CSE-284: Object Oriented Programming
Experiment 3: Static Data Member, and Function Overloading in C++

Turja Roy
ID: 2108052
Group: G-2

Objectives:

- Introduction with the Static Data Member and Member function.
- To understand the concept of function overloading in C++.

Example 1

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

Code

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

class Square {
private:
    int side;
    static int objCount;

public:
    Square () {
        objCount++;
    }

    void setSide (int s) {
        side = s;
    }
    int getSide () {
        return side;
    }

    static int getCount () {
        return objCount;
    }
};

int Square::objCount = 0;

int main ()
```

```cpp
{
    Square s1;
    s1.setSide(5);
    cout << "Total objects: " << s1.getCount() << endl;

    Square s2;
    s2.setSide(10);
    cout << "Total objects: " << Square::getCount() << endl;

    cout << "Side of s1: " << s1.getSide() << endl;
    cout << "Side of s2: " << s2.getSide() << endl;

    /*
        A static method can be called both by an object of the class and by the class itself.
        s1.getCount(), s2.getCount, and Square::getCount() are both valid and returns the same value.
        But s1.getSide() and s2.getSide() are not valid because getSide() is not a static method.
        Non-static methods can only be called by objects of the class.
        Cannot access non-static member 'Square::side' within static member function
    */

    return 0;
}
```

Output

Figure 1: Output of Exp-1.cpp

Example 2

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

Code

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

class Square {
private:
    int side;
public:
    static int objCount;

    Square () {
        objCount++;
    }
};

int Square::objCount = 0;
```

```cpp

int main ()
{
    Square s1;
    cout << "Total objects: " << Square::objCount << endl;

    Square s2;
    cout << "Total objects: " << Square::objCount << endl;

    return 0;
}
```

Output

Figure 2: Output of Exp-2.cpp

Example 3

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

Code

```cpp
#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, int var2) {
    cout << "Integer number: " << var1;
    cout << " and another Integer number: " << var2 << endl;
}
void print (int var1, float var2) {
    cout << "Integer number: " << var1;
    cout << " and float number: " << var2 << endl;
}
void print (float var1, float var2) {
    cout << "Float number: " << var1;
    cout << " and another Float number: " << var2 << endl;
}

int main ()
{
    float a=1.5, b=2.5;
    int c=5;

    print(c, b);

    return 0;
```

```
}
```

Output

Figure 3: Output of Exp-3.cpp

Lab Task

Write a C++ program with a class `Student` that contains two variables `a, b`, and a static variable `objCount` to keep track of the number of objects created starting from 100. Print the values of `a, b, objCount` for each object created.

Code

```cpp
#include <bits/stdc++.h>
using namespace std;

class Student {
private:
    int a, b;
    static int objCount;

public:
    Student () {
        objCount++;
    }
    Student (int a, int b) {
        this->a = a;
        this->b = b;
        objCount++;
    }

    void set_a_b (int a, int b) {
        this->a = a;
        this->b = b;
    }
    void print_state () {
        cout << "a: " << a << " b: " << b << " objCount: " << objCount << endl;
    }
};

int Student::objCount = 100;

int main ()
{
    Student s1;
    s1.set_a_b(5, 10);
    s1.print_state();

    Student s2(10, 20);
    s2.print_state();
```

```cpp

    Student s3(15, 30);
    s3.print_state();

    return 0;
}
```

Output

Figure 4: Output of Lab Task

Practice 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
- `showname()`: To show batsman name

Access all the data members and member functions using the objects of class Batsman.

Code

```cpp
#include <bits/stdc++.h>
using namespace std;

class Batsman {
private:
    string name;
    int batsman_ID;
    static int count;

public:
    Batsman (int batsman_ID) {
        this->batsman_ID = batsman_ID;
        count++;
    }

    static int getCount () {
        return count;
    }
    void getname () {
        cout << "Enter Batsman Name: ";
        if (cin.peek() == '\n') cin.ignore();
        getline(cin, name);
```

```cpp
    }
    void showname () {
        cout << "Name of the batsman: " << name << endl;
    }
    void showID () {
        cout << "ID of the batsman: " << batsman_ID << endl;
    }
};

int Batsman::count = 0;

int main ()
{
    cout << "Initially number of objects: " << Batsman::getCount() << endl;
    int n;
    cout << "Enter the number of entry: ";
    cin >> n;
    vector <Batsman> batsman;

    for (int i=0 ; i<n ; i++) {
        int id = 180400 + Batsman::getCount()+1;
        Batsman b = Batsman(id);
        b.getname();
        batsman.push_back(b);
    }

    cout << endl << "Finally number of objects: " << Batsman::getCount() << endl;
    for (auto b : batsman) {
        b.showname();
        b.showID();
        cout << endl;
    }

    return 0;
}
```

Output

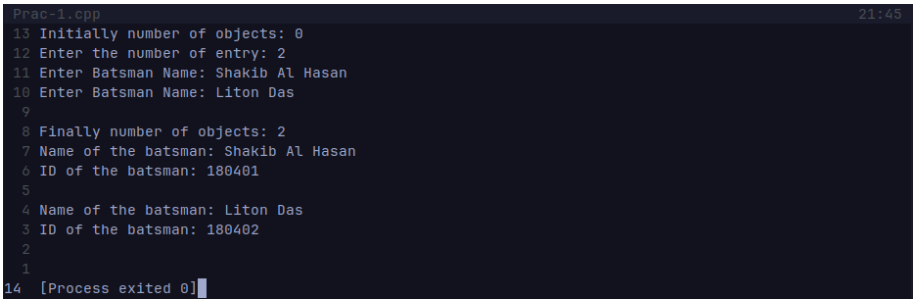

Figure 5: Output of Prac-1.cpp

Practice 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

Code

```cpp
#include <bits/stdc++.h>
using namespace std;

class Shape {
private:
    double radius;
    double side1, side2, side3;
    double width, height;

public:
    Shape (double radius) {
        this->radius = radius;
        side1 = side2 = side3 = width = height = 0;
    }
    Shape (double side1, double side2, double side3) {
        this->side1 = side1;
        this->side2 = side2;
        this->side3 = side3;
        radius = width = height = 0;
    }
    Shape (double width, double height) {
        this->width = width;
        this->height = height;
        radius = side1 = side2 = side3 = 0;
    }

    double getArea () {
        if (radius) return calculateArea(radius);
        else if (side1 && side2 && side3) return calculateArea(side1, side2, side3);
        else if (width && height) return calculateArea(width, height);
        else return 0;
    }

    double calculateArea (double radius) {
        return 3.1416 * radius * radius;
    }
    double calculateArea (double side1, double side2, double side3) {
        double s = (side1 + side2 + side3) / 2;
        return sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }
    double calculateArea (double width, double height) {
        return width * height;
    }
};

int main ()
{
    Shape s1 = Shape(5);
    Shape s2 = Shape(3, 4, 5);
    Shape s3 = Shape(4, 5);
```

```
    cout << "Area of Circle: " << s1.getArea() << endl;
    cout << "Area of Triangle: " << s2.getArea() << endl;
    cout << "Area of Rectangle: " << s3.getArea() << endl;

    return 0;
}
```

Output

Figure 6: Output of Prac-2.cpp

Discussion

- In this lab, Static Data Member and Static Member Functions were discussed.
- The concept of Function Overloading was discussed too.
- Function overloading is used to define multiple functions with the same name but with different parameters.
- In the Practice-2 task, the constructor function was overloaded to determine which shape it was.
- The area of the shapes was calculated using the overloaded function `calculateArea`.