//\* Polymorphism

#include <iostream>

using namespace std;

//! Function overloading - Compile-time Polymorphism

class variable

{

public:

    void type()

    {

        cout << "NONE" << endl;

    }

    void type(int x)

    {

        cout << "INT" << endl;

    }

    void type(float x)

    {

        cout << "FLOAT" << endl;

    }

    void type(double x)

    {

        cout << "DOUBLE" << endl;

    }

    void type(string x)

    {

        cout << "STRING" << endl;

    }

};

//! Operator overloading - Compile-time  Polymorphism

class vector2d

{

    int x, y;

public:

    vector2d(int x, int y)

    {

        this->x = x;

        this->y = y;

    }

    vector2d operator+(vector2d &obj)

    {

        return vector2d(x + obj.x, y + obj.y);

    }

    friend ostream &operator<<(ostream &stream, vector2d &obj);

};

ostream &operator<<(ostream &stream, vector2d &obj)

{

    stream << obj.x << " | " << obj.y;

    return stream;

}

//! Virtual functions - Runtime  Polymorphism - Dynamic binding : check : Chapter3\_5

int main()

{ // function overloading

    {

        variable test;

        test.type();

        test.type(1);

        test.type(1.1F);

        test.type(1.1);

        test.type("water");

    }

    // operator overloading

    {

        vector2d v1(2, 5), v2(3, 8);

        int x = 10, y = 20;

        //+ operator  is overloaded

        vector2d v3 = v1 + v2;

        int z = x + y;

        cout << "Vector addition " << v3 << "\nInteger addition " << z << endl;

    }

    // virtual

    return 0;

}

//\* Error Handling

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

    int x, y;

    cout << "ENTER THE DIVIDENT" << endl;

    cin >> x;

    cout << "ENTER THE DIVISOR" << endl;

    cin >> y;

    try

    {

        if (y == 0)

        {

            throw 0;

        }

        cout << "QUOTIENT : " << floor(x / y) << "  REMAINDER :" << x % y << endl;

    }

    catch (int e)

    {

        cout << "CAN'T DIVIDE BY  " << e << endl;

    }

    return 0;

}

//\* File Handling

#include <iostream>

//? Standard header file for file handling

#include <fstream>

using namespace std;

int main()

{

    fstream file;

    // WRITTING TO A FILE

    file.open("test.txt", ios::out);

    file << "HELLO";

    file.close();

    // READING FROM A FILE

    file.open("test.txt", ios::in);

    string text;

    file >> text;

    cout << text << endl;

    file.close();

    return 0;

}