

C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com



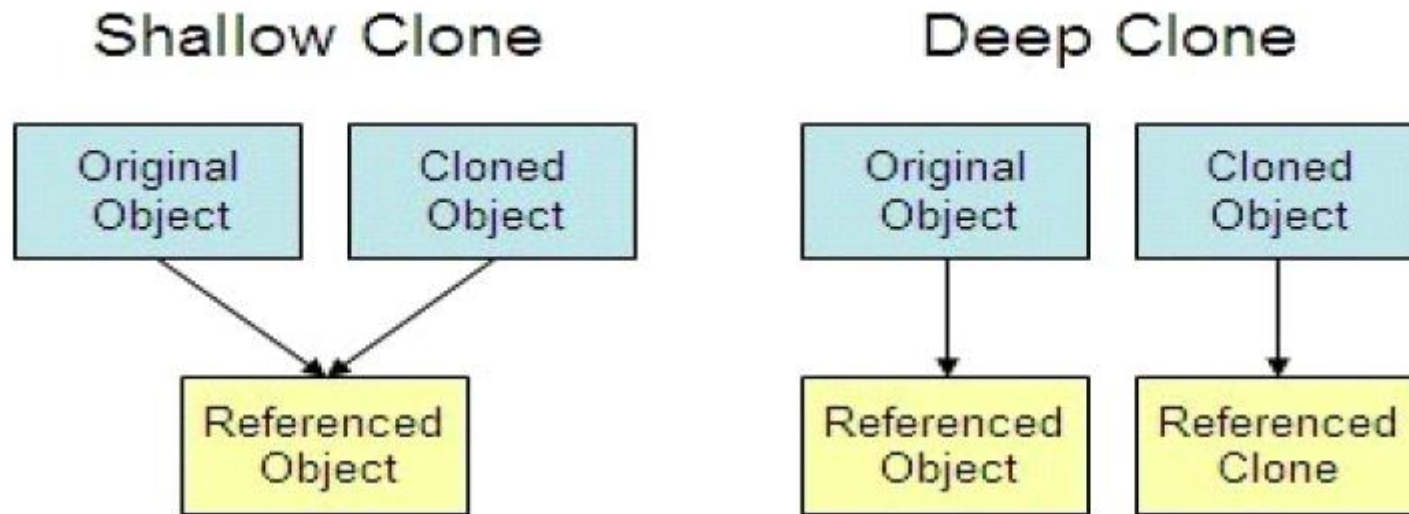
Agenda

- Shallow and Deep Copy
- Static Data Members
- Static Member Functions
- Template



Object Copying

- In object-oriented programming, “object copying” is a process of creating a copy of an existing object.
- The resulting object is called an object copy or simply copy of the original object.
- Methods of copying:
 - Shallow copy
 - Deep copy



Types of Copy

- **Shallow Copy**

- The process of copying state of object into another object.
- It is also called as bit-wise copy.
- When we assign one object to another object at that time copying all the contents from source object to destination object as it is. Such type of copy is called as shallow copy.
- Compiler by default create a shallow copy. Default copy constructor always create shallow copy.

- **Deep Copy**

- Deep copy is the process of copying state of the object by modifying some state.
- It is also called as member-wise copy.
- When class contains at least one data member of pointer type, when class contains user defined destructor and when we assign one object to another object at that time instead of copy base address allocate a new memory for each and every object and then copy contain from memory of source object into memory of destination object. Such type of copy is called as deep copy.



Shallow Copy

- Process of copying state of object into another object as it is, is called shallow copy.
- It is also called as bit-wise copy / bit by bit copy.
- Following are the cases when shallow copy taken place:
 1. If we pass variable / object as a argument to the function by value.
 2. If we return object from function by value.
 3. If we initialize object: `Complex c2=c1`
 4. If we assign the object , `c2=c1`;
 5. If we catch object by value.
- Examples of shallow copy

Example 1: (Initialization)

```
int num1=50;  
int num2=num1;
```

Example 2: (Assignment)

```
Complex c1(40,50);  
c2=c1;
```



Deep Copy

- It is also called as member-wise copy. By modifying some state, if we create copy of the object then it is called deep copy.
 - Conditions to create deep copy
 - Class must contain at least one pointer type data member.

```
class Array
{
    private:
        int size;
        int *arr;
        //Case - I
    public:
        Array( int size )
        {
            this->size = size;
            this->arr = new int[ this->size ];
        }
};
```

- Steps to create deep copy
 - 1. Copy the required size from source object into destination object.
 - 2. Allocate new resource for the destination object.
 - 3. Copy the contents from resource of source object into destination object.



Static Variable

- All the static and global variables get space only once during program loading / before starting execution of main function
- Static variable is also called as shared variable.
- Uninitialized static and global variable get space on BSS segment.
- Initialized static and global variable get space on Data segment.
- Default value of static and global variable is zero.
- Static variables are same as global variables but it is having limited scope.



Static Methods or Static Member Functions

- Except main function, we can declare global function as well as member function static.
- To access non static members of the class, we should declare member function non static and to access
- static members of the class we should declare member function static.
- Member function of a class which is designed to call on object is called instance method. In short non static member function is also called as instance method.
- To access instance method either we should use object, pointer or reference to object.
- static member function is also called as class level method.
- To access class level method we should use classname and ::(scope resolution) operator.



Template

- If we want to write generic program in C++, then we should use template.
- This feature is mainly designed for implementing generic data structure and algorithm.
- If we want to write generic program, then we should pass data type as a argument. And to catch that type we should define template.
- Using template we can not reduce code size or execution time but we can reduce developers effort.

<pre>int num1 = 10, num2 = 20; swap_object<int>(num1, num2); string str1="Pune", str2="Karad"; swap_object<string>(str1, str2);</pre>	<p>In this code, <int> and <string> is considered as type argument.</p>
<pre>template<typename T> //or template<class T> //T : Type Parameter void swap(b obj1, T obj2) { T temp = obj1; obj1 = obj2; obj2 = temp; }</pre>	<p>template and typename is keyword in C++. By passing datatype as argument we can write generic code hence parameterized type is called template</p>



Types of Template

- Function Template
- Class Template



Example of Function Template

```
//template<typename T>//T : Type Parameter
```

```
template<class T> //T : Type Parameter
```

```
void swap_number( T &o1, T &o2 )
```

```
{  T temp = o1;
```

```
    o1 = o2;
```

```
    o2 = temp;
```

```
}
```

```
int main( void )
```

```
{
```

```
    int num1 = 10;
```

```
    int num2 = 20;
```

```
    swap_number<int>( num1, num2 );    //Here int is type argument
```

```
    cout<<"Num1 : "<<num1<<endl;
```

```
    cout<<"Num2 : "<<num2<<endl;
```

```
    return 0;
```

```
}
```



Example of Class Template

```
template<class T>
class Array // Parameterized type
{
private:
    int size;
    T *arr;
public:
    Array( void ) : size( 0 ), arr( NULL )
    {
    }
    Array( int size )
    {
        this->size = size;
        this->arr = new T[ this->size ];
    }
    void acceptRecord( void ){}
    void printRecord( void ){ }
    ~Array( void ){ }
};
```

```
int main(void)
{
    Array<char> a1( 3 );
    a1.acceptRecord();
    a1.printRecord();
    return 0;
}
```



Thank You

