

GeeksforGeeks

A computer science portal for geeks

[Practice](#)[Login](#)[Write an Article](#)

Scope of
Variables in
C++

Sorting a
vector in C++

vector
insert()
function in
C++ STL

string find in
C++

map insert()
in C++ STL

map find()
function in
C++ STL

set find()
function in
C++ STL

Check if a
given graph
is Bipartite
using DFS

How to
return
multiple



values from
a function in
C or C++?

Sum of array
Elements
without
using loops
and
recursion

Pre-
increment
and Post-
increment in
C/C++

static_cast in
C++ | Type
Casting
operators

map count()
function in
C++ STL

Applications
of Pointers in
C/C++

set insert()
function in
C++ STL

vector
rbegin() and
rend()
function in
C++ STL

map erase()



function in
C++ STL

“static const”
vs “#define”
vs “enum”

Memory leak
in C++ and
How to avoid
it?

Why strcpy
and strncpy
are not safe
to use?

Loader in
C/C++

std::any
Class in C++

How to find
Segmentation
Error in C &
C++ ? (Using
GDB)

vector
emplace()
function in
C++ STL

Types of
Operator
Overloading
in C++

set
lower_bound()



function in
C++ STL

Why array
index starts
from zero ?

Modulus
function in
C++ STL

multimap
insert() in
C++ STL

Check if X
can give
change to
every person
in the Queue





Versatile.
Like you.

Switch between
multiple modes.
Seamlessly.

Find out more



Lapt



Tabl



Stud



Static Keyword in C++

Prerequisite : [Static variables in C](#)

Static keyword has different meanings when used with different types. We can use static keyword with:

Static Variables : Variables in a function, Variables in a class

Static Members of Class : Class objects and Functions in a class

WE NEED ENGINEERS TO HELP US DESIGN A BETTER FINANCIAL FUTURE

Let us now look at each one of these use of static in details:

Static Variables

- **Static variables in a Function:** When a variable is declared as static, space for it gets allocated for the lifetime of the program. Even if the function is called multiple times, space for the static variable is allocated only once and the value of variable in the previous call gets carried through the next function call. This is useful for implementing **coroutines in C/C++** or any other application where previous state of function needs to be stored.

```
// C++ program to demonstrate
// the use of static Static
// variables in a Function
#include <iostream>
#include <string>
using namespace std;

void demo()
{
    // static variable
    static int count = 0;
    cout << count << " ";

    // value is updated and
    // will be carried to next
    // function calls
    count++;
}

int main()
{
    for (int i=0; i<5; i++)
        demo();
    return 0;
}
```

Output:



0 1 2 3 4

You can see in the above program that the variable count is declared as static. So, its value is carried through the function calls. The variable count is not getting initialized for every time the function is called.

As a side note, **Java doesn't allow static local variables in functions.**

- **Static variables in a class:** As the variables declared as static are initialized only once as they are allocated space in separate static storage so, the static variables **in a class are shared by the objects**. There can not be multiple copies of same static variables for different objects. Also because of this reason static variables can not be initialized using constructors.

```
// C++ program to demonstrate static
// variables inside a class

#include<iostream>
using namespace std;


class GfG
{
    public:
        static int i;

        GfG()
        {
            // Do nothing
        };
};


int main()
{
    GfG obj1;
    GfG obj2;
    obj1.i =2;
    obj2.i = 3;

    // prints value of i
    cout << obj1.i<<" "<<obj2.i;
}
```


You can see in the above program that we have tried to create multiple copies of the static variable i for multiple objects. But this didn't happen. So, a static variable inside a class should be initialized explicitly by the user using the class name and scope resolution operator outside the class as shown below:



```
// C++ program to demonstrate static
// variables inside a class
```



```
#include<iostream>
using namespace std;
```



```
class GfG
{
public:
    static int i;

    GfG()
    {
        // Do nothing
    };
};

int GfG::i = 1;

int main()
{
    GfG obj;
    // prints value of i
    cout << obj.i;
}
```

Output:





1

Static Members of Class

- **Class objects as static:** Just like variables, objects also when declared as static have a scope till the lifetime of program.

Consider the below program where the object is non-static.





```
// CPP program to illustrate
// when not using static keyword
#include<iostream>
using namespace std;

class GfG
{
    int i;
public:
    GfG()
    {
        i = 0;
        cout << "Inside Constructor\n";
    }
    ~GfG()
    {
        cout << "Inside Destructor\n";
    }
};





int main()
{
    int x = 0;
    if (x==0)
    {
        GfG obj;
    }
    cout << "End of main\n";
}
```

Output:

```
Inside Constructor
Inside Destructor
End of main
```

In the above program the object is declared inside the if block as non-static. So, the scope of variable is inside the if block only. So when the object is created the constructor is invoked and soon as the control of if block gets over the destructor is invoked as the scope of object is inside the if block only where it is declared. Let us now see the change in output if we declare the object as static.





```
// CPP program to illustrate
// class objects as static
#include<iostream>
using namespace std;

class GfG
{
    int i = 0;

    public:
    GfG()
    {
        i = 0;
        cout << "Inside Constructor\n";
    }

    ~GfG()
    {
        cout << "Inside Destructor\n";
    }
};

int main()
{
    int x = 0;
    if (x==0)
    {
        static GfG obj;
    }
    cout << "End of main\n";
}
```


Output:

```
Inside Constructor
End of main
Inside Destructor
```

You can clearly see the change in output. Now the destructor is invoked after the end of main. This happened because the scope of static object is through out the life time of program.

- **Static functions in a class:** Just like the static data members or static variables inside the class, static member functions also does not depend on object of class. We are allowed to invoke a static member function using the object and the `.*` operator but it is recommended to invoke the static members using the class name and the scope resolution operator.

Static member functions are allowed to access only the static data members or other static member functions, they can not access the non-static data members or member functions of the class.





```
// C++ program to demonstrate static
// member function in a class
#include<iostream>
using namespace std;

class GfG
{
    public:

        // static member function
        static void printMsg()
        {
            cout<<"Welcome to GfG!";
        }
};

// main function
int main()
{
    // invoking a static member function
    GfG::printMsg();
}
```



Output:

Welcome to GfG!

Related Articles:

- [Quiz on Static Keyword](#)
- [Static data members in C++](#)
- [When are static objects destroyed?](#)
- [Interesting facts about static member functions](#)
- [Can static functions be virtual?](#)
- [Comparison of static keyword in C++ and Java](#)
- [Static functions in C](#)

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or email your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Recommended Posts:

- [C++ | Static Keyword | Question 1](#)
- [C++ | Static Keyword | Question 2](#)
- [C++ | Static Keyword | Question 6](#)
- [C++ | Static Keyword | Question 3](#)
- [C++ | Static Keyword | Question 5](#)
- [C++ | Static Keyword | Question 4](#)
- [Comparison of static keyword in C++ and Java](#)
- [Output of Java Programs | Set 48 \(Static keyword\)](#)
- [Understanding "static" in "public static void main" in Java](#)
- [override Keyword in C++](#)
- [Use of explicit keyword in C++](#)
- [C++ mutable keyword](#)
- [Function overloading and const keyword](#)
- [Static data members in C++](#)
- [When are static objects destroyed?](#)



Article Tags : [C++](#) [C++-Static Keyword](#) [Static Keyword](#)

Practice Tags : [CPP](#)



1



2.6☐ To-do ☐ DoneBased on **8** vote(s)

Feedback

Add Notes

Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!

GeeksforGeeks
A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview
Experience
Internships
Videos

@geeksforgeeks, Some rights reserved

