**Struct vs Class in C++**

In C++, a structure works the same way as a class, except for just two small differences. The most important of them is**hiding implementation details.** A structure will by default not hide its implementation details from whoever uses it in code, while a class by default hides all its implementation details and will therefore by default prevent the programmer from accessing them. The following table summarizes all of the fundamental differences.

| **Class** | **Structure** |
| --- | --- |
| Members of a class are private by default. | Members of a structure are public by default. |
| Member classes/structures of a class are private by default. | Member classes/structures of a structure are public by default. |
| It is declared using the **class** keyword. | It is declared using the **struct**keyword. |
| It is normally used for data abstraction and further inheritance. | It is normally used for the grouping of data |

Some examples that elaborate on these differences:

1) Members of a class are private by default and members of a structure are public by default.

For example, program 1 fails in compilation but program 2 works fine,

**Program 1:**

C++

// Program 1

// C++ Program to demonstrate that

// Members of a class are private

// by default

using namespace std;

class Test {

// x is private

int x;

};

int main()

{

Test t;

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

// is private

return t.x;

}

**Output:**

prog.cpp: In function ‘int main()’:

prog.cpp:8:9: error: ‘int Test::x’ is private

int x;

^

prog.cpp:13:7: error: within this context

t.x = 20;

^

**Program 2:**

C++

// Program 2

// C++ Program to demonstrate that

// members of a structure are public

// by default.

#include <iostream>

struct Test {

// x is public

int x;

};

int main()

{

Test t;

t.x = 20;

// works fine because x is public

std::cout << t.x;

}

**Output**

20

**2) A class is declared using the class keyword, and a structure is declared using the struct keyword.**

**Syntax:**

class ClassName {

private:

member1;

member2;

public:

member3;

.

.

memberN;

};

**Syntax:**

struct StructureName {

member1;

member2;

.

.

.

memberN;

};

**3) Inheritance is possible with classes, and with structures.**

For example, programs 3 and 4 work fine.

**Program 3:**

C++

// Program 3

// C++ program to demonstrate

// inheritance with classes.

#include <iostream>

using namespace std;

// Base class

class Parent {

public:

int x;

};

// Subclass inheriting from

// base class (Parent).

class Child : public Parent {

public:

int y;

};

int main()

{

Child obj1;

// An object of class Child has

// all data members and member

// functions of class Parent.

obj1.y = 7;

obj1.x = 91;

cout << obj1.y << endl;

cout << obj1.x << endl;

return 0;

}

**Output**

7

91

**Program 4:**

C++

// Program 4

// C++ program to demonstrate

// inheritance with structures.

#include <iostream>

using namespace std;

struct Base {

public:

int x;

};

// is equivalent to

// struct Derived : public Base {}

struct Derived : Base {

public:

int y;

};

int main()

{

Derived d;

// Works fine because inheritance

// is public.

d.x = 20;

cout << d.x;

cin.get();

return 0;

}

**Output**

20