



# Bahria University, Islamabad

## Department of Software Engineering

Object Oriented Programming Lab (Spring-2025)

Teacher: Engr. Muhammad Faisal Zia

---

Student: Ahtisham khan

Enrollment: 09-131242-014

---

Lab Journal: 4

Date: 05/03/2025

---

Task No:	Task Wise Marks		Documentation Marks		Total Marks (20)
	Assigned	Obtained	Assigned	Obtained	
1	2		4		
2	2				
3	2				
4	5				
5	5				

**Comments:**

**Signature**

# Lab # 4

## Static Variable and Static Functions in C++

### Objective:

The objective of this lab is to understand the concept of **static variables** and **static functions** in C++. By the end of this lab, you should be able to:

1. Define and use static data members in a class.
  2. Define and use static member functions in a class.
  3. Understand how static members are shared across all objects of a class.
  4. Apply static members in real-world scenarios like object counting and Fibonacci series.
- 

### 1. Static Variables

- Class members can be defined as **static** using the static keyword.

```
class Example {  
public:  
    static int count; // Static member declaration  
};
```

```

#include <iostream>
using namespace std;

class Example {
public:
    static int count; // Static member declaration
    Example() { count++; }
};

int Example::count = 0; // Static member initialization

int main() {
    Example obj1, obj2, obj3;
    cout << "Count: " << Example::count << endl; // Output: Count: 3
}

```

- A **static member** means that no matter how many objects of the class are created, there is only **one copy** of the static member.
- **Shared by all objects** of the class.

```

#include <iostream>
using namespace std;

class Example {
public:
    static int count;
    void increment() { count++; }
};

int Example::count = 0;

int main() {
    Example obj1, obj2;
    obj1.increment();
    obj2.increment();
    cout << "Count: " << Example::count << endl; // Output: Count: 2
}

```

- **Initialized to zero** when the first object is created, unless explicitly initialized.

```

#include <iostream>
using namespace std;

class Example {
public:
    static int count;
};

int Example::count; // Default initializes to 0

int main() {
    cout << "Initial Count: " << Example::count << endl; // Output: Initial Count: 0
}

```

- Cannot be initialized inside the class definition.

```

class Example {
public:
    static int count; // Declaration only
    // int count = 5; // ❌ Error! Cannot initialize inside the class
};

```

- Must be **initialized outside the class** using the **scope resolution operator (::)** to specify which class it belongs to.

```

#include <iostream>
using namespace std;

class Example {
public:
    static int count; // Declaration
};

int Example::count = 10; // Initialization outside the class

int main() {
    cout << "Static Count: " << Example::count << endl; // Output: Static Count: 10
}

```

## 2. Static Functions


- Declaring a function as **static** makes it independent of any object of the class.

```
class Example {
public:
    static void showMessage() { // Static member function
        cout << "Static function called!" << endl;
    }
};
```

- A **static member function** can be called even if no objects of the class exist.

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


class Example {
public:
    static void showMessage() {
        cout << "Static function called without creating an object!" << endl;
    }
};

int main() {
    Example::showMessage(); //  Calling without an object
}
```

- Static functions are accessed using only the **class name** and the **scope resolution operator (::)**.

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

class Example {
public:
    static void display() {
        cout << "Accessing static function using scope resolution operator!" << endl;
    }
};

int main() {
    Example::display(); //  Accessing using ClassName::FunctionName()
}
```

- A static member function can only access **static data members** and **other static functions** of the class.

```

#include <iostream>
using namespace std;

class Example {
private:
    static int count; // Static data member

public:
    static void increment() {
        count++; // ✅ Allowed (Accessing static data member)
    }

    static int getCount() {
        return count; // ✅ Allowed (Accessing another static function)
    }
};

int Example::count = 0; // Initialize static member

int main() {
    Example::increment(); // ✅ Calling static function
    cout << "Count: " << Example::getCount() << endl; // Output: Count: 1
}

```

- **Static member functions have class scope** and do not have access to the **this** pointer.

```

#include <iostream>
using namespace std;

class Example {
public:
    int value; // Non-static member variable

    static void show() {
        // cout << value; ❌ Error! Cannot access non-static member
        cout << "Static function has no 'this' pointer!" << endl;
    }
};

int main() {
    Example::show();
}

```

- A static function can be used to **check if objects of the class have been created**.

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

class Example {
private:
    static int objectCount; // Tracks number of objects

public:
    Example() { objectCount++; } // Constructor increments count
    ~Example() { objectCount--; } // Destructor decrements count

    static bool hasObjects() {
        return objectCount > 0; //  Checking if any objects exist
    }
};

int Example::objectCount = 0; // Initialize static member

int main() {
    cout << "Objects exist? " << (Example::hasObjects()) ? "Yes" : "No" << endl;

    Example obj1, obj2; // Creating objects

    cout << "Objects exist? " << (Example::hasObjects()) ? "Yes" : "No" << endl;

    return 0;
}
```

### Example:

Program demonstrates the use of static variables and a static function in a C++ class. The class Student has a private static integer variable x, which is initialized to 0. It also has two private non-static member variables, a string name and an integer age. The class has a constructor that takes two arguments, name and age, and initializes these member variables. The constructor also increments the static variable x by 1. The class has a member function print() that prints the name and age of the student.

The class also has a static member function getCount() that returns the value of the static variable x. This function does not have access to any non-static member variables or functions of the class, since it is a static member function.

In the main() function, three objects of the student class are created using the constructor. The print() function is called for each object to print their details. Finally, the static function getCount() is called using the class name to print the total number of students created.

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

```

//Static variables and Static Function
class Student
{
private:
    static int x;
    string name;
    int age;
public:
    Student(string n,int a)
    {
        x++;
        name=n;
        age=a;
    }
    void print()
    {
        cout<<"Name : "<<name<<endl;
        cout<<"Age : "<<age<<endl;
        cout<<"\n";
    }
    static int getCount()
    {
        return x;
    }
};
//Static Variable
int Student::x=0;
int main()
{
    Student s1("Ali",25);
    Student s2("Ahmed",22);
    Student s3("Bilal",12);
    s1.print();
    s2.print();
    s3.print();
    cout<<"Total Students : "<<Student::getCount()<<endl;
    return 0;
}

```

#### Output

```

Name : Ali
Age  : 25

Name : Ahmed
Age  : 22

Name : Bilal
Age  : 12

Total Students : 3

```



```
Ahtisham Khan (014) OOP LAB 4
Name: Maha
Age: 18

Name: Sufyan
Age: 19

Name: Sundas
Age: 20

Name: Umer
Age: 20

Total objcounts: 4

Press any key to continue . . . _
```

---

## Lab Activities:

### Activity 1: Object Counting Using Static Variable

Create a class Alpha that has:

- A static data member total to keep track of the number of objects created.
- A data member id to store the unique ID of each object.
- A constructor to increment the total and assign a unique id to each object.
- A static function showTotal() to display the total number of objects.
- A member function showId() to display the id of the object.

The variable total keeps the track of how many objects of the class are there. Use constructor to increment the value of total. The showId() function prints out the id of its object.

In the main program first define one object and print out the value of total. Then define two more objects and again print out the value of total. Also print out the id number of individual members by calling the showId() function in the main program.

The output of the program should be like:

```
Total objects created: 1
Object ID: 1
Total objects created: 3
Object ID: 2
Object ID: 3
```

**Code:**

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

class Alpha {
private:
    static int total; // Static data member
    int id;           // Non-static data member

public:
    // Constructor
    Alpha() {
        total++;
        id = total;
    }

    // Static function to display total objects
    static void showTotal() {
        cout << "Total objects created: " << total << endl;
    }

    // Function to display object ID
    void showId() {
        cout << "Object ID: " << id << endl;
    }
};

// Initialize static variable
int Alpha::total = 0;

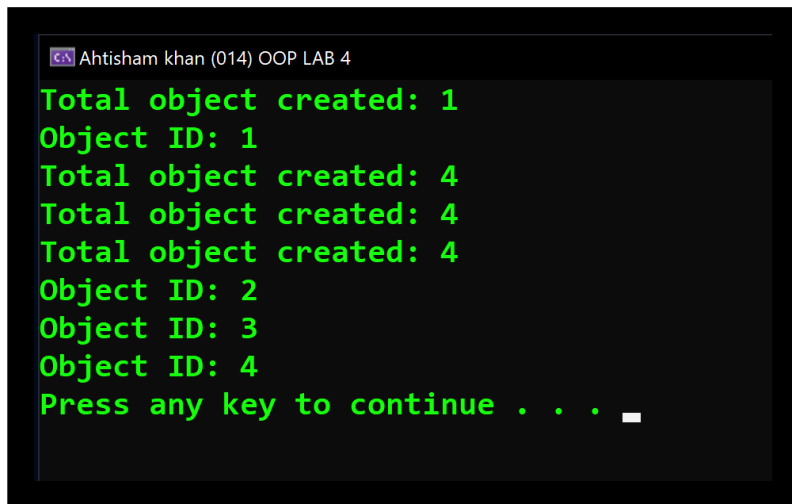
int main() {
    Alpha obj1;
    Alpha::showTotal(); // Call static function
    obj1.showId();

    Alpha obj2, obj3;
    Alpha::showTotal(); // Call static function
    obj2.showId();
    obj3.showId();

    return 0;
}
```

## Your Output Here....

In case of output snippet please make sure output snippet contains student name and id. `AliAhmed_123_Lab04_A1.exe`

A screenshot of a C++ IDE window titled "Ahtisham khan (014) OOP LAB 4". The output console shows the following text in green: "Total object created: 1", "Object ID: 1", "Total object created: 4", "Total object created: 4", "Total object created: 4", "Object ID: 2", "Object ID: 3", "Object ID: 4", and "Press any key to continue . . . \_".

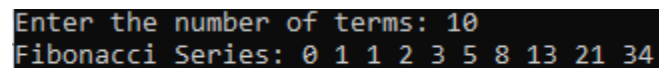
```
C:\Ahtisham khan (014) OOP LAB 4
Total object created: 1
Object ID: 1
Total object created: 4
Total object created: 4
Total object created: 4
Object ID: 2
Object ID: 3
Object ID: 4
Press any key to continue . . . _
```

---

## Activity 2: Fibonacci Series Using Static Variable

Write a program to print the Fibonacci series using a static variable. The output of program should be like this

OUTPUT:

A screenshot of a program output showing the input "Enter the number of terms: 10" and the resulting Fibonacci series "Fibonacci Series: 0 1 1 2 3 5 8 13 21 34".

```
Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
```

Code:

```

#include <iostream>
using namespace std;

class Fibonacci {
private:
    static int a, b; // Static variables to store Fibonacci numbers

public:
    // Static function to generate Fibonacci series
    static void generate(int n) {
        for (int i = 0; i < n; i++) {
            cout << a << " ";
            int next = a + b;
            a = b;
            b = next;
        }
        cout << endl;
    }
};

// Initialize static variables
int Fibonacci::a = 0;
int Fibonacci::b = 1;

int main() {
    int n;
    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Fibonacci Series: ";
    Fibonacci::generate(n); // Call static function

    return 0;
}

```

### Your Output Here....

In case of output snippet please make sure output snippet contains student name and id. `AliAhmed_123_Lab04_A2.exe`

```
Enter the number of terms:
14
Fibonnaci Sequence:
0 1 1 2 3 5 8 13 21 34 55 89 144 233
Press any key to continue . . . _
```

---

### Advantages of Static Data Members

- **Shared Across Objects:** A single copy is shared among all instances of the class, reducing memory usage.
- **Retains Value Across Objects:** The value of a static data member persists throughout the program.
- **Encapsulation:** Can be private, protecting access while allowing controlled modification through static functions.
- **Useful for Counters:** Commonly used to track the number of objects created in a class.

### Advantages of Static Member Functions

- **Can Be Called Without Objects:** Static functions can be called using the class name, even before any objects are created.
- **Access to Static Data Members:** They can manipulate static data members, maintaining class-wide information.
- **Encapsulation and Security:** Useful for utility functions that don't depend on object-specific data.
- **No Need for Object Instantiation:** Saves memory and execution time by avoiding unnecessary object creation.

---

### Conclusion:

In this lab, we learned:

1. How to define and use static data members and static functions in a class.
2. How static members are shared across all objects of a class.
3. Practical applications of static members, such as object counting and Fibonacci series generation.

---

### Additional Notes:


- Static variables are useful for maintaining shared data across all objects of a class.
- Static functions are useful for operations that do not depend on specific objects.
- Always initialize static variables outside the class definition.

## Lab Tasks/ Homework

**Task 01:** Create a class SavingsAccount that uses a static data member to store the annualInterestRate for all savers. Each object of the class should contain a private data member savingBalance to store the amount the saver currently has deposited. Provide the following member functions:

1. A constructor to initialize the savingBalance.
2. A static function setAnnualInterestRate() to set the annualInterestRate.
3. A member function monthlyInterest() to calculate the monthly interest by multiplying the savingBalance by annualInterestRate divided by 12.
4. A member function displayBalance() to display the current balance after adding the monthly interest.

Output should be like this

 Microsoft Visual Studio Debug Console

```
Saver 1:
Current Balance: $2005
Saver 2:
Current Balance: $3007.5

After changing interest rate:
Saver 1:
Current Balance: $2011.68
Saver 2:
Current Balance: $3017.53
```

**Solution:**

Enter your code here...

```
#include<iostream>
using namespace std;
class SavingAccounts{
private:
    double savingbalance;
    static double annualinterestRate;
public:
    SavingAccounts(double startbalance) {
        savingbalance = startbalance;
    }
    static void setannualinterestRate(double rate) {
```

```

        annualinterestRate = rate;
    }
    void monthlyinterest() {
        savingbalance += savingbalance * (annualinterestRate / 12);
    }
    void displaybalance() {
        cout << "Current Total Balance $: " << savingbalance <<
endl;
    }
};
double SavingAccounts::annualinterestRate = 0;
int main() {
    system("title Ahtisham khan (014) OOP LAB 4");
    SavingAccounts Account1(1200.8);
    SavingAccounts Account2(1400.9);

    SavingAccounts::setannualinterestRate(0.2);
    cout << "Account 1: " << endl;
    Account1.monthlyinterest();
    Account1.displaybalance();
    cout << "Account 2: " << endl;
    Account2.monthlyinterest();
    Account2.displaybalance();
    cout << endl;
    cout << "After Changing the AnnualInterestRate: " << endl;
    SavingAccounts::setannualinterestRate(0.5);
    cout << "Account 1: " << endl;
    Account1.monthlyinterest();
    Account1.displaybalance();
    cout << "Account 2: " << endl;
    Account2.monthlyinterest();
    Account2.displaybalance();

    system("pause");
}

```

**Output:**

Your Output here...

```
Ahtisham khan (014) OOP LAB 4
Account 1:
Current Total Balance $: 1220.81
Account 2:
Current Total Balance $: 1424.25

After Changing the AnnualInterestRate:
Account 1:
Current Total Balance $: 1271.68
Account 2:
Current Total Balance $: 1483.59
Press any key to continue . . .
```

**Task 02:** Write a program that sums integers, using static variables. The **output** of the program should be like:

```
Microsoft Visual Studio Debug Console
Enter a number: 10
Your current total: 10
Enter a number: 20
Your current total: 30
Enter a number: 30
Your current total: 60
Enter a number: 40
Your current total: 100
Enter a number: 50
Your current total: 150
```

**Solution:**

```
Enter your code here...
#include<iostream>
using namespace std;
class Totalsum {
private:
    //static variable
    static int total;
    int num;
public:
    Totalsum(int n) {
```



```

        num = n;
        //total = num;
    }
    // create a member funtion to enter the numbers
    void enternumber() {
        cout << "Enter the Number: " << endl;
        cin >> num;
        total += num;

    }
    //create the static member function to display total
    static void displaytotal() {
        cout << "Your current Total: " << total << endl;
    }
};
// initilize total=0
int Totalsum::total = 0;
int main() {
    system("title Ahtisham khan (014) OOP LAB 4");
    Totalsum s1(0);
    for (int i = 0; i < 5; i++) {
        s1.enternumber();
        s1.displaytotal();
    }

    system("pause");
    return 0;
}

```

## Output:

Your Output here...

```
Ahtisham khan (014) OOP LAB 4
Enter the Number:
12
Your current Total: 12
Enter the Number:
29
Your current Total: 41
Enter the Number:
30
Your current Total: 71
Enter the Number:
300
Your current Total: 371
Enter the Number:
32
Your current Total: 403
Press any key to continue . . .
```