Unit:4

Structure vs class: In C++, a structure is the same as a class except for a few differences. The most important of them is security. A Structure is not secure and cannot hide its implementation details from the end user while a class is secure and can hide its programming and designing details.

Structure: We often come around situations where we need to store a group of data whether of similar data types or non-similar data types. We have seen [Arrays in C++](https://www.geeksforgeeks.org/arrays-in-c-cpp/) which are used to store set of data of similar data types at contiguous memory locations.

Unlike Arrays, Structures in C++ are user defined data types which are used to store group of items of non-similar data types.

**What is a structure?**

A structure is a user-defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type. The ‘struct’ keyword is used to create a structure. The general syntax to create a structure is as shown below:

struct structureName{

member1;

member2;

member3;

.

.

.

memberN;

};

Structures in C++ can contain two types of members:

* **Data Member**: These members are normal C++ variables. We can create a structure with variables of different data types in C++.
* **Member Functions**: These members are normal C++ functions. Along with variables, we can also include functions inside a structure declaration.

struct Student

{

char stuName;

int stuRollNo;

int stuAge;

};

A program Explaining Structure

#include <iostream.h>

struct Person

{

int citizenship;

int age;

};

int main(void) {

struct Person p;

p.citizenship = 1;

p.age = 27;

cout << "Person citizenship: " << p.citizenship << endl;

cout << "Person age: " << p.age << endl;

return 0;

}

Output

Person citizenship: 1

Person age: 27

A program Explaining Array in Structure

#include <iostream>

struct Point {

    int x, y;

};

int main()

{

    // Create an array of structures

    struct Point arr[10];

    // Access array members

    arr[0].x = 10;

    arr[0].y = 20;

    cout << arr[0].x << " " << arr[0].y;

    return 0;

}

Output

10 20

# Access Specifiers

In C++, there are three access specifiers:

* public - members are accessible from outside the class
* private - members cannot be accessed (or viewed) from outside the class
* protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes. You will learn more about [Inheritance](https://www.w3schools.com/cpp/cpp_inheritance.asp) later.

In the following example, we demonstrate the differences between public and private members:

“Members of a class are private by default and members of a struct are public by default.”

#include <stdio.h>

class Test {

    int x; // x is private

};

int main()

{

  Test t;

  t.x = 20; // compiler error because x is private

  getchar();

  return 0;

}

Above Program fails in compilation. But when we was using Struct it was public as shown in the bellow program.

#include <stdio.h>

struct Test {

    int x; // x is public

};

int main()

{

  Test t;

  t.x = 20; // works fine because x is public

  getchar();

  return 0;

}