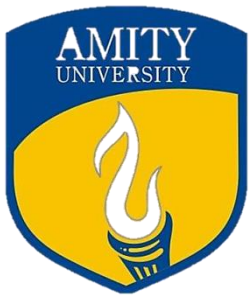
# Object Oriented Programming using C++

## Lab File

Submitted to:

## AMITY UNIVERSITY UTTAR PRADESH




**In partial fulfilment of the requirements for the award of the degree of Bachelor of technology**

**In**

Computer Science & Engineering

By

*Faizan Ahmed Syed*
A23055223216
CSE 4-X Semester 3

Submitted to:

*Dr. Harshit Bhardwaj*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**
**AMITY UNIVERSITY UTTAR PRADESH**
**NOIDA (U.P.)**

# LIST OF EXPERIMENTS

| S.No | Experiment | Date | Page No. | Sign |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

| S.No | Experiment | Date | Page No. | Sign |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

| S.No | Experiment | Date | Page No. | Sign |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Lab 1

## Program:

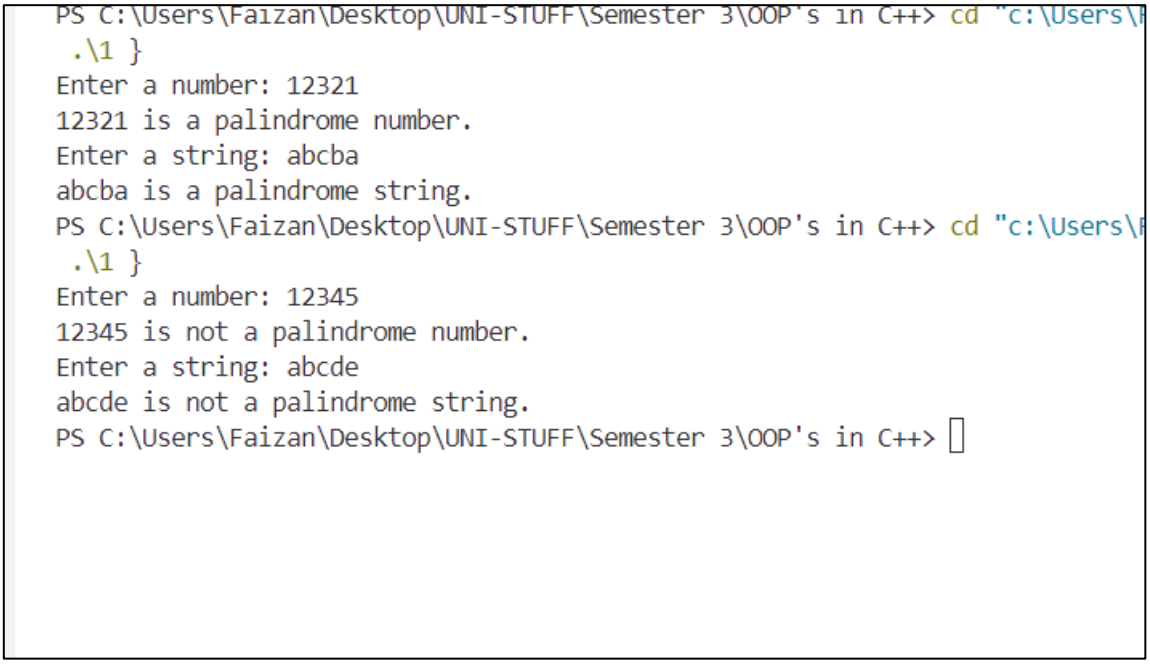
Write a program to check number palindrome and string palindrome.

## Source Code:

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
class PalindromeChecker {
public:
    bool num_palindrome(int number) {
        int originalNumber = number;
        int reversedNumber = 0;

        while (number > 0) {
            int digit = number % 10;
            reversedNumber = reversedNumber * 10 + digit;
            number /= 10;
        }

        return originalNumber == reversedNumber;
    }

    bool str_palindrome(const string& str) {
        int start = 0;
        int end = str.length() - 1;

        while (start < end) {
            if (str[start] != str[end]) {
                return false;
            }
            start++;
            end--;
        }

        return true;
    }
};

int main() {
    PalindromeChecker checker;

    int num;
    cout << "Enter a number: ";
    cin >> num;
```

```
    if (checker.num_palindrome(num)) {
        cout << num << " is a palindrome number." << endl;
    } else {
        cout << num << " is not a palindrome number." << endl;
    }

    string str;
    cout << "Enter a string: ";
    cin >> str;
    if (checker.str_palindrome(str)) {
        cout << str << " is a palindrome string." << endl;
    } else {
        cout << str << " is not a palindrome string." << endl;
    }

    return 0;
}
```

## Output:



# Lab 2

## Program:

Write a program to show the effect of call by value and call by reference functions.

## Source Code:

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

class Test {
public:
    void callByValue(int a) {
        a = a + 10;
        std::cout << "Inside callByValue function, a = " << a << std::endl;
    }

    void callByReference(int& a) {
        a = a + 10;
        std::cout << "Inside callByReference function, a = " << a <<
std::endl;
    }
};

int main() {
    Test testObj;
    int value1 = 20;
    int value2 = 20;

    cout << "Before callByValue, value1 = " << value1 << endl;
    testObj.callByValue(value1);
    cout << "After callByValue, value1 = " << value1 << endl;

    cout << "\nBefore callByReference, value2 = " << value2 << endl;
    testObj.callByReference(value2);
    cout << "After callByReference, value2 = " << value2 << endl;

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop
 .\2 }
Before callByValue, value1 = 20
Inside callByValue function, a = 30
After callByValue, value1 = 20

Before callByReference, value2 = 20
Inside callByReference function, a = 30
After callByReference, value2 = 30
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 3

## Program:

Write a program to perform the matrix operations of addition, subtraction, multiplication, and transpose using functions and switch cases.

## Source Code:

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

class Matrix {
private:
    int rows, cols;
    int matrix[10][10];

public:
    void inputMatrix(int r, int c) {
        rows = r;
        cols = c;
        cout << "Enter the elements of the matrix (" << rows << "x" << cols <<
"):" << endl;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                cout << "Element [" << i << "][" << j << "] = ";
                cin >> matrix[i][j];
            }
        }
    }

    void displayMatrix() {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                cout << "    " << matrix[i][j] << " ";
            }
            cout << endl;
        }
    }

    Matrix addMatrix(Matrix &m) {
        Matrix result;
        if (rows == m.rows && cols == m.cols) {
            result.rows = rows;
            result.cols = cols;
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                    result.matrix[i][j] = matrix[i][j] + m.matrix[i][j];
                }
```

```
            }
        } else {
            cout << "Matrices cannot be added due to dimension mismatch!" <<
endl;
        }
        return result;
    }

    Matrix subtractMatrix(Matrix &m) {
        Matrix result;
        if (rows == m.rows && cols == m.cols) {
            result.rows = rows;
            result.cols = cols;
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                    result.matrix[i][j] = matrix[i][j] - m.matrix[i][j];
                }
            }
        } else {
            cout << "Matrices cannot be subtracted due to dimension mismatch!"
<< endl;
        }
        return result;
    }

    Matrix multiplyMatrix(Matrix &m) {
        Matrix result;
        if (cols == m.rows) {
            result.rows = rows;
            result.cols = m.cols;
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < m.cols; j++) {
                    result.matrix[i][j] = 0;
                    for (int k = 0; k < cols; k++) {
                        result.matrix[i][j] += matrix[i][k] * m.matrix[k][j];
                    }
                }
            }
        } else {
            cout << "Matrices cannot be multiplied due to dimension mismatch!"
<< endl;
        }
        return result;
    }

    Matrix transposeMatrix() {
        Matrix result;
        result.rows = cols;
```

```
        result.cols = rows;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result.matrix[j][i] = matrix[i][j];
            }
        }
        return result;
    }
};

int main() {
    Matrix mat1, mat2, result;
    int choice, r1, c1, r2, c2;
    char contn;

    cout << "Enter the number of rows for Matrix 1: ";
    cin >> r1;
    cout << "Enter the number of columns for Matrix 1: ";
    cin >> c1;
    mat1.inputMatrix(r1, c1);

    cout << "Enter the number of rows for Matrix 2: ";
    cin >> r2;
    cout << "Enter the number of columns for Matrix 2: ";
    cin >> c2;
    mat2.inputMatrix(r2, c2);

    cout << "\nMatrix Operations Menu:" << endl;
    cout << " 1. Addition" << endl;
    cout << " 2. Subtraction" << endl;
    cout << " 3. Multiplication" << endl;
    cout << " 4. Transpose of Matrix 1" << endl;
    cout << " 5. Transpose of Matrix 2" << endl;

    do {
        cout << "Enter your choice (1-5): ";
        cin >> choice;

        switch (choice) {
            case 1:
                result = mat1.addMatrix(mat2);
                cout << "Result of matrix addition:" << endl;
                result.displayMatrix();
                break;

            case 2:
                result = mat1.subtractMatrix(mat2);
                cout << "Result of matrix subtraction:" << endl;
```

```
                result.displayMatrix();
                break;

            case 3:
                result = mat1.multiplyMatrix(mat2);
                cout << "Result of matrix multiplication:" << endl;
                result.displayMatrix();
                break;

            case 4:
                result = mat1.transposeMatrix();
                cout << "Transpose of Matrix 1:" << endl;
                result.displayMatrix();
                break;

            case 5:
                result = mat2.transposeMatrix();
                cout << "Transpose of Matrix 2:" << endl;
                result.displayMatrix();
                break;

            default:
                cout << "Invalid choice!" << endl;
                break;
        }
        cout << "Do you want to continue(Y/N): ";
        cin >> contn;
    }
    while(contn == 'Y' || contn == 'y');

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop
 .\3 }
Enter the number of rows for Matrix 1: 3
Enter the number of columns for Matrix 1: 3
Enter the elements of the matrix (3x3):
Element [0][0] = 1
Element [0][1] = 2
Element [0][2] = 3
Element [1][0] = 4
Element [1][1] = 5
Element [1][2] = 6
Element [2][0] = 7
Element [2][1] = 8
Element [2][2] = 9
Enter the number of rows for Matrix 2: 3
Enter the number of columns for Matrix 2: 3
Enter the elements of the matrix (3x3):
Element [0][0] = 9
Element [0][1] = 8
Element [0][2] = 7
Element [1][0] = 6
Element [1][1] = 5
Element [1][2] = 4
Element [2][0] = 3
Element [2][1] = 2
Element [2][2] = 1

Matrix Operations Menu:
 1. Addition
 2. Subtraction
 3. Multiplication
 4. Transpose of Matrix 1
 5. Transpose of Matrix 2
Enter your choice (1-5): 1
Result of matrix addition:
   10    10    10
```

```
 2. Subtraction
 3. Multiplication
 4. Transpose of Matrix 1
 5. Transpose of Matrix 2
Enter your choice (1-5): 1
Result of matrix addition:
   10    10    10
   10    10    10
   10    10    10
Do you want to continue(Y/N): y
Enter your choice (1-5): 2
Result of matrix subtraction:
   -8    -6    -4
   -2    0    2
   4    6    8
Do you want to continue(Y/N): y
Enter your choice (1-5): 3
Result of matrix multiplication:
   30    24    18
   84    69    54
   138    114    90
Do you want to continue(Y/N): y
Enter your choice (1-5): 4
Transpose of Matrix 1:
   1    4    7
   2    5    8
   3    6    9
Do you want to continue(Y/N): y
Enter your choice (1-5): 5
Transpose of Matrix 2:
   9    6    3
   8    5    2
   7    4    1
Do you want to continue(Y/N): n
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI STUFF\Semester 3\OOP's in C++>
```

# Lab 4

## Program:

Define a class whose attributes are radius, length, and width. Calculate the perimeter of a rectangle and circle. Use constructors and destructors.

## Source Code:

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

class Shape {
private:
    double radius, length, width;

public:
    Shape(double r) : radius(r), length(0), width(0) {
        cout << "Circle object created!" << endl;
    }

    Shape(double l, double w) : radius(0), length(l), width(w) {
        cout << "Rectangle object created!" << endl;
    }

    double circlePerimeter() {
        return 2 * 3.1416 * radius;
    }

    double rectanglePerimeter() {
        return 2 * (length + width);
    }

    ~Shape() {
        cout << "Shape object destroyed!" << endl;
    }
};

int main() {
    double r;
    cout << "Enter the radius of the circle: ";
    cin >> r;
    Shape circle(r);
    cout << "Perimeter of the circle: " << circle.circlePerimeter() << endl;

    double l, w;
    cout << "\nEnter the length of the rectangle: ";
    cin >> l;
    cout << "Enter the width of the rectangle: ";
```

```
    cin >> w;
    Shape rectangle(l, w);
    cout << "Perimeter of the rectangle: " << rectangle.rectanglePerimeter()
<< endl;

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop
 .\4 }
Enter the radius of the circle: 7
Circle object created!
Perimeter of the circle: 43.9824

Enter the length of the rectangle: 10
Enter the width of the rectangle: 5
Rectangle object created!
Perimeter of the rectangle: 30
Program terminated successfully...
Shape object destroyed!
Shape object destroyed!
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 5

## Program:

Create a class Person which includes a character array name of size 64, age in numeric, character array address of size 64, and total salary in real numbers (divide salary in different components, if required). Make an array of objects of class Person of size 10.

- Write an inline function that obtains the youngest and eldest of a person from a class person using arrays.
- Write a program to generate a salary slip and display result by using constructures.

## Source Code:

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

class Person {
private:
    char name[64];
    int age;
    char address[64];
    float basicSalary;
    float hra;
    float da;
    float totalSalary;

public:
    Person(const char* n, int a, const char* addr, float basic) {
        strcpy(name, n);
        age = a;
        strcpy(address, addr);
        basicSalary = basic;
        hra = 0.2 * basicSalary;
        da = 0.1 * basicSalary;
        totalSalary = basicSalary + hra + da;
    }

    inline int getAge() const {
        return age;
    }

    inline void displaySalarySlip() const {
        cout << "------------------------------\n";
        cout << "Salary Slip for " << name << ":\n";
        cout << "Address: " << address << endl;
```

```
        cout << "Basic Salary: " << basicSalary << endl;
        cout << "HRA (20%): " << hra << endl;
        cout << "DA (10%): " << da << endl;
        cout << "Total Salary: " << totalSalary << endl;
        cout << "-------------------------------\n";
    }

    static inline Person* findYoungest(Person persons[], int size) {
        Person* youngest = &persons[0];
        for (int i = 1; i < size; ++i) {
            if (persons[i].getAge() < youngest->getAge()) {
                youngest = &persons[i];
            }
        }
        return youngest;
    }

    static inline Person* findEldest(Person persons[], int size) {
        Person* eldest = &persons[0];
        for (int i = 1; i < size; ++i) {
            if (persons[i].getAge() > eldest->getAge()) {
                eldest = &persons[i];
            }
        }
        return eldest;
    }
};

int main() {
    Person persons[10] = {
        Person("John Doe", 28, "123 Street A", 30000),
        Person("Jane Smith", 35, "456 Street B", 45000),
        Person("Alice Green", 24, "789 Street C", 25000),
        Person("Bob Brown", 45, "101 Street D", 60000),
        Person("Charlie Blue", 30, "102 Street E", 32000),
        Person("David White", 40, "103 Street F", 50000),
        Person("Emma Black", 50, "104 Street G", 55000),
        Person("Frank Red", 23, "105 Street H", 27000),
        Person("Grace Yellow", 31, "106 Street I", 40000),
        Person("Hannah Purple", 38, "107 Street J", 47000)
    };

    Person* youngest = Person::findYoungest(persons, 10);
    cout << "The youngest person is:\n";
    youngest->displaySalarySlip();

    Person* eldest = Person::findEldest(persons, 10);
    cout << "The eldest person is:\n";
```

```
    eldest->displaySalarySlip();

    cout << "\nSalary slips of all persons:\n";
    for (int i = 0; i < 10; ++i) {
        persons[i].displaySalarySlip();
    }

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop\
 .\5 }
The youngest person is:
------------------------------
Salary Slip for Frank Red:
Address: 105 Street H
Basic Salary: 27000
HRA (20%): 5400
DA (10%): 2700
Total Salary: 35100
------------------------------
The eldest person is:
------------------------------
Salary Slip for Emma Black:
Address: 104 Street G
Basic Salary: 55000
HRA (20%): 11000
DA (10%): 5500
Total Salary: 71500
------------------------------

Salary slips of all persons:
------------------------------
Salary Slip for John Doe:
Address: 123 Street A
Basic Salary: 30000
HRA (20%): 6000
DA (10%): 3000
Total Salary: 39000
------------------------------
------------------------------
Salary Slip for Jane Smith:
Address: 456 Street B
Basic Salary: 45000
HRA (20%): 9000
DA (10%): 4500
Total Salary: 58500
```

# Lab 6

## Program:

Create a class called Complex to perform the following operations:

- Overload increment and decrement operators for increasing and decreasing complex number values (unary operator overload).
- Overload the '+' operator and '-' operator for complex numbers (binary operator overloading).

## Source Code:

```
#include <iostream>
using namespace std;
class Complex {
private:
    float real, imag;

public:
    Complex(float r = 0, float i = 0) : real(r), imag(i) {}

    void input() {
        cout << "Enter real part: ";
        cin >> real;
        cout << "Enter imaginary part: ";
        cin >> imag;
    }

    void display() const {
        if (imag >= 0)
            cout << real << " + " << imag << "i" << endl;
        else
            cout << real << " - " << -imag << "i" << endl;
    }

    Complex& operator++() {
        ++real;
        ++imag;
        return *this;
    }

    Complex& operator--() {
        --real;
        --imag;
        return *this;
    }
```

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

    Complex operator-(const Complex& c) const {
        Complex temp;
        temp.real = real - c.real;
        temp.imag = imag - c.imag;
        return temp;
    }
};
int main() {
    Complex c1, c2, result;

    cout << "Enter the first complex number:\n";
    c1.input();

    cout << "\nEnter the second complex number:\n";
    c2.input();

    cout << "\nComplex Number 1: ";
    c1.display();
    cout << "Complex Number 2: ";
    c2.display();

    cout << "\nIncrementing Complex Number 1: ";
    ++c1;
    c1.display();

    cout << "Decrementing Complex Number 2: ";
    --c2;
    c2.display();

    result = c1 + c2;
    cout << "\nSum of Complex Number 1 and 2: ";
    result.display();

    result = c1 - c2;
    cout << "Difference of Complex Number 1 and 2: ";
    result.display();

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop\
 .\6 }
Enter the first complex number:
Enter real part: 14
Enter imaginary part: 8

Enter the second complex number:
Enter real part: 6
Enter imaginary part: 9

Complex Number 1: 14 + 8i
Complex Number 2: 6 + 9i

Incrementing Complex Number 1: 15 + 9i
Decrementing Complex Number 2: 5 + 8i

Sum of Complex Number 1 and 2: 20 + 17i
Difference of Complex Number 1 and 2: 10 + 1i
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 7

## Program:

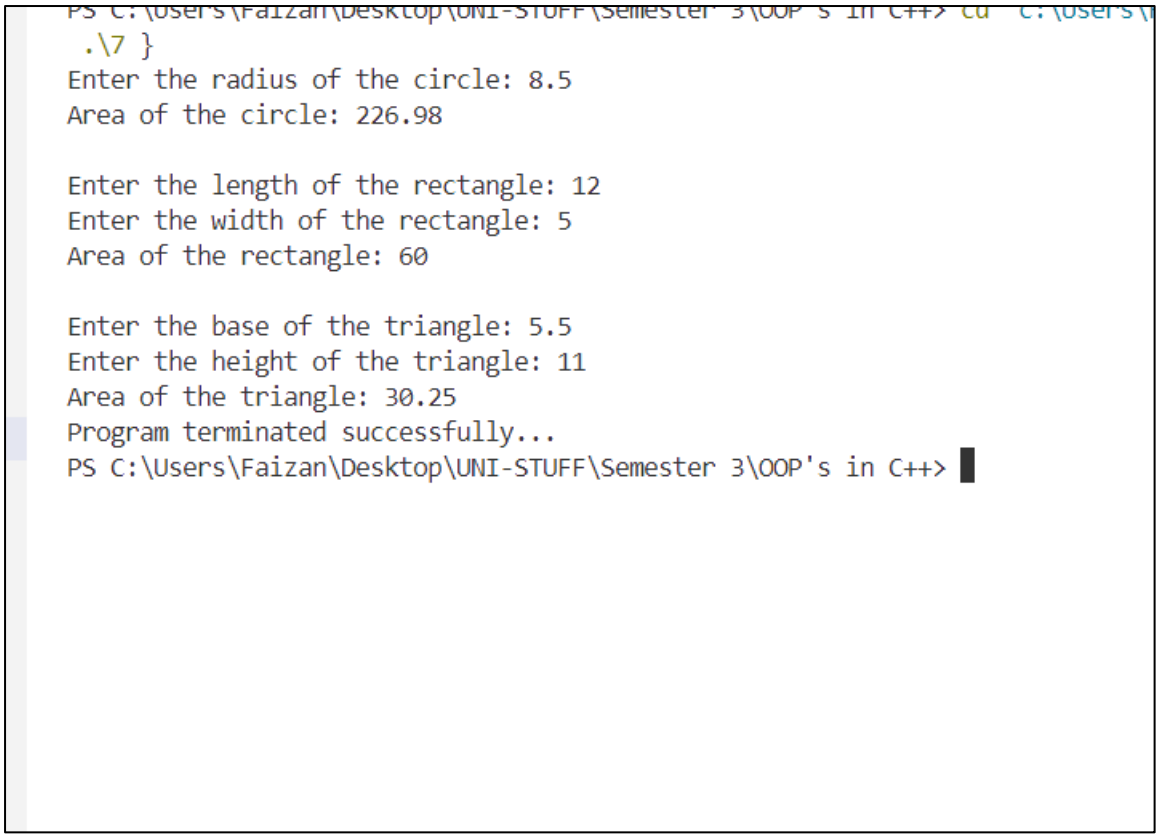
Write a program to find the area (function name AREA) of circle, rectangle, and triangle by function overloading concept.

## Source Code:

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

class Shape {
public:
    double AREA(double radius) {
        return M_PI * pow(radius, 2);
    }

    double AREA(double length, double width) {
        return length * width;
    }

    double AREA(double base, double height, bool isTriangle) {
        if (isTriangle)
            return 0.5 * base * height;
        return 0;
    }
};

int main() {
    Shape shape;
    double radius, length, width, base, height;

    cout << "Enter the radius of the circle: ";
    cin >> radius;
    cout << "Area of the circle: " << shape.AREA(radius) << endl;

    cout << "\nEnter the length of the rectangle: ";
    cin >> length;
    cout << "Enter the width of the rectangle: ";
    cin >> width;
    cout << "Area of the rectangle: " << shape.AREA(length, width) << endl;

    cout << "\nEnter the base of the triangle: ";
    cin >> base;
    cout << "Enter the height of the triangle: ";
    cin >> height;
```

```
    cout << "Area of the triangle: " << shape.AREA(base, height, true) <<
endl;

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:



# Lab 8

## Program:

Design three classes: Student, Exam, and Result. The Student class has data members such as name, roll number, etc. Create a class Exam by inheriting the Student class. The Exam class adds data members representing marks scored in six subjects. Derive the Result class from the Exam class and it has its own data members such as total marks. Write an interactive program to model this relationship. What type of inheritance does this program belong to?

## Source Code:

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

class Student {
protected:
    string name;
    int rollNumber;

public:
    void getStudentDetails() {
        cout << "Enter student name: ";
        cin >> name;
        cout << "Enter roll number: ";
        cin >> rollNumber;
    }

    void displayStudentDetails() {
        cout << "Student Name: " << name << endl;
        cout << "Roll Number: " << rollNumber << endl;
    }
};

class Exam : public Student {
protected:
    float marks[6];

public:
    void getMarks() {
        cout << "Enter marks for 6 subjects: " << endl;
        for (int i = 0; i < 6; i++) {
            cout << "Subject " << i + 1 << ": ";
            cin >> marks[i];
        }
    }

    void displayMarks() {
```

```
        cout << "Marks in 6 subjects: " << endl;
        for (int i = 0; i < 6; i++) {
            cout << "Subject " << i + 1 << ": " << marks[i] << endl;
        }
    }
};

class Result : public Exam {
private:
    float totalMarks;
    char grade;
    float averagePercentage, percentage;

public:
    void total() {
        totalMarks = 0;
        for (int i = 0; i < 6; i++) {
            totalMarks += marks[i];
        }
    }

    void assignGrade() {
        if (averagePercentage > 95 && averagePercentage <= 100) {
            grade = 'A+';
        } else if (averagePercentage > 90 && averagePercentage <= 95) {
            grade = 'A';
        } else if (averagePercentage > 80 && averagePercentage <= 90) {
            grade = 'B';
        } else if (averagePercentage > 70 && averagePercentage <= 80) {
            grade = 'C';
        } else if (averagePercentage > 60 && averagePercentage <= 70) {
            grade = 'D';
        } else {
            grade = 'F';
        }
    }

    void calcPercentAvg() {
        cout << "Percentage Marks in each subject: " << endl;
        for (int i = 0; i < 6; i++) {
            percentage = (marks[i] / 100) * 100;
            cout << "Subject " << i + 1 << ": " << percentage << "%" << endl;
        }

        averagePercentage = totalMarks / 6;
        cout << "Average Percentage: " << averagePercentage << "%" << endl;

        assignGrade();
```

```
    }

    void displayResult() {
        displayStudentDetails();
        displayMarks();
        cout << "Total Marks: " << totalMarks << endl;
        calcPercentAvg();
        cout << "Grade: " << grade << endl;
    }
};

int main() {
    Result s1;

    s1.getStudentDetails();
    s1.getMarks();
    s1.total();

    cout << "\n--- Result ---\n";
    s1.displayResult();

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
                    ^~~~
8.cpp: In member function 'void Result::assignGrade()':
8.cpp:60:21: warning: overflow in implicit constant conversion [-Woverflow]
Enter student name: John
Enter roll number: 3126
Enter marks for 6 subjects:
Subject 1: 85
Subject 2: 87
Subject 3: 96
Subject 4: 94
Subject 5: 93
Subject 6: 88

--- Result ---
Student Name: John
Roll Number: 3126
Marks in 6 subjects:
Subject 1: 85
Subject 2: 87
Subject 3: 96
Subject 4: 94
Subject 5: 93
Subject 6: 88
Total Marks: 543
Percentage Marks in each subject:
Subject 1: 85%
Subject 2: 87%
Subject 3: 96%
Subject 4: 94%
Subject 5: 93%
Subject 6: 88%
Average Percentage: 90.5%
Grade: A
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 9

## Program:

Write a program to swap two numbers (create two classes) by using a friend function.

## Source Code:

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

class Second;

class First {
private:
    int num1;

public:
    First(int n) : num1(n) {}

    void display() {
        cout << "Number 1: " << num1 << endl;
    }

    friend void swapNumbers(First &f, Second &s);
};

class Second {
private:
    int num2;

public:
    Second(int n) : num2(n) {}

    void display() {
        cout << "Number 2: " << num2 << endl;
    }

    friend void swapNumbers(First &f, Second &s);
};

void swapNumbers(First &f, Second &s) {
    int temp = f.num1;
    f.num1 = s.num2;
    s.num2 = temp;
}

int main() {
    int a, b;
```

```
    cout << "Enter the first number: ";
    cin >> a;

    cout << "Enter the second number: ";
    cin >> b;

    First first(a);
    Second second(b);

    cout << "\nBefore swapping:" << endl;
    first.display();
    second.display();

    swapNumbers(first, second);

    cout << "\nAfter swapping:" << endl;
    first.display();
    second.display();

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
 .\9 }
Enter the first number: 10
Enter the second number: 30

Before swapping:
Number 1: 10
Number 2: 30

After swapping:
Number 1: 30
Number 2: 10
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 10

## Program:

Consider an example of a book shop that sells books and video tapes. These two classes are inherited from the base class called media. The media class has command data members such as title and publication. The book class has data members for storing the number of pages in a book and the tape class has playing time in each tape. Each class will have member functions such as read() and show(). In the base class, these members have to be defined as virtual functions.

Write a program that models the hierarchy for a book shop and processes the objects of the classes using pointers to base class. Write the rules of virtual functions.

## Source Code:

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

class Media {
protected:
    string title;
    string publication;

public:
    virtual void read() {
        cout << "Enter title: ";
        cin.ignore();
        getline(cin, title);
        cout << "Enter publication: ";
        getline(cin, publication);
    }

    virtual void show() {
        cout << "Title: " << title << endl;
        cout << "Publication: " << publication << endl;
    }

    virtual ~Media() {}
};

class Book : public Media {
private:
    int numPages;

public:
```

```
    void read() override {
        Media::read();
        cout << "Enter number of pages: ";
        cin >> numPages;
    }

    void show() override {
        Media::show();
        cout << "Number of pages: " << numPages << endl;
    }
};

class Tape : public Media {
private:
    float playTime;

public:
    void read() override {
        Media::read();
        cout << "Enter playing time (in minutes): ";
        cin >> playTime;
    }

    void show() override {
        Media::show();
        cout << "Playing time: " << playTime << " minutes" << endl;
    }
};

int main() {
    const int size = 4;
    Media* mediaList[size];

    for (int i = 0; i < size; i++) {
        int choice;
        cout << "\nEnter 1 for Book, 2 for Tape: ";
        cin >> choice;

        if (choice == 1) {
            mediaList[i] = new Book();
        } else {
            mediaList[i] = new Tape();
        }
        mediaList[i]->read();
    }

    cout << "\nMedia Details:\n";
    for (int i = 0; i < size; i++) {
```

```
        mediaList[i]->show();
        cout << endl;
        delete mediaList[i];
    }

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
Enter 1 for Book, 2 for Tape: 1
Enter title: Digital Electronics for Engineers
Enter publication: M. Morris Mano
Enter number of pages: 682

Enter 1 for Book, 2 for Tape: 1
Enter title: Higher Engineering Mathematics
Enter publication: H.K Dass
Enter number of pages: 1089

Enter 1 for Book, 2 for Tape: 1
Enter title: Let us C
Enter publication: Yashwant K.
Enter number of pages: 452

Enter 1 for Book, 2 for Tape: 2
Enter title: The Daily News
Enter publication: News Times
Enter playing time (in minutes): 95.2

Media Details:
Title: Digital Electronics for Engineers
Publication: M. Morris Mano
Number of pages: 682

Title: Higher Engineering Mathematics
Publication: H.K Dass
Number of pages: 1089

Title: Let us C
Publication: Yashwant K.
Number of pages: 452

Title: The Daily News
Publication: News Times
Playing time: 95.2 minutes

Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 11

## Program:

Write a program to find the total marks of a student using the concept of virtual base class.

## Source Code:

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

class Student {
protected:
    string name;
    int rollNumber;

public:
    Student() {
        cout << "Enter student name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter roll number: ";
        cin >> rollNumber;
    }
};

class Exam : virtual public Student {
public:
    int marks1, marks2, marks3;

    void inputMarks() {
        cout << "Enter marks for subject 1: ";
        cin >> marks1;
        cout << "Enter marks for subject 2: ";
        cin >> marks2;
        cout << "Enter marks for subject 3: ";
        cin >> marks3;
    }
};

class Result : virtual public Student {
private:
    int totalMarks;

public:
    void calculateTotalMarks(Exam &exam) {
        totalMarks = exam.marks1 + exam.marks2 + exam.marks3;
    }
```

```
    void displayResult() {
        cout << "Total Marks for " << name << " (Roll Number: " << rollNumber
<< "): " << totalMarks << endl;
    }
};

int main() {
    Exam exam;
    exam.inputMarks();

    Result result;
    result.calculateTotalMarks(exam);
    result.displayResult();

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan
 { .\11 }
Enter student name: Josh
Enter roll number: 1001
Enter marks for subject 1: 85
Enter marks for subject 2: 94
Enter marks for subject 3: 93
Enter student name: Josh
Enter roll number: 1001
Total Marks for Josh (Roll Number: 1001): 272
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 12

## Program:

Write a program to show the use of ‘this’ pointer. Write the application of this pointer.

## Source Code:

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

class Box {
private:
    int length;
    int width;
    int height;

public:
    Box(int l, int w, int h) {
        this->length = l;
        this->width = w;
        this->height = h;
    }

    int volume() {
        return this->length * this->width * this->height;
    }

    bool isLargerThan(Box &b) {
        return this->volume() > b.volume();
    }

    void display() {
        cout << "Length: " << this->length << ", Width: " << this->width << ",
Height: " << this->height << endl;
    }
};

int main() {
    Box box1(3, 4, 5);
    Box box2(2, 6, 4);

    cout << "Box 1 dimensions: ";
    box1.display();
    cout << "Volume of Box 1: " << box1.volume() << endl;

    cout << "Box 2 dimensions: ";
    box2.display();
    cout << "Volume of Box 2: " << box2.volume() << endl;
```

```
    if (box1.isLargerThan(box2)) {
        cout << "Box 1 is larger than Box 2." << endl;
    } else {
        cout << "Box 2 is larger than or equal to Box 1." << endl;
    }

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\F
 { .\12 }
Box 1 dimensions: Length: 3, Width: 4, Height: 5
Volume of Box 1: 60
Box 2 dimensions: Length: 2, Width: 6, Height: 4
Volume of Box 2: 48
Box 1 is larger than Box 2.
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 13

## Program:

Write a program to implement stack functions using templates.

## Source Code:

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

template <typename T>
class Stack {
private:
    T* arr;
    int top;
    int capacity;

public:
    Stack(int size) {
        capacity = size;
        arr = new T[capacity];
        top = -1;
    }

    ~Stack() {
        delete[] arr;
    }

    void push(T element) {
        if (top == capacity - 1) {
            cout << "Stack Overflow! Cannot push " << element << endl;
            return;
        }
        arr[++top] = element;
        cout << element << " pushed to stack." << endl;
    }

    T pop() {
        if (top == -1) {
            cout << "Stack Underflow! Cannot pop from an empty stack." <<
endl;
            return T();
        }
        return arr[top--];
    }

    T peek() {
        if (top == -1) {
```

```
            cout << "Stack is empty. Cannot peek." << endl;
            return T();
        }
        return arr[top];
    }

    bool isEmpty() {
        return top == -1;
    }

    int size() {
        return top + 1;
    }
};

int main() {
    Stack<int> intStack(5);
    int choice;
    int element;

        cout << "\nStack Operations Menu:\n";
        cout << "1. Push\n";
        cout << "2. Pop\n";
        cout << "3. Peek\n";
        cout << "4. Check if Empty\n";
        cout << "5. Get Size\n";
        cout << "6. Exit\n";

    do {
        cout << "Select an operation: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter a value to push: ";
                cin >> element;
                intStack.push(element);
                break;
            case 2:
                cout << intStack.pop() << " popped from stack." << endl;
                break;
            case 3:
                cout << "Top element: " << intStack.peek() << endl;
                break;
            case 4:
                if (intStack.isEmpty()) {
                    cout << "Stack is empty." << endl;
                } else {
```

```
                    cout << "Stack is not empty." << endl;
                }
                break;
            case 5:
                cout << "Current stack size: " << intStack.size() << endl;
                break;
            case 6:
                cout << "Exiting program." << endl;
                break;
            default:
                cout << "Invalid choice. Please select again." << endl;
                break;
        }
    } while (choice != 6);

    cout << "Program terminated successflly..." << endl;

    return 0;
}
```

**Output:**

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Get Size
6. Exit
Select an operation: 1
Enter a value to push: 10
10 pushed to stack.
Select an operation: 1
Enter a value to push: 20
20 pushed to stack.
Select an operation: 1
Enter a value to push: 30
30 pushed to stack.
Select an operation: 5
Current stack size: 3
Select an operation: 2
30 popped from stack.
Select an operation: 3
Top element: 20
Select an operation: 2
20 popped from stack.
Select an operation: 2
10 popped from stack.
Select an operation: 4
Stack is empty.
Select an operation: 6
Exiting program.
Program terminated successflly...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 14

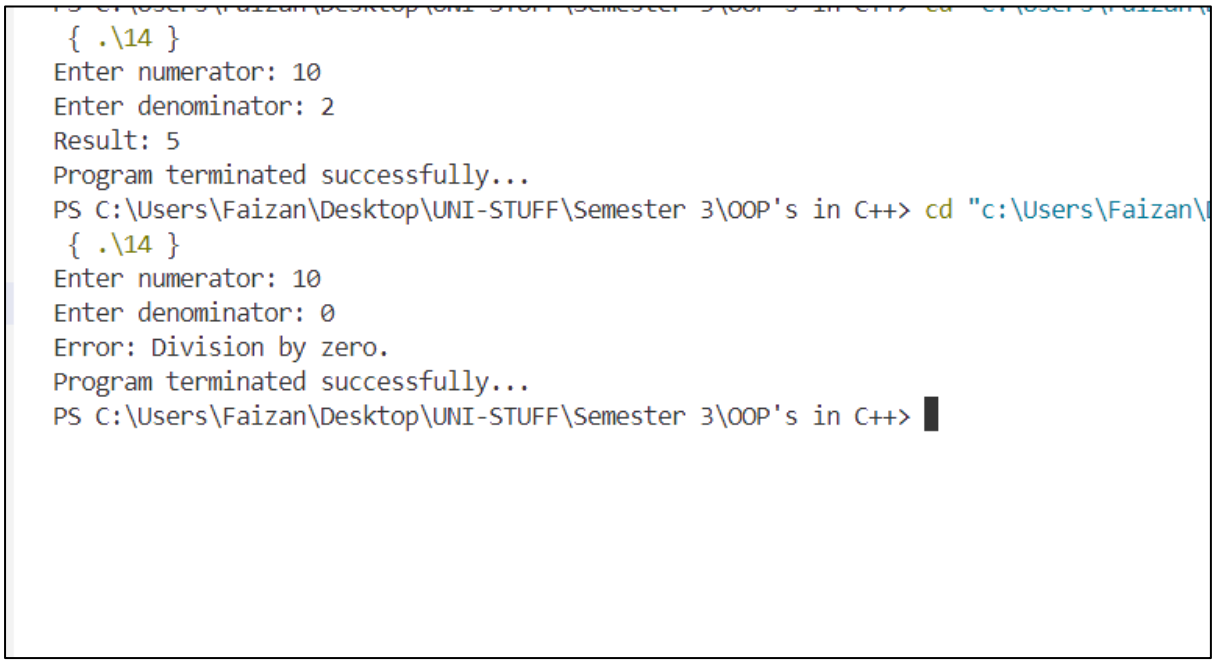
## Program:

Write a program to demonstrate exception handling.

## Source Code:

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

class DivisionException : public exception {
public:
    const char* what() const noexcept override {
        return "Error: Division by zero.";
    }
};

double divide(double numerator, double denominator) {
    if (denominator == 0) {
        throw DivisionException();
    }
    return numerator / denominator;
}

int main() {
    double num1, num2;

    cout << "Enter numerator: ";
    cin >> num1;
    cout << "Enter denominator: ";
    cin >> num2;

    try {
        double result = divide(num1, num2);
        cout << "Result: " << result << endl;
    } catch (const DivisionException& e) {
        cout << e.what() << endl;
    } catch (const exception& e) {
        cout << "An error occurred: " << e.what() << endl;
    }

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

**Output:**


{ .\14 }
Enter numerator: 10
Enter denominator: 2
Result: 5
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\
{ .\14 }
Enter numerator: 10
Enter denominator: 0
Error: Division by zero.
Program terminated successfully...
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>

# Lab 15

## Program:

Write a program that inputs a file, which determines its length, and also counts the number of word occurrences.

## Source Code:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <map>
#include <string>

using namespace std;

void count(const string& filename) {
    ifstream file(filename);

    if (!file.is_open()) {
        cout << "Error opening file!" << endl;
        return;
    }

    string line;
    int charCount = 0;
    map<string, int> wordCount;

    while (getline(file, line)) {
        charCount += line.length();
        stringstream ss(line);
        string word;

        while (ss >> word) {
            wordCount[word]++;
        }
    }

    file.close();

    cout << "Character count: " << charCount << endl;
    cout << "Word occurrences:" << endl;

    for (const auto& entry : wordCount) {
        cout << entry.first << ": " << entry.second << endl;
    }
}
```

```
int main() {
    string filename;

    cout << "Enter the filename: ";
    cin >> filename;

    count(filename);

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\D
 { .\15 }
Enter the filename: file.txt
Character count: 233
Word occurrences:
C++.: 1
File: 1
It: 1
Let's: 1
This: 2
a: 1
an: 1
and: 1
characters: 1
concept: 1
contains: 1
count: 1
file: 2
file.: 2
for: 1
fourth: 1
handling: 2
important: 1
in: 3
is: 3
line: 1
lines: 1
number: 1
of: 3
programming.: 1
several: 1
test: 2
text.: 1
the: 3
this: 1
words: 1
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 16

## Program:

Write a program to demonstrate file handling: Creating a binary file to store student details.

## Source Code:

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

struct Student {
    char name[50];
    int rollNumber;
    float marks;
};

void writeToFile(const string& filename) {
    ofstream outFile(filename, ios::binary);

    if (!outFile) {
        cout << "Error opening file for writing!" << endl;
        return;
    }

    Student student;
    char choice;

    do {
        cout << "Enter student name: ";
        cin >> student.name;
        cout << "Enter roll number: ";
        cin >> student.rollNumber;
        cout << "Enter marks: ";
        cin >> student.marks;

        outFile.write(reinterpret_cast<char*>(&student), sizeof(student));

        cout << "Do you want to add another student? (y/n): ";
        cin >> choice;

    } while (choice == 'y' || choice == 'Y');

    outFile.close();
    cout << "Student details saved to file." << endl;
}
```

```
void readFromFile(const string& filename) {
    ifstream inFile(filename, ios::binary);

    if (!inFile) {
        cout << "Error opening file for reading!" << endl;
        return;
    }

    Student student;

    cout << "\nStudent Details:\n";
    while (inFile.read(reinterpret_cast<char*>(&student), sizeof(student))) {
        cout << "Name: " << student.name << ", Roll Number: " <<
student.rollNumber << ", Marks: " << student.marks << endl;
    }

    inFile.close();
}

int main() {
    string filename = "students.dat";

    writeToFile(filename);
    readFromFile(filename);

    return 0;
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Des
 { .\16 }
Enter student name: Ted
Enter roll number: 1010
Enter marks: 95.25
Do you want to add another student? (y/n): y
Enter student name: Jane
Enter roll number: 1021
Enter marks: 97.18
Do you want to add another student? (y/n): y
Enter student name: Joe
Enter roll number: 1022
Enter marks: 87.75
Do you want to add another student? (y/n): n
Student details saved to file.

Student Details:
Name: Ted, Roll Number: 1010, Marks: 95.25
Name: Jane, Roll Number: 1021, Marks: 97.18
Name: Joe, Roll Number: 1022, Marks: 87.75
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 17

## Program:

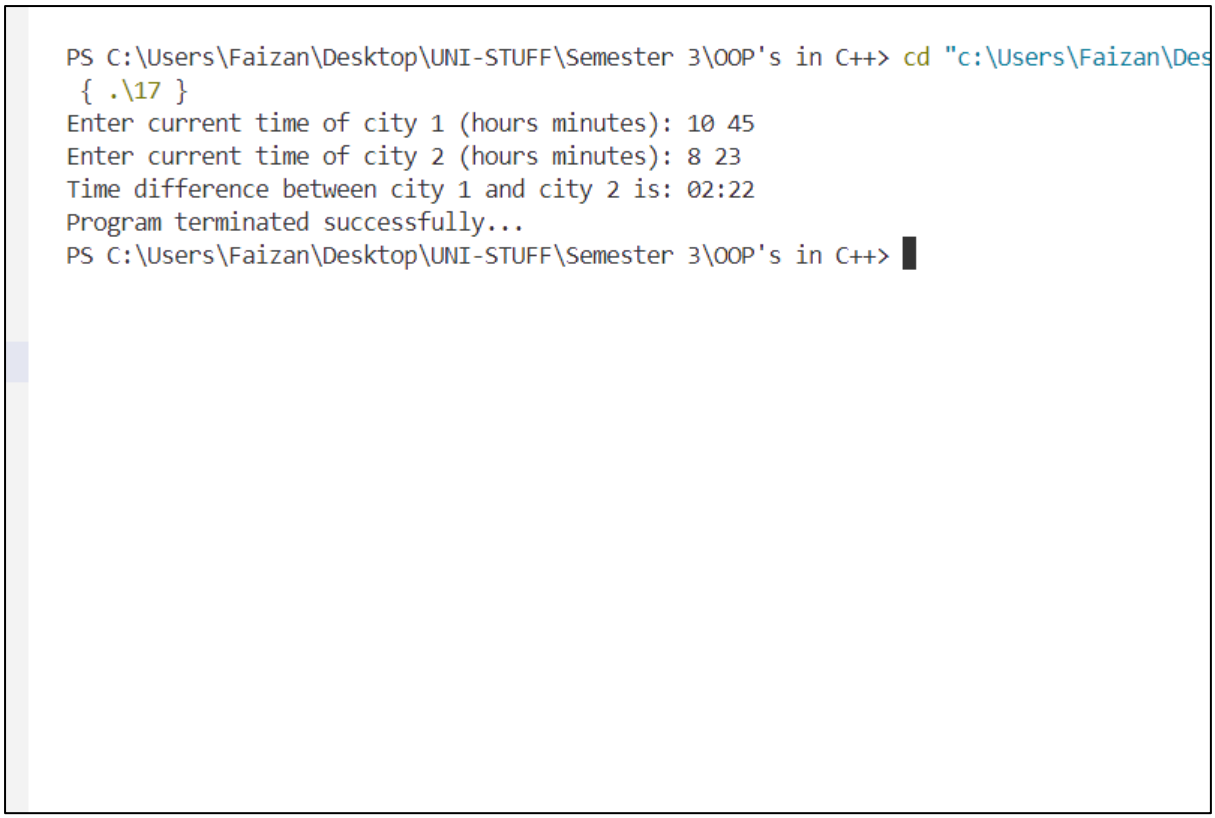
Take time of two zones, current time of city 1 and current time of city 2. Calculate the time difference between two cities. Use the concept of passing objects as parameters.

## Source Code:

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

class Time {
private:
    int hours;
    int minutes;

public:
    Time(int h = 0, int m = 0) : hours(h), minutes(m) {}

    void display() const {
        cout << (hours < 10 ? "0" : "") << hours << ":"
             << (minutes < 10 ? "0" : "") << minutes;
    }

    Time timeDifference(const Time& other) const {
        int totalMinutes1 = hours * 60 + minutes;
        int totalMinutes2 = other.hours * 60 + other.minutes;
        int diffMinutes = totalMinutes1 - totalMinutes2;

        if (diffMinutes < 0) {
            diffMinutes += 24 * 60;
        }

        return Time(diffMinutes / 60, diffMinutes % 60);
    }
};

int main() {
    int h1, m1, h2, m2;

    cout << "Enter current time of city 1 (hours minutes): ";
    cin >> h1 >> m1;
    Time city1(h1, m1);

    cout << "Enter current time of city 2 (hours minutes): ";
    cin >> h2 >> m2;
    Time city2(h2, m2);
```

```
    Time difference = city1.timeDifference(city2);

    cout << "Time difference between city 1 and city 2 is: ";
    difference.display();

    cout << "Program terminated successfully..." << endl;

    return 0;
}
```

## Output:



# Lab 18

## Program:

Write a macro to obtain the largest of three numbers. Do the same using an inline function.

## Source Code:

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

#define MAX(a, b, c) ((a) > (b) ? ((a) > (c) ? (a) : (c)) : ((b) > (c) ? (b) : (c)))

inline int max(int a, int b, int c) {
    return (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);
}

int main() {
    int x, y, z;

    cout << "Enter x: ";
    cin >> x;
    cout << "Enter y: ";
    cin >> y;
    cout << "Enter z: ";
    cin >> z;

    int maxNumMacro = MAX(x, y, z);
    int maxNumInline = max(x, y, z);

    cout << "Largest number using macro: " << maxNumMacro << endl;
    cout << "Largest number using inline function: " << maxNumInline << endl;

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\User
 { .\18 }
Enter x: 10
Enter y: 15
Enter z: 25
Largest number using macro: 25
Largest number using inline function: 25
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 19

## Program:

Write a C++ program to implement a class called BankAccount that has private member variables for account number and balance. Include member functions to deposit and withdraw money from the account.. Use appropriate inheritance for classes such as Bank, Customer, and Employee. Include functions for customers balance, bank total balance, withdraw, deposit, total customers in bank, account information, and functions to create new customer accounts by taking details such as name, account number, address, etc.

## Source Code:

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

class Customer {
    string name;
    int account_No;
    int balance = 0; // Each customer's balance is initialized to 0
    static int total_customers;
    static int Bank_balance;

public:
    void details();
    void deposit();
    void withdraw();
    void show_customers();
    void show_balance();
};

int Customer::total_customers = 0;
int Customer::Bank_balance = 0;

void Customer::details() {
    cout << "Enter name: ";
    cin >> name;
    cout << "Enter account no.: ";
    cin >> account_No;
    total_customers++;
    cout << "Total customers in bank: " << total_customers << endl;
}

void Customer::deposit() {
    int deposit_amount;
    cout << "Enter amount to deposit: ";
    cin >> deposit_amount;
```

```
    balance += deposit_amount;  // Update the customer's balance directly
    Bank_balance += deposit_amount;  // Update bank's balance
    cout << "Amount deposited: " << deposit_amount << endl;
    cout << "Current Balance: " << balance << endl;
}

void Customer::withdraw() {
    int withdraw_amount;
    cout << "Enter amount to withdraw: ";
    cin >> withdraw_amount;

    if (withdraw_amount > balance) {
        cout << "Insufficient balance..." << endl;
    } else {
        balance -= withdraw_amount;  // Update the customer's balance directly
        Bank_balance -= withdraw_amount;  // Update bank's balance
        cout << "Amount withdrawn: " << withdraw_amount << endl;
        cout << "Current Balance: " << balance << endl;
    }
}

void Customer::show_customers() {
    cout << "Total customers: " << total_customers << endl;
    cout << "Bank Total Balance: " << Bank_balance << endl;
}

void Customer::show_balance() {
    cout << "Balance is: " << balance << endl;
}

int main() {
    Customer c1, c2, c3;
    char repeat;

    // Get details of customers
    c1.details();
    c2.details();
    c3.details();

    do {
        int customer;
        char action;

        cout << "Enter customer number (1-3): ";
        cin >> customer;

        switch (customer) {
            case 1:
```

```
                cout << "To Deposit press (d)." << endl << "To Withdraw press
(w)." << endl << "To check balance press (b)." << endl;
                cin >> action;
                switch (action) {
                    case 'd':
                        c1.deposit();
                        break;
                    case 'w':
                        c1.withdraw();
                        break;
                    case 'b':
                        c1.show_balance();
                        break;
                    default:
                        cout << "Action not correct..." << endl;
                        break;
                }
                break;
            case 2:
                cout << "To Deposit press (d)." << endl << "To Withdraw press
(w)." << endl << "To check balance press (b)." << endl;
                cin >> action;
                switch (action) {
                    case 'd':
                        c2.deposit();
                        break;
                    case 'w':
                        c2.withdraw();
                        break;
                    case 'b':
                        c2.show_balance();
                        break;
                    default:
                        cout << "Action not correct..." << endl;
                        break;
                }
                break;
            case 3:
                cout << "To Deposit press (d)." << endl << "To Withdraw press
(w)." << endl << "To check balance press (b)." << endl;
                cin >> action;
                switch (action) {
                    case 'd':
                        c3.deposit();
                        break;
                    case 'w':
                        c3.withdraw();
                        break;
```

```
                    case 'b':
                        c3.show_balance();
                        break;
                    default:
                        cout << "Action not correct..." << endl;
                        break;
                }
                break;
            default:
                cout << "Invalid customer number." << endl;
                break;
        }
        cout << "Do you want to continue (Y/N): ";
        cin >> repeat;
    } while (repeat == 'Y' || repeat == 'y');

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop
 { .\19 }
Enter name: Ted
Enter account no.: 1
Total customers in bank: 1
Enter name: Bob
Enter account no.: 2
Total customers in bank: 2
Enter name: Joe
Enter account no.: 3
Total customers in bank: 3
Enter customer number (1-3): 1
To Deposit press (d).
To Withdraw press (w).
To check balance press (b).
d
Enter amount to deposit: 100
Amount deposited: 100
Current Balance: 100
Do you want to continue (Y/N): y
Enter customer number (1-3): 3
To Deposit press (d).
To Withdraw press (w).
To check balance press (b).
d
Enter amount to deposit: 150
Amount deposited: 150
Current Balance: 150
Do you want to continue (Y/N): y
Enter customer number (1-3): 2
To Deposit press (d).
To Withdraw press (w).
To check balance press (b).
w
Enter amount to withdraw: 120
Insufficient balance...
Do you want to continue (Y/N): y
Enter customer number (1-3): 1
```

```
To check balance press (b).
w
Enter amount to withdraw: 120
Insufficient balance...
Do you want to continue (Y/N): y
Enter customer number (1-3): 1
To Deposit press (d).
To Withdraw press (w).
To check balance press (b).
b
Balance is: 100
Do you want to continue (Y/N): y
Enter customer number (1-3): 3
To Deposit press (d).
To Withdraw press (w).
To check balance press (b).
w
Enter amount to withdraw: 70
Amount withdrawn: 70
Current Balance: 80
Do you want to continue (Y/N): n
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 20

## Program:

Write a C++ program to create a class called Triangle that has private member variables for the lengths of its three sides. Implement member functions to determine if the triangle is equilateral, isosceles, or scalene.

## Source Code:

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

class Triangle {
    float l1;
    float l2;
    float l3;

    public:
        void setLengths();
        void type();
};
void Triangle::setLengths()
{
    cout << "Enter lenght 1: ";
    cin >> l1;
    cout << "Enter length 2: ";
    cin >> l2;
    cout << "Enter length 3: ";
    cin >> l3;
}
void Triangle::type()
{
    if(l1 == l2 && l2 == l3) {
        cout << "Equilateral triangle.";
    }
    else if(l1 == l2 || l2 == l3 || l3 == l1)
    {
        cout << "Isosceles triangle.";
    }
    else if(l1 != l2 && l2 != l3 && l3 != l1)
    {
        cout << "Scalene triangle.";
    }
}

int main()
{
```

```
    Triangle t;

    t.setLengths();
    t.type();

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Deskt
 } ; if ($?) { .\Triangle }
Enter lenght 1: 10
Enter length 2: 10
Enter length 3: 10
Equilateral triangle.
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Deskt
 } ; if ($?) { .\Triangle }
Enter lenght 1: 10
Enter length 2: 20
Enter length 3: 10
Isosceles triangle.
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Deskt
 } ; if ($?) { .\Triangle }
Enter lenght 1: 10
Enter length 2: 20
Enter length 3: 30
Scalene triangle.
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 21

## Program:

Create a class called “TIME” that has three integer data members for hours, minutes and seconds. The class should have a constructor to initialize the object to zero and a constructor to initialize the object to some constant value. The class should have member function to add two TIME objects and member function to display time in HH:MM:SS format. Write a main function to create two TIME objects, add them and display the result in HH:MM:SS format.

## Source Code:

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

class TIME {
private:
    int hours;
    int minutes;
    int seconds;

public:
    TIME() : hours(0), minutes(0), seconds(0) {}

    TIME(int h, int m, int s) : hours(h), minutes(m), seconds(s) {
        normalize();
    }

    void normalize() {
        if (seconds >= 60) {
            minutes += seconds / 60;
            seconds = seconds % 60;
        }
        if (minutes >= 60) {
            hours += minutes / 60;
            minutes = minutes % 60;
        }
    }
    TIME add(const TIME& other) const {
        TIME result;
        result.hours = hours + other.hours;
        result.minutes = minutes + other.minutes;
        result.seconds = seconds + other.seconds;
        result.normalize();
        return result;
    }
```

```
    void display() const {
        cout << "Time: " << hours << " : " << minutes << " : " << seconds <<
endl;
    }
    void input() {
        cout << "Enter hours: ";
        cin >> hours;
        cout << "Enter minutes: ";
        cin >> minutes;
        cout << "Enter seconds: ";
        cin >> seconds;
        normalize();
    }
};
int main() {
    TIME time1, time2;

    cout << "Enter time for first object:\n";
    time1.input();

    cout << "Enter time for second object:\n";
    time2.input();

    TIME totalTime = time1.add(time2);
    totalTime.display();

    return 0;
}
```

**Output:**

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Deskto
 { .\21 }
Enter time for first object:
Enter hours: 4
Enter minutes: 27
Enter seconds: 45
Enter time for second object:
Enter hours: 10
Enter minutes: 32
Enter seconds: 29
Time: 15 : 0 : 14
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 22

## Program:

Create a class box having data member object_count as static. Create a constructor that initializes the data members length, breadth and height and define function volume. Also, create a static member function to display the number of objects created.

## Source Code:

```
#include <iostream>
#include <vector>  // Include the vector header
using namespace std;

class Box {
private:
    float length;
    float breadth;
    float height;
    static int object_count;

public:
    Box() : length(0), breadth(0), height(0) {
        object_count++;
    }

    Box(float l, float b, float h) : length(l), breadth(b), height(h) {
        object_count++;
    }

    float volume() const {
        return length * breadth * height;
    }

    static void displayObjectCount() {
        cout << "Number of Box objects created: " << object_count << endl;
    }
};

int Box::object_count = 0;

int main() {
    int n;
    cout << "Enter the number of boxes: ";
    cin >> n;

    vector<Box> boxes;  // Using vector to store Box objects
```

```
    float length, breadth, height;

    for (int i = 0; i < n; ++i) {
        cout << "Enter dimensions for Box " << (i + 1) << ":\n";
        cout << "Length: ";
        cin >> length;
        cout << "Breadth: ";
        cin >> breadth;
        cout << "Height: ";
        cin >> height;

        boxes.emplace_back(length, breadth, height);  // Use emplace_back to
create Box objects directly in the vector
    }

    for (int i = 0; i < n; ++i) {
        cout << "Volume of Box " << (i + 1) << ": " << boxes[i].volume() <<
endl;
    }

    Box::displayObjectCount();

    return 0;
}
```

**Output:**

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Deskto
 { .\22 }
Enter the number of boxes: 3
Enter dimensions for Box 1:
Length: 10
Breadth: 10
Height: 10
Enter dimensions for Box 2:
Length: 12.5
Breadth: 12.5
Height: 13
Enter dimensions for Box 3:
Length: 14
Breadth: 15
Height: 16
Volume of Box 1: 1000
Volume of Box 2: 2031.25
Volume of Box 3: 3360
Number of Box objects created: 3
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 23

## Program:

Write a program to implement the following inheritances:

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance

## Source Code:

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

class Arithmetic {
public:
    float add(float a, float b) {
        return a + b;
    }

    float subtract(float a, float b) {
        return a - b;
    }

    float multiply(float a, float b) {
        return a * b;
    }

    float divide(float a, float b) {
        if (b != 0)
            return a / b;
        else {
            cout << "Division by zero error!" << endl;
            return 0;
        }
    }
};

class BasicOperations : public Arithmetic {
public:
    void performBasicOperations(float a, float b) {
        cout << "Addition: " << add(a, b) << endl;
        cout << "Subtraction: " << subtract(a, b) << endl;
    }
```

```
};

class AdvancedOperations {
public:
    void performAdvancedOperations(float a, float b) {
        Arithmetic arith;
        cout << "Multiplication: " << arith.multiply(a, b) << endl;
        cout << "Division: " << arith.divide(a, b) << endl;
    }
};

class Operations : public BasicOperations, public AdvancedOperations {
public:
    void performAllOperations(float a, float b) {
        performBasicOperations(a, b);
        performAdvancedOperations(a, b);
    }
};

class MoreOperations : public Arithmetic {
public:
    void performMoreOperations(float a, float b) {
        cout << "Addition (MoreOperations): " << add(a, b) << endl;
    }
};

class ExtendedOperations : public MoreOperations {
public:
    void performExtendedOperations(float a, float b) {
        cout << "Subtraction (ExtendedOperations): " << subtract(a, b) <<
endl;
    }
};

class HybridOperations : public BasicOperations, public ExtendedOperations {
public:
    void performHybridOperations(float a, float b) {
        Operations op;
        op.performAllOperations(a, b);
        performExtendedOperations(a, b);
    }
};

int main() {
    float a, b;

    cout << "Enter value for a: ";
    cin >> a;
```

```
    cout << "Enter value for b: ";
    cin >> b;

    cout << "\nUsing Operations class for all operations:\n";
    Operations op;
    op.performAllOperations(a, b);

    cout << "\nUsing ExtendedOperations class:\n";
    ExtendedOperations extOp;
    extOp.performExtendedOperations(a, b);

    cout << "\nUsing HybridOperations class:\n";
    HybridOperations hybridOp;
    hybridOp.performHybridOperations(a, b);

    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop
 { .\23 }
Enter value for a: 24
Enter value for b: 17

Using Operations class for all operations:
Addition: 41
Subtraction: 7
Multiplication: 408
Division: 1.41176

Using ExtendedOperations class:
Subtraction (ExtendedOperations): 7

Using HybridOperations class:
Addition: 41
Subtraction: 7
Multiplication: 408
Division: 1.41176
Subtraction (ExtendedOperations): 7
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```

# Lab 24

## Program:

Write a program to implement the functions of a simple calculator using functions and switch cases.

## Source Code:

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

class Calculator {
        float a;
        float b;

    public:

        void result();

        float add();
        float subtract();
        float multiply();
        float divide();
};

void Calculator::result()
{
    cout << "Enter a value for A: ";
    cin >> a;
    cout << "Enter a value for B: ";
    cin >> b;
}
float Calculator::add()
{
    return a + b;
}
float Calculator::subtract()
{
    return a - b;
}
float Calculator::multiply()
{
    return a * b;
}
float Calculator::divide()
{
    if(b == 0) {
```

```
        cout << "Division by 0...\n";
        return INFINITY;
    }
    else {
        return a / b;
    }
}

int main()
{
    char contn;
    char symbol;
    Calculator c;

    cout << "********** Simple Calculator **********\n";
    cout << "\nChoose required operation: " << "\n"
    "Enter \'+\' to add a and b." << "\n"
    "Enter \'-\' to subtract a and b." << "\n"
    "Enter \'*\' to multiply a and b." << "\n"
    "Enter \'/\' to divide a and b." << "\n";

    do {
    cout << "\nEnter operator: ";
    cin >> symbol;

    switch(symbol) {
        case '+':
            c.result();
            cout << "Addition is: " << c.add() << endl;
            break;
        case '-':
            c.result();
            cout << "Subtraction is: " << c.subtract() << endl;
            break;
        case '*':
            c.result();
            cout << "Multiplication is: " << c.multiply() << endl;
            break;
        case '/':
            c.result();
            cout << "Division is: " << c.divide() << endl;
            break;
        default:
            cout << "Choose from the given operators...\n" <<
            "INCORRECT OPERATOR!" << endl;
            break;
    }
    cout << "\nDo you want to continue (Y/N): ";
```

```
    cin >> contn;
    } while(contn == 'Y' || contn == 'y');


    return 0;
}
```

## Output:

```
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++> cd "c:\Users\Faizan\Desktop
lator2 } ; if ($?) { .\Calculator2 }
********** Simple Calculator **********

Choose required operation:
Enter '+' to add a and b.
Enter '-' to subtract a and b.
Enter '*' to multiply a and b.
Enter '/' to divide a and b.

Enter operator: +
Enter a value for A: 15
Enter a value for B: 26
Addition is: 41

Do you want to continue (Y/N): y

Enter operator: -
Enter a value for A: 48
Enter a value for B: 17
Subtraction is: 31

Do you want to continue (Y/N): y

Enter operator: *
Enter a value for A: 2
Enter a value for B: 23
Multiplication is: 46

Do you want to continue (Y/N): y

Enter operator: /
Enter a value for A: 49
Enter a value for B: 12
Division is: 4.08333

Do you want to continue (Y/N): n
PS C:\Users\Faizan\Desktop\UNI-STUFF\Semester 3\OOP's in C++>
```