

```
/*
 * symmetry_group.h
 *
 * This file defines a general, not necessarily abelian group
 * for use in computing symmetry groups of manifolds.
 */

#ifndef _symmetry_group_
#define _symmetry_group_

#include "kernel.h"
#include "isometry.h"

/*
 * Symmetries are just Isometries from a manifold to itself.
 */

typedef Isometry      Symmetry;
typedef IsometryList  SymmetryList;

/*
 * The file SnapPea.h contains the "opaque typedef"
 *
 *      typedef struct SymmetryGroup      SymmetryGroup;
 *
 * which lets the UI declare and pass pointers to SymmetryGroups without
 * actually knowing what they are.  This file provides the kernel with
 * the actual definition.
 */

/*
 * Group elements are represented by integers, beginning with 0.
 * By convention, 0 will always be the identity element.
 */

struct SymmetryGroup
{
    /*
     * The order of the group.
     */
    int      order;

    /*
     * We want to keep the actual Symmetries around
     * to compute their fixed point sets, their action
     * on the cusps, etc.
     */
    SymmetryList  *symmetry_list;

    /*
     * product[][] is the multiplication table for the group.
     * That is, product[i][j] is the product of elements i and j.
     *
     * We compose Symmetries from right to left, so the product BA
     * is what you get by first doing Symmetry A, then Symmetry B.
     */
    int      **product;

    /*
     * order_of_element[i] is the order of element i.
     */
    int      *order_of_element;

    /*
     * inverse[i] is the inverse of element i.
     */
    int      *inverse;

    /*
     * Is this a cyclic group?
     */
}
```

```

    * If so the elements will be ordered as consecutive
    * powers of a generator.
    */
Boolean      is_cyclic;

/*
 * Is this a dihedral group?
 *
 * If so, the elements will be ordered as
 * I, R, R^2, . . . , R^(n-1), F, FR, . . . , FR^(n-1).
 */
Boolean      is_dihedral;

/*
 * Is this a spherical triangle group?
 * That is, is it one of the groups
 *
 *          group                (p, q, r)
 *
 * (binary) dihedral            (2, 2, n)
 * (binary) tetrahedral         (2, 3, 3)
 * (binary) octahedral          (2, 3, 4)
 * (binary) icosahedral         (2, 3, 5)
 *
 * If so, p, q and r will record the angles of the triangle
 * (pi/p, pi/q and pi/r), and is_binary_group will record
 * whether it's the binary polyhedral group as opposed to
 * the plain polyhedral group.
 *
 * When is_polyhedral is FALSE, p, q, r and is_binary_group
 * are undefined.
 */
Boolean      is_polyhedral;
Boolean      is_binary_group;
int          p,
             q,
             r;

/*
 * Is this the symmetric group S5?
 * (Eventually I'll write a more general treatment of symmetric
 * and alternating groups, but I want to get this much in place
 * before the Georgia conference so SnapPea can (I hope) recognize
 * the symmetry group of the totally symmetric 5-cusp manifold.)
 */
Boolean      is_S5;

/*
 * Is this an abelian group?
 *
 * If so, the elements will be ordered in a natural way,
 * and abelian_description will point to a description
 * of the group.
 *
 * If not, abelian_description will be set to NULL.
 */
Boolean      is_abelian;
AbelianGroup *abelian_description;

/*
 * Is this group a nonabelian, nontrivial direct product?
 *
 * If so, factor[0] and factor[1] will point to the two
 * factors. Of course, each factor may itself be a nontrivial
 * direct product, and so on, leading to a tree structure
 * for the factorization.
 *
 * The symmetry_list field is defined only at the root level,
 * but all other fields are defined at all nodes.
 *
 * If the group is not a nonabelian, nontrivial direct product,
 * factor[0] and factor[1] will be set to NULL.
 */
Boolean      is_direct_product;

```

```
SymmetryGroup    *factor[2];  
  
};  
  
#endif
```