# C++ Structs

In C++, classes and structs are blueprints that are used to create the instance of a class. Structs are used for lightweight objects such as Rectangle, color, Point, etc.

Unlike class, structs in C++ are value type than reference type. It is useful if you have data that is not intended to be modified after creation of struct.

**C++ Structure** is a collection of different data types. It is similar to the class that holds different types of data.

## The Syntax Of Structure

```
struct structure_name
{
    // member declarations.
}
```

In the above declaration, a structure is declared by preceding the **struct keyword** followed by the identifier(structure name). Inside the curly braces, we can declare the member variables of different types. **Consider the following situation:**

```
struct Student
{
   char name[20];
    int id;
    int age;
}
```

In the above case, Student is a structure contains three variables name, id, and age. When the structure is declared, no memory is allocated. When the variable of a structure is created, then the memory is allocated. Let's understand this scenario.

## How to create the instance of Structure?

Structure variable can be defined as:

**Student s;**

Here, s is a structure variable of type **Student**. When the structure variable is created, the memory will be allocated. Student structure contains one char variable and two integer variable. Therefore, the memory for one char variable is 1 byte and two ints will be 2*4 = 8. The total memory occupied by the s variable is 9 byte.

### How to access the variable of Structure:

The variable of the structure can be accessed by simply using the instance of the structure followed by the dot (.) operator and then the field of the structure.

**For example:**

```
s.id = 4;
```

In the above statement, we are accessing the id field of the structure Student by using the **dot(.)** operator and assigns the value 4 to the id field.

## C++ Struct Example

Let's see a simple example of struct Rectangle which has two data members width and height.

```
#include <iostream>
using namespace std;
 struct Rectangle
{
  int width, height;

 };
int main(void) {
   struct Rectangle rec;
   rec.width=8;
   rec.height=5;
  cout<<"Area of Rectangle is: "<<(rec.width * rec.height)<<endl;
 return 0;
}
```

**Output:**

```
le is: 40
```

## C++ Struct Example: Using Constructor and Method

Let's see another example of struct where we are using the constructor to initialize data and method to calculate the area of rectangle.

```
#include <iostream>
using namespace std;
 struct Rectangle    {
  int width, height;
 Rectangle(int w, int h)
   {
      width = w;
      height = h;
   }
  void areaOfRectangle() {
   cout<<"Area of Rectangle is: "<<(width*height); }
 };
int main(void) {
    struct Rectangle rec=Rectangle(4,6);
    rec.areaOfRectangle();
   return 0;
}
```

**Output:**

```
Area of Rectangle is: 24
```

**Structure v/s Class**

| Structure | Class |
|---|---|
| If access specifier is not declared explicitly, then by default access specifier will be public. | If access specifier is not declared explicitly, then by default access specifier will be private. |

| Syntax of Structure:<br><br>struct structure_name<br>{<br>// body of the structure.<br>} | Syntax of Class:<br><br>class class_name<br>{<br>// body of the class.<br>} |
| --- | --- |
| The instance of the structure is known as "Structure variable". | The instance of the class is known as "Object of the class". |

← Prev

Next →



## Feedback

- Send your Feedback to feedback@javatpoint.com

## Help Others, Please Share

## Learn Latest Tutorials

Digital Marketing | Elasticsearch | Entity Framework | Firewall

Functional Programming | Google Colab | Graph Theory | Groovy

⇧ SCROLL TO TOP