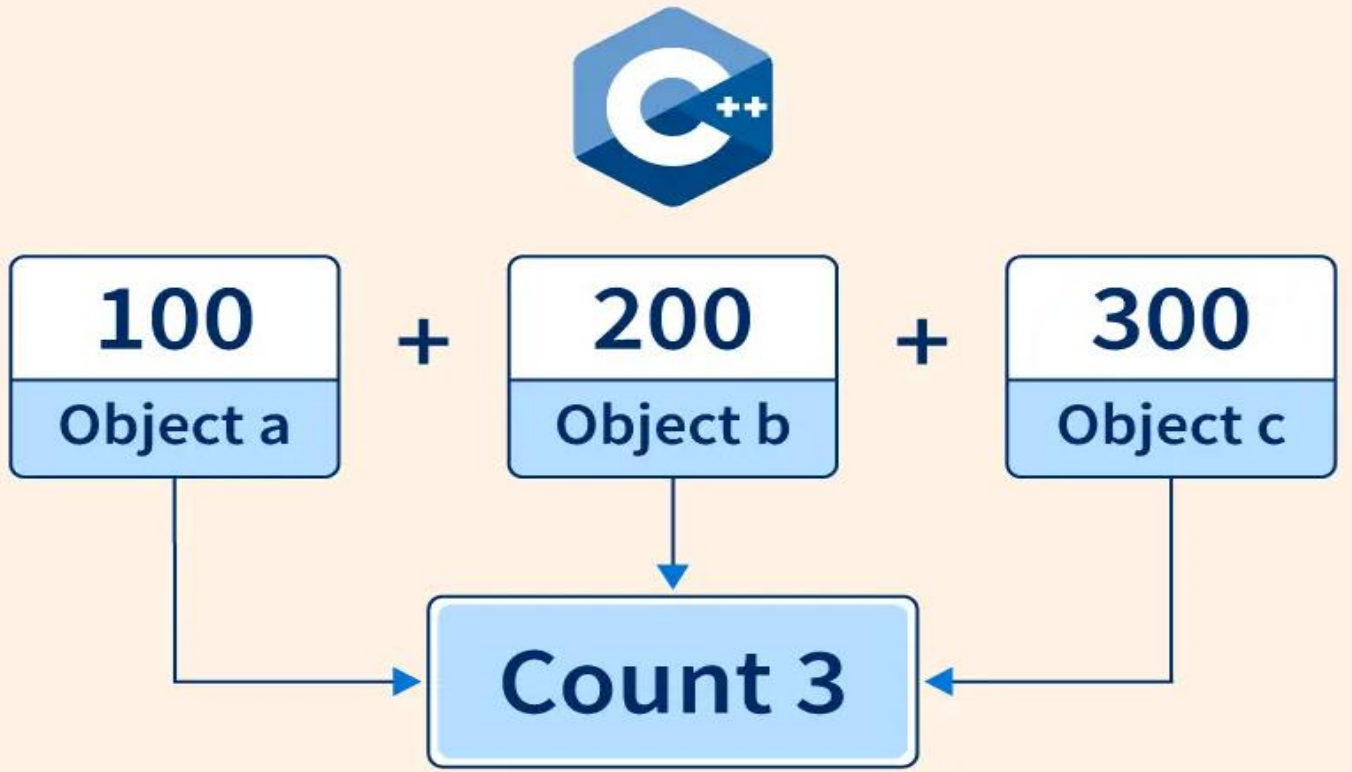
Static Data Member in C++



By – Mohammad Imran

Static Members

Introduction

A static data member in C++ is a class member that is shared by all objects of the class. There is only one copy of a static data member, even if multiple objects of the class are created. Static data members are initialized before any object of the class is created.

A static data member in C++ can be defined by using the **static** keyword.

Syntax

```
static data_type member_name
```

By – Mohammad Imran

Static Members

Introduction

A static data member in C++ is a data member defined in a class that is not instantiated with each object created of the class. Data members defined in a class are usually instantiated with every object created of the class. That is, each object of the class will have their own instances or copies of the data members defined in the class. However, if a data member is initialized with the static keyword, that data member will not be instantiated and there shall exist only one copy of that data member for all objects or instances of the class.

By – Mohammad Imran

Static Members

Introduction

Static data members are class members that are declared using **static** keywords. A static member has certain special characteristics which are as follows:

- ✓ 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 initialized before any object of this class is created, even before the main starts.
- ✓ It is visible only within the class, but its lifetime is the entire program.

By – Mohammad Imran

Static Members

Example - 1

```
#include<iostream>
using namespace std;
class Stat
{
   static int x;
  public:
   void display()
   {
      cout << "x = "<< x << endl;
      x++;
   }
};

int Stat::x = 100;

int main()
{
   Stat st;
   st.display();
   st.display();
   return 0;
}
```

OUTPUT

```
x = 100
x = 101
```

By – Mohammad Imran

Static Members Function

The static member functions are special functions used to access the static data members or other static member functions. A member function is defined using the static keyword. A static member function shares the single copy of the member function to any number of the class' objects. We can access the static member function using the class name or class' objects.

Syntax -

```
static data_type function_name(parameter);
```

By – Mohammad Imran

Static Members Function

Program - 2

```
#include <iostream>
using namespace std;
class Note
{
   static int num;
   public:
   static int func()
   {
      return num;
   }
};

int Note :: num = 5;

int main()
{
   int n = Note::func();
   cout << " Num = " << n << endl;
   return 0;
}
```

OUTPUT

```
Num = 5
```

By – Mohammad Imran

Static Members

When we create static members and static functions in the class those functions can be called using object of the class as well as directly with class name with scope resolution operator.

Let’s see an example:

Static Members

Variables and Functions

Program - 3

```
#include <iostream>
using namespace std;
class Member
{
  private:
   static int A;
   static int B;
   static int C;
  public:
   static void disp()
   {
      cout << "A = " << A << endl;
      cout << "B = " << B << endl;
      cout << "C = " << C << endl;
   }
};
```

By – Mohammad Imran

Static Members

Variables and Functions

Program - 3

```
int Member :: A = 20;
int Member :: B = 30;
int Member :: C = 40;

int main ()
{
   Member mb;
   cout << "By Object" << endl;
   mb.disp();
   cout << "By Class" << endl;
   Member::disp();
   return 0;
}
```

OUTPUT

```
By Object
  A = 20
  B = 30
  C = 40
By Class
  A = 20
  B = 30
  C = 40
```

By – Mohammad Imran

Static Objects

An object becomes static when a ***static*** keyword is used in its declaration. Static objects are initialized only once and live until the program terminates. They are allocated storage in the data segment or BSS segment of the memory.

C++ supports two types of static objects:

- ✓ **Local Static Objects**
- ✓ **Global Static Objects.**

Static Objects

Local

Local static objects in C++ are those static objects that are declared inside a block. Even though their lifespan is till the termination of the program, their scope is limited to the block in which they are declared.

By – Mohammad Imran

Local Static Objects

Example - 4

```
#include <iostream>
using namespace std;
class Test
{
   public:
   void disp()
   {
      cout << "Hello\n";
   }
};
void myfunc()
{
   static Test obj; //Local
   obj.disp();
}
int main()
{
      myfunc();
      return 0;
}
```

OUTPUT

```
Hello
```

By – Mohammad Imran

Static Objects

Global

Global static objects are those objects that are declared outside any block. Their scope is the whole program and their lifespan is till the end of the program.

By – Mohammad Imran

Global Static Objects

Example - 5

```
#include <iostream>
using namespace std;
class Test
{
  public:
   int a;
   void disp()
   {
      a = 10;
      cout << "Hello\n";
   }
};
static Test obj; //Global

int main()
{
      obj.disp();
      cout << obj.a;
      return 0;
}
```

OUTPUT

```
Hello
10
```

By – Mohammad Imran

Constants in C++

By – Mohammad Imran

Constant Members

In C++, constants are values that remain fixed throughout the execution of a program.

Constants in C++ refer to variables with fixed values that cannot be changed. Once they are defined in the program, they remain constant throughout the execution of the program. They are generally stored as read-only tokens in the code segment of the memory of the program. They can be of any available data type in C++ such as int, char, string, etc.

By – Mohammad Imran

Constant Members

In C++, constants can be

- ✓ Variables
- ✓ Functions
- ✓ Pointers

By – Mohammad Imran

Constant Members

Variables

There are two way to declare constant variable in C++.

- ✓ Using **`#define`** Preprocessor Directive
- ✓ Using **`const`** keyword

Constant Members

Key Points

The following are some major properties of C++ constants:

- ✓ Constants are the variables with fixed values.
- ✓ We have to initialize the constants at the time of declaration and we cannot change this value later in the program.
- ✓ **const** and **constexpr** can be used to define a constant.
- ✓ **#define** is also used to define a **macro constant**.

By – Mohammad Imran

Constant Variables

Using const Keyword

Defining constants using const keyword is one of the older methods that C++ inherited from C language. In this method, we add the const keyword in the variable definition as shown below:

Syntax

```
const DATATYPE variable_name = value;
```

Example

```
const int PI = 3.14;
```

By – Mohammad Imran

Constant Variables

Example - 6

```
#include <iostream>
using namespace std;
class ConstVar
{
  public:
   int hour;
   void hourtomin()
   {
      const int min = 60;
      cout << "Enter hour : ";
      cin >> hour;
      int totMin = min * hour;
      cout << "Total Minute = " << totMin;
   }
};
```

OUTPUT

```
Enter hour : 5
Total Minute = 300
```

```
int main()
{
   ConstVar cv;
   cv.hourtomin();
}
```

By – Mohammad Imran

Constant Member Function

Constant member functions are those functions that are denied permission to change the values of the data members of their class. To make a member function constant, the keyword **const** is appended to the function prototype and also to the function definition header.

We use const member functions in C++ to avoid accidental object changes. It is highly recommended to make as many functions **const** as possible so that the chances of error in the program are minimal.

By – Mohammad Imran

Constant Member Function

The **const** member function can be defined in three ways:

1. For function declaration within a class.

```
return_type function_name() const;
```

```
Example: int get_Data() const;
```

Constant Member Function

2. For function definition within the class declaration.

```
return_type function_name() const
{
   //function body;
}
```

Example:

```
int get_Data() const
{
   //function body;
}
```

Constant Member Function

3. For function definition outside the class.

```
return_type className::function_name() const
{
   //function body;
}
```

Example:

```
int Demo::get_Data() const
{
   //function body;
}
```

Constant Member Function

Key Points

- ✓ When a function is declared as const, it can be called on any type of object, const object as well as non-const objects.
- ✓ Whenever an object is declared as const, it needs to be initialized at the time of declaration. however, the object initialization while declaring is possible only with the help of constructors.
- ✓ A function becomes const when the const keyword is used in the function's declaration. The idea of const functions is not to allow them to modify the object on which they are called.
- ✓ It is recommended practice to make as many functions const as possible so that accidental changes to objects are avoided.

By – Mohammad Imran

Constant Member Function

Example - 7

```
#include <iostream>
using namespace std;
class Demo
{
   int x;
  public:
   void set_data(int a)
   {
      x = a;
   }
   int get_data() const
   {
      ++x;
      return x;
   }
};
int main()
{
   Demo d;
   d.set_data(10);
   cout << d.get_data();
   return 0;
}
```

OUTPUT

Error

By – Mohammad Imran

Constant Member Function

Arguments

If you pass an argument to the non-const function by reference and you don't want that, the function changes the value of the variable. So we add the const keyword to the function's argument so that the function does not change its value. In the program given below, we add the const keyword to variables a and b so that the get_value() function can never manipulate the values of the variables.

By – Mohammad Imran

Constant Argument Member Function

Example - 8

```
#include <iostream>
using namespace std;
class Demo
{
   int x;
  public:
   void set_data(const int a)
   {
      x = ++a;
   }
   int get_data()
   {
      ++x;
      return x;
   }
};
```

```
int main()
{
   Demo d;
   d.set_data(10);
   cout << d.get_data();
   return 0;
}
```

OUTPUT

Error

By – Mohammad Imran

Constant Object

When we use the **const** keyword to build an object, the value of data members can never change during the object's lifetime in a program. **const** objects are also referred to as read-only objects.

Constant Object

Example - 9

```
#include<bits/stdc++.h>
using namespace std;
class Example
{
  public:
   int x;
   void setValue(int a)
   {
      x = a;
   }
   int get_value() const
   {
      return x;
   }
};
```

```
int main()
{
   const Example obj;
   obj.x = 10;
   obj.setValue(100);
   cout << obj.get_value();
   return 0;
}
```

OUTPUT

Error

By – Mohammad Imran