C++ Programming

Trainer : Rohan Paramane

Email: rohan.paramane@sunbeaminfo.com

Constant in C++

- We can declare a constant variable that cannot be modified in the app.
- If we do not want to modify value of the variable then const keyword is used.
- constant variable is also called as read only variable.
- The value of such variable should be known at compile time.
- In C++ , Initializing constant variable is mandatory
- const int i=3; //VALID
- Const int val; //Not ok in c++
- Generally const keyword is used with the argument of function to ensure that the variable cannot be modified within that function.

Constant data member

- Once initialized, if we do not want to modify state of the data member inside any member function of the class including constructor body then we should declare data member constant.
- If we declare data member constant then it is mandatory to initialize it using constructors member initializer list.

```
class Test
{
private:
        const int num1;
public:
        Test( void ) : num1( 10 ) //OK
        {
        //this->num1 = 10; //Not OK
        }
};
```

Const member function

- The member function can declared as const. In that case object invoking the function cannot be modified within that member function.
- We can not declare global function constant but we can declare member function constant.
- If we do not want to modify state of current object inside member function then we should declare member function as constant.
- void display() const;
- Even though normal members cannot be modified in const function, but *mutable* data members are allowed to modify.
- In constant member function, if we want to modify state of non constant data member then we should use **mutable keyword.**
- We can not declare following function constant:
  1. Global Function
  2. Static Member Function
  3. Constructor
  4. Destructor

Const object

- If we dont want to modify state of the object then instead of declaring data member constant, we should declare object constant.
- On non constant object, we can call constant as well as non constant member function.
- On Constant object, we can call only constant member functions
- It is C language feature which is used to create alias for existing data type.
- Using typedef, we can not define new data type rather we can give short name / meaningful name to the existing data type.

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;
}
```

Constructor's member initializer list

- If we want to initialize data members according to users requirement then we should use constructor body.

```
class Test
{
private:
        int num1;
        int num2;
        int num3;
public:
        Test( void )
        {
        this->num1 = 10;
        this->num2 = 20;
        this->num3 = num2;
        }
};
```

•If we want to initialize data member according to order of data member declaration then we can use constructors member initializer list.

```
class Test
{
private:
        int num1;
        int num2;
        int num3;
public:
        Test( void ) : num1( 10 ), num2( 20 ), num3( num2)
        { }
};
```

Except array we can initialize any member inside constructors member initializer list.

Copy Constructor

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

Dynamic Memory Allocation

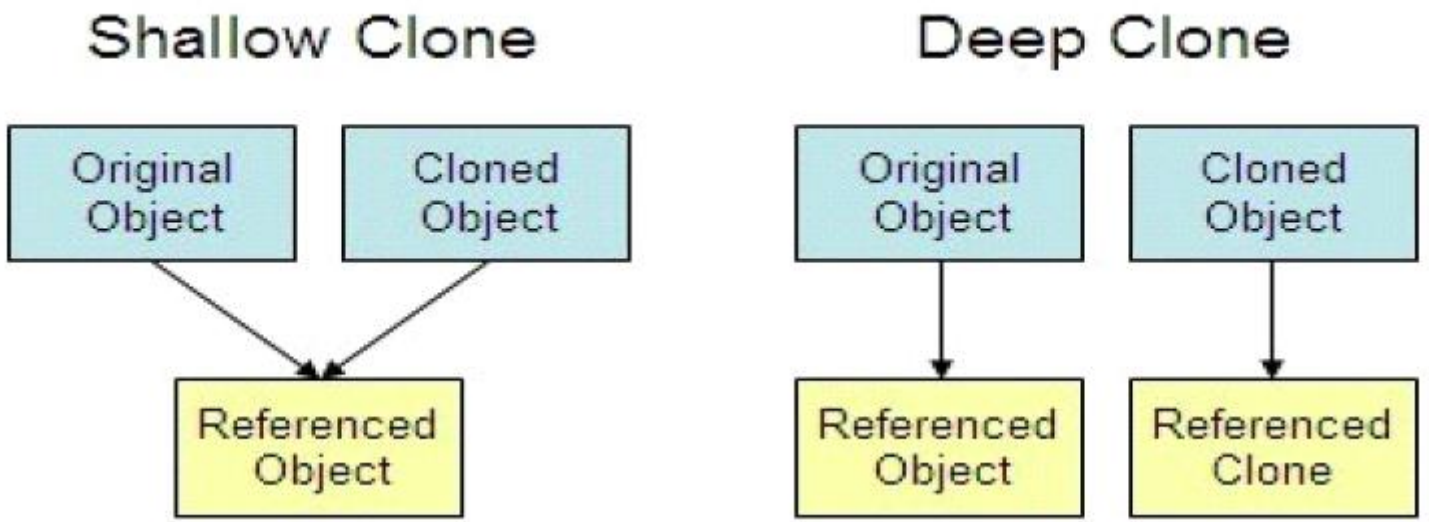
- If we want to allocate memory dynamically then we should use new operator and to deallocate that memory we should use delete operator.
- If pointer contains, address of deallocated memory then such pointer is called dangling pointer.
- When we allocate space in memory, and if we loose pointer to reach to that memory then such wastage of memory is called memory leakage.

- Example :

```
int main()
{
    int *ptr = new int;          //int *ptr = ( int* )::operator new( sizeof( int ) * 1 );
    *ptr = 125;                  //Dereferencing
    cout<<"Value   :             "<<*ptr<<endl; //Dereferencing
    delete ptr;                  //::operator delete( ptr );
    ptr = NULL;
    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.

Thank You