МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та   
інформаційні технології управління»

ЗВІТ

з лабораторної роботи № 3

з дисципліни

«ОСНОВИ ПРОГРАМУВАННЯ»

Виконала

Студентка групи КН-222В

Григор’єва Катерина Ігорівна

Перевірив

Асистент каф. ПІІТУ

Олексій КОНДРАТОВ

Харків 2023

**Тема:** Використання поліморфізму та шаблонів.

**Мета:** Ознайомитись з основними принципами використання поліморфізму та шаблонів в програмуванні, з'ясувати взаємозв'язок між ними, а також навчитись застосовувати їх для розробки більш гнучких та універсальних програм.

## **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 для зворотного виклику**

Створити клас для розв'язання завдання 1.2 [шостої лабораторної роботи](http://www.iwanoff.inf.ua/programming_1_ua/LabTraining06.html) попереднього семестру. Клас повинен містити принаймні два методи – 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 Використання шаблонів для зворотного виклику**

В окремому заголовному файлі створити шаблонну функцію для розв'язання завдання 1.2 [шостої лабораторної роботи](http://www.iwanoff.inf.ua/programming_1_ua/LabTraining06.html) попереднього семестру. Першим параметром функції повинен бути об'єкт шаблонного типу, до якого можна застосувати операцію "круглі дужки".

Роботу шаблону перевірити на двох функціях, які підлягають дослідженню. Одна з функцій повинна бути реалізована як функціональний об'єкт.

*Примітка*: Для обчислення першої (другої) похідної слід додати окремі шаблонні функції.

### **1.4 Узагальнений клас для представлення двовимірного масиву**

Переробити клас, створений у завданні 1.4 [попередньої лабораторної роботи](http://www.iwanoff.inf.ua/programming_2_ua/LabTraining02.html#Tasks), 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.

### **1.5 Бібліотека шаблонних функцій для роботи з масивом (додаткове завдання)**

Створити заголовний файл з функціями, які працюють з масивом довільного узагальненого типу. Слід реалізувати такі функції:

* обмін місцями елементів зі вказаними індексами;
* пошук елемента з певним значенням;
* обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом).

Здійснити демонстрацію роботи усіх функцій з використанням даних як мінімум трьох різних типів.

**2 Завдання**

Перше завдання

Код до першого завдання

#pragma warning(disable : 4996)

#include <cstring>

#include <iostream>

using std::strcpy;

using std::cout;

using std::endl;

class Human

{

private:

char name[30];

int age;

char occupation[10];

public:

Human(const char \*name, double age)

{

strcpy(this->name, name);

this->age = age;

strcpy(this->occupation, occupation);

}

char \*getName()

{

return name;

}

int getAge() const

{

return age;

}

char \*getOccupation()

{

return occupation;

}

virtual void show()

{

cout << endl

<< "Name: " << name << "\tAge: " << age;

}

};

class Citizen : public Human

{

private:

char country[30];

public:

Citizen(const char \*name, int age, char \*country)

: Human(name, age) { strcpy(this->country, country); }

char \*getOccupation()

{

return country;

}

void show()

{

Human::show();

cout << "\tCountry: " << country;

}

};

class Student : public Citizen

{

private:

char group[30];

public:

Student(const char \*name, int age, char \*country, char \*group)

: Citizen(name, age, country) { strcpy(this->group, group); }

char \*getGroup()

{

return group;

}

void show()

{

Human::show();

cout << "\tGroup: " << group;

}

};

class Employee : public Citizen

{

private:

char job[30];

public:

Employee(const char \*name, int age, char \*country, char \*job)

: Citizen(name, age, country) { strcpy(this->job, job); }

char \*getJob()

{

return job;

}

void show()

{

Human::show();

cout << "\tJob: " << job;

}

};

int main()

{

setlocale(LC\_ALL, "UKRAINIAN");

const int N = 4;

Human \*humans[N] = {

new Student("Іванів", 17, "Україна", "222В"),

new Citizen("Кучма", 18, "Україна"),

new Employee("Константинів", 18, "Україна", "Програміст"),

new Human("Григор'єва", 19),

};

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

{

humans[i]->show();

}

cout << endl;

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

{

delete humans[i];

}

return 0;

}

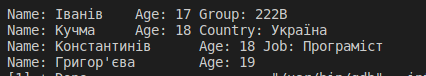


Рис 1 – Приклад роботи першої програми

Друге завдання

Код до другого завдання

task2.cpp

#include <iostream>

#include "lib.h"

#include <cmath>

using namespace std;

class SolveOurSquare : public Solvtask

{

public:

virtual double f(double x) override

{

return 2 \* x \* x + 5 \* x + 3;

}

};

void read(double &x1, double &x2, double &h)

{

cout << "-------ENTER THE VALUES-------" << endl;

cout << "Enter the beginning of interval x1 and the end of interval x2: ";

cin >> x1 >> x2;

cout << "Enter the step h: ";

cin >> h;

cout << endl;

}

int сheck(double &x1, double &x2, double &h)

{

if (x2 < x1)

{

return 1; // ERROR: x2 is less than x1.

}

else if (h > (x2 - x1))

{

return 2; // ERROR: x2-x1 is less than h.

}

}

int main()

{

double x1, x2, h;

read(x1, x2, h);

int count = сheck(x1, x2, h);

switch (count)

{

case 1:

cout << "ERROR: x2 is less than x1." << endl;

;

break;

case 2:

cout << "ERROR: x2 is less than x1." << endl;

break;

}

SolveOurSquare Solvtask;

Solvtask.func(x1, x2, h);

return 0;

}

lib.cpp

#include "lib.h"

#include <iostream>

#include <cmath>

using namespace std;

void Solvtask::func(double x1, double x2, double h)

{

int count = 0;

for (int x = x1; x <= x2; x += h)

{

double y;

y = sin(x);

if (y > -0.0001 && y < 0.0001)

{

count++;

};

}

cout << "Number of roots is " << count << endl;

}

lib.h

#ifndef Lib\_h

#define Lib\_h

class Solvtask

{

public:

void func(double x1, double x2, double h);

virtual double f(double x) = 0;

};

#endif

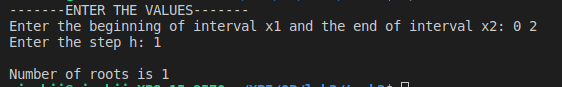


Рис 2 – Приклад роботи другої програми

Третє завдання

Код до третього завда

task3.cpp

#include <iostream>

#include "lib.h"

#include <cmath>

using namespace std;

double f1(double x)

{

return sin(x);

}

template <typename T>

class SolveOurSquare

{

private:

T a;

T b;

T c;

public:

SolveOurSquare(T a, T b, T c)

{

this->a = a;

this->b = b;

this->c = c;

}

T operator()(T x) const

{

return a \* x \* x + b \* x + c;

}

};

void read(double &x1, double &x2, double &h)

{

cout << "-------ENTER THE VALUES-------" << endl;

cout << "Enter the beginning of interval x1 and the end of interval x2: ";

cin >> x1 >> x2;

cout << "Enter the step h: ";

cin >> h;

cout << endl;

}

int сheck(double &x1, double &x2, double &h)

{

if (x2 < x1)

{

return 1; // ERROR: x2 is less than x1.

}

else if (h > (x2 - x1))

{

return 2; // ERROR: x2-x1 is less than h.

}

}

int main()

{

double x1, x2, h;

read(x1, x2, h);

int count = сheck(x1, x2, h);

switch (count)

{

case 1:

cout << "ERROR: x2 is less than x1." << endl;

;

break;

case 2:

cout << "ERROR: x2 is less than x1." << endl;

break;

}

SolveOurSquare<double> f2(2, 5, 3);

printf("\nOn range [%f; %f], with step %f, for function y = sin(x) number of roots is: %d\n", x1, x2, h, func(f1, x1, x2, h));

printf("\nOn range [%f; %f], with step %f, for function y = 2x^2 + 5x + 3 number of roots is: %d\n", x1, x2, h, func(f2, x1, x2, h));

return 0;

}

lib.h

#ifndef DICHOTOMY\_H

#define DICHOTOMY\_H

#include <cmath>

template <typename F>

int func(F f, double x1, double x2, double h)

{

int count = 0;

for (int x = x1; x <= x2; x += h)

{

double y;

y = f(x);

if (y > -0.0001 && y < 0.0001)

{

count++;

};

}

return count;

}

#endif

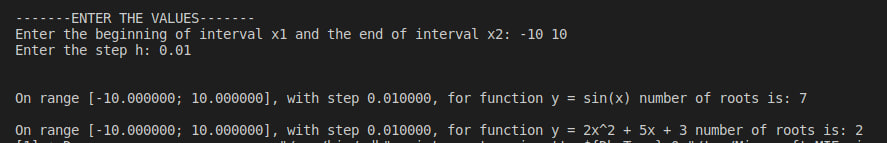


Рис 3 – Приклад роботи третьої програми

Четверте завдання

Код до четвертого завдання

task.cpp

#include <iostream>

#include <math.h>

#include "matrix.h"

#include "drib.h"

using std::cin;

using std::cout;

using std::endl;

int main()

{

cout << "Set Int UnsignedIntArray:" << endl;

UnsignedIntArray<int> a(2, 2);

cin >> a;

cout << "UnsignedIntArray a:" << a << endl;

cout << "Set Double UnsignedIntArray:" << endl;

UnsignedIntArray<double> b(2, 2);

cin >> b;

cout << "UnsignedIntArray b:" << b << endl;

cout << "Set Faction UnsignedIntArray:" << endl;

cout << "!!NOTE!! You need type factions like 1 2, that = 1/2, so for each number you need type 2 - upper and down number of faction" << endl;

UnsignedIntArray<Fraction> c(2, 2);

cin >> c;

cout << "UnsignedIntArray c:" << c << endl;

try

{

cout << "x3 Matrices:\n";

cout << "a:"

<< getx3<int>(a) << endl;

cout << "b:"

<< getx3<double>(b) << endl;

cout << "c:"

<< getx3<Fraction>(c) << endl;

cout << "Min values from Matrices:\n";

cout << "a: " << getMin<int>(a) << endl;

cout << "b: " << getMin<double>(b) << endl;

cout << "c: " << getMin<Fraction>(c) << endl;

}

catch (std::invalid\_argument &e)

{

cout << "Error! " << e.what() << endl;

}

catch (...)

{

cout << "Unknown error!" << endl;

}

return 0;

}

matrix.h

#ifndef MATRIX\_H

#define MATRIX\_H

#include <iostream>

using std::cout;

using std::endl;

using std::istream;

using std::ostream;

template <typename T>

class UnsignedIntArray

{

// Дружні функції перевантаження операцій виведення та введення:

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

{

out << "\nArray:" << endl;

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

{

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

{

out << a.pa[i][j] << "\t";

}

out << "\n";

}

return out;

}

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

{

cout << "Please enter the elements of the array, row by row:" << endl;

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

{

cout << "Row " << i + 1 << ": ";

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

{

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

cout << a.pa[i][j] << " ";

}

cout << endl;

}

return in;

}

private:

T \*\*pa; // вказівник на майбутній масив

int rows = 0; // поточний розмір масиву

int cols = 0; // розмір другого розміру масиву (кількість стовпців).

public:

// Вкладений клас для створення об'єкту-винятку

class OutOfBounds

{

int index; // індекс за межами діапазону

public:

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

int getIndex() const { return index; } // геттер для індексу

};

// Конструктори:

UnsignedIntArray() {}

UnsignedIntArray(int r, int c)

{

if (r <= 0)

{

throw OutOfBounds(r);

}

if (c <= 0)

{

throw OutOfBounds(c);

}

rows = r;

cols = c;

pa = new T \*[rows];

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

{

pa[i] = new T[cols];

}

}

UnsignedIntArray(const UnsignedIntArray &arr);

// Деструктор

~UnsignedIntArray()

{

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

{

delete[] pa[i];

}

delete[] pa;

}

T \*operator[](int index)

{

if (index < 0 || index >= rows)

{

throw OutOfBounds(index);

}

return pa[index];

}

// Геттери

int getRows() const { return rows; };

int getCols() const { return cols; };

};

template <typename T>

UnsignedIntArray<T>::UnsignedIntArray(const UnsignedIntArray &arr)

{

rows = arr.rows;

cols = arr.cols;

pa = new T \*[rows];

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

{

pa[i] = new T[cols];

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

{

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

}

}

}

template <typename T>

T getMin(UnsignedIntArray<T> &a)

{

UnsignedIntArray<T> res(a.getCols(), a.getRows());

T min = a[0][0];

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

{

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

{

if (a[i][j] < min)

{

min = a[i][j];

}

}

}

return min;

}

template <typename T>

UnsignedIntArray<T> getx3(UnsignedIntArray<T> a)

{

UnsignedIntArray<T> res(a.getCols(), a.getRows());

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

{

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

{

if (a[i][j] > 0)

{

res[i][j] = a[i][j] \* 3;

}

else

{

res[i][j] = a[i][j];

}

}

}

return res;

}

#endif

drib.h

#ifndef Drib\_h

#define Drib\_h

#include <iostream>

using std::cerr;

using std::cin;

using std::cout;

using std::endl;

using std::invalid\_argument;

using std::istream;

using std::ostream;

#include <cmath>

int findGCD(int a, int b)

{ // алгоритм Евкліда

if (b == 0)

{

return a;

}

return findGCD(b, a % b);

}

class Fraction

{

// Введення та виведення

friend istream &operator>>(istream &in, Fraction &fraction)

{

if (fraction.denominator == 0)

{

throw std::invalid\_argument("Denominator cannot be zero!");

}

return in >> fraction.numerator >> fraction.denominator;

};

friend ostream &operator<<(ostream &out, const Fraction &fraction)

{

if (fraction.numerator == 0)

{

out << 0;

}

else if (fraction.denominator == 1)

{

out << fraction.numerator;

}

else if (abs(fraction.numerator) > abs(fraction.denominator))

{

int integerPart;

if (fraction.numerator == 0)

{

integerPart = 0;

}

else

{

integerPart = fraction.numerator / fraction.denominator;

}

int newNumerator = fraction.numerator - integerPart \* fraction.denominator;

if (integerPart != 0)

{

out << integerPart << " ";

}

out << newNumerator << "/" << fraction.denominator;

}

else

{

out << fraction.numerator << "/" << fraction.denominator;

}

return out;

}

// Перевантажені оператори

friend bool operator>=(const Fraction &fraction1, const Fraction &fraction2)

{

return fraction1.numerator \* fraction2.denominator >= fraction2.numerator \* fraction1.denominator;

}

friend bool operator<=(const Fraction &fraction1, const Fraction &fraction2)

{

return (fraction1.numerator \* fraction2.denominator <= fraction2.numerator \* fraction1.denominator);

}

friend bool operator>(const Fraction &fraction1, const Fraction &fraction2)

{

return (fraction1.numerator \* fraction2.denominator > fraction2.numerator \* fraction1.denominator);

}

friend bool operator<(const Fraction &fraction1, const Fraction &fraction2)

{

return (fraction1.numerator \* fraction2.denominator < fraction2.numerator \* fraction1.denominator);

}

friend bool operator>=(const Fraction &fraction1, const int a)

{

Fraction fraction2 = Fraction(a, 1);

return (fraction1.numerator \* fraction2.denominator >= fraction2.numerator \* fraction1.denominator);

}

friend bool operator<=(const Fraction &fraction1, const int a)

{

Fraction fraction2 = Fraction(a, 1);

return (fraction1.numerator \* fraction2.denominator <= fraction2.numerator \* fraction1.denominator);

}

friend bool operator>(const Fraction &fraction1, const int a)

{

Fraction fraction2 = Fraction(a, 1);

return (fraction1.numerator \* fraction2.denominator > fraction2.numerator \* fraction1.denominator);

}

friend bool operator<(const Fraction &fraction1, const int a)

{

Fraction fraction2 = Fraction(a, 1);

return (fraction1.numerator \* fraction2.denominator < fraction2.numerator \* fraction1.denominator);

}

private:

int numerator; // чисельник

int denominator; // знаменник

public:

// Конструктори

Fraction() { numerator = denominator = 1; };

Fraction(int numerator, int denominator);

// Функція скорочення дробу

void simplify()

{

int gcd = findGCD(numerator, denominator);

if (gcd != 0)

{

numerator /= gcd;

denominator /= gcd;

}

}

// Геттери:

double getNumerator() { return numerator; }

double getDenominator() { return denominator; }

friend Fraction operator\*(Fraction fraction1, int a)

{

int mulNumerator = fraction1.numerator \* a;

int mulDenominator = fraction1.denominator \* 1;

Fraction mul(mulNumerator, mulDenominator);

mul.simplify();

return mul;

};

};

// Конструктор з параметрами

Fraction::Fraction(int numerator, int denominator)

{

this->numerator = numerator;

this->denominator = denominator;

}

#endif

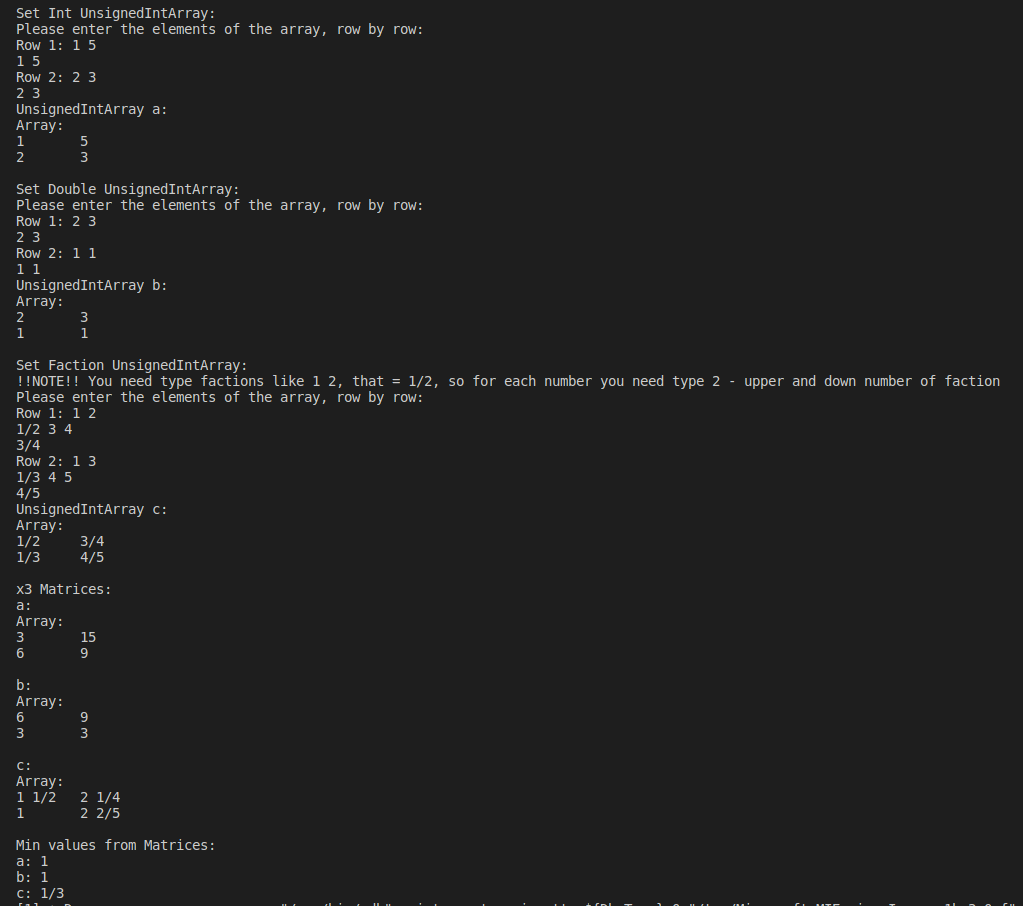


Рис 4 – Приклад роботи четвертої програми

П’яте завдання

Код до п’ятого завдання

task.cpp

#include <iostream>

#include <math.h>

#include "lib.h"

using std::cin;

using std::cout;

using std::endl;

using std::string;

template <typename T>

void cout\_for\_array(T \*arr, int size, string message)

{

cout << message << ": ";

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

{

cout << arr[i] << " ";

}

cout << endl;

}

int main()

{

const int size = 5;

int arr1[size] = {22, 11, 323, 3, 12132};

double arr2[size] = {0.21, 1.111, 2.0, 8.4, 0.5};

string arr3[size] = {"lal", "lol", "awe", "was", "waf"};

cout << "Our arrrays:\n";

cout\_for\_array<int>(arr1, size, "Int array: ");

cout\_for\_array<double>(arr2, size, "Double array: ");

cout\_for\_array<string>(arr3, size, "String array: ");

try

{

cout << "\nTask 1: swap parts with the specified indexes." << endl;

swap\_parts<int>(arr1, size, 1, 2);

swap\_parts<double>(arr2, size, 2, 3);

swap\_parts<string>(arr3, size, 0, 3);

cout << "\nTask 2: search for an element with a certain value" << endl;

cout << "323: " << find\_element<int>(arr1, 323, size) << "\n";

cout << "2.0: " << find\_element<double>(arr2, 2.0, size) << "\n";

cout << "was: " << find\_element<string>(arr3, "was", size) << "\n";

cout << "\nTask 3: swap all pairs of adjacent parts (with an even and odd index)." << endl;

swap\_couples<int>(arr1, size);

swap\_couples<double>(arr2, size);

swap\_couples<string>(arr3, size);

}

catch (int a)

{

if (a == -1)

{

cout << "ERROR: Element not found in the array" << endl;

}

else if (a == -2)

{

cout << "ERROR: To swap elements, the starting index must be smaller than the ending index" << endl;

};

}

cout << "\nOur arrrays after all the tasks:\n";

cout\_for\_array<int>(arr1, size, "Int array: ");

cout\_for\_array<double>(arr2, size, "Double array: ");

cout\_for\_array<string>(arr3, size, "String array: ");

return 0;

}

lib.h

#ifndef lib\_h

#define lib\_h

// заміна елемента

template <typename T>

void swap\_parts(T \*arr, int size, int start\_idx, int end\_idx)

{

if (start\_idx >= end\_idx)

{

throw -2;

}

int part\_size = (end\_idx - start\_idx + 1) / 2;

for (int i = 0; i < part\_size; i++)

{

T tmp = arr[start\_idx + i];

arr[start\_idx + i] = arr[end\_idx - i];

arr[end\_idx - i] = tmp;

}

}

// пошук елемента за значенням

template <typename T>

int find\_element(T \*arr, const T &value, int size)

{

int search = false;

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

{

if (arr[i] == value)

{

search = true;

return i;

}

}

if (!search)

{

throw -1;

}

}

// заміна парних з непарними

template <typename T>

void swap\_couples(T \*arr, int size)

{

for (int i = 1; i < size; i += 2)

{

T tmp = arr[i - 1];

arr[i - 1] = arr[i];

arr[i] = tmp;

}

}

#endif

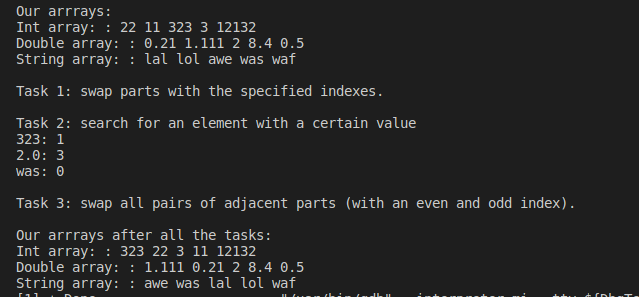


Рис 5 – Приклад роботи п’ятої програми

**Вправа для контролю номер 1**

Створити ієрархію класів "Будинок" – "Навчальний корпус". Створити масив указівників і вивести в циклі дані про об'єкти різних типів.

#include <iostream>

using namespace std;

class Building

{

public:

Building(int num, int floors) : number(num), numFloors(floors) {}

virtual void getInfo()

{

cout << "Building number " << number << " has " << numFloors << " floors." << endl;

}

protected:

int number;

int numFloors;

};

class EducationalBuilding : public Building

{

public:

EducationalBuilding(int num, int floors, int classrooms) : Building(num, floors), numClassrooms(classrooms) {}

void getInfo()

{

cout << "Educational building number " << number << " has " << numFloors << " floors and " << numClassrooms << " classrooms." << endl;

}

private:

int numClassrooms;

};

int main()

{

Building b1(101, 3);

Building b2(102, 4);

EducationalBuilding e1(201, 5, 20);

EducationalBuilding e2(202, 6, 25);

Building \*buildings[4] = {&b1, &b2, &e1, &e2};

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

{

buildings[i]->getInfo();

}

return 0;

}

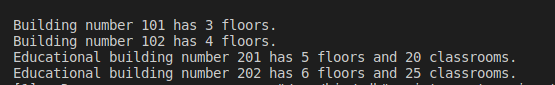


Рис 6 – Приклад роботи додаткової програми

**GitHub з завданнями:**

**Висновок:** Після успішного виконання лабораторної роботи з використанням поліморфізму та шаблонів в мові програмування С++, було отримано практичні навички створення складних ієрархій класів та використання їх для досягнення різноманітних цілей. Використання поліморфізму дозволило зменшити повторення коду та зробити програму більш гнучкою та легкою для змін. Шаблонні функції та класи дали можливість забезпечити універсальність та повторне використання коду, що зробило процес програмування ефективнішим та менш часозатратним. Загалом, робота дала важливий досвід роботи з класами та їх розширеними можливостями, який можна успішно застосовувати в різних програмних проектах.