Subject Name: Object Oriented Programming Using C++

Subject Code: BCA-301 N

Subject Topic: Static Keyword in C++

Abhishek Dwivedi

Assistant Professor

Department of Computer Application

UIET, CSJM University, Kanpur

Static Keyword in C++

- Static is a keyword in C++ used to give special characteristics to an element. Static elements are allocated storage only once in a program lifetime in static storage area. And they have a scope till the program lifetime. Static Keyword can be used with following:

1. Static variable in functions
2. Static Class Objects
3. Static Member Variable in class
4. Static Member Methods in class

Static Variables inside Functions

- Static variables when used inside function are initialized only once, and then they hold there value even through function calls.

```
void counter()
{
    static int count=0;
    cout << count++;
 }
void main()
    {
            for(int i=0;i<5;i++)
            {
                counter();
            }
getch();
 }
```

OUTPUT

Output of above program

```
0 1 2 3 4
```

output **without using static** variable

```
0 0 0 0 0
```

- If we do not use static keyword, the variable count, is reinitialized every time when counter() function is called, and gets destroyed each time when counter() functions ends. But, if we make it static, once initialized count will have a scope till the end of main() function and it will carry its value through function calls too.
- If you don't initialize a static variable, they are by default initialized to zero.

Static Class Objects

- Static keyword works in the same way for class objects too. Objects declared static are allocated storage in static storage area, and have scope till the end of program.

- Static objects are also initialized using constructors like other normal objects. Assignment to zero, on using static keyword is only for primitive datatypes, not for user defined datatypes.

Example

```
class Abc
{
      public:
                 Abc()
                 { i=0;
                   cout << "constructor“<<endl;
                 }
                  ~Abc()
                 {
                    cout << "destructor“<<endl;
                  }
};
void main()
 {
      int x=0;
        if(x==0)
                 {
                    static Abc obj;
                 }
                 cout << “End of main“<<endl;
getch();
 }
```

OUTPUT

Output of above program

```
constructor
End of main
destructor
```

output **without using static** variable

```
constructor
destructor
End of main
```

- why was the destructor not called upon the end of the scope of if condition, where the reference of object obj should get destroyed. This is because object was static, which has scope till the program's lifetime, hence destructor for this object was called when main() function exits.

Static Data Member in Class

- Static data members of class are those members which are shared by all the objects. Static data member has a single piece of storage, and is not available as separate copy with each object, like other non-static data members.
- Static member variables (data members) are not initialized using constructor, because these are not dependent on object initialization.
- Also, it must be initialized explicitly, always outside the class.

Example

```
class X
{
    public:
    static int i;
    X()
     {
            // construtor
      }
};
int X::i=1;                    // initialized explicitly
void main()
{
    X obj;
    cout << obj.i;             // prints value of i
   getch();
}
```

References:

- www.studytonight.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- “Object oriented programming in C++”, Robert Lafore
- “Object oriented programming with C++”, E.Balagurusamy