CONSTRUCTOR

Introduction of C++ Constructors

A constructor is a special type of member function that is called automatically when an object is created.

In C++, a constructor has the same name as that of the class and it does not have a return type. **For example**,

```
class  Wall {
  public:
    // create a constructor
    Wall() {
      // code
    }
};
```

Here, the function Wall() is a constructor of the class Wall. Notice that the constructor has the same name as the class, does not have a return type, and is public

C++ Default Constructor

A constructor with no parameters is known as a default constructor. In the example above, Wall() is a default constructor.

Example 1: C++ 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;
}
```

Run Code

Output

```
Creating a Wall
Length = 5.5
```

Here, when the wall1 object is created, the Wall() constructor is called. This sets the length variable of the object to 5.5.

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.

C++ Parameterized Constructor

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

Example 2: **C++ Parameterized Constructor**

```
// 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;

}
```

Run Code

Output

```
Area of Wall 1: 90.3

Area of Wall 2: 53.55
```

Here, we have created a parameterized constructor Wall() that has 2 parameters: double len and double hgt. The values contained in these parameters are used to initialize the member variables length and height.

When we create an object of the Wall class, we pass the values for the member variables as arguments. The code for this is:

```
Wall wall1(10.5, 8.6);

Wall wall2(8.5, 6.3);
```

With the member variables thus initialized, we can now calculate the area of the wall with the calculateArea() function.

C++ Copy Constructor

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

Example 3: C++ Copy Constructor

```
#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;
}
```

Run Code

Output

```
Area of Wall 1: 90.3

Area of Wall 2: 90.3
```

In this program, we have used a copy constructor to copy the contents of one object of the Wall class to another. The code of the copy constructor is:

```
Wall(Wall &obj) {
  length = obj.length;
  height = obj.height;
}
```

Notice that the parameter of this constructor has the address of an object of the Wall class.

We then assign the values of the variables of the obj object to the corresponding variables of the object calling the copy constructor. This is how the contents of the object are copied.

In main(), we then create two objects wall1 and wall2 and then copy the contents
of wall1 to wall2:

```
// copy contents of wall1 to wall2

Wall wall2 = wall1;
```

Here, the wall2 object calls its copy constructor by passing the address of the wall1 object as its argument i.e. &obj = &wall1.