Constructor:

Constructor in C++ is a special method that is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally. The constructor in C++ has the same name as the class or structure. Constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object which is why it is known as constructors.

Constructor does not have a return value, hence they do not have a return type.

To create a constructor, use the same name as the class, followed by parentheses ():

```
class MyClass {          // The class
  public:            // Access specifier
          MyClass() {      // Constructor
          cout << "Hello World!";
           }
};

int main() {
  MyClass myObj;       // Create an object of MyClass (this will call the constructor)
  return 0;
}
```

Characteristics of the constructor:

- The name of the constructor is the same as its class name.
- Constructors are mostly declared in the public section of the class though it can be declared in the private section of the class.
- Constructors do not return values; hence they do not have a return type.
- A constructor gets called automatically when we create the object of the class.
- Constructors can be overloaded.
- Constructor can not be declared virtual.

```
#include <iostream>
using namespace std;
class student {
int rno;
char name[10];
double fee;

public:
          student()
          {
```

```
        cout << "Enter the RollNo:";
        cin >> rno;
        cout << "Enter the Name:";
        cin >> name;
        cout << "Enter the Fee:";
        cin >> fee;
        }

        void display()
        {
        cout << endl << rno << "\t" << name << "\t" << fee;
        }
};

int main()
{
        student s; // constructor gets called automatically when
        // we create the object of the class
        s.display();
        return 0;
}
```

// defining the constructor outside the class

```
#include <iostream>
using namespace std;
class student {
   int rno;
   char name[50];
   double fee;

public:
   student();
   void display();
};

student::student()
{
   cout << "Enter the RollNo:";
   cin >> rno;

   cout << "Enter the Name:";
   cin >> name;

   cout << "Enter the Fee:";
   cin >> fee;
}

void student::display()
```

```
{
    cout << endl << rno << "\t" << name << "\t" << fee;
}

int main()
{
    student s;
    s.display();

    return 0;
}
```

The constructor has the same name as the class, it is always public, and it does not have any return value.

Constructor Parameters

Constructors can also take parameters (just like regular functions), which can be useful for setting initial values for attributes.

```
class Car {       // The class
  public:         // Access specifier
            string brand;  // Attribute
           string model;  // Attribute
            int year;         // Attribute
           Car(string x, string y, int z) { // Constructor with parameters
           brand = x;
           model = y;
            year = z;
           }
};

int main() {
  // Create Car objects and call the constructor with different values
  Car carObj1("BMW", "X5", 1999);
  Car carObj2("Ford", "Mustang", 1969);

  // Print values
  cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
  return 0;
}
```

Just like functions, constructors can also be defined outside the class. First, declare the constructor inside the class, and then define it outside of the class by specifying the name of the class, followed by the scope resolution :: operator, followed by the name of the constructor (which is the same as the class):

```
class Car {      // The class
  public:          // Access specifier
          string brand;  // Attribute
           string model;  // Attribute
          int year;          // Attribute
          Car(string x, string y, int z); // Constructor declaration
};

// Constructor definition outside the class
Car::Car(string x, string y, int z) {
  brand = x;
  model = y;
  year = z;
}

int main() {
  // Create Car objects and call the constructor with different values
  Car carObj1("BMW", "X5", 1999);
  Car carObj2("Ford", "Mustang", 1969);

  // Print values
  cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
  return 0;
}
```

Types of Constructors

1. Default Constructors:

Default constructor is the constructor which doesn't take any argument. It has no parameters. It is also called a zero-argument constructor. A constructor with no parameters is known as a **default constructor**.

// C++ program to demonstrate the use of default constructor

```
#include <iostream>
```

```
using namespace std;

// declare a class
class  Wall {
  private:
        double length;

  public:
        // default constructor to initialize variable
        Wall() {
        length = 5.5;
        cout << "Creating a wall." << endl;
        cout << "Length = " << length << endl;
        }
};

int main() {
  Wall wall1;
  return 0;
}


// Cpp program to illustrate the
// concept of Constructors
#include <iostream>
using namespace std;

class construct {
public:
    int a, b;

    // Default Constructor
    construct()
    {
         a = 10;
         b = 20;
    }
};

int main()
{
    // Default constructor called automatically
    // when the object is created
    construct c;
    cout << "a: " << c.a << endl << "b: " << c.b;
    return 1;
}
```

Note: If we have not defined a constructor in our class, then the C++ compiler will automatically create a default constructor with an empty code and no parameters.

2. Parameterized Constructors:

In C++, a constructor with parameters is known as a parameterized constructor. This is the preferred method to initialize member data.

```
// C++ program to calculate the area of a wall

#include <iostream>
using namespace std;

// declare a class
class Wall {
  private:
        double length;
        double height;

  public:
        // parameterized constructor to initialize variables
        Wall(double len, double hgt) {
        length = len;
        height = hgt;
        }

        double calculateArea() {
        return length * height;
        }
};

int main() {
  // create object and initialize data members
  Wall wall1(10.5, 8.6);
  Wall wall2(8.5, 6.3);

  cout << "Area of Wall 1: " << wall1.calculateArea() << endl;
  cout << "Area of Wall 2: " << wall2.calculateArea();

  return 0;
}
```

```
// CPP program to illustrate
// parameterized constructors
#include <iostream>
using namespace std;
```

```
class Point {
private:
    int x, y;

public:
    // Parameterized Constructor
    Point(int x1, int y1)
    {
            x = x1;
            y = y1;
    }

    int getX() { return x; }
    int getY() { return y; }
};

int main()
{
    // Constructor called
    Point p1(10, 15);

    // Access values assigned by constructor
    cout << "p1.x = " << p1.getX()
            << ", p1.y = " << p1.getY();

    return 0;
}
```

- **Uses of Parameterized constructor:**
    1. It is used to initialize the various data elements of different objects with different values when they are created.
    2. It is used to overload constructors.
- **Can we have more than one constructor in a class?**
  Yes, It is called Constructor Overloading.

3.Copy Constructor

The copy constructor in C++ is used to copy data of one object to another.

```
// C++ program to demonstrate the working
// of a COPY CONSTRUCTOR
#include <iostream>
using namespace std;
```

```
class Point {
private:
    int x, y;

public:
    Point(int x1, int y1)
    {
            x = x1;
            y = y1;
    }

    // Copy constructor
    Point(const Point& p1)
    {
            x = p1.x;
            y = p1.y;
    }

    int getX() { return x; }
    int getY() { return y; }
};

int main()
{
    Point p1(10, 15); // Normal constructor is called here
    Point p2 = p1; // Copy constructor is called here

    // Let us access values assigned by constructors
    cout << "p1.x = " << p1.getX()
            << ", p1.y = " << p1.getY();
    cout << "\np2.x = " << p2.getX()
            << ", p2.y = " << p2.getY();
    return 0;
}
```

```
#include <iostream>
using namespace std;

// declare a class
class Wall {
  private:
          double length;
          double height;

  public:
```

```
        // initialize variables with parameterized constructor
        Wall(double len, double hgt) {
        length = len;
        height = hgt;
        }

        // copy constructor with a Wall object as parameter
        // copies data of the obj parameter
        Wall(Wall &obj) {
        length = obj.length;
        height = obj.height;
        }

        double calculateArea() {
        return length * height;
        }
};

int main() {
  // create an object of Wall class
  Wall wall1(10.5, 8.6);

  // copy contents of wall1 to wall2
  Wall wall2 = wall1;

  // print areas of wall1 and wall2
  cout << "Area of Wall 1: " << wall1.calculateArea() << endl;
  cout << "Area of Wall 2: " << wall2.calculateArea();

  return 0;
}
```