Assignment

Object Oriented Programming in C++

Submitted by:

Arvin Pradhan

ENG17CS0038

1. Write a C++ program to define the concepts of declaring the class, its data members and member functions. Also write a main () function which declares the objects and uses the member functions of the class.

```
#include<iostream>

using namespace std;

class student

{

   public:

   int rollno;

   char name[20];

   void display()

   {

      cout<<"Rollno:"<<rollno<<endl;

      cout<<"Name:"<<name;

   }

};

int main()

{

   student s;

   cout<<"enter the rollno.:";
   cin>>s.rollno;

   cout<<"enter the name:";

   cin>>s.name;
```

```
    s.display();
    return 0;
}
```

Output for 1st program:

```
enter the rollno:38
enter the name:Vinny
Rollno:38
Name:Vinny

...Program finished with exit code 0
Press ENTER to exit console.
```

---

2. Write a C++ program by combining the concepts of friend class and inline functions.

```
#include<iostream>
using namespace std;
class si
{
    private:
        float amount;
        float years;
        float int_rate;
    public:
        void get_data(void);
        friend int simpleinterest(void);
};
void si::get_data(void)
{
  cout<<"enter amount:";
  cin>>amount; cout<<"enter
  years:"; cin>>years;
```

```
  cout<<"enter interest rate:";
  cin>>int_rate;

}
int simpleinterest(void)
{
  si s;
  int temp;
  s.get_data();
  temp=(s.amount*s.years*s.int_rate)/100;
 return temp;
int main()
{
  int sim_int;
  sim_int=simpleinterest();
  cout<<"The simple interest = "<<sim_int<<endl;
  return 0;
}
```

```
enter amount:450000
enter years:2
enter interest rate:7
The simple interest = 63000


...Program finished with exit code 0
Press ENTER to exit console.
```

---

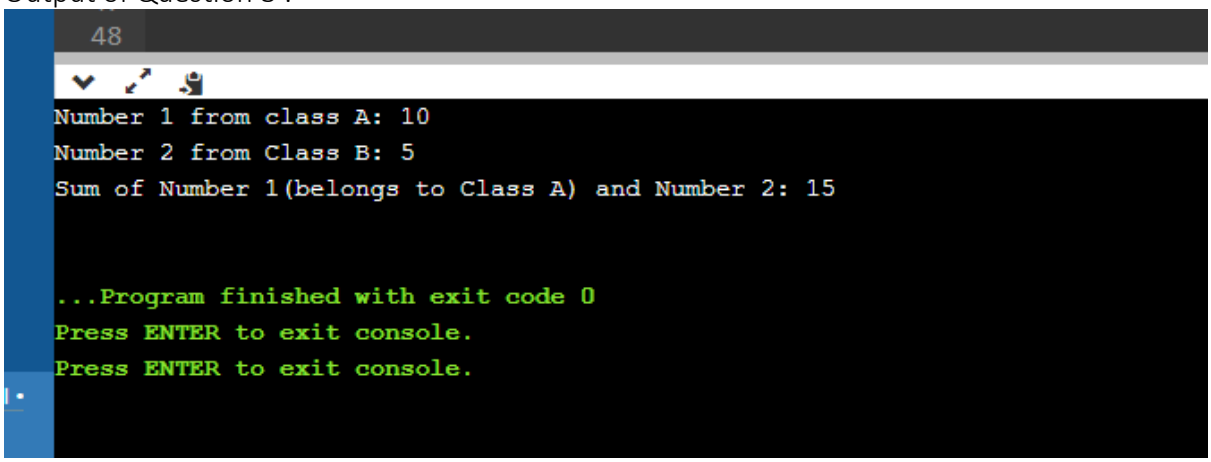
3. Write a C++ program to showcase the concepts of friend classes and constructors

```
#include<iostream>
using namespace std; class A
{
```

```
    int num1;
    public:
    A()
    {
        num1 = 10;
    }
    void show_num1()
    {
        cout<<"Number 1 from class A: "<<num1<<endl;
    }
    friend class B;
}; class
B
{
    int num2;
    public: B()
    {
    num2 =
    5;
    }
    void show_num2()
    {
        cout<<"Number 2 from Class B: "<<num2<<endl;
    }
    inline void sum_num1_and_num2(A obj)
    {
        cout<<"Sum of Number 1(belongs to Class A) and Number 2: "<<num2+obj.num1<<"\n";
    }
};
int main(){
    A a; B b;
    a.show_num1();
    b.show_num2();
```

```
    b.sum_num1_and_num2(a);

    return 0;

}
```

Output of Question 3 :



---

4. Write a C++ program to define the different types of constructors (including copy constructor) and also destructor

```
#include<iostream>
using namespace std;
class Numbers
{
    int a,b;
    public:
    Numbers()
    {
        a = 5;
        b = 10;
        cout<<"Default constructor called"<<endl;
    }
    Numbers(int num1, int num2)
    {
        a = num1;
        b = num2;
        cout<<"Parametarized constructor called"<<endl;
    }
    Numbers(Numbers &obj)
    {
        a = obj.a;
        b = obj.b;
        cout<<"Copy constructor called"<<endl;
    }
```

```
    void show_values()
    {
        cout<<"a = "<<a<<" and b = "<<b<<endl;
    }
    ~Numbers()

    {
        cout<<"Destructor is called"<<endl;
    }

};

int main(){

    Numbers n1;

    Numbers n2(10,87);

    Numbers n3 = n2;

    n1.show_values();

    n2.show_values();

    n3.show_values();

    return 0;

}
```

Output of Question 4:

```
Default constructor called
Parametarized constructor called
Copy constructor called
a = 5 and b = 10
a = 10 and b = 87
a = 10 and b = 87
Destructor is called
Destructor is called
Destructor is called
```

---

5. Write a C++ program to defining both the friend class and friend functions

```
#include<iostream>

using namespace std ;

class A

{
```

```
int x,y;

public:

A(int one, int two)

{

x=one;
y=two;
}
friend class B ;
friend void AddBoth (A First);
};
class B
{
public:
int max (A First)
{
return First.x>First.y? First.x:First.y;
}
};
void AddBoth (A First)
{
int sum;
sum=First.x+First.y;
cout << "This is the sum of values from Class A in a friend function named AddBoth() is "
<<sum<< "\n" ;
}
int main ()
{

int one, two;

cout << "Enter 2 numbers:\n" ;

cin >>one>>two;

A compare (one,two);

B maximum;

cout << "The larger number is " <<maximum.max(compare)<< ".\n" ;

cout << "\nCalling the friend function AddBoth() :\n" ;

AddBoth(compare);
return 0 ;

}
```

Output for Question 5:

```
Enter 2 numbers:
12
17
The larger number is 17.

Calling the friend function AddBoth() :
This is the sum of values from Class A in a friend function named AddBoth() is 29


...Program finished with exit code 0
```

6. Write a C++ program to for passing class objects as functional arguments

```
#include <iostream>
using namespace std; class Student
{
   public: double
   marks;
   Student(double m)
   {
      marks = m;
   }
};
void calculateAverage(Student s1, Student s2)
{
   double average = (s1.marks + s2.marks) / 2;
   cout << "Average Marks = " << average << endl;
}
int main()
{
   Student student1(72.0), student2(63.9);
   calculateAverage(student1, student2);
   return 0;
}
```

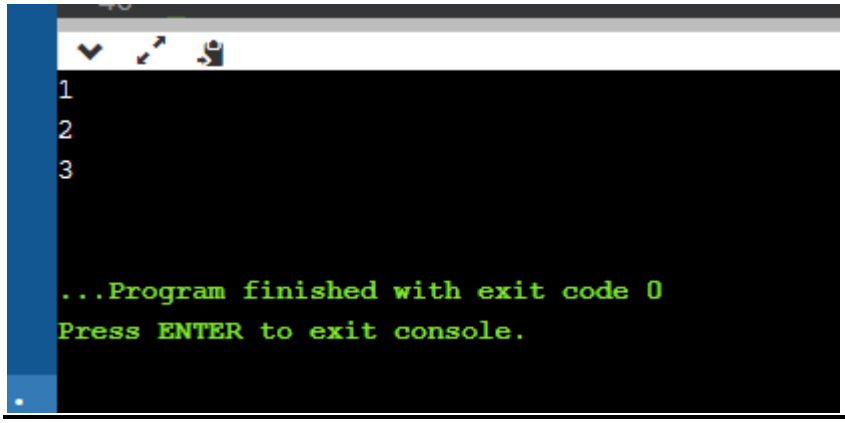
Output for question 6 :

```
Average Marks = 67.95


...Program finished with exit code 0
Press ENTER to exit console.
```

7. Write a C++ program to show the usage of arrays inside the class and array of objects

```
#include <iostream>

using namespace std;

class cl

{
        int i;
         public: void set_i(int j) { i=j; }

        int get_i() { return i; }

};

int main()

 {

cl ob[3];

int i;

for(i=0; i<3; i++)

{

         ob[i].set_i(i+1);

}

for(i=0; i<3; i++)

{

         cout << ob[i].get_i() << "\n";

}

return 0;

}
```

Output of question 7:



8. Write a C++ program to show how to pass the objects to function, returning objects. Define the constructor and destructor outside the class.

```
#include <iostream>
using namespace std;
class A
{
   public:
          int a;
   A();
          A add(A obj1, A obj2)
          {
                    A obj3;
                    obj3.a = obj1.a + obj2.a;
                    return obj3;
          }
   ~A();
};
A::A():a(5)
{
          cout<<"Constructer called!"<<" ";
}
A::~A()
{
          cout<<"Destructor called!"<<" ";
}
int main()
{
          A obj1, obj2, obj3;
          obj1.a = 5; obj2.a = 10; obj3.a = 0;
          cout << "\nInitially, Value of object 1: " << obj1.a << ", object 2: " << obj2.a << ", object
3: " << obj3.a << "\n";
          obj3 = obj3.add(obj1, obj2);
          cout << "\nValue of object 1: " << obj1.a << ", object 2: " << obj2.a << ", object 3: " <<
obj3.a << "\n";
          return 0;
```

}

Output of question 8 :

```
Constructer called! Constructer called! Constructer called!
Initially, Value of object 1: 5, object 2: 10, object 3: 0
Constructer called! Destructor called! Destructor called! Destructor called!
Value of object 1: 5, object 2: 10, object 3: 15
Destructor called! Destructor called! Destructor called!

...Program finished with exit code 0
Press ENTER to exit console.
```

---

9. Write a C++ program to define the concept of pointers to objects, passing reference to functions and return by reference.

```
#include<iostream>

using namespace std ;

class Numbers

{

public :

int one, two;

void insertData ( int a, int b)

{

one=a;

two=b;

}

void display (){

cout << "Data stored is " <<one<< " and " <<two<< ".\n" ;

}

};

void Change (Numbers *first)

{

cout << "I passed an object pointer to the function Change that multiplies the object's

data by 10;\n" ; first->one*= 10 ;
```

```
first->two*= 10 ;
cout << "Inside the function, in the reference object:\n" ;
first->display();
}
Numbers AnotherChange ()
{
Numbers temp;
int a,b;
```

```
cout << "Give me new numbers! We shall rewrite the object through this

function:\n" ;

cin >>a>>b;

temp.one=a;

temp.two=b;

cout << "-----------------\nTemp object inside the function is:\n" ;

temp.display();

return temp;

}

int main ()

{

int a,b;

cout << "Give me two numbers!\n" ;

cin >>a>>b;

Numbers first;

first.insertData(a,b);

Numbers *PointerToFirstObject=&first;

cout << "-----------------\nUsing a pointer to object in main to display the object:\n" ;

PointerToFirstObject->display();

cout << "-----------------\n" ;

Change(&first);

cout << "-----------------\n" ;

cout << "In main, AFTER we called the Change the function: \n" ;

first.display();

cout << "-----------------\nAnother function called AnotherChange exists

which returns an object reference!\n" ;

first=AnotherChange();
```

```
cout << "----------------\nRewritten object in main is :\n" ;

first.display();

cout << "----------------\n" ;

return 0 ;

}
```

Output of question 9:

```
input
Input two numbers!
21
33
-----------------
Using  a        pointer  to  object  in  main  to display the object:
Data stored is 21 and 33.
-----------------
I passed an object pointer to the function Change that multiplies the object's data by 10;
Inside the function, in the reference object:
Data stored is 210 and 330.
-----------------
In main, AFTER we called the Change the function:
Data stored is 210 and 330.
-----------------
Another function        called  AnotherChange exists which returns an object reference!
Give me new numbers! We shall rewrite the object through this function:
```

---

10. Write a C++ program to include the concept of this pointer and references

```
#include<iostream>

using namespace std ;

class Student

{

string name;

int rno,grade;

public :

//void getData();

void insert ( string name, int rno, int grade);

void display ();

void getData ();

};

void Student::insert( string name, int rno, int grade)
```

```
{
this ->name=name;
this ->rno=rno;
this ->grade=grade;
}
void Student::getData()
{
string name;
int rno, grade;
cout << "Enter name : " ;
cin >>name;
cout << "Enter Roll Number : " ;
cin >>rno;
cout << "Enter Grade : " ;
cin >>grade;
this ->insert(name, rno, grade);
}
void Student::display()
{
cout << "\n---------------\nData Stored::\nName :: " <<name<<
"\nRollNumber ::
" <<rno<< "\nGrade :: " <<grade<< "\nand this object is stored at location
" << this << "\n---------------\n" ;
}
int main ()
{
Student one;
one.getData();
one.display();
```

Student *FirstOneOnly=&one; //Pointer to object Student one, here FirstOneOnly is the reference

cout << "\n\nThe address of the reference to the first object is

" <<&FirstOneOnly<< "\n\n" ;

FirstOneOnly->display(); return 0 ;

}

Output:

```
Enter name : Vinny
Enter Roll Number : 38
Enter Grade : 5

---------------
Data    Stored::
Name    ::      Vinny
RollNumber :: 38
Grade :: 5
and this object is stored at location0x7ffdd06e2c50
---------------


The address of the reference to the first object is 0x7ffdd06e2c48


---------------
Data    Stored::
Name    ::      Vinny
RollNumber :: 38
Grade :: 5
and this object is stored at location0x7ffdd06e2c50
---------------


...Program finished with exit code 0
Press ENTER to exit console.
```

---

11. Explain with a C++ program the concepts of dynamic memory allocation along with function overloading features.

```
#include<iostream>

using namespace std ;

void Area ( int side)
```

```
{
cout << "\nThis is a square! Area is " <<side*side<< "\n\n" ;
}
void Area ( float radius)
{
cout << "\nThis is a circle! Area is " << 3.14 *radius*radius<< "\n\n" ;
}
void Area ( int length, int breadth)
{
cout << "\nThis is a rectangle! Area is " <<length*breadth<< "\n\n" ;
}
int main ()
{
int *where= new int ;
cout << "Dynamic Memory allocation demonstration using new and delete keywords!\nCome
on, give us a number! " ; cin >>*where;

cout << "Number stored : " <<*where<< "\nPointer's Address : " <<where<< "\nAddress of
the variable in heap memory : " <<&where<< "\n\n"; delete where;

cout << "\n-------------------------\n Function Overloading
Demonstration with a function called Area!\n" ;
cout << "Passing : Area(20); --> " ;
```

```
Area( 20 );

cout << "Passing : Area(12.9); --> " ; Area( 12.9f );

cout << "Passing : Area(12,5); --> " ; Area( 12 , 5 );

cout << "\nNotice how based on different parameters, the function Area still

works! Look at the code to understand how exactly!\nThank you!\n\n" ;

return 0 ;

}
```

Output:

```
Dynamic Memory allocation demonstration using new and delete keywords!
 Give us a number! 77
Number stored : 77
Pointer's Address : 0x15cdc20
Address of the variable in heap memory : 0x7fff173d8ec8


-------------------------
Function        Overloading Demonstration with a function called Area!
Passing : Area(20); -->
This is a square! Area is 400

Passing : Area(12.9); -->
This is a circle! Area is 522.527

Passing : Area(12,5); -->
This is a rectangle! Area is 60
```

---

12. Explain the concepts of constructor overloading and pointer to functions

```
#include<iostream>

using namespace std ;

class Numbers

{

int one, two;

public :

Numbers( int a, int b)

{

one=a;
```

```
two=b;
}
Numbers( int a)
{
one=a;
two= 0 ;
}
Numbers()
{
one=two= 0 ;
}
void display ()
{
cout << "Values stored are :: One : " <<one<< ", Two : " <<two<< "\n" ;
}
};
void dummy ()
{
cout << "This is inside my dummy function!\n" ;
}
int main ()
{
cout << "\nConstructor Overloading Demonstration! The class has 2 integer
containers.\n" ;
Numbers first ( 23 , 04 ), second ( 2001 ), third; cout << "\nObject
first(23,04) ::\n" ; first.display();
cout << "\nObject second(2001) ::\n" ;
second.display();
cout << "\nObject third ::\n" ;
```

```
third.display();

cout << "\n\n\n\nPointers to a function : Demonstration\n\nCalling function

normally\n" ;

dummy();

void (*ToFunction)()=&dummy;

cout << "\nUsing a pointer to call the same dummy function:\n" ;

(*ToFunction)();

return 0 ;

}
```

Output:

```
Constructor Overloading Demonstration! The class has 2 integer containers.

Object first(23,04) ::
Values stored are :: One : 23, Two : 4

Object second(2001) ::
Values stored are :: One : 2001, Two : 0

Object third ::
Values stored are :: One : 0, Two : 0




Pointers to a function : Demonstration

Calling function normally
This is inside my dummy function!

Using a pointer to call the same dummy function:
This is inside my dummy function!
```

---

13. Illustrate a C++ program to define the concepts operator overloading with and without using friend functions

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

class DoConversion {

public :
```

```
double value;
DoConversion( int something)
{
value=something;
}
void operator *(){
value*= 2.5 ;
}
void display (){
cout << "This is what I have stored : " <<value<< "\n" ;
}
};
int main (){
int value;
cout << "Enter a value for a distance in inches:" ;
cin >>value;
DoConversion A (value);
*A;
A.display();
return 0 ;
}
```

Output:

```
Enter a value for a distance in inches:95
This is what I have stored : 237.5


...Program finished with exit code 0
Press ENTER to exit console.
```

14. Write a C++ program to define operator overloading and pointer to functions

```
#include<iostream>
using namespace std ;
class DoConversion
{
public :
double value;
DoConversion( int something)
{
value=something;
}
void operator *()
{ //Operator overloading example! value*=
2.5 ;
}
void display ()
{
cout << "This is what I have stored : " <<value<< "\n" ;
}
};
void dummy ()
{
cout << "This is inside my dummy function!\n" ;
}
int main ()
{
int value;
cout << "Enter a value for a distance in inches: " ; cin >>value;
DoConversion A (value);
*A;
A.display();
```

```
cout << "\n\n\n\nPointers to a function : Demonstration\n\nCalling function normally\n" ;

dummy();

void (*ToFunction)()=&dummy;

cout << "\nUsing a pointer to call the same dummy function:\n" ;

(*ToFunction)();

return 0 ;

}
```

Output:

```
Enter a value for a distance in inches: 100
This is what I have stored : 100




Pointers to a function : Demonstration

Calling function normally
This is inside my dummy function!

Using a pointer to call the same dummy function:
This is inside my dummy function!
```

15.Write a C++ program to demonstrate the concepts of default arguments and static member variable and member functions

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

class A
{
public:
   A() { cout << "A's Constructor Called " << endl;  }
};

class B
{
   static A a;
public:
   B() { cout << "B's Constructor Called " << endl; }
};

int main()
{
   B b;
```

```
    return 0;
}
```

Output:

```
B's Constructor Called


...Program finished with exit code 0
Press ENTER to exit console.
```