C++ Copy Constructor

Previous Page Next Page

The **copy constructor** is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is used to −

- Initialize one object from another of the same type.
- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

If a copy constructor is not defined in a class, the compiler itself defines one.If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor. The most common form of copy constructor is shown here −

```
classname (const classname &obj) {
   // body of constructor
}
```

Here, **obj** is a reference to an object that is being used to initialize another object.

```
#include <iostream>

using namespace std;

class Line {

   public:
      int getLength( void );
      Line( int len );             // simple constructor
      Line( const Line &obj);  // copy constructor
      ~Line();                     // destructor

   private:
      int *ptr;
};

// Member functions definitions including constructor
Line::Line(int len) {
   cout << "Normal constructor allocating ptr" << endl;

   // allocate memory for the pointer;
   ptr = new int;
   *ptr = len;
}

Line::Line(const Line &obj) {
   cout << "Copy constructor allocating ptr." << endl;
   ptr = new int;
   *ptr = *obj.ptr; // copy the value
}

Line::~Line(void) {
   cout << "Freeing memory!" << endl;
   delete ptr;
}

int Line::getLength( void ) {
   return *ptr;
}

void display(Line obj) {
   cout << "Length of line : " << obj.getLength() <<endl;
}

// Main function for the program
int main() {
   Line line(10);
   display(line);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Normal constructor allocating ptr
Copy constructor allocating ptr.
Length of line : 10
Freeing memory!
Freeing memory!
```

Let us see the same example but with a small change to create another object using existing object of the same type −

Try it

```
#include <iostream>

using namespace std;

class Line {
   public:
      int getLength( void );
      Line( int len );             // simple constructor
      Line( const Line &obj);  // copy constructor
      ~Line();                     // destructor

   private:
      int *ptr;
};

// Member functions definitions including constructor
Line::Line(int len) {
   cout << "Normal constructor allocating ptr" << endl;

   // allocate memory for the pointer;
   ptr = new int;
   *ptr = len;
}

Line::Line(const Line &obj) {
   cout << "Copy constructor allocating ptr." << endl;
   ptr = new int;
   *ptr = *obj.ptr; // copy the value
}

Line::~Line(void) {
   cout << "Freeing memory!" << endl;
   delete ptr;
}

int Line::getLength( void ) {
   return *ptr;
}

void display(Line obj) {
   cout << "Length of line : " << obj.getLength() <<endl;
}

// Main function for the program
int main() {

   Line line1(10);

   Line line2 = line1; // This also calls copy constructor

   display(line1);
   display(line2);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Normal constructor allocating ptr
Copy constructor allocating ptr.
Copy constructor allocating ptr.
Length of line : 10
Freeing memory!
```

```
Copy constructor allocating ptr.
Length of line : 10
Freeing memory!
Freeing memory!
Freeing memory!
```

Previous Page Next Page

Advertisements