Experiment No-01: Introduction to Class and Objects in OOP

Md. Hasibul Islam Jihad; ID - 2108048

06/10/24

Objective

Write a C++ program to:

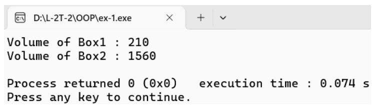
- Define a class `Box` and create objects of this class.
- Use the data members `length`, `breadth`, and `height` of the class.
- Calculate the volume of a box by accessing the members of this class using its object.

Example - 01 : Write a C++ program to define a class BOX and create objects of this class.

Code

```cpp
// C++ program to define a class Box and find the volume of a box

#include <iostream>
using namespace std;

class Box {
public:
    double length;   // Length of a box
    double breadth;  // Breadth of a box
    double height;   // Height of a box
};

int main() {
    Box Box1; // Declare Box1 of type Box
    Box Box2; // Declare Box2 of type Box
    double volume = 0.0; // Store the volume of a box here

    Box1.height = 5.0;
    Box1.length = 6.0;
    Box1.breadth = 7.0;

    Box2.height = 10.0;
    Box2.length = 12.0;
    Box2.breadth = 13.0;

    volume = Box1.height * Box1.length * Box1.breadth;
    cout << "Volume of Box1 : " << volume << endl;

    volume = Box2.height * Box2.length * Box2.breadth;
    cout << "Volume of Box2 : " << volume << endl;

    return 0;
}
```

Output



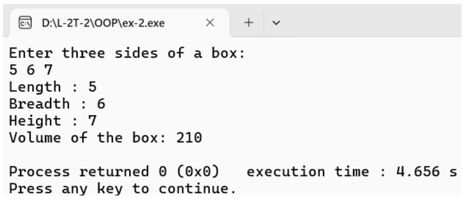
```
Volume of Box1 : 210
Volume of Box2 : 1560

Process returned 0 (0x0)   execution time : 0.074 s
Press any key to continue.
```

Example - 02: Write a C++ program to define a class BOX with member functions.

Code

```cpp

#include <iostream>
using namespace std;

class BOX {
public:
    double length, breadth, height;   //

    void input_value() {
        cout << "Enter three sides of a box: " << endl;
        cin >> length >> breadth >> height;
    }

    void print_value() {
        cout << "Length : " << length << endl;
        cout << "Breadth : " << breadth << endl;
        cout << "Height : " << height << endl;
    }

    double volume() {
        double v = length * breadth * height;
        return v;
    }
};

int main() {
    BOX myBox;
    myBox.input_value();
    myBox.print_value();
    double vol = myBox.volume();
    cout << "Volume of the box: " << vol << endl;
}
```

Output



```
Enter three sides of a box:
5 6 7
Length : 5
Breadth : 6
Height : 7
Volume of the box: 210

Process returned 0 (0x0)   execution time : 4.656 s
Press any key to continue.
```

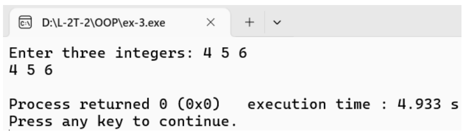
Example - 03 : Write a C++ program to understand public and private access of class data members.

Code

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

class myTest {
private:
    int a, b, c;  // Private data members

public:
    // Public member function to access and modify private data members
    void input_private() {
        cout << "Enter three integers: ";
        cin >> a >> b >> c;
    }

    // Public member function to display the values of private data members
    void access_private() {
        cout << a << ' ' << b << ' ' << c << endl;
    }
};

int main() {
    myTest v;  // Create an object of myTest class

    // Use the public member function to input private members
    v.input_private();

    // Use the public member function to display the private members
    v.access_private();

    return 0;
}
```

Output

for fixed code



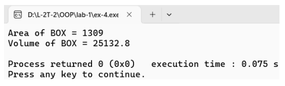
```
Enter three integers: 4 5 6
4 5 6

Process returned 0 (0x0)   execution time : 4.933 s
Press any key to continue.
```

Example - 04 : Write a C++ program to understand public and private access of class data members.

Code

```cpp

#include <iostream>
using namespace std;

class BOX {
private:
    double length, breadth, height;

public:
    void initData(double len, double brth, double hgt) {
        length = len;
        breadth = brth;
        height = hgt;
    }

    double calculateArea() {
        return length * breadth;
    }

    double calculateVolume() {
        return length * breadth * height;
    }
};

int main() {
    BOX box1;

    box1.initData(42.5, 30.8, 19.2);

    cout << "Area of BOX = " << box1.calculateArea() << endl;
    cout << "Volume of BOX = " << box1.calculateVolume() <<
        endl;

    return 0;
}
//excercise 1 is similar to this
```

Output



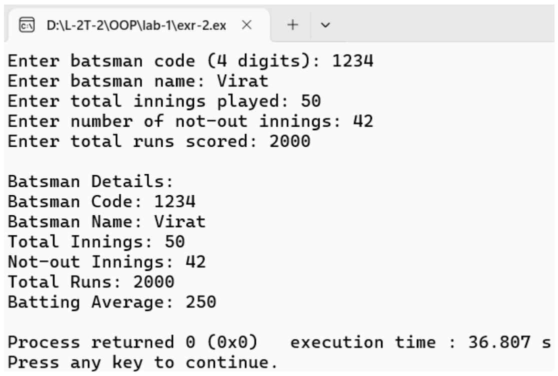
```
Area of BOX = 1309
Volume of BOX = 25132.8

Process returned 0 (0x0)   execution time : 0.075 s
Press any key to continue.
```

Exercise - 02

Code

```cpp

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

class Batsman {
private:
    int batsman_code;
    string batsman_name;
    int total_innings;
    int notout_innings;
    int total_runs;
    float batting_avg;

    void calcavg() {
        if (total_innings - notout_innings != 0) {
            batting_avg = static_cast<float>(total_runs) / (
                total_innings - notout_innings);
        } else {
            batting_avg = 0; // To avoid division by zero
        }
    }

public:
    // Function to accept values from user
    void readdata() {
        cout << "Enter batsman code (4 digits): ";
        cin >> batsman_code;
        cin.ignore();
        cout << "Enter batsman name: ";
        getline(cin, batsman_name);
        cout << "Enter total innings played: ";
        cin >> total_innings;
        cout << "Enter number of not-out innings: ";
        cin >> notout_innings;
        cout << "Enter total runs scored: ";
        cin >> total_runs;

        calcavg();


    void displaydata() const {
        cout << "\nBatsman Details:\n";
        cout << "Batsman Code: " << batsman_code << endl;
        cout << "Batsman Name: " << batsman_name << endl;
        cout << "Total Innings: " << total_innings << endl;
        cout << "Not-out Innings: " << notout_innings << endl;
        cout << "Total Runs: " << total_runs << endl;
        cout << "Batting Average: " << batting_avg << endl;
    }
};

int main() {
    Batsman player;

    // Accept and display the details of the batsman
    player.readdata();
    player.displaydata();

    return 0;
}
```

Output



```
Enter batsman code (4 digits): 1234
Enter batsman name: Virat
Enter total innings played: 50
Enter number of not-out innings: 42
Enter total runs scored: 2000

Batsman Details:
Batsman Code: 1234
Batsman Name: Virat
Total Innings: 50
Not-out Innings: 42
Total Runs: 2000
Batting Average: 250

Process returned 0 (0x0)   execution time : 36.807 s
Press any key to continue.
```

Discussion

- The goal of this experiment was to get introduced to class and objects in OOP
- The concept of private and public data members was learnt from this experiment
- Then the way of accessing private and public data and writing a program based on that was learnt.

Experiment No-02: Constructor and Destructor in OOP

Md. Hasibul Islam Jihad; ID - 2108048

23/10/24

Objective

- Introduce the Constructor Class in C++.
- Define different types of constructors.
- Learn Constructor and Destructor in C++ with the help of examples.

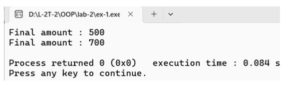
Exercise - 01 : Suppose you have a Savings Account with an initial amount of 500 and you have to add some more amount to it. Create a class 'AddMoney' with a data member named 'amount' with an initial value of 500. Now make two constructors of this class as follows:

- **without any parameter - no amount will be added to the Savings Account.**
- **having a parameter which is the amount that will be added to the Savings Account.**

Code

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

class AddMoney{
    private :
        double amount;
    public :
        AddMoney() {
            amount = 500;
        }

        AddMoney(double additionalAmount){
            amount = 500 + additionalAmount;
        }
        void displayAmount() {
            cout<<"Final amount : "<<amount<<endl;
        }
};

int main()
{
    //object without any added amount
    AddMoney account1;
    account1.displayAmount();

    //object with an added amount
    AddMoney account2(200);
    account2.displayAmount();

    return 0;
}
```

Output



```
Final amount : 500
Final amount : 700

Process returned 0 (0x0)   execution time : 0.084 s
Press any key to continue.
```

Exercise - 02 : Write a C++ Program to define a class Car with the following specifications: Class Specifications for Car Private members:

- **car name: `string` type**
- **model name: `string` type**
- **fuel type: `string` type**
- **mileage: `float` type**
- **price: `double` type**

Public members:

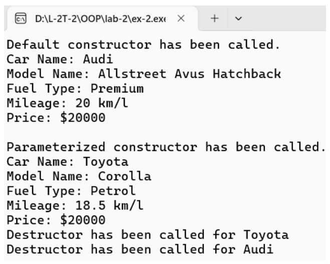
- **`displaydata()`: Function to display the data members on the screen.**
- **Use both default and parameterized constructors.**
  - **The default constructor will be called when no parameters are passed, and it will display the message:**

    "Default constructor has been called."
- **Destructor to clean up when the object goes out of scope.**

Code

```cpp

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

class Car {
private:
    string car_name;
    string model_name;
    string fuel_type;
    float mileage;
    double price;

public:
    // Default constructor
    Car() {
        car_name = "Audi";
        model_name = "Allstreet Avus Hatchback";
        fuel_type = "Premium";
        mileage = 20;
        price = 20000;
        cout << "Default constructor has been called." << endl;
    }

    // Parameterized constructor
    Car(string cn, string mn, string ft, float mil, double pr)
        {
        car_name = cn;
        model_name = mn;
        fuel_type = ft;
        mileage = mil;
        price = pr;
        cout << "Parameterized constructor has been called." << endl;
    }

    // Destructor
    ~Car() {
        cout << "Destructor has been called for " << car_name << endl;
    }

    // Function to display data members
    void displayData() {
        cout << "Car Name: " << car_name << endl;
        cout << "Model Name: " << model_name << endl;
        cout << "Fuel Type: " << fuel_type << endl;
        cout << "Mileage: " << mileage << " km/l" << endl;
        cout << "Price: $" << price << endl;
    }
};

int main() {
    // Using the default constructor
    Car car1;
    car1.displayData();
    cout << endl;

    // Using the parameterized constructor
    Car car2("Toyota", "Corolla", "Petrol", 18.5, 20000);
    car2.displayData();

    return 0;
}
```

Output



```
Default constructor has been called.
Car Name: Audi
Model Name: Allstreet Avus Hatchback
Fuel Type: Premium
Mileage: 20 km/l
Price: $20000

Parameterized constructor has been called.
Car Name: Toyota
Model Name: Corolla
Fuel Type: Petrol
Mileage: 18.5 km/l
Price: $20000
Destructor has been called for Toyota
Destructor has been called for Audi
```

Discussion

- The goal of this experiment was to get introduced to constructor and destructor in OOP and define them.
- Some bugs were found while executing the codes. They were fixed eventually with the help of the instructions from the terminal
- Through examples and exercises, the demonstration of constructor and destructor's working process with practical understanding of these essential object-oriented programming features in C++ were gained

Experiment No-03:Static Data Member, and Function Overloading in C++

Md. Hasibul Islam Jihad; ID - 2108048

24/10/24

Objective

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

Exercise - 01 : Write a C++ program to define a class Batsman with the following specifications:

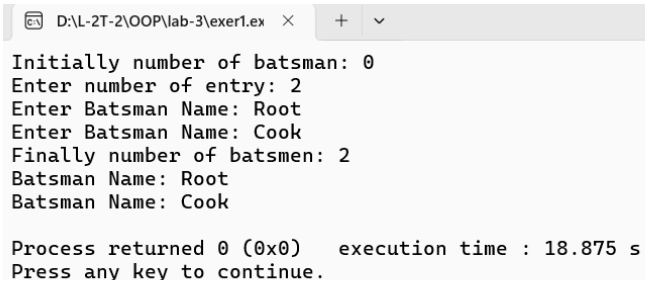
- **batsman ID: 6 digits roll number**
- **static member count: To keep track of the number of objects**
- **static function getcount(): Returns the value of count**
- **function getname(): To take batsman name as input**
- **function showname(): To show batsman name**

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

Code

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

class Batsman {
private:
    int batsman_ID;
    string name;
    static int btscount;
public:
    Batsman() {
        btscount++;
    }

    static int getcount() {
        return btscount;
    }

    void getname() {
        cout << "Enter Batsman Name: ";
        cin >> name;
    }

    void showname() {
        cout << "Batsman Name: " << name << endl;
    }
};

int Batsman::btscount = 0;

int main() {
    cout << "Initially number of batsman: " << Batsman::
        getcount() << endl;

    int n;
    cout << "Enter number of entry: ";
    cin >> n;

    Batsman batsmen[n];

    for (int i = 0; i < n; i++) {
        batsmen[i].getname();
    }

    cout << "Finally number of batsmen: " << Batsman::getcount
        () << endl;

    for (int i = 0; i < n; i++) {
        batsmen[i].showname();
    }

    return 0;
}
```

Output



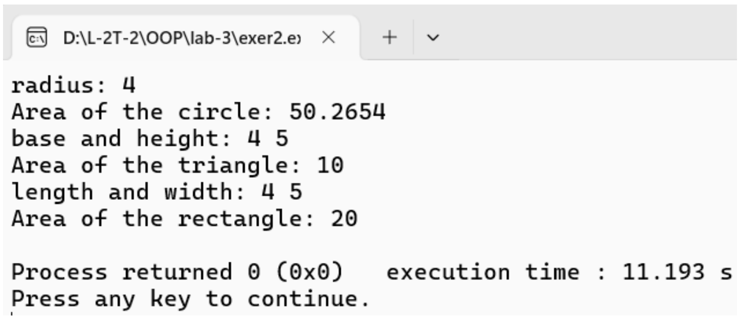
```
Initially number of batsman: 0
Enter number of entry: 2
Enter Batsman Name: Root
Enter Batsman Name: Cook
Finally number of batsmen: 2
Batsman Name: Root
Batsman Name: Cook

Process returned 0 (0x0)   execution time : 18.875 s
Press any key to continue.
```

Exercise - 02 : 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<iostream>
using namespace std;

class Shape {
public:
    double area(double radius) {
        return 3.14159 * radius * radius;
    }

    double area(double base, double height) {
        return 0.5 * base * height;
    }

    double area(double length, double width, bool isRectangle)
        {
        return length * width;
    }
};

int main() {
    Shape shape;

    double rad;
    cout << "radius: ";
    cin >> rad;
    cout << "Area of the circle: " << shape.area(rad) << endl;

    double base, height;
    cout << "base and height: ";
    cin >> base >> height;
    cout << "Area of the triangle: " << shape.area(base, height
        ) << endl;

    double length, width;
    cout << "length and width: ";
    cin >> length >> width;
    cout << "Area of the rectangle: " << shape.area(length,
        width, true) << endl;

    return 0;
}
```

Output



D:\L-2T-2\OOP\lab-3\exer2.e

```
radius: 4
Area of the circle: 50.2654
base and height: 4 5
Area of the triangle: 10
length and width: 4 5
Area of the rectangle: 20

Process returned 0 (0x0)   execution time : 11.193 s
Press any key to continue.
```

Discussion

- The goal of this experiment was to get introduced to static data members, functions and understand the concept of overloading.
- Some bugs were found while executing the codes. They were fixed eventually with the help of the instructions from the terminal
- Through examples and exercises,the concept of overloading was learnt and practical understanding of these essential object-oriented programming features in C++ were gained

Experiment No-04:Inheritance in C++.

Md. Hasibul Islam Jihad; ID - 2108048

06/11/24

Objective

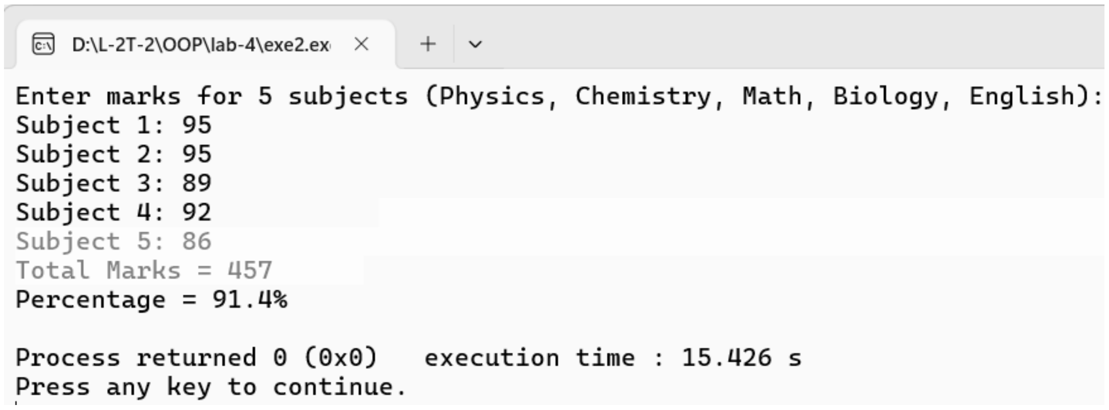
- Familiarize with Inheritance.
- Explain the concept of single and Multi level inheritance in OOP.
- Solve various problems in order to comprehend the above topics

Exercise - 01 : Write a C++ program to add two numbers. Accept these two numbers from the user in base class and display the sum of these two numbers in derived class.

Code

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

class Input{
    protected :
        int a;
        int b;
    public :
        void num1(int n1){
            a = n1;
        }

        void num2(int n2){
            b = n2;
        }
};

class Sum : public Input {
    public :
        int addSum(){
            return(a+b);
        }
};

int main(void){
    Sum s;
    s.num1(48);
    s.num2(52);
    cout<<"The sum is :"<<s.addSum()<<endl;

    return 0;
}
```

Output


D:\L-2T-2\OOP\lab-4\exe2.ex

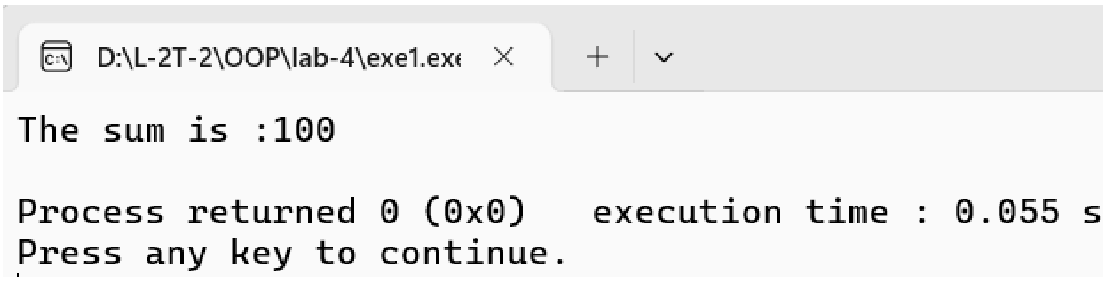
```
Enter marks for 5 subjects (Physics, Chemistry, Math, Biology, English):
Subject 1: 95
Subject 2: 95
Subject 3: 89
Subject 4: 92
Subject 5: 86
Total Marks = 457
Percentage = 91.4%

Process returned 0 (0x0)   execution time : 15.426 s
Press any key to continue.
```

Exercise - 02 : Write a C++ program to calculate the percentage of a student. Accept the marks of five subjects (Physics, Chemistry, Math, Biology, and English) in base class. A class will derived from the base class which includes a function to find the total marks obtained and another class derived from this first derived class which calculates and displays the percentage of student.

Code

```cpp

#include <iostream>
using namespace std;

class Marks {
protected:
    int marks[5];

public:
    void getMarks() {
        cout << "Enter marks for 5 subjects (Physics, Chemistry, Math, Biology, English):" << endl;
        for (int i = 0; i < 5; i++) {
            cout << "Subject " << i + 1 << ": ";
            cin >> marks[i];
        }
    }
};

class TotalMarks : public Marks {
protected:
    int total;

public:
    void calculateTotal() {
        total = 0;
        for (int i = 0; i < 5; i++) {
            total += marks[i];
        }
    }

    int getTotal() {
        return total;
    }
};

class Percentage : public TotalMarks {
public:
    void displayPercentage() {
        calculateTotal();
        float percentage = (float(total) / 500) * 100;
        cout << "Total Marks = " << total << endl;
        cout << "Percentage = " << percentage << "%" << endl;
    }
};

int main() {
    Percentage student;
    student.getMarks();
    student.displayPercentage();

    return 0;
}
```

Output



```
The sum is :100

Process returned 0 (0x0)   execution time : 0.055 s
Press any key to continue.
```

Discussion

- The goal of this experiment was to get introduced to the inheritance property in Object Oriented Programming
- Some bugs were found while executing the codes. They were fixed eventually with the help of the instructions from the IDE terminal
- Through examples and exercises,the concept of inheritance was learnt and practical understanding of these essential object-oriented programming features in C++ were gained.

Experiment No-06: Friend Function and Friend Class in C++.

Md. Hasibul Islam Jihad; ID - 2108048

26/11/24

Objective

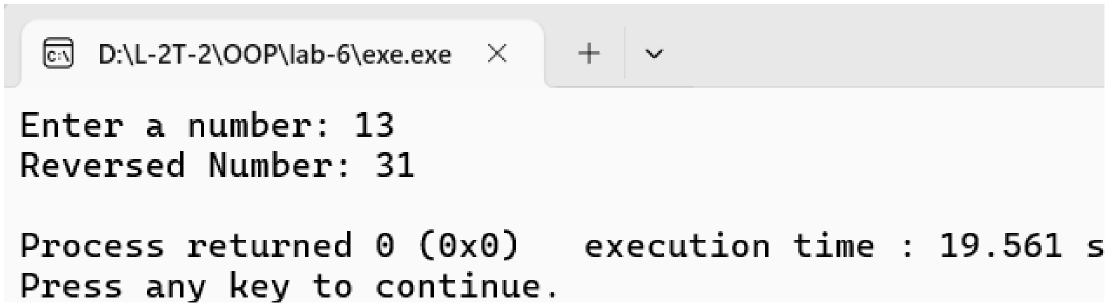
- To familiarize with friend class and function in C++ .
- To solve some problems using friend function

Exercise - 01 : Write a C++ Program to display the reverse of a number using the Friend function.

Code

```cpp

#include <iostream>
using namespace std;

class Number {
private:
    int num;
    friend int reverseNumber(Number);

public:
    Number() {
        num = 0;
    }

    void setNumber(int n) {
        num = n;
    }
};

int reverseNumber(Number n) {
    int reversed = 0;
    while (n.num != 0) {
        reversed = reversed * 10 + (n.num % 10);
        n.num /= 10;
    }
    return reversed;
}

int main() {
    Number n;
    int input;

    cout << "Enter a number: ";
    cin >> input;

    n.setNumber(input);
    cout << "Reversed Number: " << reverseNumber(n) << endl;

    return 0;
}
```

Output



D:\L-2T-2\OOP\lab-6\exe.exe

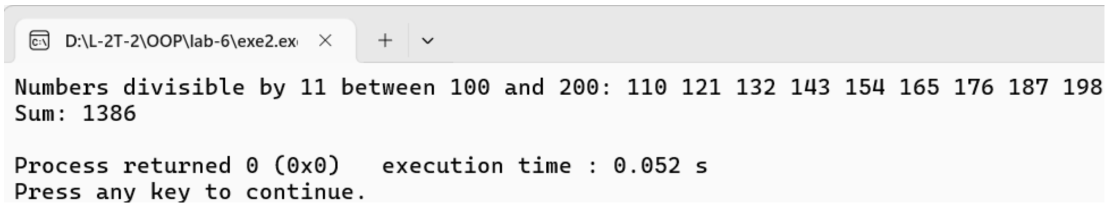
```
Enter a number: 13
Reversed Number: 31

Process returned 0 (0x0)   execution time : 19.561 s
Press any key to continue.
```

Exercise - 02 :Write a C++ program to find the number and sum of all integer between 100 and 200 which are divisible by 11 with friend function

Code

```

#include <iostream>
using namespace std;

class DivisibleBy11 {
private:
    int sum;
public:
    DivisibleBy11() {
        sum = 0;
    }

    friend void findDivisibles(DivisibleBy11);
};

void findDivisibles(DivisibleBy11 obj) {
    cout << "Numbers divisible by 11 between 100 and 200: ";
    for (int i = 100; i <= 200; i++) {
        if (i % 11 == 0) {
            cout << i << " ";
            obj.sum += i;
        }
    }
    cout << endl << "Sum: " << obj.sum << endl;
}

int main() {
    DivisibleBy11 obj;
    findDivisibles(obj);
    return 0;
}
```

Output


D:\L-2T-2\OOP\lab-6\exe2.ex

```
Numbers divisible by 11 between 100 and 200: 110 121 132 143 154 165 176 187 198
Sum: 1386

Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.
```

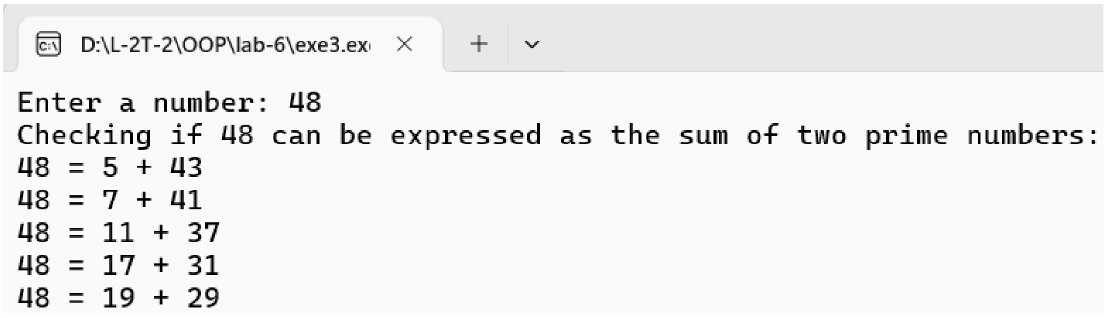
Exercise - 03 :Write a program in C++ to Check Whether a Number can be expressed as Sum of Two Prime Numbers using the friend functionn

Code

```

#include <iostream>
using namespace std;

class Number {
private:
    int num;
    friend class PrimeCheck;

public:
    Number() {
        num = 0;
    }

    void setNumber(int n) {
        num = n;
    }
};

class PrimeCheck {
public:
    bool isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i <= n / 2; i++) {
            if (n % i == 0) return false;
        }
        return true;
    }

    void checkSum(Number obj) {
        bool found = false;
        cout << "Checking if " << obj.num << " can be expressed
             as the sum of two prime numbers:" << endl;
        for (int i = 2; i <= obj.num / 2; i++) {
            if (isPrime(i) && isPrime(obj.num - i)) {
                cout << obj.num << " = " << i << " + " << obj.
                    num - i << endl;
                found = true;
            }
        }
        if (!found) {
            cout << obj.num << " cannot be expressed as the sum
```

```
                        of two prime numbers." << endl;
        }
    }
};

int main() {
    Number obj;
    PrimeCheck checker;

    int input;
    cout << "Enter a number: ";
    cin >> input;

    obj.setNumber(input);
    checker.checkSum(obj);

    return 0;
}
```

Output



D:\L-2T-2\OOP\lab-6\exe3.ex

```
Enter a number: 48
Checking if 48 can be expressed as the sum of two prime numbers:
48 = 5 + 43
48 = 7 + 41
48 = 11 + 37
48 = 17 + 31
48 = 19 + 29
```

Discussion

- The goal of this experiment was to get introduced to Friend Function and Frienc Friend class in Object Oriented Programming.
- Some bugs were found while executing the codes. They were fixed eventually with the help of the instructions from the IDE terminal
- Through examples and exercises,the concept of inheritance was learnt and practical understanding of these essential object-oriented programming features in C++ were gained.

Experiment No-07: Operator Overloading in C++.

Md. Hasibul Islam Jihad; ID - 2108048

11/12/24

Objective

- To understand operator overloading in C++.
- To implement operator overloading using the Friend function.

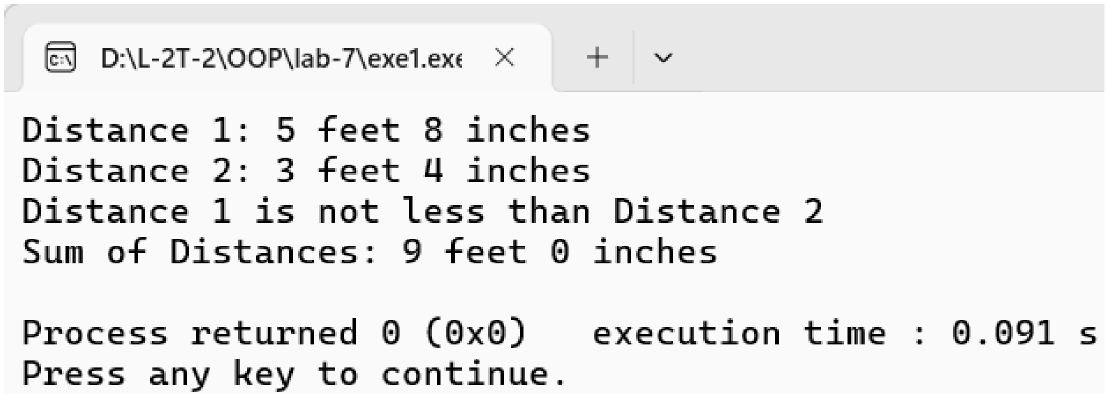
Exercise - 01 : Define a class Distance with distances in feet and inch and with a print function to print the distance.

- Overload the $<$ operator to compare two distances using a member function.
- Overload the $+$ operator to add two distances using a friend function.

Code

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

class Complex {
private:
    float real;
    float imag;

public:
    Complex() {
        real = 0;
        imag = 0;
    }

    void input() {
        cin >> real >> imag;
    }

    Complex operator+(Complex c) {
        Complex temp;
        temp.real = real + c.real;
        temp.imag = imag + c.imag;
        return temp;
    }

    void output() {
        if (imag < 0)
            cout << real << imag << "i" << endl;
        else
            cout << real << "+" << imag << "i" << endl;
    }
};

int main() {
    Complex c1, c2, result;
    c1.input();
    c2.input();
    result = c1 + c2;
    result.output();
    return 0;
}
```

Output



D:\L-2T-2\OOP\lab-7\exe1.exe

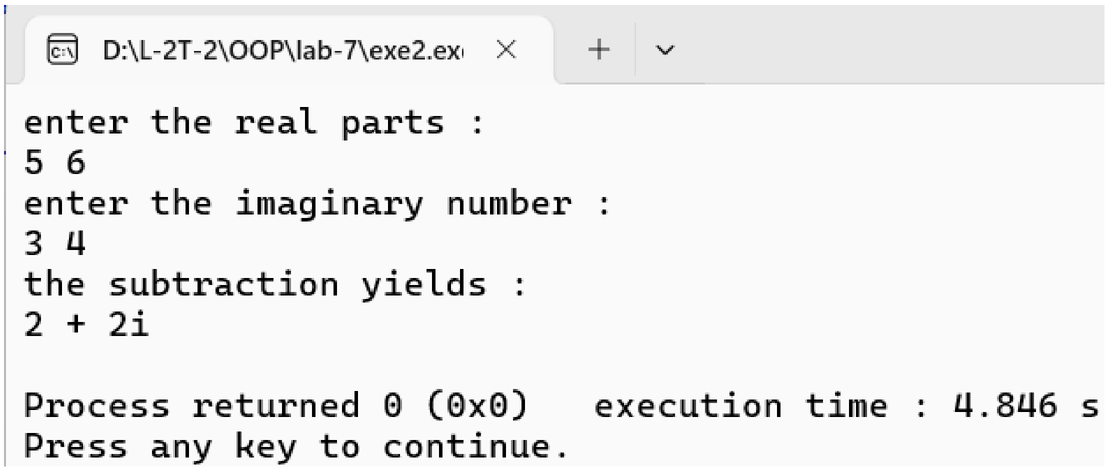
```
Distance 1: 5 feet 8 inches
Distance 2: 3 feet 4 inches
Distance 1 is not less than Distance 2
Sum of Distances: 9 feet 0 inches

Process returned 0 (0x0)   execution time : 0.091 s
Press any key to continue.
```

Exercise - 02 : Write a C++ program to Overload the - operator to subtract two complex numbers.

Code

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

class Complex {
private:
    float real;
    float imag;

public:
    Complex() {
        real = 0;
        imag = 0;
    }

    void input() {
        cin >> real >> imag;
    }

    Complex operator-(Complex c) {
        Complex temp;
        temp.real = real - c.real;
        temp.imag = imag - c.imag;
        return temp;
    }

    void output() {
        if (imag < 0)
            cout << real << imag << "i" << endl;
        else
            cout << real << "+" << imag << "i" << endl;
    }
};

int main() {
    Complex c1, c2, result;
    c1.input();
    c2.input();
    result = c1 - c2;
    result.output();
    return 0;
}
```

Output



D:\L-2T-2\OOP\lab-7\exe2.ex

```
enter the real parts :
5 6
enter the imaginary number :
3 4
the subtraction yields :
2 + 2i

Process returned 0 (0x0)   execution time : 4.846 s
Press any key to continue.
```

Discussion

- The goal of this experiment was to get introduced to the Friend Function and Friend Class in Object-Oriented Programming.
- Some bugs were found while executing the codes. They were fixed eventually with the help of the instructions from the IDE terminal.
- Through examples and exercises, the concept of inheritance was learned and practical understanding of these essential object-oriented programming features in C++ were gained.