**Structures**

Variables store only one piece of information and arrays of certain data types store the information of the same data type. Variables and arrays can handle a great variety of situations. But quite often, we have to deal with the collection of dissimilar data types. For dealing with cases where there is a requirement to store dissimilar data types, C++ provides a data type called ‘structure’. The structure is a user-defined data type that is available in C++. Structures are used to combine different types of data types, just like an array is used to combine the same type of data types.

Another way to store data of dissimilar data types would have been constructing individual arrays, but that must be unorganized. It is to keep in mind that structure elements too are always stored in contiguous memory locations.

**Features of Structs**

1. We can assign the values of a structure variable to another structure variable of the same type using the assignment operator.
2. Structure can be nested within another structure which means structures can have their members as structures themselves.
3. We can pass the structure variable to a function. We can pass the individual structure elements or the entire structure variable into the function as an argument. And functions can also return a structure variable.
4. We can have a pointer pointing to a struct just like the way we can have a pointer pointing to an int, or a pointer pointing to a char variable.

**Creating a struct element**

We use the struct keyword to define the struct.

The basic syntax for declaring a struct is,

struct structure\_name

{

//structure\_elements

} structure\_variable;

Copy

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

#include <iostream>

using namespace std;

struct employee

{

/\* data \*/

int eId;

char favChar;

int salary;

};

int main()

{

struct employee Munawar;

return 0;

}