**Exercises: OOP**

**Exercises 1**

Write a program that defines a shape class with a constructor that gives value to width and height. The define two sub-classes triangle and rectangle, that calculate the area of the shape area (). In the main, define two variables a triangle and a rectangle and then call the area() function in this two variables.

[Solution](https://erlerobotics.gitbooks.io/erle-robotics-cpp-gitbook/content/code/8.OOP/e_8.1.cpp)

#include <iostream>

using namespace std;

class Shape

{

protected:

    float width, height;

public:

    void set\_data(float a, float b)

    {

        width = a;

        height = b;

    }

};

class Rectangle : public Shape

{

public:

    float area()

    {

        return (width \* height);

    }

};

class Triangle : public Shape

{

public:

    float area()

    {

        return (width \* height / 2);

    }

};

int main()

{

    Rectangle rect;

    Triangle tri;

    rect.set\_data(5, 3);

    tri.set\_data(2, 5);

    cout << rect.area() << endl;

    cout << tri.area() << endl;

    return 0;

}

**Output :**

****

**Exercise 2**

Write a program with a mother class and an inherited daughter class. Both of them should have a method void display ()that prints a message (different for mother and daughter).In the main define a daughter and call the display() method on it.

[**Solution**](https://erlerobotics.gitbooks.io/erle-robotics-cpp-gitbook/content/code/8.OOP/e_8.2.cpp) **:**

#include <iostream>

using namespace std;

class mother

{

public:

    void display()

    {

        cout << "mother: display function\n";

    }

};

class daughter : public mother

{

public:

    void display()

    {

        cout << "daughter: display function\n\n";

    }

};

int main()

{

    daughter rita;

    rita.display();

    return 0;

}

**Output :** ****

**Exercise 3- Write a C++ Program to display Names, Roll No., and grades of 3 students who have appeared in the examination. Declare the class of name, Roll No. and grade. Create an array of class objects. Read and display the contents of the array.**

**Solution:**

##include <iostream>

using namespace std;

#define MAX 10

class student

{

private:

    char name[30];

    int rollNo;

    int total;

    float perc;

public:

    void getDetails(void); // member function to get student's details

    void putDetails(void); // member function to print student's details

};

void student::getDetails(void) // member function definition, outside of the class

{

    cout << "Enter name: ";

    cin >> name;

    cout << "Enter roll number: ";

    cin >> rollNo;

    cout << "Enter total marks outof 500: ";

    cin >> total;

    perc = (float)total / 500 \* 100;

}

void student::putDetails(void) // member function definition, outside of the class

{

    cout << "Student details:\n";

    cout << "Name:" << name << ",Roll Number:" << rollNo << ",Total:" << total << ",Percentage:" << perc;

}

int main()

{

    student std[MAX]; // array of objects creation

    int n, loop;

    cout << "Enter total number of students: ";

    cin >> n;

    for (loop = 0; loop < n; loop++)

    {

        cout << "Enter details of student " << loop + 1 << ":\n";

        std[loop].getDetails();

    }

    cout << endl;

    for (loop = 0; loop < n; loop++)

    {

        cout << "Details of student " << (loop + 1) << ":\n";

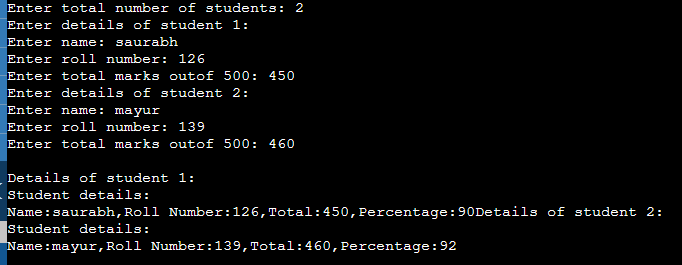
        std[loop].putDetails();

    }

    return 0;

}

**Output**

**Exercise -4: Write a C++ program to declare Struct. Initialize and display contents of member variables.**

#include <iostream>

using namespace std;

struct student

{

    char name[50];

    int roll;

    float marks;

};

int main()

{

    student s;

    cout << "Enter information," << endl;

    cout << "Enter name: ";

    cin >> s.name;

    cout << "Enter roll number: ";

    cin >> s.roll;

    cout << "Enter marks: ";

    cin >> s.marks;

    cout << "\nDisplaying Information," << endl;

    cout << "Name: " << s.name << endl;

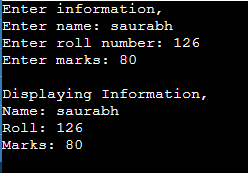
    cout << "Roll: " << s.roll << endl;

    cout << "Marks: " << s.marks << endl;

    return 0;

}

**Output:**



**Exercise -5: Write a C++ program to declare a class. Declare pointer to class. Initialize and display the contents of the class member.**

**Solution:**

#include <iostream>

using namespace std;

class Box

{

public:

    Box(double l = 2.0, double b = 2.0, double h = 2.0)

    {

        cout << "Constructor called." << endl;

        length = l;

        breadth = b;

        height = h;

    }

    double Volume()

    {

        return length \* breadth \* height;

    }

private:

    double length;

    double breadth;

    double height;

};

int main(void)

{

    Box Box1(3.3, 1.2, 1.5);

    Box Box2(8.5, 6.0, 2.0);

    Box \*ptrBox;

    ptrBox = &Box1;

    cout << "Volume of Box1: " << ptrBox->Volume() << endl;

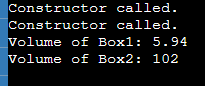
    ptrBox = &Box2;

    cout << "Volume of Box2: " << ptrBox->Volume() << endl;

    return 0;

}

**Output:**



**Exercise -6: write a program C++ Program to demonstrate Constructor Overloading.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

class Deposit

{

    long int principal;

    int time;

    float rate;

    float totalamount;

public:

    Deposit();                       // #1

    Deposit(long p, int t, float r); // #2

    Deposit(long p, int t);          // #3

    Deposit(long p, float r);        // #4

    void calculateamount(void);

    void display(void);

};

Deposit::Deposit()

{

    principal = time = rate = 0.0;

}

Deposit::Deposit(long p, int t, float r)

{

    principal = p;

    time = t;

    rate = r;

}

Deposit::Deposit(long p, int t)

{

    principal = p;

    time = t;

    rate = 0.08;

}

Deposit::Deposit(long p, float r)

{

    principal = p;

    time = 2;

    rate = r;

}

void Deposit::calculateamount(void)

{

    totalamount = principal + (principal \* time \* rate) / 100;

}

void Deposit::display(void)

{

    cout << "\nPrincipal Amount :: Rs." << principal << "\n";

    cout << "\nNo. of Years :: " << time << " years\n";

    cout << "\nRate of interest :: " << rate << "\n";

    cout << "\nTotal Amount :: Rs." << totalamount << "\n";

}

int main()

{

    Deposit d1;

    Deposit d2(2000, 2, 0.07f);

    Deposit d3(4000, 1);

    Deposit d4(3000, 0.12f);

    d1.calculateamount();

    d2.calculateamount();

    d3.calculateamount();

    d4.calculateamount();

    cout << "Object D1 Details ::-------------- \n";

    d1.display();

    cout << "\nObject D2 Details ::--------------\n";

    d2.display();

    cout << "\nObject D3 Details ::----------------\n";

    d3.display();

    cout << "\nObject D4 Details ::----------------\n";

    d4.display();

    return 0;

}

**OUTPUT : :**

Object D1 Details ::--------------

Principal Amount :: Rs.0

No. of Years :: 0 years

Rate of interest :: 0

Total Amount :: Rs.0

Object D2 Details ::--------------

Principal Amount :: Rs.2000

No. of Years :: 2 years

Rate of interest :: 0.07

Total Amount :: Rs.2002.8

Object D3 Details ::----------------

Principal Amount :: Rs.4000

No. of Years :: 1 years

Rate of interest :: 0.08

Total Amount :: Rs.4003.2

Object D4 Details ::----------------

Principal Amount :: Rs.3000

No. of Years :: 2 years

Rate of interest :: 0.12

Total Amount :: Rs.3007.2

**Exercise – 6: Write a program to find factorial of given number.**

#include <iostream>

using namespace std;

int main() {

int n;

long factorial = 1.0;

cout << "Enter a positive integer: ";

cin >> n;

if (n < 0)

cout << "Error! Factorial of a negative number doesn't exist.";

else {

for(int i = 1; i <= n; ++i) {

factorial \*= i;

}

cout << "Factorial of " << n << " = " << factorial;

}

return 0;

}

**Exercise – 7: Write a C++ program to use pointer for both base and derived classes and call the member function. Use Virtual keyword.**

#include <iostream>

using namespace std;

class base

{

public:

    virtual void print()

    {

        cout << "print base class" << endl;

    }

    void show()

    {

        cout << "show base class" << endl;

    }

};

class derived : public base

{

public:

    void print()

    {

        cout << "print derived class" << endl;

    }

    void show()

    {

        cout << "show derived class" << endl;

    }

};

int main()

{

    base \*bptr;

    derived d;

    bptr = &d;

    bptr->print();

    bptr->show();

}

**Output:**



**Exercise – 8: write a program of Unary increment (++) and decrement (--) operator overloading program in C++**

// C++ program for unary increment (++) and

// decrement (--) operator overloading

**Solution:**

#include <iostream>

using namespace std;

class NUM

{

private:

    int n;

public:

    // function to get number

    void getNum(int x)

    {

        n = x;

    }

    // function to display number

    void dispNum(void)

    {

        cout << "value of n is: " << n;

    }

    // unary ++ operator overloading

    void operator++(void)

    {

        n = ++n;

    }

    // unary -- operator overloading

    void operator--(void)

    {

        n = --n;

    }

};

int main()

{

    NUM num;

    num.getNum(10);

    ++num;

    cout << "After increment - ";

    num.dispNum();

    cout << endl;

    --num;

    cout << "After decrement - ";

    num.dispNum();

    cout << endl;

    return 0;

}

**Output**



**Exercise -9: C++ program to add two objects using binary plus (+) operator overloading**

**Solution:**

This program will demonstrate example of Binary Plus Operator Overloading, in this program we will take two objects and adding them.

Adding two objects using binary plus (+) operator overloading program in C++

// C++ program to add two objects using binary

// plus (+) operator overloading

#include <iostream>

using namespace std;

class NUM

{

private:

    int n;

public:

    // function to get number

    void getNum(int x)

    {

        n = x;

    }

    // function to display number

    void dispNum(void)

    {

        cout << "Number is: " << n;

    }

    // add two objects - Binary Plus(+) Operator Overloading

    NUM operator+(NUM &obj)

    {

        NUM x; // create another object

        x.n = this->n + obj.n;

        return (x); // return object

    }

};

int main()

{

    NUM num1, num2, sum;

    num1.getNum(10);

    num2.getNum(20);

    // add two objects

    sum = num1 + num2;

    sum.dispNum();

    cout << endl;

    return 0;

}

**Output**



#### Exercise -10:

**10.1. Program to Print a Half-Pyramid Using \***

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**Source Code**

#include <iostream>

using namespace std;

int main()

{

    int rows;

    cout << "Enter number of rows: ";

    cin >> rows;

    for (int i = 1; i <= rows; ++i)

    {

        for (int j = 1; j <= i; ++j)

        {

            cout << "\* ";

        }

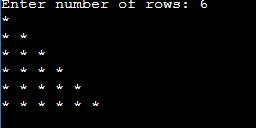
        cout << "\n";

    }

    return 0;

}

**Output :**



**10.2. Programs to Print an Inverted Half-Pyramid Using \* and Numbers**

**Inverted Half-Pyramid Using \***

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**Source Code**

#include <iostream>

using namespace std;

int main()

{

    int rows;

    cout << "Enter number of rows: ";

    cin >> rows;

    for (int i = rows; i >= 1; --i)

    {

        for (int j = 1; j <= i; ++j)

        {

            cout << "\* ";

        }

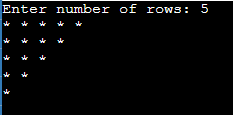
        cout << endl;

    }

    return 0;

}

**Output :**

****

**10.3. Program to Print a Full Pyramid Using \***

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

**Source Code**

#include <iostream>

using namespace std;

int main()

{

    int space, rows;

    cout << "Enter number of rows: ";

    cin >> rows;

    for (int i = 1, k = 0; i <= rows; ++i, k = 0)

    {

        for (space = 1; space <= rows - i; ++space)

        {

            cout << "  ";

        }

        while (k != 2 \* i - 1)

        {

            cout << "\* ";

            ++k;

        }

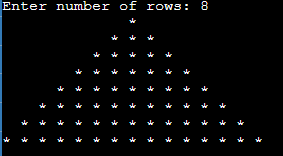
        cout << endl;

    }

    return 0;

}

**Output :**



**10.4. Inverted Full Pyramid Using \***

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \*

\* \* \*

\*

**Source Code**

#include <iostream>

using namespace std;

int main() {

    int rows;

    cout << "Enter number of rows: ";

    cin >> rows;

    for(int i = rows; i >= 1; --i) {

        for(int space = 0; space < rows-i; ++space)

            cout << "  ";

        for(int j = i; j <= 2\*i-1; ++j)

            cout << "\* ";

        for(int j = 0; j < i-1; ++j)

            cout << "\* ";

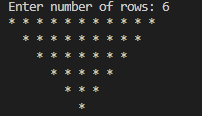
        cout << endl;

    }

    return 0;

}

**Output :**

****

**Exercise -11: Example: Simple Calculator using switch statement**

#include <iostream>

using namespace std;

int main()

{

    char op;

    float num1, num2;

    cout << "Enter operator: +, -, \*, /: ";

    cin >> op;

    cout << "Enter two operands: ";

    cin >> num1 >> num2;

    switch (op)

    {

    case '+':

        cout << num1 << " + " << num2 << " = " << num1 + num2;

        break;

    case '-':

        cout << num1 << " - " << num2 << " = " << num1 - num2;

        break;

    case '\*':

        cout << num1 << " \* " << num2 << " = " << num1 \* num2;

        break;

    case '/':

        cout << num1 << " / " << num2 << " = " << num1 / num2;

        break;

    default:

        // If the operator is other than +, -, \* or /, error message is shown

        cout << "Error! operator is not correct";

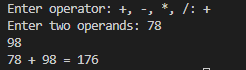
        break;

    }

    return 0;

}

**Output :**

****

**Exercise -12:  Fibonacci Series up to n number of terms**

#include <iostream>

using namespace std;

int main() {

    int n, t1 = 0, t2 = 1, nextTerm = 0;

    cout << "Enter the number of terms: ";

    cin >> n;

    cout << "Fibonacci Series: ";

    for (int i = 1; i <= n; ++i) {

        // Prints the first two terms.

        if(i == 1) {

            cout << t1 << ", ";

            continue;

        }

        if(i == 2) {

            cout << t2 << ", ";

            continue;

        }

        nextTerm = t1 + t2;

        t1 = t2;

        t2 = nextTerm;

        cout << nextTerm << ", ";

    }

    return 0;

}

**Output**

Enter the number of terms: 10

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

**Exercise -13: write a program to perform open, read, write and close a file in single program**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

    string filename = "example.txt";

    string line;

    // Open the file in write mode

    ofstream outfile;

    outfile.open(filename, ios::out);

    if (!outfile) {

        cout << "Error opening file for writing!" << endl;

        return 1;

    }

    // Write some data to the file

    outfile << "Hello, world!" << endl;

    outfile << "This is an example of file input/output in C++." << endl;

    // Close the file

    outfile.close();

    // Open the file in read mode

    ifstream infile;

    infile.open(filename, ios::in);

    if (!infile) {

        cout << "Error opening file for reading!" << endl;

        return 1;

    }

    // Read the contents of the file and print to console

    while (getline(infile, line)) {

        cout << line << endl;

    }

    // Close the file

    infile.close();

    return 0;

}