This chapter examines structures, unions, enumerations and typedefs which allow a programmer to create new data types. The ability to create new types is an important and very powerful feature of C++ and releases a programmer from being restricted to the integral types offered by the language. Structures enable a programmer to form a collection of similar or different data types into a single user-defined data type, unlike arrays, which are collections of a single data type. The data members of a structure are, by default, publicly accessible. User-defined structures integrate exactly into the language. In other words, there is a single syntax for the definition and manipulation of objects of both integral C++ and user-defined types.

A union is similar to a structure except that a union can hold only one of its data members at any given time. Union data members share the same location in memory. An enumerated data type is a collection of named members which have equivalent integer values. A typedef is in fact not a new data type but merely a synonym for an existing type.

8.1 Structures

A structure is a user-defined data type that is a collection of similar or dissimilar data items or members.

The previous chapter illustrated the array, which enables us to form an aggregate of identical data items. If we are required to form a collection of different data types, we cannot use an array. The C++ keyword struct enables a programmer to create new structure data types which encompass a collection of similar or different data types. The ability to develop new user-defined data types means that a programmer is not restricted to the integral data types offered by the C++ language. The following program illustrates a Person structure:

```cpp
// struct.cpp
// illustrates structures
```
#include <iostream.h>  // C++ I/O

// declare a structure 'Person'
struct Person
{
    char name[20];  // name
    int age;  // yr
    float height;  // m
    float mass;  // kg
};

void main()
{
    // define a structure variable 'person'
    Person person;

    // prompt user for data and assign
    // values to struct Person members
    cout << "enter your forename: " ; cin >> person.name ;
    cout << "enter your age: " ; cin >> person.age ;
    cout << "enter your height: " ; cin >> person.height ;
    cout << "enter your mass: " ; cin >> person.mass ;

    // O/P
    cout << endl << "hello "
        << person.name  << ", you are: "
        << person.age  << " years old, "
        << person.height  << "m tall and "
        << person.mass  << "kg in mass" ;

    // define and initialise a Person variable 'harry'
    Person harry = { "Harry", 25, 1.85, 80 } ;

    // O/P Harry:
    cout << endl << "hello "
        << harry.name  << ", you are: "
        << harry.age  << " years old, "
        << harry.height  << "m tall and "
        << harry.mass  << "kg in mass" ;
}

Some user interaction with the above program is:

enter your forename: Tom
enter your age: 45
enter your height: 1.8
enter your mass: 74

hello Tom, you are: 45 years old, 1.8m tall and 74kg in mass
hello Harry, you are 25 years old, 1.85m tall and 80kg in mass