C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

Copy Constructor

- Copy constructor is a single parameter constructor hence it is considered as parameterized constructor
- Example:
  - **Complex sum(const Complex &c2)**

Reference

- Reference is derived data type.
- It alias or another name given to the exisiting memory location / object.
  - Example : int a=10; int &r = a;
  - In above example a is referent variable and r is reference variable.
  - It is mandatory to initialize reference.
- Reference is alias to a variable and cannot be reinitialized to other variable
- When ‘&’ operator is used with reference, it gives address of variable to which it refers.
- Reference can be used as data member of any class
- **Using typedef we can create alias for class whereas using reference we can create alias for object.**

Reference

- We can not create reference to constant value.
  - int &num2 = 10; //can not create reference to constant value
- Reference is internally considered as constant pointer hence referent of reference must be variable/object.

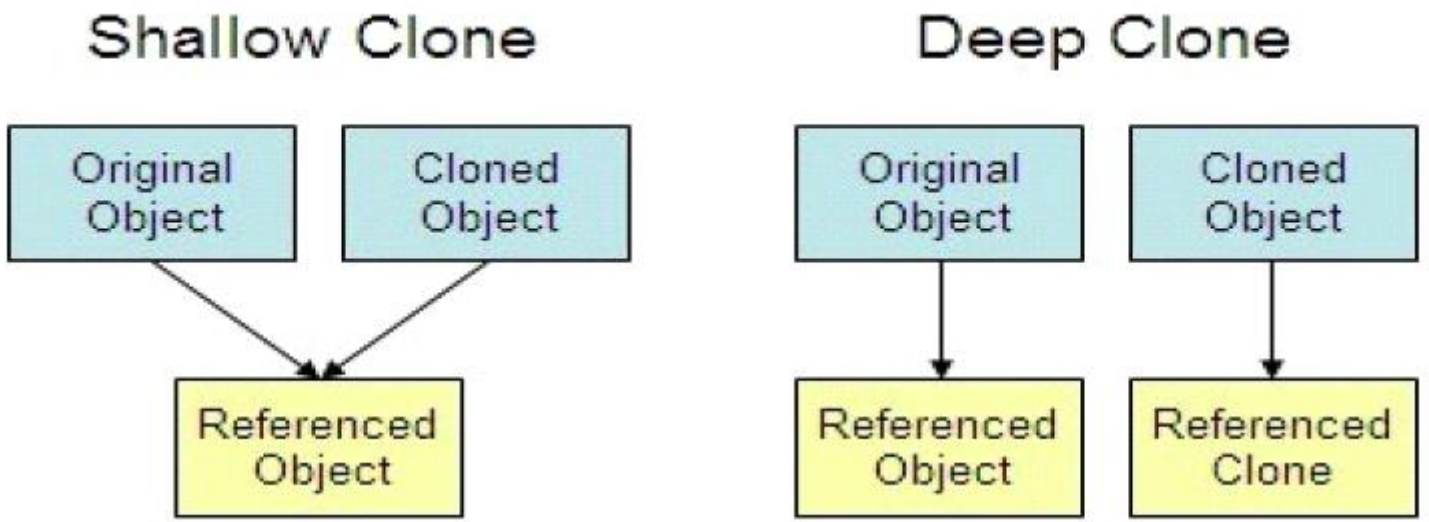
```
int main( void )
{
        int num1 = 10;
        int &num2 = num1;
        //int *const num2 = &num1;
        cout<<"Num2 : "<<num2<<endl;
        //cout<<"Num2 : "<<*num2<<endl;
        return 0;
}
```

Reference to array:

```
int main( void )
{
	int arr[ 3 ] = { 10, 20, 30 };
	int i;
	int (&arr2)[ 3 ] = arr;
	for(i = 0; i < 3; ++ i)
	    cout<<arr2[ i ]<<endl;
	return 0;
}
```

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.
- Unintialized static and global variable get space on BSS segment.
- Intialized 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 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.

Exception Handling

- If we give wrong input to the application then it generates runtime error/exception.
- Exception is an object, which is used to send notification to the end user of the system if any exceptional situation occurs in the program.
- To handle exception then we should use 3 keywords:
- 1**. try**
  - try is keyword in C++.
  - If we want to inspect exception then we should put statements inside try block/handler.
  - Try block may have multiple catch block but it must have at least one catch block.

Note : For thrown exception, if we do not provide matching catch block then C++ runtime gives call to the std::terminate() function which implicitly gives call to the std::abort() function.

- **2. catch**
  - If we want to handle exception then we should use catch block/handler.
  - Single try block may have multiple catch block.
  - Catch block can handle exception thrown from try block only.
  - A catch block, which can handle any type of exception is called generic catch block / catch-all handler.
  - For each type of exception, we can write specific catch block or we can write single catch block which can handle all types of exception. A catch block which can handle all type of exception is called generic catch block.
- **3. throw**
  - throw is keyword in C++.
  - If we want to generate exception explicitly then we should use throw keyword.
  - "throw statement" is a jump statement.
  - To generate new exception, we should use throw keyword. Throw statement is jump statement.

Consider the following code

- In C++, try, catch and throw keyword is used to handle exception.

```
int num1;
accept_record( num1 );
int num2;
accept_record( num1 );
try {
    if( num2 == 0 )
            throw "/ by zero  exception";
    int result = num1 / num2;
    print_record( result )
}
catch( const char *ex ){
    cout<<ex<<endl;
}
catch(...)
{
cout<<"Genenric catch  handler"<<endl;
}
```

Consider the following code

In this code, int type exception is thrown but matching catch block is not available. Even generic catch block is also not available. Hence program will terminate. Because, if we throw exception from try block then catch block can handle it. But with the help of function we can throw exception from outside of the try block.

```
int main( void ){
    int num1;
    accept_record(num1);
    int num2;
    accept_record(num2);
try {
    If( num2 == 0 )
            throw 0;
    int result = num1 / num2;
    print_record(result);
}
catch( const char *ex ){
    cout<<ex<<endl;  }
    return 0;
}
```

Consider the following code

If we are throwing exception from function, then implementer of function should specify “exception specification list”. The exception specification list is used to specify type of exception function
may throw.
If type of thrown exception is not available in exception specification list and if exception is raised then C++ do execute catch block rather it invokes std::unexpected() function.

```
int calculate(int num1,int num2) throw(const char* ){
    if( num2 == 0 )
            throw "/ by zero  exception";
    return num1 / num2;
}
int main( void ){
    int num1;
    accept_record(num1);
    int num2;
    accept_record(num2);
try{
   int result = calculate(num1, num2 );
    print_record(result);
}
catch( const char *ex ){
cout<<ex<<endl; }
return 0;  }
```

Thank You