girmiti software™

Introduction to C++

- Static variables inside function
- Static objects
- Static data members
- Static member functions

Static keyword

Static:
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. And they have a scope till the program lifetime.

Static Keyword can be used with following:

- Static variable in functions
- Static class objects
- Static member variable in class
- Static methods in class

Static variables inside function

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

Syntax:
static data_type var_name=value;

Example:

```
#include<iostream>
using namespace std;
void counter()
{
 static int count=1;
 cout<<++count<<endl;
}
int main()
{
for(int i=0;i<5;i++)
{
counter();
}}
```

Static objects

Static class objects:
Static keyword works in the same way for class objects too.
Objects declared static 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:

```
#include<iostream>
using namespace std;
class Abc
{ int i;
public:
Abc()
{
i=0;
cout << "constructor";
}
~Abc()
{
cout << "destructor";
}
};
```

Static objects

Example:

```
void f()
{
static Abc obj;
}
 int main()
 {
f();
cout << "END";
 }
```

Static data members

Static data members:
Static data members of class are those members which are shared by all the objects and are common to all objects of a class. 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 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. If not initialized, Linker will give error.

A static data member of a class has the following properties:

- The access rule of the data member of a class is same for the static data members also.
- Whenever a static data member is declared and it has only a single copy,it will be shared by all the instances of a class.
- The static data members should be created & initialized before the main() function control block begins.

Syntax:

```
class class_name
{
….
static data_type var_name;
….}
```

Static data members

Example:

```
#include<iostream>
using namespace std;
class sample
{ private:
            static int count;
  public:
            sample();
            void display();
};
int sample::count=1;
sample::sample()
{           count++;
}
void sample::display()
{           cout<<"counter value="<<count<<endl;
}
int main()
{
            sample obj1,obj2,obj3,obj4;
            obj4.display();
}
```

Static member functions

Static member function:

The keyword static is used to precede the member function to make a member function static. These functions work for the class as whole rather than for a particular object of a 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 only access static data member, other static member functions .
- A static member function does not have a this pointer so it can access non static members of its class only by using . or ->

Static member functions

Example:

```
#include<iostream>
using namespace std;
class sample
{
  private:
          static int count;
  public:
          sample();
          static void display();
};
int sample::count=1;
sample::sample()
{
          count++;
}
void sample::display()
{
          cout<<"counter value="<<count<<endl;
}
```

Static member functions

```
int main()
{
        cout<<"before instantiation of object "<<endl;
        sample::display();
        sample obj1,obj2,obj3,obj4;
        cout<<"after instantiation of object "<<endl;
        sample::display();
}
```

Discussions