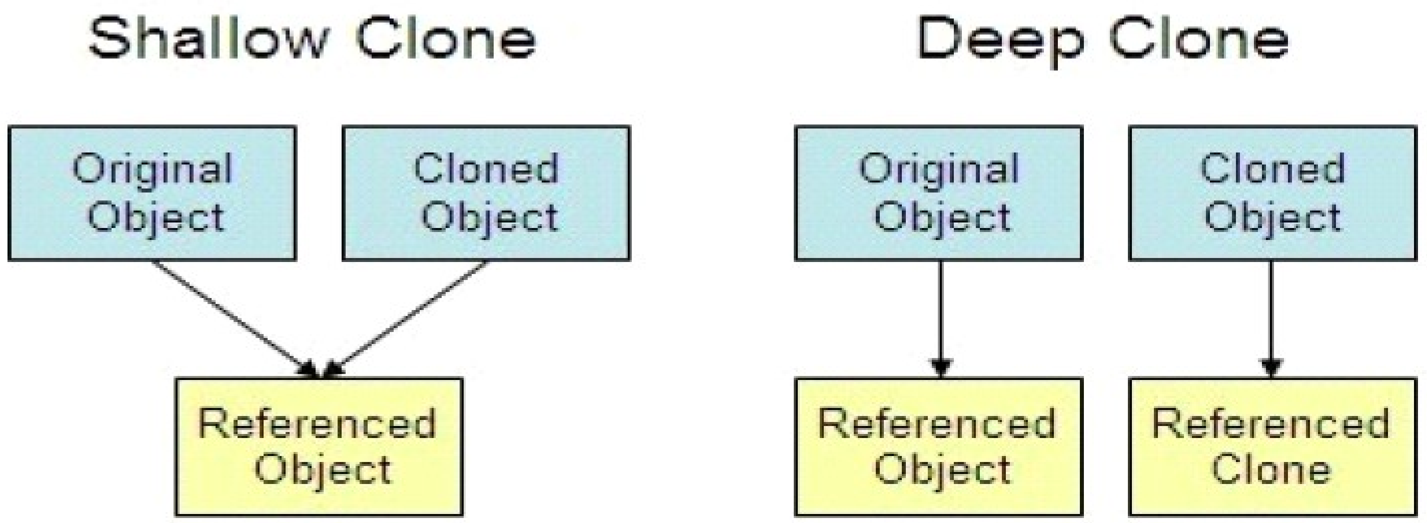
C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

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.
- <u>Examples of shallow copy</u>

  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.

Operator Overloading

- operator is token in C/C++.
- It is used to generate expression.
- operator is keyword in C++.
- Types of operator:
  - Unary operator (++,--,&,!,~,sizeof())
  - Binary Operator (Arithmetic, relational, logical , bitwise, assignment)
  - Ternary operator (conditional)
- In C++, also we can not use operator with objects of user defined type directly.
- If we want to use operator with objects of user defined type then we should overload operator.
- To overload operator, we should define **operator function.**
- **We can define operator function using 2 ways:**
  - **Using member function**
  - **Using non member function**

Program Demo without operator overloading

- Create a class point, having two fileld, x, y.
  - Point(void)
  - Point(int x, int y)

```
int main( void )
{
Point pt1(10,20);
Point pt2(30,40);
Point pt3;
pt3 = pt1 + pt2; //Not OK( implicitly)
return 0;
}
```

Need Of Operator Overloading

- we extend the meaning of the operator.
- If we want to use operator with the object of use defined type, then we need to overload operator.
- To overload operator, we need to define operator function.
- In C++, operator is a keyword
  - Suppose we want to use plus(+) operator with objects then we need to define operator+( ) function.

We define operator function either inside class (as a member function) or outside class (as a non-member function).	Point pt1(10,20), pt2(30,40), pt3;
	
	pt3 = pt1 + pt2; //pt3 = pt1.operator+(pt2); //using member function
	OR
	pt3 = pt1 + pt2; //pt3 = operator+(pt1, pt2); //using non member function

Operator Overloading

using member function

- operator function must be member function
- If we want to overload, binary operator using member function then operator function should take only one parameter.
- Example : c3 = c1 + c2; //will be called as
  - c3 = c1.operator+(c2)

- Example :

```
Point operator+( Point &other ) //Member Function
        {
        Point temp;
        temp.xPos = this->xPos + other.xPos;
        temp.yPos = this->yPos + other.yPos;
        return temp;
        }
```

using non member function

- Operator function must be global function
- If we want to overload binary operator using non member function then operator function should take two parameters.
- Example : c3 = c1 + c2; //will be called as
  - c3 = operator+(c1,c2);

- Example:

```
Point operator+( Point &pt1, Point &pt2 ) //Non Member
  Function
{
Point temp;
temp.xPos = pt1.xPos + pt2.xPos;
temp.yPos = pt1.yPos + pt2.yPos;
return temp;
}
```

We can not overload following operator using member as well as non member function

1. dot/member selection operator(.)
2. Pointer to member selection operator(.*)
3. Scope resolution operator(::)
4. Ternary/conditional operator(? :)
5. sizeof() operator
6. typeid() operator
7. static_cast operator
8. dynamic_cast operator
9. const_cast operator
10. reinterpret_cast operator

We can not overload following operators using non member function:

- Assignment operator(=)
- Subscript / Index operator([])
- Function Call operator[()]
- Arrow / Dereferencing operator(->)

Thank You