Lecture 6 Static members

19CSE201 Advanced Programming

Difference between C and C++

C	C++
Top down	Bottom up
POP	OOP
I/O statement- printf(),scanf()	cin, cout with >>,<<
Header files-stdio.h	iostream
Top down approach	Bottom up approach
Dynamic memory allocation – malloc, calloc, free	new, delete
No exception handling	Can handle run time error with exception handling
Divided into function	Divided into objects
Importance given to functions	Importance given to data
Developed by Dennis Ritchie in 1972	Developed by Bjarne Stroustrup in 1980

Difference between structure in C and C++

Structure in C	Structure in C++
No functions inside structure	Functions can be declared inside structure
No access specifiers. All data members are considered as public	Access specifiers public, private are used
Cant apply any OOP concepts	all OOP concepts can be applied
To create structure variable struct keyword should be used. Eg: struct stud s1;	No need to use struct keyword eg: stud s1;

Difference between structure and class in C++

Structure in C++	Class in C++
Public by default	Private by default

Static data member

- Data member of a class can be specified as static
- The properties of a static member variable are similar to that of a static variable in C.
- It is initialized to zero when the first object of its class is created. No other initialization is permitted.
- Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.
- It is visible only with in the class but its life time is the entire program.
- Static variables are normally used to maintain values common to the entire class.
- For example a static data member can be used as a counter that records the occurrence of all the objects.

- Type and scope of each static member variable must be defined outside the class definition .
- This is necessary because the static data members are stored separately rather than as a part of an object.
- int item :: count; // definition of static data member
- We can access static member inside the member functions or inside constructor

```
class item
{
static int count;
Int x, y;
//count is static // declaration
public:
   item(){count++;
            cout<<count<<endl;
             }
};
int item :: count;  //static variable definition
int main( )
{   item a,b,c;   }
```

```
class item
{
public:
    static int count; //count is static
   item(){count++;}
};
int item :: count=2;
int main( )
{   item a,b,c;
    cout<<item::count;
    cout<<a.count++;
    cout<<b.count;
}
```

static member function

- By declaring a **function member** as **static**, you make it independent of any particular object of the class.
- A **static member function** can be called even if no objects of the class exist and the **static functions** are accessed using only the class name and the scope resolution operator ::
- A static member function can have access to only other static members declared in the same class.
- A static member function can be called using the class name as follows:-

  classname :: functionname();

```
class test
{
    int code;
    static int count;
    public:
    void setcode(void)
    {
    code=++count;
    }
    void showcode(void)
    {
    cout<<"object member : "<<code<<endl;
    }
    static void showcount(void)
    { cout<<"count="<<count<<endl; }
};
```

```
int test:: count;
int main()
{
    test t1,t2;
    t1.setcode( );
    t2.setcode( );
    test :: showcount ( );
    test t3;
    t3.setcode( );
    test:: showcount( );
    t1.showcode( ) ;
    t2.showcode( );
    t3.showcode( );
}
```

output:

count : 2

count: 3

object number 1

object number 2

object number 3

- In static member function we can't use any other **non static data member.**
- If we try to use any other data member insider static member function, it gives error
- **The following code will give error**

Eg: static void showcount()

{

cout<<code; // code is not static

}

Defining static member function outside the class

```
class test
{
      int code;
      static int count;
      public:
      static void showcount(void);
};
void test::showcount()    // should not use static keyword again.
{ cout<<"count="<<count<<endl; }
```