument that CAI would promote more social interaction rather than reduce it. Second, as in any traditional classrooms, student/student and student/teacher interaction depends very much upon the extent in which teachers encourage and provide scope for it. Third, some CAI programs lend themselves to social interaction. For example, the use of some programs such as adventure games requires extensive cooperation and collaboration among students. With the wide range of CAI software available to date, teachers could have more control over the materials used in the classroom. It is unlikely, therefore, that CAI would dehumanize the educative process.

2. Poor quality CAI software - the design of commercial software often lacks the integration of appropriate educational theories, sound instructional sequences, and good curriculum design. Software is usually written by people with programming expertise only and although sound in a programming sense, often do not meet the criteria of good instructional design.

This is largely due to the oversimplification of the very costly and demanding processes involved in producing high quality software. One outcome is that software may not be user-friendly or well documented, and often leads to teacher and student confusion. The educational values of such programs have been questioned and doubted.

However, an increasing number of high quality CAI software has appeared on the market - for a number of reasons: first, educators have started to acquire programming expertise, hence enabling them to produce good educational software; second, initiatives from government agencies have aided coordination and production of good quality software; third, cooperation has occurred increasingly between tertiary institutions and private sectors (Au & Cook, 1989). It could perhaps also be added that whatever, the pedagogical weakness of computer software, not all human teachers are always pedagogically impeccable.

3. High cost of implementing CAI in schools - apart from the expensive initial capital outlay in setting up computers in the schools for teaching and learning purposes, the cost of developing and purchasing educationally worthwhile CAI programs is also considerable.

Ideally, effective CAI programs should be produced by a team of people consisting of at least educators, curriculum developers, instructional designers, and programmers. Given the amount of time required, the cost of producing effective CAI software may be prohibitive. The facts that commercial publishers seek profit margins and that the market is fickle (and pirating often occurred) make the production of high quality educational software a risky business proposition.

The problem of cost is further complicated by machine incompatibility. Software produced for one particular type of computer often cannot be transported to another type of computer. Such differences between brands mean that either a program is restricted to a small market, or it must be translated into a version suited to another machine - not a minor task given that many programs nowadays rely heavily on machine specific routines which are peculiar to the brand.

This lack of compatibility among computers also prevents ideas from being shared among different teams of experts working for different software publishers and/or computer firms. Hence ideas can become isolated, and dissociate from the network of educational professionals (Horton, 1986).

Some of these cost problems have now been partially overcome by, for example: i) availability of government funds to purchase both hardware and software (eg in Australia, Canada, and USA etc.); ii) commercial software publishers making available educational packages at greatly reduced prices, or in the form of labpacks and site licences; and iii) hardware producers selling computers to schools at prices much lower than those in the retail market. It is also the case that prices of hardware have dropped steadily as development research costs have been recovered, and economies of scale have occurred.

4. Unsound educational values - there are two aspects to this criticism, i) that knowledge is not always reducible to facts that may be programmed into computers; ii) that CAI software controls the learners.

i) It has often been argued that teachers are able to draw upon the cumulative experiences of their careers, both as students and teachers, they thus can apply 'lessons' learned in the past to similar situations more recently encountered. Any programs that specify the course of the lesson in advance by definition prevents advantage being taken of situation specific past experience.

Moreover, there is a possibility that the authors of CAI programs will not take into account other desirable outcomes of learning such as attitudinal

change in evaluating responses. For example, five educational objectives have been postulated in the affective domain. They are: receiving; responding; valuing; organization of values; and characterization by a value or a value complex (cf Clements, 1981; Lawton & Gerschner, 1982). Receiving and responding are built into the CAI system but the last three objectives are more likely to be absent from most CAI packages. Computers cannot mediate values other than those provided by the software, i.e. those that the author has conveyed. Because the value complex of one student will likely be dissimilar to that of another, thus it follows that the software is liable to fail in any attempt to provide education with an affective component.

A similar argument can be mounted with respect to the cognitive domain. Bloom (1956) postulated six desirable cognitive outcomes. They are, in increasing order of complexity: knowledge; communication; application; analysis; synthesis; and evaluation. The critics hold that CAI cannot meet all of these objectives. The easier learning behaviours (the first three objectives) may be programmable but the really important outcomes of learning could be under-emphasized or even ignored.

Within the cognitive domain, it has often been argued that CAI software is pedagogically restrictive (Coburn *et al*, 1982) in that it tends to: channel students into a narrow range of possible responses that: (i) keep them from exploring the complexities of concepts; (ii) trivialize important concepts; (iii) sometimes reinforce incorrect learning; and (iv) be unnecessarily boring - all because of the difficulty in anticipating all the possible responses from different users.

With the recent development of Intelligent Computer Assisted Instruction (ICAI) (or alternatively labelled Intelligent Tutoring System (ITS)), some of these concerns have been addressed. ICAI utilizes concepts and principles of artificial intelligence and knowledge engineering in the construction of CAI software (cf Clancey, 1988; Self, 1988). No longer are students required to respond to pre-programmed answers. Rather, ICAI, by incorporating expert performance and sets of inference rules, can (i) present materials and use instructional strategies according to student responses; (ii)

draw inferences on subject matter; (iii) deduce a learner's approximation of knowledge; and (iv) reduce the difference between the system and student performance. These features allow ICAI to provide more effective educational experience for students (Burns & Capps, 1988).

ii) CAI programs control the learners - when using CAI programs, learners always work within a pre-defined environment set up by the authors of the software. Accordingly, the learners are controlled by the computer rather than controlling it.

Protagonists of computer education, Luehrmann and Papert share with the philosophical position of Dewey and Piaget, that it is important for students to learn to control the learning environment, and hence, they would add, computers. They also consider that optimal learning takes place when learners are able to explore freely and discover actively their own learning (cf Luehrmann, 1981; Papert, 1980).

Papert's work, although centred on computer programming, emphasizes strongly the importance of student control over computers. In Papert's vision of a computer based learning system, a child should program the computer instead of being programmed by it (Papert, 1980). He advocates that by doing so, students will both acquire a sense of mastery over a piece of powerful modern technology and be able to reflect intelligently on their own processes of learning (Papert, 1980).

Research by Chapman & Ryba (1983) adds another dimension to the issue of learner control over the computer. They suggest that it is rather internal feelings of control (in contrast to external control) which promote cognitive benefits. Enhanced skills in problem solving and affective benefits such as self-management of behaviour and learning result from active learner participation. The importance of promoting the internal locus of control among students is further underlined by attribution theory (Weiner, 1974, 1979).

The significance of such research (Papert, 1980, Luehrmann, 1981; Chapman & Ryba, 1983; Bork, 1987) is that it provides a rationalization for

the current trend away from environments where students are controlled by computer, as exemplified in most of the traditional CAI software, to a situation where students can control and hence achieve a sense of mastery over the computer. They thus came to feel they have control over their learning environment.

Some of the more recent CAI software seem to have addressed this issue of control, especially those that have incorporated the principles of artificial intelligence (Smyrk, 1991) and multimedia (Gray & Bell, 1991; Sherwood, 1991). These software on one hand, tend to be highly interactive with the learners as a result of advanced microcomputer technology such as interactive videodisc, compact disc read only memory (CD-ROM), compact disc interactive (CDI), scanner, speech synthesizer, local area network and far network, hypertext etc. (Howard, Busch, & Watson, 1992); on the other hand, they also tend to place more control in the hands of the learners by (i) making the software more user friendly; (ii) allowing the users greater degree of freedom in choosing their own paths of learning; and (iii) providing more meaningful feedback as the abilities of the software to analyse and evaluate students' responses increase.

Learning with computers

Accompanying the advent and rapid development of microcomputer technology has been the production of highly user-friendly software that tends to place more emphasis on learner control the learning environment. Some software have been specifically developed to achieve this purpose, others are of a more general nature, e.g. word processor, database management programs, spreadsheet, graphics programs, telecommunication programs, and adventure games etc..

When equipped with such capabilities, the computer then becomes a tool for students to learn with (Taylor, 1980) in contrast with learning about computers and learning from computers. In this new role, the computer is no longer restricted to specific subject matter, rather, it becomes a tool for students to explore and facilitate their own learning in an environment controlled by the students (Papert, 1980; Wills, 1984; Nolan & Ryba, 1984).

Four important issues lend support to the arguments that it is advantageous to use computers as a learning tool. First, the learners can exercise more control over their learning environment. Second, the application of new general purpose software is not restricted to a particular subject and hence should facilitate the integration of computers across the entire curriculum irrespective of subject areas. Third, computers can enhance learner access to information and learning. Fourth, computers can be used as objects by students to learn to think with, and even develop higher order thinking skills. These four issues will now be considered in more detail.

1. Control of the learning environment. Central to the notion of using the computer as a learning tool is the control of the learners over their learning environment. With the development of features of microcomputers such as their memory capacity, versatility and processing capabilities, it is now possible to employ microcomputers to create highly interactive learning environments which place the learners in control. This has been enhanced further by the incorporation of peripheral interactive hardware such as: the mouses, voice synthesizers, tough sensitive screens, CD-ROM drives, video players, scanners, and highly realistic graphics. These collectively permit a more personal and more friendly environment to be created for learners. As well, the development of word processors, database programs, spreadsheet, graphics programs, telecommunication packages and multimedia tools have also helped.

   Such software, when used together with the microcomputer and its advanced peripherals, allows learners versatility in exploring their learning environment.

2. Integration of computers in the existing school curriculum.

   When microcomputers were first introduced into the schools, they tended to be used mainly in subject areas such as mathematics and science. However, with the availability of user-friendly software, microcomputers are now being used in many other subjects. For instance, a word processor can be used in the teaching of language by providing students with a valuable tool

to assist with the process of writing, hence avoiding the chores of writing and re-writing drafts with pencil and paper (Broadley & Au, 1988). Most word processors allow the learners more control in the editing of their writing by providing facilities such as pull-down menus, function keys editing etc. Some word processing packages such as Fredwriter and Quill can also assist the teachers when designing prompted writing programmes for their students (Wharton, 1986; Dailhou, 1986). Most recent word processing packages even provide facilities such as spelling checkers, thesaurus, style checkers, and graphic capabilities. When using the word processor of a computer as a tool, students will have more time to devote to their thinking. Students in industrial arts, architectural design etc can similarly use software packages to assist them in experimenting with their ideas of perspective, relative proportion, third dimension etc.. Database programs can also be used in many subject areas eg history, geography, and social studies, to permit students to store, organize and retrieve information (Hunter, 1985). The essence of these programs is that they save an enormous amount of time that would otherwise be devoted to re-drafting, re-writing, re-designing etc.

3. Enhancement of learner access to information and learning

When such application software is combined with telecommunication packages, microcomputers can provide a very powerful learning environment for the students. Such a learning system transcends the confines of the traditional classroom and allows information access and communication on a much broader scale - regional, national, and even international (Leonard, 1991; Williams, 1991). Students can now access electronic databases and bulletin boards virtually anywhere in the world (Chandler, Gesthuizen, & Clement, 1992). Moreover, they can communicate with students of other schools via electronic mail. For example, students at the Turramurra High School in Sydney Australia have been exchanging cultural information with Eskimo students in the Arctic region of North America (Frederick, 1986). In New Zealand, students at the Maru Maru school have been exchanging information with their counterparts at Captain John Palliser School in

Calgary, Canada within subject areas such as computer studies, geography, social studies and English composition (Ryba & Mackrell, 1986).

When used in special education setting, microcomputers provide students with disabilities with much better access to learning (Au, & Bruce 1990; Williams, 1987; Wood, 1986). For instance, the use of voice synthesizers, photonic wands, braille, and modems etc. allow even severely physically handicapped students to participate in learning that was not possible otherwise.

4. Development of higher order thinking skills. Apart from providing students with the kind of versatile learning tools illustrated above, microcomputers as learning tools are considered to have the potential to be used to assist students to develop higher order thinking skills (cf Hunter, 1985; Ryba & Anderson, 1990; Au, 1992b).

For instance, database management programs, besides helping students to organize their information whatever the subject, can also engage students in learning how to test hypotheses (using the information available in the database), plan the construction of a database, evaluate the organization of database etc. Similarly, spreadsheet programs can also involve students in the development of models, and testing of hypotheses. Because such activities are integral to thinking, it should follow that students will become more intellectually skilful and even gain more knowledge about their own cognitive processes (Hunter, 1985).

The notion of developing higher order thinking skills reinforces the idea that computers could be used as a learning tool, not just for learning, but also to learn *to think with the computer* - much in the same way that some people think "through" their pens, their typewriters or their tape recorders. There have been indications that through learning programming (Clements & Gullo, 1984), or adventure games (Ryba & Anderson, 1987; Lai & Mace, 1989; Thornburg, 1991) etc., metacognitive skills and general problem solving skills might be enhanced (Baker & Brown, 1984). Such skills, it has been argued (Campione, Brown & Ferrara, 1983; Palincsar &

Brown, 1984; Paris & Winograd, 1990b), might then assist the transfer of problem solving skills from one domain to another.

Whether the educative potential of the microcomputer is more likely to be realized through using it as a learning tool rather than as an instructional tool is an empirical question waiting to be answered.

Summary

This chapter has provided a brief history of educational computing and has discussed the various major ways of how computers might be used in education. As well, the main merits and shortcomings of using the computer as an instructional tool have been critically examined. The different applications of the computer as a learning tool and its importance have also been highlighted.

Attention now turns to one specific way in which the computer might prove helpful in developing higher cognitive skills. The next chapter focuses on the use of the computer, specifically, the LOGO language, as a tool for thinking.

CHAPTER THREE

LOGO: CHARACTERISTICS, THEORETICAL FOUNDATION AND CLAIMS

This chapter examines the LOGO programming language. The first section provides a general overview of the language. This is followed by a review of the characteristics of the language and the theoretical foundation associated with it. The last section of the chapter considers some major claims made about the benefits of learning LOGO programming.

Overview

The LOGO computer language was first developed at the Bolt, Beranek and Newman Corporation by Papert, Feurzeig, Bobrow, Solomon and others (Feurzeig & Lukas, 1972), and was modelled upon research in artificial intelligence and the computer language LISP (Reed, 1982). Implementations of LOGO were further developed by Seymour Papert, Feurzeig and associates in the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology (MIT) over a twelve-year period beginning in the late 1960's (Feurzeig, Papert, Bloom, Grant, & Solomon, 1969), and by Howe and his colleagues at the University of Edinburgh (Adams, 1985).

LOGO programming has been one of the most popular ways of using microcomputers as a learning tool. Its popularity can be partly attributed to the graphic features of the language and the ease with which it can be learned, even by very young children. It can also be partly attributed to the claim that learning to program with LOGO can enhance intellectual functioning. For instance, one of the founders of LOGO, Seymour Papert (1980), claims that within a LOGO environment, learners can reflect upon their own thinking and take conscious control of the learning process by articulating and analysing their own behaviour.

In contrast to other popular programming languages used in educational settings such as BASIC and PASCAL, LOGO is a high-level educational computing language developed for research in artificial intelligence (Abelson, 1982a). Initially however, LOGO was designed as a tool for use by school-aged children to extend

their own learning (Papert, 1980; Lawler, 1982). The aim was that LOGO would (i) provide a natural and friendly environment accessible to children of all ages and abilities for an experimental approach to mathematical ideas and processes, and (ii) a context for the use of the general heuristic of analysis, planning and review (Feurzeig *et al*, 1969; Finlayson, 1983; Noss, 1987b).

LOGO's earlier focus was on its use in teaching mathematics (Feurzeig & Lukas, 1972), and thus involved mathematical concepts and skills (Clements, 1987b). It has been argued that the way a program is constructed in LOGO illuminates the mathematical processes used by the programmer and that the programming activity itself provides a powerful aid to "decentration" which enables learners to reflect on their own thinking processes (Finlayson, 1984, 1985). It was suggested that in LOGO programming, students learn mathematics by utilizing concepts that aid them in understanding and directing the movements of a robot (in the form of a mechanical turtle) (Feurzeig & Lukas, 1972; Papert, 1980; Battista & Clements, 1986). They are said to develop problem solving skills because they are learning to be mathematicians rather than learning about mathematics (Papert, 1972). Papert (1980) remarks that,

> *The computer-based Mathland I propose extends the kind of natural, Piagetian learning that accounts for children's learning a first language to learning mathematics... No particular computer activities are set aside as "learning mathematics". (p. 48)*

Papert further asserts that LOGO provides a bridge between the abstract world of mathematics and the concrete world of reality. Hughes and Macleod (1986), observed that by writing instructions to control the turtle, children were required to use mathematics in a context where they could see the purpose of what they were doing.

Early work on LOGO at the MIT and the University of Edinburgh produced reports that students who had learnt mathematics through LOGO showed improvements in performance (eg Lawler, 1980; Watt, 1979; Howe, Ross, Johnston, Plane & Inglis, 1981). Other studies with secondary school children (Hoyles & Noss, 1985) and primary (Maxwell, 1984; Hillel, 1985), tend to confirm improved mathematical communication among students and between students and teachers

when LOGO is used. In one of the most extensive studies on LOGO to date, it was observed that students were able to see mathematics not as an immutable activity whose rules are "engraved in stone", but as a dynamic activity with many view points (Carmichael, Burnett, Higginson, Moore & Pollard, 1985).

Since the publication of Papert's seminal work, "Mindstorms: Children, computers, and powerful ideas" (Papert, 1980), and the increasing availability of the full version of LOGO compatible with various makes of microcomputers, LOGO has been popular among the educational community, most notably in the United States (Billstein, 1983; Bull & Tipps, 1983-84; Becker, 1986, 1987; Khayrallah & Meiraker, 1987). It is now regarded as one of the most popular computer languages taught in schools at present, especially at the primary and the junior high levels (Hassett, 1984; Campbell, Fein, Scholnick, Schwartz & Frank, 1986; Maddux, 1985; O'Shea & Self, 1983). There are many reasons for this and they will be dealt in more detail later in this chapter.

There have been numerous claims and counter-claims about the social, affective, cognitive and metacognitive results result of learning to program with LOGO, ranging from the concrete and practical through to the abstract and theoretical (Au, Horton & Ryba, 1987; Au & Leung, 1988). These claims will be examined in detail in the later sections of this chapter. That examination, however, depends on an understanding of the background to these claims and the attendant debates. Accordingly, the characteristics and the theoretical foundation of the LOGO language, and some of the claims about the language will now receive attention.

LOGO - its characteristics and theoretical foundation

Early experimentation with LOGO made use of a mechanical turtle that held a felt pen which drew on a large sheet of paper as the turtle moved. Children learned to program the turtle to draw shapes by firstly pacing out the steps, then by commanding the turtle accordingly. In an attempt to help children further master and examine their own thinking through the use of such concrete objects, a computer language was created. In that computer language, the turtle is represented on the screen of the computer by a cybernetic turtle (either as a little triangle or an image

of a turtle). Children, in controlling the turtle were able to move it about the screen, using elementary commands such as FORWARD 50, RIGHT 90 etc., to draw various geometric patterns. As simple as the activity might initially seem, the LOGO language incorporates concepts of increasing complexity. For example, sets of elementary commands can be repeated any number of times to make increasingly complex patterns. As well, students can "teach" the turtle new words that are in effect computer programs (Figure 3.1), and variables can be substituted for fixed values, therefore providing the full features of a computing language. All these features make the language intellectually challenging for children to use (Au, 1986; Au, 1988a; Harper, 1989).

There are a number of features in the LOGO language which make the language particularly suited to be taught in schools (Harvey, 1982a; Au & Horton, 1987). Because these features may offer some support for the claims made by Papert that LOGO provides a basis for learning the processes of problem solving that in turn serve as bases for the learners to explore their own intellectual structures, they will be discussed below.

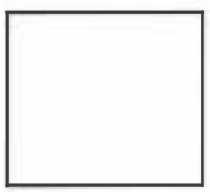
Characteristics of the LOGO language

1. LOGO has been described as a "friendly" language. It is friendly in that it "communicates" with the child in words that are non-threatening and easy to understand. For instance, when an error is made, instead of "syntax error" typical of other computer languages, LOGO will say, "I DON'T KNOW HOW TO TRIANGLE", or "NOT ENOUGH INPUTS TO FORWARD". The child in feeling that he/she has not made a mistake, will attempt to 'talk' to the computer and 'debug' the program. Russell (1983), and Weir, Russell and Valente (1982) argue, reasonably enough, that this non-punitive feeling can enhance the child's sense of control over the computer, and in turn develop a stronger sense of confidence and mastery over their learning environment.

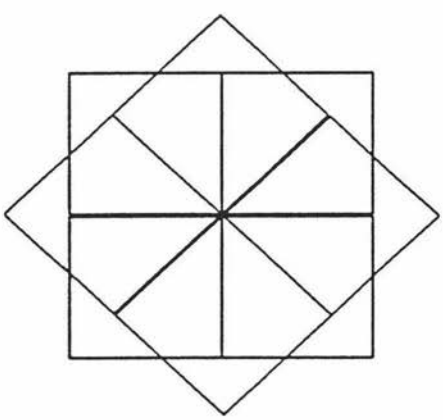
Figure 3.1.

Logo Turtle Graphics

```
TO SQUARE
REPEAT 4 [FORWARD 50 RIGHT 90]
END
```



```
TO STAR
REPEAT 8 [SQUARE RIGHT 45]
END
```



The STAR pattern is created by asking the turtle to REPEAT 8 times the comands to draw a SQUARE and turn RIGHT 45 degrees.

2. LOGO is a structured, procedural, extensible and recursive language. It is both structured and procedural in that LOGO programs, unlike programs of other languages such as BASIC and FORTRAN, consist of discrete procedures or blocks of procedures that the turtle responds to in a logical sequence. It is extensible in that once LOGO procedures are defined, they can be used as building blocks for more complicated LOGO procedures and programs (cf Figure 3.1, once the procedure SQUARE has been defined, it can be used as a building block for other procedures such as STAR). This approach can have a positive influence on children's problem solving behaviour in that it encourages children to be logical in their thinking, and it enables them to break down larger problems into smaller components and then use these part solutions as building of alternative structures - all important for the process of problem solving (cf Clements, 1985c). Moreover, LOGO is recursive, in that a procedure can be included in its own definition. This facility enables brief and elegant programs captivating the central structure of a problem to be used in complex structures, thus offering a very powerful problem solving tool (McDougall, 1983, 1988).

3. LOGO is interactive. LOGO allows commands to be typed and executed immediately, in contrast to some computer languages that require an intermediate phase of compilation. This feature is especially valuable in educational settings as even very young students can obtain immediate feedback and correct errors in the program as they occur. It also avoids the necessity of acquiring an extensive knowledge of programming before hand (Harvey, 1982a, 1982b; Carmichael *et al*, 1985).

4. The data structures of LOGO are lists. Lists consist of an ordered sequence of elements that may be numbers, words or other lists, and they are not typed like other programming languages. As a result, lists provide the means to create complex data structures and are much more flexible when dealing with variables in programming as contrast to other programming languages. As well, this flexibility renders LOGO suitable for a variety of usages that are

not necessarily algebraic or mathematical, and apply to other areas such as language and artificial intelligence (McDougall & Adams, 1982, 1983).

5. The availability of turtle graphics. This feature is unique to the LOGO language. "Turtle graphics microworld" provides an introduction to computer programming for younger children that is concrete, accessible and highly motivating. Some of the later versions of LOGO incorporate the facility of sprites, a facility that enables learners to create colourful, animated graphics displays, which in turn allows them to explore and develop ideas implicit in the interaction of time, distance, speed and velocity (Torgerson, 1985). More recently, an extension of the LOGO language, *LOGO (pronounced as STARLOGO), has become available on parallel computers. This version of LOGO allows users to simulate "artificial life" and real life by programming thousands of turtles as well as thousands of "patches" that make up the turtle environment (Resnick, 1990).

Theoretical foundation of the LOGO language

The characteristics of the LOGO language as discussed above arose from the theoretical foundation upon which the language was developed, and is well documented by Papert in his seminal work "Mindstorms: Children, Computers, and Powerful Ideas" (Papert, 1980). Basically, the foundation has very close associations with artificial intelligence and the Piagetian model of learning.

Many researchers see LOGO as being consistent with ideas employed in artificial intelligence (Bornet & Brady, 1974; Groen, 1984; Papert, 1980; Adams, 1985). For instance, Bornet and Brady (1974) suggest that (i) LOGO encourages the notion of a process as a representation of a solution to a problem, and (ii) the primitive commands of LOGO are simple to understand, being defined purely in terms of actions in the problem and not alterations to the state of the machine. Groen maintains that what a learner learns is not a programming language but a way of establishing correspondence between a concrete world and one of abstract representation (Groen, 1984). He argues that the strength of LOGO does not lie in

the generalization of programming skills, but instead in the ways of coordinating different representations of microworlds.

The notion of a microworld is central to the theoretical foundation of LOGO (Adams, 1986). Various definitions of a microworld exist however most of them are either unclear or have serious limitations. Goldenberg (1982), for instance, depicts a microworld as a well-defined, limited and interesting learning environment in which learners can acquire important ideas. Lawler (1982) pronounces that microworlds are essentially "task domains" or "problem spaces" with practical streamlined experiences designed for the learners. Piedmont (1983) describes a microworld as a computerized environment through which students can refine their thinking skills. Papert (1980) describes a microworld as:

> *A subset of reality or a constructed reality whose structure matches that of a given cognitive mechanism so as to provide an environment where the latter can operate effectively. The concept leads to the project of inventing microworlds so structured as to allow a human learner to exercise powerful ideas or intellectual skills. (p. 204)*

Although a specific definition of a computer microworld has yet to be agreed upon, there seems to be general agreement that a microworld should contain at least the following elements: (1) a dynamic and structured learning environment for exploration through which concepts can be explored; (2) a linkage to other microworlds whereby extension of learning can occur; and (3) a student project (cf Thompson & Chen Wang, 1988).

A typical example of a microworld within LOGO is turtle graphics. Within the turtle graphics microworld, learners can explore concepts of spatial relations within limited domain consisting of: a number of commands, the cybernetic turtle and the computer's screen. Conceptually, however, this microworld provides a link to a larger microworld that involves the organization and definition of paths through space (Campbell, Fein, & Scholnick, 1986).

The aim of LOGO, according to Solomon (1978), is to build in the computer a "culture" that provides a microworld for learners to (i) explore and (ii) enhance, through the writing and debugging of programming projects, the development of problem solving skills. This view is shared by Minsky and Papert (1972) who

suggest that through the active exploration of the microworlds provided by LOGO and the building of various computational models of processes, children will learn how to learn.

The notions of active exploration of, and construction of knowledge from the environment are strongly emphasized in the Piagetian model of learning, and there is a close link between the underlying assumption of the LOGO language and the Piagetian model of equilibration (Larivée, Parent, Dupré, & Michaud, 1988). This is not altogether surprising given the fact that Papert, one of the founders of LOGO, worked with Piaget in Geneva for some five years.

One central axiom of Piagetian thought is that a child learns through experience, developing a framework for dealing with the environment in relatively predictable stages. These stages are roughly and arguably linked to chronological age (Rousseau & Smith, 1981). Within each stage, a child must learn to assimilate and accommodate new information from the environment in order to modify his/her existing schema of knowledge and hence attain equilibrium. According to Piaget (1976), children can learn best if they are allowed to explore their environment by "acting" or "operating" on it.

The desirability of children exploring their own learning actively and freely is clearly underlined in Papert's exposition of LOGO's theoretical foundation. According to Papert (1980), in a LOGO environment, activities of children constitute the cornerstone for the construction of their intellectual structures as a result of the exchanges between the learners and the environment. Learners are able to set themselves goals and organize activities in order to attain the goals. The computer responds to the actions of the learner who, from then on, has to respond to the responses of the computer. The continual interaction and exchange between the computer and the learner - likely to be in the form of cognitive imbalance - would thus inform learners about their own cognitive processes, hence facilitating the attainment of cognitive equilibrium. Seen from this constructivist and interactionist perspective, programming thus becomes an ideal environment for learning (Larivée *et al*, 1988).

With the development, characteristics, and the theoretical foundation of LOGO examined, attention now turns to a review of three of the key claims: social,

affective and cognitive, which Papert makes about LOGO. It should be pointed out though, that much of argument supporting LOGO's value is conjectural and even theoretical in tone.

Claims about LOGO

Social Claims

Integral to Papert's theorising is the claim that children will interact with and support each other when using LOGO as a learning tool to develop thinking skills. Papert argues that LOGO can be used as a powerful learning tool because it provides a supportive social context for the learners to discuss and reflect on their learning (Papert, 1980). He remarks that although working at the computer is essentially a private occupation, it increases children's desire for interaction. Where the environment provides the opportunity, children will want to get together with others engaged in similar activities because they have a lot to talk about. Further, *"what they have to say to each other is not limited to talking about their products: LOGO is designed to make it easy to tell about the process of making them"* (Papert, 1980:180).

The claim that the use of LOGO in school contexts can provide a supportive learning environment has been studied by many researchers (Hawkins, 1983; Gorman, 1982; Jewson & Pea, 1982; Chiang, Thorpe, & Lubke, 1984; Clements & Nastasi, 1988; Hawkins, Homolsky, & Heide, 1984; Michayluk & Saklofske, 1985; Mitterer & Rose-Krasnor, 1986; Nastasi, Clements & Battista, 1990). They all observed that children seemed to collaborate and teach each other more when they were working with microcomputers than they did in other classroom work. For example, Hawkins (1983) noted that while children were working with LOGO, there were at least three types of peer engagement with computers - sustained collaboration on a joint project; seeking help or advice for a problem; and "pit-stopping", where children travelling around the classroom dropped in at a computer "for remarks". She argues that in all these forms, children provided support for each other in accomplishing their work.

The results of the research by Jewson and Pea (1982) indicate that children talk more to one another about works they are experiencing in the computer

situation, as opposed to other conventional work assignments in class. Similar observations have been made by Chiang *et al* (1984), Gorman (1982), and Hawkins *et al* (1984). Such studies seem to support the thesis that within a classroom where LOGO is used, there is much more interaction than in a traditional learning situation - interaction both with teachers and peers. It follows that there should be more opportunity for the learners to reflect on and discuss their thinking and learning. Recent studies by Mitterer and Rose-Krasnor (1986), Guntermann and Tovar (1987), and Clements and Nastasi (1988) tested the hypothesis that the use of LOGO in class facilitated peer-peer and learner-teacher interaction for social and cooperative problem solving and metacognitive processing.

A number of reasons have been advanced by Hawkins *et al* (1984) to explain this increased collaboration, viz.:

1. Features of the technology - The computer monitor is eye catching and makes the work of the learner public. As a result, anyone could easily see and become involved in what others were doing.
2. Features of the expertise available - As computers and LOGO were both relatively new to the classrooms and teachers (not necessarily confident experts themselves or even the only expert in the classroom), tended to encourage the children to help each other.
3. Features of the status of the work - As programming work was relatively new and has not been properly defined in the traditional curriculum, teachers were unsure about its assessment. Consequently the usual constraints on the appropriateness of collaboration did not apply, and group projects were just as acceptable as individual projects.

While it is obvious that such conditions are not necessarily confined to the use of LOGO, it is interesting to speculate, whether the use of LOGO actually provides a context in which children of various groups (eg ethnic, socioeconomic etc.) are able to overcome various cultural and socioeconomic barriers, to get together, discuss and reflect on their learning, and as a result, to develop "powerful ideas" as postulated by Papert (1980).

The importance for the learner to reflect on the learning process is central to the notion of the LOGO learning and bears significant consequence to the possibility

of transfer of learning to other contexts. They will be further examined in a later section of this chapter.

Affective Claims

The feeling of learners that they are in control of their own learning environment is said to contribute to learning (cf attribution theory) (Weiner, 1974, 1979). Papert (1980) contends that microcomputers using LOGO can provide a learning environment in which children of even preschool age are in control.

Various studies have been conducted to examine the possible affective changes within the learners when learning to program with LOGO. For example, Weir (1981), Russell (1983), Horner and Maddux (1985) and McDougall (1988), noticed significant changes in the affective component of children after they had learned to program with LOGO. Weir (1981) and McDougall (1988) both observed that the way in which LOGO placed initiative and control in the hands of the users allowed them to have direct effect on their environment. They attributed to this cause the stronger sense of confidence and control they exhibited over it.

Observations of a similar kind were made by Russell (1983) who worked with handicapped children. She noted that handicapped children using LOGO chose their own problems and solved them, communicated their ideas in a mode that was comfortable to them, and most important of all, approached and solved problems confidently. These observations led her to conclude that LOGO empowered children to be in control of their own learning and provided these children an environment of "manageable complexity" (Russell, 1983:39).

The study by Horner and Maddux (1985), although not finding that learning with LOGO would produce more internal locus of control, did find that LOGO was effective in making both learning-disabled and non-learning-disabled children feel responsible for their success with LOGO activities.

Cognitive Claims

Given a socially supportive environment and positive affective responses, Papert (1980) postulates that children in a LOGO context would then be able to explore freely and reflect on their own thinking. Children who learn to program

with LOGO would accordingly gain more insight into the different problem solving processes involved.

Papert made three noteworthy claims:

1. LOGO could provide opportunities for children to think about their own thinking. Papert (1980) contends that in explicitly teaching the computer to do something, learners learn more about their own thinking. This claim appears to be derived from artificial intelligence theory where constructing programs that model the complexities of human cognition is regarded as a way of understanding that behaviour (Pea & Kurland, 1984a). For example, it has been maintained by Papert (1980) that by deliberately learning to imitate the mechanical thinking of a computer, the learner becomes able to articulate what mechanical thinking is. Moreover, working with LOGO can provide a very concrete, down-to-earth model of a particular style of thinking which in turn facilitates the understanding that there is such a thing as a "style of thinking". The learner may accordingly have greater confidence about his/her ability to choose a cognitive style that suits solving a problem (Papert, 1980:27).

2. Learning to program with LOGO could help shift the boundary between the concrete and abstract 'thinking stages' as postulated by Piaget, since abstract knowledge can in a sense be "concretized". Papert contends that LOGO provides a concrete model for the children to think about and learn about learning. It follows that knowledge that was hitherto accessible only through abstract processes can now be approached concretely. As a result, children would understand abstract concepts more readily and rapidly. For example, concepts of angles and spatial relationships (which have been known to be difficult for younger children), can now be studied in a concrete manner as learners command the turtle on the screen in order to construct geometrical shapes. This position is supported by other theorists, eg, Lawler (1981) and Solomon (1982).

3. Some general problem solving skills can be learned when programming with LOGO and these skills might be transferred to other contexts. Papert (1980) argues that through programming, learners can learn about problem solving as they articulate assumptions and precisely specify steps in the problem solving approach. Generalizable cognitive benefits such as skills in planning and analysis should theoretically follow.

Various theorists have also argued for the possible cognitive benefits resulting from learning to program. For instance, Feurzeig, Horwitz and Nickerson (1981) postulate an extensive set of cognitive outcomes that may emerge from learning to program. They include: rigorous thinking, precise expression, enhanced self-consciousness about the process of problem solving etc. These two theorists argue that the teaching of the set of concepts related to programming can be used to provide a foundation for the teaching of mathematics, and indeed for the notions and art of logical and rigorous thinking in general.

Linn (1985), and Dalbey and Linn (1985), in examining the cognitive consequences of programming instruction, identify a chain of cognitive accomplishments from learning programming that consists of (1) learn the language features; (2) learn to design programs to solve problems; and (3) learn problem solving skills applicable to other formal systems. They argue that with appropriate instruction and computer access, *"many students can solve computer programming problems and some may gain generalizable problem-solving skills from introductory programming courses"* (p. 29). However, they suggest that such outcomes will not occur unless serious efforts are made to improve the curriculum and the preparation of teachers for programming classes.

More modestly, Harvey (1982a) points out that because LOGO is procedural, interactive, recursive, extensible, and user-friendly, it is designed to make explicit many of the fundamental ideas of computer programming such as procedural thinking and debugging.

Papert (1980) suggests that in order for these cognitive benefits to occur, the emphasis, when learning programming, should be on the processes of problem solving as well as on the content of problem solving. That is, if the achievement of

abstract thinking and efficient problem solving were the end, then the emphasis on the processes of problem solving in the form of programming would be considered as the means to that end.

The argument that the emphasis should be placed on the processes of problem solving has been gaining support from various researchers eg, Wills (1984), Clarke and Chambers (1984a), Burton and Magliaro (1986), Au and Leung (1988), and Nolan and Ryba (1986).

Concerned with the fact that most of the current LOGO textbooks and manuals put their emphases on the content rather than the process of programming, Wills (1984) proposes that an emphasis on the various problem solving processes when learning LOGO programming may produce an entirely new dimension to the nature and quality of learning experience. That in turn may lead to an improvement in problem solving skills outside a LOGO environment. Not surprisingly, Wills (1984) advocates that LOGO programming should be used as a tool for exploring and developing problem solving strategies.

A similar concern was also reflected in the work of Chambers and Clarke at Deakin University in Australia (Chambers, 1984a, 1984b; Clarke & Chambers, 1984a, 1984b) when they designed their LOGO Activities Program for children. This program has its focus upon the development of a number of learning skills such as coding, experimentation, predicting, analysing and planning, using models, and debugging. They argue that all these skills can be useful for solving problems both within the classroom and in everyday life. To them, the important emphasis should be on the knowledge of how to tackle problems in a general way rather than to solve particular problems. In other words, the problem solving processes should be attended to as well as the content of programming. However, Chambers (1984b) remarks that further research is required to study the development of general planning and implementation process, as well as the growing awareness by the learner of the nature and the use of these processes.

Concern about the importance of the process of learning as well as its content is also evident in the work of Nolan and Ryba (1986). Similar to the propositions and models suggested by Wills (1984) and, Clarke and Chambers (1984b), Nolan and Ryba (1986) developed a model to assess learning with LOGO. In this model,

the emphasis is placed upon the systematic development and evaluation of the different cognitive processes (eg coding, creativity, predicting, experimenting, analysis and planning, and debugging), the processes used when a student learns to program at different LOGO levels, from the more concrete (eg basic Turtle commands) to the more abstract levels (eg list processing and interactive programming). This model reflects a move to place more emphasis upon the process of learning in contrast with the content of learning which has commanded so much attention in the traditional curriculum and teaching orientations.

Other recent studies suggest that learners' problem solving skills might be improved when they are taught to reflect on the various problem solving processes (eg planning and debugging) and therefore on how they themselves think while solving problems (cf Gorman & Bourne, 1983; Clements & Gullo, 1984; Horton & Ryba, 1986; Au & Leung, 1988; Nastasi, Clements & Battista, 1990; Swan, 1991).

Summary

In summary then, it has been postulated by Papert and other researchers that LOGO programming can bring about certain cognitive benefits because of the supportive social environment and the positive affective changes in the learners. Within the LOGO environment as Papert conceived it, learners not only learn how to solve problems in programming but also learn a wide variety of skills such as general problem solving, metacognitive knowledge about problem solving behaviour, planning skills, and rigorous mathematical thinking (see Feurzeig, Horwitz, & Nickerson, 1981). These can then be used in contexts other than programming.

> *In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building. (Papert, 1980, p. 5)*

If he is correct, what Papert has suggested has very promising implications for education. For decades, educators have attempted but with little success to find a medium for teaching generalizable problem solving skills such as planning,

analysis, monitoring, and evaluation through the teaching of Latin, logic, writing system and mathematics (cf Ginther & Williamson, 1985; Pea & Kurland, 1987).

However, despite these various claims by Papert and support by some researchers, there remains much scepticism about the educational value of LOGO (cf Mitterer & Rose-Krasnor, 1986; Pea & Kurland, 1987), and ways in which LOGO can be integrated into the existing curriculum, especially given its emphasis on the learning processes per se (Watt, 1982; Horton, 1986).

At the practical level, given the emphasis on the content of the subject matter in the traditional approach in school teaching, this "new" approach in the teaching of LOGO is often seen as incompatible. Moreover, according to Papert (1980), the role of the teacher in a LOGO environment places a strong emphasis in terms of facilitating learning rather than the traditional conveying and teaching of subject content. This often is in conflict with the ways that most teachers were trained to teach. Higginson (1982) remarks that *"LOGO is child-centred and, at least on the surface, unstructured and nonhierarchical"* (p. 329), and as a result, many teachers would feel the pressure to have formal, hierarchical and content-centred curricula.

Further, although Papert's arguments might appear convincing, they are not yet fully articulated nor yet fully supported either by theoretical argument or empirical data (cf Pea & Kurland, 1987; Mitterer & Rose-Krasnor, 1986). For instance, Pea and Kurland (1984b, 1984d) point out the difficulty involved in transfer across knowledge domains and argue that the functioning of abstract thinking are extricably linked to specific problems in assessment as well as the level of expertise attained in programming. Thus they label Papert's ideas as "technoromanticism". Tetenbaum and Mulkeen (1984) have even called for a moratorium on work with LOGO. Confidence in the viability of Papert's claims has yet to be based upon solid empirical evidence (Au & Leung, 1988).

Accordingly, the next chapter will focus on research in LOGO programming with a view to putting some of the argument to empirical test.

CHAPTER FOUR

REVIEW OF RESEARCH ON LOGO PROGRAMMING

This chapter reviews the research in programming with the computer language LOGO, in particular, the relationship between LOGO programming and the cognitive development of the learners. The first section provides a general overview about research on LOGO programming. This is then followed by a review of the major types of LOGO research according to the outcome variables studied. A critical examination of the various issues pertaining to LOGO research, especially those of a cognitive nature, together with discussions of the major concerns and outcomes of these research will then be presented.

Overview

Research on programming with LOGO is still in its early stages of development. Most of the initial research or "evidence", although indicating potential benefits for learners, is in the form of intuitive thinking or anecdotal in nature rather than based upon empirical research (cf Watt, 1982; Ginther & Williamson, 1985; Horner & Maddux, 1985; Irwin, 1985; Burton & Magliaro, 1986; Khayrallah & Meiraker, 1987; Krasnor & Mitterer, 1984; Maddux & Johnson, 1988). Much of these earlier research consist mainly of reports of teachers or researchers randomly observing individual or groups of students interacting with computers without systematic investigation or testing (eg Blitman, Jamile, & Yee, 1984; Chiang, Thorp, & Lubka, 1984; James, 1986; Michayluk, 1986). Typically, the anecdotes resemble the following, *"At times, I could almost 'see' the children 'think' and that's a thrill"* (Blitman, Jamile, & Yee, 1984, p. 19).

Many of the pioneering studies conducted at the Artificial Intelligence Laboratory of Massachusetts Institute of Technology (MIT) by Seymour Papert and his colleagues exemplifies this approach. Virtually all of their research depends on descriptive data and small subject numbers (eg Papert 1972, 1984; Papert, Watt, disessa & Weir, 1979). For instance, in a report published in 1972, Papert (1972)

presents anecdotes about children getting "close to mathematics" by making spontaneous small discoveries while manipulating the LOGO turtle. Likewise, in "Mindstorms", Papert relates various "stories" about children who previously feared mathematics but later came to love learning mathematics using LOGO. More recently, Papert (1984) reports about an extensive experiment he conducted in the Lamplighter School in Texas. However, apparently he in reporting, confines attention to one first grader, who, against all odds, was able to explain the theory of directions and navigation. In the same report, he also described a young girl who used LOGO in language learning thus attesting herself as a person capable of commandeering adult knowledge.

The results of this type of research, although useful as pilot studies of large scale projects, in the provision of rich descriptive data, and are at times "illuminating", are insufficient at the moment to provide convincing evidence.

A number of reasons were suggested for the lack of empirical testing with much of the early research (cf Gorman and Bourne, 1983; Maddux & Johnson, 1988). First, so many of the pioneers in LOGO claimed to have seen such dramatic changes in students in case-study work that they regarded formal testing as unnecessary. Second, until recently, LOGO was available only on relatively expensive machines, which effectively prohibited research on more than a few subjects. Third, researchers were often ethically constrained from running a true experimental design.

Other earlier projects conducted in the late 1970's and the early 1980's (eg the Brookline LOGO Project, the Edinburgh LOGO Project, the Computers in Schools Project and the Lamplighter School LOGO Project etc.) seem to suggest some positive evidence of the potential beneficial effects as advocated by Papert (Watt, 1982). Consequently, much enthusiasm was generated for many educators to adopt LOGO in their classrooms, making LOGO one of the most popular programming languages taught in schools. As well, a lot more research on the various claims about the potential benefits of LOGO was carried out during mid and late 1980's.

More recent empirical studies although still limited in number and their scope to address the many broad claims that have been made about LOGO programming

(Kurland, Pea, Clement, & Mawby, 1986), seem to highlight the unresolved debate about potential cognitive gains within a LOGO environment, and in particular, the possible transfer of these gains from the LOGO context to others. Papert's claims, especially those of a cognitive nature, have been supported by some researchers but not by others. For example, studies by Gorman and Bourne (1983), Reiber (1983), Clements and Gullo (1984), Clements (1986a), Clements (1987b), Horton (1986), Hughes, Macleod and Patts (1985), Au and Leung (1988), Lehrer, Guckenberg, and Lee (1988), Clements (1990), Ortiz and MacGregor (1991), and Swan (1991) all found some positive results. However, counter findings are equally available. du Boulay and Howe (1982), Pea (1983), Pea and Kurland (1984b), Carver and Klahr (1986), Chambers (1986), Cuneo (1986a), Lehrer and Smith (1986), Mitterer and Rose-Krasnor (1986), Howell, Scott and Diamond (1987), and Mendelsohn (1984, 1985, 1987) have discerned little, if any, transfer of learning from the LOGO situation to similar non-LOGO tasks.

Such conflicting results are perhaps not surprising given the small number of subjects involved in some of the studies, and the types of research methods and analyses used. As well, a wide range of instruments was used in the assessment of the various outcome variables such as mathematical achievement, logical reasoning, and problem solving skills (Au, Horton & Ryba, 1987). Of particular importance to LOGO research is the issue of the type and/or amount of programming training provided, or the type of LOGO learning environment created for the subjects during the course of research (Leron, 1985; Papert, 1987; Rowe 1991).

The following sections will examine research with the LOGO language in more detail. They will be discussed according to the outcome variables studied.

Outcome variables

Overall, in the last twenty years or so since the initial development of LOGO, research on the effects of LOGO programming can be grouped into five clusters according to the outcome variables examined in these studies (see Appendix 1). These clusters are: (i) mathematical learning; (ii) affective changes; (iii) social interaction; (iv) cognitive changes and their transfer to other contexts; and (v) metacognitive skills. These studies were often conducted in response to the

many claims made about LOGO programming (cf Chapter 3). Although research which examined the relationship between LOGO programming and mathematical learning could arguably be classified under the development of general cognitive abilities and problem solving, they will be discussed in a separate section because (i) LOGO was initially designed to facilitate the learning of mathematics; and (ii) a number of studies were conducted to examine specifically the relationship between LOGO programming and mathematical development.

The following sections will examine each of these five clusters of research.

LOGO programming and mathematical learning

Akin to one of the original notions that LOGO could be used to facilitate the development of mathematical learning among students, a number of studies have been conducted to examine the relationship between LOGO programming and mathematics learning. These studies, *inter alia*, examined mathematics related variables such as attitudes towards mathematics (Evans, 1984; Horner & Maddux, 1985; Ortiz & MacGregor, 1991), learning and development of mathematical concepts (Milner, 1973; Howe, O'Shea & Plane, 1980; Reiber, 1983; Finlayson, 1984; Horner & Maddux, 1985; Hughes, Macleod & Patts, 1985; Noss, 1987a; McDougall, 1988; Turner & Land, 1988; Lehrer, Guckenberg, & Lee, 1988; Schaefer & Sprigle, 1988; Campbell, Fein & Schwartz, 1991; Ortiz & MacGregor, 1991), mathematical attainment (Finlayson, 1983; Battista & Clements, 1986), learning of mathematical strategies and their transfer from LOGO to normal school mathematics (Finlayson, 1985; Lehrer & Smith, 1986; Thompson & Chen Wang, 1988).

The results of these studies, although inconclusive, suggest that LOGO programming could facilitate mathematical learning. The more decisive results came from the first two categories of studies which examined the relationship between LOGO programming with (i) students' attitude towards mathematical learning; and (ii) the learning and development of mathematical concepts.

Evans (1984), for example, found that LOGO programming had a positive influence on students' attitudes towards learning mathematics after they have spent 45 hours in a year learning mathematics through LOGO. The study by Ortiz and

MacGregor (1991) with 89 sixth grade students found that students had more positive attitudes towards mathematics, especially the computer related concepts after they had spent some 10 hours learning mathematics using LOGO.

When researching the relationship of LOGO programming with the development of mathematical concepts, Milner (1973) found that LOGO programming was able to facilitate the understanding of number sequences and variables after 15 weeks of intervention. Howe, O'Shea and Plane (1979) also demonstrated that LOGO learning could improve their subjects' understanding of certain algebraic topics. Furthermore, the teachers involved in this study indicated that the LOGO students could argue sensibly about mathematical issues and explain their own mathematical difficulties more clearly than the control group. Moreover, it was found that students who programmed LOGO procedures with variables demonstrated greater long term retention of their understanding of the concept of variables than students who worked with textbook (Ortiz & MacGregor, 1991). Similarly, Reiber (1983), Hughes, Macleod and Patts (1985), Noss (1987a), Turner and Land (1988), Lehrer, Guckenberg, and Lee (1988), Schaefer and Sprigle (1988), McDougall (1988) and Campbell, Fein and Schwartz (1991) all found that LOGO programming could be used as a tool to promote the learning of mathematical concepts.

A number of factors could have contributed to these results. First, computers and the LOGO language were relatively novel to most students in the last twenty years, hence students were attracted to them. Second, LOGO, compared to traditional teaching of mathematics in the classrooms, could provide an interesting alternative learning environment where learners could learn various mathematical concepts by constructing geometrical patterns on the computer screen. Third, many mathematical concepts, eg angles, shapes, coordinate systems, and distance, could be represented in a more "concrete" manner with the LOGO language, i.e. LOGO graphics might have provided a visual mental model for the learners, hence facilitating understanding and retention. Fourth, researchers and teachers often cooperated to develop more interesting ways and curricula to teach mathematics in these research projects.

The more tentative and less convincing results came from the other two categories of research on LOGO programming and mathematics learning, i.e., mathematics attainment, and learning of mathematical strategies and their transfer from LOGO to normal school mathematics.

In the area of mathematical attainment, Finlayson (1983:10) observed that her subjects had learned to *"think mathematically through LOGO experience"* and that a great deal of enthusiasm was generated. However, the researcher also noted that these subjects could appear to be competent at turtle graphics without comprehending the underlying mathematics. Clements (1986c) found that students' mathematical achievement did not show any significant improvement after 22 weeks of learning of LOGO programming. Similarly, Battista and Clements (1986) found that there was no significant difference among the experimental and control groups on a mathematics achievement test after nearly one year of LOGO programming. As a result, Battista and Clements advanced that either the coverage of standard mathematics in LOGO programming was too slight to lead to significant gains in mathematics achievement, or, the students did not see the connection between the concepts they encountered on the computer and classroom mathematics tasks, therefore, transfer to standard mathematics achievement was minimal. Consequently, these researchers suggested that attempts should be made to make explicit connections between students' work with LOGO and their classroom mathematics work.

Different types of problems were noted in some of the studies which investigated the transfer of mathematical strategies to normal classroom settings. Although Finlayson (1985), and Thompson and Chen Wang (1988) found some evidence of transfer of mathematical strategies and concepts to normal classroom setting, both studies had serious flaws in their designs. In the study by Finlayson (1985), although it was found that the experimental group performed better than the control group in the transfer and abstraction of mathematical strategies, a pre-test was not administered, hence there were difficulties in ascertaining whether there was any prior difference between the two groups. Also, it was hard to establish the equivalence of mathematical experience between the experimental group and the control which used a number of computer packages. Thompson and Chen Wang

(1988) found in their study that LOGO programming was able to facilitate the transfer of mathematical concepts. However, there are a number of major flaws in this study such as non random assignment of subjects, and that the experimental group spent more time on mathematics learning than the control group. Therefore, the results of these two studies must be viewed with caution.

The study by Lehrer and Smith (1988) exercised much tighter control in the examination of instructional variables. Forty-seven third graders were provided with nine weeks of LOGO instructions. Two different groups were used: one with teacher mediated instruction and the other with more traditional instructions. It was found that students who were better instructed were able to use their knowledge in LOGO to solve mathematical problems when reminded how such knowledge could apply to the problems. This study provides reasonable evidence that explicit instructions in the use of mathematical knowledge and skills can facilitate the transfer of mathematical strategies from LOGO to normal classroom mathematics. However, it is also clear that the subjects in this study had not internalized the knowledge sufficiently as they needed reminding in the application of these knowledge in other contexts.

In summary, studies on the relationship between LOGO programming and mathematical learning yielded some evidence that the learning of mathematics might be assisted by LOGO programming, especially in the fostering of positive changes among learners' attitude towards mathematics, and improving the understanding of some mathematical concepts. Researchers such as Schaefer and Sprigle (1986), and Noss (1987) suggest though, that in order for better development to occur, it is necessary to examine the use of instructional methods and careful teacher intervention strategies.

Results with studies on the transfer of learning of mathematics and strategies to normal classroom settings were less convincing. Seemingly, one of the major problems was that students failed to see the connection between what they learned and applications in other contexts. Consequently, some researchers suggested that instructions should make these connections explicit to the learners. Therefore, in researching the effects of LOGO programming on the transfer of mathematical strategies to normal classroom settings and mathematical attainment, it is important

to consider the use of carefully orchestrated teacher intervention strategies so that learners can (i) become aware of the connections; and (ii) practise and internalize the strategies sufficiently so as to enable transfer. The issue of instructional strategies will be further discussed in a later section of this chapter when LOGO treatment is considered.

LOGO programming and affective changes

Attribution theorists have argued that it would be important for learners to feel that they are in control of their learning environment (cf Weiner, 1974, 1979). In particular, it has been suggested better learning could be fostered by (i) the lack of external factors to which students might attribute success or failure, and (ii) learners being able to explore their learning without penalties for making mistakes (eg Ryba & Anderson, 1990). Papert (1980) contends that LOGO could provide such a learning environment.

The second major category of LOGO research revolves around the examination of the relationship between LOGO programming and the affective changes of learners. The variables examined in these research included: motivation (Weir, 1981; Zelman, 1985; Nastasi, Clements & Battista, 1990); locus of control (Horner & Maddux, 1985; Olsen, 1985; Burns & Hagerman, 1989); and attitudes towards computing and the learners themselves (Irwin, 1985; Schibeci, 1990).

The results of these studies offered reasonably robust evidence in establishing the effect of LOGO programming on the affective development of the students. Weir (1981), for instance, observed that LOGO programming could provide a high degree of motivation for severely handicapped children although it was unclear as to what role the teachers played in the process of teaching and learning. Zelman (1985), concluded from her study that "exposure to LOGO through purely an inductive teaching method was inappropriate to motivational orientations of the students" (p. 17). She suggested that future research should examine the combination of computer program design and teaching of instructional and feedback practices. More recent work by Nastasi, Clements and Battista (1990) found that LOGO could foster more intrinsic motivation and the development of self-reward systems for the learners. This study also confirmed that carefully orchestrated social

and social-cognitive interactions (for example, encouragement of learner-directed work and cooperative learning) are important for students to develop such motivation and self-award systems.

Locus of control was another important variable studied within this category of research. Horner and Maddux (1985) studied 74 subjects and found that LOGO programming could be effective in making learning disabled and non learning disabled students feel responsible for their success with LOGO activities. The female subjects in a study by Olsen (1985) also developed increased feelings of responsibility and personal control. Similarly, Burns and Hagerman (1989) found that their subjects, after learning LOGO programming over a 4½ month period, showed significant increase in internal locus of control.

Irwin (1985) used two languages, LOGO and BASIC with a group of 140 high school subjects in New Zealand. This study showed that the LOGO group continued to exhibit high interest while there was a decline of interest in the BASIC group. As well, the LOGO group tended to spend significantly more time "on task" compared to the BASIC group. On the other hand, Schibeci (1990) used LOGO with adult learners (both pre- and in-service teachers) in Australia. The subjects demonstrated a marked improvement in their attitude towards computers and as well as their confidence in solving programming problems with LOGO.

In summary, studies which examined the relationship between LOGO programming and affective changes of the learners offered reasonable support that LOGO programming could have a facilitative effect on motivation, internalization of locus of control and attitude of learners towards computers and problem solving. Of particular relevance to the present study was the fact that two different types of LOGO learning environments were employed in research in this category, that is, the use of self-discovery learning environment (eg Evans, 1984; Olsen, 1985; Lehrer, Harckham, Archer & Prazek 1986) and more structured learning environment (eg Nastasi, Clements & Battista, 1990). It would appear on the surface that both types of learning environments were able to facilitate positive affective changes among the learners. However, studies by Zelman (1985), and Nastasi, Clements and Battista (1990) clearly suggest that a purely self-discovery learning environment was not sufficient to develop intrinsic motivation and self-

reward system for the learners, rather, carefully designed teacher intervention would be needed. The issue of learning environment will be discussed further in a later section of this chapter.

LOGO programming and social interaction

The investigation of social interactions within educational environments has been considered important. Apart from being one of the fundamental goals of education, it has also been argued that social interaction is an essential component that facilitates cognitive growth (cf Vygotsky, 1978; Emihovich & Miller, 1986; Salomon, 1988; Nastasi, Clements & Battista, 1990). According to Vygotsky (1978), higher order mental capabilities progress from external to internal processes, and that the mechanisms underlying the internalization of higher mental functions is social interaction. Emihovich and Miller (1986) contends that the transformation process from externalization to internalization occurs as children engage in meaningful mediated verbal interactions with adults and/or peers during LOGO learning.

The third major category of LOGO research focused on the relationship between LOGO programming and social interaction. A number of variables were examined: peer interactions (Hawkins, Homolsky & Heide, 1984; Carmichael, Burnett, Higginson, Moore, & Pollard, 1985; Burns & Coon, 1990); interaction and problem solving (Clements & Nastasi, 1988; King, 1989; Nastasi, Clements & Battista, 1990); effects of group size (Guntermann & Tovar, 1987); interactions of parents and children (Williamson & Silvern, 1986); behaviours of learners (Mitterer & Rose-Krasnor, 1986); and gender differences (Webb, 1985; Guntermann & Tovar, 1987; Hoyles & Sutherland, 1989).

One of the earliest studies on peer interaction within a LOGO environment was conducted by Hawkins, Homolsky and Heide (1984) who studied 100 subjects over a two year period. Various data were collected through interviews, videotaping, and collection of ethnographic materials. This study reported more collaboration among the subjects compared to other classroom tasks, and that the computer provided an engaging problem solving context in which task related talk occurred, although very little systematic results were presented, and it was also

unclear how the data were analyzed. Mitterer and Rose-Krasnor (1986) provided further insight into peer collaboration. In their study, it was found that there was a higher level of social interaction among the computer training groups (both LOGO and BASIC), although the interactions were not the same across groups. It was observed that the LOGO group had higher incidence of interactions with the tutor, while the BASIC group was characterized by relatively more peer only contacts. However, this study did not provide sufficient descriptions as to how the instructional conditions might have been different for the LOGO and BASIC groups.

In one of the more comprehensive LOGO studies, involving some four hundred students in 18 different schools, Carmichael, Burnett, Higginson, Moore, and Pollard (1985) reported that social interaction did not only facilitate socializing process, but was also a critical component in the furthering of cognitive development and of creative expression. The researchers concluded that a good learning environment must incorporate strategies that would facilitate some social interaction among the learners.

While these studies have highlighted the dynamics of social interactions within a LOGO environment and the importance of a facilitative environment for cognitive development, they really have not been able to study the interaction in depth necessary to understand the types of social interaction that might assist the development of thinking skills among their subjects. Rather more systematic methods need be developed to investigate the types of social interaction among learners in a LOGO environment, and how these interactions might facilitate the development of thinking skills.

A more recent study by Burns and Coon (1990) provides further information on this aspect. This research investigated the type of verbal interactions among their subjects while learning LOGO programming and a control programming language (Delta Drawing). Data for three different types of verbalizations were collected: process-oriented, product-oriented and additional. It was found that peer collaboration using LOGO focused more on the process relative to the product of problem solving when compared to a control programming language. However, given the design of the study, it could not be established whether the types of

verbalizations were the results of the different languages or teacher intervention within each different environment.

Results of the study by Clements and Nastasi (1988) offered further support for the hypothesis that LOGO programming could facilitate peer interaction in the aid of social problem solving. The LOGO group in their study was more able to decide on the nature of the problem, decide on solution processes, and monitor cognitive processes, when compared to a similar CAI group. The behaviours that were observed included conflict resolution, rule determination, and self-directed work, which were behaviours expected to occur in problem solving situations. This work was extended by the study of Nastasi, Clements and Battista (1990) which found that a mediated LOGO environment could assist cognitive development by fostering more resolution of cognitive conflict which resulted in better problem solving performance. These two studies highlighted the importance of the need for a carefully constructed LOGO environment with social interaction elements which would in turn facilitate problem solving behaviour.

The need to incorporate facilitative social interactions in a LOGO environment is further supported by Williamson and Silvern (1986), and King (1989). For instance, Williamson and Silvern (1986) found that children performed better on generalization tasks when they worked with parents who provided them with more directions. King (1989) observed that successful LOGO learners asked more task related questions, spent more time on strategies and reached higher levels of strategy elaboration than did unsuccessful LOGO learners.

The issue of sex differences in peer interactions was explored by Webb (1985), and Guntermann and Tovar (1987) with conflicting results. Webb (1985) found that there was no significant differences found in both the learning outcomes and interaction behaviour either as groups or as individuals. The researcher concluded that sex might not be operating as a status characteristic when students learned computer programming. On the other hand, Guntermann and Tovar (1987) observed significant differences among male, female and mixed groups. It was found that (i) males were more likely to displayed solidarity than female or mixed groups; (ii) females were much more likely to express agreement with their peers; (iii) there were also more asking of information in the male group than female

groups, and (iv) males expressed more antagonism than females or mixed groups. These results are interesting as often programming has been perceived as belonging to the mathematics or science domains, and girls tend to perform differently from boys. Results from the study by Webb (1985) suggest that LOGO programming might have a mediatory effect on the interaction between males and females whereas findings by Guntermann and Tovar (1987) indicate otherwise. A plausible explanation for this discrepancy could be the ways in which subjects were encouraged to interact with each other. However, this is unclear from the reports of either studies.

A three year longitudinal study of secondary school students by Hoyles and Sutherland (1989) using a case study approach highlighted a number of sex differences in social interactions within a LOGO environment. It was noted that boys found it difficult to share interactions with their partners and tended to dominate interactions in mixed pair. They also tended to be more competitive and worked on their own. In contrast, girls were more likely to share ideas with their partners and that they preferred to choose loosely defined goals. Moreover, this study observed that teacher intervention was crucial in facilitating social interaction among the students.

In summary, the studies on the relationship between LOGO programming and social interactions have offered some evidence that LOGO programming could influence social interactions among learners. Some of these studies have stressed the importance of social interaction in the aid of cognitive development. However, none of these studies have managed to illustrate whether the increased interaction was the result of the LOGO programming language alone, or the result of some instructional variables working in conjunction with the LOGO language. Future studies, therefore, must attempt systematically to identify the effects of individual instructional variables engendered in a LOGO environment on the type of social interactions ensued, and how these social interactions might in turn affect the problem solving performance of the subjects. Moreover, one of the major criticisms of many contemporary research on LOGO stems from the fact that many studies only focused on the end products of learning, that is, how LOGO have affected product variables such as mathematical learning and cognitive changes (cf Papert,

1987; Rowe, 1991). It was suggested by many researchers that it is important to examine the changes in process variables such as social interaction and learning strategies.

LOGO programming and cognitive changes

While a number of studies have been conducted to examine the relationship between LOGO programming and variables such as mathematics learning, affective changes, and social interaction, the vast majority of LOGO research in the last twenty years, in particular, the 1980's, focused on LOGO programming and cognitive changes. Most of these studies were conducted in response to claims made by Papert and others that LOGO programming could facilitate learners' general cognitive development and assist the transfer of learning from a LOGO environment to other contexts (cf Chapter 3).

Some of these studies attempted to investigate the correlation between various factors related to the learning of programming. These studies included the examination of the relationship between: programming and understanding of the concept of recursion (Kurland & Pea, 1983); mastery of programming and operational thinking (Mendelsohn, 1984, 1985, 1987); LOGO programming and top-down processing, field independence, holistic tendency, and academic achievement (Bradley, 1985); competence with the syntax and semantics of the LOGO language (Campbell, Fein, Scholnick, Schwartz & Frank, 1986); analogical reasoning and writing subprocedures and use of variables (Clement, Kurland, Mawby, & Pea, 1986); mastery of programming concepts (Cohen, 1987); cognitive development and understanding of LOGO commands (Fay & Mayer, 1987); extended workstations and learning of programming (Heller, 1991).

However, the majority of studies within this category were of a causal nature, i.e., how the learning of LOGO programming could effect certain cognitive outcomes. Three major types of research could be discerned from studies within this cluster depending on the outcome variables studied. These three types are: (i) global changes in the cognitive development; (ii) improvement of logical reasoning abilities; and (iii) development of problem solving skills and their transfer to other contexts. Whilst the three different types of research are inevitably closely linked,

their different emphases warrant separate discussion. The following sections will review these studies in more details.

LOGO programming and global cognitive development

Part of the underlying theoretical basis for LOGO learning is Piagetian. It has been suggested that LOGO programming can (i) provide an environment within which learners can build their own intellectual structures; and (ii) allow children to master ideas formerly thought too abstract for their developmental level. One of the often cited argument by Papert is that "computer can concretize (and personalize) the formal [abstract]" (Papert, 1980:21), i.e., LOGO programming would facilitate the shift of the boundary separating concrete and formal operational thinking. In this way, it has been proposed that LOGO may allow a child to advance their normal developmental operative level faster than children not having access to LOGO (cf Howell, Scott & Diamond, 1987). In response to these claims, a number of studies have been conducted to examine the relationship between LOGO programming and the global changes in learners' cognitive development measured by Piagetian tasks such as classification and seriation.

One of the earliest and most cited work in this area was conducted by Clements and Gullo (1984). This study compared the effects of LOGO and CAI on the following variables: cognitive style (eg reflectivity/impulsivity, creativity); cognitive skills (eg classification and serialization); and spatial orientation (describing directions). A pre-post test design was used. Eighteen subjects (6 year olds) were randomly assigned to one of the two treatment conditions, LOGO programming or computer-assisted instruction that consisted of 12 week at a rate of two 40-minute training sessions per week. The CAI group used a selection of software that provided instruction and/or practice in the skills and abilities of the school system's mathematics, language arts, and reading curriculum. A carefully designed LOGO curriculum was used and both groups received some form of teacher questioning (eg, "Why was that wrong, What will you do to fix it?") which the researchers suggested would help to the children to make their thinking and mastery of concepts explicit.

A number of significant differences were found between the two groups in the area of general cognitive development as a result of the different treatment. The LOGO group was found to perform better in creative thinking (fluency and originality), error and latency, describing directions. However, no differences were found between the groups in the two areas of general cognitive development (classification and seriation, and other specific aspects of cognitive development as measured by McCarthy Screening Test). According to the researchers, the modest results for this study could be explained by the experiments being of too short a duration in order to permit an exact evaluation of the effects of LOGO on cognitive development. In spite of this, the researchers attributed the positive results concerning cognitive style and spatial orientation to the nature of programming (planning, analysis and debugging). Although the results of this study appear to be promising, they must be viewed with caution as the number of subjects in each group was only nine. Also, the design of the study made it difficult to determine whether the positive results were due to LOGO programming alone or was it a combination of the LOGO programming and teacher intervention.

An investigation with similar design was conducted by Clements (1986c). In this study, a larger number of subjects were used (72 instead of 18, half of the subjects being third graders and the other half being first graders) with a longer training period (22 weeks instead of 12). These subjects were divided into three sub-groups - LOGO, CAI, and control. The treatments and dependent measures used were also similar to the study by Clements and Gullo (1984). However, a less structured approach was used with less adult-children interaction and more independent work by children on their own projects.

The results of this study indicate that the LOGO-group differed from the others in their cognitive performance. One of the major differences from the previous study was that in the classification test, the LOGO subjects showed a significant gain in the post-test and their score was significantly higher than that of the CAI-group. As far as the control-group is concerned, their higher performance level at the time of the pre-test prevented any comparison with the other groups. For the seriation test, only the 1st-year LOGO-group showed a significant gain in the post-test. Significant results were obtained as well in the Torrance Test of

Creative Thinking (TTCT) and the spatial orientation test. Clements (1986c) suggests that these results indicate the important contribution of LOGO to the development of operative competence in children when LOGO intervenes at a given point. This conclusion, however, could be premature, given that all subjects except those of the LOGO-group reached top performance in the pre-test of the seriation test. Similar to the study by Clements and Gullo (1984), the LOGO group in this study outperformed the other groups in creativity.

The results of studies by Horton (1986), Cathcart (1990), and Clements (1991) shed further light on the hypothesis that LOGO programming could facilitate creativity. In these three studies, it was found that the LOGO groups performed significantly better than the control groups on figural creativity and divergent thinking.

These four studies, *inter alia*, offer encouraging evidence in establishing the effects of LOGO programming in areas as such creativity and describing directions, but only some rather tentative evidence in the facilitative effects on the global cognitive development of learners in areas such as classification and seriation. In particular, results on classification, seriation, and reflectivity from the study by Clements and Gullo (1984) were in conflict with those by Clements (1986c). Both studies suffered from the problem of having fairly small number of subjects (in the case of the study by Clements, there were only 12 subjects in each cell with the group x grade analysis). Therefore, the claim that LOGO programming can accelerate the global cognitive development of children must be viewed with caution.

Studies by Lehrer, Harchham, Archer, and Pruzek (1986), Howell, Scott and Diamond (1987), and Turner and Land (1988) lend further support to such circumspection. These studies evaluated the effects of LOGO programming on the general cognitive development of subjects. A variety of dependent measures were used including Piagetian conservation tasks, Euclidean shapes drawing, Social Sciences Piagetian Inventory, and the McCarthy Scales of Children's Abilities. The subjects consisted of children ranging from pre-school to secondary school. The results of these three studies indicate that LOGO programming did not have any significant effects on the general cognitive development of the learners within the confines of these studies.

In summary, the studies reviewed in this section have shown that LOGO programming might be facilitative in certain areas such as cognitive style and spatial orientation. However, LOGO programming might not be able to alter significantly the general cognitive development of young children as postulated by Papert and others, even with reasonably long period of learning as in the study by Howell *et al* (1987), or by carefully orchestrated teacher intervention, as in the studies by Clements and Gullo (1984), and Howell *et al* (1987). Perhaps the training periods of these studies were not quite long enough to produce the results that many LOGO enthusiasts have hypothesized; or the instruments used to measure the various cognitive and metacognitive changes were not sensitive enough to gauge the appropriate changes. On the other hand, it is quite plausible that learning resulting from LOGO programming might be too specific to be transferred to a more general context assessed by Piagetian conservations tasks and the other dependent measures used. The results of these studies tend to lend support to Piagetian theory which would predict that normal developmental factors would dictate the movement of children from one stage to another.

LOGO programming and logical reasoning

Logical reasoning, often considered as a form of mathematical skills as well as an important component in problem solving and critical thinking, was the focus of a number of studies (eg Seidman, 1981; Gorman & Bourne, 1983; Reiber, 1983; Degelman, Free, Scarlato, Blackburn & Golden, 1986; Many, Lockard & Abrams, 1988; Grandgenett & Thompson, 1991). These studies were often premised upon the argument that the style of thinking that one learned from solving any of the four binary rules (conjunctive, disjunctive, conditional, and biconditional) could be attributed to computer programming (Kolata, 1982).

One of the earliest studies which investigated the relationship between LOGO programming and logical reasoning was conducted by Seidman (1981) among 41 fifth graders who were randomly assigned to experimental and control groups. The results of this study indicated that there was no significant difference between the LOGO group and the control group, after 30 hours of programming over a period of 15 weeks, on the conditional reasoning principles used in this investigation.

However, when students' misinterpretation of logical conditional statement in a biconditional manner was taken into consideration, it was found that the LOGO group performed better than the control on one of the logical reasoning principles. These results clearly highlight two issues: (i) there were problems younger children might have in understanding conditional statements in programming; and (ii) LOGO experience could provide "incorrect" learning on one principle of logical learning - inversion (the misinterpretation of normal logical conditional statements in a biconditional manner (cf Seidman, 1989-90)). Therefore future studies need to attempt to overcome these problems by the provision of appropriate curricular and pedagogical conditions.

The results of three exploratory studies with fairly small sample sizes shed further light on the hypothesis that LOGO programming could improve the logical reasoning of learners.

The study by Gorman and Bourne (1983) indicated that differential amounts of LOGO experience might have different effects on the logical reasoning among a group of 15 third-grade students. In this study, the group of children who received one-hour of LOGO instructions over one school year made significantly fewer errors with a rule learning task than the half-hour group. They also outperformed the half-hour group in the number of trials to criterion. This study also established that a conditional rule learning task was appropriate in the measurement of logical reasoning for late primary school subjects (children who are on the transition from concrete operational to formal operational). As well, it was clear from this study that the rule-learning task was more suitably scored by the more sensitive measures of number of errors to criterion and number of trials to criterion, rather than by just success-failure.

The study by Degelman, Free, Scarlato, Blackburn and Golden (1986), on the other hand, found that affirmatively defined concepts were suitable for the measurement of logical reasoning for preschoolers whereas conjunctively defined concepts were too difficult. Results of this study revealed that LOGO programming was facilitative in the development of logical reasoning among very young children when measured by affirmatively defined concepts.

Similarly, the study by Reiber (1983) found a statistically higher performance for the LOGO group than the control group in logical reasoning. However, these findings must be viewed with caution as doubtful procedures were used in the analysis of the combinatory tests (Larivée *et al*, 1988).

When a much larger sample (113 subjects in LOGO group, 58 in control group) was employed in the study by Many, Lockard, and Abrams (1988) with junior high school students, it was revealed that the LOGO group scored significantly better than the control group although further analysis suggested that the male students in the LOGO group achieved significantly higher scores than their male counterparts in control group and that the benefits appeared to accrue primarily to males. However, this study was conducted without pre-tests, hence it was difficult to (i) describe the magnitude of LOGO's effects on reasoning skills; and (ii) ascertain whether there were any differences between the two groups prior to the intervention. Similar inconclusive results were obtained by Grandgenett and Thompson (1991). The 144 subjects in this study were undergraduate students who enrolled in an introductory educational computing class. This study indicated that guided programming instruction facilitated the reasoning performance of college freshmen but hindering that of college juniors.

The studies reviewed in this section offered modest support that LOGO programming might assist the development of logical reasoning among learners ranging from kindergarten to high school. However, given the number of flaws in these studies, their results must be viewed with caution. Future studies need to exercise tighter control on a number of research design elements: (i) the use of larger number of subjects to enable statistically valid comparison; (ii) the establishment of whether there is any difference in the subjects' logical reasoning prior to intervention by pre-testing of subjects on their reasoning abilities; and (iii) most important of all, to examine more closely the possible effects of instructional variables such as teacher mediation, and peer interaction, which might influence the development of logical reasoning skills. The examination of the role of the teachers and the approach adopted in the teaching of LOGO which will in turn allow the evaluation whether the gains were facilitated by LOGO programming alone or with

the assistance of some form of teacher intervention. The issues of LOGO instruction and teacher mediation will be further discussed in a later section of this chapter.

LOGO programming and problem solving

Problem solving is often considered an important aim in modern education. It has been argued that computer programming can be a powerful means of enhancing problem solving skills as computer programming requires the use of a number of problem solving skills such as planning, analysis, and evaluation (eg Kurland, Pea, Clement & Mawby, 1986; Lawler, du Boulay, Hughes, & Macleod, 1986; Chambers, 1986). Given the popularity of LOGO in both primary and secondary schools, the emphasis on problem solving in recent curricular reforms (cf Chapter 1), and some extravagant claims made about LOGO (cf Chapter 3), considerable attention has been given especially to the use of LOGO as a means to create learning environments for the development of problem solving skills (cf Chapter 3; Yates & Moursund, 1988). Also, expectations have been such that the learning of programming will cultivate problem solving skills for transfer to other situations (eg Pea, 1983; Gallini, 1985). Consequently, there has been a large number of LOGO studies which focused on using LOGO programming to enhance the transfer of problem solving to other contexts.

However, results of this research to date have been rather inconclusive. Some studies have found that LOGO programming could facilitate the development and transfer of problem solving skills (eg Carmichael *et al*, 1985; Horton, 1986; Lehrer *et al*, 1986; Gallini, 1987; Lehrer & Randle, 1987; Mathinos, 1990; Au & Leung, 1991; Swan, 1991) while others have found results to the contrary (eg Pea, 1983; Chambers, 1986; Cuneo, 1986a; Kurland *et al*, 1986; Lehrer & Smith, 1986; Mitterer & Rose-Krasnor, 1986; Dalton & Goodrum, 1991).

The study by Carmichael *et al*, for instance, found that their subjects were able to develop problem solving skills while learning to program with LOGO and that these skills were able to transfer to other contexts. Horton's (1986) study revealed that junior high school students were able to improve on a number of problem solving skills such as prediction, exploration, creativity, planning and analysis. Similar skills were found to increase more durably among subjects in a

study by Lehrer and Randle (1987). The study by Au and Leung (1991) with upper primary students suggests that skills learned while programming with LOGO can transfer to tasks that resemble LOGO programming but not to dissimilar tasks. Similarly, a number of studies found that students were able to improve their problem solving skills after learning to program with LOGO (eg Gallini, 1987; Swan, 1991).

On the other hand, quite a number of investigations did not find any increase in problem solving abilities after the learning of LOGO programming. For example, studies by Pea (1983) and Pea and Kurland (1984) failed to find any significant improvement in the planning skills of their subjects after one year of LOGO programming. The subjects in the study by Carver and Klahr (1986) did not develop effective debugging strategies. Study by Chambers (1986) found that LOGO programming did not enhance her subjects' performance in problem solving skills such as experimenting, predicting, coding, planning and analysis.

In examining these studies, a number of factors were found to have contributed to the conflicting results, including: measurement of outcome variables, research design, and the nature of LOGO instruction, in particular, those elements that might have affected the transfer of problem solving skills learned in a LOGO context to another. The following sections will discuss these factors in turn.

Measurements of outcome variables

When analyzing the vastly different results of these studies, it is apparent that a wide variety of measures were employed to determine how LOGO programming might have affected the development of problem solving skills. Consequently, such variation of instruments led to the apparently conflicting results. The multitudes of measures used in gauging the changes of problem solving skills could be traced to the theoretical bases upon which these studies were formulated.

One of the most striking problems with research involving LOGO programming and problem solving is the general lack of theoretical basis for the studies (cf Burton & Magliaro, 1986; Khayrallah & Meiraker, 1987; Clements, 1990). This problem has resulted in the lack of any anchor points to relate the findings with the literature, and difficulty in the selection of dependent measures,

particularly in the determination of transfer of problem solving skills to other contexts.

Some studies did not identify the types of problem solving skills that were supposed to be observed during the learning of LOGO programming. Instead, these studies relied solely on teachers' anecdotes or students' self-reports on the development and transfer of problem solving skills without using any systematic measurements. For example, in a well publicized large scale study conducted on LOGO programming and problem solving in Canada, the researchers could only draw upon the following anecdotal comments in relation to the development of problem solving skills and their transfer to other context: "students themselves expressed a wide range of problem solving... that they felt they learned from working with computers or LOGO, or "teachers commented quite frequently that they perceive that such transfers did occur" (Carmichael *et al*, 1985:279-280).

Some studies, on the other hand, attempted to use a large number of dependent measures of problem solving in the hope of "catching" problem solving skill development and transfer. For instance, in the study by Clements and Gullo (1984), 18 dependent measures such as cognitive style, metacognition, cognitive development, and directionality were used. Although the use of these measures were well grounded in theories of cognition and problem solving, it is rather debatable whether such a large number of skills could be expected to develop within a relatively short duration of LOGO programming.

Some other studies even arbitrarily used dependent measures without examining closely the relationship between LOGO programming and the dependent measures. Often dependent measures were used without appropriate rationale which often gave rise to conflicting results. This has resulted in what Burton and Magliaro (1986) have termed as a "kitchen sink" approach. For example, a study by Soloway, Lockhead, and Clement (1982) investigated programming effects on problem solving without any reference to the problem solving literature. Khayrallah and Meiraker (1987) suggest that this lack of proper theoretical underpinnings has resulted in an approach that focus on testing what works and does not work, rather than attempting to specify the underlying processes that might determine success or failure.

Against this background, a number of researchers have alerted to the development of "technocentric thinking" among some researchers and educators. Papert (1987), for instance, cautions that this approach represents a tendency in thinking of LOGO as an agent that acts directly on thinking and learning. Rowe (1991) argues that computers are far more than just a treatment, that they have become inextricably intertwined not only with the way students might go about cognitive tasks, but with the whole context of learning and teaching. In relation to the selection of dependent variables, researchers need to examine critically the types of problem solving skills that learners might develop while learning to program with the LOGO language. Once the type of skills have been identified, then care needs be exercised in the justification of the choice of dependent measures on both theoretical and practical bases. At a theoretical level, it is important to consider the issue of transfer of learning to other domains. Moreover, many researchers have argued for the importance of measuring the success of LOGO learning via task and strategy oriented variables rather than the traditional product oriented measurements (King, 1989; Burns & Coon, 1990; Swan, 1991). As this issue is central to this thesis, it will be considered in more details in the next chapter. At a practical level, it is important to deliberate the extent to which problem solving skills might be developed within the time limits of most studies. Given the generally short duration of many LOGO studies, it is useful to focus on instructional conditions that would foster the development and transfer of a smaller number of problem solving skills among the learners in a more intensive manner.

Research design

The second concern relates to the design of this type of research: researchers must not neglect the most important components of learning situations, *viz*, people and cultures - in the facilitation of learning. That when researching LOGO programming and problem solving, attention needs be given to the examination and evaluation of the programming culture (viz. learning environment) as well as the use of psychometric instruments to measure any potential cognitive gains (Papert, 1980, 1987; Leron, 1985; McDougall, 1988; Mehan, 1989; Rowe, 1991). For instance, the dynamics of learning with LOGO in either small groups or individually, and the

qualitative changes among the learners might not be measured by traditional psychometric instruments.

In the overview of this chapter, it has been stated that some of the earliest LOGO research relied heavily on anecdotal evidence without being able to provide convincing evidence. Some of the studies on LOGO and problem solving conducted in the early 1980's attempted to overcome this problem by employing the more traditional experimental design. A typical study of this kind would pretest the subjects, introduce the subjects to LOGO programming, then posttest them to find if there were any gains in problem solving skills.

For example, a series of studies conducted at the Bank Street of College of Education in New York focused on the transfer of problem solving skills to other non computing contexts such as the planning of classroom chore scheduling tasks (Pea, 1983; Pea & Kurland, 1984b). However, on a large number of measures such as efficiency of planning, the quality of revisions, and the types of decisions made during the planning process, no statistically significant differences were found between the programming and non-programming groups. The authors concluded from these series of studies that programming experience did not appear to transfer to other domains which shared analogous formal properties. Studies of similar design since have yielded conflicting results (eg Clements & Gullo, 1984; Chambers, 1986; Horton, 1986).

The use of this type of research design has raised serious concerns among many LOGO researchers and there has been debate conducted regarding the suitability of such design (cf Leron, 1985; Becker, 1987; Pea, 1987; Papert, 1987; Walker, 1987; McDougall, 1988; Rowe, 1991). It has been argued by some researchers that the traditional experimental design was not sufficiently sophisticated to measure potential cognitive gains of the learners. Rather, it is important to examine the process of learning including the dynamics of learning as well as the product measured via psychometric instruments (Leron, 1985; Papert, 1987; McDougall, 1988; Rowe, 1991).

More recent studies in the late 1980's have taken such concern into consideration and often attention has been given to the study of interactions among the learners that might have affected any cognitive outcomes of such studies. Some

studies have begun to examine more closely the type of interactions that might have contributed to the changes in problem solving skills of the learners and what the learners might have gained during the processes of learning to program with LOGO. For example, Clements and Nastasi (1988) studied 24 first graders and found that the LOGO group exhibited a significantly higher percentage of social behaviours that have cognitive underpinning. The study by Nastasi, Clements and Battista (1990) found that their LOGO group evinced more cognitively oriented conflict, attempts at and successful resolution of conflicts, and rule making.

As well, there has been an increasing awareness of the need to control the instructional conditions that might facilitate cognitive growth. Recent studies (Lehrer, Guckenberg, & Lee, 1988; Au & Leung, 1991; Heller, 1991; Ortiz & MacGregor, 1991; Swan, 1991) have all employed research designs that enabled these studies to examine the effects of various instructional conditional on the development of problem skills among their subjects.

LOGO instructions

The inconclusive results of early LOGO studies on problem solving, and other areas, have prompted many researchers to examine more closely the types of LOGO instructions provided for learners during the learning of LOGO programming. In particular, it has been advanced that the use of more explicit instructions in problem solving might be facilitative in the development of problem solving skills and the transfer of these skills to other contexts (Leron, 1985; Au, Horton & Ryba, 1987).

Recent LOGO studies since the mid-1980's that incorporated explicit instructions in problem solving (eg Lehrer & Smith, 1986; Miller & Emihovich, 1986; Au & Leung, 1991; Ortiz & MacGregor, 1991; Swan, 1991) and metacognitive training (eg Clements, 1987b; Clements, 1990) have clearly highlighted the potential benefits of such exploitation of instructional conditions in the improvement and transfer of problem solving skills of learners. As the issue of LOGO instruction is central to the present investigation, it will be discussed in more detail in a later section of this chapter. The following section will examine those

studies that investigated the relationship between LOGO programming and metacognition.

LOGO programming and metacognitive skills

Within the field of educational computing, there has been an increasing attention on the development of higher order thinking skills (eg Anderson & Ryba, 1990; Au & Bruce, 1990; Lai, 1990), as well as the link between computer science and metacognitive functioning (Haller, Child & Walberg, 1988; Salomon, Perkins & Globerson, 1991). Papert advocated in his seminal work Mindstorms that LOGO could be used to teach learners to think about thinking (Papert, 1980). Some researchers have also argued that the characteristics of the LOGO language could assist learners to develop metacognitive skills (cf Larivée *et al*, 1988). As a result, a number of studies have focused on the relationship between LOGO programming and metacognition since the middle of 1980's. A main aim of these studies was to establish whether the experience of LOGO programming could enhance the development of metacognitive skills, and how such experience might facilitate the improvement of problem solving. The results of these studies have been rather encouraging.

One of the earliest study in this area by Clements and Gullo (1984) used Markman's test to evaluate their subjects' monitoring and evaluation of their own cognitive processes. It was found that the LOGO group outperformed the control in both tasks of the test. A follow up study by Clements (1986) using Sternberg's theory of metacomponents observed that subjects of the LOGO group had significant improvement in their metacomponents of problem solving, and comprehension. Based on a similar theoretical model, some more recent studies by Clements and his colleagues (Clements, 1987b, 1990; Clements and Nastasi, 1988) observed that students, after learning to program with LOGO, improved significantly on their abilities to (i) apply metacognitive skills in problem solving; and (ii) solve the problems.

When comparing LOGO with another software environment (using computer assisted instruction software) and a control group with 39 first grade students,

Lehrer and Randle (1987) found that the LOGO environment, when compared to other conditions, was most facilitative in the development of metacognition measured by problem representation, comprehension monitoring and integration of old and new information, as well as more durable problem solving efficiency.

While these studies have demonstrated that LOGO experience might be facilitative in the development of metacognition, it was unclear whether the improvement in metacognition was due to the medium (the LOGO language) or instructional conditions.

The study by Lehrer, Guckenberg and Lee (1988) addressed this issue by contrasting a number of instructional conditions (focus on programming strategies or geometric concepts) while teaching LOGO programming to their 45 subjects who were divided into three groups. The results of this study suggest that differences in instruction constituted the most significant factor in the development of metacognition. In this instance, the instructional method which required subjects to compare and integrate explicitly old and new information yielded the most significant results.

In examining these studies on LOGO programming and metacognition, it is apparent that the metacognition of the subjects improved after learning to program with LOGO. Also, the performance in problem solving was also enhanced. Future studies would benefit though by clearly outlining the effects of instructions and how instructions might increase the opportunities for the development of metacognition. As well, efforts will need to be made to establish how the improvement in metacognition would benefit the development of problem solving skills.

In summary, having reviewed a representative sample of LOGO studies according to their outcome variables, a number of concerns emerged. The first and most important of these is the type of LOGO treatment offered to the subjects. It has become quite clear that the type of LOGO treatment played a significant role in determining the outcomes of research in LOGO. Other concerns include the selection of sample and the type of research methods and analyses used. The following sections will examine these concerns in more details.

LOGO treatment

It has been suggested by many educational computing researchers that the issue of treatment is a crucial concern in programming language and problem solving research (cf Thomas, 1986; Palumbo, 1990; Seidman, 1989-90; Grandgenett & Thompson, 1991; Au, 1992a). Based on this premise, it is reasonable to expect that the type of LOGO instruction and length of treatment would play a vital role in determining the outcomes of programming instructions. The issue of LOGO treatment, in particular, the type of LOGO instruction, has been at the centre of debate in LOGO research, especially during the 1980's when researchers started to examine how various instructional variables might influence the outcomes of programming instructions.

However, in reviewing the various studies on LOGO programming, it is apparent that there is a lack of clarity in the meaning of "LOGO programming" and the level of mastery attained by the subjects. Some of the reports on LOGO research either provided little or no description of the type of "LOGO treatment" for the subjects (eg Bradley, 1985; Lehrer, Harchham, Archer & Pruzek, 1986; Gallini, 1987; Schibeci, 1990). In these reports, it was often accounted that subjects learned LOGO programming, or were exposed to some LOGO experience, for a certain period of time, without specifying what that experience was, and how it might have affected the outcome variables. Of the reports that delineated the kinds of LOGO experience provided for the subjects, often either one of the following pedagogical approaches was used. Some researchers adopted the use of a self-discovery learning environment, devoid of any teacher intervention, while others considered carefully constructed teacher mediation as important in the learning of LOGO. Appendix 1 presents a summary of a sample of studies reviewed according to the pedagogical approaches used in these studies. About two thirds of the studies either used the former approach or did not describe the pedagogical approach in their studies, while the remainder used the latter.

LOGO instructions

The types of instructions used in LOGO research reflect a much wider concern in pedagogy. Some educators have supported a self-discovery model in

learning. This model is often based on the Piagetian model of knowledge acquisition, which suggests that knowledge can best be acquired through self-discovery learning (Piaget, 1976). In other words, learners learn best by being placed in an environment where they can interact and participate in their own self-directed manner. A number of LOGO researchers, notably Seymour Papert and his colleagues at the Massachusetts Institute of Technology, have adopted this model in their creation of a LOGO learning environment. Following this model, the experience provided for the learners was of a spontaneous and undirected manner.

This approach has also been adopted by a number of researchers in their studies (see Appendix 1). However, this type of research often failed to produce the anticipated cognitive effects espoused by Papert and others (cf Palumbo, 1990). Therefore, at the conclusion of some of these studies, many researchers (e.g Pea & Kurland, 1987; Schaefer & Sprigle, 1988; Thompson & Chen Wang, 1988) suggested that it would be important to consider conscious and careful intervention strategies.

Other LOGO researchers have adopted a more structured approach in programming instruction with carefully designed mediation. They believe that LOGO programming alone cannot achieve what Papert has claimed. Rather, if the learners were to develop some form of cognitive and problem solving skills, then some type of deliberate intervention must be implemented. This approach is clearly exemplified in some of the earlier research conducted at the University of Edinburgh (eg Howe, O'Shea & Plane, 1980; Finlayson, 1983, 1985). Recently, there has been an increasing number of research that employed a more structured pedagogical approach in LOGO instructions, eg, those conducted at the University of Ohio (Clements & Gullo, 1984; Clements, 1986c; Clements & Nastasi, 1988; Clements, 1990; Nastasi, Clements & Battista, 1990), and many others (Lehrer, Guckenberg, & Lee, 1988; Au & Leung, 1991; Heller, 1991; Ortiz & MacGregor, 1991; Swan, 1991).

One of the important factors that clearly underpins the different types of intervention strategies in LOGO instructions is the roles that teachers play in a LOGO environment (eg Lehrer & Smith, 1986; Krendl & Lieberman, 1988). There has been a continuing debate about the role of teachers in LOGO instruction but

until recently, little empirical research has been conducted to offer teachers guidelines on the best ways to use LOGO in the classrooms (Khayrallah & van den Meiraker, 1987).

This debate has been further complicated by Papert's own work. In his book Mindstorms, Papert (1980) advocates "learning without curriculum", and yet he also argues that the teacher should act as an anthropologist and support the students as they build their own intellectual structures. Often, one of the central issues in the consideration of "LOGO instruction" is to resolve Papert's seemingly conflicting notion over the teacher's role in a discovery learning environment in order for children to make the cognitive gains through the use of LOGO as outlined in Mindstorms.

In the deliberation of the type of LOGO instructions provided for the subjects, researchers were often confronted by questions such as "What kind of teacher support should be offered so that on one hand, it does not violate Papert's 'learning without curriculum', but on the other hand, must be such that the learner is 'supported as they build their own intellectual structures'?"

"Is a pure discovery learning pedagogy the approach Papert intends to be adopted by teachers when using LOGO?" One gets the impression that it has never been Papert's intention for the teacher to adopt such an approach. In defining the role of the teacher as an anthropologist, Papert intends for educators to understand and work with materials which are relevant for facilitating cognitive development of the learners. Such facilitation can only take place through meaningful intervention, where the child is guided towards situations in which self-discovery can take place and is assisted to articulate problems, develop ideas, and perhaps, the most important of all, to reflect on their own thinking (Papert, 1980). What Papert fails to do though, is to prescribe specifically forms in which this intervention should occur.

It was perhaps because of this lack of clarity and direction of how LOGO should be used that has resulted in many researchers adopting a narrow interpretation of Papert's discovery-learning approach which is devoid of any teacher intervention.

Pea (1983), Pea & Kurland (1984c), Feurzeig (1986) and many other researchers, as a result of their findings, doubt the attainment of the LOGO ideal through a pure discovery-learning pedagogy. Instead, they indicate a need for educators to provide instructional guidance, so as to help learners develop advanced thinking skills, and to make them aware of the broad range of problem domains to which might be applied. For instance, Pea (1983) states that,

> *It is my hunch that wherever we see children using LOGO in the ways in its designers hoped; and learning new thinking and problem solving skills, it is because someone has provided guidance, support and ideas for how the language could be used. They will have pointed the ways through examples, rules, and help in writing programs and discussing powerful ideas. (p. 7)*

Pea (1983) argues that to call such assistance "learning without curriculum", as defined by Papert, would be a gross misinterpretation of what constitutes curriculum. Instead, Pea & Kurland (1984c) make recommendations which point the educator in the direction of creating a LOGO culture in which deliberate effort is made to bring the thinking process up for conscious scrutiny and to bridge programming skills with other domain interests.

Similarly, Feurzeig (1986), while examining the concept of a LOGO microworld, advances the view that

> *without the aid of a teacher, many children do not learn in a LOGO microworld. They are not able to set their own goals, to find effective methods of thinking about problems, or to acquire the skills of exploration, conjecture and inference ...Skilled teachers overcome these deficits by providing the guidance and support that make microworld experiences productive for their students. (p. 45)*

As well, Leron (1985) joins other educators in endorsing the need for educator intervention. He proposes a quasi-Piagetian approach to the use of LOGO which *"represents a deliberate action to trade off some of the freedom inherent in Piagetian learning for a deeper understanding of the ideas behind the programming activity"* (Leron, 1985:28). For instance, Leron (1985) suggests the use of a study

guide which has two major intended benefits. Firstly, that of working toward child independence from the teacher; and secondly, to provide the teacher with suitable material so as to aid the implementation of the LOGO philosophy as enunciated by Papert.

Research by Clements and Gullo (1984), Clements (1986c), Gorman and Bourne (1983), Lehrer and Randle (1987), Miller and Emihovich (1986), Au, Horton and Ryba (1987), Cohen (1990), and Au and Leung (1991) lend further support to the importance of teacher intervention. For instance, Gorman & Bourne raised the question *"How might curricula be designed...?"* (1983:167) to assist students to improve in combinatorial reasoning. Clements and his colleagues even suggest some specific questions which the teacher could use to encourage children to reflect on their own thinking (Clements & Gullo, 1984; Clements, 1986c). Au, Horton and Ryba (1987) have proposed a LOGO environment checklist against which teachers could use to check the type of programming environment they provide for their students by focusing on the development of problem solving skills.

Other researchers (Clarke & Chambers, 1984b; Nolan & Ryba, 1986) have also argued for the importance of making explicit to the learners the type of problem solving processes which will facilitate cognitive gains and the transfer of learning. For instance, Nolan and Ryba have identified six thinking skills that could be developed via the learning of LOGO in a hierarchy of nine levels of increasing difficulty. These researchers advanced that it is important to foster the development of these thinking skills at each one of these nine levels. In other words, what the learners acquire is not only the programming language *per se*, but also the various problem solving skills required in computer programming.

This view is also shared by Lehrer and Randle (1987), and Miller and Emihovich (1986), who suggest the use of mediated learning by using teacher scaffolding techniques, similar to those advocated by metacognitive theorists. Researchers such as Clements (1986c), Clements and Gullo (1984), and Au and Leung (1991) have called for the incorporation of metacognitive training in LOGO programming in order to facilitate the transfer of learning to other knowledge domains.

In the previous discussion, the importance of a meaningful and deliberate teacher intervention in a LOGO environment has been highlighted. If teacher intervention is to be a crucial element in determining whether children can make cognitive gains from the use of LOGO, what then is required by researchers and educators is specific guidelines and material on how LOGO could be used by them to develop thinking processes and powerful ideas in children (eg Clarke & Chambers, 1984b; Nolan & Ryba, 1986). Such points of consideration must relate specifically to how the teacher fulfils the role of an "anthropologist" as described by Papert (1980) and, in turn, draws the relevant materials from the environment in order to support children's intellectual development. As well, such consideration must be underlined by sound theories of problem solving. As this issue is central to the present study, it will be addressed in more details in the next chapter where various problem solving theories and training strategies will be discussed.

Length of treatment

Another related factor that could explain the discrepancies of results with LOGO research is the length of exposure to the LOGO language, in other words, the duration of "LOGO treatment". In examining the various studies related to LOGO programming, it is apparent that there has been a large variation in the length of LOGO treatment for the subjects.

Some studies provided rather short "LOGO treatment" for the subjects. In a study by Cuneo (1986a), only three to six 30 minute sessions were provided for the subjects. A study by Campbell, Fein, Scholnick, Schwartz and Frank (1986) gave their subjects a total of 50 - 60 minutes of LOGO instructions. Similarly, many studies only provided a fairly limited amount of LOGO training to their subjects, eg Lee and Lehrer (1988) - 1.5 hours each week for eight weeks; Williamson and Silvern (1986) - one hour each day for 10 days; Miller and Emihovich (1986) - three weeks of training; Degelman, Free, Scarlato, Blackburn and Golden (1986) - less than seven hours of programming; Ortiz and MacGregor (1991) - less than nine hours of learning in total.

On the other hand, there were other studies which provided much more extensive LOGO training for their subjects. For instance, in a study of severely

handicapped children (Weir, 1981), subjects learned to program LOGO over a period of two years. When working with average classroom students, Noss (1987a) provided LOGO instruction for his subjects 75 minutes per week over one full school year.

Arguably, this extensive variation in the duration of LOGO treatment had contributed to the discrepancies in the results of these research. While it is difficult to suggest what constitutes sufficient or optimal LOGO training for the students, it is reasonable to contend that the length of LOGO treatment is directly related to the level of mastery of LOGO programming, which might in turn influence the measures of the outcome variables such as problem solving skills development and transfer.

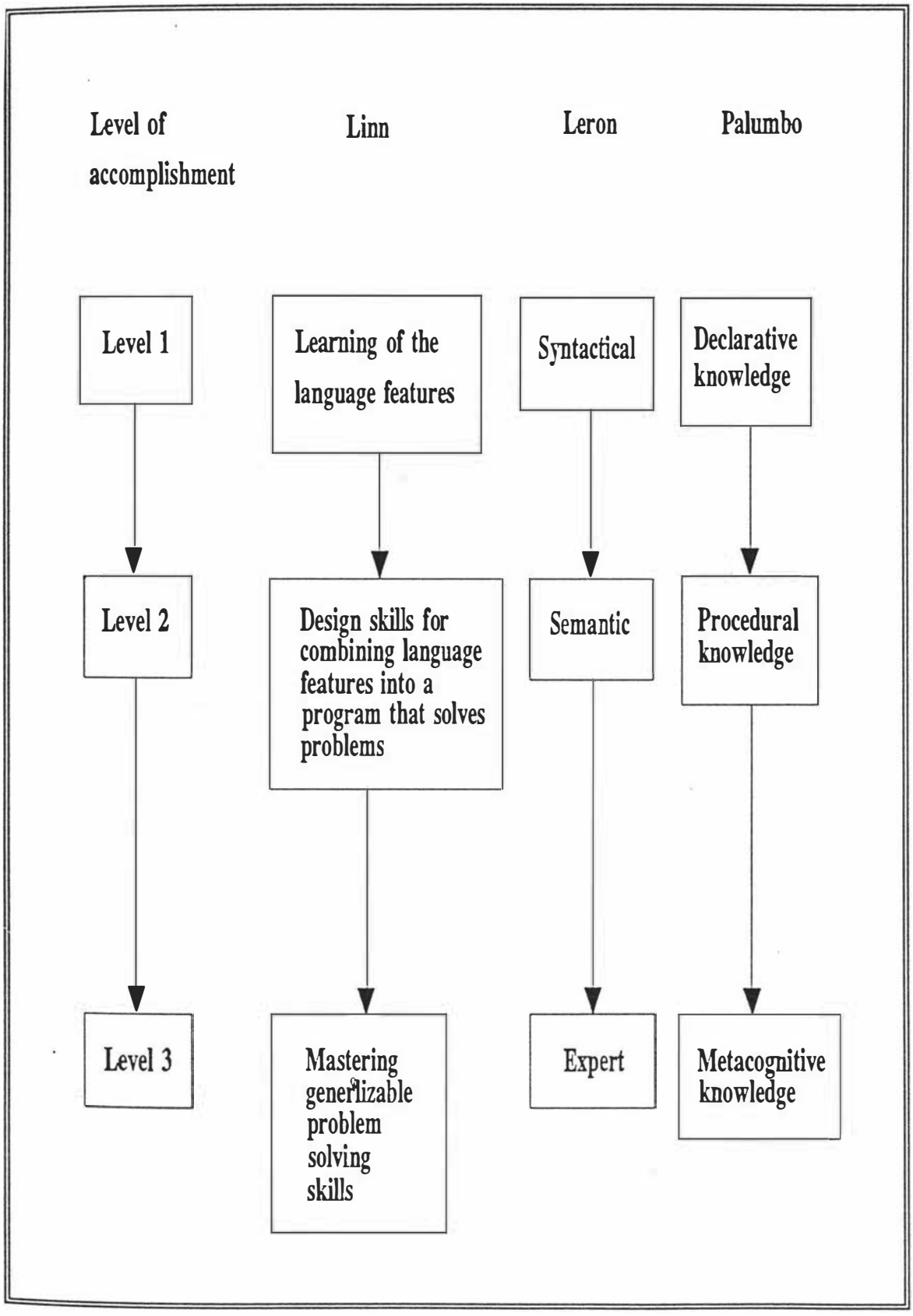
Researchers such as Pea and Kurland (1987), Linn (1985), Dalbey and Linn (1985), Leron (1985), Khayrallah & Meiraker (1987), Mayer and Fay (1987), and Palumbo (1990) have repeatedly called for the examination of the relationship between the level of mastery of programming and the cognitive consequences of programming learning including the transfer of learning to other knowledge domains. Linn (1985), Leron (1985), and Palumbo (1990) have outlined a chain of cognitive accomplishment through the learning of programming as identified in Figure 4.1 overleaf.

Linn and Dalbey (1985) describe an ideal chain of cognitive accomplishment which consists of three major components. They assert that learners of programming need to attain the third level of accomplishment before they are capable of using the general problem solving skills acquired through programming in other domains. This chain roughly parallels the three different levels of programming attainment (syntactical, semantic, and expert) espoused by Leron (1985) who claims that subjects in LOGO research normally only reach the second level. As a result, these learners are not in a position to apply whatever problem solving skills that they might have acquired in a programming context to others.

A similar chain has also been advocated by Palumbo (1990) when he attempted to integrate programming instructions and information processing theories. This chain comprises three components, viz, declarative knowledge, procedural knowledge, and metacognitive knowledge. Palumbo argues that students of programming need to reach the third level before they are capable of applying

Figure 4.1

Chain of cognitive accomplishment



appropriate and effective problem solving strategies to solve a variety of problems in different contexts.

These researchers, *inter alia*, have argued that there are stages that a student must go through, and that the level of programming abilities a student has mastered would be a predictor of the kinds of concepts and skills that the student will transfer beyond programming. In particular, Pea and Kurland (1984a) and Dalbey and Linn (1985) have argued that cognitive gains are related to the mastery of programming. In other words, learning will only be transferred to other knowledge domains if certain level of mastery has been attained. Therefore, it would be unreasonable to expect transfer to take place after only a short duration of learning with LOGO.

While it is quite debatable whether a learner might not encounter metacognitive experience or develop metacognitive knowledge while working at the first and the second levels, it is clear that from these models that the higher the level reached by the learners, the more probable that transfer of learning would occur. In order to increase the likelihood of transfer of learning, it would be advisable to increase the incidence of metacognitive experience even during the early stages of learning of LOGO. Also, the direct teaching of metacognitive knowledge may be another way facilitate transfer of learning. The issue of training in problem solving and metacognition will be further discussed in the next chapter.

Selection of sample

In reviewing a substantial number of research in LOGO, there emerged a number of concerns related to the selection of sample. These concerns are: (i) small number of subjects used in some studies; (ii) the non-random selection of subjects; and (iii) the appropriateness of selection of subjects.

Apart from many anecdotal studies which often observed only a very small number of students, some studies, owing to their nature, tended to use small number of subjects. For instance, studies which investigated students of physical handicap, could obviously obtain a limited number of subjects (eg Weir, 1981). There are a number of studies which used fairly small numbers of subjects in typical experimental studies. For example, Gorman and Bourne (1983) had 15 subjects in their study divided into two groups (one with 10 and the other with five students).

Similarly, Evans (1984) used 15 subjects (eight in the experimental group and seven in the control); McAllister (1985) studied eight subjects; Zelman (1985) observed only four subjects; Miller and Emihovich (1986) worked with 14 subjects (seven in experimental group and seven in control). The use of small number of subjects in such studies clearly warrant caution in interpreting their results.

A second concern is the non-random selection of subjects in some of these studies. Owing to possible ethical and logistical constraints, some studies did not select their subjects randomly. For instance, Noss (1987a) worked with students from five different classrooms without allocating the students randomly into experimental and control groups. On the other hand, Schaefer and Sprigle (1988) were only able to study 20 children enroled in a university laboratory school, hardly a random sample. The non-random selection of subjects and allocation to groups, and the lack of statistical control to compensate, clearly reflects a concern in the use of experimental studies.

A third concern relates to the selection of samples of the appropriate age range for optimal training and benefit. Current studies on LOGO have selected samples ranging from pre-school to adults. A question that needs be addressed is: what age group of subjects are likely to benefit from programming instruction in LOGO?

Some researchers have argued that students at the concrete operational stage of cognitive development are unlikely to improve significantly in higher order thinking skills, and hence unlikely to benefit from problem solving skills instruction through the learning of programming (cf Piaget, 1977; Palumbo, 1990). However, such argument apparently runs contrary to what Papert has advocated in Mindstorms - that LOGO can concretize the abstract - implying that children's transition from the concrete operational stage to formal operational thinking could be expedited by the learning of LOGO programming.

One of the reasons that has been advanced was that younger children have not had sufficient exposure to a variety of problem solving domains, therefore, it would be unlikely for transfer of learning to occur. If that is the case, then future research with LOGO need to address such issue by providing students with more

exposure in different problem solving situations in order to increase the likelihood of transfer of problem solving skills.

Research methods and analyses

The last major issue that emerged from a review of the literature relates to the research design and analyses used in a number of LOGO studies. Three major concerns can be identified in this section: (i) use of control group; (ii) use of traditional experimental design; and (iii) equivalence of training. The use of traditional experimental design has been discussed in an earlier section of this chapter and hence will not be repeated here. The following section will examine the other two concerns.

Use of control group

A number of studies, for instance, did not employ control groups in their investigation. For instance, Carmichael *et al* observed more than 400 students in 18 different classrooms and concluded that LOGO could be a powerful medium for developing problem solving skills. However, there was really no basis for making such a comparison. More serious concerns could be found in studies that employed more traditional experimental design. In the study by Mayer and Fay (1987), a conclusion was reached that LOGO programming could modestly influence children's thinking in areas of similar to those involved in programming. Nonetheless, without the use of a control group, this study could not ascertain whether the observations were the results of LOGO's influence only. Similar problems could be found in a number of studies, for example, Finlayson (1983), Schaefer and Sprigle (1988), and Schibeci (1990).

Equivalence Of Training

Another major problem inherent with many of the studies reviewed relates to the equivalence of training provided for the various groups of subjects. Quite often, researchers would provide "LOGO training" for the experimental groups and "other methods" such as CAI or "normal mathematic classes" for the controls. This type of design often posed difficulty in determining the equivalence of the various kinds

of training for the subjects. A number of questions would arise from such considerations, for instance, "why would certain type of CAI learning influence the logical reasoning of the subjects?", "Is LOGO programming superior to other types of programming in the facilitation of cognitive gains and transfer of learning?", "Does mediated learning in the teaching of LOGO better than the traditional method of instruction in LOGO?". Therefore, careful consideration must be given to the use of comparison groups in future studies.

It is in light of these consideration that the present study will attempt to exercise tighter control in the provision of equivalence of training. To begin with, two LOGO groups will be used - a process-oriented group and a content-oriented group. In this way, the possible differences in the two methods of instruction will be able to be examined. Also, a BASIC group will be used which will enable this study to compare the effects of two different programming languages.

Summary

This chapter has reviewed a number of LOGO studies and has highlighted some of the major concerns of LOGO research. One of the main concerns is that the teaching and learning of LOGO has been interpreted differently by different investigators. In some studies, a purely self-discovery approach has been adopted whereas on the other hand, a number of studies has used various forms of teacher intervention. Although studies that utilized teacher intervention seemed to yield more consistent results in assisting learners in their cognitive development, the effectiveness of instructional approaches used in the teaching of LOGO still remained unclear. In particular, much work remain to be done in identifying instructional approaches that may assist the transfer of problem solving skills to other domains.

The second major concern lies in the measurement of problem solving skills and their transfer to other domains. Few studies have given consideration of the need to identify the type of transfer that may take place after learning LOGO programming, and how such transfer could be measured.

The third major concern relates to the social interactions of learners while learning to program with LOGO. Few studies have attempted to measure nor

control the type of social interactions that might have influenced the problem solving skills of learners.

In order to address these three major issues, it is important to turn to the literature in problem solving and metacognition for further guidance in the development of problem solving skills in relation to the learning of LOGO. It is in light of these considerations that this thesis now turns to an examination of the literature in problem solving and metacognition, as well as their relationship with LOGO programming.

CHAPTER FIVE

PROBLEM SOLVING AND COMPUTER PROGRAMMING: INSTRUCTIONAL IMPLICATIONS

This chapter examines the research on problem solving and its implications on the training of problem solving skills. The first section provides an overview of problem solving and an outline of the various historical approaches in the study of problem solving. The second section analyses contemporary models and issues in the training of problem solving skills, in particular, in the context of learning of programming and LOGO. In the last section of this chapter the research questions for the present study are tendered.

Overview

Problem solving, it has been suggested, is closely related to the notion of intelligence (cf Resnick & Glaser, 1976; Rowe, 1985). For instance, Resnick and Glaser (1976) argues that a major aspect of intelligence is the ability to solve problems, and that the analysis of problem solving behaviour constitutes a means of specifying many of the psychological processes that intelligence comprises. Wagner and Sternberg (1984) have proposed three alternative conceptions of intelligence: psychometric, Piagetian, and information processing. Their analyses demonstrate that problem solving plays a central part in the development and measurement of intelligence irrespective of the perspectives adopted. Indeed, Sternberg (1982a) observes that problem solving and intelligence are so closely interrelated that it is often difficult to make a distinction between these two concepts.

Moreover, there are also close relationships between problem solving, intelligence, learning, thinking, and cognitive strategies. Their close relationship could be established from a review of the literature on training of problem solving (eg Frederiksen, 1984), learning abilities (eg Derry & Murphy, 1986), intellectual skills (eg Wagner & Sternberg, 1984), thinking skills (Nickerson, 1988-89), and cognitive strategies (eg McCormick, Miller & Pressley, 1989). At a conceptual level, it could be argued that problem solving pervades all areas of learning and

cognitive activities, eg reading, writing, and thinking (Frederiksen, 1984). For instance, Gagné (1966) points out that the ability to formulate situationally relevant learning strategies is a form of strategic problem solving capability. At a practical level in terms of skills to be trained, there is a vast degree of overlap. Nisbet and Shucksmith (1986:28), based on the work of Feuerstein, Rand, Hoffman, Hoffman and Miller (1979), and Butterfield and Belmont (1977), listed a number of learning strategies (cf Table 5.1). These strategies bear a high degree of similarity to the skills often used in the training of problem solving including skills such as problem representation, planning, analysis, predicting, monitoring and evaluating (eg Simon, 1980; Baker & Brown, 1984; Mayer, 1984; Baron, 1985). It should be noted though that there is some confusion in the literature about the distinction between strategy and skills. For instance, strategies are generally considered as composite methods comprising many skills, however, planning and monitoring are commonly referred to as metacognitive skills.

Table 5.1

A list of common learning strategies

a.	Asking questions	defining hypotheses, establishing aims and parameters of a task, discovering audience, relating task to previous work, etc.
b.	Planning	deciding on tactics and timetables, reduction of task or problem into components: what physical or mental skills are necessary?
c.	Monitoring	continuous attempt to match efforts, answers and discoveries to initial questions or purposes.
d.	Checking	preliminary assessment of performance and results.
e.	Revising	may be simple re-drafting or re-calculation or may involve setting or revised goals.
f.	Self-testing	final self-assessment both of results and performance on task.

Therefore, when discussing problem solving and the training of relevant skills in this chapter, apart from examining those literature that focus directly on problem solving, it is necessary to draw upon the relevant literature on training of learning abilities, intellectual skills, thinking skills and cognitive strategies.

The study of problem solving

The basis for problem solving behaviour - problem situations - have often been characterized in a number of ways. Johnson (1955), for instance, has suggested that a problem situation exists when an individual's first goal-directed response is unrewarding. Köhler (1927) has maintained that a problem situation exists when an individual must take a detour to reach a goal. Vinacke (1952) has taken a similar position, claiming that a problem situation exists when there is an "obstacle" to overcome. Woodworth and Schlosberg (1954) have argued that a problem situation exists when an individual has a goal, but without a clear or well-learned route to the goal. Still, other definitions have been proposed by Humphrey (1951), Maltzman (1955), Ray (1955), Underwood (1952), and van de Geer (1957). The characteristics most frequently mentioned are the integration and re-organization of past experience in the discovery of correct responses. This is similar to the views of Newell, Shaw and Simon (1960), who consider a genuine problem solving process involves the repeated use of available information to initiate exploration, which discloses, in turn, more information until a way to attain the solution is finally discovered. A related view of Miller *et al* (1960) states that solving a problem is a matter of turning up a lot of likely hypotheses until either one satisfies the test or the stop-rule is applied. Resnick and Glaser (1976) suggest that the term "problem" refers to a situation in which an individual is called to perform a new task although processes or knowledge available can be used for solution.

While it seems difficult to reach consensus on a definition of problem solving from this plethora of descriptions, one can discern some general characteristics of problem solving. First, problem solving is goal-directed. Therefore, one of the very first steps in solving a problem is to identify the goal to be reached. Second, it involves the search for a possible solution in order to reach the goal state, or, the search of a problem space which consists of physical states or knowledge states

(Newell & Simon, 1972; Anderson, 1980). In the process of searching for a solution, different problem solving skills or heuristics such as functional analysis, means-end analysis, search, and planning can be employed (cf Newell, Shaw & Simon, 1980; Klahr & Robinson, 1981).

In the study of problem solving behaviours and how problem solving skills might be improved, three distinct traditions could be identified: (i) Gestalt; (ii) behavioural; and (iii) information processing models.

(i) Gestalt psychologists (eg Duncker, 1945; Köhler, 1927; Wertheimer, 1959), the earliest group of psychologists who studied problem solving using an experimental approach in the 1930's and the 1940's, conceptualized problem solving as a process of cognitive organization. In their view, problems were analysed as situations for which cognitive representations have gaps or inconsistencies, and problem solving was the process to organize the situation to provide a good structure, including satisfactory achievement of the problem goal (Greeno, 1978b). Problem solving, hence, within the Gestalt tradition, focuses on the restructuring of a problem so that it becomes soluble (Resnick & Glaser, 1976). Emphasis in the Gestalt analyses is on the insightful nature of the process, and the way in which solution follows almost immediately upon recognition of a new form of the problem. One of the classic experiment within this tradition was the study by Köhler (1927) who observed how a chimpanzee was able to get hold of the bananas outside a cage after achieving some insightful discovery. However, Gestalt psychologists have failed to provide a detail analysis of how insight might occur during problem solving, and consequently, suggestion as to how learners might improve their problem solving skills.

(ii) In the analyses by behavioural and associationist psychologists (eg Maltzman, 1955), a problem occurs when a) the response needed to achieve some goal is less strong than other responses; or b) several responses are required and it is unlikely that they all will be performed. Behavioural analyses emphasized the need for problem solvers to perform a variety of responses and to raise the probabilities of unusual responses, since by definition, successful problem solving depends on giving responses that are relatively improbable. One of the major shortcomings with such analysis is that the underlying processes associated with problem solving were never

made explicit. As a result, little progress was made regarding the training of problem solving skills.

(iii) Studies by cognitive psychologists based on an information processing perspective since the 1970's (e.g Newell & Simon, 1972), perhaps represent the most current systematic attempt in the study of problem solving behaviours to date. The information processing approach often draws upon parallels between the cognitive processes of the mind and the operation of a digital computer. Using a metaphor similar to the structure and processing of a computer, information processing psychologists postulate that the mind receives information and data from the environment, processes the information in a central processor, then stores it in memory and/or provides the necessary output when required. The focus of this approach is on the measurement of how information (input) can be processed to provide output (Rowe, 1988b).

Contemporary information processing models have diverged into two classes (cf Siegler, 1983) - those that focus on the information processing system *per se*, examining issues such as storage and memory (eg Atkinson & Shiffrin, 1968); and those that focus on the interaction between information processing and the task environment (eg Newell & Simon, 1972). A high percentage of contemporary work on problem solving and its training since the 1970's has been influenced by theories in the latter category, in particular, that by Newell and Simon.

In contrast to behaviourist and gestalt approaches, the major advantage of adopting information-processing theory in studying problem solving behaviours, is that performance in problem solving is analysed in detail, and theoretical interpretations include specific assumptions about the component cognitive processes involved in the performance. Information processing psychologists have taken up the detailed analysis of problem solving that was begun by Gestalt psychologists, and this is being done in much more rigorous and systematic ways than were characteristic of Gestalt theory (cf Baron, 1985; Sternberg, 1982b; Pressley, 1986). But the analyses have been relatively specialized, concerned with the details of performance in individual tasks and often by individual subjects (Scandura, 1977; Symons, Snyder, Cariglia-Bull, & Pressley, 1989). Information-processing theorists have provided strong concepts for use in analysing specific tasks but there have been

concerns in the development of a coherent body of theory made up of general psychological principles that explain performance in broad classes of problems and strategies in the development of general problem solving skills.

These concerns have led to the emergence of a body of literature over the last ten years which examines the role of metacognition - often referred to as thinking about thinking and self-regulation - in the acquisition and development of general problem solving skills (Flavell, 1979; Baker & Brown, 1984). A number of researchers have concluded that metacognition plays an important role in oral communication of information, oral comprehension, reading comprehension, writing, language acquisition, attention, memory, problem-solving social cognition, and various types of self-control and self-instruction (eg Flavell, 1979; Reeve & Brown, 1985; Borko, Livingston, & Shavelson, 1990; Paris & Winograd, 1990b; Derry, 1990; Chan, 1991). A more detailed discussion of metacognition and its role in problem solving will be presented later in this chapter.

Training of problem solving skills

In the search for ways to enhance problem solving skills, a number of training models and theories have been advanced. Among these models and theories, three distinct perspectives of training can be discerned (Derry & Murphy, 1986). These perspectives highlight a number of issues concerning the training of problem solving skills: (i) domain-specific versus domain-general problem solving strategies; (ii) transfer of learning from one context to another; and (iii) the provision of direct instruction in problem solving training. These issues will be discussed further in the context of these three perspectives.

Perspectives on training

The first category represents an attempt to increase the impact of conventional instruction through greater and better efforts to improve students' problem solving and learning abilities by focussing on particular skills that are domain-specific. For instance, Gagné and Briggs (1974) suggest that five types of skills can be trained. These skills include: discrimination, concrete concepts, defined concepts, rules, and higher order rules. Based on this perspective, learners

need to acquire these skills through prolonged practice in order for improvement in problem solving to occur.

The major shortcoming of this approach is that the main purpose of instruction focuses on the achievement of "terminal objective" (Gagné & Briggs, 1974) instead of on the development of generalizable problem solving skills that can be applied to contexts other than that within which the skills were developed. Moreover, the training within this category does not provide direct instruction as to how specific cognitive process are involved in problem solving. Consequently, although students might attain mastery of problem solving skills within a specific domain through continual practice, any possibility of generalizing these skills to other contexts is incidental and required prolonged practice.

This concern over the transfer of problem solving skills from one context to another has prompted a shift in emphasis from the lower order task performance components as suggested in the training approaches in the first category, to the metacognitive level, that is, the level of executive skills involved in problem solving in general (Wagner & Sternberg, 1984). This change in emphasis is reflected in the training approaches based upon the second perspective, based on Sternberg's process-oriented training; and the third perspectives, based on the training of metacognitive skills.

The second perspective of training owes much to the work of Sternberg and his colleagues (Sternberg, 1982b, 1983; Wagner & Sternberg, 1984) although early cognitive research on problem solving have suggested training approaches that are comparable to some aspects of this approach (cf Polya, 1957; Newell, Shaw & Simon, 1960).

The major impact of this approach comes from a process oriented training approach which attempts to improve the general processing capabilities of the learners that are applicable across domains (Sternberg, 1982b, 1983, 1984, 1985b; Wagner & Sternberg, 1984). This is in direct contrast to the approaches used in the first category (eg Gagné and Briggs, 1974) which focus only on domain-specific strategies. Research by Sternberg and his associates subdivide the training into three components: (i) microcomponents (eg recall strategies, perception speed etc), (ii) macrocomponents (eg note taking and outlining skills), and (iii) metacomponents

(executive control skills such as planning, monitoring, and evaluating one's information processing that enable a person to solve a problem by mobilizing and organizing relevant micro- and macro- components) (Sternberg, 1983).

There are a number of programmes that teach general problem solving skills based on the second approach. Some programmes teach general problem solving skills for pa ticular education settings, for example, the Strategy Intervention Model (SIM) at the University of Kansas (Deshler, Schumaker & Lenz, 1984); or teaching these skills via solving specific problems, eg the Training A ithmetic Problem Solving Skills (TAPS) programme at Florida State University (Derry, Hawkes & Tsai, 1987). Another group of programmes aims to teach general problem solving skills that are applicable in many different contexts. Among other more noted programmes are the Cognitive Research Trust (CoRT) program (de Bono, 1973, 1985) and the Productive Thinking Program (Covington, 1985).

The most important component of training based upon this approach is the development of the metacomponents in problem solving. Sternberg argues that it is the metacomponents, being higher order cognitive skills, which are most important to problem solving and most relevant to the measurement of intelligence. Moreover, the main value of metacomponents is that they can be operationalized and directly observed and measured. For instance, a study of planning behaviour in problem solving by Sternberg indicates that good problem solver tend to spend more time on higher order planning rather than lower order planning (Sternberg, 1981). The importance of the training of the metacomponents is also reflected in the study by Kendall, Borkowski, and Cavanaugh (1980). This study found that subjects who were trained in the use of maintenance and generalization of interrogative strategies were able to transfer these strategies to other problems of a similar nature.

More recent research has lent further support to the benefits of focussing on the metacomponents in problem solving. Dansereau and his associates (1978, 1985) have taught their subjects to use domain-general planning heuristic models successfully in their problem solving. Studies by Baron (1981), Bransford (1984), and Hayes (1981), *inter alia*, have all emphasized plans that include steps similar to the following: (a) analysis and goal identification, (b) planning a strategy, (c)

carrying out the strategy, (d) checking results of the strategy, and (e) modifying the strategy.

Indeed, support for focussing on the metacomponents could also be obtained from some early cognitive research on problem solving. Polya (1957), for instance, suggested a model which consisted of four main steps: understanding a problem, planning, hypothesis testing, and evaluation. Newell, Shaw, and Simon (1980) proposed a General Problem Solving Program, which apart from suggesting the separation of problem content from problem solving technique as a way of increasing the generality of the problem solving skills, suggested the use of generate and test, means-end analysis, planning, identification of recursive nature of problem solving activity and the principle of sub-goal reduction as training strategies to enhance problem solving. Simon (1960) argues that these strategies are true general problem solving mechanisms which should be the core of any instructional programmes that attempt to teach general problem solving skills.

It is the metacomponents in the second training approach that are most closely associated with the third perspective of training approaches - the training of metacognitive strategies. As this approach bears important consideration to the training approaches adopted in this study, it is appropriate to review it in more detail.

Metacognition

The third training perspective is based upon the notion of metacognition. Broadly speaking, metacognition is identified as that body of knowledge and understanding that reflects on cognition itself (cognition about cognition), including knowledge such as a person's knowledge of cognitive processes and states such as memory, attention, perception, knowledge, and inference (Wellman, 1985). In other words, metacognition is that mental activity for which other mental activities become the object of concern and reflection (Yussen, 1985), or the knowledge and regulation of cognition (Armbruster & Brown, 1984).

Numerous theorists have attempted to provide detailed descriptions and delineations of metacognition, as well as how metacognitive training could be

conducted in order to improve problem solving skills. Flavell, one of the leading researchers in this field, describes

> *metacognition as one's knowledge concerning one's own cognitive processes and products or anything related to them, eg, the learning-relevant properties of information or data... Metacognition refers, among other things, to the active monitoring and consequent regulation and orchestration of these processes in relation to the cognitive objects on which they bear, usually in the service of some concrete goal or objective.*
>
> (Flavell, 1976:232)

Furthermore, Flavell makes a distinction between metacognitive knowledge and metacognitive experience. He considers metacognitive knowledge as consisting primarily of knowledge or beliefs about what factors or variables act and interact in what ways to affect the course and outcome of cognitive and cognitive enterprises whereas metacognitive experience are considered as conscious experiences of a cognitive and affective nature that is pertinent to one's intellectual life (Flavell, 1979).

Flavell argues further that it is important to provide training in actions, goals, metacognitive experience and metacognitive knowledge. Based on this model of Flavell, a number of problem solving training strategies could be used: (i) help learners to build a library of problem solving heuristics (actions); (ii) train learners to recognize the goals of the problems (goals); (iii) enhance the frequency and quality of experience that lead to insights about problem solving (metacognitive experience); and (iv) help learners to build a store of information about the utility of problem solving heuristics, including when and how to use them (metacognitive knowledge).

A second major line of research in metacognition comes from Brown and her colleagues (eg Brown, 1978; Campione, Brown & Ferrara, 1983). These researchers, while agreeing with Flavell's definition of metacognition, make a further distinction between metacognitive knowledge and executive control. This distinction is the result of two diverse lines of research within metacognition (Brown, Campione & Day, 1981). The first is concerned with people's knowledge

of their own cognitive resources. The second line of research focus on clusters of activities consisting of the self-regulatory mechanisms used by n active learner during n ongoing attempt to solve problems. These mechanisms include: checking the outcome of any attempt to solve problem, planning one's next move, monitoring the effectiveness of any attempted action, and testing, revising, and evaluating one's str tegies for learning (Brown, 1978, 1982).

In light of the above clusters of activities, these researchers (eg Campione *et al*, 1983) consider it necessary to make a distinction between metacognition nd executive control. The former refers to the knowledge about cognition and the latter to denote the overseeing, management functions. Moreover, they argue that executive control appears more central to notions of intelligence. The premise of their argument is that inducing executive control seems to lead to increased transfer or to more intelligent behaviour. For instance, their research with retarded children demonstrate that inducing executive control does facilitate both immediate response to training and transfer whereas enhancing knowledge about memory does not appear to do so, at least not to the same extent (cf Brown, 1978, 1980).

Various reviews of the literature have consistently pointed out that poor problem solvers lack these executive control strategies (cf Brown, 1980). There is emerging evidence to suggest that metacognitive training, in the form of teaching general problem-solving principles, has been particularly successful in the intellectual performance of children with learning problems (Brown & Campione, 1982; Campione & Brown, 1978; Belmont & Butterfield, 1977; Palincsar & Brown, 1984; Palincsar, 1986; Paris, Newman, & McVey, 1983; Reeve & Brown, 1985; Ellis, Lenz, & Sabornie, 1987a; Chan, Cole & Morris, 1990; Paris & Winograd, 1990a; Chan, 1991). Belmont and Butterfield (1977) reviewed a total of 114 studies on cognitive instruction and found that none of them provided metacognitive training and that none of them reported generalization of training. On the other hand, six of the seven studies reviewed by Belmont, Butterfield and Ferretti (1982) that produced generalized cognition by young and mentally retarded children have instructed some aspect of superordinate processing. Doyle (1983), after reviewing a number of studies on training of various forms of problem solving, concluded that direct instruction in higher level regulatory processes would likely to assist in the

improvement of problem solving. Similar conclusion was reached by Chan (1991) who proceeded to suggest a number of conditions under which generalization of training might occur, including informed training, direct executive control training, and explicit generalization training. These reviews provide support for the proposition that the explicit instruction of superordinate self-management skills and generalization training can assist the development and transfer of problem solving skills.

One of the key features of metacognition appears to relate to consciousness (cf the notion of Flavell on metacognitive experience & the notion of Brown *et al* on executive control). The application of metacognitive strategies to solve a problem represents a conscious effort to, (i) identify problem solving strategies irrespective of the contents of the problem and solution, and (ii) to apply these strategies across different knowledge domains. The implication is that in order to improve problem solving skills, it is beneficial to provide learners with explicit instructions in both problem solving and metacognitive strategies, and how they could be applied to other contexts.

Metacognition also seems to be closely related to the Piagetian notion of equilibration. Piaget (1977) postulates that each organism is an open, active, self-regulating system. The fact that, in healthy children and adolescents in our civilization, this continual mental transformation tends toward order and not toward chaos would indicate - according to this hypothesis - the influence of self-regulating processes such as those involved in a principle of equilibrium. According to Piaget equilibration is a process of increased reflection, a turning inwards or an interiorization of action that changes coordinated external actions into systems of interior, reversible operations (Furth, 1969). The notions of introspection and self-regulation are central to the study of metacognition and that of intelligence. Furthermore, it has been argued that equilibration as the inner regulating factor which in development leads to an increasing dissociation of the general forms of structured behaviour from particular content (Furth, 1969; Wadsworth, 1989). In particular, it is important to the transferral of problem solving strategies from a knowledge-specific domain to more general domains. This argument underpins one of the basic premises in metacognition which suggests the need to develop executive

control strategies that are common to all forms of problem solving irrespective of the content knowledge of a particular problem.

Some major conclusions that could be drawn from the literature on metacognition are that successful problem solving training would need to include: (i) training and practice of domain-specific problem solving skills; (ii) increasing the metacognitive experience of learners during the process of problem solving; (iii) instruction in the orchestration, overseeing, and monitoring of these skills; and (iv) information concerning the significance and outcome of these activities and their range of utility (cf Flavell, 1979; Baker & Brown, 1984; Haller, Child & Walberg, 1988; Chan, 1991).

In summary, the review of the literature on the training of problem solving, learning abilities, intellectual skills, thinking skills and cognitive strategies have clearly pointed to the benefits of the provision of instruction of self-management skills for learners in order to assist them in developing problem solving skills, in particular, in applying these problem solving skills across different knowledge domains. The next section will discuss issues in the training of problem solving in relation to computer programming, in particular, the use of the LOGO language.

LOGO programming and problem solving

The major part of cognitive theories is based on an information processing model of intelligence which in turn owes much to the concepts in artificial intelligence and computer programming. When one talks about metacognitive strategies to solve problems, it is necessary to refer to an analysis of the step-by-step break-down of the whole process. As well, it is also necessary to look at the output, the input and the other resources available to solve the problem, for instance, availability of memory, focus of attention. The advantage to talk about metacognitive strategies (executive control) in terms of computer programs is that terms like memory and strategy can be defined in precisely stated instructions for a computer. Furthermore, the requirement that the programs must work, that is, must be able to solve the problem, provides a guarantee that no steps have been left unspecified.

In general, learning programming is quite similar to the metacognitive steps for problem-solving as proposed by Belmont *et al* (1982) in their self-management model in solving a problem, which are:

1. Decide on a goal;
2. Make a plan to reach the goal;
3. Try the plan;
4. Ask whether the plan worked;
5. Ask whether the plan was actually followed;
6. Ask what was wrong with the plan and then return to step 2.

When applying the model in a programming context, this model could be modified as a plan can be evaluated at two different points. First, the learner can evaluate the plan without using the computer; second, once the learner has ascertained the plan would work, this plan can then be tested with the computer (Figure 5.1).

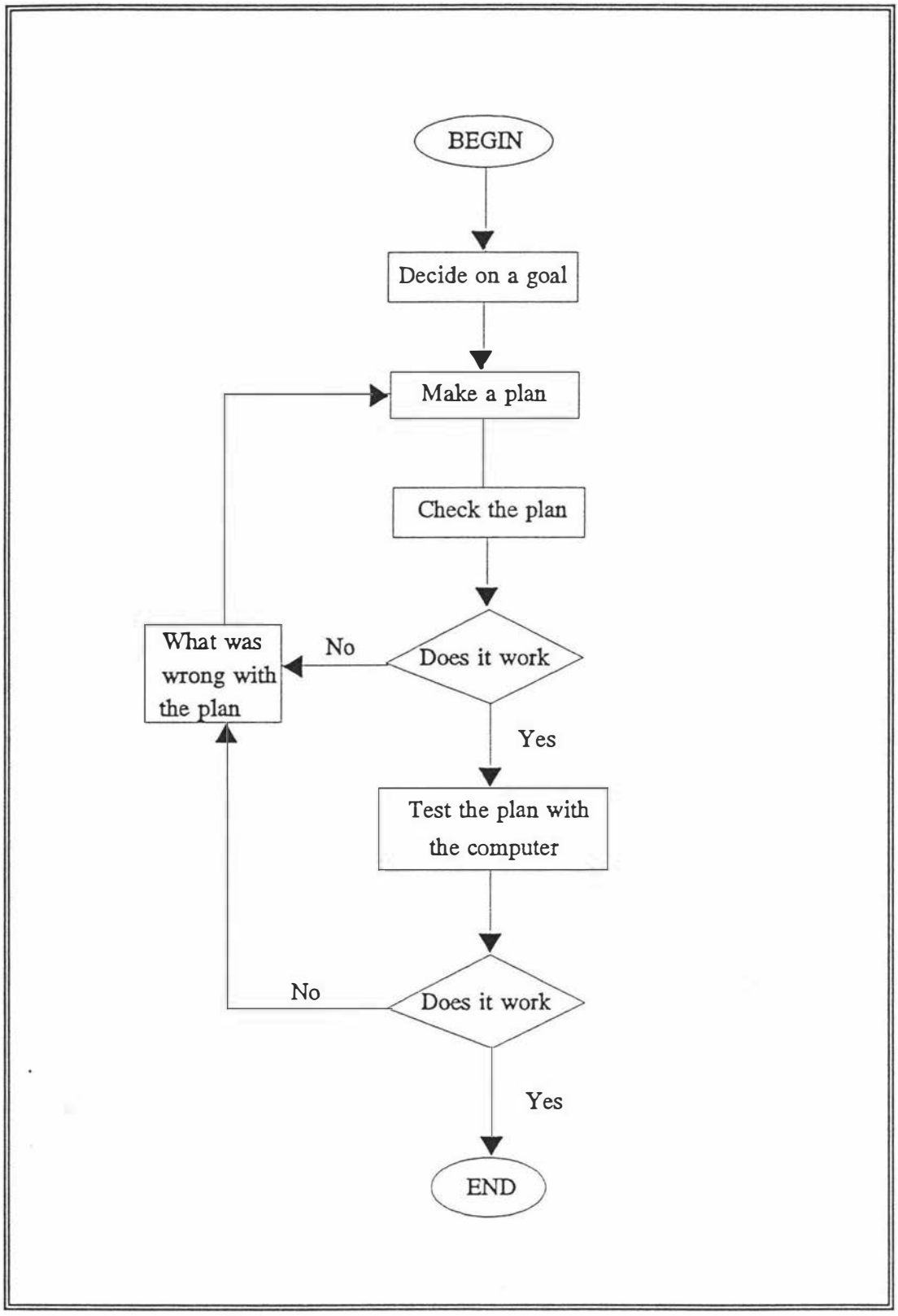
In essence, computer programming provides an excellent medium for problem solving training to take place. On one hand, it is highly interactive, on the other hand, the inherent activities of computer programming facilitates the occurrence of such metacognitive training.

The structured and interactive nature of the LOGO language lends itself particularly suitable to put this self-management model into practice (cf Chapter 3). A number of steps can be involved in the practice - first, breaking down the problem into different sub-problems; second, further subdividing these sub-problems into manageable sub-problems; third, devising a plan for each sub-problem and joining these plans together to form an overall solution for the problem; fourth, carrying out the checking whether the plan worked; and fifth, making necessary corrections.

The use of LOGO to develop problem solving skills has been discussed in detail in the previous chapter. It was noted that one of the major issues involved the transfer of learning of problem solving skills to another context. Therefore, when using LOGO to develop the problem solving skills of the learners, one must also consider the issue of transfer of problem solving skills.

Figure 5.1.

A guide to self-management in solving a problem



Transfer of problem solving skills

A major issue in problem solving research has been that of the transfer of problem solving skills from one context to another. One of the central tenets in the training of problem solving rests on an assumption of transfer. Indeed, this issue is a crucial feature of research in computer programming and problem solving as one of the most persuasive arguments in favour of teaching of programming concerns its potential promotion of generalizable problem solving skills. However, to date, research in LOGO programming and problem solving have yielded mixed results (cf Chapter Four).

How can training with LOGO programming facilitate the transfer of problem solving skills? A corollary to this question is: why did some of the LOGO training did not result in transfer of problem solving skills? Two important issues need be examined - (i) the type of transfer desired; and (ii) conditions under which transfer may take place.

Types of transfer

Research in problem solving have consistently highlighted the distinct differences between expert and novice problem solvers (cf Rohwer & Thomas, 1989). Expert problem solvers are those that possess the factual and declarative knowledge, as well as effective strategic knowledge required to solve a problem. It has been suggested that in order for novices to acquire these knowledge, extensive practice is needed (Norman, 1978; Simon, 1980). In the case of computer programming - often thousands of hours of practice are required (Pea & Kurland, 1984d; Dalbey & Linn, 1985). Even then, research has pointed out it is difficult for experts to transfer these skills across domains (c.f Frederiksen, 1984; Paris & Winograd, 1990b).

Transfer of problem solving skills could be viewed along a number of continua (Salomon & Perkins, 1987; Lehrer, 1989; Palumbo, 1990). One of the most commonly referenced continuum is that of near and distant transfer. Broadly speaking, near transfer refers to the transfer of skills to a new domain that is of similar logical structure but different surface form while distant transfer refers to the transfer of skills to a new domain that is of dissimilar logical structure (Gick &

Holyoak, 1980; Hayes & Simon, 1977; Pea & Kurland, 1984c; Burton & Magliaro, 1986). In essence, the distinction between near and distant transfer rests on the similarity between the task environment involved in the training domain and the task environment of the problem solving domain to which the skills is to be transferred. Ultimately, whether transfer of skills occur or not depends on the problem solver's ability to recognized the connections between "problem isomorphs" - problems of identical logical structure but different surface form - and to apply problem solving skills learned in the training domain to the new problem solving domain (Pea & Kurland, 1987).

In general, it is reasonable to expect that near transfer is more likely to occur than distant transfer as the initial environment and the environment to be transferred to have many similarities in terms of function of commands and the rationale of the concepts (Palumbo, 1990).

Research on LOGO programming and transfer of problem solving skills has often failed to identify what type of transfer these studies were attempting to measure. More often than not, researchers chose certain problem solving measures in their studies without examining how the logical structures of these problems correspond with LOGO programming. Therefore, it is important that future studies should differentiate between near and distant transfer by identifying and using the appropriate problem solving measures.

A related, but also crucial question is, what kind of mechanisms can facilitate transfer of problem solving skills? More specifically, how do computer educators provide a programming environment that can optimize the transfer of problem solving skills?

Conditions for transfer

Recently, Salomon and Perkins (1987) have identified two mechanisms with which transfer can occur - low road and high road transfers. Low road transfer depends on practice of skills to near automaticity in one context and these skills become activated spontaneously in another context. On the other hand, high road transfer involves mindful abstraction from one context and application to another. While low road transfer in general requires prolonged practice in order to reach

automaticity, high road transfer involves conscious efforts to transfer through reflection and monitoring, not necessarily with prolonged practice (Belmont *et al*, 1982; Brown *et al*, 1983). In this context, concepts underpinning high road transfer are similar to those involved in metacognitive training where deliberate effort is required to apply skills in one domain to another.

Using their model to analyse the research findings about programming and problem solving, Salomon and Perkins (1987) concluded that transfer of problem solving skills was more likely to occur when high road transfer was "forced" by instruction that directly and vigorously helps students to thinking about programming at an abstract level, in terms of general problem solving strategies.

Two questions arise from this conclusion: how and when should high road transfer be "forced" upon the learners in the learning of programming? Some answers to these two questions could be obtained by examining the social interactions within the learning environment and the type of instructions offered to the learners.

Social interaction

It has been suggested that the development of cognitive and metacognitive abilities is very much a social phenomenon (Turnure, 1987; Stone, 1989; Wadsworth, 1989) and that cognitive development is facilitated by peer interaction (Nastasi, Clements & Battista, 1990). For instance, according to Piaget, knowledge has a social origin in that knowledge derives meaning from social discourse, and that learners are able to "decentralize" through constant interchange with the environment (Piaget, 1963). Based on this premise, it can be deduced that cognitive growth, and indeed, the development of both problem solving and metacognitive skills may have origins in such interaction with the environment (Clements & Nastasi, 1988).

Vygotsky (1978) also argues that all higher psychological functions (eg perception, voluntary attention) have social origins. Specifically, he claims that adults and more capable peers mediate a child's experience. Many of the successful training studies have been influenced by the Vygotskian notion of guided learning within a learner's zone of proximal development, described as

> *the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers. (Vygotsky, 1978:86)*

In other words, it is the distance between what a child can do working alone and what the child can achieve with assistance (Day, Cordon & Kerwin, 1989).

Vygotsky further noted that the actual developmental level of a child characterized his/her mental development retrospectively, while the zone of proximal development characterized mental development prospectively. He also argued that an essential feature of learning was the creation of the zone of proximal development,

> *that is, learning awakens a variety of internal developmental processes that are able to operate only when the child is interacting with people in his environment and in cooperation with his peers. Once these processes are internalized, they become part of the child's independent developmental achievement. (Vygotsky, 1978:90)*

An important implication from Vygotsky's argument is that within a programming learning environment, there needs to be an increase of interaction between teachers and learners, as well as between learners. Further support for this argument can be obtained from the works of Piaget which suggest that interaction with others can serve to provoke cognitive disequilibrium which leads to learners questioning their own thinking. Similarly, metacognitive psychologists have also advanced that children internalize and develop their individual competencies through collaborative social and linguistic interactions with more knowledgeable and experienced persons with whom they come in contact, such as parents, teachers and peers (cf Flavell, 1979; Palincsar & Brown, 1984; Chan, 1991). This implication further underlines Papert's (1980) notion of teachers fulfilling the role of anthropologists by making provisions for an environment to support children's intellectual development.

Recent research on LOGO programming has also noted the importance of social interactions within a LOGO environment (cf Chapter 4). For instance, the work by McDougall (1988) with children learning recursion through LOGO, clearly highlights the importance of peer teaching and learning in achieving the "prospective" cognitive development of the learners. Similarly, research by Clements and his associates (eg Nastasi & Clements, 1988; Nastasi, Clements & Battista, 1990) has clearly demonstrated the effects of social interaction on the learning of their subjects.

A corollary to this proposition is that if the intended outcome of such learning experience is the improvement of problem solving skills, then the focus of such interaction should be on the skills and processes involved with problem solving, similar to those in metacomponent or metacognitive training (cf Au, Horton & Ryba, 1987). For instance, learners could be encouraged to reflect on their problem solving experience and skills, and then share the experience with each other. Teachers and more capable peers could be encouraged to assist the less capable learners initially, but gradually transfer the control of tasks to the less able learners (Day, Cordon, & Kerwin, 1989).

Therefore, in order to facilitate the development of problem solving skills within a LOGO programming environment, it is important that (i) there be an increase in social interactions between teachers and learners as well as among learners themselves; and (ii) the focus of these interactions be on the problem solving processes.

Teacher Intervention

The role of a teacher in LOGO programming environment, apart from fostering the social interactions as discussed above, will also need to take into consideration of the provision of metacognitive training that is facilitative of the development of problem solving skills.

Researchers (cf Wood, Bruner & Ross, 1976; Paris & Winograd, 1990a) have suggested that a teacher provide a scaffold which consists essentially of the teacher controlling those elements of the task that are initially beyond the capacity of learners thus permitting them to concentrate upon and complete only those elements

that are within their range of competence. The metaphor is suggested by the fact that a scaffold is a support system that is temporary and adjustable. The initial scaffold may include the expression of the knowledge and cognitive strategies involved in problem solving. In the programming context, this knowledge may include: how to decompose a complex problem into smaller ones, how to plan a solution, how to monitor and evaluate programming solutions, and how to analyse and interpret error messages. In essence, this knowledge consists of both procedural and conditional knowledge involved in solving programming problems. However, as the learner demonstrates increasing competence, control needs be ceded to the learner.

In conducting metacognitive instruction, one aspires to teach students to plan, implement, and evaluate strategic approaches to learning and problem solving. Students, therefore, assume control of their own learning. Research evaluating metacognitive strategy instruction suggests that this empowerment of students can be achieved when teachers provide explicit instruction regarding efficient strategies and gradually relinquish control for the application of these strategies to learners who are informed regarding the purpose and consequences of their activity (cf Paris & Winograd, 1990a; Chan, 1991).

Although there is emerging evidence in reading comprehension that scaffolded instructions have resulted in significant gains on comprehension assessment (eg Palincsar, 1986; Paris & Winograd, 1990a), their applications and usefulness in a programming environment have yet to be explored.

In the teaching of programming in schools, traditionally, a content-oriented approach has been adopted, that is, the focus has been on the content of the language (the use of syntax of the language) rather than a process-oriented approach (with emphasis on the processes and skills involved in constructing the solution). If one of the assumptions in teaching programming is that of the facilitation of transfer of problem solving skills, then there is clear need to evaluate the use of a process-oriented approach in achieving such an aim.

A process-oriented approach

It is in light of the above discussion this study developed a process-oriented approach in teaching LOGO programming with the focus on the development of problem solving skills of the learners (Au, Horton & Ryba, 1987). This approach takes into consideration the notions of scaffolded instructions (e.g Wood, Bruner & Ross, 1976), self-management skills (eg Belmont *et al*, 1982), and social interaction in facilitating the development of general problem solving skills (eg Vygotsky, 1978).

In particular, this approach has taken into account previous research in LOGO programming where there was confusion of the role of a teacher. In this instance, teacher intervention is not simply a matter of what one knows about LOGO or the type of activities that are provided. Rather, it has to do with the ways in which teachers talk with students, the types of questions they ask, and the sort of discussions that take place between students and their teachers. Such points of consideration relate very closely to how the teachers fulfil the role of anthropologist as described by Papert (1980), and how they make provisions for an environment to support children's intellectual development.

Leron (1985) has suggested that teachers need specific guidance and teaching materials to help promote students' independence from the teacher and to provide suitable material so as to aid the implementation of the LOGO philosophy as enunciated by Papert. Pea, Kurland and Hawkins (1987) have also made recommendations which point the educator in the direction of creating a LOGO culture that is socially interactive and rich in opportunities to build bridges from LOGO to thinking about other domains of school and life.

Based on these considerations, a process-oriented approach consists of three important elements: (i) a series of LOGO worksheets which consist of activities for the learners; (ii) questioning techniques used by teachers; and (iii) provision of a socially reflective and interactive environment. The following sections will elaborate on each of these three elements.

LOGO Worksheets

The worksheets provide a sequence of activities of increasing difficulty for the learners (cf Chambers, 1986; Watt & Watt, 1986; Nolan & Ryba, 1986). The general outline consists of: (i) elementary Turtle commands, eg Forward, Left etc.; (ii) the concept of angles and turning; (iii) the Repeat command; (iv) the concept of procedures and editing commands; (v) subprocedures and superprocedures; (vi) management of the workspace and saving, etc.; (vii) the concept of variables; (viii) the conditional commands; and (ix) the concept of recursion.

Some of the activities are taken from Apple LOGO (Abelson, 1982b), Apple LOGO in the Classroom (Minnesota Educational Computing Consortium, 1983), Learning with LOGO (Watt, 1983b), Turtle Power Activity Book (Sharp, 1984a), Turtle Power Thinker's Guide (Sharp, 1984b), LOGO in the Classroom (Torgerson, Kriley & Stone, 1984), and Assessing Learning with LOGO (Nolan & Ryba, 1986). These activities form the basis for the development of a series of worksheets used by students in the present study. Within these activities, students are asked to exercise their thinking skills and to reflect upon their thinking. For instance, they are asked to experiment with a variety of commands. They are also asked to predict command outcomes, and if their predictions are incorrect they are asked to not only explain what is wrong with their plans, but to re-plan their programs. As students progress into more advanced activities and their own personal projects, they are encouraged to follow a general problem solving model based on the work of Belmont, Butterfield and Ferretti (1982) (cf Figure 5.1). This model encourages the students not only to reflect upon their own thinking in a systematic manner, but also to develop self-management and general problem solving skills such as planning, predicting, analysis, evaluation etc.

1. Think of a plan (analysis and planning);
2. Ask yourself if the plan would work by following every single step in the plan (analysis and prediction);
3. Try the plan out with the computer (experimenting and monitoring);
4. If the plan does not work, ask yourself what went wrong. Make sure you can find the mistakes (analysis and evaluation);

5. Change your plan and then try it out again (repeat the same process al over again).

Within this approach, students are also taught the skills such as breaking down a complex problem into increasingly smaller problems until the smaller problems are at a level simple enough for students to solve. For instance, in the more advanced activity sheets and their personal projects, students are reminded every now and again that "When you try to solve a complicated problem, it is useful to break down this problem into smaller subproblems and then solve these simpler subproblems one by one". Structure diagrams are used as tools to help students break down these complicated problems until the subproblems are manageable. The structured nature of the LOGO language provides an excellent medium for teaching these skills.

Teacher Questioning

Teacher questioning is used to complement the worksheets. Instead of providing answers to the students, teachers always try to encourage them to think about their own thinking by asking them questions which were embedded in a natural a dialogue as possible. For instance, when a student has problems, a teacher would ask questions such as, "Why did you do that?" "Is that what you want to do?" "Where do you think you have gone wrong?" "What do you think you should do?" "Have you planned that?" "How are you going to fix it?" etc. The teachers always avoid giving answers to the students except when it is apparent that they cannot proceed any further. Instead, the teachers impel the students to reflect on their own thinking, hence helping them to develop and practise a set of general problem solving skills. For instance, these types of questions and suggestions are classified under the sorts of processes that students are being encouraged to use:
Prediction: Draw what you guess will happen.
Experimentation: Play around with different numbers for the REPEAT command.
Planning: Make a plan first, then try it out on the computer.
Analysis and Evaluation: What was wrong with your plan, and how are you going to change it?

Moreover, within a process-oriented approach, teachers help students foster their abstract thinking. Initially, when students are trying to predict the movements of the Turtle or to plan a certain drawing, they are encouraged to walk out the movements of the Turtle either individually or in a group, or to draw the Turtle's path on a piece of paper. Later on, they are urged to think it through in their own minds rather than act it out physically. In addition to the exercises offered in the worksheets, at the end of each learning lesson, students are presented with challenges from the teacher in order to facilitate and promote abstract thinking.

Socially Interactive and Reflective Environment

The role of the teacher also includes providing for a socially interactive and reflective environment. In a process-oriented approach, teachers encourage students to discuss with others the reasons and the ways by which they obtain certain solutions. These discussions are carried out either in small groups or by the class as a whole. The major focus of these discussions is on the processes by which they arrive at their solutions rather than on the product alone. As well, at the beginning or the end of each session, the teacher provides some interesting problems for the whole class. Then students explain to the class how they come up with their solutions. They also share with the others different ways that could be used to obtain the same solution.

Moreover, students are required to do their own experimenting, predicting, planning, evaluation etc, *off the computer* so that they would have time to think through their steps rather than being too preoccupied with the use of the computer. Hence this environment of learning, besides being socially interactive, is also reflective; students are encouraged to reflect upon and monitor their own learning. This socially interactive and reflective environment forms an integral part of the process-oriented approach in teaching LOGO.

In summary, a process-oriented approach, apart from teaching students the syntaxes of a programming language, also attempts to provide appropriate training for the students in order to facilitate the development of their problem solving skills. One of the main aims of this study is to examine if there is any difference in the development of problem solving skills among students who learn LOGO using

process-oriented and content-oriented approaches. Table 5.2 provides a comparison between a process-oriented approach and a traditional content-oriented approach in the teaching of programming. The differences between the two approaches have been polarised to make these differences clearer.

Table 5.2

Comparison between process-oriented and content-oriented approaches

Process-oriented approach	Content-oriented approach
Explicit teaching of problem solving processes and programming language	Explicit teaching of the programming language
Students praised for using appropriate problem solving skills	Students praised for arriving at correct solutions
Self-referential thinking taught and encouraged	Spontaneous development of self-referential thinking
Group interactions encouraged	Individual work encouraged
Social skills directly taught	Social skills assumed
Teacher questioning focuses on problem solving processes	Teacher questioning focus on programming syntax
Worksheets emphasize problem solving skills and self-referential thinking	Worksheets emphasize learning of programming syntax
Students encouraged to share and discuss problem solving processes	Spontaneous sharing and discussing problem solving processes
Self-discovery highly encouraged	More structured activities
Explicit teaching of applications of problem solving skills to other contexts	Spontaneous recognition of application of problem solving skills to other contexts

Research questions of this study

It has been highlighted in Chapter Four that one of the major concerns with existing LOGO research is the lack of attention to the instructional methods used. It

has also been raised in this chapter that consideration needs be given to the relationship between instructional methods and the development of problem solving skills. Therefore, when examining LOGO programming and problem solving, it is important to exercise tighter control on the types of instructional methods used.

While the comparison of two different instructional approaches would provide information on the teaching of LOGO programming and the development of problem solving skills, some researchers have argued that BASIC - another popular computer language used in schools - could be used in a similar way to develop problem solving skills (Dalbey & Linn 1986; Mayer, Dyck & Vilberg, 1989; Norris, Jackson, & Poirot, 1992). Therefore, it was decided to include the comparison of a third group of students learning BASIC using a content-oriented approach which has been the typical approach used in the teaching of BASIC in schools.

In examining the literature of problem solving and transfer in this chapter and the measurement of development of problem solving skills in Chapter Four, it has become apparent that one of the major issues in the study of problem solving related to the measurement of problem solving skills. This literature suggests that it is important to identify the type of problem solving measures used, in particular, whether the measures used are of near or distant nature to the problem solving skills used in LOGO programming. Thus it was decided to incorporate measures both of a distant- and near- transfer nature in this study in order to gauge the development of problem solving skills of the subjects.

In light of the above discussion, this study seeks to examine if there is any difference in:

(i) problem solving skills of learners with the two programming languages - LOGO & BASIC;

(ii) problem solving skills of learners with the two instructional methods - process-oriented and content-oriented approaches; and

(iii) the interaction among students in the various programming groups.

CHAPTER SIX

RESEARCH DESIGN AND METHODOLOGY

This chapter focuses on the design and methodology of the present study. A rationale is established for using an experimental approach as well as an observational approach to data collection. A detailed account of the evolution of the research through a pilot study is then given. The development of the measuring instruments, teaching modules and teaching strategies for the three different programming groups are reported. The final section of this chapter tenders the hypotheses of this study and describes procedures used in the analyses of the data.

Overview of research design

On the basis of Trow's well established research principle, namely, that 'the research problem under investigation properly dictates the method of investigation' (Trow, 1957), it was decided that two different data collection methods would best answer the research questions in this study.

First, in order to evaluate any changes in the problem solving skills of the subjects as a result of learning programming, a traditional experimental design with pre- and post-tests was considered suitable to answer the first two questions of this study (see Chapter Five, p. 109). Therefore, evaluations of the problem solving skills of the subjects were made before and after the intervention phase. The independent variables in this study were the instructional conditions and programming languages. The dependent variables measured by the experimental design were problem solving skills assessed by a number of instruments including (i) the Mathematics sub-test of the Progressive Achievement Test; (ii) three sub-tests of the Wechsler Intelligence Scale for Children Revised - Picture Arrangement, Block Design, and Object Assembly; (iii) the Raven's Standard Progressive Matrices; (iv) Rule Naming Test; (v) the Torrance Test of Creative Thinking; and (vi) the Tower of Hanoi.

Four subject groups were used in both the pilot and the main studies - three experimental groups and a control group. Subjects were randomly allocated to one

of the groups with different instructional conditions. Further elaboration on these instructional conditions will be provided in a later section of this chapter. A schematic representation of the experimental design is provided in Table 6.1.

Table 6.1

Schematic representation of the experimental design

Subject Group	Instructional Condition
LOGO process-oriented	Using a process-oriented instructional condition with emphasis being placed upon the skills and processes of problem solving as well as the syntaxes of the LOGO programming language
LOGO content-oriented	Using a content-oriented instructional condition with emphasis being placed upon the content (syntaxes) of the LOGO programming language
BASIC	Using a content-oriented instructional condition with emphasis being placed upon the content (syntaxes) of the BASIC programming language
Control	No treatment

Along with the experimental measures, a decision was made to collect information on the processes of teaching and learning during the intervention phase. Hence systematic observations were made during the intervention phase of this study in relation to the teaching strategies, individual students and group interactions among the students. These measurements and observations will be described in detail in later sections of this chapter.

Pilot Phase

The results of ongoing research and systematic observation provided a logical basis upon which to design the teaching programmes ultimately used in this

Note On Ecological validity

The decision to collect observational data was made in order to strengthen the ecological validity of this study. Some researchers have argued that a pre-post experimental design is not sufficiently sophisticated to evaluate the effectiveness of LOGO learning that occurs during the intervention phase (cf Papert, 1987). For this reason, the approach adopted in this study has been to obtain further information on the intervention methods through observations and analysis of students and teachers interactions in the LOGO environment. It is argued that observational data will provide the researcher with a deeper understanding of the teaching and learning processes within each intervention method which could not be gauged by a simple pre-post experimental design.

investigation. Following are some of the developments made in preparation for trial evaluation of the teaching programmes:

1. From observations in previous research, it was considered important for the students to work with worksheets and computers individually so that they would be able to proceed at their own pace.
2. Activities contained in the worksheets were designed in sequence that progressed from the simple to the complex (cf Chapter 3 and 5). As well, the activities were designed to be challenging for subjects of the appropriate age group and appropriate to the various instructional conditions of this study.
3. In conjunction with the worksheets, teachers were trained to reflect the appropriate instructional conditions used in the design of this study.
4. Printers were interfaced with the microcomputers so that students could obtain hard copies of their work in the form of graphics and programs. The students could then take their work home and analyse their work if needed. As well, the hardcopy could then be given to the students as a personal copy of their work while a duplicate printout was retained for file records.
5. Careful consultations were made with the principal and the teachers of the school so as to ensure their full cooperation. As well, arrangements were made with parent volunteers to transport the children involved with the pilot study from the school to the university where the teaching phase of the pilot study took place.

After completion of the implementation of the modifications described in the previous section, a pilot study was undertaken prior to the major study.

The purpose of the pilot phase was to test out some of the innovative elements (eg the use of various instructional conditions including the construction of the worksheets and the questioning techniques of the teachers) and implementation procedures of the project. It was decided to carry out a small scale study identical in format to the main study, but which involved only eight subjects, that was equivalent to one-tenth of the number required for the major project. The pilot

phase took place over a period of four months, from September to December in 1985. A brief account of the pilot study and its evaluation is presented here.

Subjects

Eight students randomly selected from 45 students in the standard four classes of the Hokowhitu School, Palmerston North, participated in the pilot phase. As the major study would involve standard three and standard four children in this school in the following year (1986), only the students in standard four classes were included in the pilot, to leave current standard three students free to participate next year. Permission for their participation in the study was obtained from the school, the regional Education Board, and the children's parents. A record was kept on each child. The record included the child's name and identification, sex, age, birthday, classroom, address, parents' occupation, group assignment and results of all dependent measures. The age of the children ranged from 9 to 10 years.

Eight subjects were randomly assigned such that two subjects were placed in each of the four instructional groups as discussed in the overview of this chapter. Each pair consisted of one boy and one girl.

(1) Process-oriented LOGO pair. Subjects were taught to program in LOGO using a process-oriented approach, with the emphasis on the processes of programming as well the contents.

(2) Content-oriented LOGO pair. Subjects were taught to program in LOGO using the traditional way of instruction, with the emphasis on the content of programming only.

(3) BASIC pair. Subjects were taught to program in BASIC using the traditional way of instruction, with the emphasis on the content of programming only.

(4) Control pair. This was a "no-treatment" group. However, subjects were given the opportunity to learn how to program in LOGO upon the conclusion of the post-test.

Procedure

All the eight subjects were pre- and post-tested with a number of academic achievement and problem solving measures which will be described in the latter part of this chapter. After the pre-testing had concluded, the subjects in the experimental groups were then taught how to program. A brief description of the teaching approaches used with each pairs of students is given below.

1. Process-oriented LOGO pair

The first pair was taught to program in LOGO using a process-oriented approach (cf Chapter Five). In essence, this group was taught using a guided-discovery approach with plenty of opportunities for the learners to explore and reflect upon their processes of problem solving. The teacher acted as a facilitator rather than as an instructor. For instance, instead of providing answers to the students' questions, the teacher would ask questions in an attempt to assist the students to clarify their problems and reflect on their own thinking. The teacher also provided a socially interactive and reflective environment for the learners by encouraging the students to discuss their work and problems. As well, in the worksheets provided, the students were encouraged to plan their work carefully, and if their solutions were not correct, to evaluate their work carefully in order to find the right solutions.

2. Contented-oriented LOGO pair

The second pair was taught to program in LOGO using a conventional form of instruction. This approach primarily stresses the learning of program operations and syntax through a series of instructional modules which systematically introduced students to the various LOGO concepts (cf Chapter Five). However, the emphasis here was on the content of programming. By contrast with the process-oriented pair, there was relatively less freedom to explore and develop own ideas. Children were free, however, to work independently at the tasks contained in the worksheets. The lessons for this pair were adapted from several popular LOGO books and manuals (eg McDougall, Adams & Adams, 1982; Ross, 1983).

3. BASIC pair

The third pair was taught to program in BASIC, the language that was most commonly available for microcomputers. The approach adopted for this group was similar to that in the LOGO content-oriented pair. Emphasis was on the teaching of the syntaxes and various operations in BASIC. Most of the commercially available BASIC programming books (eg Zaks, 1983; Boren, 1984; Grauer, Gordon & Schemel, 1984) tend to follow this instructional approach. The lessons for this group were adapted from several popular BASIC books and teaching manuals.

4. Control pair

The fourth pair served as the "no-treatment" group to account for factors such as maturation and other experience during the pilot phase.

During the programming phase, each child was given two programming sessions each week for six weeks learning to program at the Microcomputer Learning Centre of the Education Department of Massey University. In each session which lasted one hour, students spent about half of their time at a computer individually and half of their time off the computer, in segments of about 15 minutes alternately. While off the computer, they spent their time working through the worksheets either individually or in small groups.

Three graduates from the Massey University course "Computers in Education" served as the programming instructors. All of them were familiar with the programming languages that they taught. In addition, the researcher was present at each session to monitor their teaching. Regular meetings were held between the researcher and individual teachers so as to ensure that they had adhered to the teaching strategies appropriate to each instructional condition.

The programming phase concluded in late December, 1985 with all the students having had 12 programming sessions as noted above. At the end of the programming phase, all the subjects, including those in the non-programming group, were post-tested with the battery of tests used during the pre-test. The following is a description of the measurement instruments used.

These measurements included: three sub-tests of the Progressive Achievement Tests (PAT) - Mathematics, Listening Comprehension, and Reading Comprehension;

three sub-tests of the Wechsler Intelligence Scale for Children Revised (WISC-R) - Picture Arrangement, Block Design, and Object Assembly; Raven's Standard Progressive Matrices; Rule Naming Test; Torrance Test of Creative Thinking; and Tower of Hanoi.

The following is a description of the specific tests used in this study:

1. The Progressive Achievement Tests. These tests were published by the New Zealand Council for Educational Research to assess the academic performance of students in New Zealand schools. Three sub-tests were chosen in this study - Mathematics, Listening Comprehension and Reading Comprehension. The PAT Listening Comprehension and Reading Comprehension sub-tests were used in the pre-test only to screen the subjects and establish academic achievement levels. The Mathematics sub-test was used to gauge any improvement of general mathematical problem solving ability of the subjects.

   The administration of these three sub-tests followed strictly the procedures stipulated by the Teacher's Manuals (Elley & Reid, 1969, 1971; Reid & Hughes, 1974) of these tests. Alternate forms of these three tests were used for each of these tests.

2. The Wechsler Intelligence Scale for Children Revised (Wechsler, 1974). Three sub-tests were chosen among this battery of tests. These sub-tests were: Picture Arrangement, Block Design, and Object Assembly. Following is a description of these three sub-tests.

   (i) Picture Arrangement. This sub-test consists of eleven different cut-up pictures, or picture sequences to be assembled, graded in order of difficulty. The first four are given to children below the age of eight or to older suspected mentally retarded children. The remaining eight are qualitatively different in that they consisted of picture sequences, which when placed side by side in the proper order tell a logical story of actions or consequences. Time credits are given for speed of arrangement and there are time limits.

This sub-test, according to the publisher, measures non-verbal intelligence factors such as planning involving sequential and causal events and synthesis into intelligible wholes. Therefore, this sub-test was used in the present study to gauge the ability of the subjects to plan sequentially as well as their ability to synthesize components into logical wholes. These are the abilities that are supposed to have developed after learning how to program, especially with the LOGO language.

(ii) Block Design. This sub-test consists of ten two-dimensional designs to be reproduced with multicoloured blocks within time limits. These blocks have red on two sides, white on two sides, and red/white on two sides. The first two designs are copied from the examiner's block construction rather than a picture, and given only to children below the age of 8 or to older suspected mentally retarded children. The remaining eight designs are reproduced from a pictorial pattern shown to the child. The first seven patterns use four blocks, the last three call for nine blocks. All designs are symmetrical and thus involve some degree of pattern repetition, either top-down or right-left.

This sub-test, according to the publisher, measures non-verbal intelligence factors such as analysis, synthesis, and reproduction of abstract designs. As well, it also measures logic and reasoning applied to space relationships. This sub-test was used in the present research to gauge the non-verbal problem solving ability of the subjects, in particular, their ability of developing a purposeful use of three essential planning and analysis components, including identifying the components, envisaging each as a separate entity and putting them to make up the whole (synthesis).

(iii) Object Assembly. This sub-test consists of four cut-up picture puzzles of a girl, a horse, a car, and a face to be assembled within time limits. Time credits are given for speed.

According to the publisher, this sub-test measures the ability of the subject to anticipate spatial part-whole relationship and flexibility in working

toward a goal. One important difference between Block Design and Object Assembly should be noted. Whereas the blocks must be assembled to match a pattern, the objects must be assembled with no clues beyond naming the "Girl" and the "Horse", and no leads at all for the "Face and the "Car". Thus the subject must look for the key to each object, figuring out in advance what he/she is constructing, and then solve the puzzle systematically. Hence this sub-test was used to gauge the ability of the subjects to identify a problem and its components, and then formulate the solution accordingly. It was also used to measure the ability of the subjects to synthesize parts of a solution together. The skills to analyse and synthesize are important skills to be developed in programming.

3. Raven's Standard Progressive Matrices (SPM). This sub-test contains five sections each of twelve items printed in a booklet for use with a separate answer sheet. Each of the sixty items is a design or "matrix" from which a part has been removed. A subject is required to examine the design and decide from a number of pieces given below it, which is the right one to complete it. In each of the five sets the first problem is as nearly as possible self-evident. The problems that follow the first one become progressively more difficult. The order of the items provides the standard training in the method of working. The five sets provide five opportunities for grasping the method and five progressive assessments of a person's capacity for intellectual activity (Raven, Court & Raven, 1984).

This test was designed by Raven as a test of a person's capacity at the time of the test to apprehend meaningless figures presented for his/her observation, see the relations between them, conceive the nature of the figure completing each system of relations presented, and by so doing, develop a systematic method of reasoning (Raven, 1956). Hence this test has generally been used to measure the general non-verbal problem solving ability of a person such as visual analysis and checking. A study by Horton & Ryba (1986) observed an apparent increase in a person's scores with the Raven's Standard Progressive Matrices after learning to program with the LOGO

language, suggesting that a person's ability measured by this test might increase after learning how to program.

4. Torrance Test of Creative Thinking. This is a test designed by Torrance and his colleagues (Torrance, 1966, 1972, 1974) to assess the creative thinking potential of the subjects. This test consists of two forms, the verbal and the figural. Only the figural test was used in this project as one of the major objectives of this study was to assess the non-verbal problem solving skills of the subjects. The Figural test includes three activities with an overall administration time of 30 minutes, 10 minutes for each activity. Four components of divergent thinking can be discerned in this test: originality, flexibility, fluency and elaboration. The first task, Picture Construction, is designed to stimulate originality and elaboration. The two succeeding tasks, Incomplete figures and Repeated Figures, increasingly elicit greater variability in fluency, flexibility, originality, and elaboration. There is not enough time to complete all of the possible units and make them highly elaborate or original. Thus, response tendencies and preferences emerge as a result of time pressure.

In the first activity, Picture Construction, subjects are required to think of a picture in which the given shape made of coloured paper with an adhesive backing (in Form A, a tear drop or pear shape; in Form B, a jelly bean shape) is an integral part. An effort is made to elicit an original response by asking subjects to try to think of something that no one else in the group will produce. Elaboration is encouraged by the instructions to add ideas that will make the picture tell as complete and as interesting a story as possible. Thus the product is evaluated only for originality and elaboration.

The Incomplete Figures Activity consists of ten incomplete figures. Subjects are asked to complete each figure. Each completed figure is scored for flexibility, fluency, originality, and elaboration.

The Repeated Figures Activity is similar to the Incomplete Figures Activity. The stimulus material in Form A is 30 parallel lines and in Form B it is 40 circles. The common element tested is the ability to make multiple

associations to a single stimulus. In this activity, a deliberate attempt is made to stimulate all four types of divergent thinking and to set up a conflict among the response tendencies represented by them. Fluency is stimulated by the instructions, "see how many objects or pictures you can make"; flexibility, by "make as many different pictures and objects as you can"; originality, by "try to think of many ideas as you can into each one and make them tell as complete and interesting a story as you can." The time is not adequate to permit emphasis on all four kinds of thinking. Thus, individual response tendencies come into play.

The instrument was used as part of the battery of tests in this study to measure any changes in the creativity of the learners. Creativity is considered by many problem researchers as part of problem solving (cf Frederiksen, 1984). A study by Clements & Gullo (1984) observed a significant increase in learners' creativity after having learnt to program with LOGO. The study by Clements & Gullo suggests that a person's divergent thinking might improve after learning to program with LOGO. Similarly, a study by Horton (1986) also indicated that a person's creativity might improve after having learned to program with LOGO. Therefore, it was considered important to examine the possible development of a person's creative thinking in this study.

5. Rule Naming Test. This test was initially designed by Bourne (1970) and since has been used in numerous research projects, among which was the Lamplighter Project which investigated the relationship between computer programming and logical thinking (Gorman & Bourne, 1983). This project used the rule naming test to assess the possible gains in logical thinking of subjects after the learning of computer programming.

For the rule naming test, a stimulus universe of four trinary variables was created. Cards of the 81 combinations of colour (red, yellow, or blue), shape (circle, square, or triangle), size (small, medium, or large), and number (one, two or three) were made. Forty cards of each of four problems were arranged such that every run of 4 cards contained one

exemplar of each truth-table category and every run of 10 cards contained two exemplars of each truth-table category. Otherwise, the cards were ordered randomly. Four rule naming tests were used, all based on the conditional (if...then) rule, but different in relevant attributes - blue and square, circle and yellow, red and triangle, and square and one. The first one was used as a non-scoring trial run so that all the subjects could become familiar with the procedures of completing the test.

The subjects were tested individually. The rule naming test was presented as a game in which the objective was to determine the rule between the attributes named (see Appendix 3, instructions for administration). The subjects were told that they would see cards that varied in colour, size, shape and number. They then saw 11 sample cards that they described completely. Next, they viewed the test cards, one at a time, indicated whether each card obeyed or broke the unidentified rule, and received feedback on their answers in relation to whether the answers were correct. Prior to each problem, the subjects were told what the relevant attributes were and that they would be given "hints" during the game. The hints were merely reminders of the two relevant attributes and were to the students at the beginning of each problem and after cards 20, 40, 60, and 80. On each of the four problems, the students worked to a criterion of 12 consecutive correct judgments or until they had seen 100 cards. Answers to each individual problem were recorded on separate scoring sheets (see Appendix 4).

The rationale for choosing a rule naming test in the present study was fivefold: First, Bourne (1970) showed (i) that there was a style of thinking that one learns from solving any of the four binary rules (conjunctive, disjunctive, conditional, and biconditional), (ii) this cognitive style transfers positively to learning all of the other rules, and (iii) this style involves a process of categorization like that of sorting by truth tables. Such a mode of thinking, was attributed to computer programming (Kolata, 1982). Second, the LOGO language is especially rich in exemplars of independent attributes such as turtle steps, turtle angles, coordinates, turtle heading, number of repeats etc. Third, the rule naming test can be scored by the more sensitive

measures of number of errors to criterion and number of trials to criterion, rather than by just success-failure. Fourth, Bourne & O'Banion (1971) found a developmental trend in difficulty of problem solution such that the conjunctive rule was easiest and the biconditional most difficult. Fifth, the rule naming test appears to correspond with one of Sternberg's metacomponents, i.e. selection and combination of attributes relevant to task completion (cf Clements, 1985a, 1985b). The conditional rule was selected for the present study because it is moderately difficult for 8 - 10 year old children.

6. Tower of Hanoi. The Tower of Hanoi is structured as a set of nested sub-problems having the property of recursion. There are three pegs and on one peg are arranged a number of disks of increasing size from top to bottom. The task is to reconstruct the Tower on either the second or the third peg in the minimum number of moves under the constraint of two rules: (1) a larger disk cannot be placed on top of a smaller one, and (2) only one disk can be moved at a time. For each set of disks there is a minimum number of moves according to the formula 2^n -1 with n equal to the number of disks. The problem is recursive in that a problem of n disks can be decomposed into sub-problems of the n-1 form.

This puzzle was chosen because of the structural similarities between this puzzle and the LOGO language and the similar forms of problem solving that this puzzle and LOGO programming facilitate. One method that has proved successful in studying individual differences in approaching programming has been to present subjects with two tasks, an "indicator' task and a programming "target" task (Coombs, Gibson & Alty, 1981). For a task to be a good indicator of its target, it must be well-understood, performance on it must be easily studied, and there must be similarities between the two tasks which make performance on the indicator task a basis for generating hypotheses about strategies and performance on the target task.

The Tower of Hanoi was chosen because it fulfilled these three requirements for an indicator task for LOGO programming:

(1) It is a well-known task that has been extensively studied in the literature on problem-solving (Anzai & Simon, 1979; Klahr & Robinson, 1981; Luger, 1976; Luger & Steen, 1981; Neilsson, 1971; Piaget, 1976; Simon, 1976, 1979).

(2) The puzzle is representative of a class of transformation problems which involve reaching a goal through a sequence of moves. As such and because it is a physical puzzle, it involves a series of observable steps so the decision-making process of the child is accessible for analysis.

(3) It has structural features in common with the LOGO language and facilitates a similar approach to problem solving.

In summary then, there is a number of similarities between the two tasks which provide a basis for the relationship of indicator task to target task. There are structural characteristics in common and fundamental similarities in the way in which the problems posed in the two tasks can be broken down into sub-problems and these elements built into a solution. Therefore, the Tower of Hanoi was considered appropriate to measure the transfer of problem solving skills from a LOGO programming context to a near-transfer context.

Four problems were administered to subjects individually: 2-disk problem, which was used as a non-scoring trial; 3-disk problem; 4-disk problem; and 5-disk problem. The rules of this puzzle were explained to the subjects and they were urged to regard this test as a game (see Appendix 5, instructions for administration). Scoring sheets (Appendix 6) were used so that each move by the subject could be recorded for later analysis.

To quantify the subjects' performance on the Tower of Hanoi, two scoring systems were used. The first one was the total number of moves a subject required to solve each problem. The second one was according to the recursive, sub-problem nature of the Tower of Hanoi. The second scoring system was used to measure the percentage of sub-problems solved correctly. To solve a three disk problem, a subject must solve a series of 2-disk sub-

problems. The second scoring system measured the percentage of 2-disk sub-problems solved correctly. Similarly, a four-disk problem consisted of a series of 2-disk and 3-disk sub-problems, the second scoring system then measured the percentage of 2-disk and 3-disk sub-problems correctly; a five-disk problem consisted of a series of 2-disk, 3-disk and 4-disk sub-problems, the second scoring system then measured the percentage of 2-disk, 3-disk and 4-disk sub-problems correctly. Appendix 7 provides an example of the calculation of the 2-disk and 3-disk sub-problem scores of a four-disk problem of the Tower of Hanoi.

Evaluation of pilot phase

Given the small number of subjects in each group, it was not meaningful to perform any statistical test on the pre- and post-test scores. However, the trial evaluation served as a basis for obtaining observational and anecdotal data on practical aspects of this project. The following sections provide an evaluation of this pilot phase.

During this period, much information was obtained in relation to the procedures of implementation. For instance, it was observed that given proper instructions, children did not have many problems learning how to operate a computer physically by themselves. The subjects were given full control of the operations of the computer, such as switching on and off the computers, and using floppy discs to store and retrieve their own work. As well, the subjects in the pilot study demonstrated much enthusiasm in learning how to program with a computer. This was reflected in that absenteeism was minimal and only occurred when the subjects were sick. Moreover, the children involved in the pilot study did not seem to have too much difficulty in learning the programming languages.

During the pilot study, the researcher was able to observe the social interaction of the subjects while they were learning how to program. Although a small number of subjects were used in the pilot study, it was observed that there were substantially more interactions among the subjects, especially among those in the two LOGO pairs, as compared to those in a traditional classroom.

As well, the pilot study provided opportunities for the researcher to examine the different procedures involved with the administration of the various problem solving measurements to the subjects both in groups and individually. These observations served as a basis for the standardization of the administering of the problem solving instruments to the subjects in the main study.

In this particular study, there were three innovative aspects that needed to be examined and developed in the pilot phase before proceeding to the major study. These aspects were:

(A) The teaching modules for the three different programming groups;

(B) The role of the teachers for the three different programming groups;

(C) Some of the instruments for measuring problem solving skills. Observations related to these three aspects will now be discussed.

A. The Teaching Modules

In general, it was found that the subjects did not have much difficulty with the understanding of the modules, showing that the ways that the modules were designed were suitable to the reading and comprehension levels of the subjects who had an average reading level of approximately standard three to standard four.

Based on the comments of the teachers and the students, a number of changes were made to the teaching modules for the three teaching groups.

Modifications were made in relation to the content of each module to enable the children to finish each module within one hour. This was considered desirable for administrative purposes and to maintain pupil self-confidence.

Additional exercises were included at the end of each module so as to provide (i) extra practices for the students; and (ii) continuity from session to session.

Other alterations were made to the structure of the teaching modules so that they could reflect the differences between the process-oriented and content-oriented approaches (Au, Horton & Ryba, 1987). For instance, more questions that could help children to reflect on their thinking were added to the modules for the process-oriented group. As well, systematic introduction to the various problems solving skills and a model of problem solving (Belmont, Butterfield & Ferretti, 1982; cf

Figure 5.1) were incorporated into these modules. Further modifications were made to the content-oriented modules so that a clearer distinction was made between the process-oriented and content-oriented approaches. These distinctions were further enhanced with the input of the teachers who were responsible for teaching the three different groups. These input included the type of questions asked of the students, how teachers might respond to the questions of the students, the structure and content of the worksheets, and the organization of group work among the students. The role of the teachers including the types of questions they used will now be discussed.

B. The Role of the Teachers

The role of the teachers was considered vital in the distinction between the process-oriented and content-oriented approaches. Two components could be discerned: (i) teacher questioning, and (ii) provision of appropriate learning environments.

Teacher questioning was considered as an important element in the process-oriented approach to promote self-referential thinking. It was used to complement the sets of structured activities in the teaching modules. Teachers, instead of providing answers to the students, always tried to encourage them to think about their own thinking by asking them questions which were embedded in as natural a dialogue as possible. For instance, when a student had problems, the teacher would ask questions such as "explain how it can draw?", "why did you do that?", "is that what you want to do?", "where do you think you have gone wrong?", "check it carefully.", "what do you think you should do?", "have you planned that?", "how are you going to fix it?", "follow your plan right through." etc. The teacher always avoided giving answers to the students except when they were really stuck. Instead, the teacher always impelled the students to reflect on their own thinking, hence helping the learners to develop and practise a set of general problem-solving skills.

As the students became better able to perform these skills, the teachers then increased their demands until the students became increasingly able to control their cognitive processing, shifting from conscious other-regulation to conscious self-regulation. The change to self-monitoring constitutes a major step in the students'

learning and enhances the transferability of those problem-solving skills to other contexts (Papert, 1980; Campione, Brown & Ferrara, 1983). It has been demonstrated that metacognitive training, which teaches these general problem-solving skills, can produce durable and generalizable improvement in performance in other domains (cf Baker & Brown, 1984).

Moreover, students were assisted to foster their abstract thinking. Initially, when the students were trying to predict the movements of the turtle or to plan a certain drawing, they were encouraged to "walk-out" the movements of the turtle either individually or in a group. This activity forms part of Papert's exposition of concretizing the formal (Papert, 1980). Later on, they were urged to think through in their own mind rather than acting it out physically. Besides the exercises offered in the worksheets, at the end of each learning session, students were presented with challenges from the teacher with the view to improve their abstract thinking.

Apart from the use of structured worksheets and meaningful questioning, the role of the teacher also included that of the provision of a socially interactive and reflective environment (cf Vygotsky, 1978). In a process-oriented approach, teachers would encourage a student to confer with other students the reasons and the ways by which they obtained certain solutions. These discussions were carried out either in small groups or in the class as a whole. The major focus of these discussions though, would be on the processes by which they arrived at their solutions rather than just on the products alone. As well, at the beginning or the end of each session, the teacher would provide some interesting problems for the whole class. And then the students would explain to the class how they came up with their solutions. They would also be able to share with the others how different ways could be used to obtain the same solution. Moreover, students were required to do their predicting, planning and evaluation etc. off the computer so that they would have time to think through their own prediction, planning etc. rather than being too preoccupied with the use of the computer. Hence this environment of learning, besides being socially interactive, was also reflective in the sense that students were encouraged to reflect upon and monitor their own thinking. This socially interactive and reflective environment formed an integral part of a successful process-oriented approach in the teaching of LOGO.

For the students in the content-oriented groups, a more traditional approach of instruction was used. The emphasis was on the contents of programming. The teachers would go through the syntaxes of the language carefully with the students, and the ways of writing programs were also explained clearly to the students. And when the students had problems, answers were provided by the teachers directly, for instance, "place a space between forward and 40", "you have forgotten to turn the turtle by 30 degrees first before moving forward". This was in direct contrast to the approach adopted in the process-oriented group where the students were actively encouraged to think about their own thinking.

In order to ensure that the teachers adopt the approaches appropriate to each group, the researcher and the individual teachers had meetings before and after each session. The researcher was present at every session to observe the teaching and learning processes. Hence the researcher was able to inform individual teacher whether the correct approach was adopted such as the types of questions asked and the ways that discussions were conducted.

C. Problem Solving Measures

Apart from examining the various teaching aspects of this project, the pilot phase also provided opportunities to experiment with the administration of the various academic ability and problem solving measures used in this study. Research assistants were given extensive training as to how these measures should be administered. For instance, research assistants who administrated the WISC-R were trained and approved by registered psychologists. For the other tests, they were given rigorous training until proficiency was attained.

Little modifications were made to the measuring instruments as they were mostly published tests. Two modifications were made to the Tower of Hanoi test. First, the pegs were numbered so that the subjects would have a clearer idea of which peg they were moving the disks to. Second, a more detailed chart for recording the movements of the disks was drawn up so as to facilitate the recording the movements of the disks (Appendix 6).

In summary then, the following modifications or developments were made as a result of experiences and information obtained in the pilot study:

1. Three sets of learning modules were developed ready to be used in the major study.
2. One of the measuring instruments, viz, the Tower of Hanoi was modified so as to enhance its administration.
3. The observational instrument for recording social interaction was also developed during the pilot phase. Various categories for coding the interactions in the classroom were established.
4. The teachers and research assistants in this project were given ample practices during the pilot study regarding the teaching and administration of the various tests. When the major study began in early 1986, they were all well prepared.
5. A good working relationship was established with the principal and the school teachers. This was important for the smooth conduct of the main study.
6. A good reputation of the computer programming courses was established among the parents. This was important for obtaining parents' approval for the children to participate in the main study.

THE MAJOR STUDY

Some nine months prior to the major study, negotiations started with IBM (NZ) Ltd regarding the loan of necessary computer equipment and software for the major study. After a number of meetings with the Education and Marketing Manager and other personnel, IBM (NZ) Ltd agreed to loan six IBM JX computers, two IBM Pro Printers and the required software packages, viz LOGO and BASIC for the study. As well, the company also agreed to fund the employment of research assistants to conduct pre- and post-testing.

The major study took place between February and November of 1986. It was virtually identical to the pilot study which was undertaken in the previous year with the exception of modifications noted above. The only major difference was

that systematic observations were carried out by the researcher during the major study in three different areas. These are: (i) teaching behaviours of the teachers; (ii) the interactions of individual students with their teachers and their individual group; and (iii) the interaction of individual groups. The methods of observation will be described in detail in the procedure section.

Subjects

All the standard three and standard four students studying at the Hokowhitu Primary School in 1986 were initially involved in this study. There were 96 children altogether. Permission for their participation in this project was obtained from the school, the regional Education Board, their parents, and the individual students. Of these 96 students, 84 of them agreed to take part. Those who did not agree to take part indicated that it was due to their heavy schedule of extracurricular activities such as sports, dancing and piano lessons etc. Two screening methods were used to determine the suitability of these subjects to take part in this research. These methods were (i) a questionnaire; and (ii) results of the PAT sub-tests.

A questionnaire (Appendix 8) was administered to all the subjects. The information elicited in this questionnaire included: age, sex, birthday, address, parents' occupation, and computer experience. Interviews were also held between the researcher and individual potential subjects to cross-check with their answers to the questionnaires.

The three sub-tests of the PAT were then administered to the subjects to determine their academic achievement levels, and hence their suitability to participate. These sub-tests were: (i) Mathematics; (ii) Listening Comprehension; and (iii) Reading Comprehension.

As a result of these two screening procedures, two students were excluded from this study. The first one was excluded based on the information that he had had extensive experience with computers and programming. For instance, he was conversant with three different programming languages, including LOGO, BASIC and PASCAL. Many of the other students had had some experience with computers although the main involvement was with computer games. It was deemed to have no significant influence on the purposes of this study. The other student was not

selected because he scored 0 in all the three PAT sub-tests, demonstrating great difficulty in listening and comprehension of instructions. He was also considered to be a problem child by the principal and his teachers as he had had many serious behavioural problems in school.

Of the 82 students left, 80 were then selected randomly to participate in the major study (N = 80). These 80 subjects came from a variety of background. Their parents' occupations ranged from unskilled labourers to professionals. They also came from different ethnic groups, including three major ethnic groups of the general New Zealand population: European, Maori, and Chinese. On their academic achievement, the PAT scores represented students who had attained very high as well as relatively low levels in Mathematics, Listening Comprehension and Reading Comprehension. The age of these students ranged from 8 to 10 years old. Only one of them reached the age of 11. No attempt was made to control for sex, age, or IQ factors. Tables 7.1, 7.2, and 7.4 in Chapter Seven provide the age and sex distribution of subjects across groups, and summaries of their PAT scores.

Procedure

The 80 subjects selected for participation in the investigation were then administered the rest of the tests during March, 1986. These tests included: three sub-tests of the WISC-R, Raven's Standard Progressive Matrices, Torrance Test of Creative Thinking, Rule Naming Test, and the Tower of Hanoi. Detailed descriptions of these tests have been made in the pilot phase of this chapter. All of these tests were administered individually except the Raven's Standard Progressive Matrices and Torrance Test of Creative Thinking which were administered to small groups of 20.

These subjects were then randomly assigned to the various groups:

1. Process-oriented LOGO Group. 20 children were taught to program in LOGO using a process-oriented approach, with the emphasis on the processes as well as the contents of programming.
2. Content-oriented LOGO Group. 20 children were taught to program in LOGO using the traditional way of instruction, with the emphasis on the contents of programming only.

3. BASIC Group. 20 children were taught to program in BASIC using the traditional way of instruction, with the emphasis on the contents of programming only.
4. Control Group. 20 children were assigned to the "no-treatment" group. However, these children were given opportunities to learn how to program upon the conclusion of data collection.

In an attempt to assess whether this randomization process had created four groups of essentially similar children in relation to their listening comprehension, reading comprehension and mathematics achievement, one-way analysis of variance tests were performed to ensure that no statistically significance existed between the various groups for each of these measures (cf Chapter Seven on the results).

From April to September in 1986, all the children in the programming groups were then taught to program using either the LOGO (IBM LOGO) or BASIC (BASICA) languages. A special room was allocated by the school authority to be used as the computer room which housed six IBM JX microcomputers on loan from the IBM (NZ) Ltd. Each computer station was equipped with the appropriate software. As well, each subject was provided with floppy disk to store their work. Two printers were also available to the students to print out their work when required.

For management purposes, such as limited number of computers, small size of the computer room, extra space in the computer room for off-computer activities and so forth, each group was further subdivided into subgroups of 10. Hence there were six sub-groups. Four of these groups received their instructions after school hours while the other two had their programming lessons during school hours as arranged with the principal and their teachers.

Each child had one hour of programming learning each week for 20 weeks during the research period apart from the school holidays. During each session, students spent about half of their time at a computer individually and half of their time off the computer, in segments of about 15 minutes alternately. While they were off the computer, they spent their time working through the worksheets either

individually or in small groups. Students in all groups were also given take home exercises so that continuity of learning was maintained in between two sessions.

The teachers normally started off the lesson by introducing the key concepts of each lesson with appropriate examples to the students. Students then proceeded to work at their own pace. At the end of each session, appropriate summaries (eg the programming concepts & syntaxes, problem solving skills etc) were provided to each group. Charts were put up on the wall to facilitate the learning of the students. For instance, for the LOGO groups, charts of the angles were posted on the wall. They were also given individual charts to help them to understand the turning of angles. For the process-oriented group, a poster that explained the different problem solving steps was made available to them.

Three teacher assistants who had completed a Massey University undergraduate course on "Computers in Education" served as the programming instructors. All of them were familiar with the programming languages that they taught. Also, apart from having substantial experience in the teaching of programming, they were extensively trained during the pilot phase of this study. In addition, the researcher was present at each session to observe that they were teaching according to the prescribed methods. Regular meetings were held between the researcher and individual teachers so as to ensure that they had adhered to the appropriate strategies of teaching.

Apart from observing the teachers and their teaching strategies, the researcher was present during all the learning sessions to observe each individual student in turn as well as the group interactions in each group. Systematic schedules were designed, based on the observations during the pilot phase, to facilitate the observation. The following is an account of the various ways in which observation took place.

There were three types of observations conducted within the classroom in this project. They were observations relating to the following targets:

1. Teachers;
2. Individual students;
3. Group.

Note: Training of teachers

The training included: (i) discussion with the teaching assistants of the various strategies entailed in each of the two approaches used in this project, viz., the process-oriented and the content-oriented approaches; (ii) demonstration and modelling of strategies by the researcher (ii) the trying out of these strategies with the students in the pilot study; and (iii) the observation of each others' teaching and subsequent reflection on the strategies used.

The observations that had been done in a computing classroom to date had all been of an anecdotal nature. It is hoped that through the quantification of these observational data, a more systematic and valid basis can be established towards the identification of teacher and student behaviour within a computing environment.

1. Observation of Teachers. The major concerns of observing the teachers lay in the distinction between the process-oriented and the content-oriented approaches. Therefore, responses from teachers were classified according to their relationship to the process-oriented and content-oriented approaches. Any other responses that fell outside these two categories were classified as others.

A. The process-oriented category included responses of the teachers which were used by the teachers to encourage the students to think about the various problem solving processes. There were four major problem solving processes involved (cf Chambers, 1986; Nolan & Ryba, 1986). The following is a list of the processes and the exemplars of responses involved (Table 6.2):

Table 6.2

Processes and exemplars of responses of the process-oriented approach

a.	Prediction	P1	Guess
		P2	Predict
		P3	Estimate
b.	Experimentation	E1	Try
		E2	Play
		E3	Test
c.	Planning	PL1	Plan
		PL2	Think it through
		PL3	Think it ahead
d.	Analysis	A1	Find out... (Check, Where do you think you have gone wrong?)
		A2	Fix
		A3	What does... do?
		A4	To see if...
		A5	Why...?
		A6	What do you do... ?
		A7	Which one... ?
		A8	How did you... ?
		A9	Is that what you want?

These categories were coded as A1, E2, PL3 etc.

B. The Content-oriented category included responses of the teachers which were used to inform the students about the syntaxes of the programming language but which did not include the encouragement of students to think about the various problem solving processes. It was coded as C.

C. The Other category included responses of the teachers which fell outside the two previous categories. It was coded as O.

The classification of the various categories were based on **Episodes.** Episodes are designated as segments of classroom time that are completely devoid of any change throughout their nature duration (Adams, 1965). Based on this way of categorizing, if a student raised his/her hand to ask a question, this was considered as the beginning of an episode. However, if during this questioning time, the major concern shifted from one process to another, then another episode was considered as involved, and so on for a third, and other episodes. If possible, the names of the subjects involved in each episodes was also recorded to see if the teacher tended to interact more with some particular students. The observations were recorded in the schedule for observation of teachers (Appendix 9).

2. Observation of individual students. The observation of individual students was based on two main categories:

A. Substantive;

B. Non-substantive.

Substantive interactions occurred when a student interacted with one or more other students relating to the learning of computer programming, while non-substantive interactions referred to the type of interactions that did not involve the learning of programming.

There was a further sub-division within the substantive category, they were the verbal and non-verbal sub-categories. Verbal substantive interaction referred to the interactions between a student and one or more other students when the major contents of the interaction related to their learning contents. Non-verbal substantive interaction was said to occur when a student observed the other students' work or the interaction between the teacher and other students relating to the learning of programming.

A set of codes was developed:

SV - substantive verbal interaction;

SN - substantive non-verbal interaction;

NS - non-substantive interaction.

If possible, the names of the other subjects involved in the interactions were also recorded. These observations were recorded on the schedule of individual students (Appendix 10).

3. Observation of Groups. The observation of groups only focussed on two types of interactions:

A. verbal;

B. change of physical location.

Hence interaction was said to occur if a student had verbal interaction with one or more other students or if a student changed his/her physical location from his/her initial working location in order to observe others' work or interactions.

The duration of these interactions was also recorded along a time line as shown on the schedule for observation of groups (Appendix 11). It could be easily visualised that there might be none or more than one group interaction going on at the same time. A set of codes was hence used to identify the types of group interactions:

0 - no group interaction;

1 - one group was involved in interaction;

2 - two groups were involved in group interaction;

3 - 1 large group (i.e. more than half of the class, or more than two groups irrespective of size).

The procedures of observing. There was a total of 50 minutes within each session for individual work. Observations were carried out on three targets, namely, the teacher, individual student, and group. So each type of observation lasted a total of about 16 minutes with 2 minutes allowed for the switching of observing. These 16 minute intervals were further subdivided into 8 minutes each. The order of these observations was arranged on a random basis. First, the teacher was observed 8 minutes, then the individual student, finally , the overall group was observed. This cycle was repeated twice during each learning session. Observations were carried out during all the learning sessions for all the groups in an unobtrusive manner.

In one particular session, only two students could be observed. Given that there were 10 students in each group and that there were 20 sessions altogether for one group, it was decided to observe each student three times throughout the research project. The order of observing these students was established on a random basis.

Feedback was given to each of the three teachers on an individual basis after each session so as to reinforce or correct their behaviour in relation to either the process-oriented or the content-oriented approaches. This procedure was important in the context of this study as "the strategy of teaching" was one of the independent variables.

The observation of the individual students and the groups as a whole, though peripheral to the objectives of this project, helped to capture the dynamics of the interactions in a computing environment and hence provided a better understanding of the social context of a programming environment. It has been pointed out by many studies that students tended to interact much more than in a traditional classroom setting (cf Hawkins, 1983; Russell, 1983). However, data obtained from previous studies, i.e. prior to the conduct of the present study, often tended to be of an anecdotal nature. By quantifying the interactions of the students, more concrete evidence could be established which might help to point to possible future research in this area.

The teaching phase of this project was concluded in the middle of October. In all, each subject had 20 sessions of programming. They were given 12 teaching modules during this period. After the completion of these teaching modules, they then proceeded to do their personal projects based on the learning they had achieved. They were provided with worksheets to record their personal projects. In particular, the process-oriented group was given worksheets that impelled them to follow through the various steps of problem solving.

Post-testing was then conducted in a similar way as the pre-testing. It took three weeks for the post-tests to be completed. Students in the control group were then offered the opportunity to learn programming until the end of the academic year.

Note: Reliability of Observation

The observations of the teachers, individual students and groups were conducted by the researcher in consultation with one of his thesis supervisors. The observations were informally verified by this supervisor as an independent observer on a number of occasions.

Research hypotheses and data analyses

It was stated earlier in this chapter that the major purpose of this study is threefold: first, to evaluate any changes in the problem solving skills of learners as a result of learning different programming languages (LOGO and BASIC); second, to evaluate any changes in the problem solving skills of learners as a result of two different instructional methods (process-oriented and content-oriented); and, third, to examine systematically the different kinds of interactions within a programming-learning environment, in particular, two different types of interactions - those of individual students with teachers, and those among students.

Research reviewed in the preceding chapters suggest that LOGO may facilitate the development of problem solving skills especially when proper considerations have been given to the appropriate instructional conditions and social interactions within a LOGO learning environment. Recent works on problem solving and metacognition highlight the possibility of increasing transfer by focussing on the explicit teaching of problem solving and cognitive monitoring skills. However, the literature on problem solving also suggest that it is difficult to achieve far transfer than near transfer. Therefore, in this study, a number of problem solving measures, including those of near and far transfer nature were used to gauge the development of problem solving skills of the learners.

Accordingly this study was designed to test the two following clusters of research hypotheses.

Mathematics achievement and problem solving skills

The first cluster of hypotheses relates to the changes in mathematics achievement and various problem solving measures including those of near and far transfer nature. The following overall hypotheses were tested.

(A) . It was predicted that the learning of LOGO programming would facilitate the transfer of problem solving skills to a non-programming context that was of near-transfer nature (Tower of Hanoi) but not those of a far-transfer nature (mathematics achievement, Raven's Standard Progressive Matrices, WISC-R Picture Arrangement, WISC-R Block Design, WISC-R Object Assembly, Rule Naming Task, Torrance Test of Creative Thinking); and

(B) It was predicted that the degree of transfer from the LOGO environment to non-programming problem solving context of a near transfer nature (Tower of Hanoi) would be greater for children taught with the process-oriented approach compared with those taught with the content-oriented approach.

Testing of the hypotheses in the first cluster presented in this study was carried out by means of a 4 x 2 repeated measures analysis of variance design on the data collected during the pre- and post- testing periods. These data included measures of the subjects' mathematics achievement and problem solving skills.

In the event of significant overall group x testing occasion interaction, four *a priori* contrasts were planned for the comparison among groups:

(i) contrast between the two LOGO groups (LOGO process-oriented and content-oriented) and the two other groups (BASIC and control);

(ii) contrast between the LOGO process-oriented group and the LOGO content-oriented group;

(iii) contrast between the BASIC and control group; and

(iv) contrast between the LOGO-content oriented group and the BASIC group.

The .05 significance level ($\alpha = .05$) was employed in testing these hypotheses. It was planned to examine the treatment main effect only in the case of a non-significant group x testing occasion interaction.

Accordingly, the following specific hypotheses were set up:

Hypothesis One

It was predicted that there would not be significant group x testing occasion interaction among the groups in their mathematics achievement.

Hypothesis Two

It was predicted that there would not be significant group x testing occasion interaction among the groups in the scores measured by Raven's Standard Progressive Matrices.

Hypothesis Three

It was predicted that there would not be significant group x testing occasion interaction among the groups in the scores measured by WISC-R Picture Arrangement.

Hypothesis Four

It was predicted that there would not be significant group x testing occasion interaction among the groups in the scores measured by WISC-R Block Design.

Hypothesis Five

It was predicted that there would not be significant group x testing occasion interaction among the groups in the scores measured by WISC-R Object Assembly.

Hypothesis Six

It was predicted that there not would be significant group x testing occasion interaction among the groups in the scores measured by Rule Naming Task.

Hypothesis Seven

It was predicted that there would not be significant group x testing occasion interaction among the groups in the scores measured by Torrance Test of Creative Thinking.

Hypothesis Eight

It was predicted that there would be significant group x testing occasion interaction among the groups in the scores measured by Tower of Hanoi.

Classroom interactions

The second cluster of hypotheses relates to the interactions of the teachers and students during the process of learning. Three types of observations were carried out. The first observation was used to monitor the instructional methods used by teachers of the three programming group so only descriptive statistics of the

incidences of process-oriented, content-oriented, and other interactions between teachers and students will be presented.

The literature reviewed in Chapter 4 suggested that LOGO would facilitate social interactions among the learners. Moreover, the process-oriented instruction approach was used to encourage more interactions among the learners. Thus it is reasonable to expect that there would be more interactions among subjects in the process-oriented group than in the content-oriented group. In light of these considerations, the following hypotheses were tested in relation to the second and the third observation.

Hypothesis Nine

It was predicted that there would be more group interaction in the LPO group than the LCO group, and that there would be more group interaction in the LCO group than the BASIC group.

Hypothesis Ten

It was predicted that there would be more substantive verbal and non-verbal interactions among subjects in the LPO group than those in the LCO group, and that there would be more substantive verbal and non-verbal interactions among subjects in the LCO group than those in the BASIC group.

The outcomes of this study and the results of these data analyses will now be presented in the following chapter.

CHAPTER SEVEN

RESULTS

This chapter presents the results of the study. First, sample characteristics regarding sex, age, reading comprehension, listening comprehension for the various subgroups are presented. This is then followed by the results related to the problem solving skills of the subjects. The final section of the chapter presents the data related to the classroom interactions of the teachers and the students.

Sample Characteristics

Sample characteristics for the subjects are reported in Table 7.1 and Table 7.2. Table 7.1 displays the age and sex distribution for subjects in all the four groups - the LOGO process-oriented (LPO), the LOGO content-oriented (LCO), the BASIC and the control groups. The reason for the unequal number of subjects in each group was due to the fact that of the original 80 subjects, seven left the school in the middle of the study. Therefore, the number of subjects at the completion of the study for the four groups were 17, 20, 17, and 19 respectively (Total N = 73). There were no significant age differences among the groups (Total group mean = 9.79, S.D. = .60). The age range of the subjects was from 8.42 to 10.92, fairly characteristic of the groups attending standard three and four classes in New Zealand schools. The original sex distribution of subjects across the sub-groups was reasonably homogenous. However, of the seven subjects who left the school, six of them were females (three from the LOGO process-oriented group, two from the BASIC group, and one from the control group), the male subject was from the BASIC group.

Sample characteristics were also considered with regard to the students' listening comprehending and reading comprehension abilities, measured by the Progressive Achievement Tests before the intervention. Table 7.2 displays the scores of the subjects in these two tests prior to the intervention. One way analyses of variance of these scores revealed that there were no significant differences among the groups - listening comprehension ($F(3,69) = .52$, $p < .69$), and reading comprehension ($F(3,69) = .41$, $p < .75$).

Table 7.1

Age and Sex Distribution of Subjects

Group	Age		Number		
	Mean	S.D.	Males	Females	Total
Logo Process-Oriented	9.81	.69	12	5	17
Logo Content-Oriented	9.77	.64	11	9	20
Basic	9.84	.59	12	5	17
Control	9.74	.59	12	7	19
Total	9.79	.60	47	26	73

Table 7.2

Listening Comprehension and Reading Comprehension Score

Distribution of Subjects

Group	Listening Comprehension		Reading Comprehension	
	Mean	S.D.	Mean	S.D.
Logo Process-Oriented	27.76	8.43	21.35	9.89
Logo Content-Oriented	28.50	7.48	21.35	10.79
Basic	26.59	6.83	20.71	9.17
Control	29.11	6.12	18.11	9.24
Total	28.04	7.15	20.36	9.71

In sum, these results indicate that the groups were similar in composition with regard to age, reading comprehension and listening comprehension.

Analyses of mathematics achievement and problem solving measures

Hypotheses relating to mathematics achievement and the various problem solving measures were tested by performing a repeated measures analysis of variance using the SPSS package on VAX computer (SPSS Inc., 1988). In particular, a 4 (group) x 2 (testing occasion) two-way analysis of variance (ANOVA) was performed on all the problem solving dependent variables: Mathematics Achievement, the three subtests of the WISC-R, Torrance Test of Creative Thinking, Rule Naming Task, and Tower of Hanoi. It was also decided that in the event of significant overall group x testing occasion interaction, four *a priori* contrasts among the groups would be conducted (c.f. Chapter Six). The following sections will present the results of these analyses.

Mathematics Achievement

This section summarizes the results of the mathematics achievement. The results themselves are contained in Tables 7.3 and Tables 7.4 on page 146.

The results of the two-way analysis of variance with repeated measures indicate no significant interaction effects among groups with regard to pre- and post- test scores on the Mathematics Achievement Test of the Progressive Achievement Test. There was also no significant group main effect on this measure. However, the testing occasion main effect was significant, $F(1,69)$, $p < .000$ (Table 7.3). In examining the means of the groups and the total mean, it could be determined that all groups improved on their Mathematics Achievement (Table 7.4), the mean improvement being 4.86.

These findings lead to acceptance of null hypothesis one which states that there would not be significant group x testing occasion interaction among the groups in their mathematics achievement.

As predicted, these data indicate that there are no differences in mathematics achievement between the performance of subjects who learned LOGO programming using either the process-oriented approach or the content-oriented approach and those who either learned BASIC or were in the control group.

Table 7.3

ANOVA Summary Data for Mathematics Achievement

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S..	F-ratio	p
Between subjects					
Instructional Condition (A)	3	164.71	54.90	.33	.80
Error	69	11352.45	164.53		
Within subjects					
Testing Occasion (B)	1	858.92	858.92	69.15	.00*
A X B	3	13.21	4.40	.35	.786
Error	69	857.11	12.42		

* Significant Effects

Table 7.4

Means and Standard Deviations for Mathematics Achievement

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	30.18	9.23	34.71	9.83
LOGO Content-Oriented	31.60	10.34	37.00	9.64
BASIC	31.00	10.68	36.47	9.21
Control	29.63	7.73	33.68	8.33
Total	30.62	9.38	35.48	9.17

Raven's Standard Progressive Matrices

This section reports the results of the Raven's Standard Progressive Matrices. The results themselves are contained in Tables 7.5 and Tables 7.6 on page 148.

The results of the two-way analysis of variance with repeated measures indicate no significant interaction effects among groups with regard to pre- and post-test scores on the Raven's Standard Progressive Matrices. There was also no significant group main effect on this measure. However, the testing occasion main effect was significant, $F(1,69) = 37.85$, $p < .000$ (Table 7.5). In examining the means of the groups and the total mean, it could be seen that all groups improved on the Raven's Standard Progressive Matrices scores with the mean improvement being 3.80 (Table 7.6).

These results suggest acceptance of null hypothesis two which states that there would not be significant group x testing occasion interaction among the groups in the scores measured by Raven's Standard Progressive Matrices. In other words, subjects who learned LOGO programming using either approaches did not perform better in Raven's Standard Progressive Matrices when compared to subjects in the BASIC or control groups.

Table 7.5

ANOVA Summary Data for Raven's Standard Progressive Matrices

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	307.05	102.35	.83	.482
Error	69	8519.25	123.48		
Within subjects					
Testing Occasion (B)	1	525.22	525.22	37.85	.000*
A X B	3	34.28	11.43	.82	.485
Error	69	957.37	13.87		

* Significant Effects

Table 7.6

Means and Standard Deviations for Raven's Standard Progressive Matrices

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	38.65	7.86	41.29	5.89
LOGO Content-Oriented	39.20	9.31	42.30	9.70
BASIC	35.65	8.93	40.06	9.52
Control	34.79	8.01	39.84	5.87
Total	37.10	8.60	40.90	7.90

WISC-R

This section reviews the results of the WISC-R. The results themselves are contained in Tables 7.7, 7.8, 7.9, 7.10, 7.11 and 7.12 on pages 150, 151 and 152 respectively.

The results of the two-way analysis of variance with repeated measures indicate no significant interaction effects among groups with regard to pre- and post-test scores on the three sub-tests of WISC-R, namely, Picture Arrangement, Block Design and Object Assembly. There was also no significant group main effects on these three measures. However, the testing occasion main effects were significant. For Picture Arrangement, it was $F(1,69) = 32.48$, $p < .000$ (Table 7.7). For Block Design, it was $F(1,69) = 35.39$, $p < .000$ (Table 7.9). For Picture Arrangement, it was $F(1,69) = 41.92$, $p < .00$ (Table 7.11). In examining the means of the groups and total means for each of the three measures, it could be seen that all groups improved on the three sub-tests (Tables 7.8, 7.10, & 7.12). The improvement of each of the measures were - Picture Arrangement, 4.86; Block Design, 4.69; and Object Assembly, 2.93 respectively.

Findings with the three sub-tests of WISC-R lead to acceptance of null hypotheses three, four and five which state that the pre- versus post-test comparisons would not be different across the four groups in this study. These results confirm that subjects who learned LOGO programming using either the process-oriented or the content-oriented approach would not perform any better in the three WISC-R sub-tests when compared to the BASIC or the control group.

Table 7.7

ANOVA Summary Data for WISC-R Picture Arrangement

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	377.04	125.95	1.96	.128
Error	69	4436.79	64.30		
Within subjects					
Testing Occasion (B)	1	858.64	858.64	32.48	.000*
A X B	3	43.30	14.43	.55	.653
Error	69	1824.02	26.44		

* Significant Effects

Table 7.8

Means and Standard Deviations for WISC-R Picture Arrangement

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	27.71	7.36	31.24	6.01
LOGO Content-Oriented	29.20	6.10	33.25	5.53
BASIC	28.00	8.55	33.77	3.49
Control	24.16	9.81	30.26	4.90
Total	27.26	8.12	32.12	5.19

Table 7.9

ANOVA Summary Data for WISC-R Block Design

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	258.18	86.06	.26	.850
Error	69	22414.33	324.85		
Within subjects					
Testing Occasion (B)	1	791.83	791.83	35.39	.000*
A X B	3	22.02	7.34	.33	.805
Error	69	1543.66	22.37		

* Significant Effects

Table 7.10

Means and Standard Deviations for WISC-R Block Design

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	34.39	11.44	39.18	12.57
LOGO Content-Oriented	34.35	13.88	38.95	12.99
BASIC	34.41	14.81	38.06	14.25
Control	30.68	12.47	36.53	12.73
Total	33.47	13.05	38.16	12.88

Table 7.11

ANOVA Summary Data for WISC-R Object Assembly

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	244.48	81.49	2.34	.08
Error	69	2598.47	34.76		
Within subjects					
Testing Occasion (B)	1	315.80	315.80	41.92	.00*
A X B	3	21.51	7.17	.95	.42
Error	69	519.82	7.53		

* Significant Effects

Table 7.12

Means and Standard Deviations for WISC-R Object Assembly

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	21.35	5.15	25.59	4.12
LOGO Content-Oriented	22.95	3.62	25.90	4.24
BASIC	23.41	4.94	25.71	5.00
Control	20.21	5.16	22.53	4.49
Total	21.97	4.80	24.90	4.60

Rule Naming Task

This section reviews the results of the Rule Naming Task. The results themselves are contained in Tables 7.13, 7.14, 7.15 and 7.16 on pages 154, and 155 respectively.

Before comparing the groups' performance with the Rule Naming Task, the scores for both measures, number of trials and number of errors, were first correlated and no significant correlation was found. Hence a 4 x 2 two-way analysis of variance was conducted with each of the measures separately.

The results of the two-way analysis of variance with repeated measures indicate no significant interaction effects among groups with regard to pre- and post-test scores on both measures with the Rule Naming Task, viz., number of trials and number of errors. There was also no significant group main effect on these two measures. However, the testing occasion main effects for both measures were both significant; for the number of errors, it was $F(1,69) = 29.11$, $p < .00$ (Table 7.13); for the number of trials, it was $F(1,69) = 24.16$, $p < .000$ (Table 7.15). In examining the mean of the number of errors, it could be seen that all groups improved on the number of errors for the Rule Naming Task, the mean being 6.70 (Table 7.14), that is, there was a decrease in the mean number of errors made by the subjects during completion of the tasks. The total number of trials by all groups also improved by 12.06 (Table 7.16), i.e., the subjects required less number of trials to complete the tasks.

The above findings lead to acceptance of null hypothesis six which states that there would not be significant group x testing occasion interaction among the groups in the scores measured by Rule Naming Task. In other words, subjects who learned LOGO using either the process-oriented approach or the content-oriented approach did not perform any better in relation to the total number of trials to criterion, or the number of errors when compared to subjects in the BASIC or control groups.

These results confirmed the prediction that subjects who learned of LOGO, irrespective of instructional methods, would not perform any better than subjects who learned BASIC or were in the control group, either in the total number of trials criterion, or in the total number of errors when solving these tasks.

Table 7.13

ANOVA Summary Data for Rule Naming Task - Number of Errors

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	486.78	162.26	.87	.46
Error	69	12864.97	186.45		
Within subjects					
Testing Occasion (B)	1	1661.64	1661.64	29.11	.00*
A X B	3	32.09	10.70	.19	.905
Error	69	3938.60	57.08		

* Significant Effects

Table 7.14

Means and Standard Deviations for Rule Naming Task - Number of Errors

Pre- Versus Post Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	22.18	12.56	14.64	8.57
LOGO Content-Oriented	20.15	11.08	14.75	10.14
BASIC	26.18	11.96	18.47	12.53
Control	23.16	11.34	16.74	9.73
Total	22.81	11.67	16.11	10.22

Table 7.15

ANOVA Summary Data for Rule Naming Task - Number of Trials to Criterion

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	1458.00	486.00	.64	.595
Error	69	52809.24	765.35		
Within subjects					
Testing Occasion (B)	1	5246.86	5246.86	24.16	.000*
A X B	3	34.15	11.38	.05	.984
Error	69	14986.74	217.20		

* Significant Effects

Table 7.16

Means and Standard Deviations for Rule Naming Task - Number of Trials to Criterion

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	75.53	20.31	63.24	21.15
LOGO Content-Oriented	72.50	23.19	59.05	27.06
BASIC	79.94	21.55	68.77	25.59
Control	77.11	16.04	65.95	20.24
Total	76.14	20.21	64.08	23.53

Torrance Test of Creative Thinking

This section relates the results of the Torrance Test of Creative Thinking. The results themselves are contained in Tables 7.17, 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.25 and 7.26 on pages 157, 158, 159, 160 and 161 respectively.

The results of the two-way analysis of variance with repeated measures indicate no significant interaction effects among groups with regard to pre- and post-test scores on the total score as well as the four component scores of the Torrance Test of Creative Thinking which were Fluency, Flexibility, Originality, and Elaboration. There were also no significant main effects on the total and the four component scores. However the testing occasion main effects were significant:

Fluency - $F(1,69) = 19.89$, $p < .000$ (Table 7.17);

Flexibility - $F(1,69) = 9.77$, $p < .003$ (Table 7.19);

Originality - $F(1,69) = 27.27$, $p < .000$ (Table 7.21);

Elaboration - $F(1,69) = 11.04$, $p < .001$ (Table 7.23); and

Total score - $F(1,69) = 35.79$, $p < .000$ (Table 7.25).

In examining the means of the total and the four component scores, it could be seen that all groups improved with the mean improvement being:

Fluency - 3.60 (Table 7.18);

Flexibility - 1.93 (Table 7.20);

Originality - 9.25 (Table 7.22);

Elaboration - 12.93 (Table 7.24); and

Total score - 27.68 (Table 7.26).

These results lead to acceptance of null hypothesis seven that there would not be significant group x testing occasion interaction among the groups in the scores measured by Torrance Test of Creative Thinking. In other words, the performance of subjects in the Torrance Test of Creative Thinking did not differ irrespective of which groups they were in.

The above findings confirm the prediction that subjects who learned LOGO programming did not perform any better than subjects in the other two groups. However, it is interesting to note that the gain scores in both fluency and elaboration for all three programming groups were significantly better than those of the control group (cf Table 7.18 & Table 7.22).

Table 7.17

ANOVA Summary Data for Torrance Test of Creative Thinking - Fluency

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	177.64	59.21	1.36	.264
Error	69	3013.53	43.67		
Within subjects					
Testing Occasion (B)	1	480.37	480.37	19.89	.000*
A X B	3	130.89.	43.63	1.81	.154
Error	69	1666.85	24.16		

* Significant Effects

Table 7.18

Means and Standard Deviations for Torrance Test of Creative Thinking - Fluency

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	18.65	6.57	23.18	6.65
LOGO Content-Oriented	17.75	6.09	22.70	5.47
BASIC	17.82	5.19	22.47	6.38
Control	22.63	5.97	23.05	4.01
Total	19.25	6.20	22.85	5.55

Table 7.19

ANOVA Summary Data for Torrance Test of Creative Thinking - Flexibility

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	69.68	23.23	.66	.577
Error	69	2412.10	34.96		
Within subjects					
Testing Occasion (B)	1	135.87	135.87	9.77	.003*
A X B	3	6.37	2.12	.15	.928
Error	69	959.96	13.91		

* Significant Effects

Table 7.20

Means and Standard Deviations for Torrance Test of Creative Thinking - Flexibility

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	15.59	5.30	17.00	5.01
LOGO Content-Oriented	14.85	5.40	16.85	4.92
BASIC	14.53	4.20	17.12	4.31
Control	16.63	5.25	18.37	4.88
Total	15.41	5.04	17.34	4.74

Table 7.21

ANOVA Summary Data for Torrance Test of Creative Thinking - Originality

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	636.72	212.24	1.02	.391
Error	69	14419.23	208.97		
Within subjects					
Testing Occasion (B)	1	3268.22	3268.22	27.27	.000*
A X B	3	622.12	207.37	1.73	.169
Error	69	8270.13	119.86		

* Significant Effects

Table 7.22

Means and Standard Deviations for Torrance Test of Creative Thinking - Originality

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	23.06	6.46	34.41	17.48
LOGO Content-Oriented	25.80	10.23	28.90	10.29
BASIC	24.18	8.78	38.35	17.23
Control	27.90	10.90	37.21	16.68
Total	25.33	9.33	34.58	15.67

Table 7.23

ANOVA Summary Data for Torrance Test of Creative Thinking - Elaboration

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	3715.08	1238.36	.83	.482
Error	69	102949.59	1492.02		
Within subjects					
Testing Occasion (B)	1	6244.30	6244.30	11.04	.001*
A X B	3	1039.51	346.50	.61	.609
Error	69	39023.82	565.56		

* Significant Effects

Table 7.24

Means and Standard Deviations for Torrance Test of Creative Thinking - Elaboration

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	79.24	31.12	96.06	36.32
LOGO Content-Oriented	75.30	30.60	90.05	29.67
BASIC	69.00	27.85	85.82	34.29
Control	72.26	32.22	76.32	34.11
Total	73.96	30.13	86.89	33.62

Table 7.25

ANOVA Summary Data for Torrance Test of Creative Thinking - Total

Pre- Versus Post- Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	1607.11	535.70	.22	.883
Error	69	169368.81	2454.62		
Within subjects					
Testing Occasion (B)	1	28819.44	28819.44	35.79	.000*
A X B	3	2785.50	928.50	1.15	.334
Error	69	55559.38	805.21		

* Significant Effects

Table 7.26

Means and Standard Deviations for Torrance Test of Creative Thinking - Total

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	136.53	39.22	170.65	45.75
LOGO Content-Oriented	133.70	37.68	158.50	35.64
BASIC	125.53	31.90	163.77	49.91
Control	139.42	37.79	154.95	43.39
Total	133.95	36.41	161.63	43.12

Tower of Hanoi

This section reports the results of the Tower of Hanoi task. Four separate Tower of Hanoi problems were administered to the subjects both before and after the intervention phase of this study including the two-, three-, four-, and five-disk problems. However, the two-disk problem was used as a practice for the subjects and was not scored. Scoring for each of the other three problems was based on the total number of moves and the percentage of solving each of the sub-problems. The results of the Tower of Hanoi will be presented according to each of the three-disk, four-disk, and five-disk problems.

Three-disk problem

The following sections review the results of the three-disk problem and the two-disk sub-problem of the Tower of Hanoi. The results themselves are contained in Tables 7.27, 7.28, 7.29, and 7.30, on pages 164, 165, 167 and 168 respectively. The graphs comparing the pre versus post results are presented in Figures 7.1 and 7.2 on pages 166 and 169 respectively.

Before comparing the groups' performance on the three-disk problem, the scores for the number of moves and the percentage of solving two-disk sub-problems were first correlated and no significant correlation was found. Therefore, it was decided to conduct a 4 x 2 two way ANOVA with each of the measures.

Number of moves. The results of the two-way analysis of variance with repeated measures indicate significant interaction effects among groups with regard to pre- and post-test scores on the number of moves of the three-disk problem, $F(3,69) = 3.87$, $p < .013$) (Table 7.27). Therefore, four *a priori* contrasts among the four groups were conducted, the first three being orthogonal. The first contrast examined the two LOGO groups and the other two groups; the second contrast examined the LOGO process-oriented group (LPO) and the LOGO content-oriented group (LCO); the third contrast examined the BASIC group and the control group; and the fourth contrast examined the LOGO content-oriented group and the BASIC group. These planned comparisons revealed that the significant interaction was located in the interaction of the LPO and LCO versus BASIC and control groups,

$F(1,69) = 10.60$, $p < .002$ (Table 7.27). Examination of the graph in Figure 7.1 and the data in Table 7.28 reveals the significantly better performance of both the LPO and LCO groups in the post-test as contrast to the BASIC and control groups.

Two-disk sub-problem. Similar results were obtained with the two-disk sub-problem. Two-way analysis of variance with repeated measures indicate significant interaction effects among groups with regard to pre- and post-test scores, $F(3,69) = 3.3$, $p < .025$ (Table 7.29). Planned comparison revealed that the significant interaction was located in the interaction of the LPO and LCO versus BASIC and control groups, $F(1,69) = 9.34$, $p < .003$ (Table 7.29). Examination of the graph in Figure 7.2 and Table 7.30 reveals the significantly better performance of both the LPO and LCO groups in the post-test as contrast to the BASIC and control groups.

In summary, the findings with the three-disk problem of Tower of Hanoi supports hypothesis eight that there would be significant group x testing occasion interaction among the groups in the scores measured by Tower of Hanoi.

The planned comparisons indicate that subjects who learned to program with the LOGO language, irrespective of instructional methods, were better able to solve the three-disk Tower of Hanoi problem when compared to subjects in the other two groups.

Table 7.27

ANOVA Summary Data for Tower of Hanoi - three-disk problem

Number of moves

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	18.48	6.16	.97	.411
Error	69	437.19	6.34		
Within subjects					
Testing Occasion (B)	1	24.16	24.16	6.11	.016*
A X B	3	45.93	15.31	3.87	.013*
Error	69	272.73	3.95		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	41.88	41.88	10.60	.002*
BASIC vs Control	1	2.52	2.52	.64	.427
LPO vs LCO	1	1.53	1.53	.39	.536
LCO vs BASIC	1	10.10	10.10	3.29	.078

* Significant Effects

Table 7.28

Means and Standard Deviations for Tower of Hanoi - three-disk problem

Number of moves

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	9.29	3.31	7.12	.49
LOGO Content-Oriented	8.65	2.93	7.05	.22
BASIC	7.71	1.26	7.59	1.12
Control	8.26	1.88	8.90	3.78
Total	8.48	2.50	7.67	2.12

Figure 7.1

Tower of Hanoi

Three disk problem - Number of moves

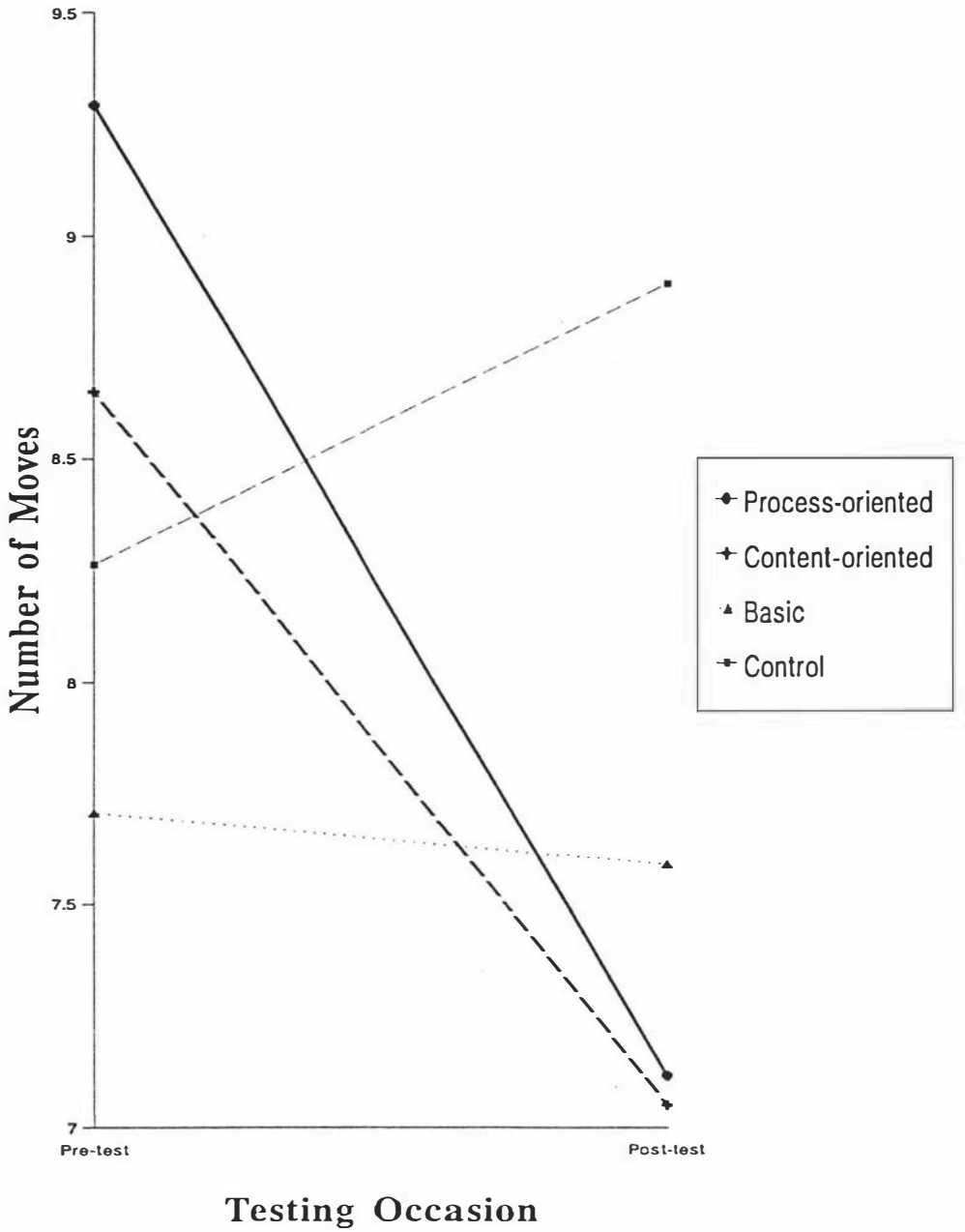


Table 7.29

ANOVA Summary Data for Tower of Hanoi - Three-disk problem

Two-disk sub-problem

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	.19	.06	.80	.449
Error	69	5.52	.08		
Within subjects					
Testing Occasion (B)	1	.40	.40	9.55	.003*
A X B	3	.42	.14	3.30	.025*
Error	69	2.92	.04		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	.40	.40	9.34	.003*
BASIC vs Control	1	.00	.00	.00	1.000
LPO vs LCO	1	.02	.02	.56	.455
LCO vs BASIC	1	.14	.14	3.14	.085

* Significant Effects

Table 7.30

Means and Standard Deviations for Tower of Hanoi - Three-disk problem

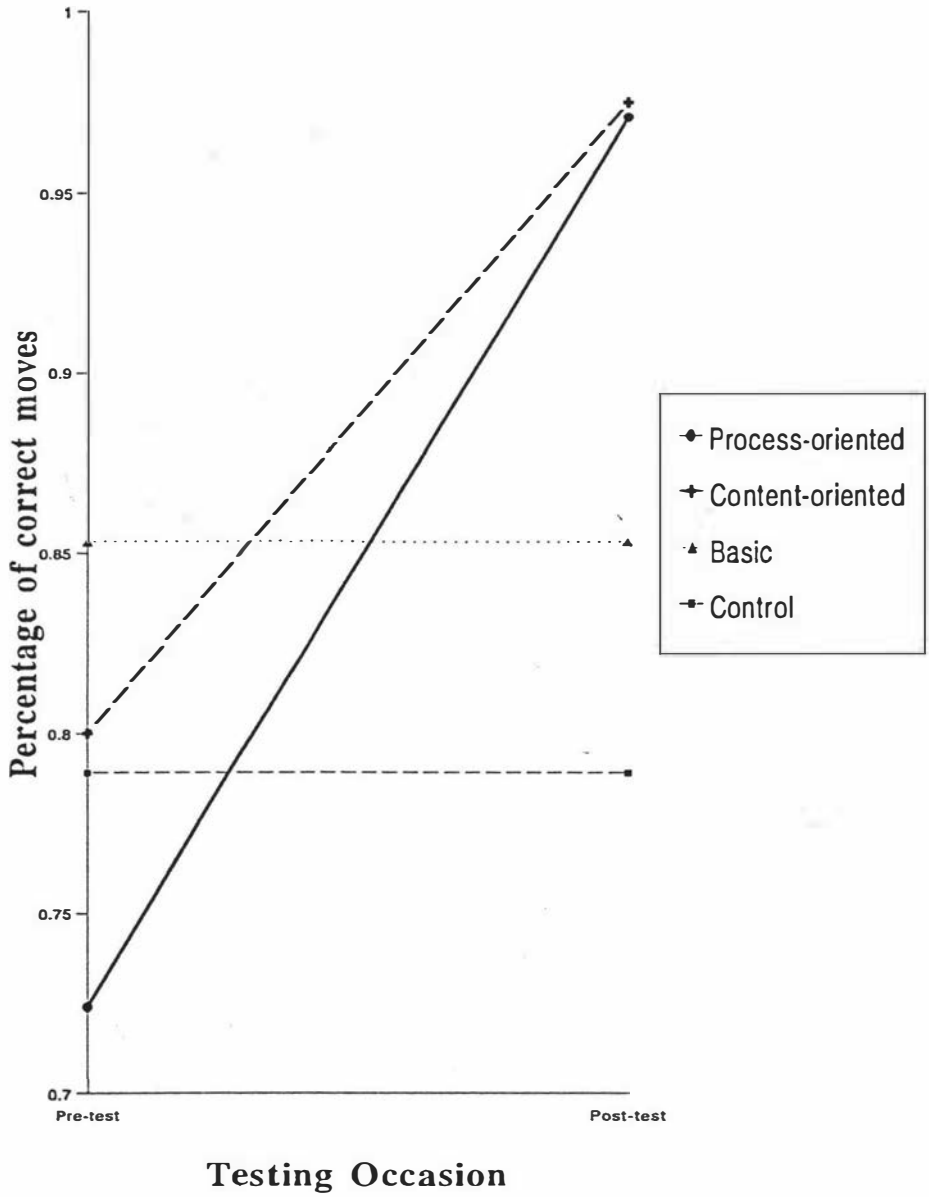
Two-disk sub-problem

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	.72	.33	.97	.12
LOGO Content-Oriented	.80	.25	.98	.11
BASIC	.85	.24	.85	.24
Control	.79	.30	.79	.30
Total	.79	.28	.90	.22

Figure 7.2

Three disk problem - two-disk sub-problems



Four-disk problem

The following sections report the results of the four-disk problem, the two-disk and three-disk sub-problems of the Tower of Hanoi. The results themselves are contained in Tables 7.31, 7.32, 7.33, 7.34, 7.35, and 7.36, on pages 172, 173, 175, 176, 178 and 179 respectively. The graphs comparing the pre versus post results are presented in Figures 7.3, 7.4 and 7.5 on pages 174, 177 and 180 respectively.

The intercorrelations among the scores of the number of moves, two-, and three-disk sub-problems were computed, and no significant correlation was found. Thus a separate 4 x 2 two-way ANOVA was conducted for each of the measures.

Number of moves. Two-way analysis of variance with repeated measures indicate interaction effects among groups with regard to pre- and post-test scores on the number of moves of the four-disk problem approached significance, $F(3,69) = 2.63$, $p < .057$. Thus the four *a priori* contrasts were conducted. Planned comparisons revealed that the significant interaction was located in two areas. The first area was the interaction between the LPO and LCO versus BASIC and control groups, $F(1,69) = 7.47$, $p < .008$; the second area was the interaction between the LCO versus the BASIC groups, $F(1,35) = 4.76$, $p < .036$ (Table 7.31). Examination of the graph in Figure 7.3 and the data in Table 7.32 reveals the significantly better performance of both LPO and LCO groups in the post-test results as contrast to the BASIC and control groups.

Two-disk sub-problem. In the examination of the two-disk sub-problem with two-way analysis of variance, it was found that the both interaction effect with regard to pre- and post-test scores were significant, $F(3,69) = 3.34$, $p < .024$ (Table 7.33). Planned comparison reveals that the interaction was located in the interaction of the LPO and LCO versus BASIC and control groups, $F(1,69) = 7.71$, $p < .007$ (Table 7.33). Examination of the graph in Figure 7.4 and Table 7.34 shows that both LPO and LCO performed significantly better than the BASIC and control groups.

Three-disk sub-problem. Similar results were obtained in the analysis of the three-disk sub-problem. The interaction effect with regard to pre- and post-test scores were significant, $F(3,69) = 3.32$, $p < .025$ (Table 7.35). Planned comparison uncovers that the interaction was also located in the interaction of the LPO and LCO versus BASIC and control groups, $F(1,69) = 9.40$, $p < .003$ (Table 7.35). The graph in Figure 7.5 and the data in Table 7.36 show that both LPO and LCO outperformed the BASIC and control groups in the post-test scores.

To sum up, the results with the four-disk problem of Hanoi support hypothesis eight which states that there would be significant group x testing occasion interaction among the groups in the scores measured by Tower of Hanoi. Planned comparison also revealed the interactions effects were due to the significantly better performance of both the LPO and LCO groups in the post-test when compared to the BASIC and control groups.

Table 7.31

ANOVA Summary Data for Tower of Hanoi - Four-disk problem

Number of moves

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	112.96	37.65	.79	.503
Error	69	3286.55	47.63		
Within subjects					
Testing Occasion (B)	1	164.59	164.59	11.90	.001*
A X B	3	109.06	36.35	2.63	.057
Error	69	954.29	13.83		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	103.34	103.34	7.47	.008*
BASIC vs Control	1	3.59	3.59	.26	.612
LPO vs LCO	1	2.12	2.12	.15	.697
LCO vs BASIC	1	79.13	79.13	4.76	.036*

* Significant Effects

Table 7.32

Means and Standard Deviations for Tower of Hanoi - Four-disk problem

Number of moves

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	23.41	5.68	19.94	6.23
LOGO Content-Oriented	23.05	4.84	18.90	3.31
BASIC	21.35	5.31	21.35	6.82
Control	23.68	5.73	22.79	6.08
Total	22.90	5.35	20.73	5.78

Figure 7.3

Four disk problem - number of moves

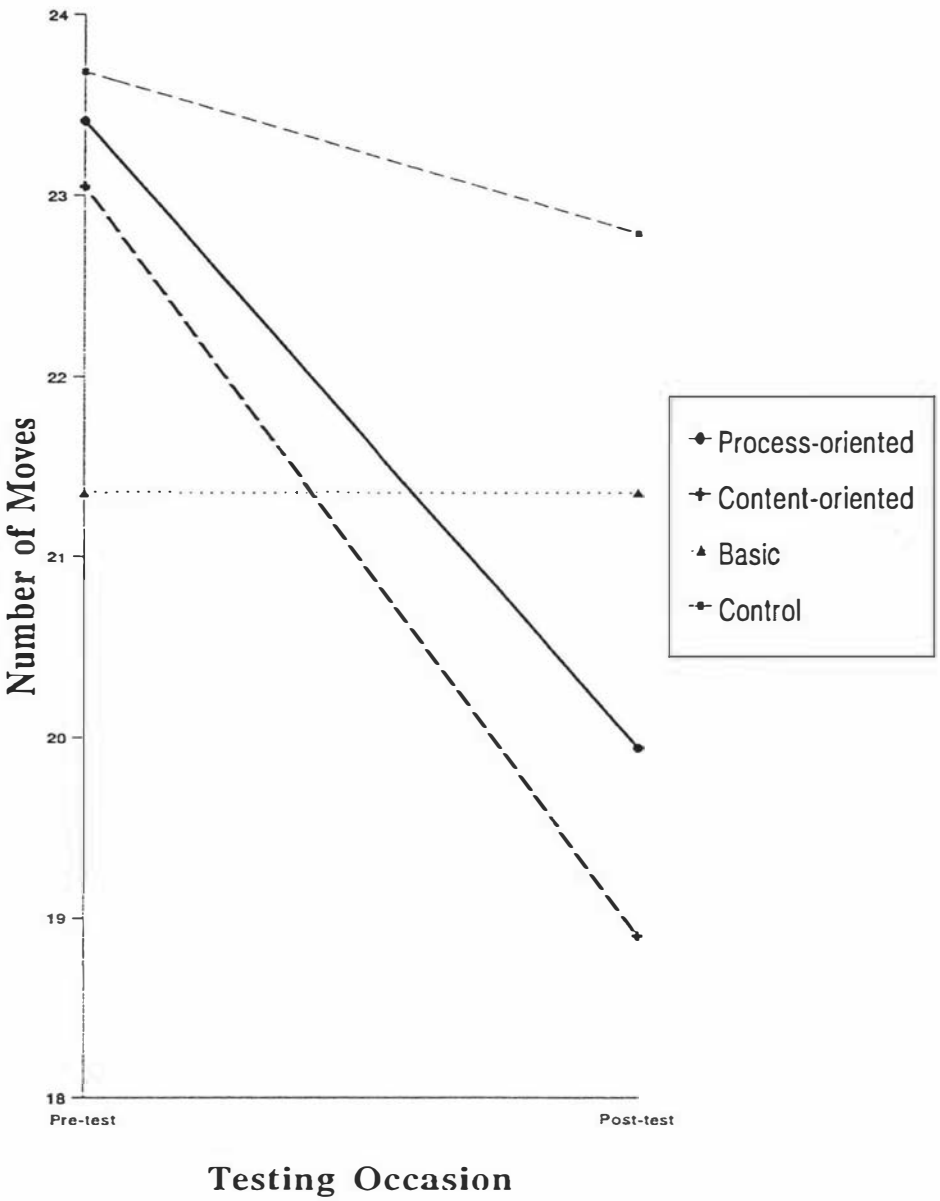


Table 7.33

ANOVA Summary Data for Tower of Hanoi - Four-disk problem

Two-disk sub-problem

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	.51	.17	3.02	.035
Error	69	3.87	.06		
Within subjects					
Testing Occasion (B)	1	.42	.42	19.83	.000*
A X B	3	.21	.07	3.34	.024*
Error	69	1.45	.02		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	.16	.16	7.71	.007*
BASIC vs Control	1	.01	.01	.24	.623
LPO vs LCO	1	.04	.04	2.08	.154
LCO vs BASIC	1	.02	.02	.74	.394

* Significant Effects

Table 7.34

Means and Standard Deviations for Tower of Hanoi - Four-disk problem

Two-disk sub-problem

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	.57	.14	.79	.20
LOGO Content-Oriented	.64	.22	.76	.14
BASIC	.63	.23	.68	.25
Control	.54	.16	.56	.21
Total	.59	.19	.70	.22

Figure 7.4

Four disk problem - two-disk sub-problem

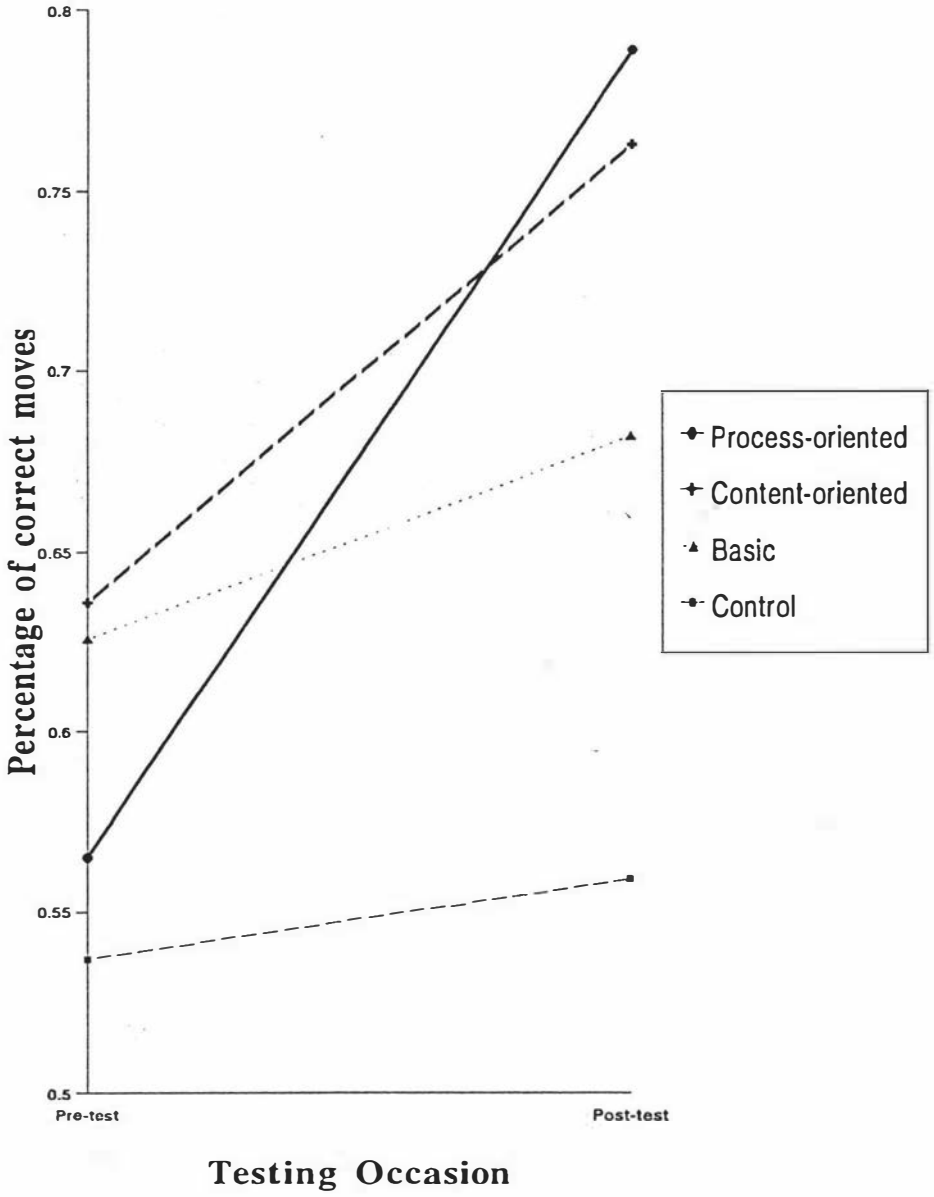


Table 7.35

ANOVA Summary Data for Tower of Hanoi - Four-disk problem

Three-disk sub-problem

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	.44	.15	.95	.422
Error	69	10.71	.16		
Within subjects					
Testing Occasion (B)	1	.87	.87	11.54	.001*
A X B	3	.75	.25	3.32	.025*
Error	69	5.17	.07		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	.71	.71	9.40	.003*
BASIC vs Control	1	.00	.00	.04	.839
LPO vs LCO	1	.04	.04	.51	.478
LCO vs BASIC	1	.29	.29	2.77	.105

* Significant Effects

Table 7.36

Means and Standard Deviations for Tower of Hanoi - Four-disk Problem

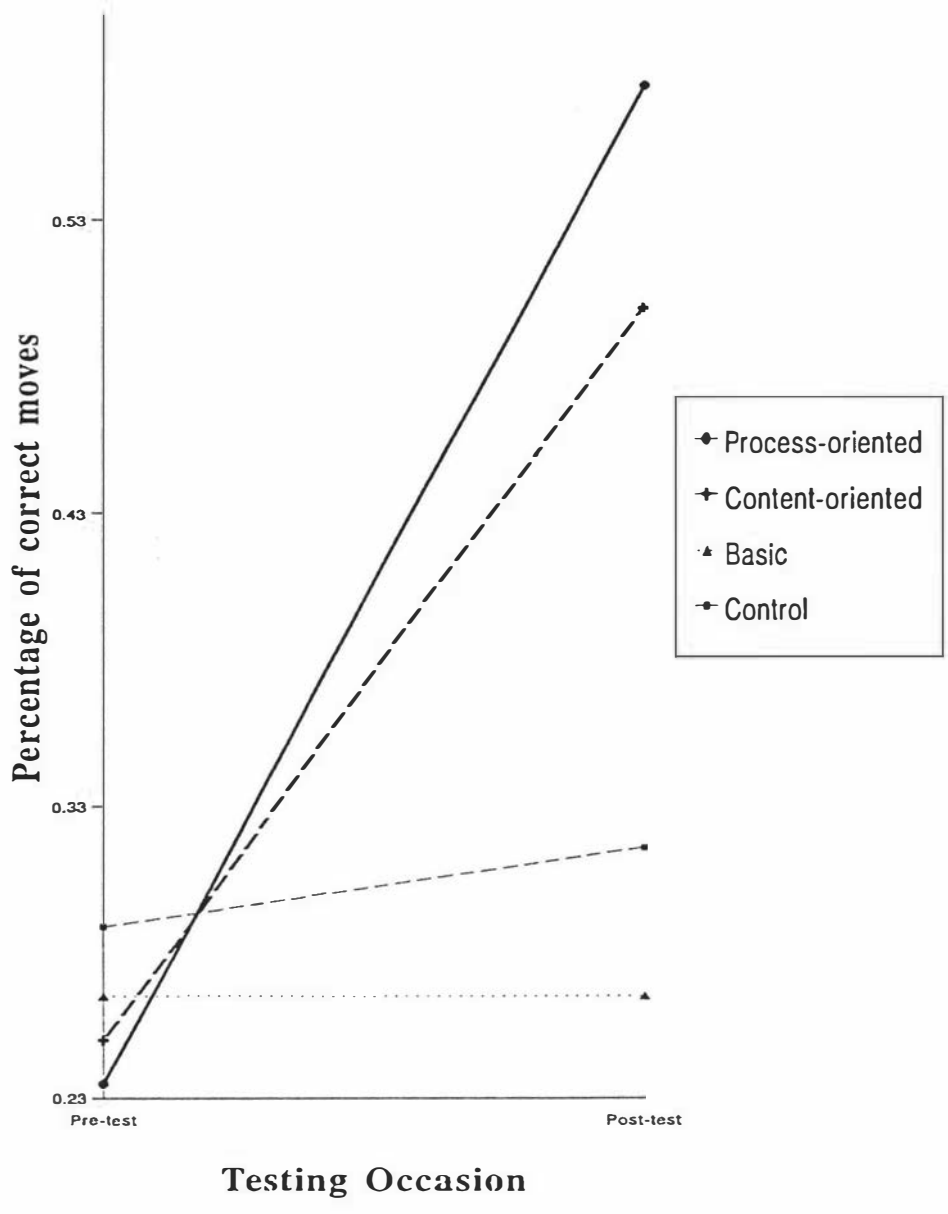
Three-disk Sub-problem

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	.24	.36	.58	.37
LOGO Content-Oriented	.25	.38	.50	.23
BASIC	.27	.36	.27	.40
Control	.29	.30	.32	.30
Total	.26	.35	.42	.34

Figure 7.5

Four disk problem - three-disk sub-problem



Five-disk problem

The following sections report the results of the five-disk problem, the two-disk, three-disk and four disk sub-problems of the Tower of Hanoi. The results themselves are contained in Tables 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, and 7.44 on pages 183, 184, 186, 187, 189, 190, 192 and 193 respectively. The graphs comparing the pre versus post results are presented in Figures 7.6, 7.7, 7.8 and 7.9 on pages 185, 188, 191 and 194 respectively.

Before comparing the performance of the four groups on the five-disk problem of the Tower of Hanoi, intercorrelations among the scores of the number of moves, two-, three-, and four-disk sub-problems were computed, it was found that there was no significant correlation. Hence a 4 x 2 ANOVA was conducted for each of the measures.

Number of moves. The results of the two-way analysis of variance with repeated measures indicate significant interaction effects among groups with regard to pre- and post-test scores on the number of moves of the five-disk problem, $F(3,69) = 2.89$, $p < .041$ (Table 7.37). Planned comparisons reveal that the significant interactions were located in two areas. The first one was with the interaction of the LPO and LCO versus BASIC and control groups, $F(1,69) = 4.16$, $p < .045$ (Table 7.37); the second one was located in the interaction of the LPO versus LCO groups, $F(1,69) = 4.47$, $p < .038$ (Table 7.37). Examination of the data in Table 7.38 and the graph in Figure 7.6 shows that it was the LPO group which significantly outperformed the other three groups in the post-test scores.

Similar patterns of results were obtained with the two-disk, three-disk, and four-disk sub-problems.

Two-disk sub-problem. Two-way analysis of variance with repeated measures indicate significant interaction effect among groups with regard to pre- and post-test scores, $F(3,69) = 7.77$, $p < .000$ (Table 7.39). Planned comparison revealed that the significant interaction was located in two areas. The first area was in the contrast with the LPO and LCO groups versus with the BASIC and control groups, $F(1,69) = 8.90$, $p < .004$ (Table 7.39); the second area was in the interaction with

contrast with the LPO versus LCO groups. Examination of the data in Table 7.40 and the graph in Figure 7.7 revealed that it was the LPO group which significantly outperformed all the other three groups in the post-test scores.

Three-disk sub-problem. Results of the two-way analysis of variance with repeated measures indicate significant interaction effect among groups with regard to pre- and post-test scores, $F(3,69) = 11.55$, $p < .000$ (Table 7.41). Moreover, planned comparison revealed that the interaction effect was located in two areas. The first area was located in the interaction of the LPO and LCO versus BASIC and control groups, $F(1,69) = 17.57$, $p < .000$ (Table 7.41); the second area was in the contrast between the LPO and LCO groups, $F(1,69) = 15.36$, $p < .000$ (Table 7.41). Examination of the data in Table 7.42 and the graph in Figure 7.8 revealed that it was the LPO group which performed significantly better than the other three groups.

Four-disk sub-problem. Results of the two-way analysis of variance indicate a significant interaction effect with regard to pre- and post-test scores in the percentage of correctly solving four-disk sub-problem, $F(3,69) = 4.93$, $p < .004$ (Table 7.43). Planned comparison using the four *a priori* contrasts revealed that the interaction effect was located in the contrast between the LPO and the LCO groups, $F(1,69) = 10.27$, $p < .002$ (Table 7.43). Examination of the graph in Figure 7.9 and the data in Table 7.44 reveals that there were differential effects of the LPO training condition. Subjects in the LPO group clearly outperformed subjects in the other three groups in the post-test scores on the percentage of correctly solving four-disk sub-problems.

Results from the four measures of the five-disk problem of Tower of Hanoi supports hypothesis eight which states that there would be significant interaction among the groups in the scores measured by Tower of Hanoi. Planned comparisons also revealed that the subjects who received process-oriented instructions in LOGO programming were able to perform significantly better than subjects in the other three groups. In other words, subjects who learned LOGO programming with a process-oriented approach were better able solve the more complicated five-disk

problem of Tower of Hanoi when compared to subjects with the other three groups. In particular, the general performance of the subjects in the LCO group was not any better than subjects in the BASIC and control groups.

Table 7.37

ANOVA Summary Data for Tower of Hanoi - Five-disk Problem

Number of Moves

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	929.48	309.83	.64	.590
Error	69	33276.63	482.27		
Within subjects					
Testing Occasion (B)	1	3244.29	3244.29	15.15	.000*
A X B	3	1859.23	619.74	2.89	.041*
Error	69	14773.73	214.11		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	891.76	891.76	4.16	.045*
BASIC vs Control	1	10.45	10.45	.05	.826
LPO vs LCO	1	957.02	957.02	4.47	.038*
LCO vs BASIC	1	27.58	27.58	.10	.758

* Significant Effects

Table 7.38

Means and Standard Deviations for Tower of Hanoi - Five-disk Problem

Number of Moves

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	64.29	20.02	42.41	10.23
LOGO Content-Oriented	60.85	22.23	53.40	17.29
BASIC	63.18	21.80	58.18	19.59
Control	59.58	19.25	56.11	15.77
Total	61.86	20.52	52.66	16.88

Figure 7.6

Five disk problem - number of moves

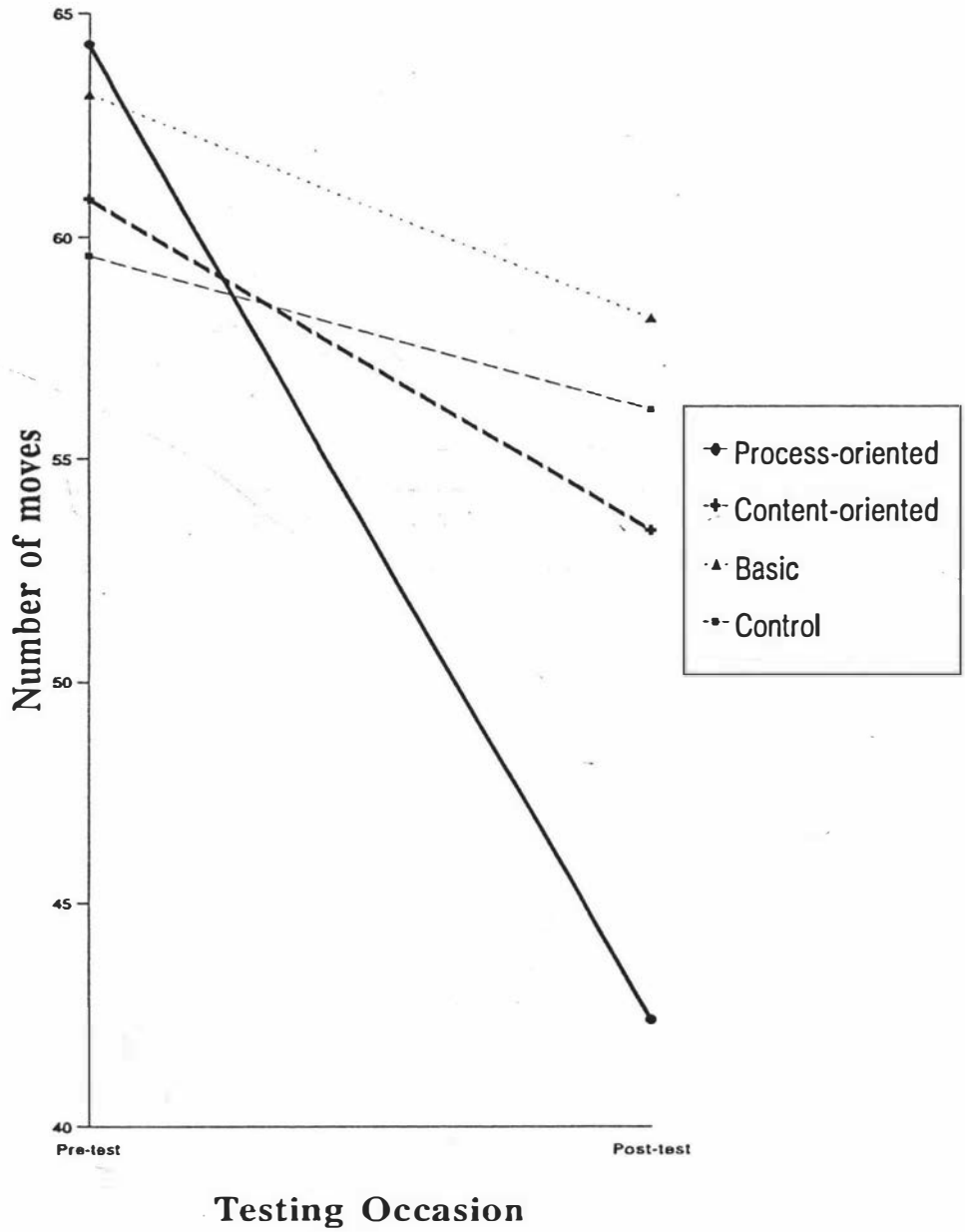


Table 7.39

ANOVA Summary Data for Tower of Hanoi - Five-disk Problem

Two-disk Sub-problem

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	.17	.06	.99	.402
Error	69	4.01	.06		
Within subjects					
Testing Occasion (B)	1	.45	.45	19.40	.000*
A X B	3	.54	.18	7.77	.000*
Error	69	1.60	.02		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	.21	.21	8.90	.004*
BASIC vs Control	1	.03	.03	1.30	.258
LPO vs LCO	1	.30	.30	13.09	.001*
LCO vs BASIC	1	.00	.00	.02	.882

* Significant Effects

Table 7.40

Means and Standard Deviations for Tower of Hanoi - Five-disk Problem

Two-disk Sub-problem

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	.45	.24	.77	.17
LOGO Content-Oriented	.52	.23	.58	.19
BASIC	.48	.18	.55	.22
Control	.54	.19	.53	.18
Total	.50	.21	.61	.21

Figure 7.7

Five disk problem - two-disk sub-problem

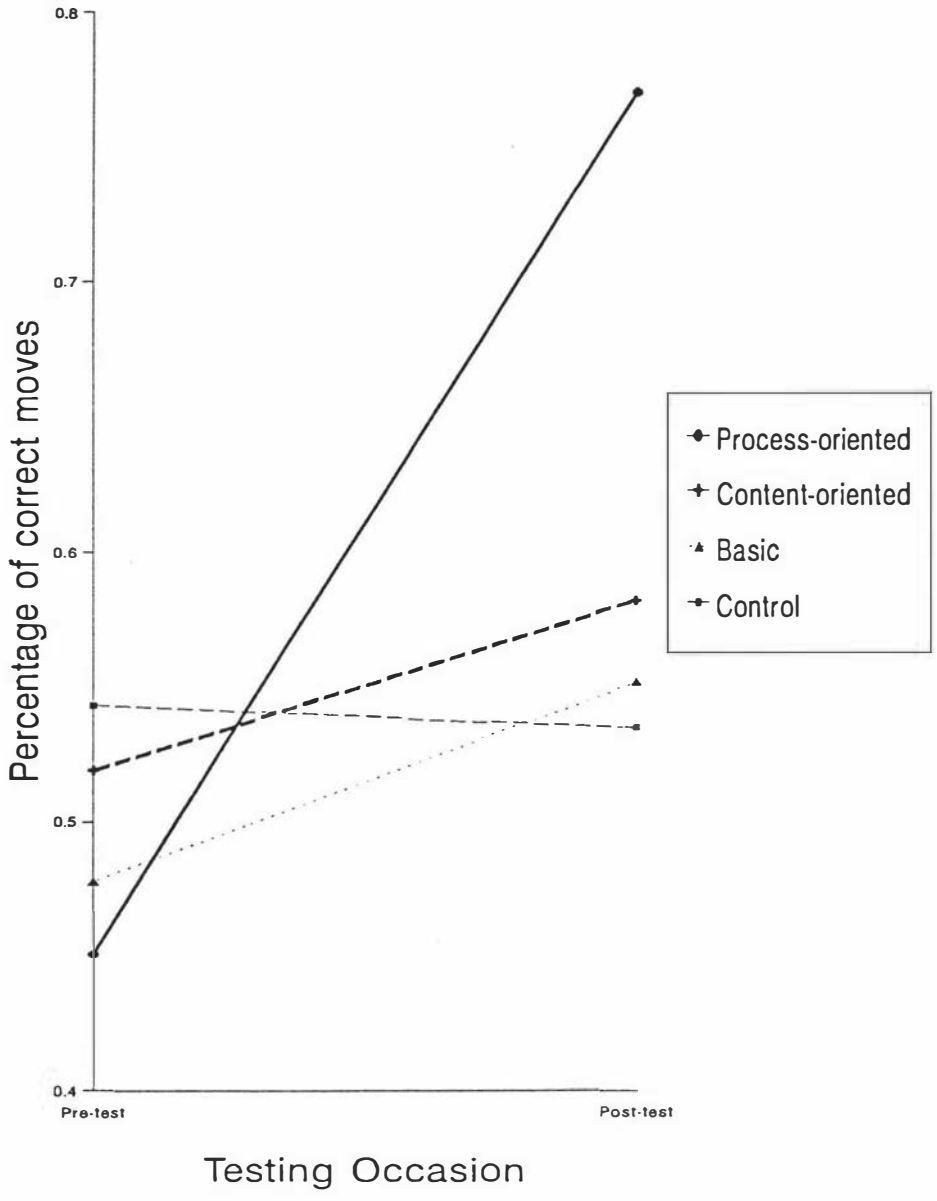


Table 7.41

ANOVA Summary Data for Tower of Hanoi - Five-disk Problem

Three-disk Sub-problem

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	.21	.07	.83	.680
Error	69	5.87	.09		
Within subjects					
Testing Occasion (B)	1	.37	.37	13.18	.001*
A X B	3	.98	.33	11.55	.000*
Error	69	1.95	.03		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	.50	.50	17.57	.000*
BASIC vs Control	1	.05	.05	1.71	.195
LPO vs LCO	1	.43	.43	15.36	.000*
LCO vs BASIC	1	.01	.01	.18	.670

* Significant Effects

Table 7.42

Means and Standard Deviations for Tower of Hanoi - Five-disk Problem

Three-disk Sub-problem

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	.14	.25	.52	.25
LOGO Content-Oriented	.25	.28	.32	.18
BASIC	.22	.22	.25	.29
Control	.28	.21	.21	.22
Total	.23	.24	.32	.26

Figure 7.8

Five disk problem - three-disk sub-problem

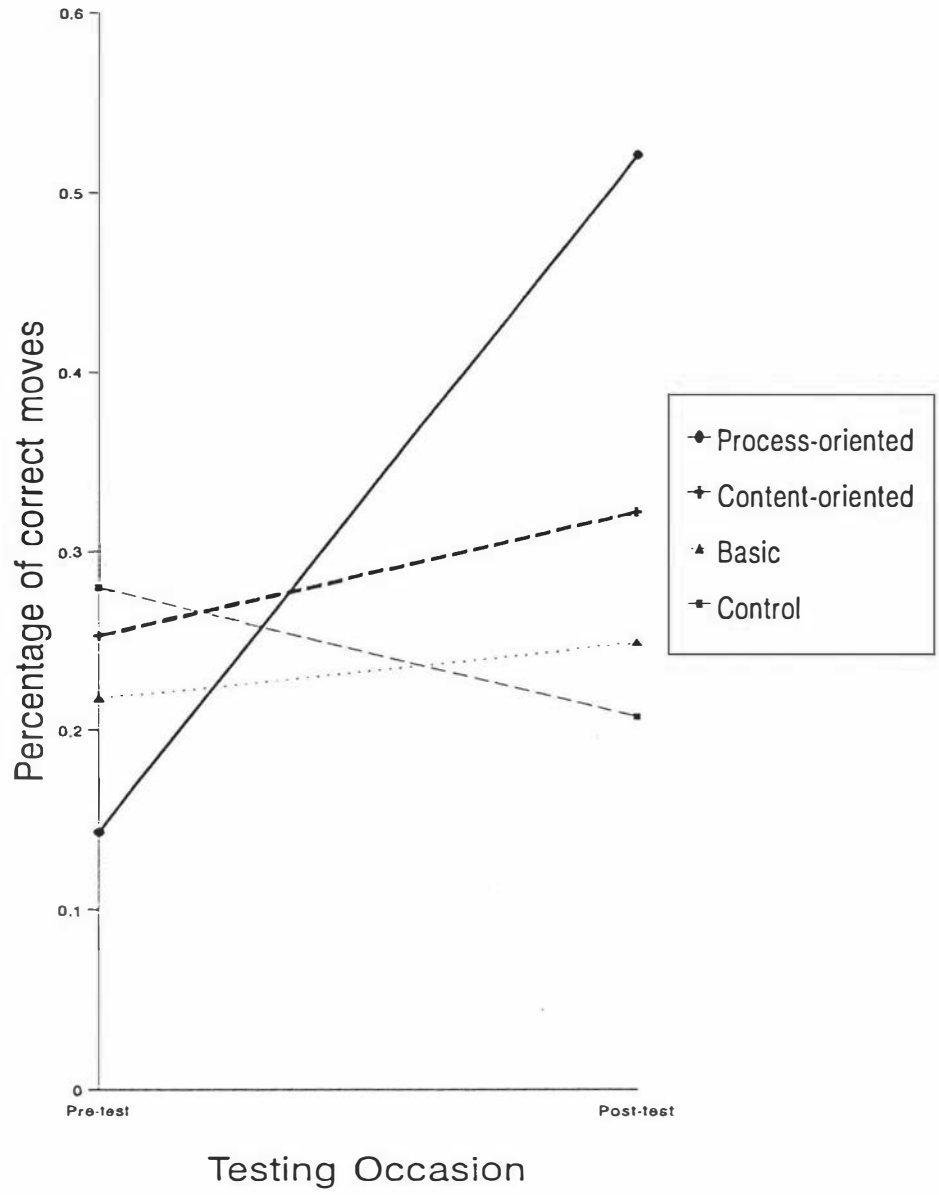


Table 7.43

ANOVA Summary Data for Tower of Hanoi - Five-disk Problem

Four-disk Sub-problem

Pre- Versus Post-Test Comparison

Source of variance	D.F.	S.S.	M.S.	F-ratio	p
Between subjects					
Instructional Condition (A)	3	.29	.10	1.39	.254
Error	69	4.89	.07		
Within subjects					
Testing Occasion (B)	1	.11	.11	3.51	.065
A X B	3	.45	.15	4.93	.004*
Error	69	2.09	.03		
a priori contrasts					
LPO & LCO vs BASIC & Control	1	.08	.08	2.69	.105
BASIC vs Control	1	.06	.06	1.84	.180
LPO vs LCO	1	.31	.31	10.27	.002*
LCO vs BASIC	1	.03	.03	.84	.365

* Significant Effects

Table 7.44

Means and Standard Deviations for Tower of Hanoi - Five-disk Problem

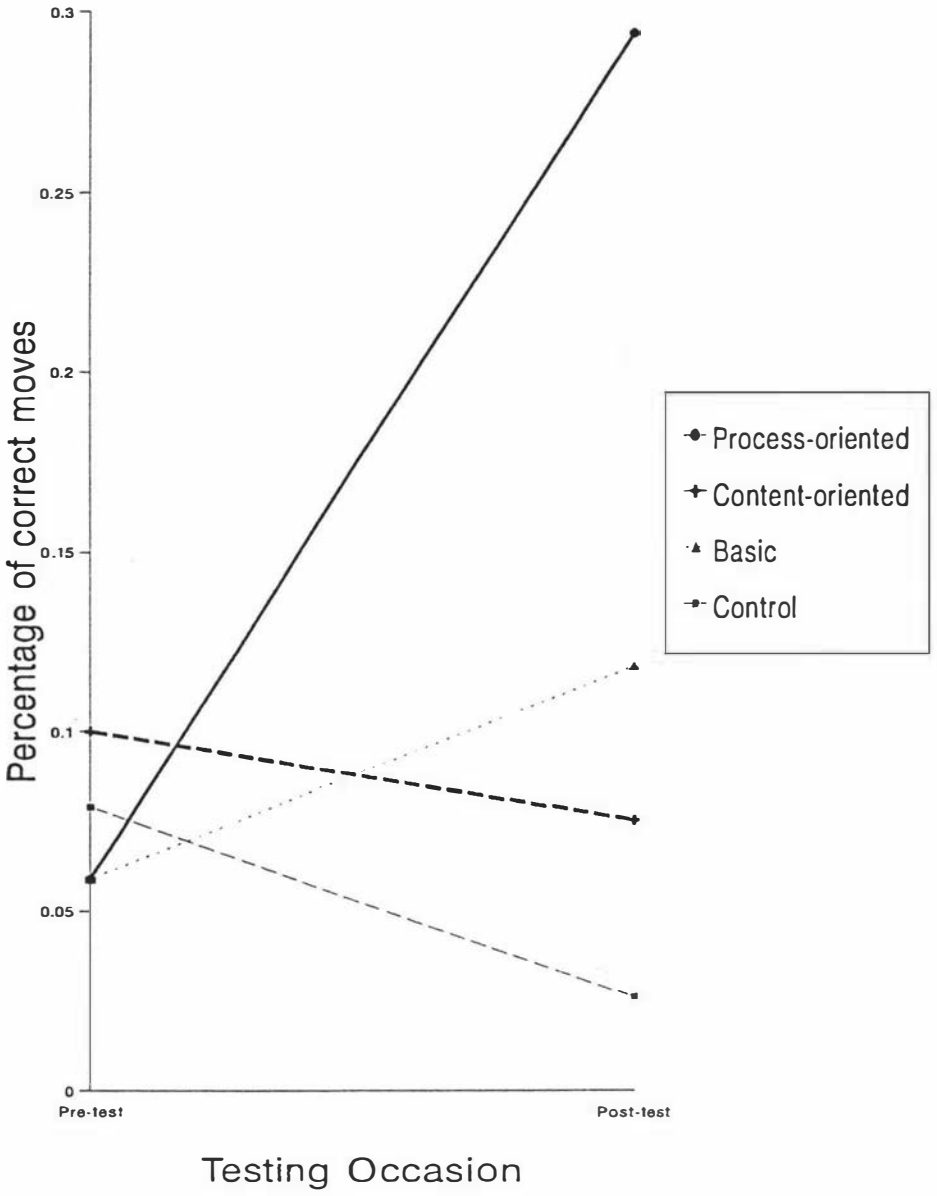
Four-disk Sub-problem

Pre- Versus Post- Test Comparison

Group	Pre-Test		Post-Test	
	Mean	S.D.	Mean	S.D.
LOGO Process-Oriented	.06	.24	.29	.31
LOGO Content-Oriented	.10	.26	.08	.18
BASIC	.06	.17	.12	.28
Control	.08	.19	.03	.12
Total	.08	.22	.12	.25

Figure 7.9

Five disk problem - four-disk sub-problem



Conclusion

In summary then, subjects in this study who learned LOGO programming, irrespective of instructional methods, did not perform any better than subjects in the BASIC and control groups in mathematics achievement and problem solving measures that were of a far-transfer nature, namely, Raven's Standard Progressive Matrices, the three sub-tests of the WISC-R, the Rule Naming Task, and the Torrance Test of Creative Thinking. However, subjects in both LOGO groups consistently outperformed subjects in the other two groups in the three-disk, four-disk problems of the Tower of Hanoi, but not in the five-disk problem. Therefore, the findings with mathematics achievement and various problem solving measures partially supports the overall hypothesis A which states that the learning of LOGO programming would facilitate problem solving in a non-programming context that was of near transfer nature (Tower of Hanoi) but not those of a far transfer nature (mathematics achievement, Raven's Standard Progressive Matrices, WISC-R Picture Arrangement, WISC-R Block Design, WISC-R Object Assembly, Rule Naming Task, Torrance Test of Creative Thinking).

The above findings also partially support the overall hypothesis B which states that the degree of transfer from the LOGO environment to non-programming problem solving context of a near transfer nature (Tower of Hanoi) would be greater in children taught with the process-oriented approach compared with those taught with the content-oriented approach. This is evident from the fact that subjects in the LOGO process-oriented group consistently outperformed the subjects in the LOGO content-oriented group in the more complicated five-disk problem of the Tower of Hanoi and its associated sub-problems but not in the simpler three-disk and four-disk problems of the Tower of Hanoi.

Classroom Interactions

Three major types of classroom interactions were observed: interactions of teachers with students, interactions of groups, and interactions of individual students with other students and teachers. The following sections will report the results of these three types of interactions.

Interactions of teachers with students

The major purpose of this observation related to the monitoring of the ways teachers taught in the three different groups, viz, LOGO process-oriented group (LPO), LOGO content-oriented group (LCO), and the BASIC group. Three major categories were used in the observation, that is, process-oriented, content-oriented, and other (which included episodes in relation to general administration and reinforcement of student learning). Table 7.45 displays the results of the observation. It could be determined from the data that the majority of the episodes in the LPO group focussed on the process-oriented interactions (61%), whereas the majority of episodes of the other two groups related to content-oriented interactions (62% for the LCO group and 74% for the BASIC group). These results provide further evidence to support the differences in the instructional methods used with the three groups in this study.

Table 7.45

Interaction of teachers with students

Episode	LPO		LCO		BASIC	
	Number	Percentage	Number	Percentage	Number	Percentage
Process-oriented	551	61	60	6	15	2
Content-oriented	107	12	632	62	500	74
Other	246	27	332	32	162	24
Total	904	100	1024	100	677	100

In the process-oriented group, the focus of teacher-student interactions was on the processes involved in learning the language and the steps in solving the related programming problems. When students encountered problems, the teachers tried to encourage them to think and to find out their own answers. The following examples illustrate some of the process-oriented interactions between teachers and students:

'What happened? What about the turtle? What does show turtle do?

'Make these predictions and then when your turn comes, see if your prediction is correct.'

'Check it carefully. Where do you think you have gone wrong?'

'How are you going to fix it? What was wrong?'

'What's missing from that one? You think about it.'

'Have you followed your plan through?'

'What do you do to get rid of those stuff?'

'Student: "I can't predict!!!"; Teacher: "Imagine you are a turtle..."'

'What do you want to do?'

'What do you have to do? You try it...'

It was through exchanges like these that students were encouraged to think, to experiment with different ideas, to plan their work systematically, to analyse their work and to monitor their own progress while learning to program.

On the other hand, in both the LOGO content-oriented and BASIC groups, the focus of the teacher-student interactions was on the content such as syntax of the languages and steps of solving programming problems. The following are typical examples of the content-oriented episodes:

'You need to leave a space between FD and the number.'

'You have to press ENTER after each command.'

'To hide the turtle, you type HIDETURTLE.'

'You typed too many 0's.'

'Move the turtle to the bottom of the screen. You can move it back or turn around. Then count the number of steps.'

'Get out of the Editor by pressing ESC.'

'Load your file back from the disk.'

'Put in the PRINT command.'

'Type CLS to clear the screen.'

'Put a quotation mark first, then type the next line.'

'These are string variables. Put a string after it.'

'You did not use the line numbers. Type this in first.'

In the content-oriented episodes, teachers always tried to provide students with direct answers to their questions instead of asking students to find out the answers themselves through critical thinking and self-exploration.

As expected in any classroom situation, a fair amount of interaction (between 24% to 32%) in all three groups related to the other category such administration and reinforcement of student learning. Some of the typical interactions are as follows:

'You can take these modules home.'

'Work through last week's module.'

'Use blue pen to put down what actually happened.'

'Start where you finished last time.'

'Read through the module.'

'Answer those questions that you can answer'

'Have you finished last week's module?'

'Yes, keep trying.'

'That's how you do it!'

'Yes, that's right.'

'You know what you are doing. I won't say a word.'

'That's good'

It could be determined from these data that (i) there were more interactions between the teachers and the students in both the LPO group (904 episodes) and the LCO (1024) group as contrast to the BASIC group (677 episodes); and (ii) there were substantially more process-oriented interactions between the teachers and the students in the LPO group during student learning.

Student Group Interactions

The major purpose of this observation was to find out the extent to which group interaction among students occurred. Four major categories of observation were used:

Zero - no group interaction;

One - one group was involved in interaction;

Two - two groups were involved in group interaction;

Three - 1 large group (i.e. more than half of the class, or more than two groups irrespective of size).

The results of these observations are displayed in Table 7.46. It should be noted that the numbers contained in this table refer to the number of episodes that occurred rather than the actual duration of interaction.

Table 7.46

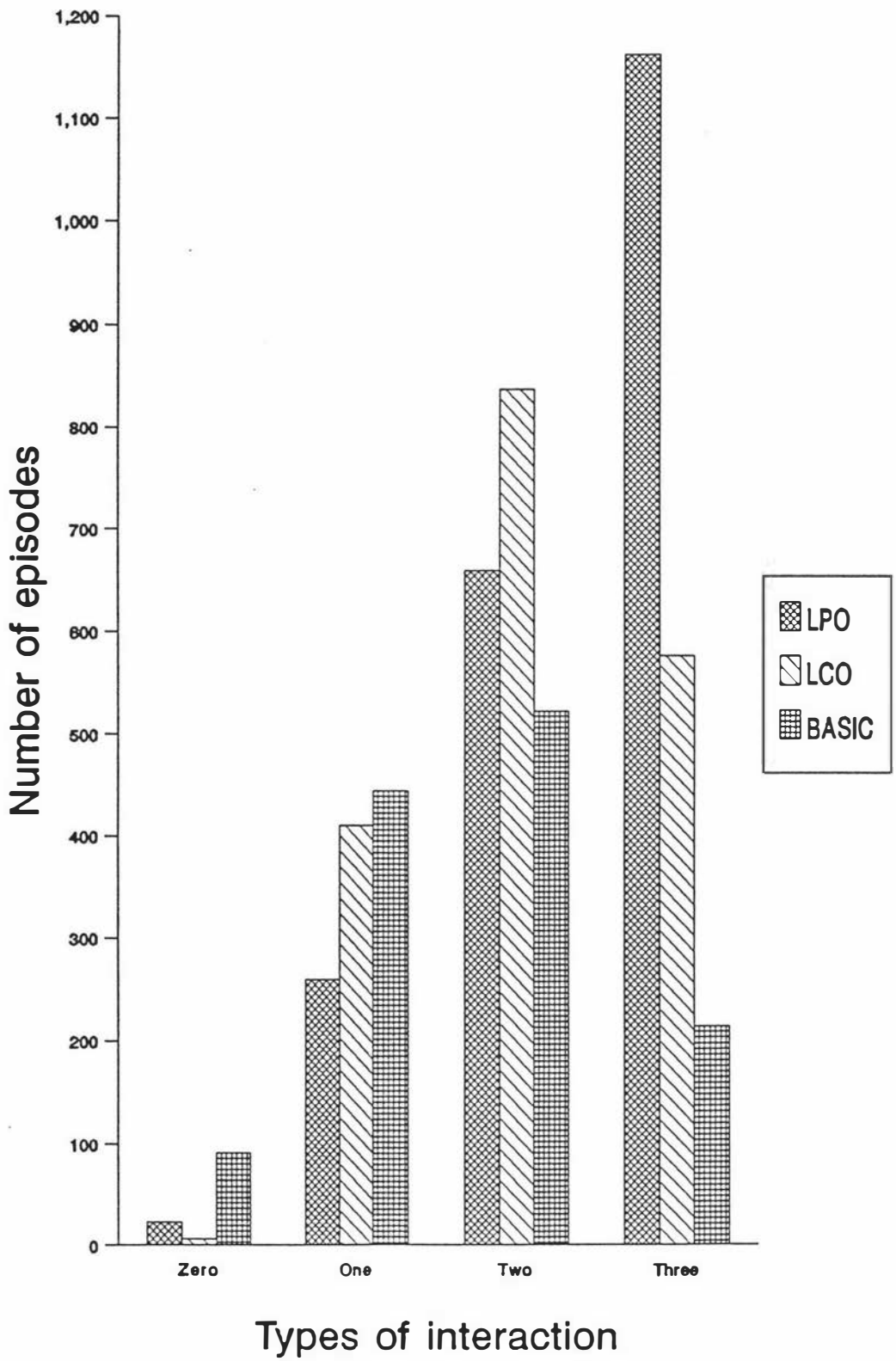
Student group interaction

Episode	LPO		LCO		BASIC	
	Number	Percentage	Number	Percentage	Number	Percentage
Zero	23	1.09	6	.33	91	7.16
One	259	12.32	410	22.42	444	34.93
Two	658	31.30	837	45.76	522	41.07
Three	1162	55.28	576	31.49	214	16.84
Total	2102	100.00	1829	100.00	1271	100.00

Similar to the observations made with teacher-student interactions, it could be determined from the data on group interactions that the two LOGO groups students were involved in a more dynamic learning environment when compared to students in the BASIC group. Several kinds of evidence could be found under this category of observation.

Figure 7.10

Student Group Interaction



First, there were more episodes of interactions among students in both the LOGO groups. In particular, the LPO group had 2102 episodes of interactions, the LCO group had 1829, and the BASIC group had 1271.

Second, 7% of episodes of the BASIC group were involved with no student group interactions as compared to only 1.09% with the LPO group and .33% with the LCO group.

Third, substantially more episodes of the two LOGO groups were involved with two or more groups of students interacting with each other. In particular, it was 86% for the LPO group and 77% for the LCO group compared with 53% with the BASIC group.

Fourth, 42% of the episodes for the BASIC group was involved with either no group interaction or only 1 group of students interacting as compared to 13.41% for the LPO group and 22.75% for the LCO group.

Although it was not possible to record all the group interactions, some of the examples below illustrate several kinds of group interactions that occurred. Quite often, the students were seen cooperating with each other when solving problems. For instance:

1. *"Let's work on drawing this robot together"*
   "Good idea. Why don't you draw the arms and the legs and I'll draw the rest".
2. *"What have I done wrong in drawing this star?"*
   "Look, the angle is wrong".
3. *"How are we going to get this roof into its proper place?"*
   "Let's try turning 30 degrees before drawing the roof".

Students were also seen engaging in resolving their conflict while attempting to arrive at a solution. Sometimes they were successful, but at times they were not. Following are typical of these episodes:

1. *"I think we should turn 60 degrees to draw this triangle"*
   "No, I think it should be 120 degrees"
   "Why don't you act as the turtle and I'll give you the commands to act out to see who is right"
   "Alright".

2. *"The width of the screen is 300"*

   "You are wrong. It is 310"

   "Far off! You are both wrong. It is 320"

   "How did you get 320?"

   "Let me show you".

3. *"How many degrees do you think we should turn to draw this star using the repeat command?"*

   "100 degrees"

   "No, I think it should be 120"

   "Well, why don't you use yours and I'll use mine".

There were also other group interactions that did not relate to the students' learning tasks. For example:

1. *"What did you watch on tele last night?"*

   "Those cartoons were quite interesting".

2. *"What are you going to do this weekend?"*

   "We are going to Wanganui".

In summary, it could be determined from the observations made on the group interactions of the three programming groups that the two LOGO groups were involved in more group interactions when compared to the BASIC group. As well, it could be judged that the LPO students were involved in more group interactions when compared to the LCO group. Therefore, hypothesis nine could be upheld that there would be more group interaction in the LPO group than the LCO group, and that there would be more group interaction in the LCO group than the BASIC group.

The records of some of the detailed group interactions illustrate that during the process of learning to program, students may be involved in shared problem solving, and resolution of conflict.

Individual student interactions

The major purpose of this observation was to find out how students interact with other students and the teachers. Three categories of observation were used with the following coding variable names:

(i) substantive verbal interactions (SV);

(ii) substantive non-verbal interactions (SN); and

(iii) non-substantive interactions (NS).

Before comparing the groups' various types of interactions, the frequencies of the three different types of interactions were first correlated and no significant correlation was found. Hence three separate analyses of variance were conducted with each type of interactions. The results will now be presented according to these three categories of observations.

Substantive verbal interactions

The one-way analysis of variance indicate that there was significant difference among the groups in relation to the occurrence of substantive verbal interactions, $F(2,51) = 11.92$, $p < .000$ (Table 7.47). It can be determined from Table 7.48 that it was in the LPO group where substantive verbal interactions occurred most. Also, the occurrence of substantive verbal interactions was higher in the LCO group than in the BASIC group.

Table 7.47

ANOVA Summary Data for Substantive Verbal Interactions

Source of variance	D.F.	S.S.	M.S..	F-ratio	p
Between Groups	2	481.91	240.96	11.92	.000*
Within Groups	51	1030.68	20.21		
Total	53	1512.59			

* Significant Effects

Table 7.48

Means and Standard Deviations for Substantive Verbal Interactions

Group	Number of Subjects	Episodes	
		Mean	S.D.
LOGO Process-Oriented	17	14.12	5.12
LOGO Content-Oriented	20	10.40	5.13
BASIC	17	6.59	2.62
Total	54	10.37	5.34

In addition to some of the examples cited in the last section, following are some typical examples of the substantial verbal interaction:

"Think, turtle think!!!"

"Something is wrong with your program. You should have FORWARD 50 instead"

"We got it!! Let's see it again"

"I know. If there is something wrong, I should go back to my plan and look at it carefully".

Substantive non-verbal interactions

Similarly, the one-way analysis of variance indicate that there was significant difference among the groups in relation to the occurrence of substantive non-verbal interactions, $F(2,51) = 7.33$, $p < .002$ (Table 7.49). From Table 7.50, it can be determined that the LPO group had the highest incidence of substantive non-verbal interactions. Also, the mean number of substantive non-verbal interactions was higher in the LCO group than in the BASIC group.

Table 7.49

ANOVA Summary Data for Substantive Non-Verbal Interactions

Source of variance	D.F.	S.S.	M.S..	F-ratio	p
Between Groups	2	43.63	21.82	7.33	.002*
Within Groups	51	151.87	2.98		
Total	53	195.50			

* Significant Effects

Table 7.50

Means and Standard Deviations for Substantive Non-Verbal Interactions

Group	Number of Subjects	Episodes	
		Mean	S.D.
LOGO Process-Oriented	17	2.65	1.94
LOGO Content-Oriented	20	2.25	2.00
BASIC	17	.53	1.01
Total	54	1.83	1.92

There were two major types of substantive non-verbal interactions. First, a student would watch what happened on a classmate's computer screen. Second, a student would listen/watch what other classmates and/or the teacher were doing.

Non-substantive interactions

The one-way analysis of variance indicates that there was no significant difference among the groups in relation to the occurrence of non-substantive interactions, $F(2,51) = .21$, $p < .810$ (Table 7.51). The means and standard deviations of the occurrence of non-substantive behaviour for the three groups are

displayed in Table 7.52. It can be determined from Table 7.52 that the LPO group had the least number of non-substantive interactions but the difference with the other groups was not significant.

Table 7.51

ANOVA Summary Data for Non-Substantive Interactions

Source of variance	D.F.	S.S.	M.S..	F-ratio	p
Between Groups	2	5.79	2.89	.21	.810
Within Groups	51	696.36	13.65		
Total	53	702.15			

Table 7.52

Means and Standard Deviations for Non-Substantive Interactions

Group	Number of Subjects	Episodes	
		Mean	S.D.
LOGO Process-Oriented	17	2.71	2.69
LOGO Content-Oriented	20	3.45	4.16
BASIC	17	3.35	3.97
Total	54	3.19	3.64

Typical non-substantive interactions involved students talking about something that were unrelated to the learning of programming. The examples cited in group interaction (watching tele, and going away during the weekend) were quite typical of these episodes.

To sum up, the results on classroom interaction support hypothesis ten which states that there would be more substantive verbal and non-verbal interactions among

subjects in the LPO group than those in the LCO group, and that there would be more substantive verbal and non-verbal interactions among subjects in the LCO group than those in the BASIC group.

Summary

This chapter has tendered the results of the present study in relation to problem solving skills and classroom interactions among the subjects. It was found that subjects who learned LOGO programming were able to transfer their problem solving skills to a near-transfer context when compared to the BASIC and the control groups. Also, students who received their LOGO learning with a process-oriented approach were able to transfer their problem solving skills better than those who learned LOGO with a content-oriented approach. Moreover, it was found that subjects in both LOGO group exhibited more substantive interactions than their counterparts in the BASIC group. The next chapter will discuss the findings of this study.

CHAPTER EIGHT

DISCUSSION

This chapter discusses the results of the present research. It focuses initially on the relationship between the development of problem-solving skills and learning to program with the computer languages LOGO and BASIC when different instructional approaches are used. This is followed by a discussion of the effects of different instructional methods on interactions within the programming environment. Suggestions for teaching and learning strategies with the LOGO language are then discussed. Finally, the limitations of the study and recommendations for future research are considered.

Overview

The present study investigated the effects of computer programming on children's general problem-solving skills and classroom interactions. It also examined whether the method of instruction and the type of languages used would affect the transfer of such skills to other non-programming domains, including those of a near-transfer and far-transfer nature. Further, the investigation incorporated some aspects of metacognitive training into the process-oriented approach to teaching LOGO programming in order to test for transfer of problem-solving skills. The present research also included observation of classroom interactions between teachers and students so that more information could be obtained on the relationship between programming, instructional methods and social interaction.

The results of the study provide some evidence of related gains from learning LOGO, irrespective of instructional methods used. Students from both LOGO groups showed transfer of problem-solving skills to a near-transfer context to a greater extent than did students in the BASIC and control groups. Similar evidence was not forthcoming for the far-transfer context. These findings provide some support for the use of LOGO as a vehicle to develop problem-solving skills.

The consistently higher scores from the LOGO process-oriented (LPO) group when compared to those from the LOGO content-oriented (LCO) group in the more complicated Tower of Hanoi task seem to suggest that instructional method may

have a significant part to play in the transfer of problem-solving skills. The LPO group results gave evidence of application to another context the skills that the students were previously taught.

The observational data of the study also suggest that LOGO could facilitate social interaction among students and that these in turn may affect the acquisition and transfer of problem-solving skills. Moreover, the higher incidence of both substantive verbal and non-verbal behaviours of the LPO group when compared to those of the LCO group indicated that instructional method may facilitate social-cognitive interactions among students.

The remainder of the chapter will discuss these results in more detail and comment on their significance.

Programming and problem solving

In general terms, the results on problem solving, including mathematics achievement, indicated that the post-test performances of subjects in all four groups were usually higher than pre-training performances. Without looking in detail into group differences and the types of problem-solving measures, this result may signify a practice effect. However, group comparisons showed the results of the two LOGO groups to be statistically significantly different from the BASIC and control groups on some measures of the Tower of Hanoi. As well, the LOGO process-oriented (LPO) group also significantly outscored the LOGO content-oriented (LCO) group on some sub-problem measures of the more complicated Tower of Hanoi problems. Accordingly, the following sections will examine the results separately according to the types of problem-solving measures used.

LOGO programming and mathematics achievement

The lack of statistically significant differences in the PAT Mathematics Achievement scores indicates that the learning of LOGO programming produced no effect on the general mathematics achievement of the learners (see Tables 7.3 & 7.4). These results are in accord with findings of previous studies. For instance, Clements (1986c) found that LOGO programming did not affect the mathematics

achievement of the subjects who received 22 weeks of learning in LOGO. In the study of Battista and Clements (1986), students showed no gains in their mathematics achievement after learning LOGO programming for a year. The absence of evidence of the transfer of mathematical learning from a LOGO environment was also shown in a study by Finlayson (1983) who found that although students appeared to be competent with turtle graphics they failed to comprehend fully the underlying mathematical concepts. These researchers suggested that one of the major problems was that students failed to see the connection between what they learned and applications in other contexts. An attempt was made in this study to provide more explicit links between LOGO programming and the applications of mathematical concepts through the use of a process-oriented approach. Students were asked to reflect on the applications of mathematical concepts in contexts other than LOGO programming (eg, spatial distance in travelling around the school, turning of angles etc.). Nonetheless, it was quite clear that students who learned LOGO programming using either a process-oriented approach or a content-oriented approach did not perform in mathematical achievement any differently from their counterparts. Transfer of specific mathematical strategies or concepts from LOGO to normal classroom mathematics was not measured in this study so comparison cannot be made with studies that did (eg, Lehrer & Smith, 1988; Thomson & Chen Wang, 1988; Turner & Land, 1988; Ortiz & MacGregor, 1991).

The main point that must be considered in studies that examine LOGO programming and mathematics achievement is that the coverage of standard mathematics in LOGO programming is perhaps too slight to cause significant gains in mathematics achievement. Typical mathematics achievement tests include a wide range of mathematical concepts and computational skills. Short intervention studies like the present one do not lend themselves to the more extensive coverage of mathematical contents that are contained in the standard mathematics achievement tests. Similar arguments have been advanced by Irwin (1985) who suggested that there was little similarity in the content of the mathematics syllabus and the type of problem solving that occurred in LOGO programming session. As well, the short duration of the present study did not allow students sufficient opportunities to practise and to internalize the mathematical strategies that they might have acquired

through the learning of LOGO programming. Future studies may need to focus on the learning and transfer of more specific mathematical concepts and strategies if any gains are to be found. On the other hand, if the students are expected to show significant improvement in mathematics achievement, then a more elaborate curriculum in mathematics through LOGO programming may need be developed. More careful "mappings" between the students' work in LOGO and classroom mathematics may need be made and brought to a level of explicit awareness for the students (Clements, 1987b). In other words, specially designed LOGO microworlds that focus on a large number of mathematical concepts and strategies may be required to be established which may enable students to learn and practise a wider range of mathematical skills (eg, Niess, 1992) but whether the result would warrant the effort is problematical.

LOGO programming and problem solving

The prime focus of this study was the examination of the effects of LOGO programming on transfer of general problem-solving skills to non-programming contexts. A number of problem-solving measures were used to assess the problem-solving skills of the students both before and after the intervention. These measures, *inter alia*, were used to gauge skills in: planning, analysis, synthesis, spatial relationships, logical reasoning and creative thinking. Unlike most of the previous studies, the present research made a distinction between problem-solving tasks that were of a far-transfer nature and those that were of a near-transfer nature. This distinction was made based on the literature on problem solving that showed that it might be difficult to achieve far-transfer but relatively easier to achieve near-transfer (cf Ellis, 1965; Ginther & Williamson, 1985; Alexander & Judy, 1988; Palumbo, 1990). The next sections will examine the results according to the two different types of problem-solving measures.

Far-transfer problem-solving measures

The following measures were used to assess far-transfer of problem-solving skills: Raven's Standard Progressive Matrices; the three sub-tests of the WISC-R, viz. Picture Arrangement, Block Design, and Object Assembly; the Rule Naming

Task; and the Torrance Test of Creative Thinking. Although all these measures were of a figural nature, which to some extent, resembled the largely graphic domain in which students of the LOGO groups were working, results from both LPO and LCO groups did not show any statistically significant differences when compared to those from the other two groups (see Tables 7.5 to 7.26). These results lead to the partial acceptance of the overall hypothesis A that the learning of LOGO programming would not facilitate the transfer of problem-solving skills to another domain that is of a far-transfer nature.

These non-significant findings are consistent with a number of LOGO studies which employed problem-solving measures that were of a far-transfer nature. For example, in studies by Pea (1983), and Pea and Kurland (1984b) which used classroom planning tasks as problem-solving transfer measures, the researchers did not find any transfer of planning skills even after one year's learning of LOGO. Chambers (1986) found that her subjects' performance in problem-solving skills such as planning and analysis showed no improvement after learning to program with LOGO. Similar results are to be found in a number of other studies reviewed in Chapter Four (eg, Horner & Maddux, 1985; Carver & Klahr, 1986; Mitterer & Rose-Krasnor, 1986).

The findings of these previous studies, together with those of the present one, highlight the difficulty in either effecting or demonstrating transfer of problem-solving skills from a LOGO environment to another context that is of a far-transfer nature. Deliberate attempts were made in the present research to teach problem-solving skills through the use of a process-oriented approach, and the application of these skills in other contexts. For example, students in the LPO group were taught specific skills such as breaking down problems into simpler sub-problems, planning their solutions carefully, and monitoring these solutions both at and away from the computer. They were also required to apply these skills in simulated real life problem situations such as planning a trip to the capital or buying a bottle of milk from a corner dairy. Nevertheless, students in the LPO group did not show any greater improvement than did the students in the other groups.

However, it is interesting to note that although there were no significant interaction effects with the four measures of creativity as assessed by the Torrance

Test of Creativity (flexibility, fluency, originality and elaboration), the gain scores of all the three programming groups were consistently higher than the control group in areas of fluency and elaboration. Previous studies on LOGO programming and creativity have shown some positive relationships. For instance, the study by Clements and Gullo (1984) found that their subjects improved in fluency. Similarly, Clements (1986c) and Horton (1986) also found their subjects improved on elaboration.

These results and the non-statistically significant tendency observed in the present study give rise to several observations. First, the tendency towards more comprehensive and elaborate drawing may have been a reflection of more systematic procedural thinking developed by the subjects during programming. This is certainly the case with LOGO programming as the language tends to encourage students to solve their problems by writing various procedures. The gains with students in the BASIC group remains speculative although perhaps being beginners, these students tended to write rather short programs to solve their problems one at a time. Also, a number of BASIC programming exercises used in the present study involved graphics. Second, creativity in a figural domain may have resulted from an increase in overall organizational adaptability, or from experience verbalising information or representations that are held in an encoding that is not isomorphic with language (Clements, 1987b). Because such verbalisation implies considerable processing, children in the programming groups may have encoded information in long term memory with a relatively extensive array of verbal, as well as visual association or symbols which could later be accessed. These might then serve as links in associative chains which lead to new re-organizations of memory, and thus to greater fluency and elaboration.

The non-significant interactions from the far-transfer problem-solving measures in this study are consistent with the literature on problem solving and strategy training in general. A number of theorists (eg, Gick & Holyoak, 1980; Hayes & Simon, 1977; Pea & Kurland, 1984d) note that it is very difficult for people to apply problem-solving strategies learned in one context to new problem forms and that the expectation of spontaneous transfer across diverse knowledge domains must be viewed cautiously. Studies on the transfer of problem-solving

skills have demonstrated that the lack of domain-specific knowledge would often hamper the successful transfer of general problem-solving skills to another context (see review of Alexander and Judy, 1988). More recent studies on problem solving have attempted to incorporate various forms of strategy training to enhance transfer but again, their results have highlighted the importance of domain-specific knowledge in solving problems within a particular context (cf Kuhn, 1990; Okagaki & Sternberg, 1990; Lawson, 1991; Stevenson, 1991).

Further, some LOGO researchers have asserted that failure to find far-transfer may be due to the low level of students' programming expertise (Leron, 1985; Dalbey & Linn, 1985; Kurland, Pea, Clement, & Mawby, 1986; Khayrallah & Meiraker, 1987; Palumbo, 1990; Dalton & Goodrum, 1991). Chapter Four indicated how some researchers have theorised about a chain of cognitive accomplishment related to the development and potential transfer of problem-solving skills to another context (see Figure 4.1). The short duration of the intervention used in the present study might have been insufficient for students to attain sufficient mastery of the LOGO language, not to mention the necessary acquisition of problem-solving skills which may be essential for transfer to take place.

Near-transfer problem-solving measures

The results with the near-transfer problem-solving measures used in this study - the number of moves to complete a Tower of Hanoi (TOH) problem and the correct percentage of solving the corresponding sub-problems - highlight two fairly distinct aspects that warrant attention. First, both LOGO groups outperformed the BASIC and control groups in the relatively less complicated Tower of Hanoi problems. These include: the three-disk problem and its two-disk sub-problem, and the four-disk problem and its two-disk and three-disk sub-problems. Second, scores from the LPO group were significantly higher than those of the other three groups (including the LCO group) in the five-disk problem and the related sub-problems (see Tables 7.37 to 7.44; Figures 7.6 to 7.9). The following sections will examine these two aspects of the near-transfer results.

LOGO programming and near-transfer. The significant interaction effects with the three-disk and four-disk problem scores in this study (see Tables 7.27 to 7.36; Figures 7.1 to 7.5) partially support the overall hypothesis A which stated that the learning of LOGO programming would facilitate the transfer of problem-solving skills to a non-programming context that was of near-transfer nature. The planned comparisons indicate that both LOGO groups outperformed the two other groups in these measures.

These findings are consistent with a number of findings from LOGO studies that employed problem-solving measures that were of a near-transfer nature. For instance, McAllister (1985) used Tower of Hanoi in his study. Although he made no comparison of pre versus post treatment gains, he found that the scores of Tower of Hanoi correlated positively with measures such as program writing, program creating, and program reading. This led McAllister to suggest that skills acquired while learning LOGO programming might positively transfer to other non-programming environments bearing similar properties. Horton (1986) found that when her subjects were using commands which were very similar to LOGO commands, they were able to transfer their problem-solving skills learned in a LOGO context to non-programming contexts. Similar types of direction-describing tasks were used in a study by Gallini (1987) who found that the LOGO subjects were able to achieve significantly higher scores than those in the control group. Comparable results were obtained in studies that employed measures that bore resemblance to tasks carried out in a LOGO environment (eg, Clements & Gullo, 1984; Mayer & Fay, 1987). Moreover, Au and Leung (1991) obtained very similar results in a study of students who used English as their second language. In that study, the Tower of Hanoi was also employed as one of the problem-solving measures. The students who learned LOGO using either the process-oriented approach or the content-oriented approach outperformed those in the control group. As in the present study, the subjects only showed improvement in the simpler problems of the Tower of Hanoi.

The findings of these previous studies, together with those of the present research, suggest that LOGO may be facilitative in the transfer of problem-solving skills to another context that is of a near-transfer nature. The facilitative effects

might be due to the characteristics of the LOGO language. As reviewed in Chapter Three of this thesis, some of these inherent characteristics include: learners are encouraged to break down complex problems into simpler ones, solve the simpler problems by writing the appropriate procedures, and then combine these simpler procedures to solve the more complicated problems. The structure of the Tower of Hanoi task consists of a number of sub-problems that are identical in form. Solutions can be constructed by firstly devising a solution for one of the sub-problems and then progressively achieving the total solution by repeatedly using the "simple" solution (McDougall, 1988). The skills required to solve the TOH resemble very much skills acquired while learning to program with the LOGO language. The hierarchical sub-problem structure, and the fact that the recursive nature of the moves toward the program goal, make the TOH isomorphic to LOGO programming (Luger, 1976; McAllister, 1985). In other words, the TOH is a problem which has the same structure as programming in LOGO, but presented in a different form. It is perhaps because of this structure of the TOH that students in both LOGO groups were able to score significantly higher than those in the other two groups. Although the students in the LOGO groups may not necessarily have understood the concept of recursion in the programming sense, the fact that they have used skills such as breaking down complex problems in LOGO programming may have helped their subsequent performance.

This aspect of the results is consistent with findings obtained from research on problem solving. For example, a study by Luger and Bauer (1978) assessed the relationship between human problem-solving behaviour and the structural properties of certain problems. They found that transfer effects are easier to demonstrate when the two problems are isomorphic in structures. Similar isomorphic transfer was also found in other studies (eg, Simon & Hayes, 1976; Gallini, 1987; Clements, 1987b; Clements & Gullo, 1984). The results of the present study are consistent with the view that generalization of problem-solving capability is more likely when structures in contexts are isomorphic.

One of the research questions addressed in this study was whether LOGO provides a better medium to develop problem-solving skills than does the language BASIC. The results of the study showed that subjects in the LCO group scored

significantly higher than the subjects in the BASIC group although both groups were taught with a content-oriented approach (see Tables 7.27 to 7.36; Figures 7.3 to 7.5). The planned comparisons with some of the measures of the three-disk and four-disk problems of the Tower of Hanoi revealed that the contrasts were either significant or approaching significance. This could perhaps be due to the difference in the nature of the two languages. In this study, subjects in the BASIC group reported comparatively more difficulties in mastering the syntaxes and commands of the BASIC language than did those in the LOGO groups. These subjects' difficulty with the BASIC language lead to two observations in relation to their performance with the problem-solving measures. First, the subjects in the BASIC group might have spent more time in trying to master the language rather than mastering the procedural and conditional knowledge that are essential in problem solving. Second, as Chapter Four reported, some researchers (cf Linn, 1985; Leron, 1985; Palumbo, 1990) have argued that the development of problem-solving skills progresses along a chain of cognitive accomplishment, through which learners move from learning the syntactical and declarative aspects of the language through to more generalizable problem-solving skills. With the BASIC language, perhaps the learners were not able to proceed beyond the simple syntactical and declarative knowledge of the commands, therefore preventing them from acquiring let alone transferring, the more generalizable problem-solving skills. LOGO, on the other hand, has a less complicated command structure and therefore may promote more rapid movement along the chain of cognitive accomplishment.

Instructional methods and near-transfer. The second aspect where there were differential results with the more complicated five-disk TOH problems between students in the LPO group and those in the LCO group suggests that there may be a relationship between the process-oriented instructional method used in this study and transfer of problem solving. When examining the total number of moves to complete a five-disk problem as well as the correct percentages in solving the related two-disk, three-disk and four-disk sub-problems, the planned comparison demonstrated that the interaction effects lay in the higher scores of the LPO group when compared to those of the other three groups (see Tables 7.37 to 7.44, Figures

7.6 to 7.9). These findings indicate that students who learned LOGO programming with explicit problem-solving instructions were better able to transfer their problem-solving skills to another context than were students in the other three groups.

These results are similar to a number of previous LOGO studies. For example, Au and Leung (1991) found that in their study, students who learned LOGO programming with instruction in problem solving, scored significantly higher in near-transfer problem-solving measures than did subjects who received LOGO instructions without any direct instruction in problem solving. Dalton and Goodrum (1991) also found that when LOGO programming instructions were augmented by problem-solving instructions, their students performed better when compared with others who had just learned programming. Similarly, studies that consistently demonstrate transfer of problem-solving skills to other domains tend to be those where problem-solving skills were taught explicitly. The results in the present study and others suggest that there may be some relationship between instructional methods used in the teaching of programming and the acquisition and transfer of problem-solving skills.

Instructional methods and transfer of problem-solving skills

One of the research questions raised in this study was whether instructional methods might have some effect on the transfer of problems-solving skills. The significant interaction effects among the groups with the five-disk problem and its sub-problem measures indicated that the LOGO group taught with a process-oriented approach achieved significantly higher scores than did the LOGO group taught with a content-oriented approach (see Tables 7.37 to 7.44). Moreover, these results suggest that the students in the LPO group might be able to transfer to another domain the problem-solving skills they had been taught explicitly both more consistently and to more complicated problems. Informal observation and conversation with the subjects in this group confirmed that, while attempting to solve the problems, they did try to use the skills that they were taught, such as breaking down a complex problem into simpler sub-problems. This was particularly evident with the Tower of Hanoi exercise where the structures of the problem lend themselves to such endeavours.

On the basis of these results, the present study suggests that an instructional approach that emphasizes problem-solving skills in the learning of programming may have a facilitative effect on the transfer of problem-solving skills to the specific non-programming context tested. These findings lend support to the second overall hypothesis B in that the degree of transfer from the LOGO environment to non-programming context of a near-transfer nature would be greater for children taught with the process-oriented approach compared with those taught with the content-oriented approach.

From the results of the TOH, it was obvious that the gains obtained from the traditional content-oriented approach of programming instruction were only restricted to the simpler TOH problems. Once the students in the LCO group encountered the more complicated TOH problems, their performance was no different from those in the BASIC and control groups (see Tables 7.37 to 7.44). These results and others (eg, Pea, 1983; Pea & Kurland 1984b; Mitterer & Rose-Krasnor, 1986; Clements; 1990; Au & Leung, 1991; Dalton & Goodrum, 1991) support the conclusion that a pedagogy of programming devoid of problem-solving instructions would not help realizing Seymour Papert's vision of a LOGO microworld in which young children's cognitive development may be accelerated.

A close analysis of the instructional methods used in studies that did observe transfer of problem-solving skills reveals that the instructional methods employed were very much process-oriented (eg, Clements, 1986c; Clements & Gullo, 1984; Horton, 1986; Lehrer & Smith, 1986; Dalton & Goodrum, 1991; Swan, 1991). They usually encouraged students to adopt procedural techniques such as identifying problems, breaking down complex problems, planning, self-monitoring and checking during the development of LOGO programs. Furthermore, the instructors, rather than just providing learners with informational assistance, also tended to use questioning techniques similar to those proposed by Au, Horton and Ryba (1987). The reasoning was that by explicitly teaching the students these problem-solving skills, and allowing them the opportunities to reflect on and to practise these skills, there is more likelihood that transfer would take place.

The issue of transfer has been examined by researchers for decades (cf Ellis, 1965; Mayer & Fay, 1987). For instance, Ellis (1965) provides a set of

requirements for the teaching for transfer which includes the need to teach students some general principles and the application of these principles in a variety of contexts. More recently, Salomon & Perkins (1987) propose a theory for the mechanisms of transfer involving low-road transfer and high-road transfer (cf Chapter Five). They suggest that high-road transfer involves deliberate mindful abstraction from one context to another, and requires genuine understanding of the abstraction and self-conscious efforts to apply the abstraction in new situations. The LPO group's results with the TOH problems seem to suggest evidence of high-road transfer. In other words, the students who learned LOGO with a process-oriented approach seemed to be able to make a conscious effort to apply the abstraction in new situation.

Theoretically speaking, the process-oriented pedagogy incorporates some aspects of metacognitive training. Metacognition involves the monitoring and control of one's cognitive processes such as memory, comprehension and attention etc. Metacognitive psychologists, especially those who take a developmental perspective, have often been interested in training general basic skills which are considered to be needed for successful problem solving (eg, Anderson, 1980; Bransford & Stein, 1984; Brown & DeLoache, 1978; Newell & Simon, 1972; Sternberg, 1984; Chan, 1991). Extensive reviews of the cognitive literature on children's reading and problem solving (eg, Brown, Bransford, Ferrara & Campione, 1983; Belmont, Butterfield & Ferretti, 1982) have shown that the transfer of problem-solving skills is more likely to occur when students are given the appropriate instructions. Examples of successful intervention (eg, Brown, Campione & Barclay, 1979; Palincsar & Brown, 1984; Ellis, Lenz & Sabornie, 1987a, 1987b; Paris & Winograd, 1990a) often stress that explicit instruction of superordinate self-management skills and generalization can assist the development and transfer of problem-solving skills. Such findings invariably lend further support in the explicit instruction of problem-solving skills when teaching LOGO.

The empirical nature of this study on the problem-solving behaviours of the students focussed attention on the more objective and measurable aspects of problem-solving behaviours. The shortcomings of such an approach have been highlighted by a number of researchers (cf Chapter Four) in that many theoretically

possible changes in problem-solving behaviours could not be gauged by a pre versus post design using traditional problem-solving measures. In light of such criticisms, the present research sought to observe informally the problem-solving behaviours of the students while they were actually solving their programming problems.

Informal observations suggest that LOGO, when taught with a process-oriented approach, can be of particular value to some students who are low academic achievers. A few students who were considered by their own teachers to be of low ability, achieved considerably both in terms of confidence and solving LOGO programming problems. Two illustrations follow.

1. One day prior to the beginning of a programming session, one of the teachers in the school approached the researcher. She was undertaking a computer-related course at the local Teachers College and to complete her LOGO assignment, wanted access to the computers in the school (normally reserved for the purposes of the present research). While the request was being made, it was overheard by one of the students in the LPO group who was about to begin his learning session. The student then volunteered to teach "his teacher" how to use the computers for LOGO. Once in the computer room, he showed his teacher how to switch on the computer and how to draw various geometric shapes (including his own initials) using LOGO. The teacher had previously considered the student to be rather reserved and of low academic ability (his Reading and Listening Comprehension scores were 16 and 36 respectively). That he was able to introduce her to the use of LOGO quite systematically came as a considerable surprise.

2. This example is more general. In the LPO groups, students were observed to apply systematically the problem-solving skills that they were taught in solving LOGO programming problems. Quite often, students (especially those with low ability) walked around the classroom or drew on pieces of paper in order to determine the distances and angles they needed to complete certain geometrical drawings. Instead of just trying out commands on the computer screen, they planned and checked their work carefully before going to work with the computers. Once they had verified with the computers that their solutions were correct, their delight was self-evident.

Similar examples could be cited to illustrate such achievement not measured by the problem-solving tasks used in this study. These examples, when considered in conjunction with the results of the testing on problem solving, suggest that instructional methods may play a significant part in the teaching and learning of LOGO programming.

In summarizing the findings for the problem-solving measures, several conclusions can be drawn. First, the learning of LOGO or BASIC programming (under the conditions of the present study) produced no effect on the general mathematics achievement of the learners. Second, the learning of LOGO or BASIC showed no effect on the transfer of problem-solving skills to a far-transfer context. Third, the learning of LOGO programming produced some effects on the transfer of problem-solving skills to a near-transfer context. Fourth, the incorporation of explicit problem-solving instructions appear to enhance the transfer of problem-solving skills. Fifth, it would appear that some gains of problem-solving skills might not have been measured by the tests used in this study.

LOGO Programming and Social interaction

One of the initial fears that computers might "dehumanize" students, isolating them from normal interactions with teachers and peers, as well as leaving them deficient in important social experiences, was not corroborated by the study. The subjects were seen to be constantly and spontaneously involved in interactions with each other, irrespective of the instructional methods used.

In general, motivation among all three programming groups was high although this could be because of a Hawthorne effect. This was reflected in the minimal amount of absenteeism and disciplinary problems among the students throughout the intervention and was noteworthy considering that the programming classes were held after school. Informal observation also showed that there was a very low level of off-task behaviour among the students and that the disciplinary problems expected in a normal classroom were almost non-existent. Students usually came to the programming sessions with much visible enthusiasm and with lots of questions about computers and programming. Informal conversation with the

students further indicated that they did enjoy learning about programming and they all thought that they did learn something worthwhile. Conversations with the principal, the teachers and the parents also conveyed similar impressions.

Such informal observations are similar to results in other LOGO studies which examined affective changes of learners. Studies by Weir (1981), Irwin (1985), Nastasi, Clements and Battista (1990) all found that LOGO could provide a high level of motivation for the learners.

The apparent high levels of motivation might have assisted students to focus on their programming work and fostered more social interactions with the teachers and other students. Although the interactions of the students in all three programming groups cannot be compared in any systematic manner to interactions in a normal classroom, informal observation during the learning sessions conveyed the impression that there were substantially more interactions among the students than tends to be the case in normal classrooms. A number of reasons, apart from the high level of motivation, could be advanced in explanation. First, the existence of the computer screens tended to make students' work more public, hence leading to more opportunities for students to look at each other's work and to discuss it. During the present research, students were often seen crowding around another student's computer because an interesting figure was drawn on the screen. Second, the students were working with materials which were quite novel to them, therefore, they may have been more likely to share their ideas with the other students. Third, as Papert (1980) has postulated, LOGO provides the students with something of interest to talk about.

More central to the purposes of the present study though, were the observations of interactions among teachers and students during the programming sessions. There were three main forms of observation: (i) interactions between teachers and students; (ii) interactions of individual students with others; and (iii) interactions of groups. The following sections will examine the observation results in more details.

Quite expectedly, because of the different instructional methods used, there was a substantially higher percentage of process-oriented interaction between teachers and students in the LPO group than there was in the other groups (61% for

the LPO group, 6% for the LCO group and 2% for the BASIC group) (see Table 7.45). This percentage of process-oriented interactions may partly explain the reasons that students in the LPO group were able to transfer their problem-solving skills to other contexts better and more consistently. Moreover, there were also more episodes of interactions between teachers and students in both LOGO groups than in the BASIC group (LPO: 904 episodes; LCO: 1024; BASIC: 677) (see Table 7.45).

There were also more group interactions observed among students in both LOGO groups (see Table 7.46). Both LOGO groups had more episodes of group interaction involving two or more groups at the same time. These results, when considered in conjunction with the previous findings, suggest that there is a relationship between LOGO learning and social interaction. The higher incidence of interaction with the teachers and fellow students appears to suggest that LOGO encourages students to discuss their work and share their problem-solving experiences.

Moreover, it was also observed that the LPO group had a higher incidence of interaction involving two or more groups of students at the same time. These results suggest that instructional methods might play a significant role in determining the extent of student social interaction. In particular, the process-oriented approach used in the present study explicitly encouraged students to discuss their learning and related problems with each other.

The observation on individual student interactions shed further light on the interactions of students in a programming environment. From the observation data, it could be seen that there were significantly more episodes of both substantial verbal and non-verbal interactions for students in both LOGO groups (see Tables 7.47 to 7.50). These results indicate that students who learn LOGO were more likely to interact with their peers, either discussing their programming work and problem solving with other students in the group, or observing what others were doing.

Moreover, the statistically significant results of both substantive verbal and non-verbal interactions also suggest students in the LPO group were more likely than their counterparts in the LCO group to interact with their fellow students. This is not really surprising given that the process-oriented approach, by definition, was to

encourage discussion of problem solving among the learners. The implication though, is that when students were engaged more in discussion of their problem-solving strategies, they were likely to improve their problem-solving skills (eg, Nastasi, Clements & Battista, 1990; Vygotsky, 1978).

Although the results of this study do not provide a basis for comparison with those of Clements and Nastasi (1988) and Nastasi, Clements and Battista (1990), occurrence of social-cognitive interaction similar to those observed in these two studies was noted during the research. For instance, students were quite often observed to be sitting in front of their plans discussing how a certain geometric figure could be achieved using LOGO commands. At times, they were also observed to be engaged in a conflict situation where they were in debate as to which commands would be more suitable to solve a certain problem, ultimately either arriving at some form of consensus or going separately to test their solutions with the computers.

These results are consistent with previous studies. For example, Mitterer and Rose-Krasnor (1986) reported that interaction levels were high among both the LOGO and BASIC groups in their study. The LOGO group in particular had a higher incidence of interactions with the tutors. Burns and Coon (1990) also found that peer collaboration using LOGO focussed more on the process relative to the product of problem solving in their LOGO experimental group than in their control programming group.

The results of the present study, together with others, suggest that LOGO could be used to provide a learning environment which facilitates social interaction - interaction that tends to be more related to the learning tasks and to the solving of problems. Also, the appropriate use of instructional methods could further influence the social-cognitive interactions of the learners.

In summary, the results of the observational aspects of this study, when considered in conjunction with the problem-solving aspects, indicate some circumstances under which LOGO could be used to provide a more dynamic learning environment for the learners, particularly when compared to BASIC. This may in turn result in better development of problem-solving skills. The results of the present study also indicate that the suitable use of a socially interactive and

reflective environment could assist the acquisition and transfer of problem-solving skills. In this case, the LPO group exhibited a higher incidence of social interactions. Also, it could be inferred from the results that students in this group were able to transfer problem-solving skills to another context better than students who learned LOGO using a content-oriented approach.

Implications for the teaching and learning of LOGO

Two of the major concerns about the teaching and learning of LOGO that have emerged from the literature are "What is meant by learning to program in LOGO?" and "How can learner cognitive gains be maximised through learning to program in LOGO?" From the results of this study and others, what clearly appears to be a crucial factor in whether cognitive gains are made or not, is the way in which LOGO is being taught and learned. LOGO studies in the last few years have clearly indicated that if learners are to achieve the kind of cognitive gains postulated by Papert (1980) in his book Mindstorms, then there is a need to examine more closely the learning environment created for the learners. In particular, careful consideration must be given to the role of the teacher and to the instructional strategies used. Also, contemporary research in problem solving and metacognition has suggested the importance of the provision of explicit instructions in problem solving and in self-management skills if learners are to acquire and transfer problem-solving skills. The following discussion will attempt to provide some practical suggestions as to how programming lessons could be designed to assist the acquisition and transfer of problem-solving skills. These suggestions are based on the outcomes of this study and other LOGO studies. To begin, two major points need be considered here.

First, what do children learn about LOGO? Proponents of LOGO have often failed to articulate what it means to teach and to learn LOGO programming. Frequently, LOGO programming was treated as a "black box", "an unanalysed activity whose effects are presumed to irradiate those who are exposed to it" (Pea & Kurland, 1984a:9). Thus it will be necessary to identify what it means to learn to program in LOGO in the first instance. Educators and researchers alike will then have a much better idea of what learning outcomes they might reasonably expect.

The results of the present study and other LOGO research since the middle of 1980's have clearly suggested the need for a closer examination of the types of skills supposedly learned by students. If a major objective of learning LOGO is to assist students to develop problem-solving skills, then there is a clear need to identify the types of problems-solving skills that may be acquired through learning LOGO.

More importantly, recent research in LOGO programming, problem solving and metacognition has provided strong arguments that to achieve cognitive gains, these skills need be taught to the students in a more explicit manner. Accordingly, programming activities should be designed to encourage the mindful application of problem-solving strategies such as planning, analysis, and monitoring, derived from sound cognitive theory (cf Salomon & Perkins, 1987).

Also, programming lessons should quickly develop an elementary mastery of language syntax and move quickly to procedure application and problem solving. The sequencing of these lessons should be of such a manner that students are introduced to aspects of LOGO programming language, ranging from simple to the complex, similar to those proposed by a number of LOGO researchers (eg, Chambers, 1986; Nolan & Ryba, 1986; Watt & Watt, 1986). Moreover, it is also important that these programming lessons consist of activities within which students are asked to exercise their thinking skills and reflect on their problem solving (cf Au, Horton & Ryba, 1987).

How should LOGO be taught? The debate on how LOGO should be taught has been an ongoing one (see Chapter Three and Four). Papert (1980), for example, has called for LOGO teachers to take on the role as "anthropologists" but did not really articulate how this could be achieved. However, the results from the present study and those since the middle of 1980's, have provided strong arguments that it is important for a "LOGO environment" to consist of careful teacher intervention that assists students to develop their problem solving skills. For instance, students need be taught problem-solving skills directly and to be given sufficient opportunity to practise these skills and their applications to other contexts. Teacher questioning could be used to assist the students to develop and practise these skills, with the aim that the students will be able to master their own problem solving without prompts from the teachers. In essence, the role of a teacher is

critical in developing problem-solving skills and metacognitive skills among the students, and in the teaching for transfer.

The alternation between on- and off-computer activities could also be an important consideration. Research has indicated that students tend to become more "impulsive" problem solvers if they only solve their problems in front of the computer (cf Carver & Klahr, 1986; Pea & Kurland, 1984d). By encouraging students to plan and to reflect on their solutions carefully and systematically away from the computer, they are more likely to do so. For instance, students in the present project were often seen carefully checking their plans on paper, identifying and correcting the errors in the plans, before going to test them at the computer. Moreover, off-computer activities can provide opportunity for students to practise their problem-solving skills in non-computer contexts, which can arguably, increase the likelihood of transfer.

Also, the teaching of programming should be of sufficient intensity and duration to provide opportunities for acquiring both declarative, procedural and conditional knowledge (cf Keller, 1990; Palumbo, 1990; Au, 1992a). Many researchers have theorised that there is chain of cognitive accomplishment (cf Chapter Four) along which students develop their programming and problem-solving skills. Unless the learners have moved sufficiently along this chain, they are less likely to acquire the required problem-solving skills. There is emerging evidence to suggest that the intensity of learning of programming is related to the transfer of problem-solving skills (cf Palumbo, 1990).

Coupled with a more structured LOGO environment for the learners, somehow a balance must be found between allowing pupils the freedom to work on their own extended projects and structuring the activity for specific learning outcomes - to avoid "gaps" in pupils' awareness of the potential use of LOGO and to confront any misconceptions (Hoyles & Sutherland, 1987). By doing so, students are more likely to devise projects that are of interests to them, important for the sustenance of long-term interests and motivation. Also, students are more likely to develop better self-management skills which are important for good problem solving. In the present studies, students were given ample opportunities to choose their own projects. Much enthusiasm was evident. Often, they would go away thinking about

their projects after the completion of a session and then return during the following session with very thoughtful plans that were carefully checked before these plans were tested with the computer.

The encouragement of social interaction forms a very important part of the strategies to assist the development of problem-solving skills. It has been highlighted in both Chapters Four and Five that appropriate social interactions that are of a cognitive nature may assist students to better develop their problem-solving skills. A number of more recent studies on LOGO (eg, Clements & Nastasi, 1988; Nastasi, Clements & Battista, 1990) have all noted that social interactions within a LOGO environment may assist students in a number of problem-solving areas such as cooperative problem solving, rule determination, and conflict resolution. The examples of student dialogues cited on pages 201 to 202, and 204 clearly highlight such experience for students in the present research. Therefore, when designing strategies in the teaching of LOGO, the encouragement of social and reflective interactions becomes an important consideration.

In summary then, the results of the present study and other research have suggested that a teacher plays a critical role in facilitating a student's development and transfer of problem-solving skills. It is through the provision of a socially interactive and reflective environment by a teacher within which a student may acquire the necessary problem-solving skills for transfer. More specifically, the teacher intervention has to do with ways in which teachers talk with students, the types of questions they ask and the sort of discussion that take place between students and teachers.

Based on the results of the present study and other research (eg, Riordan, 1982; Au, Horton & Ryba, 1987; Clements & Merriman, 1988; Ryba & Anderson, 1990), a list of practical recommendations for improving problem solving with LOGO is presented. This list of recommendations represents ways in which teachers can create an interactive and reflective LOGO environment within which children can learn to acquire and transfer problem-solving skills.

1. observe how students talk to each other and how they solve their problems as this information provides the basis for monitoring student progress;

2. show and discuss with students how to apply problem-solving skills to other contexts;
3. ensure that students are explicitly aware of thinking processes and problem-solving skills;
4. ask students questions that encourage reflective thinking and that give control to the students;
5. encourage students to explore various ways of solving problems;
6. reinforce students for using problem-solving skills;
7. encourage students to think aloud about their problem solving and share their problems with each other as these social interactions provide a context for cognitive growth;
8. promote child/teacher and child/child interaction;
9. provide sufficient time for programming problem-solving both at and off the computer.

An example of a typical LOGO lesson on problem solving is included in Appendix 12 (adapted from Au, Horton & Ryba, 1987).

Limitations of the present project and suggestions for future research

The results of this study provide some support for the benefits of LOGO in facilitating the development and transfer of problem-solving skills among students. As well, the findings of the present research generally support the value of specific instructional strategies and social interaction in the teaching of programming with the language LOGO. Past research with LOGO programming often tended to ignore the significance of instructional factors. Often, these studies concentrated on the so called "LOGO environment" or "LOGO microworld" as a total entity without considering the importance of the various factors within such an environment (Au, 1992a, 1992b; Jones, 1992). In so doing, the effects of factors such as instructional strategies and social interactions among learners were neglected.

Moreover, past research has frequently omitted to consider the relationship between LOGO programming and the types of problem-solving measures used. The present research highlights the importance of distinguishing the type of problem-

solving measures used in the gauging of transfer of problem-solving skills. Specifically, it shows that the transfer of problem-solving skills to a far-transfer domain is very difficult.

As with any research project, there are a number of limitations within the present study which must be acknowledged. These limitations are especially apparent given that the present study was designed in 1985 and the fieldwork completed in 1986. Since, subsequent studies have either pointed out or addressed some of the limitations inherent in the present investigation. The following sections will list some of the limitation of the present study and concurrently make suggestions for future research.

First, the study confined its attention to a comparison of a LOGO content-oriented group with a BASIC content-oriented group. The study's scope would have increased if it had proved possible to include an additional BASIC group taught with a process-oriented approach. A number of studies that were published after the conduct of the present study employed research designs that included groups that were given problem-solving instructions without computer programming. Through these studies, it can been seen that students could benefit from explicit instructions on problem-solving skills. Future research would clearly need to address this issue more closely.

Second, the duration of fieldwork in the study was brief and not particularly intensive. This was owing to the need to fit in with the activities of the school for that year. Like other studies that have been conducted in school settings, it was necessary to modify the study so as not to disrupt the activities of the school and the students. One session a week for each student was all that was possible. It would also have been advantageous if the problem-solving skills of the students could have been tested, say six months later, in order to examine the retention factor. Unfortunately, the school year ended some two months after the conclusion of the project and the year four students all left the school.

Third, there were some problems of ecological validity with this study. The students stayed after schools to receive their programming learning in small groups - a teacher-student ratio significantly different from the normal. It is often easier to guarantee a high quality of instruction when dealing with a smaller group of students

than when teaching a larger group. With a smaller group, the teacher may focus more on the management of learning than the management of learners, a factor which Bloom (1976) considers critical. The project was also seen by the students as a kind of extracurricular activity rather than a normal aspect of their school learning.

Fourth, the learning styles of the students were not examined. It has been suggested by some researchers that the learning styles that students bring to a learning situation might have important influence on learning outcomes. For instance, Biggs (1987) made a distinction between surface, deep and achieving learners. In many studies that employ such distinctions, it has been found that the deep and achieving learners performed better. Future studies may show whether learner characteristics may impinge upon the learning of programming.

Finally, there are grounds to believe that more in-depth observation of the interactions of the students in relation to the social-cognitive interaction framework used by Clements & Nastasi (1988) and Nastasi, Clements & Battista (1990) would provide more insight into the nature of interactions that could have contributed to the cognitive development of the students.

Conclusion

This study was conceptualized at a time (in the middle of the 1980's) when there was substantial conjecture over the benefits of learning LOGO. Many LOGO enthusiasts argued that LOGO could assist the development of problem-solving skills of the students. Other researchers more cautiously suggested that there was a need for more systematic and empirical evaluation of the potential benefits of LOGO. Also at issue was the way that LOGO should be taught. One group of LOGO researchers supported a self-discovery model in the teaching of LOGO devoid of any teacher intervention. Others adopted a more structured approach with carefully designed teacher intervention.

It was amidst these debates that the present study evolved. Four important questions guided the design of this study. First, what is meant by the learning of LOGO and what kind of instructional strategies are required? Second, can another programming language such as BASIC assist the development and transfer of problem-solving skills? Third, what kind of transfer of problem-solving skills will

LOGO facilitate? Fourth, what kind of interactions are there within a programming-learning environment? Consequently, two different instructional approaches were used with the teaching of LOGO, and BASIC was included for comparison with LOGO. Also, two types of problem-solving measures were used to gauge the far- and near-transfer of problem-solving skills. Observation was also made on the classroom interactions.

From the results of this study, it could be seen that LOGO did have some beneficial effects on the transfer of some problem-solving skills, albeit to a near-transfer context. The LOGO learners were able to solve the Tower of Hanoi problems better than were those in the BASIC and control groups. These results are similar to those of a number of LOGO studies conducted in the last ten years. Most tend to highlight the difficulty of transfer of problem-solving skills.

The present study also showed that the incorporation of a process-oriented approach was associated with higher levels of problem-solving performance in a near-transfer context. The students in the process-oriented group were able to score higher than did their counterparts in the content-oriented group with the more complicated Tower of Hanoi problem.

Closer attention to the use of instructional methods and the role of the teacher was evident in LOGO studies that were conducted since the middle of the 1980's. Almost invariably, studies that demonstrated the cognitive gains from the learning of LOGO were those that taught students cognitive and metacognitive skills on an explicit and systematic basis. The use of such teaching strategies is also apparent in studies on problem solving and metacognition. Indeed, metacognitive researchers are continuing to search for better teaching methods that would assist learners to become better learners and problem solvers.

The difficulty with learning transfer was clearly manifested in the present study. First, the transfer was only demonstrated in one problem-solving measure, viz., the Tower of Hanoi. Second, the transfer was only of a near-transfer nature. This difficulty would inevitably lead to one of the questions that started the present research: "Is LOGO a suitable medium in teach problem-solving skills effectively and efficiently?" There is no simple answer to this question. However, a number of points warrant further consideration.

There is a need to acknowledge difficulties with the transfer of learning. Educators have been grappling with this issue for decades (cf Ellis, 1965; Mayer & Fay, 1987). Recent work in metacognition and problem solving has provided some encouraging results. If general problem-solving skills are supposed to be acquired through learning LOGO, it is evident that instructional strategies used with LOGO need to be examined. By itself, it could not be expected that students could acquire general problem-solving skills through the learning of LOGO. Complementary instructions in problem-solving skills are crucial.

It was obvious in this study that LOGO was able to facilitate social interactions among the learners. Also, the creation of a suitable learning environment could further enhance the social interactions which tended to focus on the actual problem-solving processes themselves. If one accepts the hypotheses that cognition is a consequence of interactions and that more social interactions could lead to increased cognition, then LOGO could be considered as a medium in the teaching of problem-solving skills.

The "how" question is a difficult one. The present study incorporated some form of metacognitive training in the teaching of LOGO. However, the gains in problem-solving skills, and hence their transfer were still limited. This brings back one of the original questions asked in this study: does a process-oriented approach in the teaching of programming assist the development and transfer of problem-solving skills. A simple answer to this question is "yes, but limited". A more considered answer would be "a number of factors need to be contemplated". These factors include the intensity and duration of treatment and a consideration of the learning characteristics of the learners. More importantly, perhaps, is the type of LOGO learning provided for the learners. The training provided in this study perhaps was quite limited. More recently, metacognitive researchers have started to examine the use of attribution training in order to enhance the problem-solving skills of the learners. Future studies may take this aspect of metacognitive training into consideration. Also, more focussed training on the ability of the learners to identify the structures of "novel" problems may be useful as well since one of the difficulties in the transfer of problem-solving strategies lies in a learner's ability to recognize isomorphic problems.

To summarize, the present study extends previous work in showing that LOGO programming does have some beneficial effects on children's problem-solving behaviour. In addition, this study also demonstrates the importance of instructional methods and the role of teachers in the teaching of programming for transfer. The importance of social interaction and its relationship to the acquisition and transfer of problem-solving skills within a programming-learning environment has also been highlighted.

In the areas of further research, both the findings from this study and other subsequent ones provide avenues for further investigation. In particular, future research may prove beneficial in the following areas:

(i) Instructional strategies including duration and intensity of treatment that may assist learners to develop and transfer of problem-solving skills more effectively and efficiently;

(ii) types of social interactions that may help learners to become better problem solvers;

(iii) relationship between LOGO programming and problem-solving measures;

(iv) relationship between the learning characteristics of learners, learning of programming and transfer of problem-solving skills; and

(v) the translation of the results of these investigations into normal classroom settings.

Finally, in this work, a modest step has been taken in the study of LOGO programming, instructional methods and the development and transfer of problem-solving skills. Many more steps remain to be taken in the understanding of the learning of programming and development and transfer of problem solving. What can be concluded from this study is that LOGO is a useful medium to facilitate the acquisition and transfer of problem-solving skills, but that earlier claims about LOGO need to be regarded with caution. Also, closer attention to instructional strategies and the role of the teacher in a LOGO environment may help to bridge Seymour Papert's vision and reality. As always, the test of science, explanation and prediction have to be met.

BIBLIOGRAPHY

Abelson, H. (1982a). A beginner's guide to Logo. *Byte*, *7*(8), 88-112.

Abelson, H. (1982b). *Apple Logo*. N.H.: Byte Publications Inc.

Adams, R.S. (1965). *The classroom setting: A behavioural analysis*. Unpublished doctoral dissertation. University of Otago, Dunedin, New Zealand.

Adams, T. (1985). Logo environments: the evolution of the language. In B. Rasmussen (Ed.) *The information edge: The future for educational computing* (pp. 133-140). Brisbane: Computer Education Group of Queensland.

Adams, T. (1986). Towards a theory of microworlds. In A.D. Salvas & C. Dowling (Eds.). *On the crest of a wave?* (pp. 312-320). Balaclava, Victoria: Computer Education Group of Victoria.

Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory and cognition*, *9*(4), 422-433.

Alexander, P.A., & Judy, E.J. (1988). The interaction of domain-specific and strategic knowledge in academic performance. *Review of Educational Research*, *58*(4), 375-404.

Alderman, D.L., Appel, L.R., & Murphy, R.T. (1978). PLATO and TICCIT: An evaluation of CAI in the community college. *Educational Technology*, *18*, 40-45.

Armbruster, B., & Brown, A. (1984). Learning from reading: The role of metacognition. In R. Anderson, J., Osborn, & R. Tierney (Eds.), *Learning to read in American schools: Basal readers and content tests* (pp. 273-281). Hillsdale, N.J.: Lawrence Erlbaum.

Anastasio, E.J., & Morgan, J.S. (1972). *Study of factors that have inhibited a more widespread use of computers in the instructional process*. Princeton: Educom.

Anderson, J. (1984). *Computing in schools*. Australian Education Review No. 21. Hawthorn: ACER.

Anderson, J.R. (1980). *Cognitive psychology and its implications*. San Francisco: W.H. Freeman and Company.

Anderson, J.R. (1982). Acquisition of cognitive skills. *Psychological Review*, *89*(4), 369-406.

Anver, R.A. (1978). Cost-effective applications of computer-based education. *Educational Technology*, *18*, 24-25.

Anzai, Y., & Simon, H.A. (1979). The theory of learning by doing. *Psychological Review*, *86*(21), 124-140.

Atkinson, R.C., & Shiffrin, R.M. (1968). Human memory: A proposed system and its control processes. In K.W. Spence & J.T. Spence (Eds.), *The psychology of learning and motivation: Advances in theory and research (Vol. 2)*. N.Y.: Academic Press.

Au, W.K. (1986, December). *Logo programming and problem solving*. Paper presented at the annual meeting of the New Zealand Association of Research in Education, Hamilton, New Zealand.

Au, W.K. (1988a, August). *Logo programming and problem solving*. Paper presented at a seminar of the N.S.W. Institute of Educational Research (Newcastle Branch), Newcastle.

Au, W.K. (1988b). Using computers in the teaching of English as a second language. *Proceedings of the Hunter Region English as a Second Language Conference* (pp. 42-50). Newcastle: University of Newcastle.

Au, W.K. (1992a). Logo research - where is it heading? In *Computing the clever country* (pp. 232-243). Melbourne: Computing in Education Group of Victoria.

Au, W.K. (1992b). Using computers to develop problem solving skills. In S.W. Wawrzyniak & L. Samootin (Eds.) *Kids, Curriculum, Computers and ..."* (pp. 27-37). Sydney: New South Wales Computer Education Group.

Au, W.K., & Bruce, M. (1990). Using Computers in Special Education. *Journal of Remedial Education*, *22*(1-2), 13-18.

Au, W.K., & Cook, T.E. (1989). *Joint venture between university and private sector in the development of computer assisted instruction software.* Paper presented at a research seminar at the University of Newcastle, Newcastle, Australia.

Au, W.K., & Horton, J. (1986). Logo: A process-oriented approach. *Computer News and Views for Wanganui Board Primary Schools*, *3*, 2-9.

Au, W.K., & Horton, J. (1987). Teaching Logo with a process-oriented approach. *Delta*, *39*, 49-58.

Au, W.K., Horton, J. & Ryba, K. (1987). Logo, teacher intervention, and the development of thinking skills. *The Computing Teacher*, *15*(3), 12-16.

Au, W.K., & Leung, J.P. (1988). Transfer of problem solving in Logo programming. In P. Alp (Ed.) *Golden opportunities* (pp. 46-59). Mount Lawley: Educational Computing Association of Western Australia.

Au, W.K., & Leung, J.P. (1991). Problem solving, instructional methods, and Logo programming. *Journal of Educational Computing Research*, *7*(4), 455-467.

Australian Information Industry Association (1990). *Towards a new approach to computing education in schools.* ACT: Australian Information Industry Association.

Avner, A, Moore, C., & Smith, S. (1980). Active external control: A basis for suepriority of CBI. *Journal of Computer-based Instruction*, *6*, 115-118.

Babbs, P.J., & Moe, A.J. (1983). Metacognition: A key for independent learning from text. *Reading Teacher*, *36*(4), 422-426.

Baker, L., & Brown, A.L. (1984). Metacognitive skills in reading. In D.P. Pearson (Ed), *Handbook of Reading Research* (pp. 353-394). N.Y.: Longman.

Barlett, F. (1958). *Thinking, an experimental and social study*. London: Allen & Unwin Ltd.

Barnes, B.J., & Hill, S. (1983). Should young children work with microcomputers - Logo before Lego. *The Computing Teacher*, *10*(9), 11-14.

Baron, J. (1978). Intelligence and general strategies. In G. Underwood (Ed.), *Strategies of information processing* (pp. 403-450). London: Academic Press.

Baron, J. (1981). Reflective thinking as a goal of education. *Intelligence*, *5*, 291-309.

Baron, J. (1985). *Rationality and intelligence.* Cambridge, England: Cambridge University Press.

Battista, M. & Clements, D. (1986, April). *The effects of Logo and CAI problem-solving environments on elementary students' problem-solving processes and mathematics achievement.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Bearden, D. (1988). Thinking in Logo - A review. *The Computing Teacher*, *15*(6), 41-2.

Bearison, D.J. (1982). New directions in studies of social interaction and cognitive growth. In F.C. Serafica (Ed.), *Social-cognitive development in context* (pp. 199-221). N.Y.: Guilford.

Becker, H.J. (1985, April). *National school uses of microcomputers survey: Review of past and promise of future data.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Becker, H.J. (1986). Instructional uses of school computers: Reports from the 1985 national survey. *Issues 1-3*. Center for Social Organization of Schools, John Hopkins University.

Becker, H.J. (1987). The importance of a methodology that maximizes falsifiability: Its applicability to research about Logo. *Educational Researcher*, *16*(5), 11-16.

Becker, H.J., & Sterling, C.W. (1987). Equity in school computer use: National data and neglected considerations. *Journal of Educational Computing Research*, *3*(3), 289-311.

Belmont, J.M., & Butterfield, E.C. (1977). The instructional approach to the developmental cognitive research. In R.V. Kail, Jr., & J.W. Hagen (Eds.), *Perspectives on the development of memory and cognition*, (pp. 437-481). Hillsdale, N.J.: Erlbaum.

Belmont, J.M., Butterfield, E.C., & Ferretti, R.P. (1982). To secure transfer of training instructions in self-management skills. In D. Detterman & R. Sternberg (Eds.), *How and how much can intelligence be increased*, (pp. 147-154). N.J.: Ablex Publishing Corporation.

Bergin, D.A., Ford, M.E., & Mayer-Gaub, G. (1986, April). *Social and motivational consequences of microcomputer use in kindergarten.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Berkowitz, S.J. (1986). Effects of instruction in text organization on sixth-grade. *Reading Research Quarterly*, *XXI*(2), 161-178.

Berry, D.C. (1983). Metacognitive experience and transfer of logical reasoning. *Quarterly Journal of Experimental Psychology*, *35A*, 39-49.

Biggs, J.B. (1987). *Student approaches to learning and studying*. Hawthorne, Victoria: Australian Council for Educational Research.

Billstein, R. (1983). Turtle Fever. *The Computing Teacher*, *11*(2), 34-36.

Bitter, G.G., & Camuse, R.A. (1988). *Using A Microcomputer In The Classroom. 2nd Edition*. N.J.: Prentice Hall.

Black, J.B., Swan, K., & Schwartz, D.L. (1988). Developing thinking skills with computers. *Teachers College Record*, *89*(3), 384-407.

Blaschke, C.L. (1979). Microcomputer software development for schools: What, who, how? *Educational Technology*, *19*, 26-28.

Blitman, E., Jamile, B., & Yee, D. (1984). Computers, children and epistemology. *Educational Perspectives*, *22*(4), 16-20.

Bloom, B.S. (Ed.). (1956). *Taxonomy of educational objectives. Handbook: Cognitive domain.* N.Y.: David McKay.

Bloom, B.S. (1976). *Human characteristics and school learning*. N.Y.: McGraw-Hill.

Bondy, E. (1984). Thinking about thinking. *Childhood Education*, 234-238.

Boren, S. (1984). *An apple for kids*. Beaverton, Oregon: Dilithium Press.

Bork, A, & Franklin, S. (1979). Personal computers in learning. *Educational Technology*, *19*, 7-12.

Bork, A. (1978). Machines for computer-assisted learning. *Educational Technology*, *18*, 17-20.

Bork, A. (1980). Interactive learning. In R. Taylor (Ed.), *The computer in the school* (pp. 53-66). N.Y.: Teachers College, Columbia University.

Bork, A. (1987). *Learning with personal computers.* N.Y.: Harper & Row.

Borko, H., Livingston, C., & Shavelson, R.J. (1990). Teachers' thinking about instruction. *Remedial and Special Education*, *11*(6), 40-49.

Borkowski, J.G., & Cavanaugh, J.C. (1981). Metacognition and intelligence theory. In M.P. Friedman, J.P. Das, & N. O'Connor (Eds.), *Intelligence And learning* (pp. 253-258). N.Y.: Plenum Press in cooperation with NATO Scientific Affairs Division.

Bornet, R., & Brady, J. (1974). The linguistics of Logo. *Memo No. CSM-4*. Essex: University of Essex.

Bourne, L.E. (1970). Knowing and using concepts. *Psychological Review*, *77*(6), pp. 546-556.

Bourne, L.E., & O'Banion, K. (1971). Conceptual rule learning and chronological age. *Developmental Psychology*, *5*(3), pp.525-534.

Bradley, C.A. (1985). The relationship between students' information-processing styles and Logo programming. *Journal of Educational Computing Research*, *1*(4), 427-394.

Brady, M.P., O'Donoghue, J., & Bajpai, A.C. (1989). Top down programming for schools. *The Computing Teacher*, *16*(6), 29-34.

Bramble, W.J., & Mason, E.M., & Berg, P. (1985). *Computers in schools*. N.Y.: McGraw-Hill Book Company.

Bransford, J.D. (1979). *Human cognition*. Belmont, C.A.: Wadsworth Publishing.

Bransford, J.D. (1984). *The ideal problem solver*. N.Y.: Freeman.

Bransford, J.D., & Stein, B.S. (1984). *The ideal problem solver*. N.Y.: W.H. Freeman.

Bransford, J.D., Stein, B.S., Arbitman-Smith, R., & Vye, N.J. (1985). Improving thinking and learning skills: An analysis of three approaches. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 133-206). N.J.: Lawrence Erlbaum.

Bransford, J.D., Stein, B., Delclos, V.R., & Littlefield, J. (1986). Computers and problem solving. In J.D. Bransford, B. Stein, V.R. Delclos & J. Littlefield (Eds.), *Computer strategies for education: Foundations and content-area applications* (pp. 147-180). Columbus: Merrill Publishing Company.

Breznitz, Z. (1987). Increasing first graders' reading accuracy and comprehension by accelerating their reading rates. *Journal of Educational Psychology*, *79*(3), 236-242.

Broadley, K., & Au, W.K. (1988). Word processing and process writing. In V. Bickley (Ed.), *Languages in education in a bi-lingual or multi-lingual setting* (pp. 290-302). Hong Kong: Institute of Language in Education.

Brown, A.L. (1978). Knowing when, where, and how to remember: A problem of metacognition. In R. Glaser (Ed.), *Advances in instructional psychology Vol. 1* (pp. 77-165). Hillsdale, N.J.: Erlbaum.

Brown, A.L. (1980). Metacognitive development and reading. In R.J. Spiro, B.C. Bruce, & W.F. Brewer (Eds.), *Theoretical issues in reading comprehension* (pp. 453-481). N.J.: Lawrence Erlbaum.

Brown, A.L. (1982). Learning and development: The problems of compatibility, access and induction. *Human Development*, *25*, 89-115.

Brown, A.L. (1985). Mental orthopedics, the training of cognitive skills: An interview with Alfred Binet. In S.F. Chipman, J.W. Segal, & R. Glaser (Eds.), *Thinking and learning skills: Vol 2. Research and open questions* (pp. 319-337). N.J.: Lawrence Erlbaum.

Brown, A.L., Bransford, J.D., Ferrara, R.A., & Campione, J.C. (1983). Learning, remembering, and understanding. In J.H. Flavell & E.M. Markman (Eds.), *Handbook of child psychology: Vol 1. Cognitive development* (pp. 77-166). N.Y.: Wiley.

Brown, A.L., & Campione, J.C. (1982). Modifying intelligence or modifying cognitive skills: More than a semantic quibble? In D.K. Detterman and R.J. Sternberg (Eds.), *How and how much can intelligence be increased* (pp. 215-230). N.J.: Ablex Publishing Corporation.

Brown, A.L., Campione, J.C., & Barclay, C.R. (1979). Training self-checking routines for estimating test readiness: Generalization from list learning to prose recall. *Child Development, 50*, 501-512.

Brown, A.L., Campione, J.C., & Day, J.D. (1981). Learning to learn: On training students to learn from text. *Educational Researcher, 10*(2), 14-21.

Brown, A.L., & DeLoache, J.S. (1978). Skills, plans and self-regulation. In R.S. Siegler (Ed.), *Children's thinking: What develops*, (pp. 3-35). Hillsdale, N.J.: Erlbaum.

Brown, J.S. (1985). Process versus product: A perspective on tools for communal and informal electronic learning. *Journal of Educational Computing Research, 1*(2), 179-201.

Brown, J.S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*(1), 32-42.

Bruce, M.E., & Chan, L.K.S. (1991). Reciprocal teaching and transenvironmental programming: A program to facilitate the reading comprehension of students with reading difficulties. *Remedial and Special Education, 12*(5), 44-54.

Bull, G., & Tipps, S. (1983-84). Problem spaces in a project-oriented Logo environment. *The Computing Teacher, 11*(5), 54-57.

Burns, B., & Coon, H. (1990). Logo programming and peer interactions: An analysis of process- and product-oriented collaborations. *Journal of Educational Computing Research, 6*(4), 393-410.

Burns, B, & Hagerman, A. (1989). Computer experience, self-concept and problem-solving: The effects of Logo on children's ideas of themselves as learners. *Journal of Educational Computing Research, 5*(2), 199-212.

Burns, H.L., & Capps, C.G. (1988). Foundations of intelligent tutoring systems: an introduction. In M.C. Polson & J.J. Richardson (Eds.), *Foundations of intelligent tutoring systems*, (pp. 1-19). Hillsdale, N.J.: Lawrence Erlbaum.

Burton, J.K., & Magliaro, S.G. (1986, April). *Computer programming and generalized problem solving skills: a critical review of the literature.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Butterfield, E.C., & Belmont, J.M. (1977). Assessing and improving the executive cognitive functions of mentally retarded people. In I. Bialer & M. Sternlicht (Eds.), *Psychological issues in mental retardation.* Chicago: Aldine.

Byrt, P.N. (1986). A tool to think with: A new look at some old problems. *Australian Educational Computing, 1*(1), 24-29.

Calfee, R. (1985). Computer literacy and book literacy: Parallels and contrasts. *Educational Researcher, 14*(5), 8-13.

Calvert, S.L., Watson, J.A., Brinkley, V.M., & Bordeaux, B. (1989). Computer presentational features for young children's preferential selection and recall of information. *Journal of Educational Computing Research, 5*(1), 35-49.

Campbell, P.F., Fein, G.G., & Scholnick, E.K. (1986, April). *Young Children's learning of Logo positioning commands: A conceptual model.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Campbell, P.F., Fein, G.G., Scholnick, E.K., Schwartz, S.S., Frank, & R.E. (1986). Initial Mastery of the Syntax and Semantics of Logo. *Journal of Educational Computing Research, 2*(3), 357-378.

Campbell, P.F., Fein, G.G., & Schwartz, S.S. (1991). The effects of Logo experience on first-grade children's ability to estimate distance. *Journal of Educational Computing Research*, *7*(3), 331-349.

Campione, J.C., & Brown, A.L. (1978). Toward a theory of intelligence: Contributions from research with retarded children. *Intelligence*, *2*, 279-304.

Campione, J.C., & Brown, A.L., & Ferrara, R.A. (1983). Mental retardation and Intelligence. In R.J. Sternberg (Ed.), *Handbook of human intelligence* (pp. 392-490). Cambridge: Cambridge University Press.

Carmichael, H.W., Burnett, J.D., Higginson, W.C., Moore, B.G., & Pollard, P.J. (1985). *Computers, children, and classrooms: A multisite evaluation of the creative use of microcomputers by elementary school children.* Ontario: Queen's Printer for Ontario.

Carver, S.M., & Klahr, D. (1986). Assessing children's Logo debugging skills with a formal model. *Journal of Educational Computing Research*, *2*(4), 487-525.

Cathcart, W.G. (1990). Effects of Logo instruction on cognitive style. *Journal of Educational Computing Research*, *6*(2), 231-242.

Cawley, J.F., & Miller, J.H. (1989). Cross-sectional comparisons of the mathematical performance of children with learning disabilities: Are we on the right track toward comprehensive programming? *Journal of Learning Disabilities*, *22*(4), 250-254, 259.

Chambers, S.M. (1984a, July). *Development of thinking in primary school children using Logo.* Paper presented at the IT, AI and Child Development Conference, University of Sussex, Sussex.

Chambers, S.M. (1984b). The development of thinking in primary school children using Logo. In J. Hughes (Ed.), *Dreams and reality* (pp. 67-72). Broadway: Computer Education Group of New South Wales.

Chambers, S.M. (1986). So Papert was right?: Evidence of general skills enhancement due to Logo experience. In A.D. Salvas & C. Dowling (Eds.). *On the crest of a wave?* (pp. 261-264). Balaclava, Victoria: Computer Education Group of Victoria.

Chan, L.K.S. (1988). The role of language in metacognitive instruction. In V. Bickley (Ed.), *Languages in education in a bi-lingual or multi-lingual setting* (pp. 18-28). Hong Kong: Institute of Language in Education.

Chan, L.K.S. (1990). Metacognition and remedial education. *Australian Journal of Remedial Education*, *23*(1), 4-10.

Chan, L.K.S. (1991). Promoting strategy generalization through self-instructional training in students with reading disabilities. *Journal of Learning Disabilities*, *24*(7), 427-433.

Chan, L.K.S., Cole, P.G., & Morris, J.N. (1990). Effects of instruction in the use of a visual-imagery strategy on the reading-comprehension competence of disabled and average readers. *Learning Disability Quarterly*, *13*, 2-11.

Chance, P. (1986). *Thinking in the classroom: A survey of programs.* N.Y.: Teachers College Press.

Chandler, P.D., Gesthuizen, R.J., & Clement, J.D. (1992). Raising the level of access to computer communications in schools. In *Computing the clever country* (pp. 197-29). Melbourne: Computing in Education Group of Victoria.

Chapman, J.W., & Ryba, K. (1983). Towards improving learning strategies and personal adjustment with computers in education. *The Computing Teacher*, *11*(1), 48-53.

Char, C.A. (1984). *Research and design issues concerning the development of educational software for children. Technical report No. 14.* N.Y.: Bank Street College of Education, Center for Children and Technology.

Chiang, B., Thorpe, H.W., & Lubke, M. (1984). LD students tackle the Logo language: strategies and implications. *Journal of Learning Disabilities*, *17*(5), 303-304.

Chipman, S.F., & Segal, J.W. (1985). Higher cognitive goals for education: An introduction. In S.F. Chipman, J.W. Segal, & R. Glaser (Eds.), *Thinking and learning skills: Vol 2. Research and open questions* (pp. 1-19). N.J.: Lawrence Erlbaum.

Clampit, M.K., & Silver, S.J. (1989). Distribution of relative attention deficits on the WISC-R by age, sex, social class, and region. *Journal of Learning Disabilities*, *22*(4), 258-259.

Clancey, W.J. (1988). The knowledge engineer as student: Metacognitive bases for asking good questions. In H. Mandl and A. Lesgold (Eds.), *Learning issues for intelligent tutoring system* (pp. 80-113). N.Y.: Springer-Verlagg N.Y. Inc.

Clarke, V.A., & Chambers, S.M. (1984a). Developing Logo activities for the primary classroom. In A.D. Salvas (Ed.), *Computing and education - 1984 and beyond*, (pp. 309-312). Balacava: Computer Education Group of Victoria.

Clarke, V.A., & Chambers, S.M. (1984b, September). *Classroom activities using Logo.* Paper presented at the Australian Computer Education Conference, Macquarie University, Sydney.

Clayson, J. (1982). Computer games teach problem solving. *Impact of science on society*, *32*(4), 435-441.

Clement, C.A., Kurland, D.M., Mawby, R., & Pea, R.D. (1986). Analogical reasoning and computer programming. *Journal of Educational Computing Research*, *2*(4), 473-486.

Clements, D.H. (1981). Affective considerations in computer based education. *Educational Technology*, January, 37-39.

Clements, D.H. (1985a). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology*, *78*(4), 309-318.

Clements, D.H. (1985b, April). *Effects of Logo programming on cognition, metacognition skills, and achievement.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Clements, D.H. (1985c). Research on Logo in Education: Is the turtle slow but steady, or not even in the race? *Computers in the schools*, *2*(2/3), 55-71.

Clements, D.H. (1986a, April). *Delayed effects of computer programming in Logo on mathematics and cognitive skills.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Clements, D.H. (1986b, April). *Learning and teaching Logo: An information-processing perspective.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Clements, D.H. (1986c). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology*, *78*(4), 309-318.

Clements, D.H. (1987a). *Computers in early and primary education.* Englewood Cliffs, N.J.: Prentice-Hall.

Clements, D.H. (1987b). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research*, *3*(1), 73-94.

Clements, D.H. (1988). Problem-solving processes: The mental company. *Logo Exchange*, *7*(3), 27-29.

Clements, D.H. (1989a). The nature of the problem. *Logo Exchange*, *7*(6), 28-29.

Clements, D.H. (1989b). Planning for planning. *Logo Exchange*, *7*(8), 26-27.

Clements, D.H. (1990). Metacomponential development in a Logo programming environment. *Journal of Educational Psychology*, *82*(1), 141-149.

Clements, D.H. (1991). Enhancement of creativity in computer environments. *American Educational Research Journal, 28*(1), 173-181.

Clements, D.H., & Gullo, D.F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*(6), 1051-1058.

Clements, D.H., & Merriman, S. (1988). Componential developments in LOGO programming environments. In R.E. Mayer (Ed.), *Teaching and learning computer programming: Multiple research perspectives* (pp. 13-54). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Clements, D.H., & Nastasi, B.K. (1988). Social and cognitive interactions in educational computer environments. *American Educational Research Journal, 25*(1), 87-106.

Coburn, P., Kelman, P., Roberts, N., Snyder, T.F.F., Watt, D.H., & Weiner, C. (1982). *Practical guide to computers in education*. Reading, Massachusetts: Addison-wesley Publishing Company.

Cohen, R. (1987). Implementing Logo in the grade two classroom: Acquisition of basic programming concepts. *Journal of Computer-Based Instruction, 14*(4), 124-132.

Cohen, R. (1990). Computerized learning supports in pre-Logo programming environments. *Journal of Research on Computing in Education, 22*(3), 310-335.

Coleman, L.J. (1986, April). *Reflecting on one's problem solving using videogames.* Paper presented at the annual meeting of the American Education Research Association, San Francisco.

Collier, P.A., & Samson, W.B. (1982). Prolog as a teaching tool for relational database interrogation. *Computer Education*, November, 1982, 26-27.

Collins, A., & Smith, E.E. (1982). Teaching the process of reading comprehension. In D.K. Detterman and R.J. Sternberg (Eds.), *How and how much can intelligence be increased* (pp. 173-185). N.J.: Ablex Publishing Corporation.

Conabere, T., & Anderson, J. (1985). *Towards a rationale for the educational use of computer technology in schools* (Occasional Paper No. 8). Carlton, Victoria: The Australian College of Education.

Coombs, M.J., Gibson, R., & Alty, J.L. (1981). Acquiring a first computer language: a study of individual differences. In M.J. Coombs & S.L. Ally (Eds.), *Computing skills and the user interface* (pp. 289-313). N.Y.: Academic Press.

Covington, M.V. (1985). Strategic thinking and the fear of failure. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 389-416). N.J.: Lawrence Erlbaum.

Cox, D.A.H. (1980). *Early adolescent use of selected problem solving skills using microcomputers* (ERIC Document Reproduction Service No. ED 200 449).

Critchfield, M. (1979). Beyond CAI: Computers as personal intellectual tools. *Educational Technology, 19*, 18-25.

Cuffaro, H.K. (1984). Microcomputers in education: Why is earlier better. *Teachers College Record, 85*(4), 561-568.

Cumming, G. (1985). Mental models - the world inside our heads. *COM 3, 11*(1), 7-9.

Cumming G., & Abbot, E. (1986). It's the educational strategy that matters, even if the language is Prolog. *Australian Educational Computing, 1*(1), 34-39.

Cuneo, D.O. (1986a, April). *Young children and turtle graphics programming: generating and debugging simple turtle programs.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Cuneo, D.O. (1986b, May). *Young children's misconceptions of simple turtle graphics commands*. Paper presented at the annual symposium of the Jean Piaget Society, Philadelphia.

Cyert, R.M. (1980). Problem solving and educational policy. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 3-23). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Dailhou, P. (1986). Computers and the process approach to writing. In B. Frederick (Ed.), *What to do with what you've got* (pp. 94-98). Broadway, N.S.W.: Computer Education Group of New South Wales Ltd.

Dalbey, J., & Linn, M.C. (1984, April). *Spider world: A robot language for learning to program. Assessing the cognitive consequences of computer environments for learning*. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Dalbey, J., & Linn, M.C. (1985). The demands and requirements of computer programming: A literature review. *Journal of Educational Computing Research*, *1*(3), 253-274.

Dalbey, J., & Linn, M.C. (1986). Cognitive consequences of programming: Augmentations to BASIC instruction. *Journal of Educational Computing Research*, *2*(1), 75-93.

Dalton, D.W., & Gooddrum, D.A. (1991). The effects of computer programming on problem-solving skills and attitudes. *Journal of Educational Computing Research*, *7*(4), 483-506.

Dansereau, D.F. (1978). The development of a learning strategies curriculum. In H.F. O'Neil (Ed.), *Learning strategies*, (pp. 35-61). N.Y.: Academic Press.

Dansereau, D.F. (1985). Learning strategy research. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 209-239). N.J.: Lawrence Erlbaum.

Day, J.D., Cordon, L.A., & Kerwin, M.L. (1989). Informal instruction and development of cognitive skills: A review and critique of research. In C.B. McCormick, G.E. Miller & M. Pressley (Eds.) *Cognitive strategy research: From basic research to educational applications* (pp. 83-103). N.Y.: Springer-Verlag.

Day, J., French, L.A., & Hall, L. (1985). Social influences on cognitive development. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller (Eds.), *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives* (pp. 33-56). Orlando, Florida: Academic Press.

de Bono, E. (1973). *CoRT thinking materials*. London: Direct Education Services.

de Bono, E. (1985). The CoRT thinking program. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 363-388). N.J.: Lawrence Erlbaum.

Degelman, D., Free, J.U., Scarlato, M., Blackburn, J.M., & Golden, T. (1986). Concept learning in preschool children: effects of a short-term Logo experience. *Journal of Educational Computing Research*, *2*(2), 199-205.

Delclos, V.R., & Kulewicz, S.J. (1986). Improving computer-based problem solving training: The role of the teacher as mediator. *Computers in Human Behavior*, *2*, 1-12.

Delclos, V.R., & Littlefield, J.D. (1984, April). *Does Logo lead to better learning*. Paper presented at the spring conference of the Tennessee Association for Educational Data System, Nashville, TN.

Dennis, J.R., & Kansky, R.J. (1984). *Instructional computing: An Action guide for educators*. Glenview, Illinois: Scott Foresman.

Derry, S.J. (1990). Remediating academic difficulties through strategy training: the acquisition of useful knowledge. *Remedial and Special Education*, *11*(6), 19-31.

Derry, S.J., Hawkes, L.W., & Tsai, C-J. (1987). A theory for remediating problem solving skills of older children and adults. *Educational Psychologist*, *22*, 55-87.

Derry, S.J., & Murphy, D.A. (1986). Designing systems that train learning ability: From theory to practice. *Review of Educational Research*, *56*(1), 1-39.

Deshler, D.D., Schumaker, J.B., & Lenz, B.K. (1984). Academic and cognitive interventions for LD adolescents (Part i). *Journal of Learning Disabilities*, *17*, 108-117.

Dewey, J. (1933). *How we think: A restatement of the relation of reflective thinking to the educative process.* Lexington, Massachusetts: D.C. Heath.

Dickson, W.P. (1985). Thought-provoking software: Juxtaposing symbol systems. *Educational Researcher*, *14*(5), 30-38.

D'Ignazio, F. (1991). The starship enterprise: New opportunities for classroom learning in the 1990s. In *Navigating the nineties* (pp. 3-7). Brisbane: Computer Education Group of Queensland.

Dillashaw, F.G., & Bell, S.R. (1985, April). *Learning outcomes of computer programming instruction for middle-grade students: A pilot study*. Paper presented at the annual meeting of the National Association for Research in Science Teaching, French Lick Springs, IN.

DiSessa, A.A. (1986). From Logo to Boxer, a new computational environment. *Australian Educational Computing*, *1*(1), 8-15.

Dolan, D.T., & Williamson, J. (1983). *Teaching problem-solving strategies.* Menlo Park, California: Addison-Wesley Publishing Company.

Doyle, W. (1983). Academic work. *Review of Educational Research*, *53*, 159-199.

du Boulay, J.B.H., & Howe, J.A.M. (1982). Logo building blocks: Student teachers using computer-based mathematics apparatus, *Computers & Education*, *6*, 92-98.

Duncker, K. (1945). On problem solving. *Psychological Monographs*, *58*(5), Whole No. 270.

Dyck, J.L., & Mayer, R.E. (1989). Teaching of transfer of computer program comprehension skill. *Journal of Educational Psychology*, *81*(1), 16-24.

Eimas, P.D. (1969). A developmental study of hypothesis behavior and focusing. *Journal of Experimental Child Psychology*, *8*, 160-172.

Eisner, E.W. (1981). On the differences between scientific and artistic approaches to qualitative research. *Educational Researcher*, *10*(4), 5-9.

Eisner, E.W. (1983). Anastasia might still be alive, but the monarchy is dead. *Educational Researcher*, *12*(5), 13-24.

Elley, W.B., & Reid, N.A. (1969). *Progressive Achievement Tests, Teacher's Manual: Reading Comprehension, Reading Vocabulary*. Wellington, N.Z.: New Zealand Council for Educational Research.

Elley, W.B., & Reid, N.A. (1971). *Progressive Achievement Tests, Teacher's Manual: Listening Comprehension.* Wellington, N.Z.: New Zealand Council for Educational Research.

Ellis, E.S., Lenz, K., & Sabornie, E.J. (1987a). Generalization and adaptation of learning strategies to natural environments: Part 1: Critical agents. *Remedial and Special Education*, *8*(1), 6-20.

Ellis, E.S., Lenz, K., & Sabornie, E.J. (1987b). Generalization and adaptation of learning strategies to natural environments: Part 2: Research into practice. *Remedial and Special Education*, *8*(2), 6-23.

Ellis, H. (1965). *The transfer of learning*. N.Y.: Macmillan.

Emihovich, C., & Miller, G.E. (1986, April). *Verbal mediation in Logo instruction: learning from a Vygotskian perspective.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Emihovich, C., & Miller, G.E. (1988). Learning Logo: The social context of cognition. *Journal of Curriculum Studies*, *20*(1), 57-70.

Emihovich, C., & Miller, G.E. (in press). Talking to the turtle: A discourse analysis of Logo instruction. *Discourse Processes.*

Evans, H.R. (1984). *Some effects of Logo programming instruction with fourth grade children.* Unpublished doctoral dissertation. Virginia: University of Virginia.

Fay, A.L., & Mayer, R.E. (1987). Children's naive conceptions and confusions about Logo graphics commands. *Journal of Educational Psychology*, *79*(3), 254-268.

Feibel, W. (1978). On applying metacognition to metacognition about metacognition: A redundant reaction? In J.M. Scandura & C.J. Brainerd (Eds.), *Structural/process models of complex human behaviour* (pp. 247-258). Alphen aan den Rijn, the Netherlands: Sijthoff & Noordhoff.

Fein, G.G., Scholnick, E.K., Campbell, P.F., Schwartz, S., & Frank, R. (1985, June). *Computing space.* Paper presented at the Jean Piaget Society Symposium, Philadelphia.

Feuerstein, R., Jensen, M., Hoffman, M.B., & Rand, Y. (1985). Instructional enrichment, an intervention program for structural cognitive modifiability: Theory and practice. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 43-82). N.J.: Lawrence Erlbaum.

Feuerstein, R., Rand, Y., Hoffman, M.B., Hoffman, M., & Miller, R. (1979). Cognitive modifiability in retarded adolescents: Effects of instrumental enrichment. *American Journal of Mental Deficiency*, *83*(6), 539-550.

Feurzeig, W. (1986, July). Towards intelligent microworlds. *Logo 86 Proceedings* (pp. 44-45). Cambridge, MA: Massachusetts Institute of Technology.

Feurzeig, W., Horwitz, P., & Nickerson, R.S. (1981). *Microcomputers in education.* Report for Department of Health, Education, and Welfare, National Institute of Education, and Ministry for the Development of Human Intelligence, Republic of Venezuela. Cambridge, M.A.: Bolt, Beranek & Newman.

Feurzeig, W., & Lukas, G. (1972). Logo: A programming language for learning mathematics. *Educational Technology*, March, 1972, 39-46.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). *Programming languages as a conceptual framework for teaching mathematics* (BBN Report no. 1889). Cambridge, Massachusetts: Bolt, Beranek and Newman.

Finlayson, H.M. (1983). *The development of mathematical thinking through Logo. D.A.I. Research paper no. 205.* Paper presented to the British Logo User's Group conference, September, 1983.

Finlayson, H.M. (1984). *What do children learn through using Logo? D.A.I. Research paper no. 237.* Paper presented to the British Logo Users Group conference, Loughborough, September, 1984.

Finlayson, H.M. (1985). *The transfer of mathematical problem solving skills from Logo experience. D.A.I. Research paper no. 238.* Paper presented to the World Conference of Computers in Education, Norfolk, Virginia, 1985.

Fisher, C., & Mandinach, E. (1985, April). *Individual differences and acquisition of computer programming skill.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Fisher, R. (1987). *Problem solving in primary schools.* Oxford: Basil Blackwell.

Flavell, J.H. (1970). Developmental studies of mediated memory. In H.W. Reese & L.P. Lipsitt (Eds.), *Advances In Child Development And Behaviour Vol. 5* (pp. 181-211).

Flavell, J.H. (1976). Metacognitive aspects of problem solving. In L.B. Resnick (Ed.), *The nature of intelligence* (pp. 231-235). N.J.: Lawrence Erlbaum Associates.

Flavell, J.H. (1977). *Cognitive development.* Englewood, N.J.: Prentice-Hall.

Flavell, J.H. (1978). *Metacognitive development.* In J.M. Scandura & C.J. Brainerd (Eds.), *Structural/process models of complex human behaviour* (pp. 213-245). Alphen aan den Rijn, the Netherlands: Sijthoff & Noordhoff.

Flavell, J.H. (1979). Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. *American Psychologist, 34*(10), 906-911.

Flavell, J.H., Speer, J.R., Green, F.L., & August, D.L. (1981). The development of comprehension monitoring and knowledge about communication. *Monographs of the society for research in child development, 46*(5), Serial No. 192.

Forrest-Pressley, D.L., MacKinnon, G.E., & Waller, T.G. (Eds.). (1985). *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives.* Orlando, Florida: Academic Press.

Frampton, A. (1989). Educational Computing in New Zealand. *Information Transfer, 9*(1), 37.

Frederick, B. (1986). Introduction to telecommunications. In B. Frederick (Ed.), *What to do with what you've got* (pp. 277-283). Broadway, N.S.W.: Computer Education Group of New South Wales Ltd.

Frederiksen, N. (1984). Implications of cognitive theory for instruction in problem solving. *Review of Educational Research, 54*(3), 363-407.

Freehand, G.A. (1966). Epilogue: Constructs and strategies for problem-solving research. In B. Kleinmuntz (Ed.), *Problem solving: Research, method, and theory* (pp. 355-383). N.Y.: John Wiley & Sons.

Freeman, C., Hawkins, J., & Char, C. (1985). *Information management tools for classrooms: Explore database management systems. Technical Report No. 28.* N.Y.: Centre for Children and Technology, Bank Stree College of Education.

Friedland, E., & Friedland, M. (1984). Beyond turtle graphics. *Logo and Educational Computing Journal, 2*(1), 7, 24-25.

Funkhouser, C., & Dennis, J.R. (1992). The effects of problem-solving software on problem-solving ability. *Journal of Research on Computing in Education, 24*(3), 338-347.

Furth, H.G. (1969). *Piaget and knowledge: Theoretical foundations.* Englewood Cliffs, N.J.: Prentice-Hall.

Gagné, R.M. (1966). Human problem solving: Internal and external events. In B. Kleinmuntz (Ed.), *Problem solving: Research, method, and theory* (pp. 127-148). N.Y.: John Wiley & Sons.

Gagné, R.M., & Briggs, L.J. (1974). *Principles of instructional design.* N.Y.: Holt, Rinehart and Winston.

Gallini, J.K. (1985). Instructional conditions for computer-based problem solving environments. *Educational Technology*, February, 1985, 7-11.

Gallini, J.K. (1987). A comparison of the effects of Logo and a CAI learning environment on skills acquisition. *Journal of Educational Computing Research, 3*(4), 461-477.

Gholson, B. (1980). *The cognitive-development basis of human learning: Studies in hypothesis testing.* N.Y.: Academic Press.

Gick, M.L., & Holyoak, K.J. (1980). Analogical Problem Solving. *Cognitive Psychology, 12*, 306-355.

Ginther, D.W., & Williamson, J.D. (1985). Learning Logo: What is really learned? *Computers in the Schools, 2*(2/3), 73-78.

Glaser, R., & Pellegrino, J. (1982). Improve the skills of learning. In D.K. Detterman and R.J. Sternberg (Eds.), *How and how much can intelligence be increased* (pp. 197-212). N.J.: Ablex Publishing Corporation.

Goldenberg, E.P. (1982). Logo - A cultural glossary. *Byte*, *7*(8), 210-228.

Gorman, H. Jr. (1982). The Lamplighter project. *Byte*, *7*(8), pp. 331-333.

Gorman, H. Jr., & Bourne, L.E. Jr. (1983). Learning to think by learning Logo: rule learning in third-grade computer programmers. *Bulletin of the Psychonomic Society*, *21*(3), 165-167.

Gourgey, A.F. (1987). Coordination of instruction and reinforcement as enhancers of the effectiveness of computer-assisted instruction. *Journal of Educational Computing Research*, *3*(2), 219-230.

Grabe, M., & Mann, S. (1984). A technique for the assessment and training of comprehension monitoring skills. *Journal of Reading Behaviour*, *XVI*(2), 131-144.

Grandgenett, N., & Thompson, A. (1991). Effects of guided programming instruction on the transfer of analogical reasoning. *Journal of Educational Computing Research*, *7*(3), 293-308.

Grauer, R.T., & Gordon, J., & Schemel, M. (1984). *Basic is child's play*. Englewood Cliffs, N.J.: Prentice-Hall Inc.

Gray, A., & Bell, S. (1991). Multimedia and multimodal texts. In *Navigating the nineties* (pp. 280-287). Brisbane: Computer Education Group of Queensland.

Green, B.F. (1966). Current trends in problem solving. In B. Kleinmuntz (Ed.), *Problem solving: Research, method, and theory* (pp. 3-18). N.Y.: John Wiley & Sons.

Greeno, J.G. (1978a). A study of problem solving. In R. Glaser (Ed.), *Advances in instructional psychology Vol. 1* (pp. 13-75). N.J.: Lawrence Erlbaum Associates.

Greeno, J.G. (1978b). Natures of problem-solving abilities. In W.K. Estes (Ed.), *Handbook of learning and cognitive processes, Vol. 5, Human informational processing* (pp. 239-270). Hillsdale, N.J.: Lawrence Erlbaum Associates, Publishers.

Greeno, J.G. (1985). Looking across the river: Views from the two banks of research and development in problem solving. In S.F. Chipman, J.W. Segal, & R. Glaser (Eds.), *Thinking and learning skills: Vol 2. Research and open questions* (pp. 209-213). N.J.: Lawrence Erlbaum.

Groen, G. (1984). Theories in Logo. In R. Sorkin (Ed.), *Pre-proceedings of the national logo conference, Logo 84* (pp. 49-54). Cambridge: MIT.

Grogono, P., & Nelson, S.H. (1982). *Problem solving And computer programming*. Reading, Massachusetts: Addison-Wesley Publishing Company.

Guntermann, E., & Tovar, M. (1987). Collaborative problem-solving with Logo: Effects of group size and group composition. *Journal of Educational Computing Research*, *3*(3), 313-334.

Guthrie, J.T., & Kirsch, I.S. (1987). Distinctions between reading comprehension and locating information in text. *Journal of Educational Psychology*, *79*(3), 220-227.

Guttormsen, R. (1986). Computer based information handling in the primary school: Learning tool or simply awareness? In R. Guttormsen (Ed.), *Computers in the curriculum - Realising the potential*. Brisbane: Computer Education Group of Queensland.

Halford, G.S. (1988). Cognitive process approaches to intelligence. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 61-72). Hawthorn, Victoria: Australian Council of Educational Research.

Hall, R. (1988). Does the concept of intelligence have a future? A discussion of the Ward and Halford papers. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 73-75). Hawthorn, Victoria: Australian Council of Educational Research.

Haller, E.P., Child, D.A., & Walberg, H.J. (1988). Can comprehension be taught? A quantitative synthesis of "metacognitive" studies. *Educational Researcher*, *17*(9), 5-8.

Harper, D. (1989). *Logo: Theory & Practice.* Pacific Grove, California: Brooks/Cole Publishing Company.

Harvey, B. (1982a). Why Logo? *Byte*, *7*(8), 163-193.

Harvey, B. (1982b). Why Logo? In M. Yazdani (Ed.), *New horizons in educational computing* (pp. 21-39). Chichester: Ellis Horwood.

Hassett, J. (1984). Computers in the classroom. *Psychology Today*, *18*(9), 22-28.

Hativa, N. (1988). Computer-based drill and practice in arithmetic: Widening the gap between high- and low-achieving students. *American Educational Research Journal*, *25*(3), 366-397.

Hawkins, J. (1983). Learning Logo together: the social context. In K. Sheingold (Ed), *Chamelon in the classroom: Developing roles for computers* (pp.40-49). N.Y.: Bank Street College of Education, Center for Children and Technology.

Hawkins, J. (1987). The interpretation of Logo in practice. In R.D. Pea & K. Sheingold (Eds.) *Mirrors of minds: Patterns of experience in educational computing* (pp. 3-34). Norwood, N.J.: Ablex Publishing Corporation.

Hawkins, J., Homolsky, M., & Heide, P. (1984). *Paired problem solving in a computer context. Technical report No. 33.* N.Y.: Bank Street College of Education, Center for Children and Technology.

Hayes, J.R. (1966). Memory, goals and problem solving. In B. Kleinmuntz (Ed.), *Problem solving: Research, method, and theory* (pp. 149-170). N.Y.: John Wiley & Sons.

Hayes, J.R. (1980). Teaching problem-solving mechanisms. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 141-147). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Hayes, J.R. (1981). *The complete problem solver.* P.A.: Franklin Institute Press.

Hayes, J.R. & Simon, H.A. (1977). Psychological differences among problem isomorphs. In N. Castellan, Jr., D. Pisoni, & G. Potts (Eds.), *Cognitive theory Vol. II* (pp. 138 -165). Hillsdale, NJ: Erlbaum.

Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science*, *3*, 275-310.

Heller, R.S. (1991). Toward a student workstation: extensions to the Logo environment. *Journal of Educational Computing Research*, *7*(1), 77-88.

Herriott, R.E., & Firestone, W.A. (1983). Multisite qualitative policy research: Optimizing description and generalizability. *Educational Researcher*, *12*(2), 14-19.

Hess, R.D., & McGarvey, L.J. (1987). School-relevant effects of educational uses of microcomputers in kindergarten classrooms and homes. *Journal of Educational Computing Research*, *3*(3), 269-287.

Higgenson, W. (1982). Leading fish to water: Early observations on the use of Logo. *Byte*, *7*(8), 328-329.

Hillel, J. (1985). For the learning of mathematics. *An International Journal of Mathematics Education*, *5*(2), 38-45.

Hofmeister, A. (1984). *Microcomputer applications in the classroom.* N.Y.: Holt, Rinehart & Winston.

Hooper, S., & Hannafin, M.J. (1988). Cooperative CBI: The effects of heterogeneous versus homogeneous grouping on the learning of progressively complex concepts. *Journal of Educational Computing Research*, *4*(4), 413-424.

Horner, C.M., & Maddux, C.D. (1985). The effects of Logo on attributions toward success. *Computers in the Schools*, *2*(2/3), 45-54.

Horton, J.H., & Ryba, K.A. (1986). Assessing learning with Logo: A pilot study. *The Computing Teacher*, *14*(1), 24-28.

Horton, J.K. (1986). *Assessing children's thinking skills with Logo*. Unpublished Diploma of Education Dissertation. Massey University, Palmerston North, New Zealand.

Howard, J.R., Busch, J.C., & Watson, J.A. (1992). The change-over to computer-based technology in early childhood special education. *Journal of Research on Computing in Education*, *23*(4), 530-544.

Howe, J., Ross, P., Johnston, K., Plane, F., & Inglis, R. (1981). *Teaching mathematics through programming in the classroom. Research paper No. 157*. Edinburgh: Department of Artificial Intelligence, University of Edinburgh.

Howe, J.A.M., O'Shea, T., & Plane, F. (1980). Teaching mathematics through Logo programming: an evaluation study. In R. Lewis & D. Tagg (Eds.) *Computer assisted learning scope, progress and limits* (pp. 85-102). Amsterdam, Holland: North Holland.

Howe, K.R. (1985). Two dogmas of educational research. *Educational Researcher*, *14*(8), 10-18.

Howe, K.R. (1988). Against the quantitative-qualitative incompatibility thesis or dogmas die hard. *Educational Researcher*, *17*(8), 10-16.

Howell, R.D., Scott, P.B., & Diamond, J. (1987). The effects of "instant" Logo computing language on the cognitive development of very young children. *Journal of Educational Computing Research*, *3*(2), 249-260.

Hoyles, C., & Noss, R. (1985). *Synthesising mathematical conceptions and their formalisation through the construction of a Logo-based school mathematics curriculum*. London: Department of mathematics, statistics and computing, University of London, Institute of Education.

Hoyles, C., & Sutherland, R. (1987). Ways of learning in a computer-based environment: some findings of the Logo Maths Project. *Journal of Computer Assisted Learning*, *3*, 67-80.

Hoyles, C., & Sutherland, R. (1989). *Logo mathematics in the classroom*. London: Routledge.

Hughes, M., & Macleod, H. (1986). Why Logo for very young children. In R.W. Lawler, B. du Boulay, M. Huges, & H. Macleod (Eds.) *Cognition and computers* (pp. 180-181). Sussex: Ellis Horwood Limited.

Hughes, M., Macleod, H., & Patts, C. (1985). Using Logo with infant school children. *Edinburgh Educational Psychology*, *5*(3-4), 287-301.

Humphrey, G. (1951). *Thinking: An introduction to its experimental psychology*. N.Y.: Wiley.

Hunter, B. (1985). Problem solving with data bases. *The Computing Teacher*, *12*(8), 20-27.

Inhelder, B., & Piaget, J. (1958). *The growth of logical thinking*. London: Routledge & Kegan Paul Ltd.

Irwin, K. (1985, August). *Form II children working with Logo and Basic*. Paper presented to the First National Conference of the New Zealand Computer Education Society, Auckland.

Jacob, E. (1988). Clarifying qualitative research: A focus on traditions. *Educational Researcher*, *17*(1), 16-19, 22-24.

James, L. (1986). Logo study. In A.D. Salvas & C. Dowling (Eds.). *On the crest of a wave?* (pp. 190-193). Balaclava, Victoria: Computer Education Group of Victoria.

Jamison, D., Suppes, P., & Wells, S. (1974). Effectiveness of alternative instructional media. *Review of Educational Research*, *44*, 1-67.

Jewson, J., & Pea, R.D. (1982). Logo research at Bank Street College. *Byte*, *7*(8), 332-333.

Johanson, R.P. (1988). Computers, cognition and curriculum: retrospect and prospect. *Journal of Educational Computing Research*, *4*(1), 1-30.

Jones, A.J. (1992). Encouraging teachers to use computers: A necessary step toward a clever country. In *Computing the clever country* (pp. 322-329). Melbourne: Computing in Education Group of Victoria.

Johnson, D.L. (1985). What do we know about Logo? *Computers in the Schools*, *2*(2/3), 1-2.

Johnson, D.M. (1955). *The psychology of thought and judgment*. N.Y.: Harper & Row.

Johnson, R.T., Johnson, D.W., & Stanne, M.B. (1986). Comparison of computer-assisted cooperative, competitive, and individualistic learning. *American Educational Research Journal*, *23*(3), 382-392.

Karat, J. (1982). A model of problem solving with incomplete constraint knowledge. *Cognitive Psychology*, *14*, 538-559.

Keat, J.A. (1988). Difficulties in studying the intellect: Comments on the Halford and Ward papers. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 76-77). Hawthorn, Victoria: Australian Council of Educational Research.

Keller, J.K. (1990). Characteristics of Logo instruction promoting transfer of learning: A research review. *Journal of Research on Computing in Education*, *23*(1), 55-71.

Kendall, C.R., Borkowski, J.G., & Cavanaugh, J.C. (1980). Metamemory and the transfer of an interrogative strategy by EMR children. *Intelligence*, *4*, 255-270.

Kersteen, Z.A., Linn, M.C., Clancy, M., & Hardyck, C. (1988). Previous experience and the learning of computer programming: The computer helps those who help themselves. *Journal of Educational Computing Research*, *4*(3), 321-333.

Khayrallah, M.A., & van den Meiraker, M. (1987). Logo programming and the acquisition of cognitive skills. *Journal of Computer-Based Instruction*, *14*(4), 133-137.

King, A. (1989). Verbal interactions and problem-solving within computer-assisted cooperative learning groups. *Journal of Educational Computing Research*, *5*(1), 1-15.

Kinzer, C., Littlefield, J., Delclos, V.R., & Bransford, J.D. (1985). Different Logo learning environments and mastery: Relationships between engagement and learning. *Computers in the Schools*, *2*(2/3), 33-44.

Kinzer, C.K., Sherwood, R.D., & Bransford, J.D. (1986). *Computer strategies for education: Foundations and content-area applications*. Columbus: Merrill Publishing Company.

Klahr, D., & Robinson, M. (1981). Formal assessment of problem solving and planning processes in preschool children. *Cognitive Psychology*, *13*, 113-148.

Kleinmuntz, B. (Ed.). (1966). *Problem solving: Research, method, and theory*. N.Y.: John Wiley & Sons.

Köhler, W. (1927). *The mentality of apes*. N.Y.: Harcourt Brace.

Kolata, G. (1982). How can computer get common sense? *Science*, *217*, 1237-1238.

Kozmetsky, G. (1980). The significant role of problem solving in education. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 151-157). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Krendl, K.A., & Lieberman, D.A. (1988). Computers and learning: A review of recent research. *Journal of Educational Computing Research*, *4*(4), 367-389.

Krasnor, L., & Mitterer, J. (1984). Logo and the development of general problem solving skills. *The Alberta Journal of Educational Research*, *30*(2), 133-144.

Kuhn, D. (1990). Introduction. In D. Kuhn (Ed.), *Developmental perspectives on teaching and learning thinking skills* (pp. 1 - 8). Basel: Karger.

Kulik, J.A., & Kulik, C.C. (1987). Review of recent research literature on computer-based teaching. *Contemporary Educational Psychology*, *12*, 222-230.

Kulik, J.A., Kulik, C.C., & Bangert-Downes, R.L. (1985). Effectiveness of computer-based education in elementary schools. *Computer in Human Behaviour*, *1*, 59-74.

Kull, J.A. (1986, April). *A Brunerian approach to teaching and learning Logo.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Kurland, D.M., & Pea, R.D. (1983). *Children's mental models of recursive Logo programs. Technical Report No. 10.* N.Y.: Bank Street College of Education, Centre for Children & Technology.

Kurland, D.M, Pea, R.D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, *2*(4), 429-458.

Kurshan, B., & Williams, J. (1985). *The effect of the computer on problem solving skills.* (ERIC Document Reproduction Service No. ED 259 714).

Lai, K. (1990). Problem solving in a Lego-Logo learning environment: Cognitive and metacognitive outcomes. In A. McDougall & C. Dowling (Eds.) *Computers in Education* (pp. 403 - 408). Amsterdam: North-Holland.

Lai, K., & Mace, R. (1989). Is there a place for computer games in secondary schools? *Computers in New Zealand Schools*, *1*(1), 37-42.

Larivée, S., Parent, S., Dupré, S., & Michaud, N. (1988). Programming Logo, cognition and metacognition. *Canadian Journal of Special Education*, *4*(1), 49-77.

Lawler, R.W. (1980). Extending a powerful idea. *Logo memo no. 58.* Cambridge: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Lawler, R.W. (1981). The progressive construction of mind. *Cognitive Science*, *5*, 1-30.

Lawler, R.W. (1982). Designing computer-based microworlds. *Byte*, *7*(8), 138-160.

Lawler, R.W., du Boulay, B., Hughes, M., & Macleod, H. (1986). *Cognition and computers: Studies in learning.* Sussex: Ellis Horwood Limited.

Lawler, R.W., & Yazdani, M. (Eds.). (1987). *Artificial intelligence and education. Vol. 1. Learning environments and tutoring systems.* Norwood, N.J.: Ablex Publishing.

Lawson, M.J. (1991). Testing for transfer following strategy training. In G. Evans (Ed.), *Learning and teaching cognitive skills* (pp. 208 - 228). Hawthorne, Victoria: ACER.

Lawton, J, & Gerschner, V.T. (1982). A review of the literature on attitudes towards computers and computerised instruction. *Journal of Research and Development in Education*, *16*(1), 50-55.

Lee, O., & Lehrer, R. (1988). Conjectures concerning the origins of misconceptions in Logo. *Journal of Educational Computing Research*, *4*(1), 87-105.

Lehrer, R. (1989). Computer-assisted strategic instruction. In C.B. McCormick, G.E. Miller, & M. Pressley (Eds.), *Cognitive strategy research: From basic research to educational applications* (pp. 303-320). N.Y.: Springer-Verlag.

Lehrer, R., Guckenberg, T., & Lee, O. (1988). Comparative study of the cognitive consequences of inquiry-based Logo instruction. *Journal of Educational Psychology*, *80*(4), 543-553.

Lehrer, R., Harckham, L.D., Archer, P, & Pruzek, R.M. (1986). Microcomputer-based instruction in special education. *Journal of Educational Computing Research*, *2*(3), 337-355.

Lehrer, R., & Randle, L. (1987). Problem solving, metacognition and composition: The effects of interactive software for first-grade children. *Journal of Educational Computing Research*, *3*(4), 409-427.

Lehrer, R., & Smith, P.C. (1986, April). *Logo learning: Is more better?* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Leonard, R. (1991). Factors influencing the use of telecommunications by schools. In *Navigating the nineties* (pp. 326-331). Brisbane: Computer Education Group of Queensland.

Leron, U. (1985). Logo today: Vision and reality. *The Computing Teacher*, *12*(5), 26-32.

Levin, H.M., Glass, G.V., & Meister, G.R. (1987). Cost-effectiveness of computer-assisted instruction. *Evaluation Review*, *11*(1), 50-72.

Lewis, S.K., & Lawrence-Patterson, E. (1989). Locus of control of children with learning disabilities and perceived locus of control by significant others. *Journal of Learning Disabilities*, *22*(4), 255-257.

Liao, Y. (1992). Effects of computer-assisted instruction on cognitive outcomes: A meta-analysis. *Journal of Research on Computing in Education*, *24*(3), 367-380.

Lieber, J., & Semmel, M.I. (1987). The relationship between group size and performance on a microcomputer problem-solving task for learning handicapped and nonhandicapped students. *Journal of Educational Computing Research*, *3*(2), 171-187.

Lieberman, D.A., & Linn, M.C. (1991). Learning to learn revisited: Computers and the development of self-directed learning skills. *Journal of Research on Computing in Education*, *23*(2), 373-395.

Linn, M.C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher*, *14*(5), 14-16, 25-29.

Linn, M.C. (1988). [Review of *Mirror of minds: Patterns of experience in educational computing*]. *Journal of Educational Computing Research*, *4*(3), 349-358.

Linn, M.C., Rohwer, W.D., & Thomas, J. (1986). *Annual Report: Autonomous classroom computer environments for learning (ACCEL).* Berkeley, CA: Lawrence Hall of Science, University of California.

Linn, M.C., Sloane, K.D., & Clancy, M.J. (1986, April). *Ideal and actual outcomes from precollege Pascal instruction.* Paper presented at the annual meetings of the American Educational Research Association, San Francisco.

Lipman, M. (1985). Thinking skills fostered by philosophy for children. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 83-108). N.J.: Lawrence Erlbaum.

Lockard, J., Abrams, P.D., & Many, W.A. (1990). *Microcomputers for educators, 2nd Edition.* Glenview, Illinois: Scott, Foresman/Little, Brown Higher Education.

Lockhead, J. (1985). Teaching analytic reasoning skills through pair problem solving. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 109-131). N.J.: Lawrence Erlbaum.

Lockheed, M.E., & Mandinach, E.B. (1986). Trends in educational computing: Decreasing interest and the changing focus of instruction. *Educational Researcher, 15*(5), 21-26.

Longeot, F. (1974). *L'échelle de Développement de la Pensée Logique.* Manuel d'instructions, Issy-les-Moulineaux: Editions Scientifiques et Psychométriques.

Loper, A.B. (1980). Metacognitive development: Implications for cognitive training. *Exceptional Education Quarterly, 1*(1), 1-8.

Lough, T., & Tipps, S. (1983). Is there Logo after turtle graphics? *Classroom Computer News, 3*(5), 50-53.

Luehrmann, A. (1981). Computer literacy: what should it be: *Mathematics Teacher, 74*(9),

Luger, G.F. (1976). The use of the state space to record the behavioural effects of subproblems and symmetries in the Tower of Hanoi problem. *International Journal of Man-Machine Studies, 8*(4), 411-421.

Luger, G.F., & Bauer, M.A. (1978). Transfer effects in isomorphic problem solving situations. *Acta Psychologica, 42*, 121-131.

Luger, G.F., & Steen, M. (1981). Using the state space to record the behavioural effects of symmetry in the Tower of Hanoi problem and an isomorph. *International Journal of Man-Machine Studies, 14*(4), 449-460.

MacGregor, S.K., Shapiro, J.Z., & Niemic, R. (1988). Effects of a computer-augmented learning environment on math achievement for students with differing cognitive style. *Journal of Educational Computing Research, 4*(4), 453-465.

Maddison, J. (1983). *Education in the microelectronics era: A comprehensive approach.* Milton Keynes: Open University Press.

Maddux, C.D. (Ed.). (1985). *Logo in the schools.* N.Y.: Haworth Press.

Maddux, C.D., & Johnson, D.L. (1988). *Logo: Methods and curriculum for teachers.* N.Y.: The Haworth Press.

Maltzman, I. (1955). Thinking: From a behavioristic point of view. *Psychological Review, 25*, 275-286.

Mandell, C.J., & Mandell, S.L. (1989). *Computers in education today.* N.Y.: West Publishing Company.

Mandinach, E.B. (1986, April). *Aspects of programming courses that foster problem solving.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Mandinach, E.B., & Linn, M.C. (1986). The cognitive effects of computer learning environments. *Journal of Educational Computing Research, 2*(4), 411-427.

Mandinach, E.B., & Linn, M.C. (1987). Cognitive consequences of programming: Achievements of experienced and talented programmers. *Journal of Educational Computing Research, 3*(1), 53-72.

Many, W.A., Lockard, J., & Abrams, P.D. (1988). The effect of learning to program in Logo on reasoning skills of junior high school students. *Journal of Educational Computing Research, 4*(2), 203-213.

Marshall, R. (1987). Computers in education: Confidence and control. *Bits & Bytes, 5*(6), 63-65.

Martin, A. (1985). *Teaching and learning with Logo.* N.Y.: Teachers College Press, Teachers College, Columbia University.

Martin, K., & Riordon, T. (1983). Logo - beginning a new school year. *The Computing Teacher, 11*(1), 42-44

Mathinos, D.A. (1990). Logo programming and the refinement of problem-solving skills in disabled and nondisabled children. *Journal of Educational Computing Research, 6*(4), 429-446.

Maxwell, B. (1984). Why Logo? In A. Kelly (Ed.), *Microcomputers and the curriculum* (pp. 84-106). N.Y.: Harper & Row.

Mayer, R.E. (1983). *Thinking, problem solving, cognition.* N.Y.: W.H. Freeman and Company.

Mayer, R.E. (1984, April). *Learnable aspects of problem solving: Some examples.* Paper presented at the annual meeting of the American Educational Research Association, New Orleans, L.A.

Mayer, R.E., Dyck, J.L., & Vilberg, W. (1989). Learning to program and learning to think: What's the connection? In E. Soloway & J.C. Spohrer (Eds.), *Studying the novice programmer* (pp. 113-124). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Mayer, R.E., & Fay, A.L. (1987). A chain of cognitive changes with learning to program in Logo. *Journal of Educational Psychology*, *79*(3), 269-279.

McAllister, A. (1985). *Problem solving at the threshold of computer programming (Bulletin No. 13).* Toronto: Toronto Board of Education.

McClurg, P.A., & Chaillé, C. (1987). Computer games: Environments for developing spatial cognition? *Journal of Educational Computing Research*, *3*(1), 95-111.

McCormick, C.B., Miller, G.E., & Pressley, M. (Eds.) (1989). *Cognitive strategy research: From basic research to educational applications.* N.Y.: Springer-Verlag.

McDonald, R.P. (1988). The first and second laws of intelligence. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 78-85). Hawthorn, Victoria: Australian Council of Educational Research.

McDougall, A. (1983). Logo for senior secondary school students. In A.D. Salvas (Ed.), *Could you use a computer* (pp. 50-54). Balaclava: Computer Education Group of Victoria.

McDougall, A. (1985). Logo: its use in education and research. In B. Rasmussen (Ed.), *The information edge: The future for educational computing* (pp.141-150). Brisbane: Computer Education Group of Queensland.

McDougall, A. (1986). Children's difficulties in perceiving structure and using subprocedures in Logo. In A.D. Salvas & C. Dowling (Eds.). *On the crest of a wave?* (pp. 53-55). Balaclava, Victoria: Computer Education Group of Victoria.

McDougall, A. (1988). *Children, recursion and Logo programming.* Unpublished Ph.D. thesis. Melbourne: Monash University.

McDougall, A., & Adams, T. (1982). Logo environments: The development of the language and its use in education and research. In A. Sale & G. Hawthorne, G. (Eds.), *Proceedings of the ninth Australian computer conference* (pp. 116-132. Sydney: Australian Computer Society.

McDougall, A., & Adams, T. (1983). Children can teach computers to write poetry! *Australian Journal of Reading*, *6*(4), 222-228.

McDougall, A., Adams, T., & Adams, P. (1982). *Learning Logo on the Apple II.* Sydney: Prentice-Hall.

McGee, G.W. (1987). Social context variables affecting the implementation of microcomputers. *Journal of Educational Computing Research*, *3*(2), 189-206.

McGrath, D. (1988). Programming and problem solving: Will two languages do it? *Journal of Educational Computing Research*, *4*(4), 467-484.

McMillan, B.W. (1987). Logo and the teaching of Logo: A Piagetian perspective. In J. Hancock (Ed.), *Tomorrow's technology today* (pp. 179-186). Magill, S.A.: Computers in Education Group of South Australia.

Mehan, H. (1989). Microcomputers in classrooms: Educational technology or social practice? *Anthropology and Education Quarterly*, *20*, 4-22.

Mendelsohn, P. (1984). L'analyse psychologique des activités de programmation chez l'enfant. *Compte-rendu d'une recherche exploratoire sur la compréhension et la production de programmes en Logo chez des enfants de 10-12 ans.* Grenoble: Laboraoire de Psychologie Expérimentale.

Mendelsohn, P. (1985). L'analyse psychologique des activités de programmation chez l'enfant de CM-1 et CM-2. *Enfance, 2-3*, 313-221.

Mendelsohn, P. (1987). L'apprentissage des concepts informatiques: du déplacement de la tortue Logo à la coordination d'objets graphiques. *Revue Canadienne de Psycho-éducation, 16*(2), 97-118.

Merrill, P.F., Tolman, M.N., Christensen, L, Hammons, K., Vincent, B.R., & Reynolds, P.L. (1986). *Computers in education.* N.J.: Prentice-Hall.

Messick, S., & Sigel, I. (1982). Conceptual and methodological issues in facilitating growth in intelligence. In D.K. Detterman and R.J. Sternberg (Eds.), *How and how much can intelligence be increased* (pp. 187-195). N.J.: Ablex Publishing Corporation.

Michayluk, J.O. (1986). Logo: more than a decade later. *British Journal of Educational Technology, 17*(1), 35-41.

Michayluk, J.O., & Saklofske, D.H. (1985). Some effects of Logo with emotionally disturbed children. *Canadian Journal of Educational Communication, 14*(4), 4-7.

Michayluk, J.O., & Saklofske, D.H. (1988). Logo and Special Education. *Canadian Journal of Special Education, 4*(1), 43-48.

Miles, M.B., & Huberman, A.M. (1984). Drawing valid meaning from qualitative data: toward a shared craft. *Educational Researcher, 13*(5), 20-30.

Miller, G.A., Galanter, E., & Pribram, K.H. (1960). *Plans and the structure of behaviour.* N.Y.: Holt, Rinehart & Winston.

Miller, G.E., & Emihovich, C. (1986). The effects of mediated programming instruction on preschool children's self-monitoring. *Journal of Educational Computing Research, 2*(3), 283-297.

Miller, G.E., Emihovich, C., Clare, V., & Froning, D. (1986, April). *The effects of interactive programming on preschool children's self-monitoring.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Milner, S.D. (1973). *The effects of teaching computer programming on performance in mathematics.* Unpublished doctoral dissertation. Pittsburgh: University of Pittsburgh.

Minnesota Educational Computer Consortium. (1983). *Apple Logo in the classroom.* St. Paul, Minnesota: Minnesota Educational Computer Consortium.

Minsky, M., & Papert, S. (1972). Artificial intelligence. *A.I. memo No. 252.* Cambridge: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Mitterer, J., & Rose-Krasnor, L. (1986). Logo and transfer of problem solving: an empirical test. *The Alberta Journal of Educational Research, XXXII*(3), 176-194.

Moore, P.J. (1982). Children's metacognitive knowledge about reading: A selected review. *Educational Research, 24*(2), 120-128.

Moray, N. (1978). The strategic control of information processing. In G. Underwood (Ed.), *Strategies of information processing* (pp. 301-328). London: Academic Press.

Moursand, D. (1988). Problem solving. *The Computing Teacher, 16*(4), 5, 56.

Nastasi, B.K., Clements, D.H., & Battista, M.T. (1990). Social-cognitive interactions, motivation, and cognitive growth in Logo programming and CAI problem-solving environments. *Journal of Educational Psychology*, *82*(1), 150-158.

Neilsson, N.J. (1971). *Problem-solving methods in artificial intelligence*. N.Y.: McGraw-Hill.

New South Wales Department of Education (1986). *Using computers for problem solving in mathematics*. Erskinville, N.S.W.: NSW Department of Education.

Newell, A. (1980). One final word. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 175-189). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Newell, A., Shaw, J.C., & Simon, H.A. (1960). Report on a general problem-solving program. In W.R. Reitman (Ed.), *Information processing: Proceedings of the international conference on information processing* (pp. 256-264). Paris: UNESCO.

Newell, A., & Simon, H.A. (1972). *Human problem solving*. Englewood Cliffs, N.J.: Prentice Hall.

Newman, D. (1987). Functional environment for microcomputers in education. In R.D. Pea & K. Sheingold (Eds.) *Mirrors of minds: Patterns of experience in educational computing* (pp. 57-66). Norwood, N.J.: Ablex Publishing Corporation.

Nickerson, R.S. (1982). Computer programming as a vehicle for teaching thinking skills. *Thinking: The Journal of Philosophy for Children*, *4*, 42-48. Cited in Kurland, Pea, Clement & Mawby, 1986.

Nickerson, R.S. (1988-89). On improving thinking through instruction. *Review of Research in Education*. *15*, 3-57.

Nielsen, J. (1986). Not the computer but human interaction is the basis for cognitive development and education. *Education and Computing*, *2*, 53-61.

Niemiec, R.P., & Walberg, H.J. (1985). Computers and achievement in the elementary schools. *Journal of Educational Computing Research*, *1*(4), 435-440.

Niemiec, R.P., & Walberg, H.J. (1987). Comparative effects of computer-assisted instruction: A synthesis of reviews. *Journal of Educational Computing Research*, *3*(1), 19-37.

Niess, M.L. (1992). Seeing stars in a mathematical microworld. *The Computing Teacher*, *20*(2), 36-39.

Nisbet, J., & Shucksmith, J. (1986). *Learning strategies*. London: Routledge & Kegan Paul.

Nolan, P., & Ryba, K. (1984). The microcomputers as a learning experience. *New Zealand Journal of Educational Studies*, *19*(1), 24-33.

Nolan, C.J.P., & Ryba, K. (1986). *Assessing learning with Logo*. Eugene: International Council of Computers in Education.

Norman, D.A. (1978). Notes toward a theory of complex learning. In A.M. Lesgold, J.W. Pellegrino, S.D. Fokkema, & R. Glaser (Eds.), *Cognitive psychology and instruction*. N.Y.: Plenum.

Norris, C., Jackson, L., & Poirot, J. (1992). The effect of computer science instruction on critical thinking skills and mental alertness. *Journal of Research on Computing in Education*, *24*(3), 329-337.

Norris, C.A., & Poirot, J.L. (1991). Problem solving, critical thinking, and computing: An overview. In C.A. Norris & J.L. Poirot (Eds.), *Problem solving and critical thinking for computer science educators* (pp. 1-7). Eugene, Oregon: International Society for Technology in Education.

Noss, R. (1987a). Children's learning of geometrical concepts through Logo. *Journal for Research in Mathematics Education*, *18*(5), 343-362.

Noss, R. (1987b). How do children do mathematics with Logo? *Journal of Computer Assisted Learning*, *3*, 2-12.

Okagaki, L. & Sternberg, R.J. (1990). In D. Kuhn (Ed.), *Developmental perspectives on teaching and learning thinking skills* (pp. 63 - 78). Basel: Karger.

Oliver, R. (1986). *Using Computers in Schools.* Western Australia: Heron Computing.

Olson, D.R. (1985). Computers as tools of the intellect. *Educational Researcher, 14*(5), 5-8.

Olson, D.R. (1987). Mind and the technologies of communication. In J. Hattie, R. Kefford, & P. Porter (Eds.), *Skills, technology and management in education* (pp. 83-90). Deakin, A.C.T.: The Australian College of Education.

Olsen, J.K. (1985). *Using Logo to supplement the teaching of geometric concepts in the elementary school classroom.* Unpublished doctoral dissertation. Oklahoma: Oklahoma State University.

Ortiz, E. & MacGregor, S.K. (1991). Effects of Logo programming on understanding variables. *Journal of Educational Computing Research, 7*(1), 37-50.

O'Shea, T., & Self, J. (1983). *Learning and teaching with computers: Artifical intelligence in education.* Brighton: Harvester.

Palincsar, A.S. (1986). Metacognitive strategy instruction. *Exceptional Children, 53*(2), 118-124.

Palincsar, A.S. (1986, April). *The unpacking of a multi-component, metacognitive training package.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Palincsar, A.S., & Brown, A.L. (1984). Reciprocal teaching of comprehension monitoring activities. *Cognition and Instruction, 1*, 117-175.

Palumbo, D.B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research, 60*(1), 65-89.

Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. *International Journal of Mathematical Education in Science and Technology, 3*, 249-262.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Sussex: Harvester Press Ltd.

Papert, S. (1984). Tomorrow's classroom. In M. Yazdani (Ed.), *New horizons in educational computing* (pp. 17-20). Chichester: Ellis Horwood.

Papert, S. (1986). Are computers bad for children? In P.F. Campbell & G.G. Fein (Eds.), *Young children and microcomputers* (pp. 171-180). Reston, V.A.: Reston Books. Cited in Silvern & Williamson (1986). An extremely useful quote concerning the learning without curriculum. A must.

Papert, S. (1987). Computer criticism vs technocentric thinking. *Educational Researcher, 16*(1), 22-30.

Papert, S, Watt, D., diSessa, A., & Weir, S. (1979). *Final report of the Brookline Logo Project. Part II: project summary and data analysis.* Massachusetts: Massachusetts Institute of Technology.

Paris, S.G., Newman, R.S., & McVey, K.A. (1983). Learning the functional significance of mnemonic actions: A microgenetic study of strategy acquisition. *Journal of Experimental Schild Psychology, 34*(3), 490-509.

Paris, S.G., & Oka, E.R. (1986). Self-regulated learning among exceptional children. *Exceptional Children, 53*(2), 103-108.

Paris, S.G., & Winograd, P. (1990a). Promoting metacognition and motivation of exceptional children. *Remedial and Special Education, 11*(6), 7-15.

Paris, S.G., & Winograd, P.W. (1990b). How metacognition can promote academic learning and instruction. In B.J. Jones & L. Idol (Eds.), *Dimensions of thinking and cognitive instruction* (pp. 15-51). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Pea, R.D. (1983). *Logo programming and problem solving. Technical report No. 12.* N.Y.: Bank Street College of Education, Center for Children and Technology.

Pea, R.D. (1986). Language-independent conceptual "bugs" in novice programming. *Journal of Educational Computing Research*, *2*(1), 25-36.

Pea, R.D. (1987). Integrating human and computer intelligence. In R.D. Pea & K. Sheingold (Eds.), *Mirrors of minds: Patterns of experience in educational computing* (pp. 128-146). Norwood, N.J.: Ablex Publishing Corporation.

Pea, R.D., & Hawkins, J. (1984). *Children's planning processes in a chore-scheduling task (Technical report No. 11).* N.Y.: Bank Street College of Education.

Pea, R.D., & Kurland, D.M. (1984a). *On the cognitive effects of computer programming: A critical look (Technical report No. 9).* N.Y.: Bank Street College of Education.

Pea, R.D., & Kurland, D.M. (1984b). *Logo programming and the development of planning skills (Technical report No. 16).* N.Y.: Bank Street College of Education.

Pea, R.D., & Kurland, D.M. (1984c). *On the cognitive prerequisites of learning computer programming (Technical report No. 18).* N.Y.: Bank Street College of Education.

Pea, R.D., & Kurland, D.M. (1984d). On the cognitive effects of learning computer programming. *New Ideas Psychology*, *2*(2), 137-168.

Pea, R.D., & Kurland, D.M. (1987). On the cognitive effects of learning computer programming. In R.D. Pea & K. Sheingold (Eds.) *Mirrors of minds: Patterns of experience in educational computing* (pp. 147-177). Norwood, N.J.: Ablex Publishing Corporation.

Pea, R.D., & Sheingold, K. (Eds.). (1987). *Mirrors of minds: Patterns of experience in educational computing.* Norwood, N.J.: Ablex Publishing Corporation.

Pea, R.D., Kurland, D.M., & Hawkins, J. (1987). Logo and the development of thinking skills. In R.D. Pea & K. Sheingold (Eds.) *Mirrors of minds: Patterns of experience in educational computing* (pp. 178-197). Norwood, N.J.: Ablex Publishing Corporation.

Perkins, D.N. (1985). The fingertip effect: How information-processing technology shapes thinking. *Educational Researcher*, *14*(7), 11-17.

Perkins, D.N., & Salomon, G. (1989). Are cognitive skills context-bound? *Educational Researcher*, *18*(1), 16-25.

Perkins, D.N., & Simmons, R. (1988). Patterns of misunderstandings: An integrative model for science, math, and programming. *Review of Educational Research*, *58*(3), 303-326.

Peterson, P.L., Swing, S.R., Braverman, M.T., & Buss, R. (1982). Students' aptitudes and their reports of cognitive processes during direct instruction. *Journal of Educational Psychology*, *74*(4), 535-547.

Phillips, D.C. (1981). Toward an evaluation of the experiment in educational contexts. *Educational Researcher*, *10*(6), 13-20.

Phillips, D.C. (1983). After the wake: Postpositivistic educational thought. *Educational Researcher*, *12*(5), 4-12.

Piaget, J. (1963). The child's conception of the world. Paterson, N.J.: Littlefield, Adams.

Piaget, J. (1976). *The grasp of consciousness.* Cambridge, Massachusetts: Harvard University Press.

Piaget, J. (1977). *The development of thought: Equilibrium of cognitive structures.* N.Y.: Viking.

Piedmont, C. (1983). Computer currents: Logo power grows. *Curriculum Review*, *12*, 36.

Polson, P.G., & Jeffries, R. (1985). Instruction in general problem-solving skills: An analysis of four approaches. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 417-455). N.J.: Lawrence Erlbaum.

Polya, A. (1957). *How to solve it: A new aspect of mathematical method.* N.Y.: Doubleday-Anchor.

Pressley, M. (1986). The relevance of the good strategy user model to the teaching of mathematics. *Educational Psychologist, 21,* 139-161.

Pressley, M., Borkowski, J.G., & O'Sullivan, J.O. (1985). Children's metamemory and the teaching of memory strategies. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller (Eds.), *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives* (pp. 111-153). Orlando, Florida: Academic Press.

Preston, G.D., & Au, W.K. (1989). Gender differences in computer access of secondary school students. In D. Sutton (Ed.), *Hands on* (pp. 219-231). Cheltenham: N.S.W. Computer Education Group Ltd.

Probert, P. (1985). *Computer assisted instruction.* Unpublished Bachelor of Educational Studies Dissertation. University of Newcastle, Newcastle, Australia.

Putnam, R.T., Sleeman, D, Baxter, J.A., & Kuspa, L.K. (1986). A summary of misconceptions of high school Basic programmers. *Journal of Educational Computing Research, 2*(4), 459-472.

Rankin, R.J., & Trapper, T. (1978). Retention and delay of feedback in a computer-assisted instructional task. *Journal of Experimental Education, 64*(4), 67-70.

Raven, J.C. (1956). *Guide to using progressive matrices (1938).* London: H.K. Lewis & Co.

Raven, J.C., & Court, J.H., & Raven, J. (1984). *Manual for Raven's progressive matrices and vocabulary scales.* London: H.K. Lewis & Co.

Ray, W.S. (1955). Complext tasks for use in human problem-solving research. *Psychological Bulletin, 52,* 134-149.

Reed, K. (1982). An interview with Wallace Feurzeig. *COM-3, 29,* 9.

Reeve, R.A., & Brown, A.L. (1985). Metacognition reconsidered: Implications for intervention research. *Journal of Abnormal Child Psychology, 13*(3), 343-356.

Reid, N.A., & Hughes, D.C. (1974). *Progressive Achievement Tests, Teacher's Manual: Mathematics.* Wellington, N.Z.: New Zealand Council for Educational Research.

Reif, F. (1980). Theoretical and educational concerns with problem solving: Bridging the gaps with human cognitive engineering. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues In teaching and research* (pp. 39-49). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Renwick, W.L. (1985, January). *Teachers, learning, and information technology.* Opening Address to the New Zealand Educational Administration Society, Dunedin, New Zealand.

Resnick, L.B. (1981). Instructional psychology. *Annual Review of Psychology, 32,* 659-704.

Resnick, L.B. (1990). Instruction and the cultivation of thinking. In N. Entwistle (Ed.), *Handbook of educational ideas and practices* (pp. 694-707). London: Routledge.

Resnick, L.B., & Glaser, R. (1976). Problem solving and intelligence. In L.B. Resnick (Ed.), *The nature of intelligence* (pp. 205-230). N.J.: Lawrence Erlbaum Associates.

Resnick, M. (1990). *Overcoming the centralized mindset: Towards an understanding of emergent phenomena. E&L Memo No. 11.* Massachusetts: Media Laboratory, Massachusetts Institute of Technology.

Reiber, L. (1983). *The effect of Logo on increasing systematic and procedural thinking according to Piaget's theory of intellectual development and on its ability to teach geometric concepts to young children.* (Report

No. IR 011 583). Pennsylvania, PA: The Pennsylvania State University. (ERIC Document Reproduction Service No. ED 256 288).

Riordon, T. (1982). Creating a Logo environment. *The Computing Teacher*, *10*(3), 46-50.

Roblyer, M.D., Castine, W.H., & King, F.J. (1988). *Assessing the impact of computer-based instruction: A review of recent research*. N.Y.: The Haworth Press.

Rohwer Jr, W.D., & Thomas, J.W. (1989). Domain-specific knowledge, metacognition, and the promise of instructional reform. In C.B. McCormick, G.E. Miller, & M. Pressley (Eds.), *Cognitive strategy research: From basic research to educational applications* (pp. 104-132). N.Y.: Springer-Verlag.

Ross, P. (1983). *Logo programming*. London: Addison-Wesley Publishing Company.

Ross, P., & Howe, J.A.M. (1981). Teaching mathematics through programming: Ten years on. In R. Lewis, & D. Tagg (Eds.) *Computers in education* (pp. 143-148). Amsterdam: North-Holland.

Rousseau, J.F., & Smith, S.M. (1981). Whither goes the turtle. *Microcomputing*, September, 1981, 52-54.

Rowe, H.A.H. (1985). *Problem solving and intelligence*. Hillsdale, N.J.: Lawrence Erlbaum.

Rowe, H.A.H. (1988a). A promising step: From psychometric to process measures of intelligence, competency, and educational achievement. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 33-43). Hawthorn, Victoria: Australian Council of Educational Research.

Rowe, H.A.H. (1988b). Cognitive strategies and time-on task: An information-processing approach to the study of time-on-task and achievement. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 152-178). Hawthorn, Victoria: Australian Council of Educational Research.

Rowe, H.A.H. (1991). Learning with computers: what type of research? In *Navigating the nineties* (pp. 130-141). Brisbane: Computer Education Group of Queensland.

Rubinstein, M.F. (1980). A decade of experience in teaching an interdisciplinary problem-solving course. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 25-38). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Russell, S.J. (1983). Logo in special education. *Classroom Computer Learning*, *4*(3), 34-39.

Ryba, K. (1987, July). *Educational computing cultures: Hit or myth*. Paper presented to the Otago Institute of Educational Research, Dunedin.

Ryba, K., & Anderson, B. (1987). Teaching metacognitive strategies with computer games: A case study. *Delta*, *39*, 41-48.

Ryba, K., & Anderson, B. (1990). *Learning with computers: effective teaching strategies*. Eugene, Oregon: International Society for Technology in Education.

Ryba, K., Anderson, B., & Chapman, J.W., (1986, November). *Metacognitive strategy training: The effects of maze and adventure games programs on the development of children's thinking skills*. Paper presented at the annual meeting of the New Zealand Association for Research in Education, Hamilton, New Zealand.

Ryba, K. & Mackrell, T. (1986). *International telecommunications education project*. Unpublished research proposal, Department of Education, Massey University, Palmerston North, New Zealand.

Ryba, K.A. (1980). *An evaluation of microcomputer assisted instruction for teaching word recognition to mentally retarded adults*. Unpublished doctoral dissertation. Massey University, Palmerston North, New Zealand.

Salomon, D.N., & Simmons, R. (1988). Patterns of misunderstanding: An integrative model for science, math, and programming. *Review of Educational Research*, *58*(3), 303-326.

Salomon, G. (1984). On ability development and far transfer: a response to Pea and Kurland. *New Ideas Psychology*, *2*(2), 169-174.

Salomon, G. (1988). AI in reverse: Computer tools that turn cognitive. *Journal of Educational Computing Research*, *4*(2), 123-139.

Salomon, G., & Gardner, H. (1986). The computer as educator: lessons from television research. *Educational Researcher*, *15*(1), 13-19.

Salomon, G., & Perkins, D.N. (1987). Transfer of cognitive skills from programming: When and how? *Journal of Educational Computing Research*, *3*(2), 149-169.

Salomon, G., Perkins, D.N., & Globerson, T. (1991). Partners in cognition: Extending human intelligence with intelligent technologies. *Educational Researcher*, *20*(3), 2-9.

Savell, J.M., Twohig, P.T., & Rachford, D.L. (1986). Empirical status of Feuerstein's "instrumental enrichment" (FIE) as a method of teaching thinking skills. *Review of Educational Research*, *56*(4), 381-409.

Scandura, J.M. (1977). *Problem solving: A structural/process approach with instructional implications*. N.Y.: Academic Press.

Scardamalia, M., Bereiter, C., McLean, R.S., Swallow, J., & Woodruff, E. (1989). Computer-supported intentional learning environments. *Journal of Educational Computing Research*, *5*(1), 51-68.

Scott, T., Cole, M., & Engel, M. (1992). Computers and education: A cultural constructivist perspective. *Review of Research in Education*, *18*, 191-251.

Schaefer, L., & Sprigle, J.E. (1988). Gender differences in the use of the Logo programming language. *Journal of Educational Computing Research*, *4*(1), 49-55.

Schauble, L. (1984). The feasibility of a developmental cognitive science: a response to Pea and Kurland. *New Ideas Psychology*, *2*(2), 181-183.

Schibeci, R.A. (1990). Logo in pre-service and in-service teacher education. *Computer Education*, *14*(1), 53-60.

Schneider, W. (1985). Developmental trends in the metamemory-memory behaviour relationship: An integrative review. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller (Eds.), *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives* (pp. 57-109). Orlando, Florida: Academic Press.

Scriven, M. (1980). Prescriptive and descriptive approaches to problem solving. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 127-147). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Seidman, R.H. (1981, April). *The effects of learning a computer programming language on the logical reasoning of school children*. Paper presented at the annual meeting of the American Educational Research Association, Los Angeles, CA.

Seidman, R.H. (1989-90). Computer programming and logical reasoning: unintended cognitive effects. *Journal of Eudcational Technology Systems*, *18*(2), 123-141.

Self, J. (Ed.). (1988). *Artificial intelligence and human learning*. London: Chapman and Hall.

Sharp, P. (1984a). *Turtle power activity book*. Boca Raton, Florida: International Business Machines Corporation.

Sharp, P. (1984b). *Turtle power thinker's guide*. Boca Raton, Florida: International Business Machines Corporation.

Shavelson, R.J., & Salomon, G. (1985). Information technology: Tool and teacher of the mind. *Educational Researcher*, *14*(5), 4.

Sheinfold, K., Martin, L.M.W., & Endreweit, M.E. (1987). Preparing urban teachers for the technological future. In R.D. Pea & K. Sheingold (Eds.) *Mirrors of minds: Patterns of experience in educational computing* (pp. 67-85). Norwood, N.J.: Ablex Publishing Corporation.

Sheingold, K. (1987). The microcomputer as a symbolic medium. In R.D. Pea & K. Sheingold (Eds.) *Mirrors of minds: Patterns of experience in educational computing* (pp. 198-208). Norwood, N.J.: Ablex Publishing Corporation.

Sherwood, C. (1991). Classroom 2000: Challenges and changes. In *Navigating the nineties* (pp. 312-318). Brisbane: Computer Education Group of Queensland.

Shrock, S., Matthias, M., Vensel, C., & Anastasoff, J. (1985, April). *Microcomputers and peer interaction: A naturalistic study of an elementary classroom.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Shulman, L. (1981). Disciplines of inquiry in education: An overview. *Educational Researcher, 10*(6), 5-12.

Shute, V.J. (1991). Who is likely to acquire programming skills. *Journal of Educational Computing Research, 7*(1), 1-24.

Siegler, R.S. (1983). Information processing approaches to development. In P.H. Mussen (Ed.), *Handbook of child psychology Vol 1. History, theory and methods* (4th Edition) (pp.129-211). N.Y.: John Wiley & Sons.

Silvern, S.B., & Williamson, P.A. (1986, April). *A constructivist perspective for Logo curriculum.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Simon, H.A. (1976). Identifying basic abilities underlying intelligent performance of complex tasks. In J. Resnick (Ed), *The nature of intelligence* (pp. 65-97). N.Y.: Harper and Row.

Simon, H.A. (1979). The functional equivalence of problem solving skills. *Models of thought* (pp. 230-244). New Haven: Yale University Press.

Simon, H.A. (1980). Problem solving in education. In D.T. Tuma & F. Reif (Eds.), *Problem solving and education: Issues in teaching and research* (pp. 81-107). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Simon, H.A., & Hayes, J.R. (1976). The understanding process: Problem isomorphs. *Cognitive Psychology, 8*, 165-190.

Skinner, B.F. (1966). An operant analysis of problem solving. In B. Kleinmuntz (Ed.), *Problem solving: Research, method, and theory* (pp. 225-257). N.Y.: John Wiley & Sons.

Smilansky, J. (1984). Problem solving and the quality of intervention: an empirical investigation. *Journal of Educational Psychology, 76*(3), 377-386.

Smith, C.D. (1986). Learning Logo: effects on learning BASIC and statistics. *Journal of Computer Assisted Learning*. 2, 102-109.

Smith, J.K., & Heshusius, L. (1986). Closing down the conversation: The end of the quantitative-qualitative debate among educational inquiries. *Educational Researcher, 15*(1), 4-12.

Smyrk, J.R. (1991). Artificial neural networks: Today's promise - tomorrow's too. In *Navigating the nineties* (pp. 8-18). Brisbane: Computer Education Group of Queensland.

Solomon, C. (1978). Teaching young children to program in a Logo culture. *Sigcue Bulletin, 12*(3), 20-29.

Solomon, C. (1982). Introducing Logo to children. *Byte, 7*(8), 196-208.

Soloway, E. (1985). From problems to programs via plans: The content and structure of knowledge for introductory Lisp programming. *Journal of Educational Computing Research, 1*(2), 157-172.

Soloway, E., Lockhead, J, & Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R.J. Seidel, R.E., Anderson, & B. Hunter (Eds.), *Computer literacy: Issues and directions for 1985* (pp. 171-185). N.Y.: Academic Press.

Solso, R.L. (1988). *Cognitive psychology, 2nd Edition.* Boston: Allyn and Bacon.

Soulier, J.S. (1988). *The design and development of computer based instruction.* Boston: Allyn and Bacon.

SPSS Inc. (1988). *SPSSX user's guide, 3rd Edition.* N.Y.: McGraw-Hill Book Company.

Stallings, J.A. (1983, November). *The stallings observation system.* Unpublished manuscript.

Steinberg, E.R. (1980). Evaluation processes in young children's problem solving. *Contemporary Educational Psychology, 5,* 276-281.

Steinberg, E.R., Baskin, A.B., & Hoffer, E. (1986). Organizational/Memory tools: A technique for improving problem solving skills. *Journal of Educational Computing Research, 2*(2), 169-187.

Sternberg, R.J. (1981). Intelligence and nonentrenchment. *Journal of Educational Psychology, 73,* 1-16.

Sternberg, R.J. (1982a). Reasoning, problem solving, and intelligence. In R.J. Sternberg (Ed.), *Handbook of human intelligence* (pp. 225-307). Cambridge: Cambridge University Press.

Sternberg, R.J. (1982b). Introduction: Some common themes in contemporary approaches to the training of intelligent performance. In D.K. Detterman and R.J. Sternberg (Eds.), *How and how much can intelligence be increased* (pp. 141-146). N.J.: Ablex Publishing Corporation.

Sternberg, R.J. (1983). Criteria for intellectual skills training. *Educational Researcher, 12*(2), 6-12.

Sternberg, R.J. (1984). What should intelligence tests test? Implications of a triarchic theory of intelligence for intelligence testing. *Educational Researcher, 13*)1), 5-15.

Sternberg, R.J. (1985a). *Beyond IQ.* Cambridge, M.A.: Cambridge University Press.

Sternberg, R.J. (1985b). Instrumental and componential approaches to the nature and training of intelligence. In S.F. Chipman, J.W. Segal, & R. Glaser (Eds.), *Thinking and learning skills: Vol 2. Research and open questions* (pp. 215-243). N.J.: Lawrence Erlbaum.

Stevenson, J.C. (1991). Cognitive structures for the teaching of adaptability in vocational education. In G. Evans (Ed.), *Learning and teaching cognitive skills* (pp. 144 - 163). Hawthorne, Victoria: ACER.

Stone, C.A. (1989). Improving the effectiveness of strategy training for learning disabled students: The role of communicational dynamics. *Remedial and Special Education, 10*(1), 35-42.

Sullivan, E.V. (1984). On the cognitive and educational benefits of teaching children programming: a response to Pea and Kurland. *New Ideas Psychology, 2*(2), 175-179.

Swan, K. (1991). Programming objects to think with: Logo and the teaching and learning of problem solving. *Journal of Educational Computing Research, 7*(1), 89-112.

Symons, S., Snyder, B.L., Cariglia-Bull, T., & Pressley, M. (1989). Why be optimistic about cognitive strategy instruction? In C.B. McCormick, G.E. Miller & M. Pressley (Eds.) *Cognitive strategy research: From basic research to educational applications* (pp. 3-32). N.Y.: Springer-Verlag.

Taylor, R. (1980). *The computer in the school: Tutor, tool, tutee.* N.Y.: Teachers College, Columbia University.

Telfer, R., & Probert, P. (1986). For and against: The pros and cons of computer assisted instruction. *Educational News, 19*(9), 24-27.

Tetenbaum, T.J., & Mulkeen, T.A. (1984). Logo and the teaching of problem solving: A call for a moratorium. *Educational Technology*, November, 1984, 16-19.

Thomas, J.W. (1986, April). *Aspects of high school programming courses that foster autonomous learning activities.* Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Thompson, A.D., & Chen Wang, H. (1988). Effects of a Logo microworld on student ability to transfer a concept. *Journal of Educational Computing Research*, *4*(3), 335-347.

Thornburg, D.D. (1991). *Education, technology, and paradigms of change for the 21st century.* U.S.A.: Starsong Publications.

Tobias, S. (1982). When do instructional methods make a difference? *Educational Researcher*, *11*(4), 4-9.

Tobias, S. (1985). Computer-assisted instruction. In M. Wang & H. Walberg (Eds.), *Adaptive education.* Berkeley, California: McCutchan.

Torgerson, S. (1983-84). Classroom Management for Logo. *The Computing Teacher*, *11*(5), 12-14.

Torgerson, S.R., Kriley, M.K., & Stone, J.T. (1984). *Logo in the classroom.* Eugene: International Society for Technology in Education.

Torgesen, J.K. (1986). Computers and cognition in reading: A focus on decoding fluency. *Exceptional Children*, *53*(2), 157-162.

Torrance, E.P. (1966). *Torrance tests of creative thinking: Norms-technical manual.* N.J.: Personnel Press, Inc.

Torrance, E.P. (1972). *Torrance tests of creative thinking: Directions manual and scoring guide, figural test booklet A.* N.J.: Personnel Press, Inc.

Torrance, E.P. (1974). *Torrance tests of creative thinking: Directions manual and scoring guide, figural test booklet B.* Bensenville, Illinois: Scholastic Testing Service Inc.

Trow, M. (1957). Comment on "Participant observation and interviewing: a comparison". *Human Organization*, *16*(3), 33-35.

Tuma, D.T., & Reif, F. (Eds.) (1980). *Problem solving and education: Issues in teaching and research.* Hillsdale, N.J.: Erlbaum.

Turner, S.V., & Land, M.L. (1988). Cognitive effects of a Logo-enriched mathematics program for middle school students. *Journal of Educational Computing Research*, *4*(4), 443-452.

Turnure, J.E. (1987). Social influences on cognitive strategies and cognitive development: The role of communication and instruction. *Intelligence*, *11*(1), 77-89.

Underwood, B.J. (1952). An orientation for research on thinking. *Psychological Review*, *59*, 209-220.

Underwood, G. (Ed.). (1978). *Strategies of information processing.* London: Academic Press.

van de Geer, J.P. (1957). *A psychological study of problem solving.* Haarlem: Uitgeverij De Toorts.

Vinacke, W.E. (1952). *The psychology of thinking.* N.Y.: McGraw-Hill.

Vockell, E.L., & Rivers, R.H. (1984). *Computerized science simulations stimulus to generalized problem solving capabilities.* (ERIC Document Reproduction Service No. ED 253 397).

Vockell, E., & Schwartz, E. (1988). *The computer in the classroom.* CA: Mitchell Publishing Inc.

Vygotsky, L.S. (1978). *Mind in society: The development of higher psychological processes.* Cambridge, Massachusetts: Harvard University Press.

Wadsworth, B.J. (1989). *Piaget's theory of cognitive and affective development* (4th Edition). N.Y.: Longman.

Wagner, R.K., & Sternberg, R.J. (1984). Alternative conceptions of intelligence and their implications for education. *Review of Educational Research*, *54*(2), 179-223.

Walker, D.F. (1987). Logo needs research: A response to Papert's paper. *Educational Researcher*, *16*(5), 9-11.

Ward, J. (1988). Developments in intelligence testing and its application: A personal viewpoint. In A. Watson (Ed.), *Intelligence: Controversy & change* (pp. 44-60). Hawthorn, Victoria: Australian Council of Educational Research.

Watson, A. (Ed.). (1988). *Intelligence: Controversy & change*. Hawthorn, Victoria: Australian Council for Educational Research.

Watt, D. (1979). Final report of the Brookline Logo project, Part III: Profiles of individual students' work. *Logo memo No. 54*. Cambridge: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Watt, D. (1982). Logo in the schools. *Byte*, *7*(8), 116-134.

Watt, D. (1983a). Learning with Logo. *Classroom Computer News*, *3*(5), 40-43.

Watt, D. (1983b). *Learning with Logo*. N.Y.: McGraw-Hill Book Company.

Watt, D. (1989a). Strategies for learning through exploration. *Logo Exchange*, *7*(6), 23-26.

Watt, D. (1989b). Planning, carrying out and completing a Logo project. *Logo Exchange*, *7*(8), 20-25.

Watt, M., & Watt, D. (1986). *Teaching with Logo: Building blocks for learning*. Menlo Park, California: Addison-Wesley Publishing Company.

Webb, N.M. (1984). Microcomputer learning in small groups: cognitive requirements and group processes. *Journal of Educational Psychology*, *76*(6), 1076-1088.

Webb, N.M. (1985). The role of gender in computer programming learning processes. *Journal of Educational Computing Research*, *1*(4), 441-458.

Webb, N.M., Ender, P., & Lewis, S. (1986). Problem solving strategies and group processes in small groups learning computer programming. *American Educational Research Journal*, *23*(2), 243-261.

Wechsler, D. (1974). *Wechsler intelligence scale for children - revised*. N.Y.: The Psychological Corporation.

Weiner, B. (1979). A theory of motivation for some classroom experiences. *Journal of Educational Psychology*, *71*(1), 3-23.

Weiner, B. (Ed.). (1974). *Achievement motivation and attribution theory*. Morristown, N.J.: General Learning Corporation.

Weinreb, W. (1982). Problem solving with Logo. *Byte*, *7*(11), 118-134.

Weinstein, C.E., & Underwood, V.L. (1985). Learning strategies: The how of learning. In J.W. Segal, S.F. Chipman, & R. Glaser (Eds.), *Thinking and learning skills: Vol 1. Relating instruction to research* (pp. 241-258). N.J.: Lawrence Erlbaum.

Weir, S. (1981). *Logo as an information prosthetic for the handicapped* (Working Paper No. 9). Cambridge, MA: MIT Division for Studies and Research in Education.

Weir, S. (1987). *Cultivating minds: A Logo casebook*. N.Y.: Harper & Row. Cited in Noss, R. (1987).

Weir, S., Russell, S.J., & Valente, J.A. (1982). Logo: An approach to educating disabled children. *Byte*, *7*(9), 342-360.

Wellman, H. (1985). The origins of metacognition. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller (Eds.), *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives* (pp. 1-31). Orlando, Florida: Academic Press.

Wertheimer, M. (1959). *Productive thinking*. N.Y.: Harper & Row.

Wharton, J.S. (1986). Prompted writing using a computer. *Information Transfer*, *6*(4), 26-28.

White, C.S. (1987). Developing information-processing skills through structured activities with a computerized file-management program. *Journal of Educational Computing Research*, *3*(3), 355-375.

Wickelgren, W.A. (1974). *How To solve problems: Elements of a theory of problems and problem solving*. San Francisco: W.H. Freeman and Company.

Williams, A. (1987). Computer based learning environments for children with special needs. *Australian Journal of Special Education*, *11*(1), 36-43.

Williams, F., & Williams, V. (1984). *Microcomputers in elementary education*. London: Wadsworth.

Williams, M. (1991). National Keylink projects. In *Navigating the nineties* (pp. 332-337). Brisbane: Computer Education Group of Queensland.

Williamson, P.A., & Silvern, S.B. (1986). Parental teaching style while working with their children on Logo concepts and the effects of teaching styles on a subsequent task. *Early Childhood Research Quarterly*, *1*, 407-415.

Wills, S. (1984). Is there life after Logo? In A.D. Salvas (Ed), *Computing and education - 1984 and beyond* (pp. 281-284). Balaclava: Computer Education Group of Victoria.

Wood, D., Bruner, J.S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology*, *17*, 89-100.

Wood, D.J. (1978). Problem solving - The nature and development of strategies. In G. Underwood (Ed.), *Strategies of information processing* (pp. 329-356). London: Academic Press.

Wood, D.L. (1986). Microcomputer applications beyond drill and practice: new challenges in special education. *Australian Journal of Special Education*, *10*(2), 29-32.

Woodward, J., Carnine, D., & Collins, M. (1988). Closing the performance gap: CAI and secondary education for the mildly handicapped. *Journal of Educational Computing Research*, *4*(3), 265-286.

Woodworth, R.S., & Schlosberg, H. (1954). *Experimental psychology*. N.Y.: Holt, Rinehart, & Winston.

Yates, B.C., & Moursand, D. (1988). The computer and problem solving: How theory can support classroom practice. *The Computing Teacher*, *16*(4), 12-16.

Young, R.M. (1978). Strategies and the structure of a cognitive skill. In G. Underwood (Ed.), *Strategies of information processing* (pp. 357-402). London: Academic Press.

Ysseldyke, J.E., O'Sullivan, P.J., & Thurlow, M.L. (1989). Qualitative differences in reading and math instruction received by handicapped students. *Remedial and Special Education*, *10*(1), 14-20

Yussen, S.R. (1985). The role of metacognition in contemporary theories of cognitive development. In D.L. Forrest-Pressley, G.E. MacKinnon, & T.G. Waller (Eds.), *Metacognition, cognition, and human performance: Vol. 1. Theoretical perspectives* (pp. 253-283). Orlando, Florida: Academic Press.

Zaks, R. (1983). *Your first IBM PC program*. Berkeley, California: Sybex Inc.

Zelman, S. (1985, April). *Individual differences and the computer learning environment: Motivational constraints to learning Logo*. Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Zuk, D. (1986). The effects of microcomputers on children's attention to reading. *Computers in Schools*, *3*(2), 39-51.

Zuk, D., & Danner, F. (April, 1986). *The effects of microcomputers on children's attention to reading tasks*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

APPENDIX 1

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Studies without teacher mediation								
Milner	1973			Logo programming & the facilitation of the understanding of number sequences & variables.			18 5th graders. 15 weeks of intervention: 1st five weeks - logo basics; 2nd five weeks - writing procedures (3 instructional groups); last five weeks - independent programming tasks.	Logo programming can be a useful tool in the teaching of elementary mathematics. Logo group was able to write programs successfully & demonstrated an ability to use variables.
Weir	1981	Socio-affective issues.					11 subjects who were severely handicapped. They learned Logo over a period of two years.	Logo could be used successfully with severely physically handicapped but mentally alert cerebral palsy adolescents. Activities included programming, maths and problem solving activities. These activities provided a high degree of motivation for these learners.
Gorman & Bourne	1983				Logical reasoning - rule learning task.		15 3rd grade students received Logo instructions for one school year (5 of them 1 hour per week; 10 1.5 hours per week).	Logo students improved on logical reasoning.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Pea	1983				Planning skills.		50 subjects (2 classes of 25 each; 8-9, 11-12 yrs old) with a matched group of non-programming students. 30 hours of programming in one year.	No statistical significance on: elliciency of planning, quality of revisions, types of decisions made during the planning process. Most children exhibited very poor understanding of commands and fundamental concepts such as variables and conditional statements. They also have problems with procedural errors, sequentiality of program execution, and the model of recursion within Logo programming.
Rieber	1983				Logo programming & geometric concepts. Solving problems of a combinatory nature.		22 (age: 7-9) 11 in Logo group; 11 in control. Logo group learned Logo programming one hour per week over a period of 3 months.	Logo group performed better in systematically solving abstract problems of a combinatory nature (combination & permutations).
Evans	1984	Attitudes towards learning mathematics.		Attitudes towards learning mathematics.	Cognitive abilities.		8 4th grade classes, 7 4th grade classes acted as control. Subjects worked in pairs for 45 hours in a year.	Logo programming had a positive influence on cognitive abilities & on attitudes towards learning mathematics.
Hawkins, Homolsky, & Heide	1984		Collaboration when learning with Logo.				100 subjects (8-9 and 11-12 year olds in two classrooms; two cycles of 50 students). The Logo sessions were conducted over a two year period with two different groups of 50 subjects each although the actual time spent on Logo programming was not reported.	Subjects did collaborate more when they were working on microcomputer problems than they did on other classroom tasks. But as the year progressed, there was a greater occurrence of individuals working alone at the computer in a very focussed way. The computer provided an engaging problem solving context in which task-related talk occurred.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Mendelsohn	1984, 1985, 1987				Psychological analysis of cognitive mechanisms peculiar to programming, which he later sets in relation.		25 subjects (11 -13 yrs). 16 one hour sessions on Logo.	Mastery of programming related to operational thinking level of learners.
Bradley	1985				Relationships among Logo programming, information processing styles, academic achievement, and cognitive abilities.		26 subjects (7-11 yrs; grades 2,3,4,5,& 6; 18 boys & 8 girls). 15 weeks of 1 hour each.	Top-down processing, as measured by writing activity, is positively related to Logo programming, field independence, holistic tendencies, & academic achievement. Logo success & academic achievement were positively related.
Carmichael, Burnett, Higginson, Moore, & Pollard	1985		Peer interaction.		Problem solving in real classroom settings.		433 students 18 classrooms that involved 13 different teachers over a two year period (of these, 5 teachers & 40 students were involved in the study). Logo and word processing were studies extensively.	Extended pairing may lead to conflicts. Logo can be a powerful medium for developing problem solving skills based on real needs rather than on hypothetical and irrelevant situations.
Horner & Maddux	1985	Locus of control.		Mathematics attitude; geometric angle recognition.	Problem solving skills.		74 subjects (junior high) (two experi-mental groups & 2 control groups) (one intact group of mixed 7th & 8th LD mathematics students). 14 hours in 6 weeks. Control groups received regular mathematics classes.	No significant differences although further analysis on mathematics attitudes indicated that Logo might be effective in making both LD and non-LD students feel responsible for their success with the Logo activities.
Hughes, Macleod & Patts	1985			Comprehension of mathematics concepts	Programming competence; changes in cognitive planning(BAS)		17 subjects - 11 boys & 6 girls (mean age: 11.6 yrs). Subjects learned Logo programming for 24 sessions of 15-20 minutes each.	Improved comprehension in certain mathematical concepts; gains in block design, number memory, and arithmetic.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Irwin	1985	Attitude towards computing.	Attention to Logo and Basic.		Problem solving with Logo and Basic		140 subjects (36 in each of two Logo groups; 34 in each of Basic groups). High & Low ability groups for both Logo and Basic. 60 ½ hour sessions over three months.	The level of attention of most groups were remarkably high although Logo groups tended to spend significantly more time "on task". Children of lower ability were much less interested in working with Basic. This study did not pursue in the testing whether Logo would improve problem solving skills as researcher argued that there was little similarity in the content of the mathematics syllabus and the type of problem solving that was occurring in the computing sessions. Logo groups continued to report high level of interests whereas Basic groups showed a decrease.
McAllister	1985				Cognitive strategies. Problem solving strategies used in Logo programming & transfer (Tower of Hanoi).		8 subjects (4 boys & 4 girls). 6 weeks of Logo treatment (basic turtle graphics) with weekly morning sessions with the whole group, or the instructor worked with children in small groups of 3-5. One to one training was given toward the end of the project.	Positive correlation between the scores of Tower of Hanoi with measures such as program writing, program creating, program reading, and the total for programming measures, thus suggesting that skills learnt while learning Logo programming might transfer positively to other non-programming environment bearing similar properties.
Zelman	1985	Motivational constraints.					4 girls (12 - 16 years of age) were observed for approximately 50 hours while programming with Logo. Attitudinal questionnaires were also administered. 17 students interviewed.	An inductive teaching method was inappropriate to motivational orientations of students. Researchers called for more controlled studies to examine how instructional practices might change the orientation of learners over time.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Olsen	1985	Locus of control.		Geometry achievement, spatial visualization ability.			42 6th graders. 8 weeks of Logo instructions.	All children showed gains in geometry achievement, but those in the Logo group improved in spatial visualization ability as well. Logo was effective in helping girls develop increased feelings of responsibility and personal control.
Webb	1985		Group processes when learning with Logo.		Planning and debugging behaviours re gender differences.		35 subjects (grades 7 to 9; 15 girls & 20 boys) learned Logo in three-person groups. Learned Logo for a total of 15 to 20 hours.	Males and females showed no differences on any learning outcome and showed very few differences in verbal behaviour in planning and debugging activity.
Campbell, Fein, Scholnick, Frank, Schwartz	1986				Competence with the syntax & semantics of the Logo language.		20 subjects (5 - 6 year olds; 10 females, 10 males). Subjects received a total of 50 - 60 minutes of individualized instant Logo instructions.	Subjects reorganized their model of Logo and became more systematic although the researchers suggested that further study is needed to verify this reorganization.
Carver & Klahr	1986				Debugging skills.		9 subjects (7 - 9 year olds; 5 females & 4 males). 24 hours of Logo programming over a three week period.	Subjects did not develop effective debugging strategies. The researchers concluded that it is important to teach debugging skills directly to the learners.
Chambers	1986				Cognitive measurements such as experimenting, predicting, using analogies, coding, analysis and planning, and debugging.		312 subjects (aged between 5 to 12). The mean number of Logo sessions was 63 with a standard deviation of 25 and a range from 5 to 100 sessions. Does not provide explicit instruction in general problem solving skills or transfer training.	Logo experiences enhance: computing performance, some general thinking skills (Ravens test), but does not enhance performance on similar tasks. Researchers suggest more substantial experience is needed.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Clement, Kurland, Mawby, & Pea	1986				Analogical reasoning.		17 subjects (11 grade, all females). Discovery learning. Received 45 hours of instructions in 6 weeks.	High correlations between structure mapping of analogical reasoning with two aspects in programming: writing subprocedures and use of variables.
Cuneo	1986a				Problem solving in a Logo environment.		32 4- & 5- year olds. A very simplified graphics environment. 3 to 6 30 minute sessions.	The subjects could not easily generate a two- or three-command program. Their ability to give the correct sequence, and at least the appropriate number and type of commands, improved in the course of study. Limited ability in debugging program.
Degelman, Free, Scarlato, Blackburn & Golden	1986				Logical thinking - Rule learning.		15 kindergarten students 8 in Logo group 7 in control. Students in Logo group received instructions for 15 minutes per day for 5 weeks; they worked in pairs.	Logo group performed significantly better in rule-learning problems involving affirmatively defined concepts but not conjunctively defined concepts.
Kurland, Pea, Clement, & Mawby	1986			Mathematical abilities.	Procedural reasoning ; planning; understanding of programming.		45 subjects in three groups (Grade 10-12). They studies 6 programming languages (9 weeks each) with the Logo curriculum designed by the researchers. The Logo programming lessons (at the end of the year) were of 40 minute each day, 5 days a week for 9 weeks.	Many students only have a rudimentary understanding of the concepts in programming. Programming experience did not appear to transfer to other domains which shared analogous formal properties.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Lehrer, Harckham, Archer, & Pruzek	1986	Affective development.			Cognitive development. Problem solving skills acquisition.		120 pre-school children (Logo group and instructional software group). 3 times a week for 25 mins over 12.5 weeks. Subjects in Logo group usually worked in pairs or small groups (2-5).	Logo based environment enhanced children's problem solving skills and acquisition of linguistic pragmatics. Instructional software promotes acquisition of specific skills such as verbal directions, colours, & sorting. However, neither software environment enhanced children's global levels of cognitive or affective development.
Miterer & Rose-Krasnor	1986		A systematic observation to measure different behaviours in the learners.		Problem solving including: block design, Tower of Hanoi, flexibility for problem solving (fluency & originality), operational tests (balance & probability).		96 subjects (age: 6.11 yrs), 4 groups (Logo, Basic, problem solving and control). 40 hours of training over 7 weeks.	Logo learning did not promote the transfer of problem solving to other domains. Suggest that students should be made explicitly aware of the general utility of the "powerful ideas" which might in turn enhance the transfer of problem solving skills.
Williamson & Silvern	1986		Interaction of parents & children.		Problem solving of a logico-mathematical nature.		22 dyads of parents & children. One hour each day for 10 days.	Children with directive parents performed better on a generalization tasks than children who were less directive.
Cohen	1987		General effects of implementing a Logo program in a typical primary classroom.		Mastery of Logo programming concepts.		23 2nd graders. The computer activity was carried out in teams of two students, 20-30 minute sessions throughout the whole day, throughout the school year.	Logo activities tended to generate an atmosphere of excitement and enthusiasm. Subjects did not reach level of proficiency needed for successful completion of projects.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Fay & Mayer	1987				Misconceptions and confusions about Logo graphics commands.		99 subjects (34 4th graders; 34 5th graders; 31 8th graders). 30 to 45 min lesson: an introduction to the turtle & graphics commands. Subjects then tested on their understanding of turtle commands.	Children, especially of elementary school age, often harbour preconceptions about spatial reference that conflict with the conceptions underlying Logo. The effectiveness of Logo may depend upon on the instructor's sensitivity to the characteristics of each student. Care must be taken to consider student's level of development so they will benefit from the learning of Logo.
Gallini	1987				Enhancing cognitive outcomes: follow directions. Constructing directions in the process of problem solving.		44 4th graders (22 in control (CAI treatment)). 75 mins, 3 times a week for 5 weeks (under 20 hours in total).	Logo group was able to better formulate directions than the control group. Researcher maintains that: (i) a potentially positive relationship between Logo training & success on similar types of tasks; (ii) Metacognitive types of questions helped to encourage reflective thinking among learners.
Guntermann & Tovar	1987		Effects of group size and group composition on learning Logo.				36 subjects (10 yrs) learned individually or in groups of two or three for one session, had a practice session, then were required to produce a graphic in Logo for the experimental session.	No differences were found between individuals and groups, in terms of productivity. Group interaction was found to be similar in two and three person groups. Significant differences were observed among male, female, and mixed groups: males displayed more solidarity than female or mixed groups; females were much more likely to express agreement with their peers; there were also more asking of information in the male groups than female groups; males expressed more antagonism than females or mixed groups.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Lehrer & Randle	1987				Planning, prediction, and revision.	Test of metacognition.	39 subjects matched with respect to their scores on the Brigance developmental inventory, and then assigned to 3 conditions: Logo programming, software to aid composition & problem solving, control. Logo group received instructions for 5 months, 2 times per week for 20 minutes each time.	Both software conditions were associated with increased problem solving efficiency, but only Logo condition results in durable increases in problem solving efficiency. Logo group increased in comprehension monitoring, and ability to monitor and establish relationship between old and new information.
Mayer & Fay	1987				Cognitive changes (model by Linn).		30 grade four students. Students only received 3 sessions of Logo instructions totalling approximately 130 minutes.	Logo programming can modestly influence children's thinking in areas similar to those involved in programming. When teaching programming, some diagnosis and guidance, and mediational learning would likely to assist in the transfer of skills from the programming domain to other domains such as map reading.
Noss	1987a			Children's learning of geometrical concepts.			Logo group: 84 Control: 92 five classrooms: one from each of five schools one grade 3, one grade 4, three grade 5. aged: 8-11. The Logo group programmed in pairs for a median time of about 75 minutes per week over one school year.	Logo learning helped to improve children's development of geometric concepts such as length conservation, length measurement, angle conservation, and angle measurement.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Lee & Lehrer Two studies	1988				General properties of cognition and specific instructional practices.		(i) 7 graduate students. (ii) 24 adult students. Logo instructions for 1.5 hours each week for eight weeks. Unlimited access to computers.	Researchers found that previous experience in Basic programming resulted in negative transfer while learning to program with Logo. Many of the students' misconceptions were remediable through better pedagogy.
Many, Lockard, & Abrams	1988				Reasoning skills.		113 in Logo classes, 58 in control group. Students in Logo group received Logo instructions for 45 minutes per day over a 9 week period.	Logo group scored better than the control although further analysis indicated that the males in Logo group achieved significantly higher scores than their male counterparts in the control.
Schaefer & Sprigle	1988			Preschoolers' development of mathematics concepts together with computer terminology.			Preschool children: 10 boys & 10 girls. Enrolled in a university laboratory school. They received instruction in Logo on a daily basis for three months in sessions ranging from 10 to 30 minutes.	All subjects improved in all the three areas on a pre-post test design.
Thomson & Chen Wang	1988			Transfer of learning of mathematical concepts.			40 subjects, grade 6 (23 females & 17 males). Logo and control (20 each). Logo group learned Logo programming for 45 minutes each day for 3 days. Control group worked on their normal mathematics lessons.	Logo performed better on both knowledge of the concepts and ability to transfer the concepts although the greater transfer occur in the measure of transfer.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Turner & Land	1988			Mathematical concepts such as properties of polygons, angle measurements, estimation, rectangular coordinate systems, negative numbers, and variables.	Levels of cognitive development.		181 subjects (91 in Logo group & 90 in control; 5th, 6th, 7th & 8th graders). Logo group learned Logo one hour per week for 16 weeks.	No significant group differences in the understanding of mathematical concepts or cognitive development. However, those who learned most Logo gained significantly more than those with a minimal mastery both in understanding of mathematical concepts & level of cognitive development. This study suggests that cognitive development, achievement in mathematics, and achievement in Logo programming all share a common factor.
Burns & Hagerman	1989	Self-concept; locus of control.					22 3rd graders (11 in experimental group; 11 in control group). Experimental group learned Logo programming - 20 - 25 minutes per week over 4½ months; control group used Delta Drawing.	Logo group showed significant increases in internal locus of control.
Hoyles & Sutherland	1989		Gender differences in a Logo environment	Learning of mathematics through the use of Logo			A longitudinal study of: 4 pairs of students (aged 11-14; one boy pair, one girl pair, and two mixed pairs) for 3 years; 4 pairs of students (aged 11-12) for 1 year; 32 pairs of students (aged 11 -14) for 2 years. Children worked in pairs while learning to program with Logo during mathematics lessons.	Logo programming provided an engaging problem solving context. Collaborative exchanges were found to be important for children's learning. Gender differences: (i) boys - difficult to share interactions & tended to dominate in mixed pairs; (ii) girls preferred to choose loosely defined goals. Teacher's intervention was crucial in students' learning.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
King	1989		Verbal interactions & problem solving behaviours.		Problem solving & verbal interactions.		36 4th graders assigned to groups of three to form 6 groups of high and 6 of average academic ability. Subjects used a non-programming version of Logo turtle graphics to reproduce a given line design on the computer screen.	No relationship between success & ability, and that successful groups asked more task-related questions, spend more time on strategy, and reached higher levels of strategy elaboration than did unsuccessful groups. High ability groups made a greater number of long task statements than did average groups.
Burns & Coon; 2 studies.	1990		Logo programming & peer interaction.		Peer interaction, Logo programming & problem solving.		Study 1: 20 3rd graders (8.5 yrs). Experimental group learned Logo programming individually - 20 - 25 minutes per week over 4½ months; control group used Delta Drawing. Study 2: 18 3rd graders (8.6 yrs). Both experimental group & control group (10 & 8) received instructions in pairs; 9 20 minute sessions over 6 weeks.	Extensive Logo experiences may influence peer collaboration on problem solving tasks. Peer collaborations using Logo were shown to focus more on the process relative to the product of problem solving.
Cathcart	1990				Cognitive styles: field independence dependence, divergent thinking, impulsivity reflectivity.		43 5th graders (25 experimental group, 18 control). Two 45-minute sessions per week for 14 weeks. Total hours = 20.	Logo group performed better in divergent thinking. Both groups gained significantly in field dependence independence. Decline in latency for Logo group was significant.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Mathinos	1990				Changes in the types of problem solving skills. Possible transfer of these skills to noncomputer situations.		40 LD & 40 ND subjects (grades 4-6). One hour a day each day for 16 weeks (80 hours).	Programming with Logo under specific conditions used in this study allowed some children to refine and extend their use of problem solving skills both within & across computer and noncomputer contexts.
Schibeci	1990	Attitude towards computers & learners themselves.			Development of problem solving strategies.		63 pre- and in- service teachers (4 different groups according to their enrolment in different courses at the university). Logo treatment varied according to course requirements (unclear as to how many hours subjects actually spent on Logo).	Subjects showed a marked improvement in their attitude towards computers. They were also more confident in solving problems while programming with Logo.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Studies with teacher mediation								
Howe, O'Shea, Plane	1980			Algebraic topic.			Two classes of Primary 7 students (22 subjects, with 11 in each; 11 - 12 years old). 1 hour per week over two school years. One class learned Logo programming during first year, used Logo to explore troublesome topics in mathematics in the second year. Structured worksheets were used to develop various problem solving skills such as decomposing, debugging etc.	An item analysis of the school maths tests suggests that the Logo group's performance was marginally better than that of the control group. Logo students could argue sensibly about mathematic issues and explain their own mathematical difficulties more clearly.
Seidman	1981				Conditional reasoning ability.		42 5th graders - half in control. 15 weeks of treatment, 2 hours per week.	Traditional scoring of tests showed no significant different in scores. However, an additional scoring procedure, indicated significantly better performance of the experimental group on one of the conditional reasoning principles.
Finlayson	1983			Mathematical development.			26 subjects (primary 5: 9 yr; primary 6: 10 yr); no control Average of 20 hours of Logo learning time.	Children were able to "think mathematically through Logo experience", and that a great deal of enthusiasm was generated. However, the researchers also noted that children could appear to be competent at turtle graphics without comprehending the underlying mathematics.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Pea & Kurland	1983				Understanding of the concept of recursion.		7 subjects (2 girls & 5 boys, 11-12 yrs) who have spent more than 50 hours of classroom programming. Discovery learning.	Systematic misunderstanding of recursive programs. Poor understanding of sequential execution.
Clements & Gullo	1984				Cognitive style (creative thinking, matching familiar figures), cognitive skills (screening tests, classification, serialization), spatial orientation. Metacognitive skills in aiding problem solving.	Metacognitive skills.	18 subjects (6 year olds) randomly assigned to Logo and control (CAI) groups. 2 40 minute sessions a week for 12 weeks; worked in groups of 2 or 3.	Logo group scored higher on measures of reflectivity & two measures of divergent thinking; outperformed CAI group on measures of metacognitive ability and ability to describe directions. No differences were found on measures of cognitive development.
Pea & Kurland	1984				Planning skills.		32 subjects (16 8-9 yrs; 16 11-12 yrs). Logo group spent about 2 45-minute sessions per week, with a total of about 30 hours programming in Logo.	Students who have spent a year programming did not differ on the effectiveness of their plans and their processes of planning from same age controls.
Finlayson	1985			Transfer of mathematical strategies from Logo to normal school mathematics.			64 subjects (from two parallel mixed ability groups (11 yrs old) of 32 students each).	Experimental group showed overall superiority in: understanding of concepts of angles and variables; ability to use mathematical strategies of generalization and abstraction; and ability to pick out relevant information in novel problems not directly related to Logo.
Kinzer, Littlefield, Declos, & Bransford	1985		Effects of Logo on discipline and organization..		Instructional approaches on mastery of Logo learning.		38 5th graders into 2 groups (2 different Logo instructional conditions). 1 hour per day for 25 days.	Logo classrooms exhibit more learning-oriented interactions than do normal classrooms. No observable differences between instructional approaches but that might be due to inadequate measures.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Battista & Clements	1986			Mathematics achievement.	Problem solving processes.		17 4th graders & 39 6th graders. Three groups within each grade: Logo, CAI problem solving, and computer literacy (control). 42 sessions of 40 minutes per week.	No difference in the first problem solving test but Logo group performed better in the second problem solving test (test of metacomponential problem solving processes). Researchers concluded that Logo programming can increase certain problem solving abilities, especially those related to executive processes such as cognitive monitoring, selecting a mental representation, and deciding on performance level processes.
Clements	1986				Cognitive style (creative thinking, matching familiar figures), cognitive skills (screening tests, classification, serialization), spatial orientation.	Metacognitive skills.	36 1st graders (6 yrs & 10 mons), 36 3rd graders (8 yrs & 10 mons). 3 sub-groups: Logo, CAI, and control). 22 weeks, two 45-min periods per week.	Significantly better performance in classification, seriation, TCCT, and spatial orientation tests. Researchers suggest that these results indicate the important contribution of Logo to the development of operative competence in children when Logo intervenes at a given point. Logo group performed better in the metacomponents of problem solving, comprehension monitoring, and creativity.
Horton	1986				Cognitive skills: exploration, analysis and planning, creativity, debugging, coding, and predicting.		16 subjects (8 in expt groups and 8 in control). Pre-post tests. 14 sessions over 7 weeks. Learning was based on the development of thinking skills.	Logo group performed significantly better in: exploration, analysis and planning, and prediction. Control performed better in debugging skills.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Lehrer & Smith	1986			Mathematical understanding.	Cognitive consequences.	Metacognitive consequences.	47 3rd graders from two randomly selected classes. 45 minutes each week for 9 weeks. Two instructional conditions: (i) teacher mediated (24); (ii) traditional (23).	Students who were better instructed were able to (i) use their knowledge in Logo to solve mathematical problems when reminded how such knowledge could apply to the problem; (ii) integrate new with old information (based on a measure of metacognition). However, there was little difference in problem solving strategies between the two groups.
Miller & Emihovich	1986				Problem solving - a block building task.		14 subjects (8 boys & 6 girls; 5 yrs & 4 mons): 2 groups, Logo and CAI. 11 Logo lessons over a 3 week period. Control group: computer game. Logo instructions were provided within a mediated instructional framework.	Logo group better able to detect embedded errors.
Clements	1987				Cognitive abilities.	Application of metacognitive skills.	Same subjects as in Clements (1984) (delayed effects of Logo programming.	Logo group was better able to apply metacognitive skills such as those involved in solving analogies and sequences, which include the ability.
Howell, Scott & Diamond	1987				General cognitive development: conservation of number, length & drawing of Euclidean shapes.		67 subjects - (5-6 yrs old); 34 in Logo group, the rest in control. Using an expanded form of instant Logo with guided discovery learning. Logo group learned Logo programming for 75-80 minutes per week over 5 months.	No significant statistical results although anecdotal reports by teachers suggest that Logo experience did make a positive impact on some areas of cognitive development such as directional understanding, shape labelling & construction.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Clements & Nastasi	1988		Social competence components of problem solving.		Informational processing components of problem solving.	Metacognitive processing.	24 1st graders (6 yrs & 6 mons), 24 3rd graders (8 yrs & 8 mons). 28 training sessions of 45 mins over 14 weeks. Intervention similar to those in earlier studies.	Logo group exhibited a significantly higher percentage of social behaviours that have cognitive underpinning and/or would be expected to occur in problem solving situations. Logo group exhibited a significantly higher frequency of behaviours indicative of metacognitive functioning.
Lehrer, Guckenberg, Lee	1988			Description of geometric concepts.	Solving a planning task.	Increase in metacognitive skills.	45 3rd graders (two Logo groups: programming strategies; geometry instructions; control). 47 ½ hours sessions, two times a week).	Children in Logo groups solved a planning task more efficiently; and developed more dynamic descriptions of geometric concepts (enhanced level of understanding of geometry). Children learning geometry with Logo also demonstrated increased metacognitive skills.
McDougall	1988		Social interactions between learners	Recursion			Two children (6 and 9). Learned Logo in a home learning environment rich in materials and opportunities for learning about recursion. A cases study methodology was used.	The importance of teacher expectations and of social interactions between learners in determining children's levels of achievement (abstract thinking) were emphasized by events in this study.
Clements	1990					Metacomponential development.	48 3rd graders (20 boys & 28 girls; randomly assigned to one of two group: Logo & control). These groups met three times a week (45 mins each) over 26 weeks.	Logo group scored significantly higher on the total assessment of executive processing. Features of the instructional environment such as explicitness and completeness, help account for these facts.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Nastasi, Clements, & Battista	1990	Motivation	Social-cognitive interactions.		Cognitive growth.		40 subjects (12 4th graders & 28 6th graders) randomly assigned to either Logo or CAI groups. 2 40 min sessions per week for a total of 42 sessions during the school year. Subjects worked in pairs.	Logo group evinced more cognitively oriented conflict, attempts at and successful resolution of conflicts, rule making, and pleasure at discovery.
Au & Leung	1991				Effects of Logo learning on the facilitation of problem solving in a non-programming context.		60 subjects (20 in each: process-oriented; content-oriented; control). One hour per week for 25 weeks. Pre post design. Three measures: Ravens, Tower of Hanoi, Rule learning. Logo groups outperformed control group in Tower of Hanoi.	Process-oriented group performed better than content-oriented group in some sub-tests of Tower of Hanoi. Based on these evidence, researchers conclude that: (i) Logo programming might facilitate near transfer of problem solving skills; (ii) transfer of skills could perhaps be enhanced by a process-oriented approach in the teaching of Logo.
Campbell, Fein & Schwartz	1991			Estimation of distance			48 first graders (23 in experimental group & 25 in control group). Experimental group received 20-25 hours of Logo instruction using a guided instructional approach	Children who has Logo instruction were significantly more accurate in estimating distance, and more reflective of the strength of the inverse relationship between unit size and number.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Clements	1991				Creativity		73 subjects (33 boys & 40 girls; 8 yrs) in three groups (Logo programming, nonLogo creativity, and nontreatement). 3 45-55 min sessions per week for 25 weeks. Subjects worked in pairs under the guidance of one or two teachers.	Logo group had significantly higher scores than the other two groups on figural creativity. Both Logo and creativity group scored performed better on verbal creativity. An implication is that certain computer environment may offer opportunities for enhancement of both figural and verbal creativity.
Grandgenett & Thompson	1991				Transfer of analogical reasoning		144 university students of introductory educational computing class, given 12 hours of Logo instruction. One group experienced Logo programming instruction guided toward the development of general analogical reasoning; the other group experienced more traditional exploratory Logo programming instruction. Both near and far transfers were examined.	Far transfer results indicated significant interaction between a student's college year and the experimental treatment, with guided programming instruction facilitating the performance of college freshman, and hindering the performance of college juniors. Near transfer results indicated that the guided instruction did not significantly increase student reuse of subprocedures between programming problems.
Heller	1991				Learning of programming.		17 3rd graders. Nine worked with extended workstations, while 8 used only traditional Logo programming environment. Each subject worked for one period a week for twenty weeks.	When students were provided with extended workstations, they could deepen their understanding of the semantics of Logo commands. Also, they were able to explore Logo as evidenced by the more complicated programming products.

Authors	Year	Affective	Social	Mathematics	Cognitive skills and problem solving	Metacognitive	Subjects	Results
Ortiz & MacGregor	1991			Understanding the concept of variables, & attitudes towards mathematics.			89 6th graders (47 female, 42 male) from four classrooms. All subjects received 5 50-min lessons on general Logo programming (not involving variables). The Logo experimental group then received another 5 50-min lessons on Logo programming involving variables whilst the other experimental group received variable instruction using textbooks.	Students who programmed Logo procedures with variables demonstrated greater long term retention of their understanding of the concept of variable than the students in the textbook group. Students had more positive attitudes toward computer related aspects of instruction. This study also underscores the importance of providing a direct link between programming instruction and specific content area skills.
Swan	1991				Problem solving abilities: subgoals formation, forward chaining, systematic trial and error, and analogy.		101 subjects - 4th graders (30); 5th graders (35); 6th graders (36). All of them had at least one year (30 hours) prior experience programming in Logo. Subjects were randomly assigned by grade to one of three treatment conditions: Logo graphics & problem solving condition, a cut-paper manipulatives condition, or a Logo discovery learning condition. They worked in pairs or groups of three.	Explicit instructions with Logo programming practice supported the development and transfer of four problem solving strategies; whereas neither discovery learning in a Logo environment nor explicit instruction with concrete manipulatives practice did so.

APPENDIX 2

LOGO - PROCESS-ORIENTED GROUP

SESSION 1

MEET THE TURTLE

1. Type: **SHOWTURTLE**
   Press: Enter

   If you make any typing mistake, use the Backspace key to erase the mistake and type again.

   The little triangle in the middle of the screen represents the TURTLE.

   Type: **HIDETURTLE**
   Press: Enter

   Type: **SHOWTURTLE**
   Press: Enter

   What does the command SHOWTURTLE do? _____

   What does the command HIDETURTLE do? _____

   If you are not sure, try these two commands again until you understand what they can do.

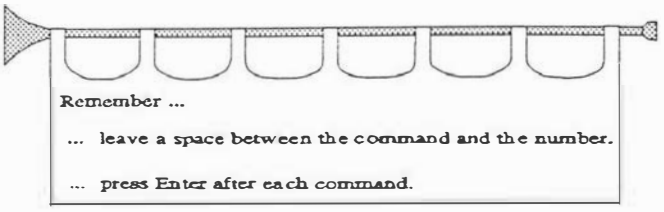
2. Play with the commands below. Try different numbers with the commands, for example, FORWARD 20, BACK 30, RT 60, LT 200 etc.

   FORWARD

   BACK

   RIGHT

   LEFT



3. Guess what will happen with the following commands. Draw your guesses in the space below. When you have finished drawing your guesses, try them out with the computer to see if your predictions were correct. Remember to press the Enter key after each command.

   CLEARSCREEN
   FORWARD 50
   RIGHT 90

CLEARSCREEN
RIGHT 90
FORWARD 50

What does CLEARSCREEN do? _____

Are the turtle's tracks the same? _____

How are they different? _____

_____

4. Try to predict what the turtle will draw by drawing your prediction in the space below. Then try the commands out with the computer to see if your predictions were correct.

   Type: **CLEARSCREEN**
   FORWARD 60
   LEFT 90
   FORWARD 60
   HOME

   What does HOME do?_____

   Did you succeed in predicting the turtle's track? _____

   If you did not, find out what was wrong with your prediction.

   Find out the difference between the CLEARSCREEN and the HOME commands. If you are not sure, play with these two commands until you can find out the difference.

   _____

5. Guess what the following commands will draw. Draw your guess in the space below and then try them out with the computer to see if your guesses were correct.

   CLEARSCREEN
   FORWARD 20
   RIGHT 60
   FORWARD 20
   LEFT 60
   FORWARD 20

   Did you succeed in predicting the turtle's track? If not, find out what was wrong with your prediction.

Now

Type: **CLEARSCREEN**
Type: **FORWARD 20 RIGHT 60 FORWARD 20**
Type: **LEFT 60 FORWARD 20**

Do you get the same picture? _____

Which way of typing commands do you like better:

One command at a time?

or

More than one command at a time?

Why do you like that way better? _____

6. Record the turtle's complaint that is printed when you type each command below.

   Record how you would fix each command.

Type this:	Turtle's Complaint	Fixed Command
FORWARD100		
FORWARD		
RIGHT		
LEFT30		
BACK		
LEFT LOTS		
FORWARD 999999		

7. Guess what will happen with the following commands.

   Type: CLEARSCREEN
   FORWARD 100
   FORWARD 100
   FORWARD 100
   FORWARD 100

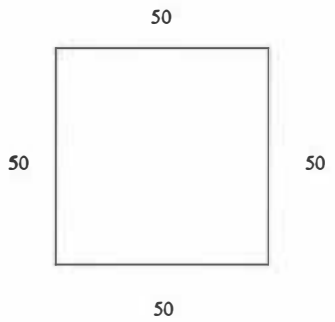
   What happens?

Did you succeed in guessing the turtle's track? If not, find out what was wrong with your guess.

8. EXERCISE

Use the commands that you have learnt today, draw pictures such as a square, a rectangle, a house etc. Write down your commands on a piece of paper. Make sure that you plan your drawings first. Try these commands with the computer during the next lesson to find out if the turtle will draw according to your plan. If it doesn't, then fix the mistake so that the turtle will draw what you want.

For example, in order to draw a square



You ask the turtle to go FORWARD 50 steps first, then you ask the turtle to make a RIGHT turn (how many degrees?).

After the RIGHT turn is made, you ask the turtle to go FORWARD again, and so on.

Just remember, planning systematically is most important.

If something goes wrong with your plan, ask yourself why something went wrong.

Change your plan and then try again.

LOGO - PROCESS-ORIENTED GROUP

SESSION 4

TURTLE TRICKS AND PATTERNS

1. There are some tricks that you can ask the turtle to perform:

   HIDETURTLE (HT)
   SHOWTURTLE (ST)
   PENUP (PU)
   PENDOWN (PD)
   PENERASE (PE)

   Try these commands with the computer and find out what they can do. (Use the shortcuts!!)

   First, try **HIDETURTLE**

   What does it do? _____

   Can you make the turtle appear again? _____

   How? _____

   Now try

   PENUP
   FD 50

   What happens, did the turtle draw? _____

   Can you explain that? _____

   Can you make the turtle draw again, how? _____

   Now try

   PENDOWN
   FD 50
   PENERASE
   BK 50

   What happens? _____

Can you explain that? _____

Which command erases the line that the turtle has just drawn? _____

Can you make the turtle draw again? _____

How? _____

5 STEPS IN PLANNING

Remember, always use this tactic when you plan:

- FIRST, THINK OF A PLAN.
- SECOND, ASK YOURSELF IF THE PLAN WOULD WORK BY FOLLOWING EVERY SINGLE STEP IN THE PLAN.
- THIRD, TRY THE PLAN OUT WITH THE COMPUTER.
- FOURTH, IF THE PLAN DOES NOT WORK, ASK YOURSELF WHAT WENT WRONG. MAKE SURE YOU CAN FIND OUT THE MISTAKES.
- FIFTH, CHANGE YOUR PLAN AND THEN TRY IT OUT AGAIN.

2. Draw a favourite shape of your own, and then erase it bit by bit after you have finished drawing it.

First, draw the shape here.

1. Think of a plan. Write it down here

2. Check your plan again. Make sure you follow every step in your plan.

3. Try your plan with the computer.

Did the computer draw what you have planned? _____

4. If not, what problems needed fixing? Write down the problems here:

5. What is your new plan? Write it down here:

Can you erase the picture bit by bit? _____

How? _____

Did you have any problems in erasing it? _____

How did you fix these problems?

_____

_____

3. There are some other tricks that you can teach the computer:

Explore the following commands and record here how they can be used:

TEXTSCREEN _____

MIXEDSCREEN _____

FULLSCREEN _____

In fact, there are shortcuts for these commands too:

Try these keys (at the top of your keyboard):

PF1
PF2
PF4

What do they do? (Are they the same as some of the commands that you have just learned?)

PF1 _____

PF2 _____

PF4 _____

How can you make use of these commands?

_____

_____

4. Following is a list of commands to draw a square (Can you recognize these commands?)

FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90

Try these commands with the computer.

Write down the set of the commands that is repeated to draw the square

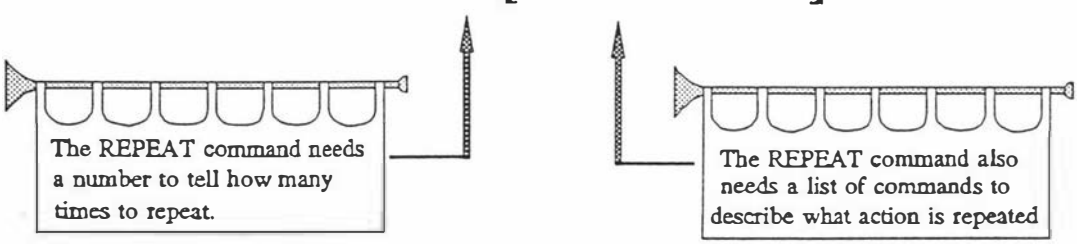
_____

How many times is this set of commands repeated?

_____

You can use the REPEAT command to draw the same square as a repeated list of commands

REPEAT 4 [FD 50 RT 90]



Can you clear the screen? _____

Which command should you use? _____

Draw a square with sides that are 50 turtle steps long using the **REPEAT** command.

1. First, write down your plan here.

2. Check your plan again, follow it through step by step.

   Do you think it will work? _____

3. Now, try your plan with the computer.

   Did your plan work? _____

4. If not, find out the mistake, write down your mistakes here:

5. Write down your new plan here and then try with the computer again.

Can you draw another square with sides that are 30 turtle steps long? Remember, follow the 5 steps in planning.

1. Your plan

2. Check your plan by going through the plan step by step.
   Do you think it will work? _____

3. Try your plan with the computer
   Did it work? _____

4. If not, what are the mistakes?

5. What is your new plan?

5. EXERCISE

Write down in your own words the 5 steps in planning that you have learned earlier in this session:

See whether you can draw a triangle and a hexagon with the **REPEAT** command (remember how to draw a triangle and a hexagon in Session 3?)

Drawing a triangle

1. Your plan:

2. Check it step by step. Should it work? _____

3. Try it out with the computer. Did it work? _____

4. What are the mistakes and how did you fix them?

5. What is your new plan?

Drawing a hexagon

1. Your plan:

2. Check it step by step. Should it work? _____

3. Try it out with the computer. Did it work? _____

4. What are the mistakes and how did you fix them?

5. What is your new plan?

PROCESS-ORIENTED GROUP

SESSION 8

SUBPROCEDURES AND SUPERPROCEDURES

1. You have learned the REPEAT command, e.g.

   REPEAT 360 [FD 1 RT 1]

   In order to use the REPEAT command, you have to tell the turtle two things,
   first, the number of times to REPEAT,
   second, what action or actions to REPEAT.

   Do you that you can actually REPEAT another REPEAT command or a procedure.

   Let's look at these procedures

```
TO SQUARE
  REPEAT 4 [FD 50 RT 90]
END

TO PATTERN
  REPEAT 4 [REPEAT 4 [FD 50 RT 90] RT 90]
END

TO PAT
  REPEAT 4 [SQUARE RT 90]
END
```

   Sketch your predictions of what these procedures will draw:

   SQUARE PATTERN PAT

   Teach these procedures to the turtle.

   Remember, the best way to define a procedure is to go into the LOGO EDITOR.

   Now you do it.

Try these procedures with the computer.

Did you predict correctly? _____

If not, what was wrong with your predictions?

Is PAT the same as PATTERN? _____

When a procedure uses another procedure, it is called a SUPERPROCEDURE.

A procedure that has been used by another procedure is called a SUBPROCEDURE.

In the above example, the superprocedure is _____

the subprocedure is _____

2. Let's try another one. Look at these procedures:

```
TO TRIANGLE
  REPEAT 3 [FD 50 RT 120]
END

TO SHAPE
  REPEAT 3 [ REPEAT 3 [FD 40 RT 120]  LT 120]
END

TO SHAPING
  REPEAT 3 [TRIANGLE LT 120]
END
```

Predict what the turtle will draw before you try it out with the computer.

Draw your predictions here:

TRIANGLE SHAPE SHAPING

Did you succeed with your prediction?

_____

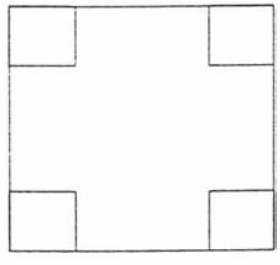
If not, what was wrong with your predictions?

Is SHAPE the same as SHAPING? _____

Which is the superprocedure? _____

Which is the subprocedure? _____

3. Look at this figure:

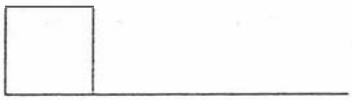


You are supposed to define a procedure that can draw this figure.

You may find this figure quite complicated at the beginning.

Now, remember, if a problem is too complicated, try to break it down into smaller and simpler problems.

Look at this figure:



Can you write a procedure to draw this simpler figure? _____

Use the five steps in planning:

First, write down your plan:

Second, check through your plan step by step to make sure that your plan will work.

Third, try your plan out with the computer.

Fourth, if something goes wrong, make sure you can DEBUG (find out the mistakes) in your plan.

Fifth, change your plan so that it will work.

Now since you can solve the smaller and simpler problem, can you use this answer (plan) to solve the original and more complicated problem?

_____

Try to think of a superprocedure to draw the original complicated figure, using the procedure that you have just defined.

Again, use the 5 steps in planning:

First, write down your plan:

Second, check your plan step by step to make sure that it will work.

Third, try your plan with the computer.

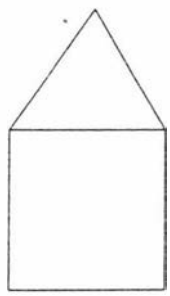
Fourth, if anything goes wrong, make sure you can debug your plan.

Fifth, change your plan so that it will work.

Did you succeed in drawing the original figure? _____

Just to remind you again, if a problem is too complicated, try to break it down into smaller and simpler problems. Then solve the smaller problems one by one. Combine these small solutions together to solve the bigger and more complicated problem.

4. Think about how you would make this house using one SUPERPROCEDURE.



Can you break it down into smaller problems? _____

How?

_____

Can you solve these smaller problems using subprocedures? _____

Write down your plans here:

Now, use the 5 steps in planning to make sure that your plans work.

After you have done that, can you combine these answers (plans) to form a bigger plan to draw the house? _____

Write down your plan here:

Check through your plan to make sure that it will work with the computer.

Try your plan out with the computer.

If something goes wrong, what are they?

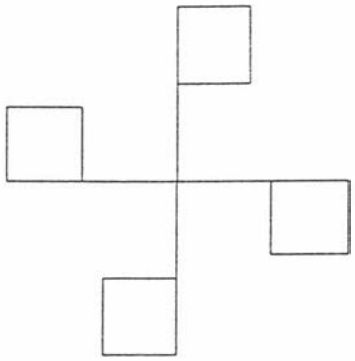
Can you fix the plan so that it will work? _____

What is your new plan?

5. EXERCISE

Write down in your own words how you would solve a big and complicated problem

Use the method you have learned today to draw this design using a SUPERPROCEDURE.



How do you break it down?

What are your plans for the smaller and simpler problems?

Have you checked that your smaller plans would work?

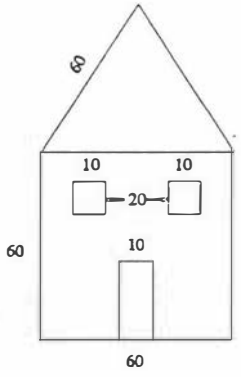
If any one of them doesn't work out as you have predicted, find out the error (DEBUG) and try them again.

PROCESS-ORIENTED GROUP

SESSION 11

BUILDING BLOCKS AND STRUCTURE DIAGRAMS

1. Look at this house below:



Can you tell the building blocks for this house? _____

Write down the building blocks here:

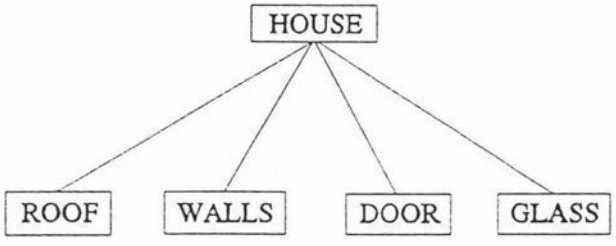
(a) _____

(b) _____

(c) _____

(d) _____

The following is a **structure diagram** for the drawing of the building blocks above:



When you try to solve a complicated problem, it is useful if you could break down this problem into smaller subproblems and then solve these simpler, smaller subproblems one by one.

A structure diagram is a very useful tool to help you to see how a complicated problem can be broken down into smaller subproblems.

Now use the 5 steps of planning (it is very important that you do that) that you have learnt, write procedures that can draw each of the building blocks above, i.e. ROOF, WALLS, DOOR, and GLASS.

TO ROOF TO WALLS TO DOOR TO GLASS

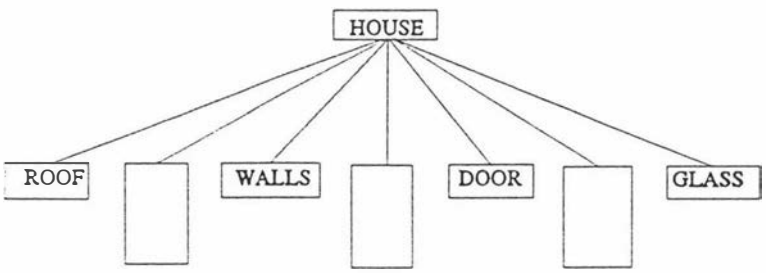
Try out these building blocks one by one to make sure that each one of them works.

Now write a superprocedure called HOUSE which combines these subprocedures together.

The important thing you need to remember is the position of the turtle after each building block is drawn.

Again, you must use the 5 steps of planning.

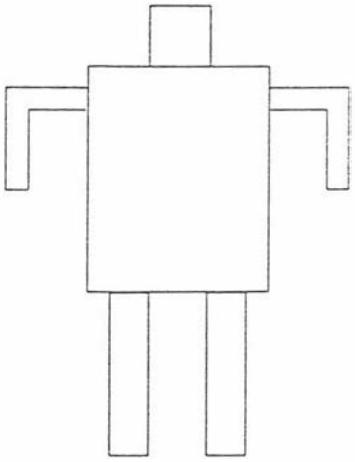
In the structure diagram below, add in the movements of the turtle that are needed to link the different subprocedures together.



Now based on the new structure diagram above, write a superprocedure HOUSE:

TO HOUSE

2. Use the same method that you have just learned above, write a superprocedure that can draw the following picture.



Remember to use the five steps of planning whenever you try to solve a problem.

If a problem is too complicated, always try to break down into simpler and smaller subproblems.

Now try drawing the different building blocks in this picture (give them a name each)

Try to draw a structure diagram that shows are you can break down this complicated problem into smaller and simpler subproblems.

Now, write subprocedures that can draw each building block (make sure you use the five steps of planning in each case)

Remember:
1. Plan your solution carefully,
2. Check through your solution step by step,
3. Try out your solution with the computer,
4. If anything goes wrong, make sure you can find out what is wrong.
5. Change your original solution, go back to step 2.

Now, write a superprocedure that can draw the whole picture (again, you must use the five steps of planning to make sure that your superprocedure will work)

Also, draw your new structure diagram that shows the connecting movements of the turtle between each subprocedure.

LOGO - CONTENT ORIENTED GROUP

SESSION 1

MEET THE TURTLE

1. Drawing in LOGO

   To set up the screen for drawing,
   Type: **SHOWTURTLE**
   and then press Enter.

   If you make any typing mistake, use the **Backspace** key to erase the mistake and type again.

   Can you see the little triangle in the centre of the screen? That is the **TURTLE.**

   You can also hide the turtle, type:

   HIDETURTLE

   Now show the turtle again by typing **SHOWTURTLE.**

2. Basic Turtle Commands

   In order to ask the turtle to draw, you would need to know the basic commands.

   There are four basic commands to move the turtle. the commands

   FORWARD and

   BACK

   Make the turtle move in the direction it is pointing.

   You will have to tell the turtle the number of steps to move forward or back though. For example, you will need to type something like

   FORWARD 40 or

   BACK 50

   The other two commands

LEFT and

RIGHT

make the turtle turn in either the left or the right direction. Again, you will need to tell the turtle the extent to turn by typing a number like

LEFT 90 or

RIGHT 150

Now, can you draw something using these four commands.

Remember, leave a space between the command and the number. Press the **Enter** key after the each command.

Practice moving around the screen using these commands:

FORWARD 50
RIGHT 90
FORWARD 30
RIGHT 60
FORWARD 60
LEFT 100
BACK 80
LEFT 50
FORWARD 60

3. After you have drawn quite a bit of things on the screen, you may like to clear up the screen and start from scratch again. The command

CLEARSCREEN

clears the screen and places the turtle in the centre of the screen.

Every time you type **CLEARSCREEN**, the screen is cleared and the turtle is brought back to the centre of the screen, which is the home of the turtle. The command

HOME

always brings the turtle back to its home no matter where the turtle is.

Now type:

HOME

CLEARSCREEN

You can see that the turtle went back to its home first, but the drawings remain. After you have typed the **CLEARSCREEN** command, the drawings were cleared too.

4. So far, you have learned the commands

FORWARD
BACK
LEFT
RIGHT
CLEARSCREEN
HOME

How about practising the following exercises:

Type:

CLEARSCREEN
FORWARD 50
RIGHT 150
FORWARD 50

Now type:

CLEARSCREEN FORWARD 50 RIGHT 150 FORWARD 50

Do you get the same picture?

Which way of typing commands do you like better?

One command at a time?

or

More than one command at a time?

Why do you like that way better?

Let's have some more practices with the commands you have learned today:

Type:

CLEARSCREEN
FORWARD 20
RIGHT 60
FORWARD 20
LEFT 60
FORWARD 20

CLEARSCREEN
FORWARD 60
LEFT 90
FORWARD 60
HOME

5. Type:
   CLEARSCREEN
   FORWARD 100
   FORWARD 100
   FORWARD 100

   You can see that the turtle first disappeared from the top of the screen, and then it re-appeared from the bottom of the screen.

6. **EXERCISE**

   How about drawing some of your favourite pictures using the commands you have learned today? You can draw pictures such as a square, a rectangle, a house etc. Write down your commands on a piece of paper and try out these commands with the computer during the next lesson.

LOGO - CONTENT-ORIENTED GROUP

SESSION 4

TURTLE TRICKS AND PATTERNS

1. There are some ticks that you can ask the turtle to perform:

HIDETURTLE (HT)
SHOWTURTLE (ST)
PENUP (PU)
PENDOWN (PD)
PENERASE (PE)

a. **HIDETURTLE (HT)**

This command will hide the turtle from your view.
Type **HIDETURTLE** or **HT**
Did it hide the turtle? _____

b. **SHOWTURTLE (ST)**

This command will show the turtle if the turtle is hidden.
Type **SHOWTURTLE** or **ST**
Did the turtle appear? _____

c. **PENUP (PU)**

This command will ask the turtle to move around the screen without drawing.
Type **PENUP**
FD 50
Did the turtle draw a line? _____

d. **PENDOWN (PD)**

This command will ask the turtle to move around the screen and leave a trace as well.
Type **PENDOWN**
BK 50
Did the turtle draw a line? _____

e. **PENERASE (PE)**

This command will ask the turtle to move around the screen and erase as it moves along its track.

Type **PENERASE**
FD50
Did the turtle erase the line it just drew? _____

2. Now we are going to draw a square and then erase it bit by bit.

Type:

CS
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
FD 90
RT 90

Did you draw a square? _____

If you did not, do it again.

O.K. We are now going to erase the square by using the **PENERASE** command.

Type:

PE
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90

Did you succeed? _____

If you did not, make sure that you have typed exactly the same as the above.

3. There are some other tricks that you can teach the computer:

TEXTSCREEN	-	this command asks the computer to show the words only.
MIXEDSCREEN	-	this command asks the computer to show the words as well as the drawing.
FULLSCREEN	-	this command asks the computer to show the drawing only.

Type these commands into the computer.

Did they do what they were supposed to do? _____

4. In fact, there are shortcuts for these commands too:

instead of typing	you can press
TEXTSCREEN	**PF1**
MIXEDSCREEN	**PF2**
FULLSCREEN	**PF4**

Press these keys (at the top of the keyboard) a few times and you should be able to see the different types of displays of the screen.

5. The following is a list of commands to draw a triangle (Can you recognise these commands?)

Type these commands into the computer.

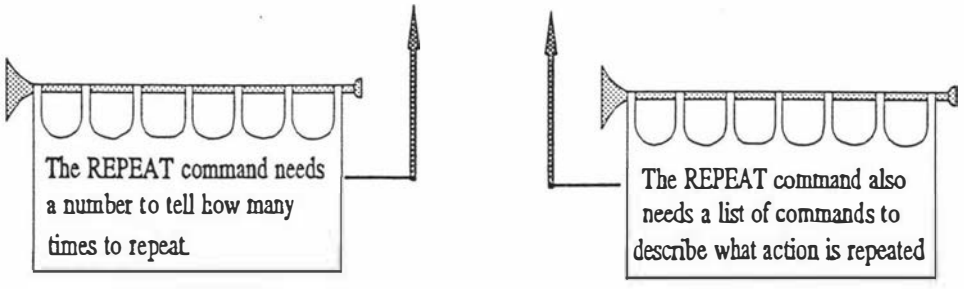
Type:

```
FD 50
RT 120
FD 50
RT 120
FD 50
RT 120
FD 50
RT 120
```

They have been repeated three times.

You can use the **REPEAT** command to draw the same triangle as a repeated list of commands.

REPEAT 3 [FD 50 RT 120]



Now clear the screen by using the **CLEARSCREEN** command.

Draw a triangle that are 40 turtle steps long using the **REPEAT** command that you have just learned.

Write down the commands that you have used here:

How about drawing another triangle that is of 30 turtle steps each side?

Yes, you can use the following commands:

REPEAT 4 [FD 30 RT 90]

Clearscreen the screen first.

Type this command into the computer.

6. Exercise

Draw a square and a hexagon that have 50 turtle steps on each side using the **REPEAT** command.

Write down your commands here.

LOGO - CONTENT-ORIENTED GROUP

SESSION 8

SUPERPROCEDURES AND SUBPROCEDURES

1. You have learned to use the **REPEAT** command, e.g.

   REPEAT 6 [FD 40 LT 60]

   In order to use the REPEAT command, you have to tell the turtle two things.

   First, the number of the times to REPEAT.

   Second, what action or actions to REPEAT.

   Do you know that you can actually REPEAT another REPEAT command or a procedure.

   Define these procedures with the computer (remember, the best way to define a procedure is to go into the LOGO EDITOR).

```
TO TRIANGLE
  REPEAT 3 [FD 50 RT 120]
END

TO SHAPE
  REPEAT 3 [REPEAT 3 [FD 40 RT 120] LT 120]
END

TO SHAPING
  REPEAT 3 [TRIANGLE LT 120]
END
```

   Ask the computer to draw these procedures.

   Does SHAPE draw the same picture as SHAPING does? _____

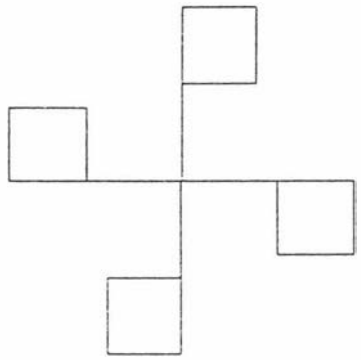
   SHAPING is called a **SUPERPROCEDURE** because it uses another procedure.

   TRIANGLE is called a **SUBPROCEDURE** because it is used by another procedure.

2. This procedure will draw a flag:

```
TO FLAG
FD 80
RT 90
FD 20
RT 90
FD 20
RT 90
FD 20
RT 90
BK 60
END
```

Type it into the computer to make sure that it can draw a flag. Can you write a superprocedure WINDMILL, using FLAG as a subprocedure to draw a windmill like the following figure.



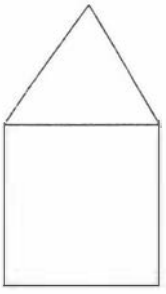
Write down your answer here:

```
TO WINDMILL
```

So you see, since WINDMILL uses another procedure in itself, it is called a SUPERPROCEDURE.

And since FLAG is used by another procedure, it is called a SUBPROCEDURE.

3. In fact, SUPERPROCEDURE can use more than one SUBPROCEDURE in itself. Look at the HOUSE below,



Now write a SUPERPROCEDURE called HOUSE which contains two SUBPROCEDURES - ROOF and WALL.

First of all, you have to write the two subprocedures ROOF and WALL. Test them out with the computer to make sure that they can draw the roof and the wall.

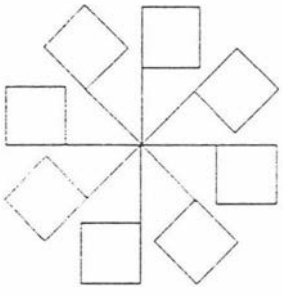
Then write a superprocedure HOUSE which uses these two subprocedures.

Write down your answer here:

TO ROOF TO WALL **TO HOUSE**

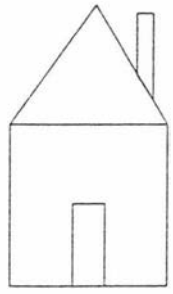
4. **EXERCISE**

a. Use the FLAG or WINDMILL procedures to write a superprocedures PINWHEEL that can draw a pinwheel like the following:



Write your answer here:

b. Based on the procedures ROOF, WALL and HOUSE, can you add a door and a chimney (using procedures DOOR and CHIMNEY) to the house like the following:



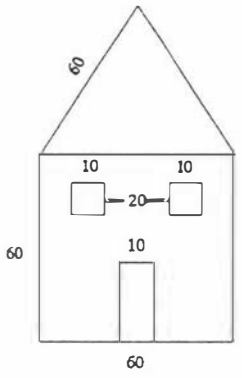
Write down your answer here:

LOGO - CONTENT-ORIENTED GROUP

SESSION 11

BUILDING BLOCKS AND STRUCTURE DIAGRAM

1. Look at the house below:



These are the building blocks for the house:

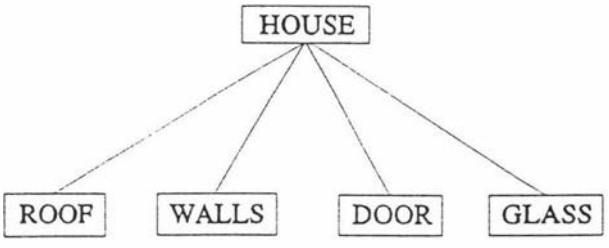
a. ROOF

b. WALLS

c. DOOR

d. GLASS

The following is a structure diagram for the drawing of the building blocks above:



A **structure diagram** will help us to see how a complicated picture can be divided into smaller ones.

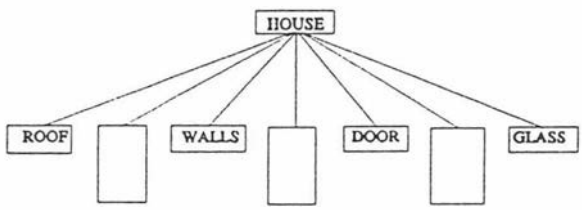
Now, you write procedures that can draw each of the building blocks:

TO ROOF **TO WALLS** **TO DOOR** **TO GLASS**

Now write a superprocedure called **HOUSE** combines these subprocedures together in order to draw the house.

The important thing you need to remember is the position of the turtle after each building block is drawn.

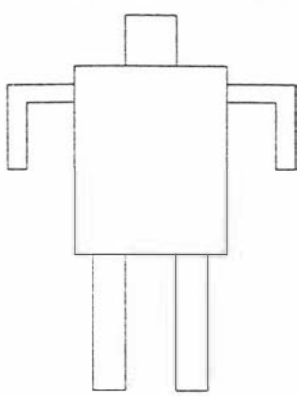
In the structure diagram below, add in the movements of the turtle that are needed to link the different subprocedures together.



Now based on the new structure diagram above, write a superprocedure HOUSE.

HOUSE

3. Use the same method that you have just learned above, write a superprocedure that can draw the following picture.



Don't forget to draw a structure diagram first.

These are the building blocks:

a. **HEAD**
b. **BODY**
c. **LEFTARM**
d. **RIGHTARM**
e. **LEG**

Now you write subprocedures that can draw each building block:

TO HEAD **TO BODY** **TO LEFTARM** **TO RIGHTARM** **TO LEG**

Now write a superprocedure **ROBOT** that can draw this robot.

TO ROBOT

BASIC GROUP

SESSION 1

BASIC BASIC

1. There are many things that a computer can do. For example, you can use the computer as a calculator to help you do some arithmetic. Try typing the following. You must press **Enter** after you have finished typing each line in order to send the message to the computer.
Write down the answer given by the computer.

PRINT 3 + 5 _____

PRINT 4 * 3 _____

PRINT 64 / 4 _____

PRINT 260 - 161 _____

See, the computer can actually help you to do some complicated arithmetic.

Find the answers for the following problems. Write down the answers next to the question.
Remember, you would need to use the PRINT command in front of each problem.

160 + 231 _____

23 * 31 _____

72 - 25 _____

121 / 11 _____

Can you do them? How about doing some arithmetic of your own. You can ask the computer to do some very complicated arithmetic, e.g.

1234567 + 9876543
1024 / 64 etc.

Write down your problems and answers in the space below.

_____
_____
_____
_____

2. Besides helping you to do arithmetic, the command PRINT can also print a message on the screen. Type:

PRINT "I AM THE GREATEST NEW ZEALAND HERO."

What happened?

Now type:

PRINT "3 + 5"

What happened?

Did the computer do the arithmetic for you?
Is the computer on strike?
Not really.

Let's try again. Type:

PRINT 7 + 12

PRINT "7 + 12"

Remember, the computer will print exactly what you type between speech marks.

How about typing some of your favourite messages?
Write them down here before you type.

3. Now you probably have a lot of things on the screen.
The screen is like a blackboard, you can actually wipe the things off the screen. Type:

CLS

The computer wipes everything off the screen when you type CLS.
Type a few more messages and then type CLS. Did it work?

4. You can also ask the computer to print the problem and then the answer.
For example, you can type the problem between the speech marks and then just the problem,

PRINT "5 + 16 = " 5 + 16

PRINT "182 / 13 = " 182 / 13

PRINT "The sum of five and six is " 5 + 6

Try these few examples to see if they work.
Also, try to do 5 more problems of your own in this way. Record your problems and answers on the back of this page.

5. Exercise

Write down some more arithmetic problems for the computer to solve for you. Use the computer to find out the answers next week. Record your problems on the back of this page.

BASIC GROUP

SESSION 4

MORE PRINTING

1. You have learned the **GOTO** and the **END** command during the last session. Now there is a challenge for you. Look at the following program:

   20 ?"CAT"
   40 ?"DOG"

   Since dogs and cats used to right, we certainly don't want them to be printed together.

   Can you find a way that only one of them is printed without removing any of the statements or changing the line numbers of these two statements.

   (Hint: you may use the **GOTO** command.)

   How? Record your answers here:

   _____
   _____
   _____
   _____
   _____

2. Now that you have written a program, give a title to the program so that when you look at this program in two months, you will know what this program is about. The rule is the computer is not allowed to print the title when the program is **RUN**ed.

   Record your answer:

   _____

3. There are two ways that you can control the computer to output (print sentences).

   Type:

   ?"Close?;"together"

   What
   happens?_____

Type: **?"Spreading","out"**

What happens?_____

So now you know that you can use ; and , to control the computer to output (print statements).

; will tell the computer to print sentences on the same line close to each other, and

, will tell the computer to spread the output (sentences) on the screen with some spaces separating them.

Type:

```
?2 + 2, 3 + 3, 4 + 4
?2 + 2; 3 + 3; 4 + 4
?"Ready", "Set", "Go"
?"Ready"; "Set"; "Go"
```

Can you see the difference now?_____

Use some output of your own and record your commands here:

_____

_____

_____

_____

_____

4. You can print some patterns on the screen too.
Type (remember to count the spaces between the X's):

```
NEW
10 HOME
20?"X   X"
30?" X X "
40?"  X  "
50?" X X "
60?"X   X"
70END
RUN
```

You should see a big X on the screen.

You can print any letter or word that you want.

You must remember to count the number of spaces.

Now try printing the patterns for **H** and **GO**.
Record your answers here.

5. EXERCISE

You can also draw pictures with PRINT statements.

Type: (remember to count the spaces between the numbers and letters)

```
NEW
110 CLS
120 ?"     88888     "
130 ?"    8     8    "
140 ?"   8       8   "
150 ?" 8         8  "
160 ?"8    0   0    8"
170 ?"8             8"
180 ?"8      X      8"
190 ?"8   *     *   8"
200 ?"8    *   *    8"
210 ?" 8    ***    8 "
220 ?"   8       8   "
230 ?"    8     8    "
240 ?"     88888     "
250 END
END
```

Did you see a picture?

How about drawing a picture of your own?

Record your answer here:

BASIC GROUP

SESSION 8

LOOPS

1. You are going to learn three things today:

   LOOP One or more instructions that are repeated.

   FOR First statement in a loop.

   NEXT Last statement in a loop.

2. A **LOOP** is a set of one or more statements.
   These instructions can be repeated as many times as you like.

   You can make a loop by typing two new words - **FOR** AND **NEXT** - in your set of instructions.

   Type:

```
NEW
10 FOR N = 1 TO 10
20 ? "HELLO"
30 NEXT N
RUN
```

   How many times is HELLO printed on the screen?

   _____

   Can you get GOODBYE to print ten times? _____
   Record your commands here:

   _____
   _____
   _____

3. Try a new loop.
Type:

```
NEW
10 FOR N = 1 TO 5
20 ?"HOKOWHITU"
30 NEXT N
RUN
```

How many times was the word "HOKOWHITU" printed? _____

N stands for the number of times that the computer goes through the loop. How many times does the program tell the computer to print the word "HOKOWHITU"? _____

Try this program.
Type:

```
NEW
5 CLS
10 FOR N = 1 TO 6
20 ? N
30 NEXT N
RUN
```

What did you see on the screen? _____

You see, the computer used a loop to count from 1 to 6.

Can you make this program to count to 10 instead of 6?_____

How? _____
_____

Run this program again to make sure that it works.

5 The computer can count by any number.

The computer can count by twos instead of ones.

Type:

LIST

Now, change line 10 in the program.
Type:

10 FOR N = 2 TO 20 STEP 2
RUN

What did you
see?_____
That's right.
First, you told the computer to start counting at 2 and stop by 20.
Second, you told the computer to count by twos by the command **STEP 2.**

Look at the whole program again.

LIST

Do you understand?_____

6. You can tell the computer to count by fives, start with 5 and finish with 30. All you have to do is change one line.

Type:

10 FOR N = 5 TO 30 STEP 5
RUN

Did it work?_____

Now, try counting by eights from 8 to 80.

Record your commands here:

_____
_____
_____
_____
_____
_____

7. Besides counting by twos, fives, eights, or any number, the computer can also count **backwards.**

Type:

10 FOR N = 10 TO 1 STEP -1
LIST

What did you see on the screen?

_____
_____
_____
_____

Draw a box around the loop.

The **FOR** statement begins the loop.
It shows the starting number, ending number, and the step.

FOR N = 10 TO 1 STEP -1

starting number — ending number

Where does the computer start counting?_____

Where does the computer stop counting?_____

Before you **RUN** the program, add a line
Type:

40 ?"BLAST OFF"

8. Write a program so that your name is printed eight times.
Type **NEW** first before you start writing your program.

Record your commands here:

_____
_____
_____
_____
_____
_____

Save this program on your disk.
What is the name of your program?_____

Draw a flow chart for this program on the next page:

9. Write another program so that the computer can count by sevens from 7 to 70.
Type **NEW** before you write your program.

Record your commands here:

Save this program on your disk and then draw a flow chart of the program:

10. Write a program so that the computer can count by twelves from 240 to 12 backwards.
Record your commands here:

BASIC GROUP

SESSION 11

DRAWING PICTURES

1. In this session, you will learn how to draw lines and pictures with the BASIC language.

   In order to draw pictures with BASIC, type:

   SCREEN 1

   The screen can be pictured in this way with 320 columns across and 200 lines down:

The command **LINE** tells the computer to draw a line from one point to the other on the screen.

Now type each of the following line followed by **Enter**:

LINE
LINE
LINE
LINE

You should see a box drawn around the screen.

We can add in the diagonals. Type:

Now draw some more lines using the same method.

If you want to start a new picture, just type **CLS**.

2. It is always nice to add a bit of colour to our life.
We can do the same to our screen.

Type (be aware of the American spelling, it is **COLOR**):

COLOR 1
COLOR 2
COLOR 3
COLOR 4
COLOR 5
COLOR 6
COLOR 7

In fact, you can use any numbers from 0 to 15 with the command **COLOR**.

Now add another number to COLOR. Type:

COLOR 1,0
COLOR 1,1
COLOR 2,0
COLOR 2,1

What did you see?_____

_____

3. Besides drawing vertical and horizontal lines by using the LINE command, you can actually draw some other pictures using **DRAW** command.

You always start at the centre of the screen with the DRAW command.

Let's say we want to draw a house.

When we start drawing, we probably want to draw the walls first.

Type:

DRAW "R100 D83 L100 U83"

There are 5 commands here:

DRAW asks the computer to start drawing;

R100 asks the computer to move to the right by 100 units;

D83 asks the computer to move down 83 units;

L100 asks the computer to move left 100 units;

U83 asks the computer to move up 83 units

After the walls are drawn, we are back to the top left hand corner of the "walls". So we want to draw our roof now.

First of all, we would need to turn in order to draw one side of the roof.

Type:

DRAW "TA-30"
DRAW "U83"

In order to draw the other side of the roof, type:

DRAW "TA30"
DRAW "U83"

The last thing we want to draw is a door. Type:

DRAW "TAO D83 L69 U40 R20 D40"

So the house is done!!

If you want the computer to move without drawing a line, then you add **B** in front of the commands such as BU83, or BL100 etc.

We can put all these drawings into a program so that the house can be drawn at once when the program is **RUN**.

Type this program and save it on to your own disk:

```
5 REM A PROGRAM TO DRAW A HOUSE
10 CLS
20 SCREEN 1
30 DRAW "R100 D83 L100 U83"
40 DRAW "TA-30"
50 DRAW "U83"
60 DRAW "TA30"
70 DRAW "D83"
8O DRAW "TAO D83 L60 U40 R20 D40"
90 END
```

4. Exercise

Now can you add two windows to the house that you have just drawn?

Add your commands to the program above and save it on your disk.

Write down your own commands here:

APPENDIX 3

RULE NAMING TEST

INSTRUCTIONS FOR ADMINISTRATION

1. Tell the subject:

   We are going to play four games today

2. There are four attributes on each card. They are the shape, colour, size and number of figures. Explain to the subject by pointing to the pile of cards say:

   See these cards here. They contain figures of different shapes, colour, size and number. Some of them contain circles, squares or triangles. Some of them contain blue, red or yellow figures. Some of them contain large, medium or small figures. Some of them contain one, two or three figures.

3. Hold up the first card and say slowly:

   You can tell four things from a card - the shape, colour, size and number. For example, this card contains three, small, red, triangles.

   Hold up the next card and say:

   This card contains two, medium, yellow, squares.

4. Now ask the subject to describe the 11 sample cards. Say,

   Now you describe these cards to me.

5. After the subject has finished describing the cards, say to the subject:

   Now let me tell you how to play these games.

   In each game, there is a rule between two things on the card. You have to find out this rule. I will tell you what these two things are in each game, for example, red and circles, and then you have to find out what the rule between these two things is.

   Pause 5 seconds.

   I will show you one card at a time. You will then tell me whether the rule is obeyed or not. If the rule is obeyed, you will say yes. If the rule is not obeyed, you will say no. In return, I will tell you whether your

answer is right or wrong. You will be given hints from time to time during the game.

Pause 5 seconds.

6. Say to the subject:

Now let us start playing the first game. In the first game, there is a rule between blue and square. You have to find out the rule between blue and square.

7. Re-prompt the subject by saying:

Remember, just concentrate on the rule between the two things. The first thing is blue. The second thing is square.

Let us start playing the game.

Start showing the cards for problem A one by one to the subject.
Provide the subject with the appropriate feedback by saying either right or wrong.
Record on the scoring sheets whether the answer is correct or not.
There are 40 cards to each problem. Repeat the attributes to the subject after every twenty cards by saying:

Remember, just concentrate on the rule between the two things. The first thing is blue. The second thing is square.

8. A problem comes to an end if the subject can provide 12 consecutive correct answers or 100 cards are shown (as there are only 40 cards for each problem, if the subject cannot identify the rule within 40 cards, repeat the cards until either the rule is identified or 100 cards are shown.)

9. The three other problems are:

B. circle and yellow;
C. red and triangle;
D. square and one figure.

10. When problem A is finished, proceed with problems B, C and D. Repeat the procedures for problem A.

11. When all the problems are finished, say to the subject:

Thank you for playing the games.

APPENDIX 4

1.
2.
3.
4
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.

APPENDIX 5

TOWER OF HANOI

INSTRUCTIONS FOR ADMINISTRATION

1. Tell the subject:

Today we are going to play a game.

Point to the disks.

See these disks here, they are of different sizes.

Point to the corresponding pegs.

The aim of this game is to move all these disks from peg number 1 to either peg number two or peg number three.

There are two rules that you must follow:

The first rule, you can only move one disk at a time.

Demonstrate to the subject by moving the smallest disk from peg number 1 to peg number two.

The second rule, you cannot place a bigger disk on top of a smaller one.

Demonstrate to the subject by moving the second smallest disk on top of the smallest disk which is at peg number 2.

Move all the disk back to peg number 1.

Ask the subject if the two rules are understood.

Do you understand these two rules?

If yes, proceed to the next step.
If not, repeat the two rules.

2. Tell the subject:

Now I shall show you how to play this game with two disks.

Take away all the disks from peg number 1 except the two smallest ones. Demonstrate to the subject how to move two disks from peg number one to peg number two. Follow the steps below and do it slowly and clearly:

i move the smallest disk to peg number 3.
ii move the second smallest disk to peg number 2.
iii move the smallest disk from peg number 1.

3. Tell the subject:

Now you do it. Go ahead and play the game with two disks.

4. When the subject is moving the disks, record on the scoring sheets the detail movements made by the subject.

5. If the subject fails to solve a two disk problem, re-demonstrate the solution and ask subject to try again. If the subject fails after three attempts, score 0 for all problems and go to the last step.

6. When the subject has finished the two disk problem, place the three smallest disks at page number 1.

Tell the subject:

Now, go ahead and play the game with three disks.

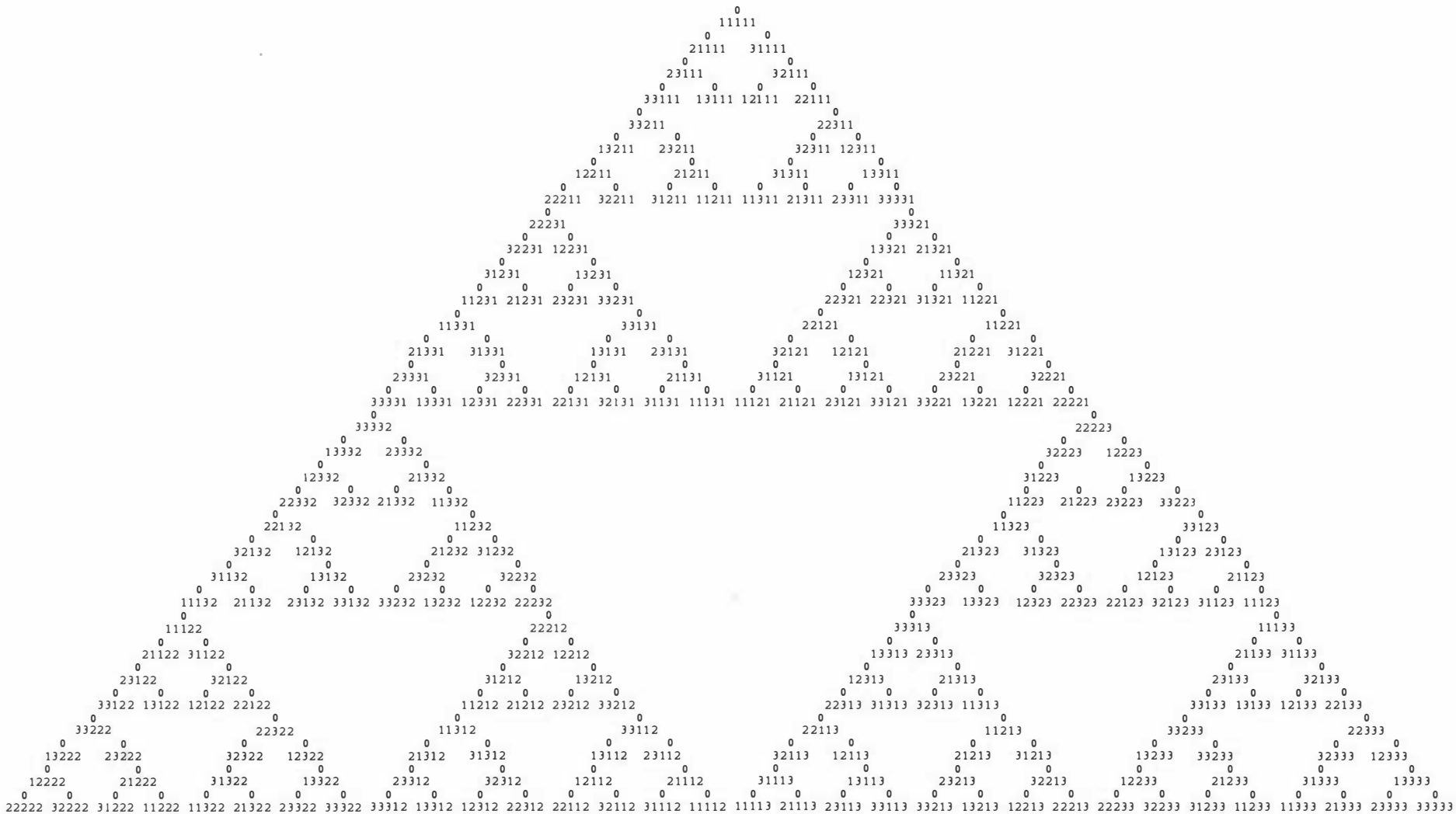
Record their movements on the scoring sheets as above.

7. Continue the test with the four and five disks problem.

8. When the subject has finished all the problems, tell the subject:

Thank you for playing this game.

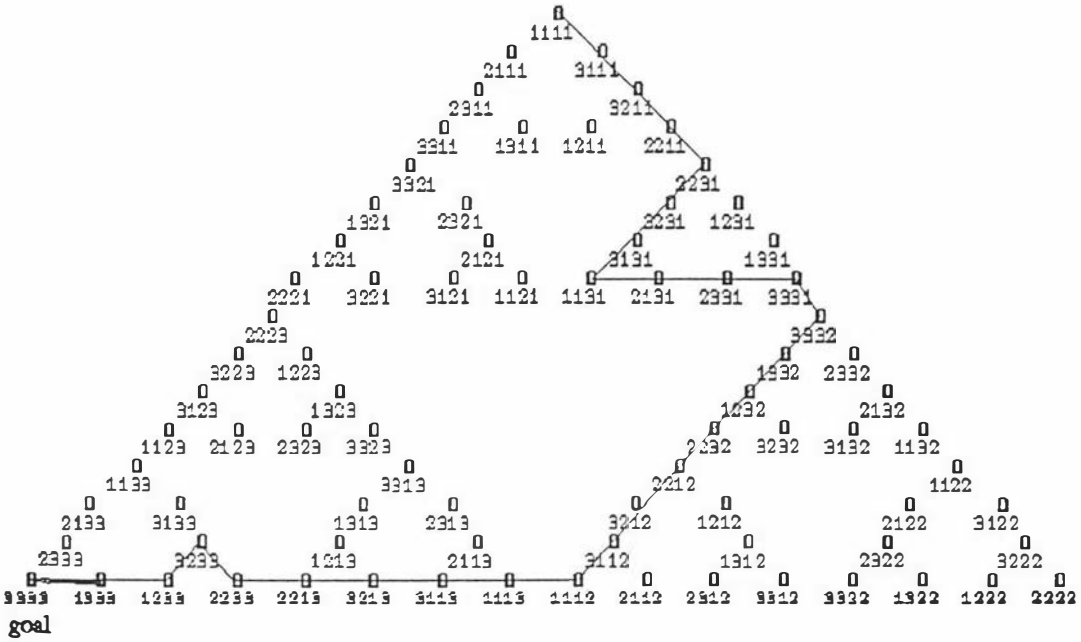
APPENDIX 6



APPENDIX 7

TOWER OF HANOI

Calculation of solving sub-problems



Two disk level:	1 0 1 1 1 0
Three disk level:	0 1 0
Two disk level score:	4/6 = .67
Three disk level score:	1/3 = .33

APPENDIX 8

COMPUTING RESEARCH PROJECT

1986

Please answer the following questions carefully:

1. Name: _____

2. Birthday: _____

3. Age at last birthday: _____

4. Address: _____
   _____

5. What kind of work does your father do? _____

6. What kind of work does your mother do? _____

7. Have you ever used a computer? (answer yes or no)

   _____

   If the answer to this question is no, then you have finished answering this questionnaire.

8. How often do you use a computer?

   _____ Everyday
   _____ times a week
   _____ times a month
   _____ times a year

9. What experience do you have with computers?

	None	A little	Quite a bit	A lot
Playing games	_____	_____	_____	_____
Programming	_____	_____	_____	_____
Others	_____	_____	_____	_____

10. What computer languages have you used?

	None	A little	Quite a bit	A lot
LOGO	_____	_____	_____	_____
BASIC	_____	_____	_____	_____
PASCAL	_____	_____	_____	_____
Others	_____	_____	_____	_____

11. What kind of computers have you used?

	None	A little	Quite a bit	A lot
Apple				
Commodore				
Spectrum				
Atari				
IBM				
BBC				
Others				

12. What kind of computers(s) do you use most often?

_____

13. What computer language(s) do you know well?

_____

APPENDIX 9

TEACHER OBSERVATION

NAME OF TEACHER: _____
DATE: _____
TIME: _____

Time	Episode	Name of subject/Content of interaction
1		
2		
3		
4		
5		
6		
7		
8		

APPENDIX 10

GROUP OBSERVATION

GROUP: _____

DATE: _____

TIME: _____

Time	Types of group interaction
1	
2	
3	
4	
5	
6	
7	
8	

APPENDIX 11

INDIVIDUAL OBSERVATION

NAME OF STUDENT: _____
DATE: _____
TIME: _____

Time	Episode	Types of interaction/Other subjects involved
1		
2		
3		
4		
5		
6		
7		
8		

APPENDIX 12

A Typical LOGO Lesson

1. For the first few minutes of the session, the teacher introduced the key concepts of the worksheets, eg, defining procedures. Students were first asked to explore what happened when they typed SQUARE, for example. The teacher would help the students' exploration by asking questions such as, "What does the Turtle do when you type SQUARE?" Based on the Turtle's response, "I DO NOT KNOW HOW TO SQUARE," students were then guided in how to teach the Turtle a new word. Predictions were made on the outcome of these new words, eg, "Predict what commands need to be included in a procedure to draw a circle," and further explorations were encouraged, eg, "How about teaching the Turtle a new word to draw a triangle?" Once students were familiar with the key concepts of the lesson, they were able to progress with their individual work.

2. For the next hour or so, students were asked to work individually, alternating between 15 minutes on the computer and 15 minutes off. While off the computer they followed the general problem solving model by spending their time predicting and planning their worksheet activities, either individually or in groups. Their plans and prediction were then tried out when working on the computer. If their plans did not work, they were encouraged to ask themselves where they went wrong, and then to attempt to find their mistakes and change their plans, then to try them out at the computer again. During this time the teacher was there to facilitate and monitor the progress of the students' learning by using questioning techniques. The questions and suggestions impelled the students to develop their thinking skills and problem solving processes. Some examples are: "Estimate how many degrees the Turtle needs to turn by walking it out yourself," "Experiment with those commands on the computer," " Think through your steps carefully when you are planning," "Think it ahead before you actually try it out," "Where do you think you have gone wrong?" "What do these commands actually do?" "Check through your plan carefully."

3. At the end of the session, the teacher provided opportunities for the students to discuss and share their ideas and any problems that had arisen during the session. Such problems formed the basis for class discussions on how they could be solved. Students were also encouraged to focus specifically on the processes they used to solve the problems, eg, "I predicted the commands by walking them out first. This way I found out where I had gone wrong in my planning." The teacher also provided opportunities for programming challenges that specifically related to the session's main concepts, eg, "Let's see who can predict the outcome of this procedure," "Draw your predictions on the whiteboard, so that we can all compare the outcomes," "Now let's discuss how we made these predictions," "Where have some of us gone wrong?" etc. These types of challenges were used to actively encourage students in shifting from a concrete method (walking out or drawing the Turtle's path) to an abstract method (thinking through in their own minds) of problem solving.