**Unions**

Just like Structures, the union is a user-defined data type.  They provide better memory management than structures. All the members in the unions share the same memory location.

The union is a data type that allows different data belonging to different data types to be stored in the same memory locations. One of the advantages of using a union over structures is that it provides an efficient way of reusing the memory location, as only one of its members can be accessed at a time. A union is used in the same way we declare and use a structure. The difference lies just in the way memory is allocated to their members.

**Creating a Union element**

We use the union keyword to define the union.

The syntax for defining a union is,

union union\_name

{

//union\_elements

} union\_variable;

Here’s one example of how a union is defined and used in main as a user-defined data type.

#include <iostream>

using namespace std;

union money

{

/\* data \*/

int rice;

char car;

float pounds;

};

int main()

{

union money m1;

}

Copy

**Initialising and accessing union elements**

Different from how we used to initialise a struct in one single statement, union elements are initialised one at a time.

And also, one can access only one union element at a time. Altering one union element disturbs the value stored in other union elements.

#include <iostream>

using namespace std;

union money

{

/\* data \*/

int rice;

char car;

float pounds;

};

int main()

{

union money m1;

m1.rice = 34;

cout << m1.rice;

return 0;

}