

Class Constructor

Poli Nemkova, CSCE1040, UNT Summer2021

Credit to: <https://www.tutorialspoint.com/cplusplus/>

Class

A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

Class Definition

```
class Box {  
    public:  
        double length;    // Length of a box  
        double breadth;   // Breadth of a box  
        double height;    // Height of a box  
};
```

Object Definition

A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types.

```
Box Box1;           // Declare Box1 of type Box
Box Box2;           // Declare Box2 of type Box
```

Accessing Object Members

The public data members of objects of a class can be accessed using the direct member access operator (.)

```
#include <iostream>

using namespace std;

class Box {
public:
    double length;    // Length of a box
    double breadth;    // Breadth of a box
    double height;    // Height of a box
};

int main() {
    Box Box1;          // Declare Box1 of type Box
    Box Box2;          // Declare Box2 of type Box
    double volume = 0.0; // Store the volume of a box here

    // box 1 specification
    Box1.height = 5.0;
    Box1.length = 6.0;
    Box1.breadth = 7.0;

    // box 2 specification
    Box2.height = 10.0;
    Box2.length = 12.0;
    Box2.breadth = 13.0;
```

Member Functions

A member function of a class is a function that has its definition or its prototype within the class definition like any other variable. It operates on any object of the class of which it is a member, and has access to all the members of a class for that object.

```
class Box {  
    public:  
        double length;           // Length of a box  
        double breadth;         // Breadth of a box  
        double height;          // Height of a box  
        double getVolume(void); // Returns box volume  
};
```

Member Function Definition:

Member functions can be defined:

- 1) within the class definition

or

- 2) separately using scope resolution operator (::).

Define a member function: Option 1

```
class Box {  
    public:  
        double length;           // Length of a box  
        double breadth;         // Breadth of a box  
        double height;          // Height of a box  
        double getVolume(void); // Returns box volume  
};
```

```
double Box::getVolume(void) {  
    return length * breadth * height;  
}
```


Define a member function: Option 2

```
class Box {  
    public:  
        double length;        // Length of a box  
        double breadth;       // Breadth of a box  
        double height;        // Height of a box  
  
        double getVolume(void) {  
            return length * breadth * height;  
        }  
};
```

What is the difference?

Defining a member function within the class definition declares the function **inline**, even if you do not use the inline specifier.

What is inline function?

C++ provides inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.

(read more [here](#), it's short but descriptive)

How to access member function?

It's easy. A member function will be called using a dot operator (.) on a object where it will manipulate data related to that object only:

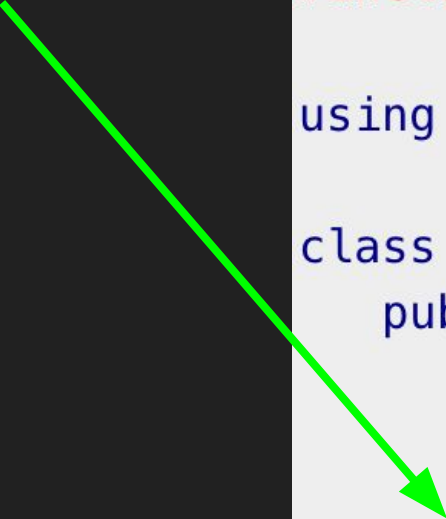
```
myBox.getVolume(); // Call member function for the object
```

Constructors <3

A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Default Constructor



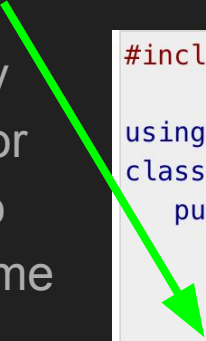
```
#include <iostream>

using namespace std;

class Line {
public:
    void setLength( double len );
    double getLength( void );
    Line(); // This is the constructor
private:
    double length;
};
```

Parameterized Constructor

A default constructor does not have any parameter, but if you need, a constructor can have parameters. This helps you to assign initial value to an object at the time of its creation.



```
#include <iostream>

using namespace std;
class Line {
public:
    void setLength( double len );
    double getLength( void );
    Line(double len); // This is the constructor

private:
    double length;
};
```