

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

```
1 #include <iostream>
2 using namespace std;
3
4 class Square {
5 private:
6     int side;
7     static int objCount;
8
9 public:
10    Square () {
11        objCount++;
12    }
13
14    void setSide (int s) {
15        side = s;
16    }
17    int getSide () {
18        return side;
19    }
20
21    static int getCount () {
22        return objCount;
23    }
24 };
25
26 int Square::objCount = 0;
27
28 int main ()
```

```

29 {
30     Square s1;
31     s1.setSide(5);
32     cout << "Total objects: " << s1.getCount() << endl;
33
34     Square s2;
35     s2.setSide(10);
36     cout << "Total objects: " << Square::getCount() << endl;
37
38     cout << "Side of s1: " << s1.getSide() << endl;
39     cout << "Side of s2: " << s2.getSide() << endl;
40
41     /*
42      A static method can be called both by an object of the class and by
43      the class itself.
44      s1.getCount(), s2.getCount(), and Square::getCount() are both valid and
45      returns the same value.
46      But s1.getSide() and s2.getSide() are not valid because getSide() is
47      not a static method.
48      Non-static methods can only be called by objects of the class.
49      Cannot access non-static member 'Square::side' within static member
50      function
51     */
52
53     return 0;
54 }

```

Output

```

Exp-3.cpp 1:1
5 Total objects: 1
4 Total objects: 2
3 Side of s1: 5
2 Side of s2: 10
1
6 [Process exited 0]

```

Figure 1: Output of Exp-1.cpp

Example 2

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

Code

```

1 #include <iostream>
2 using namespace std;
3
4 class Square {
5 private:
6     int side;
7 public:
8     static int objCount;
9
10    Square () {
11        objCount++;
12    }
13 };
14
15 int Square::objCount = 0;

```

```

16
17 int main ()
18 {
19     Square s1;
20     cout << "Total objects: " << Square::objCount << endl;
21
22     Square s2;
23     cout << "Total objects: " << Square::objCount << endl;
24
25     return 0;
26 }

```

Output

```

Exp-2.cpp 1:1
3 Total objects: 1
2 Total objects: 2
1
4 [Process exited 0]

```

Figure 2: Output of Exp-2.cpp

Example 3

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

Code

```

1 #include <iostream>
2 using namespace std;
3
4 void print (int var) {
5     cout << "Integer number: " << var << endl;
6 }
7 void print (float var) {
8     cout << "Float number: " << var << endl;
9 }
10 void print (int var1, int var2) {
11     cout << "Integer number: " << var1;
12     cout << " and another Integer number: " << var2 << endl;
13 }
14 void print (int var1, float var2) {
15     cout << "Integer number: " << var1;
16     cout << " and float number: " << var2 << endl;
17 }
18 void print (float var1, float var2) {
19     cout << "Float number: " << var1;
20     cout << " and another Float number: " << var2 << endl;
21 }
22
23 int main ()
24 {
25     float a=1.5, b=2.5;
26     int c=5;
27
28     print(c, b);
29
30     return 0;

```

31 }

Output

```
Exp-1.cpp 1:1
2 Integer number: 5 and float number: 2.5
1
3 [Process exited 0]
```

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

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Student {
5 private:
6     int a, b;
7     static int objCount;
8
9 public:
10    Student () {
11        objCount++;
12    }
13    Student (int a, int b) {
14        this->a = a;
15        this->b = b;
16        objCount++;
17    }
18
19    void set_a_b (int a, int b) {
20        this->a = a;
21        this->b = b;
22    }
23    void print_state () {
24        cout << "a: " << a << " b: " << b << " objCount: " << objCount <<
endl;
25    }
26 };
27
28 int Student::objCount = 100;
29
30 int main ()
31 {
32     Student s1;
33     s1.set_a_b(5, 10);
34     s1.print_state();
35
36     Student s2(10, 20);
37     s2.print_state();
```

```

38
39     Student s3(15, 30);
40     s3.print_state();
41
42     return 0;
43 }

```

Output



```

Lab-Test.cpp 1:1
4 a: 5 b: 10 objCount: 101
3 a: 10 b: 20 objCount: 102
2 a: 15 b: 30 objCount: 103
1
5 [Process exited 0]

```

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

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Batsman {
5 private:
6     string name;
7     int batsman_ID;
8     static int count;
9
10 public:
11     Batsman (int batsman_ID) {
12         this->batsman_ID = batsman_ID;
13         count++;
14     }
15
16     static int getCount () {
17         return count;
18     }
19     void getname () {
20         cout << "Enter Batsman Name: ";
21         if (cin.peek() == '\n') cin.ignore();
22         getline(cin, name);

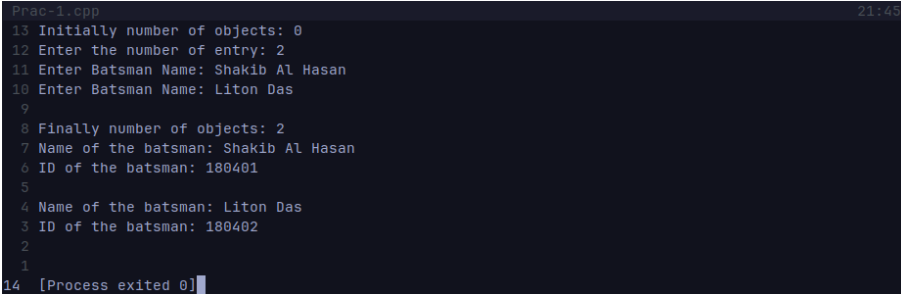
```

```

23     }
24     void showname () {
25         cout << "Name of the batsman: " << name << endl;
26     }
27     void showID () {
28         cout << "ID of the batsman: " << batsman_ID << endl;
29     }
30 };
31
32 int Batsman::count = 0;
33
34 int main ()
35 {
36     cout << "Initially number of objects: " << Batsman::getCount() << endl;
37     int n;
38     cout << "Enter the number of entry: ";
39     cin >> n;
40     vector <Batsman> batsman;
41
42     for (int i=0 ; i<n ; i++) {
43         int id = 180400 + Batsman::getCount()+1;
44         Batsman b = Batsman(id);
45         b.getname();
46         batsman.push_back(b);
47     }
48
49     cout << endl << "Finally number of objects: " << Batsman::getCount() <<
endl;
50     for (auto b : batsman) {
51         b.showname();
52         b.showID();
53         cout << endl;
54     }
55
56     return 0;
57 }

```

Output



```

Prac-1.cpp 21:45
13 Initially number of objects: 0
12 Enter the number of entry: 2
11 Enter Batsman Name: Shakib Al Hasan
10 Enter Batsman Name: Liton Das
9
8 Finally number of objects: 2
7 Name of the batsman: Shakib Al Hasan
6 ID of the batsman: 180401
5
4 Name of the batsman: Liton Das
3 ID of the batsman: 180402
2
1
14 [Process exited 0]

```

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.

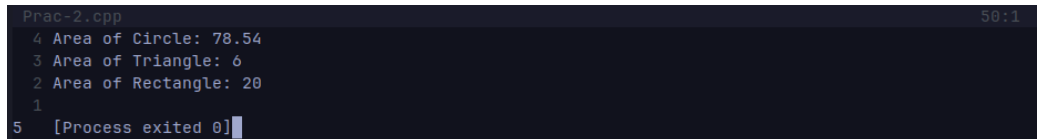
Class Name: Shape

Code

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Shape {
5 private:
6     double radius;
7     double side1, side2, side3;
8     double width, height;
9
10 public:
11     Shape (double radius) {
12         this->radius = radius;
13         side1 = side2 = side3 = width = height = 0;
14     }
15     Shape (double side1, double side2, double side3) {
16         this->side1 = side1;
17         this->side2 = side2;
18         this->side3 = side3;
19         radius = width = height = 0;
20     }
21     Shape (double width, double height) {
22         this->width = width;
23         this->height = height;
24         radius = side1 = side2 = side3 = 0;
25     }
26
27     double getArea () {
28         if (radius) return calculateArea(radius);
29         else if (side1 && side2 && side3) return calculateArea(side1, side2,
side3);
30         else if (width && height) return calculateArea(width, height);
31         else return 0;
32     }
33
34     double calculateArea (double radius) {
35         return 3.1416 * radius * radius;
36     }
37     double calculateArea (double side1, double side2, double side3) {
38         double s = (side1 + side2 + side3) / 2;
39         return sqrt(s * (s - side1) * (s - side2) * (s - side3));
40     }
41     double calculateArea (double width, double height) {
42         return width * height;
43     }
44 };
45
46 int main ()
47 {
48     Shape s1 = Shape(5);
49     Shape s2 = Shape(3, 4, 5);
50     Shape s3 = Shape(4, 5);
```

```
51     cout << "Area of Circle: " << s1.getArea() << endl;
52     cout << "Area of Triangle: " << s2.getArea() << endl;
53     cout << "Area of Rectangle: " << s3.getArea() << endl;
54
55     return 0;
56 }
57 }
```

Output



```
Prac-2.cpp 50:1
4 Area of Circle: 78.54
3 Area of Triangle: 6
2 Area of Rectangle: 20
1
5 [Process exited 0]
```

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`.