THE DESIGN AND IMPLEMENTATION OF A STRUCTURED PROGRAMMING LANGUAGE WITH FEW ARBITRARY RESTRICTIONS - THE COMPILATION PHASE

A dissertation presented by

NOLA M. SIMPSON

In partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

at

Massey University

June, 1973

ABSTRACT

This thesis outlines the design and implementation of a structured teaching language with particular emphasis on the compilation phase. This language, which has been called MUSSEL (Massey University Structured Student Language) is intended for instruction of first year Computer Science students. In this respect it is a language which is as free as possible from arbitrary syntactic restrictions and is in a form that the student should find both easy and natural to write, with a minimum incidence of programming errors.

It is evident that the language a student uses profoundly influences the way he develops his techniques and habits of construction of algorithms. MUSSEL has been designed with this influence in mind and has been deliberately designed as a structured language i.e. the language reflects the basic structure of programs, so that algorithms can be refined and expressed naturally in it. In this sense MUSSEL aims to teach the student programming as a constructive intellectual discipline rather than just as a tool to communicate with the computer.

MUSSEL has been implemented as an interpretive system i.e. during the compilation phase the source language is translated into an intermediate language, which, is then interpreted. The reasons influencing this type of implementation are the need in a teaching language to minimise compile time and to maximise diagnostics, both at compile-time and at execution-time.

ACKNOWLEDGEMENTS

In being able to present this thesis, I wish to offer sincere thanks to the following people.

To my supervisors, Professor G. Tate and R.W. Doran for their help, guidance and forbearance.

To Peter Gibbons, with whom I worked for many long hours in discussing the design and implementation of MUSSEL.

To the Mathematics Secretary, Mrs. I.M. Cornish for her kindness, tolerance and competence in typing the script.

Finally to the Staff of both the Computer Unit and Mathematics Department for their constant reminders during the past six months, that my thesis was due for completion.

I would also like to acknowledge the fact that changing my name has not been exactly conducive to completing this thesis within the respective time limit.

Massey University
June, 1973.

Nola M. Simpson

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION - A STRUCTURED LANGUAGE FOR STRUCTURED PROGRAMMING.

It has been common practice when introducing programming to students for the first time, to furnish them with the details and idiosyncrasies of a given programming language, mostly FORTRAN or BASIC, and then by means of examples attempt to teach students how to write programs in the particular language.

As a result it is not surprising if the student after this course is hardly an effective programmer, and often has not even been taught a basis on which to further develop his capabilities. After all, he has only been subjected to what Wirth [6] has called a "computer appreciation course" or maybe a "disappreciation course"

It should be recognised that programming is a more difficult subject than is normally estimated and that the real art of programming is firstly construction of algorithms, and this has in the past normally been left to the intuition of the student alone. Increasing the complexity of the problems used as examples, merely increases the demands and strain on the students intuition, but not further his capability to control and check his intuition by reason - he may even be misguided into believing that the essence of programming is concocting ad-hoc ideas and tricks to solve given problems.

The only effective solution to this dilemma is to extend the education in programming to that of construction of algorithms.

Up to the present time there exists neither an established theory on this subject or any well-based methods, as these are still in the process of development.

In an attempt to overcome this failure in programming education Dijkstra [3] and later Wirth [6] have proposed a structured approach to programming and algorithm development. Their method is in effect a process of analysing an algorithm into smaller and smaller independent component parts until it has been broken down into basic elementary instructions. Dijkstra used this approach initially to prove the correctness of programming, but has since advocated its use as a general method and maintains programming efficiency is stepped up by a significant factor.

Wirth [8] developed his own language PASCAL, and in a paper [6] describes how he uses it in the structured approach to programming. PASCAL is a very highly structured language the sequencing being expressed without goto statements and it also deals with structured data. In his paper Wirth shows how to develop in parallel a program and the structure of the data on which it is to operate.

Using his approach, program development can be achieved provided a teacher had at his disposal any suitable structured language, but, unfortunately many teachers do not have such a language available, or if they do have one e.g. PL/1, Algol, it may not be possible to use it because the compiler is too slow or the error diagnostics are bad for a beginning student.

Up to the present time, this has been the situation at Massey University where the only student language available has been FORTRAN. FORTRAN of course is an unstructured language i.e. the sequence of execution of basic instructions by the computer has to be laid out explicitly with goto or similar statements. For describing programs written in an unstructured language such as FORTRAN the flow diagram has been widely used as it is also explicitly structured and so is very suitable for this purpose.

In attempting to teach structured programming to students Doran [10], [11], [12] has devised what he has called "Structure Diagrams" , as a notation to replace flow diagrams for the description of algorithms. The basic idea behind producing a structure diagram is to successively divide an algorithm into smaller and smaller subalgorithms. The notation used, follows the structure of Wirth's language PASCAL, and in fact closely resembles the parse trees in his language. The process of successively dividing an algorithm has been called by Dijkstra "refining an algorithm". In a structure diagram this refinement of an algorithm is represented pictorially by a "tree", the nodes of which are boxes essentially containing statements of subalgorithms. Since the algorithms represented may be recursive these diagrams are really 'lists' not 'trees'.

This method of formulating algorithms has proved to be very effective, but there was no language available, which was suitable or compatible with these ideas. Algol is a structured language, although not sufficiently so, and also it is not the easiest language to introduce to beginning students.

To overcome this difficulty it was decided to design a new language called MUSSEL, which would be compatible with the ideas on structured programming and yet simple enough for the average student to be able to grasp it quickly and naturally. Of course, a new language should not be developed just for the sake of novelty and existing languages should be used as a basis for development wherever possible. In this sense this has been true in the design of MUSSEL and PL/1 has been used as the basis since it was the language which met most of the requirements to a higher degree than most other languages.

To summarise, MUSSEL has been designed and implemented with the following principal objectives in view:

(i) To design a language with the structure and syntax to resemble very closely the form of Doran's Structure Diagrams, so that programs written in the language may be expressed in a systematic, precise and appropriate way.

(ii) To design a language which will enable programming to be taught from basic concepts which are clearly and naturally reflected in the language.

(iii) To make a language available with a syntax which is as natural as possible for a student to use, and also with as few arbitrary restrictions as possible.

(iv) To illustrate that a language with a rich set of program structuring facilities can be implemented on a small computer and be reasonably efficient for student use.

(v) To provide a framework, which due to its systematic structure, can be at a later date adapted to the various needs for which it has been designed, for example a compiler system for first year computer science students, with a large set of error checking facilities and diagnostics and the ability to gather a wide range of user statistics.

We believe that in the design of MUSSEL the first three of our objectives have been achieved, and thus we hope, using MUSSEL a teacher could shift the emphasis in teaching programming from that of learning the details of a language, to that of the construction and development of algorithms, which is where the emphasis should be. In the implementation of MUSSEL although hampered by the computer available at the present time, we trust the remaining two objectives will be achieved.