# Міністерство Освіти і Науки України

# Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Програмної інженерії та інформаційних технологій управління»

**Лабораторна робота № 2**

Використання поліморфізму та шаблонів у C++

**Виконав:**

Студент групи КН-36А

Кулик В.В.

**Перевірив:** Смолін П.А.

Харків 2017

**Завдання**

**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](http://iwanoff.inf.ua/algorithmization/LabTraining06.html) 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 1.3 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.

**Хід роботи**

**1.1** Реалізував класи «Людина», «Громадянин», «Студент», «Співробітник».

*Код програми 1.1:*

#pragma warning(disable:4996)

#include <cstring>

#include <iostream>

using std::strcpy;

using std::cout;

using std::endl;

class Human

{

private:

char full\_name[30];

public:

Human(char \*full\_name) { strcpy(this->full\_name, full\_name); }

char\* getfull\_name() { return full\_name; }

virtual void show() { cout << endl << "ФИО: " << full\_name; }

virtual ~Human() { }

};

class National : public Human

{

private:

char national[20];

public:

National(char\* full\_name, char\* national)

: Human(full\_name) {

strcpy(this->national, national);

}

char\* getNational() { return national; }

virtual void show() {

Human::show();

cout << "\tГражданство:" << national;

}

};

class Student : public National

{

private:

char university[40];

public:

Student(char\* full\_name, char\* national, char\* university)

: National(full\_name, national) {

strcpy(this->university, university);

}

public:

char\* getStudent() { return university; }

virtual void show() {

National::show();

cout << "\tСтудент университета " << university;

}

};

class Employee : public National

{

private:

char company[30];

public:

Employee(char\* full\_name, char\* national, char\* company)

: National(full\_name, national) {

strcpy(this->company, company);

}

public:

char\* getCompany() { return company; }

virtual void show() {

National::show();

cout << "\tСотрудник " << company;

}

};

void main()

{

setlocale(LC\_ALL, "UKRAINIAN");

const int N = 3;

Human \*humans[N] = {

new National("Иван Посейдон", "Украина"),

new Student("Таня Горгона", "Россия", "МГУ"),

new Employee("Валентин Гермес", "Украина", "BMI"),

};

for (int i = 0; i < N; i++)

{

humans[i]->show();

}

for (int i = 0; i < N; i++)

{

delete humans[i];

}

}

**1.2** Використання поліморфізму.

*Код програми 1.2:*

**Source.cpp**

#include <iostream>

#include"Header1.h"

using namespace std;

int main()

{

Sin obj;

cout << "Enter value" << endl;

double x;

cin >> x;

cout << "Second der of sin(" << x << ") is:" << endl;

cout << obj.second\_der(x) << endl;

cout << "enter strt and end" << endl;

double y;

cin >> x >> y;

cout << "min root of second der at the (" << x << ";" << y << ") interval is" << endl;

cout << obj.search(x, y) << endl;

system("pause");

return 0;

}

**Header1.cpp**

#include"Header1.h"

double Sin::funk(double x)

{

return sin(x);

}

**Header.cpp**

#include"Header.h"

double ind::first\_der(double x, double eps)

{

return ((funk(x) - funk(x + eps)) / (eps));

}

double ind::second\_der(double x, double eps)

{

return ((first\_der(x, eps) - first\_der(x + eps, eps)) / eps);

}

double ind::search(double start, double end, double eps)

{

double i = start;

while (start <= end)

{

if (funk(i)\*funk(i + eps) < 0)

{

return i;

}

i += eps;

}

}

**Header1.h**

#include"Header.h";

#include<Math.h>

#ifndef second

#define second

class Sin : public ind

{

public:

double funk(double x);

};

#endif

**Header.h**

#ifndef first

#define first

class ind

{

public:

virtual double funk(double x) = 0;

double search(double start, double end, double eps = 0.0001);

double first\_der(double x, double eps = 0.0001);

double second\_der(double x, double eps = 0.0001);

};

#endif

**1.3** Узагальнений клас для представлення двовимірного масиву.

*Код програми 1.3:*

**HEADER**

#ifndef IntArray\_h

#define IntArray\_h

#include <iostream>

using std::cout;

using std::endl;

using std::cin;

using std::istream;

using std::ostream;

template <typename T> class IntArray

{

friend ostream& operator<<(ostream& out, const IntArray& a)

{

for (int i = 0; i < a.m; i++)

{

for (int j = 0; j < a.n; j++)

{

out << a.pa[i][j] << ' ';

}

cout << "\n";

}

return out;

}

friend istream& operator >> (istream& in, IntArray& a)

{

cout << "Enter " << a.n \* a.m << " numbers:" << endl;

for (int i = 0; i < a.m; i++)

{

for (int j = 0; j < a.n; j++)

{

in >> a.pa[i][j];

}

}

return in;

}

friend int operator\*(IntArray<T>& v1, IntArray<T>& v2)

{

if (v1.n == v2.m)

{

IntArray a = IntArray(v1.m, v2.n);

for (int i = 0; i < v1.m; i++)

{

for (int j = 0; j < v2.n; j++)

{

int sum = 0;

for (int k = 0; k < v1.n; k++)

{

sum += v1.pa[i][k] \* v2.pa[k][j];

}

a.pa[i][j] = sum;

}

}

cout << a;

return 0;

}

else

{

return 1;

}

}

friend int operator+(IntArray<T>& v1, IntArray<T>& v2)

{

if (v1.m == v2.m && v1.n == v2.n)

{

IntArray<T> a = IntArray<T>(v1.m, v1.n);

for (int i = 0; i < v1.m; i++)

{

for (int j = 0; j < v1.n; j++)

{

a.pa[i][j] = v1.pa[i][j] + v2.pa[i][j];

}

}

cout << a;

return 0;

}

else

{

return 1;

}

}

friend int operator-(IntArray<T>& v1, IntArray<T>& v2)

{

if (v1.m == v2.m && v1.n == v2.n)

{

IntArray<T> a = IntArray<T>(v1.m, v1.n);

for (int i = 0; i < v1.m; i++)

{

for (int j = 0; j < v1.n; j++)

{

a.pa[i][j] = v1.pa[i][j] - v2.pa[i][j];

}

}

cout << a;

return 0;

}

else

{

return 1;

}

}

private:

T \*\*pa;

int m;

int n;

public:

class OutOfBounds

{

int index;

public:

OutOfBounds(int i) : index(i) { }

int getIndex() const { return index; }

};

IntArray() { pa = 0; m = 0; n = 0; }

IntArray(int j, int k);

IntArray(IntArray& arr);

~IntArray() { if (pa) delete[] pa; };

T& operator()(int index1, int index2);

int elemget(int index1, int index2) { return pa[index1][index2]; }

int getSize() const { return n\*m; }

int getM() const { return m; }

int getN() const { return n; }

};

template <typename T> IntArray<T>::IntArray(int j, int k)

{

pa = new T\*[m = j];

for (int i = 0; i < m; i++)

{

pa[i] = new T[n = k];

}

}

template <typename T> IntArray<T>::IntArray(IntArray& arr)

{

m = arr.m;

n = arr.n;

pa = new int\*[m];

for (int i = 0; i < m; i++)

{

pa[i] = new int[n];

}

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

pa[i][j] = arr.pa[i][j];

}

}

}

template <typename T> T& IntArray<T>::operator()(int index1, int index2)

{

return pa[index1][index2];

}

template <typename T> T fun(IntArray<T>& f)

{

if (f.getM() == 4 || f.getN() == 3)

{

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 3; j++)

{

if (f(i, j) % 2 != 0)

{

f(i, j) = f(i, j) \* 2;

}

}

}

cout << f << endl;

return 0;

}

else

{

cout << "Error";

return 1;

}

}

template <typename T> T mini(IntArray<T>& f)

{

T min = f(0, 0);

for (int i = 0; i < f.getM(); i++)

for (int j = 0; j < f.getN(); j++)

if (min > f(i, j))

{

min = f(i, j);

}

return min;

}

#endif IntArray\_h

**HEADER1**

#ifndef Fraction\_h

#define Fraction\_h

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

using std::istream;

using std::ostream;

class Fraction {

friend istream& operator >> (istream& in, Fraction& fr) { return in >> fr.numerator >> fr.denominator; }

friend ostream& operator<<(ostream& out, const Fraction& fr) { return out << fr.numerator << "/" << fr.denominator; }

friend Fraction operator\*(Fraction fr1, Fraction fr2) { return Fraction(fr1.numerator \* fr2.numerator, fr1.denominator \* fr2.denominator); }

friend Fraction operator\* (Fraction fr, int val) { return Fraction(fr.numerator \* val, fr.denominator); }

friend Fraction operator/(Fraction fr1, Fraction fr2) { return Fraction(fr1.numerator \* fr2.denominator, fr1.denominator \* fr2.numerator); }

friend Fraction operator/ (Fraction fr, int val) { return Fraction(fr.numerator, fr.denominator \* val); }

friend Fraction operator+(Fraction fr1, Fraction fr2)

{

int new\_den = fr1.denominator\*fr2.denominator;

int new\_num = (new\_den / fr1.denominator\*fr1.numerator) + (new\_den / fr2.denominator\*fr2.numerator);

return Fraction(new\_num, new\_den);

}

friend Fraction operator-(Fraction fr1, Fraction fr2)

{

int new\_den = fr1.denominator\*fr2.denominator;

int new\_num = (new\_den / fr1.denominator\*fr1.numerator) - (new\_den / fr2.denominator\*fr2.numerator);

return Fraction(new\_num, new\_den);

}

friend bool operator>(Fraction fr1, Fraction fr2)

{

double fr1num = fr1.numerator;

double fr2num = fr2.numerator;

double fr1den = fr1.denominator;

double fr2den = fr2.denominator;

if ((fr1num / fr1den) > (fr2num / fr2den))

{

return true;

}

else

{

return false;

}

}

private:

int numerator;

int denominator;

public:

Fraction() { numerator = denominator = 0; }

Fraction(int numerator, int denominator);

void Socr(int a, int b);

double getX() { return numerator; }

void setX(int numerator) { this->numerator = numerator; }

double getY() { return denominator; }

void setY(int denominator) { this->denominator = denominator; }

};

Fraction::Fraction(int numerator, int denominator)

{

if (denominator == 0)

throw std::runtime\_error("zero division error");

Socr(numerator, denominator);

}

void Fraction::Socr(int a, int b)

{

int x = (a < b) ? a : b;

for (int i = 2; i <= x; i++)

if (a%i == 0 && b%i == 0)

{

a /= i;

b /= i;

}

numerator = a;

denominator = b;

}

#endif Fraction\_h

**MAIN**

#include <iostream>

#include "Header.h"

#include "Header1.h"

using std::cout;

using std::endl;

void main()

{

IntArray<int> a(4, 3);

IntArray<double> b(2, 3);

IntArray<Fraction> fractionArray(2, 2);

cin >> a;

cin >> b;

cin >> fractionArray;

cout << endl << a << endl;

cout << b << endl;

cout << fractionArray << endl;

cout << mini(a) << endl;

cout << mini(b) << endl;

cout << mini(fractionArray) << endl;

system("pause");

}

***Висновок:***

*У даній роботі я за допомогою мови програмування С++ та використовуючи попередній досвід я розробив різного роду програми, під час розробки яких я навчився використовувати поліморфізм та шаблони у C++.*