

## Department of Electronics & Telecommunication Engineering

### Name of the Experiment:

Static Data Member, and Function Overloading in C++.

**Course No.** : CSE-284  
**Course Title** : Object Oriented Programming Sessional  
**Experiment No.** : 03  
**Date of Exp.** : 23-10-2024  
**Date of Submission** : 30-10-2024

### Remarks :

--

**Name** : Tasnia Afrin  
**ID** : 2108050  
**Level** : 02  
**Term** : 02  
**Group** : 02

## Experiment Name :

Static Data Member, and Function Overloading in C++.

## Objectives:

- Introduce with the Static Data Member and Member function.
- Understand the concept of function overloading in C++.

## Example-1 :

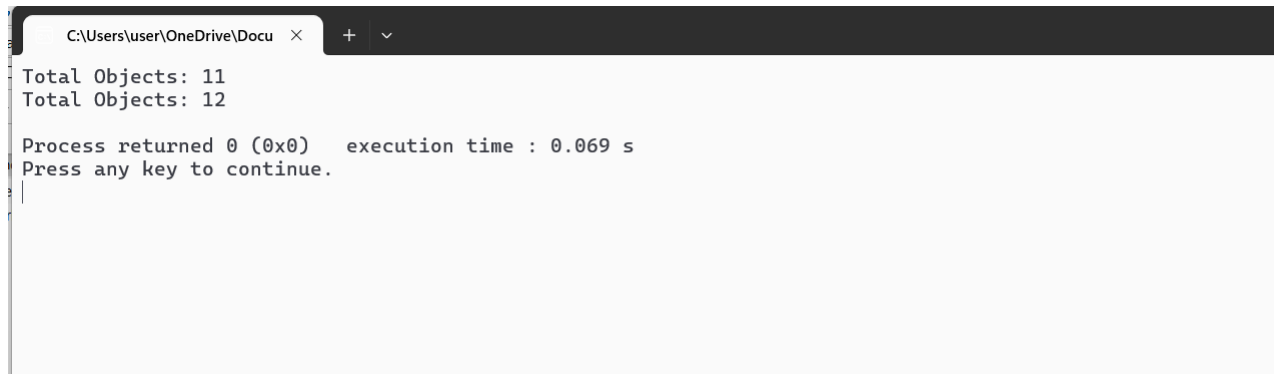
A C++ program to demonstrate the use of static data member.

### Input:

```
#include<iostream>
using namespace std;
class Square{
private:
    int side;
public:
    static int objectCount;
    Square()
    {
        objectCount++;
    }
};

int Square::objectCount=10;
int main(){
    Square a1;
    cout<<"Total Objects: "<<Square::objectCount<<endl;
    Square a2;
    cout<<"Total Objects: "<<Square::objectCount<<endl;
}
```

## Output :



```
C:\Users\user\OneDrive\Docu X + v
Total Objects: 11
Total Objects: 12

Process returned 0 (0x0)   execution time : 0.069 s
Press any key to continue.
```

## Example-2 :

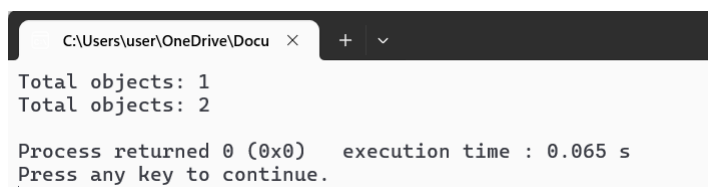
A C++ program to demonstrate the use of static member function.

### Input:

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

class Square {
private:
    int side; // normal data member
    static int objectCount; // static data member
public:
    Square()
    {
        // Increase every time object is created
        objectCount++;
    }
    // creating a static function that returns static data member
    static int getCount() {
        return objectCount;
    }
};
// Initialize static member of class Square
int Square::objectCount = 0;
int main() {
    Square s1;
    cout << "Total objects: " << Square::getCount() << endl;
    Square s2;
    cout << "Total objects: " << Square::getCount() << endl;
    return 0;
}
```

## Output :



```
C:\Users\user\OneDrive\Docu X + v
Total objects: 1
Total objects: 2

Process returned 0 (0x0)   execution time : 0.065 s
Press any key to continue.
```

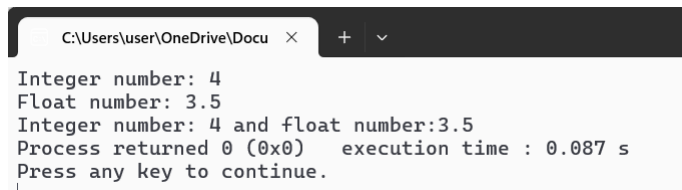
### Example-3 :

A program to understand the Function Overloading in C++.

#### Input:

```
#include <iostream>
using namespace std;
void print(int var) {
    cout << "Integer number: " << var << endl;
}
void print(float var) {
    cout << "Float number: " << var << endl;
}
void print(int var1, float var2) {
    cout << "Integer number: " << var1;
    cout << " and float number:" << var2;
}
int main()
{
    int a = 4;
    float b = 3.5;
    print(a);
    print(b);
    print(a, b);
    return 0;
}
```

#### Output :



The screenshot shows a terminal window with the following output:

```
Integer number: 4
Float number: 3.5
Integer number: 4 and float number:3.5
Process returned 0 (0x0)   execution time : 0.087 s
Press any key to continue.
```

### Practice Exercise: 1

Write a C++ program to define a class Batsman with the following specifications, batsman ID: 6 digits roll number static member count: To keep track on number of object static function getcount(): return the value of count function getname(): To take batsman name as input function showname(): To show batsman name access all the data members and member functions using the objects of class Batsman.

#### Input:

```
#include <iostream>
#include <string>
using namespace std;
class Batsman {
public:
    int id;
    string name;
```

```

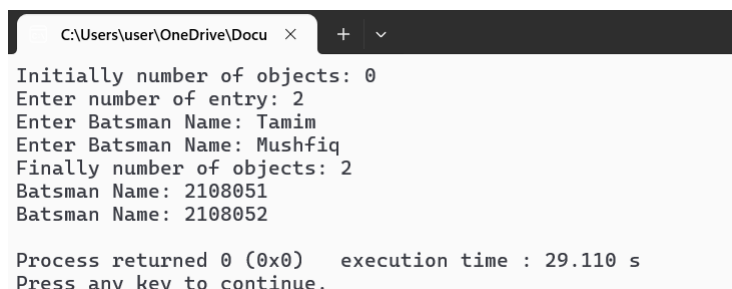
static int count;
Batsman() {
    id = 2108050 + count + 1;
    count++;
}
void getname() {
    cout << "Enter Batsman Name: ";
    getline(cin, name);
}
void showname() {
    cout << "Batsman Name: " << id << endl;
}
static int getcount() {
    return count;
}
};
int Batsman::count = 0;

int main() {
    int n;
    cout << "Initially number of objects: " << Batsman::getcount() << endl;
    cout << "Enter number of entry: ";
    cin >> n;
    cin.ignore(); // Consume the newline character
    Batsman batsman[n];
    for (int i = 0; i < n; i++) {
        batsman[i].getname();
    }

    cout << "Finally number of objects: " << Batsman::getcount() << endl;
    for (int i = 0; i < n; i++) {
        batsman[i].showname();
    }
    return 0;
}

```

## Output :



The screenshot shows a terminal window with the following output:

```

Initially number of objects: 0
Enter number of entry: 2
Enter Batsman Name: Tamim
Enter Batsman Name: Mushfiq
Finally number of objects: 2
Batsman Name: 2108051
Batsman Name: 2108052

Process returned 0 (0x0)   execution time : 29.110 s
Press any key to continue.

```

## Practice Exercise: 2

Write a C++ Program to calculate the area of different geometric shapes such as Circle, Triangle, and Rectangle. Use function overloading. Class Name: Shape

**Input:**

```

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

class Shape {
public:
    // Function to calculate the area of a circle
    double area(double radius) {
        return M_PI * radius * radius;
    }
    // Function to calculate the area of a rectangle
    double area(double length, double width) {
        return length * width;
    }
    // Function to calculate the area of a triangle
    double area(double base, double height, bool isTriangle) {
        return 0.5 * base * height;
    }
};

int main() {
    Shape shape;
    double radius = 3.0;
    cout << "Area of Circle with radius " << radius << " is " << shape.area(radius) << endl;

    double length = 20.0, width = 15.0;
    cout << "Area of Rectangle with length " << length << " and width " << width << " is " << shape.area(length, width) << endl;

    double base = 7.0, height = 3.0;
    cout << "Area of Triangle with base " << base << " and height " << height << " is " << shape.area(base, height, true) << endl;
    return 0;
}

```

## Output :

```

C:\Users\user\OneDrive\Docu  ×  +  ▾
Area of Circle with radius 3 is 28.2743
Area of Rectangle with length 20 and width 15 is 300
Area of Triangle with base 7 and height 3 is 10.5

Process returned 0 (0x0)   execution time : 0.083 s
Press any key to continue.

```

## Discussion:

In this experiment above, state data member, function and overloading in C++ were introduced. Here, public and private access of data members and member functions were also familiarized. Different types of functions were created and their overloading was observed. While solving the problems, I faced a little difficulty in understanding function overloading. But after implementing part by part, the concept was understood. Also some errors occurred while doing with static member functions but then I fixed the errors soon and solved the problems as well.