

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра програмної інженерії та інтелектуальних технологій управління

Звіт з лабораторної роботи № 2
з дисципліни «Основи програмування (частина 2)»

Виконав:

ст. гр. КН-221в Є.Р. Шулюпов

Перевірив:

доцент. каф. ПІТУ А.А. Пашнєв

Харків 2022

ТЕМА: ВИКОРИСТАННЯ ПОЛІМОРФІЗМУ ТА ШАБЛОНІВ

1. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

1.1 Ієрархія класів

Реалізувати класи "Людина", "Громадянин", "Студент", "Співробітник". В кожному класі визначити віртуальну функцію виведення даних про об'єкт на екран. Створити масив указівників на різні об'єкти ієрархії. В циклі для кожного об'єкта вивести на екран рядок даних про нього.

1.2 Використання поліморфізму

Створити клас для розв'язання завдання 1.2 шостої лабораторної роботи попереднього семестру. Клас повинен містити принаймні дві функції-члена – функцію, яка повертає значення відповідно до індивідуального завдання, а також суто віртуальну функцію, яка викликається з попередньої та визначає ліву частину рівняння або функцію для дослідження (відповідно до завдання).

Клас розташувати в окремому заголовному файлі. У відповідному файлі реалізації здійснити визначення однієї з двох функцій класу.

В іншій одиниці трансляції створити похідний клас із визначенням конкретної функції, яка підлягає дослідженню. У функції `main()` створити об'єкт похідного класу та здійснити виконання індивідуального завдання.

Примітка: Для обчислення першої (другої) похідної слід додати окремі функції-члени базового класу.

1.3 Узагальнений клас для представлення двовимірного масиву

Переробити клас, створений у завданні 1.2 попередньої лабораторної роботи, на шаблон класу. Реалізувати зовнішню узагальнену функцію знаходження мінімального елемента масиву. В функції `main()` створити масиви цілих, дійсних і простих дробів (раніше створений клас). Для цих трьох масивів здійснити перевірку роботи функції знаходження мінімального значення серед елементів масиву,

здійснити тестування всіх можливостей класу з перехопленням можливих винятків, а також розв'язати індивідуальну задачу.

Примітка: для того, щоб можна було знаходити мінімальне значення у масиві дробів, у класі "Простий дріб" необхідно перевантажити операції порівняння.

2. ОСНОВНА ЧАСТИНА

2.1 Описання розробленого застосунку

Завдання 1.1

```
#pragma warning(disable:4996)
#include <iostream>
#include <cstring>

using namespace std;

class human {
private:
    char name[30];
    char surname[30];
    char gender;
    int age;
public:
    human(){
        strcpy(this->name, "");
        strcpy(this->surname, "");
        gender = ' ';
        age = 0;
    }
    human(const char* name, const char* surname, char gender, int age)
    {
        strcpy(this->name, name);
        strcpy(this->surname, surname);
        this->gender = gender;
        this->age = age;
    }
    char* getName()
    {
        return name;
    }
    char* getSurname()
    {
        return surname;
    }
    char getGender()
    {
        return gender;
    }
    int getAge() const
    {
        return age;
    }

    virtual void inform() {
```

```

        cout << "Ім'я: " << name << ".\t Прізвище: " << surname << ".\t Вік: " << age << ".\t
Гендер: " << gender;
    }

    virtual ~human() { }
};

class citizen : public human {
private:
    char citizenship[30];
public:
    citizen(const char* name, const char* surname, char gender, int age, const char* citizenship)
        : human(name, surname, gender, age) {
        strcpy(this->citizenship, citizenship);
    }

    char* getCitizenship() {
        return citizenship;
    }

    void inform() override {
        human::inform();
        cout << "\t Громадянство: " << citizenship;
    }
};

class student : public citizen {
private:
    char faculty[30];
    int course;
public:
    student(const char* name, const char* surname, char gender, int age, const char* citizenship,
const char* faculty, int course)
        : citizen(name, surname, gender, age, citizenship) {
        strcpy(this->faculty, faculty);
        this->course = course;
    }

    char* getFaculty() {
        return faculty;
    }

    int getCourse() const {
        return course;
    }

    void inform() override {
        citizen::inform();
        cout << "\t Факультет: " << faculty << ".\t Курс:" << course;
    }
};

class employee : public citizen {
private:
    char job[30];
public:
    employee(const char* name, const char* surname, char gender, int age, const char*
citizenship, const char* job)
        : citizen(name, surname, gender, age, citizenship) {
        strcpy(this->job, job);
    }

    char* getJob() {
        return job;
    }
}

```

```

    void inform() override {
        citizen::inform();
        cout << "\t Місто праці:" << job;
    }
};

int main()
{
    setlocale(LC_ALL, "UKRAINIAN");

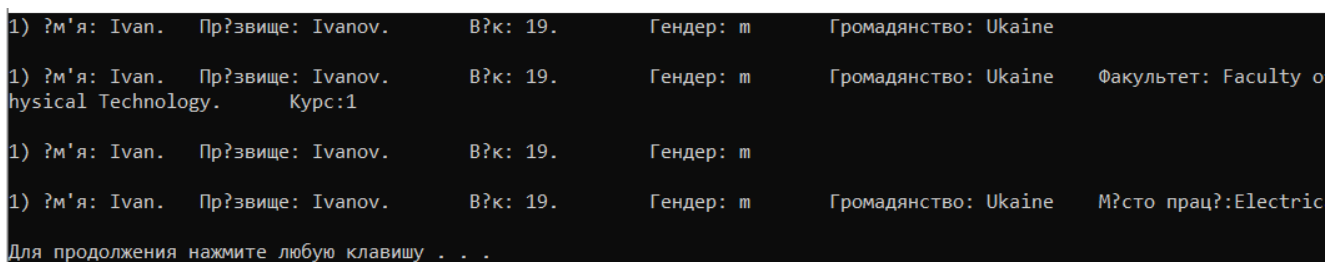
    const int n = 4;
    human *arr[n] = {
        new citizen ("Ivan", "Ivanov", 'm', 19, "Ukraine"),
        new student("Ivan", "Ivanov", 'm', 19, "Ukraine", "Faculty of Physical Technology", 1),
        new human("Ivan", "Ivanov", 'm', 19),
        new employee("Ivan", "Ivanov", 'm', 19, "Ukraine", "Electrician"),
    };

    for (int i = 0; i < n; i++)
    {
        cout << "1) ";
        arr[i]->inform();
        cout << "\n\n";
    }
    for (int i = 0; i < n; i++)
    {
        delete arr[i];
    }

    system("pause");
    return 0;
}

```

Приклад працездатності програми до завдання 1.1 (Рисунок 2.2.1)



```

1) Ім'я: Ivan. Прізвище: Ivanov. Вік: 19. Гендер: m Громадянство: Ukraine
1) Ім'я: Ivan. Прізвище: Ivanov. Вік: 19. Гендер: m Громадянство: Ukraine Факультет: Faculty of Physical Technology. Курс:1
1) Ім'я: Ivan. Прізвище: Ivanov. Вік: 19. Гендер: m
1) Ім'я: Ivan. Прізвище: Ivanov. Вік: 19. Гендер: m Громадянство: Ukraine Місто праці: Electrician
Для продовження натисніть будь-яку клавішу . . .

```

Рисунок 2.2.1 – Приклад працездатності програми до першого завдання

Завдання 1.2

main.cpp:

```

#include <iostream>
#include "task.h"

```

```

using namespace std;

class taskFunc : public tasks{
public:
    virtual double mathFunc(double x_1) override{
        return x_1 * x_1 - 2 * x_1 + 4;
    }
};

int main()
{
    setlocale(LC_ALL, "UKRAINIAN");

    taskFunc a;
    int q = -111, w = 11, e = 5;
    cout << "x_1 = " << q << "\t" << "x_2 = " << w << "\t" << "step = " << e << "\n";
    cout << "result = " << a.var27(q, w, e) << "\n";
    system("pause");
    return 0;
}

```

task.h

```

#include <iostream>
#include "task.h"
using namespace std;

class taskFunc : public tasks{
public:
    virtual double mathFunc(double x_1) override{
        return x_1 * x_1 - 2 * x_1 + 4;
    }
};

int main()
{
    setlocale(LC_ALL, "UKRAINIAN");

    taskFunc a;
    int q = -111, w = 11, e = 5;
    cout << "x_1 = " << q << "\t" << "x_2 = " << w << "\t" << "step = " << e << "\n";
    cout << "result = " << a.var27(q, w, e) << "\n";
    system("pause");
    return 0;
}

```

task.cpp

```

#include <cmath>
#include "task.h"

double tasks::var27(double x_1, double x_2, double step) {
    double max = x_1;
    for (x_1; x_1 <= x_2; x_1 += step)
    {
        if (Proizv(x_1) == 0) {

```

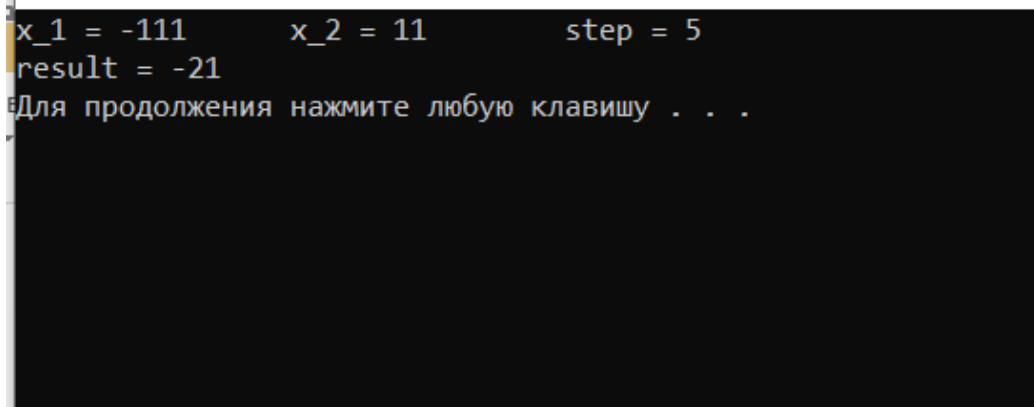
```

        if (x_1 > max) {
            max = x_1;
        }
    }
    return max;
}

// производная второго порядка (формула)
double tasks::Proizv(double x_1) {
    const double eps = 0.0000001;
    double proizv;
    proizv = ((mathFunc(x_1 + eps)) - 2 * mathFunc(x_1) + (mathFunc(x_1 - eps))) / (eps *
eps);
    return proizv;
}

```

Приклад працездатності програми до завдання 1.2 (Рисунок 2.2.2)



```

x_1 = -111      x_2 = 11      step = 5
result = -21
Для продовження натисніть будь-яку клавішу . . .

```

Рисунок 2.2.2 –
Приклад
працездатності
програми до
другого завдання

Завдання 1.3

main.cpp

```

#include <iostream>
#include "matrix.h"
#include "fraction.h"
using namespace std;

int main() {
    setlocale(LC_ALL, "Ukrainian");
    try {
        int m, n;
        cout << "введіть розміри матриці m, n: \n";
        cin >> m >> n;
        matrix<int> intMatrix(m,n);
        cout << "введіть int матрицю: \n";
        cin >> intMatrix;
        cout << "матриця int \n" << intMatrix;
        cout << "індивідуальне завдання: \n" << individualTask(intMatrix) << "\n";
        cout << "мінімальне значення: \n" << findMin(intMatrix) << "\n";
    }
}

```

```

    matrix<double> doubleMatrix(m, n);
    cout << "введіть double матрицю\n";
    cin >> doubleMatrix;
    cout << "матриця double \n" << doubleMatrix;
    cout << "мінімальне значення: \n" << findMin(doubleMatrix) << "\n";

    matrix<fraction> fractionMatrix(m, n);
    cout << "введіть fraction матрицю\n";
    cin >> fractionMatrix;
    cout << "матриця fraction \n" << fractionMatrix;
    cout << "мінімальне значення: \n" << findMin(fractionMatrix) << "\n";

}
catch (const char* ex) {
    cout << ex;
}
catch (Exception::badData& ex) {
    cout << "Incorrect data!\n";
}
catch (Exception::impossibleAction& ex) {
    cout << "Imposibile math action!\n";
}
system("pause");
return 0;
}

```

matrix.h

```

#pragma once
#include <iostream>

using namespace std;

class Exception {
public:
    class badData //клас перевірки вводу
    {
        int bad_value;
    public:
        badData(int value) : bad_value(value) {}
        badData() {}
    };

    class impossibleAction //клас неможливих дій
    {
        int bad_data;
    public:
        impossibleAction(int value) : bad_data(value) {}
        impossibleAction() {}
    };
};

template <typename T>
class matrix
{

```



```

private:
    int Row, Col;
    T** Value = nullptr;

public:

    matrix() { }
    matrix(int, int);
    matrix(const matrix&);

    int GetRow();
    int GetCol();

    T& operator()(int, int);
    const T& operator()(int, int) const;

    T* operator[] (int index) {
        return this->Value[index];
    }

    ~matrix();

    friend istream& operator>>(istream& in, matrix& m)
    {
        for (int i = 0; i < m.Row; i++)
        {
            for (int j = 0; j < m.Col; j++)
            {
                in >> m.Value[i][j];
            }
        }
        return in;
    }

    friend ostream& operator<<(ostream& out, const matrix& m)
    {
        for (int i = 0; i < m.Row; i++)
        {
            for (int j = 0; j < m.Col; j++)
            {
                out << m.Value[i][j] << "\t";
            }
            out << "\n";
        }
        return out;
    };

    friend matrix operator+(matrix& m1, matrix& m2);
    friend matrix operator-(matrix& m1, matrix& m2);
    friend matrix operator*(matrix& m1, matrix& m2);
    friend matrix operator*(matrix& m1, int& number);

};

template <typename T>
matrix<T>::matrix(int row, int col)
{
    if (row < 1) {
        throw Exception::badData(row);
    }
    if (col < 1) {

```

```

        throw Exception::badData(col);
    }
    else {
        Row = row;
        Col = col;
        Value = new T * [row];
        for (int i = 0; i < row; i++) Value[i] = new T[col];
    }
}

```

```

template <typename T>
matrix<T>::matrix(const matrix& m) //копирующий конструктор
    :Row(m.Row), Col(m.Col)
{
    Value = new int* [Row];
    for (int i = 0; i < Row; i++) Value[i] = new int[Col];
    for (int i = 0; i < Row; i++)
    {
        for (int j = 0; j < Col; j++)
            Value[i][j] = m.Value[i][j];
    }
}

```

```

template <typename T> int matrix<T>::GetRow()
{
    return Row;
}

```

```

template <typename T> int matrix<T>::matrix::GetCol()
{
    return Col;
}

```

```

template <typename T>
matrix<T> operator+(matrix<T>& m1, matrix<T>& m2)
{
    if (m1.GetRow() != m2.GetRow() || m1.GetCol() != m2.GetCol()) {
        throw Exception::impossibleAction;
    }
    else {
        matrix<T> temp(m1.GetRow(), m1.GetCol());
        for (int i = 0; i < m1.GetRow(); i++)
        {
            for (int j = 0; j < m1.GetCol(); j++)
            {
                temp(i, j) = m1(i, j) + m2(i, j);
            }
        }
        return temp;
    }
}

```

```

template <typename T>
matrix<T> operator-(matrix<T>& m1, matrix<T>& m2)
{
    if (m1.GetRow() != m2.GetRow() || m1.GetCol() != m2.GetCol()) {
        throw Exception::impossibleAction;
    }
    else {
        matrix<T> temp(m1.GetRow(), m1.GetCol());
        for (int i = 0; i < m1.GetRow(); i++)

```

```

        {
            for (int j = 0; j < m1.GetCol(); j++)
            {
                temp(i, j) = m1(i, j) - m2(i, j);
            }
        }
        return temp;
    }
}

template <typename T>
matrix<T> operator*(matrix<T>& m1, matrix<T>& m2)
{
    if (m1.GetCol() != m2.GetRow()) {
        throw Exception::impossibleAction;
    }
    else {
        matrix<T> temp(m1.GetRow(), m2.GetCol());
        for (int i = 0; i < m1.GetRow(); i++)
        {
            for (int j = 0; j < m2.GetCol(); j++) {
                temp(i, j) = 0;
                for (int cell = 0; cell < m2.GetRow(); cell++)
                {
                    temp(i, j) += m1(i, cell) * m2(cell, j);
                }
            }
        }
        return temp;
    }
}

```

```

template <typename T>
matrix<T> operator*(matrix<T>& m1, int& number)
{
    matrix<T> temp(m1.GetRow(), m1.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
        for (int j = 0; j < m1.GetCol(); j++)
        {
            temp(i, j) = m1(i, j) * number;
        }
    }
    return temp;
}

```

```

template <typename T>
T& matrix<T>::operator()(int row, int col)
{
    return Value[row][col];
}

```

```

template <typename T>
const T& matrix<T>::operator()(int row, int col) const
{
    return Value[row][col];
}

```

```

template <typename T>
matrix<T>::~matrix()
{
    for (int i = 0; i < Row; i++)
        delete[] Value[i];
    delete[] Value;
}

```

```

}

template <typename T>
matrix<T> individualTask(matrix<T>& m1)
{
    matrix<T> temp(m1.GetRow(), m1.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
        for (int j = 0; j < m1.GetCol(); j++)
        {
            if ((m1(i, j) < 0) && (m1(i, j) % 2 != 0))
            {
                temp(i, j) = m1(i, j) * 3;
            }
            else {
                temp(i, j) = m1(i, j);
            }
        }
    }
    return(temp);
}

template <typename T>
T findMin(matrix<T>& m) {
    T min;
    min = m[0][0];
    for (int i = 0; i < m.GetRow(); i++)
    {
        for (int j = 0; j < m.GetCol(); j++)
        {
            if (m[i][j] < min) {
                min = m[i][j];
            }
        }
    }
    return min;
}

```

fraction.h

```

#pragma once
#include <iostream>

using namespace std;

class fraction {
private:
    int n;
    int d;
public:

    fraction(int n = 1, int d = 1) {
        this->n = n;
        this->d = d;
    }

    void simpl() {
        if (n != 0) {
            if (d != 0) {

```

```

int nEmpty = n;
int dEmpty = d;
int remainder = dEmpty % nEmpty;
while (remainder != 0) {
    dEmpty = nEmpty;
    nEmpty = remainder;
    remainder = dEmpty % nEmpty;
}
int gcd = nEmpty;
if (gcd != 0) {
    if (gcd < 0) {
        if (n < 0 && d < 0 || n > 0 && d < 0) {
            n /= gcd; d /= gcd;
        }
        else if (n < 0 && d > 0) {
            n /= -gcd; d /= -gcd;
        }
    }
    else {
        if (n < 0 && d > 0 || n > 0 && d > 0) {
            n /= gcd; d /= gcd;
        }
        else if (n > 0 && d < 0) {
            n /= -gcd; d /= -gcd;
        }
    }
}
}
}
}
}

```

```

friend fraction operator+ (fraction f1, fraction f2) {
    fraction temp;
    if ((f1.d == 0) || (f2.d == 0)) {
        temp.n = f1.n * f2.d + f1.d * f2.n;
        temp.d = 0;
    }
    else
    {
        temp.n = f1.n * f2.d + f1.d * f2.n;
        temp.d = f1.d * f2.d;
        temp.simpl();
    }
    return temp;
}

```

```

friend fraction operator- (fraction f1, fraction f2) {
    fraction temp;
    if ((f1.d == 0) || (f2.d == 0)) {
        temp.n = f1.n * f2.d + f1.d * f2.n;
        temp.d = 0;
    }
    else {
        temp.n = f1.n * f2.d - f1.d * f2.n;
        temp.d = f1.d * f2.d;
        temp.simpl();
    }
    return temp;
}

```

```

friend fraction operator* (fraction f1, fraction f2) {

```

```

    fraction temp;
    if ((f1.d == 0) || (f2.d == 0)) {
        temp.n = f1.n * f2.d + f1.d * f2.n;
        temp.d = 0;
    }
    else
    {
        temp.n = f1.n * f2.n;
        temp.d = f1.d * f2.d;
        temp.simpl();
    }
    return temp;
}

friend fraction operator/ (fraction f1, fraction f2)
{
    fraction temp;
    if ((f1.d == 0) || (f2.d == 0)) {
        temp.n = f1.n * f2.d + f1.d * f2.n;
        temp.d = 0;
    }
    else {
        temp.n = f1.n * f2.d;
        temp.d = f1.d * f2.n;
        temp.simpl();
    }
    return temp;
}

friend bool operator<(const fraction& frst, const fraction& scnd)
{
    if (frst.n / frst.d < scnd.n / scnd.d)
    {
        return true;
    }
    return false;
}

friend istream& operator>>(istream&, fraction& frctn);
friend ostream& operator<<(ostream& out, const fraction& frctn);

};

// input and output
ostream& operator<<(ostream& out, const fraction& frctn) {
    if (frctn.d == 0) {
        out << "zero division error";
    }
    else if (frctn.n == 0) {
        out << 0;
    }
    else if (frctn.d == 1) {
        out << frctn.n;
    }
    else {
        out << frctn.n << "/" << frctn.d;
    }
    return out;
}

istream& operator>>(istream& in, fraction& number)
{
    in >> number.n;
    cout << "/" << "\n";
}

```

```

in >> number.d;
cout << "\n";
return in;
}

```

Приклад працездатності програми до завдання 1.3 (Рисунки 2.2.3 – 2.2.4)

```

введ?ть розм?ри матриц? m, n:
2
2
введ?ть int матрицю:
1
2
3
4
матриця int
1      2
3      4
?див?дуальне завдання:
1      2
3      4

м?н?мальне значення:
1
введ?ть double матрицю
1.3
2.4
3.5
4.651
матриця double
1.3    2.4
3.5    4.651
м?н?мальне значення:
1.3

```

Рисунок 2.2.3 –
Приклад
працездатності
програми до
другого завдання(1)

```

м?н?мальне значення:
1.3
введ?ть fraction матрицю
1
/
100

3
/
5

4
/
6

2
/
1

матриця fraction
1/100  3/5
4/6    2
м?н?мальне значення:
1/100
Для продовження натисніть будь-яку клавішу . . .

```

Рисунок 2.2.4 –
Приклад
працездатності
програми до
другого завдання(2)

ВИСНОВКИ

В результаті виконання лабораторної роботи №2, були вивчені та закріплені на практиці знання про роботу з поліморфізмом та шаблонами у C++.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

- 1 Іванов Л. В. Основи програмування (частина 2) Лабораторна робота 2 – ВИКОРИСТАННЯ ПОЛІМОРФІЗМУ ТА ШАБЛОНІВ C++, Методичні вказівки; http://www.iwanoff.inf.ua/programming_2_ua/LabTraining02.html (дата звернення до джерела: 19.05.22)