

C++ Copy Constructor

Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type.

Copy Constructor is of two types:

- **Default Copy constructor:** The compiler defines the default copy constructor. If the user defines no copy constructor, compiler supplies its constructor.
- **User Defined constructor:** The programmer defines the user-defined constructor.

Syntax Of User-defined Copy Constructor:

It is usually of the form **X (X&)**, where X is the class name. The compiler provides a default Copy Constructor to all the classes.

Class_name(const class_name &old_object);

```
Classname(const classname & objectname)
{
}
```

```
main.cpp
1  #include<iostream>
2  using namespace std;
3  class CopyConstructorDemo
4  {
5
6  public:
7      int x, y;
8      CopyConstructorDemo(int x1, int y1)
9      {
10         x = x1;
11         y = y1;
12     }
13     // Copy constructor
14     CopyConstructorDemo(const CopyConstructorDemo &p2)
15     {
16         x = p2.x;
17         y = p2.y;
18     }
19 };
```

Join Our Telegram Group to Get Notifications, Study Materials, Practice test & quiz:

<https://t.me/ccatpreparations> Visit: <https://ccatpreparation.com>

```

21 int main()
22 {
23     CopyConstructorDemo p1(10, 15); // Normal constructor is called here
24     CopyConstructorDemo p2 = p1; // Copy constructor is called here
25     CopyConstructorDemo p3(p1); // Copy constructor is called here
26
27     // Let us access values assigned by constructors
28     cout << "p1.x = " << p1.x << ", p1.y = " << p1.y;
29     cout << "\np2.x = " << p2.x << ", p2.y = " << p2.y;
30     cout << "\np3.x = " << p3.x << ", p3.y = " << p3.y;
31
32     return 0;
33 }

```

```

p1.x = 10, p1.y = 15
p2.x = 10, p2.y = 15
p3.x = 10, p3.y = 15

```

When Copy Constructor is called

- When we initialize the object with another existing object of the same class type. For example, Demo s1 = s2 or s1(s2), where Demo is the class.
- When the object of the same class type is passed by value as an argument.
- When the function returns the object of the same class type by value.

Differences b/w Copy constructor and Assignment operator (=)

```

21 int main()
22 {
23     CopyConstructorDemo p1(10, 15); // Normal constructor is called here
24     CopyConstructorDemo p2 = p1; // Copy constructor is called here
25     p1.x = 19;
26     CopyConstructorDemo p3(p1); // Copy constructor is called here
27
28     // Let us access values assigned by constructors
29     cout << "p1.x = " << p1.x << ", p1.y = " << p1.y;
30     cout << "\np2.x = " << p2.x << ", p2.y = " << p2.y;
31     cout << "\np3.x = " << p3.x << ", p3.y = " << p3.y;
32

```

input

```

p1.x = 19, p1.y = 15
p2.x = 10, p2.y = 15
p3.x = 19, p3.y = 15

```

Copy Constructor

It is an overloaded constructor.

Assignment Operator

It is a bitwise operator.

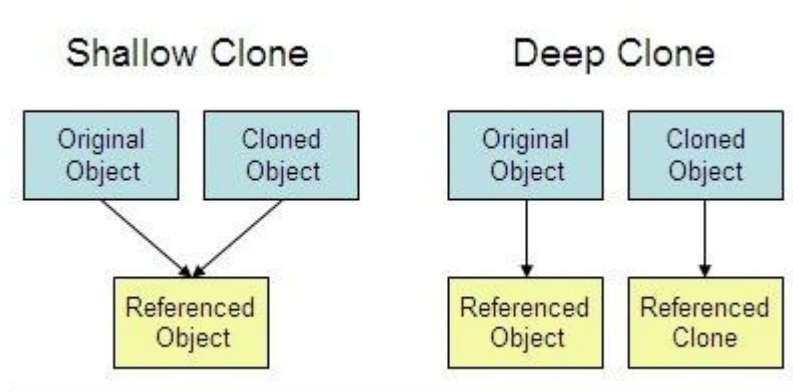
Join Our Telegram Group to Get Notifications, Study Materials, Practice test & quiz:

<https://t.me/ccatpreparations> Visit: <https://ccatpreparation.com>

It initializes the new object with the existing object.	It assigns the value of one object to another object.
Syntax of copy constructor: <pre>Class_name(const class_name &object_name) { // body of the constructor. }</pre>	Syntax of Assignment operator: <pre>Class_name a,b; b = a;</pre>
<ul style="list-style-type: none"> ○ The copy constructor is invoked when the new object is initialized with the existing object. ○ The object is passed as an argument to the function. ○ It returns the object. 	The assignment operator is invoked when we assign the existing object to a new object.
Both the existing object and new object shares the different memory locations.	Both the existing object and new object shares the same memory location.
If a programmer does not define the copy constructor, the compiler will automatically generate the implicit default copy constructor.	If we do not overload the "=" operator, the bitwise copy will occur.

Two types of copies are produced by the constructor:

- Shallow copy – **with default Constructor**
- Deep copy – **with user defined copy constructor**



Join Our Telegram Group to Get Notifications, Study Materials, Practice test & quiz:

<https://t.me/ccatpreparations> Visit: <https://ccatpreparation.com>

Shallow Copy Constructor with pointers

The concept of shallow copy constructor is explained through an example. Two students are entering their details in excel sheet simultaneously from two different machines shared over a network. Changes made by both of them will be reflected in the excel sheet. Because same excel sheet is opened in both locations. This is what happens in shallow copy constructor. Both objects will point to same memory location.

Shallow copy copies reference to original objects. The compiler provides a default copy constructor. Default copy constructor provides a shallow copy as shown in below example. It is a bit-wise copy of an object.

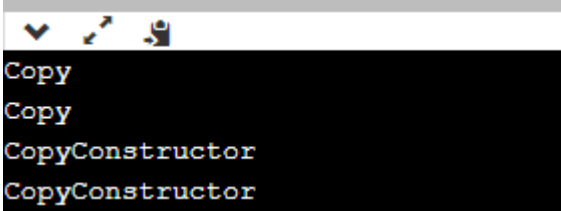
Shallow copy constructor is used when class is not dealing with any dynamically allocated memory.

main.cpp

```
1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  class ShallowCopyConstructorWithPointer
5  {
6      char *s_copy;
7      public:
8      ShallowCopyConstructor(const char *str)
9      {
10         s_copy = new char[16]; //Dynamic memory allocation
11         strcpy(s_copy, str);
12     }
13     /* concatenate method */
14     void concatenate(const char *str)
15     {
16         strcat(s_copy, str); //Concatenating two strings
17     }
18     /* copy constructor */
19     ~ShallowCopyConstructor ()
20     {
21         delete [] s_copy;
22     }
23     void display()
24     {
25         cout<<s_copy<<endl;
26     }
27 };
28 /* main function */
29 int main()
30 {
31     ShallowCopyConstructor c1("Copy");
32     ShallowCopyConstructor c2 = c1; //Copy constructor
33     c1.display();
34     c2.display();
35     c1.concatenate("Constructor"); //c1 is invoking concatenate()
36     c1.display();
37     c2.display();
38     return 0;
39 }
```

Join Our Telegram Group to Get Notifications, Study Materials, Practice test & quiz:

<https://t.me/ccatpreparations> Visit: <https://ccatpreparation.com>



Deep Copy Constructor

Let's consider an example for explaining deep copy constructor. You are supposed to submit an assignment tomorrow and you are running short of time, so you copied it from your friend. Now you and your friend have same assignment content, but separate copies. Therefore any modifications made in your copy of assignment will not be reflected in your friend's copy. This is what happens in deep copy constructor.

Deep copy allocates separate memory for copied information. So the source and copy are different. Any changes made in one memory location will not affect copy in the other location. When we allocate dynamic memory using pointers we need user defined copy constructor. Both objects will point to different memory locations.

General requirements for deep copy:

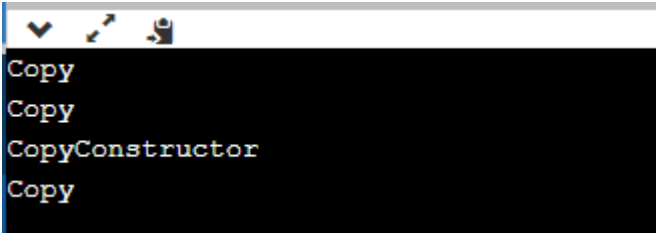
- A normal constructor.
- A destructor to delete the dynamically allocated memory.
- A copy constructor to make a copy of the dynamically allocated memory.
- An overloaded assignment operator.

Join Our Telegram Group to Get Notifications, Study Materials, Practice test & quiz:

<https://t.me/ccatpreparations> Visit: <https://ccatpreparation.com>

main.cpp

```
1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  class DeepCopyConstructor
5  {
6      char *s_copy;
7      public:
8      DeepCopyConstructor(const char *str)
9      {
10         s_copy = new char[16]; //Dynamic memory allocation
11         strcpy(s_copy, str);
12     }
13     DeepCopyConstructor (const DeepCopyConstructor &str)
14     {
15         s_copy = new char[16]; //Dynamic memory allocation
16         strcpy(s_copy, str.s_copy);
17     }
18     /* concatenate method */
19     void concatenate(const char *str)
20     {
21         strcat(s_copy, str); //Concatenating two strings
22     }
23     /* copy constructor */
24     ~DeepCopyConstructor ()
25     {
26         delete [] s_copy;
27     }
28     void display()
29     {
30         cout<<s_copy<<endl;
31     }
32 };
33 /* main function */
34 int main()
35 {
36     DeepCopyConstructor c1("Copy");
37     DeepCopyConstructor c2 = c1; //Copy constructor
38     c1.display();
39     c2.display();
40     c1.concatenate("Constructor"); //c1 is invoking concatenate()
41     c1.display();
42     c2.display();
43     return 0;
44 }
```



```
Copy
Copy
CopyConstructor
Copy
```

Join Our Telegram Group to Get Notifications, Study Materials, Practice test & quiz:

<https://t.me/ccatpreparations> Visit: <https://ccatpreparation.com>