Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

**про виконання лабораторних та практичних робіт блоку № 3**

На тему:  «Цикли. Вкладені Цикли. Завершення виконання циклів. Функції. Простір імен. Перевантаження функцій. Функції з змінною кількістю параметрів (еліпсис). Рекурсія. Вбудовані функції.»

***з дисципліни:*** «Мови та парадигми програмування»

до:

ВНС Лабораторної Роботи № 2

ВНС Лабораторної Роботи № 3

ВНС Лабораторної Роботи № 7

Практичних Робіт № 3

***Виконав:***

студент групи ШІ-14

Шевченко Данііл Вікторович

# **Тема роботи:**

# Цикли for(+foreach), while, do while, вкладенні цикли, функції та типи повернення функцій, функції типу void, перевантаження функцій, рекурсивні функції

# **Мета роботи:**

**Ознайомитись з різними типами циклів, такими як: while, do while, for(+foreach для масивів), опробувати застосування вкладених циклів. Ознайомитись з функціями, параметрами та аргументами функцій, типами повернення функцій, зокрема void, навчитись перевантажувати функції для різних типів та різної кількості параметрів.**

Завдання:

* Epic 3 Task 1 - Theory Education Activities
* Epic 3 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)
* Epic 3 Task 3 - Lab# programming: VNS Lab 3
* Epic 3 Task 4 - Lab# programming: VNS Lab 7
* Epic 3 Task 5 - Practice# programming: Class Practice Task

Epic 3 Task 6  - Practice# programming:  Self Practice Task

* Epic 3 Task 7 - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
* Epic 3 Task 8 - Results Evaluation and Release

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Цикли while, do while, for i foreach(C++ 14v)
* Тема №2: Функції та різновиди типів повернень функції
* Тема №3: Еліпсис(функції з нефіксованою кількістю параметрів)
* Тема №4: Рекурсія
* Тема №5: Перевантаження функцій
* Тема №6: Типи передачі аргументів у функції

1. Індивідуальний план опрацювання теорії:

* Тема №1: Цикли while, do while, for i foreach(C++ 14v)
  + Джерела Інформації
    - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Цикли while та do while і різниця між ними
    - Використання for як більш короткого аналога до while/do while
    - Модифікований цикл for(foreach) для роботи з масивами
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
  + Звершення опрацювання теми: 26.11.23
* Тема №2: Функції та різновиди типів повернень функції
  + Джерела Інформації
    - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Поняття функції та як їх використовувати
    - Різні типи повернень в функціях, зокрема void
    - Параметри та аргументи функцій
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
  + Звершення опрацювання теми: 26.11.23
* Тема №3: Еліпсис(функції з нефіксованою кількістю параметрів)
* Джерела Інформації
  + - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Ознайомлення з еліпсисом – об’єктом для визначення функцій з наперед невідомою кількістю параметрів
    - Використання макросів va\_list, va\_start і va\_end для доступу до еліпсису
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23
* Тема №4: Рекурсія
  + Джерела Інформації
    - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Рекурсивні функції та їх використання
    - Числа Фібоначчі
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23
* Тема №5: Перевантаження функцій
* Джерела Інформації
  + - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Перевантаження функцій для різної кількості параметрів
    - Перевантаження функцій для різних типів параметрів
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23

Тема №6: Типи передачі аргументів у функції

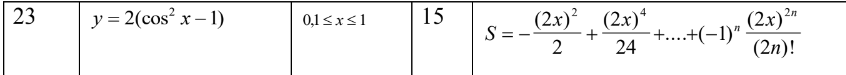
* + Джерела Інформації
    - Практика в НУ “ЛП”
    - Курси https://acode.com.ua/
  + Що опрацьовано:
    - Передача за значенням
    - Передача за посиланням
    - Передача за адресою
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.23
* Звершення опрацювання теми: 26.11.23

# **Виконання роботи:**

**VNS LAB3**

## **1. Опрацювання завдання та вимог до програм та середовища:**

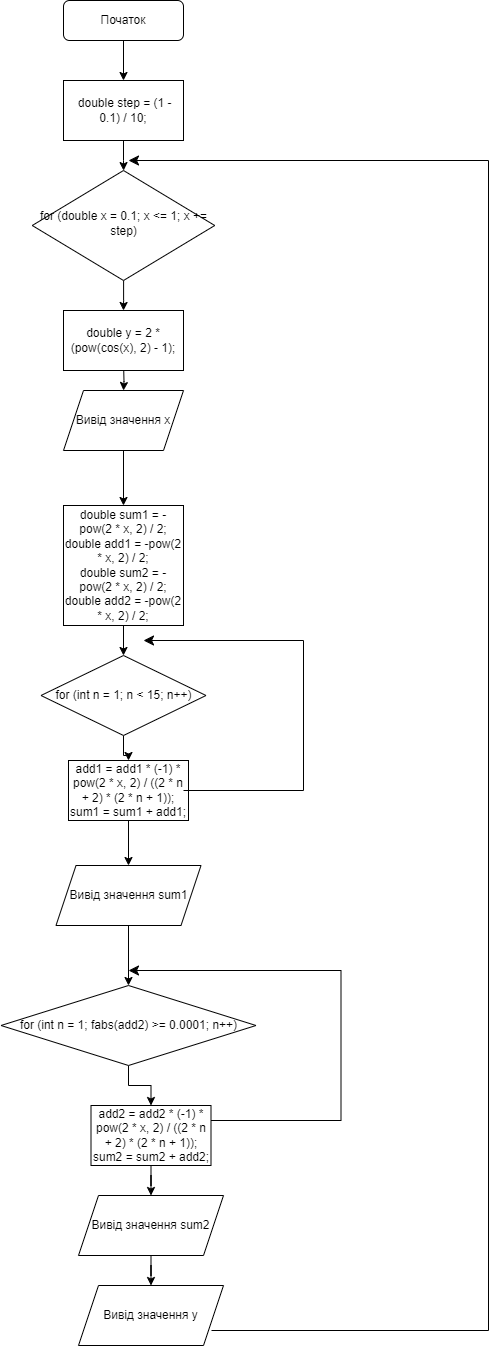
Завдання №1 VNS Lab3

* Варіант 23
* 
* Обчислення n+1 члена за рекурентною формулою:

)

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма VNS Lab3

* 
* Планований час на реалізацію – 30 хв

## **3. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <math.h>;

using namespace std;

int main()

{

double step = (1 - 0.1) / 10;

for (double x = 0.1; x <= 1; x += step)

{

double y = 2 \* (pow(cos(x), 2) - 1);

cout << "X=" << x << " ";

double sum1 = -pow(2 \* x, 2) / 2;

double add1 = -pow(2 \* x, 2) / 2;

double sum2 = -pow(2 \* x, 2) / 2;

double add2 = -pow(2 \* x, 2) / 2;

for (int n = 1; n < 15; n++)

{

add1 = add1 \* (-1) \* pow(2 \* x, 2) / ((2 \* n + 2) \* (2 \* n + 1));

sum1 = sum1 + add1;

}

cout << "SN=" << sum1 << " ";

for (int n = 1; fabs(add2) >= 0.0001; n++)

{

add2 = add2 \* (-1) \* pow(2 \* x, 2) / ((2 \* n + 2) \* (2 \* n + 1));

sum2 = sum2 + add2;

}

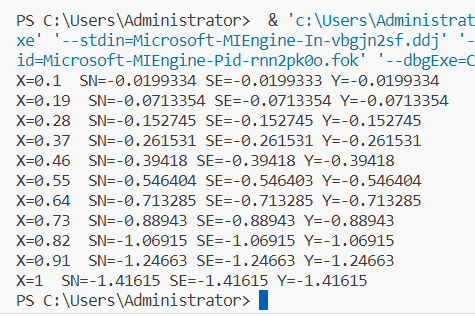
cout << "SE=" << sum2 << " ";

cout << "Y=" << y;

cout << endl;

}

## **4. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 50 хв

**VNS LAB7(Task1)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №2 VNS Lab7 Task1

* Варіант 23

Написати функцію (або макровизначення), що знаходить довжину сторони

за координатами його точок.. Написати функцію belong, що визначає чи

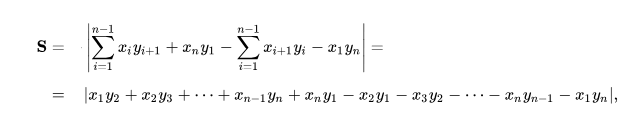
належить точка М з координатами (х,у) трикутнику, заданому координатами

вершин. Написати функцію c змінною кількістю параметрів, що визначає чи

належить точка М опуклому багатокутнику, заданому координатами своїх

вершин.

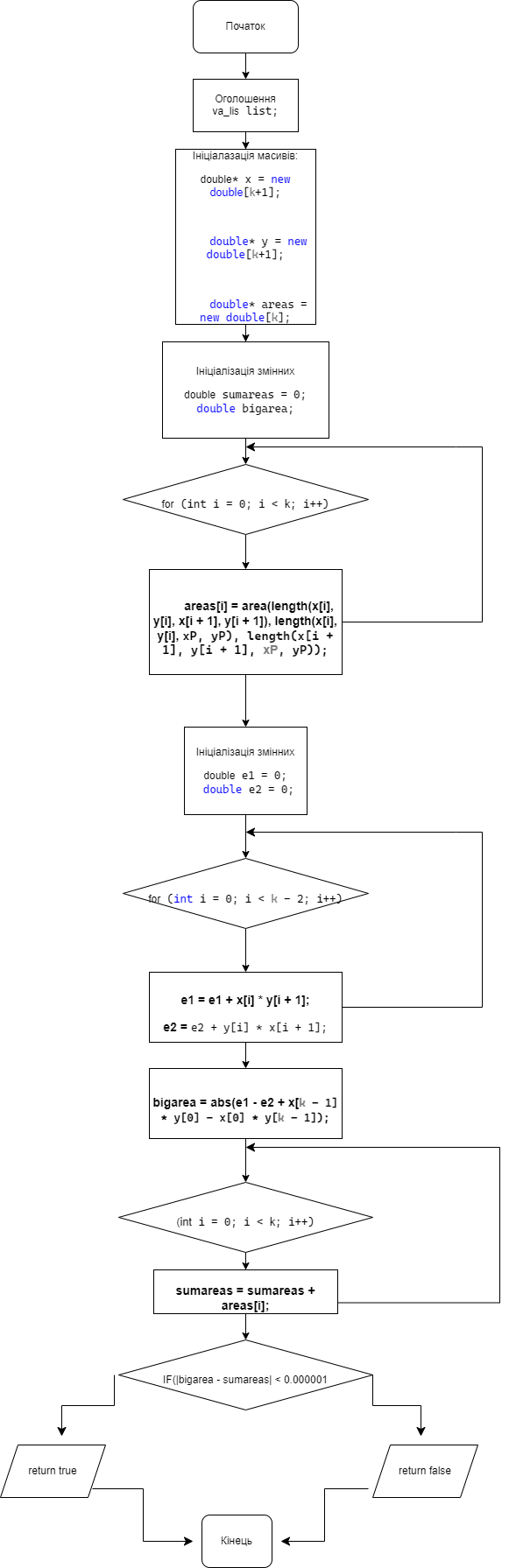
В завданні використана формула Герона та формула Гаусса для площі:



## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма VNS Lab7 Task1

Спочатку в програмі прописані функції для знаходження відстані між двома точками за їх координатами. Потім прописані функція для знаходження площі трикутника за трьома вершинами: спочатку за допомогою попердньої функції знаходяться сторони трикутника, а потім використовується формула Герона. Для визначення, чи лежить точка всередині многокутника, n-кутник розбивається на n трикутників. Послідовно шукається площа кожного трикутника і знаходиться сума їх площ. Далі за формулою Гаусса знаходиться площа многокутника по координатах його вершин. У випадку, якщо площа многокутника дорівнює сумі площ усіх трикутників, то точка лежить всередині многокутника. В іншому випадку – лежить поза многокутником.

* 
* Планований час на реалізацію – 90 хв

## **3. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <cstdarg>

#include <cmath>

using namespace std;

double length(double x1, double y1, double x2, double y2)

{

double l = sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2));

return l;

}

double area(double a, double b, double c)

{

double p = (a + b + c) / 2;

double ar = sqrt(p \* (p - a) \* (p - b) \* (p - c));

return ar;

}

bool belongtriangle(double xP, double yP, double xA, double yA, double xB, double yB, double xC, double yC)

{

double area1 = area(length(xA, yA, xB, yB), length(xA, yA, xP, yP), length(xB, yB, xP, yP));

double area2 = area(length(xB, yB, xC, yC), length(xB, yB, xP, yP), length(xC, yC, xP, yP));

double area3 = area(length(xC, yC, xA, yA), length(xC, yC, xP, yP), length(xA, yA, xP, yP));

double bigarea = area(length(xA, yA, xB, yB), length(xA, yA, xC, yC), length(xB, yB, xC, yC));

if (fabs(bigarea - (area1 + area2 + area3)) < 0.0001)

{

return true;

}

else

{

return false;

}

}

bool belongpolygon(double xP, double yP,int k ...)

{

va\_list list;

va\_start(list, k);

double\* x = new double[k+1];

double\* y = new double[k+1];

double\* areas = new double[k];

double sumareas = 0;

double bigarea;

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

{

x[i] = va\_arg(list, double);

y[i] = va\_arg(list, double);

}

x[k] = x[0];

y[k] = y[0];

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

{

areas[i] = area(length(x[i], y[i], x[i + 1], y[i + 1]), length(x[i], y[i], xP, yP), length(x[i + 1], y[i + 1], xP, yP));

}

double e1 = 0;

double e2 = 0;

for (int i = 0; i < k - 2; i++)

{

e1 = e1 + x[i] \* y[i + 1];

e2 = e2 + y[i] \* x[i + 1];

}

bigarea = abs(e1 - e2 + x[k - 1] \* y[0] - x[0] \* y[k - 1]);

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

{

sumareas = sumareas + areas[i];

}

cout << "The sum of areas of triangles is " << sumareas << endl;

cout << "The area of the polygon is " << bigarea << endl;;

if (abs(bigarea - sumareas) < 0.0000001)

{

return true;

}

else

{

return false;

}

}

int main()

{

bool belong = belongpolygon(11.0, 6.0, 4, 0.0, 0.0, 0.0, 13.0, 13.0, 13.0, 13.0, 0.0);

if (belong)

{

cout << "The point is in the polygon";

}

else

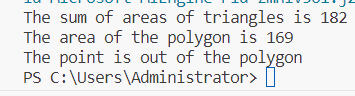
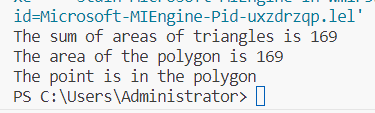
{

cout << "The point is out of the polygon";

}

}

## **4. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 180 хвилин

**VNS LAB7(Task2)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №3 VNS Lab7(Task2)

* Варіант 19

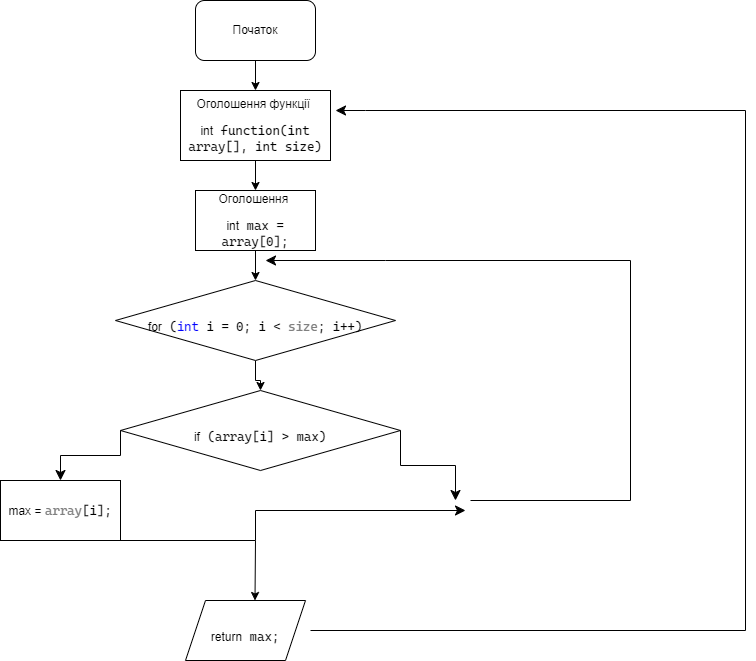
Написати перевантажені функції й основну програму, що їх викликає.

а) для масиву цілих чисел знаходить мінімальний елемент;

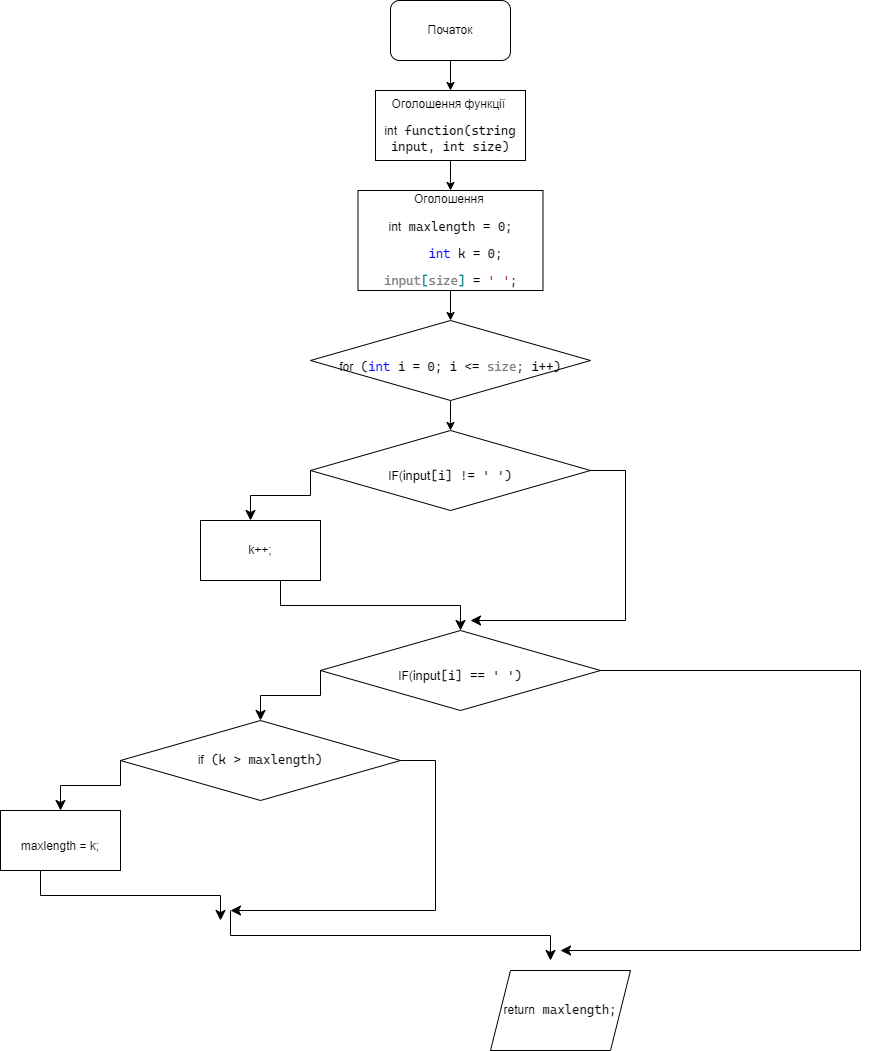
б) для рядка знаходить довжину найкоротшого слова .

**2. Дизайн та планована оцінка часу виконання завдань:**

**Для першої функції:**

****

**Для другої функції:**



Програма VNS Lab7(Task 2)

* Планований час на реалізацію – 1 година

## **3. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <string>

using namespace std;

int function(int array[], int size)

{

int max = array[0];

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

{

if (array[i] > max)

{

max = array[i];

}

}

return max;

}

int function(string input, int size)

{

int maxlength = 0;

int k = 0;

input[size] = ' ';

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

{

if (input[i] != ' ')

{

k++;

}

if (input[i] == ' ' )

{

if (k > maxlength)

{

maxlength = k;

}

k = 0;

}

}

return maxlength;

}

int main()

{

int N;

cout << "N=";

cin >> N;

int\* arr = new int[N];

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

{

cin >> arr[i];

}

cin.ignore(30000, '\n');

cout << "The max value of an array is " << function(arr, N) << endl;

string test;

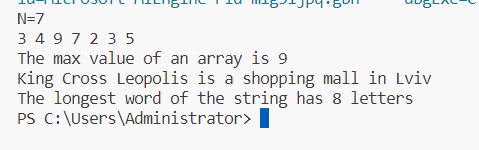
getline(cin, test);

int b = test.size();

cout << "The longest word of the string has " << function(test, b) << " letters";

}

## **4. Результати виконання завдань, тестування та фактично затрачений час:**



Результат виконання програми

Час затрачений на виконання завдання – 1 година

**VNS Lab7(extra task)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №4 Algotester Lab7(без перевантаження)

* Варіант 23
* а) для масиву, який починається на парне число виконує циклічний зсув вліво
* на кількість елементів, що дорівнює першому елементу масиву.
* б) для масиву, який починається на непарне число виконує циклічний зсув
* вправо на кількість елементів, що дорівнює останньому елементу масиву

## **2. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

using namespace std;

int\* arrmod(int array[],int size)

{

int\* modarray = new int[size];

if (array[0] % 2 == 0)

{

int step = array[0] % size;

for (int i = size - 1, j = 0; i >= 0; i--)

{

if (i >= step)

{

modarray[i - step] = array[i];

}

else

{

modarray[size - 1 - j] = array[i];

j++;

}

}

}

if (array[0] % 2 != 0)

{

int step = array[size - 1] % size;

for (int i = 0, j = 0; i <= size - 1; i++)

{

if ((size - i - 1) >= step)

{

modarray[i + step] = array[i];

}

else

{

modarray[j] = array[i];

j++;

}

}

}

return modarray;

}

int main()

{

int size;

cin >> size;

int\* array = new int[size];

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

{

cin >> array[i];

}

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

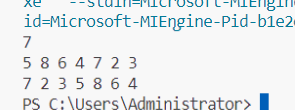
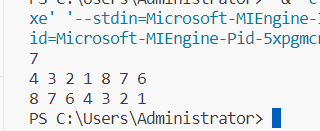
{

cout << arrmod(array,size)[i] << " ";

}

}

## **3 . Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 1.5 години

**Practice Task 3**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №5 Practice Task 3

* Менеджмент бібліотеки
* Ви створюєте просту програму керування бібліотекою. Книги в бібліотеці є, користувачі можуть їх взяти або повернути.
* Програма повинна вміти
* ⁃ Перерахувати всі книги.
* ⁃ Дозволити взяти книгу (за наявності).
* ⁃ Дозволити повернення книги.
* Структури даних
* ⁃ Використовуйте масив або вектор для зберігання назв книг.
* ⁃ Використовуйте інший масив або вектор для збереження стану доступності кожної книги.
* Вимоги
* ⁃ while: продовжувати працювати, доки користувач не вирішить вийти.
* ⁃ do while: Після кожної операції (позичити, повернути, перерахувати) запитуйте користувача, чи хоче він виконати іншу операцію. Якщо так, поверніться назад.
* ⁃ for: список усіх книг за допомогою циклу for.
* ⁃ для кожної: перевірити наявність кожної книги.
* ⁃ goto: якщо користувач вводить неправильний вибір, використовуйте goto, щоб перенаправити його до головного меню.

## **2. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

using namespace std;

int main()

{

    vector<string> books;

    books = { "50 shapes of grey", "Harry Potter", "Zakhar Berkut", "Kobzar", "Faust", "1984", "Vacation" };

    vector<string> available;

    available = books;

    int size = books.size();

    int choice = 0;

    while (true)

    {

    tryagain0:

        cout << "1 - list the books\n2 - borrow a book\n3 - return a book\n4 - check avaibility of a book\n5 - exit" << endl;

        cin >> choice;

        if (choice == 1)

        {

            cout << "\033[35m" << "All existing books:" << "\033[37m" << endl;

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

            {

                cout << books[i] << endl;

            }

            cout << endl;

            cout << "\033[35m" << "Available books:" << "\033[37m" << endl;

            for (string x : available)

            {

                if (x != "none")

                {

                    cout << "\033[32m" << x << "\033[37m" << endl;

                }

            }

            cout << endl;

        }

        if (choice == 2)

        {

            cin.ignore(30000, '\n');

            string strchoice;

            do

            {

                cout << "Enter the name of a book or type <<return>> to quit: ";

                getline(cin, strchoice);

                bool ok = false;

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

                {

                    if (strchoice == available[i])

                    {

                        available[i] = "none";

                        ok = true;

                    }

                }

                if (ok)

                {

                    cout << "\033[32m" << "You successfully borrowed a book" << "\033[37m" << endl;

                }

                if (!ok && strchoice != "return")

                {

                    cout << "\033[31m" << "There is no such book. Try again" << "\033[37m" << endl;

                }

            } while (strchoice != "return");

        }

        if (choice == 3)

        {

            cin.ignore(30000, '\n');

            string strchoice2;

            do

            {

                cout << "Enter the name of a book you want to return or type <<return>> to quit: ";

                getline(cin, strchoice2);

                bool ok2 = false;

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

                {

                    if (strchoice2 == books[i] && available[i] == "none")

                    {

                        available[i] = strchoice2;

                        ok2 = true;

                    }

                }

                if (ok2)

                {

                    cout << "\033[32m" << "You successfully returned the book" << "\033[37m" << endl;

                }

                if (!ok2 && strchoice2 != "return")

                {

                    cout << "\033[31m" << "The book doesn't belong to our library or is already available" << "\033[37m" << endl;

                }

            } while (strchoice2 != "return");

        }

        if (choice == 4)

        {

            cin.ignore(30000, '\n');

            string strchoice3;

            do

            {

                cout << "Enter the name of a book you want to check on avaibility: ";

                getline(cin, strchoice3);

                bool ok3 = false;

                for (string y : available)

                {

                    if (y == strchoice3)

                    {

                        ok3 = true;

                    }

                }

                if (ok3)

                {

                    cout << "\033[32m" << "The book is available" "\033[37m";

                }

                if (!ok3 && strchoice3!="return")

                {

                    cout << "\033[31m" <<  "The book isn't available" << "\033[37m";

                }

                cout << endl;

            } while (strchoice3 != "return");

        }

        if (choice == 5)

        {

            cout << "\033[36m" << "Powered by Daniil Shevchenko. Googbye" << "\033[37m";

            Sleep(3000);

            break;

        }

        if (cin.fail())

        {

            cin.clear();

            cin.ignore(30000, '\n');

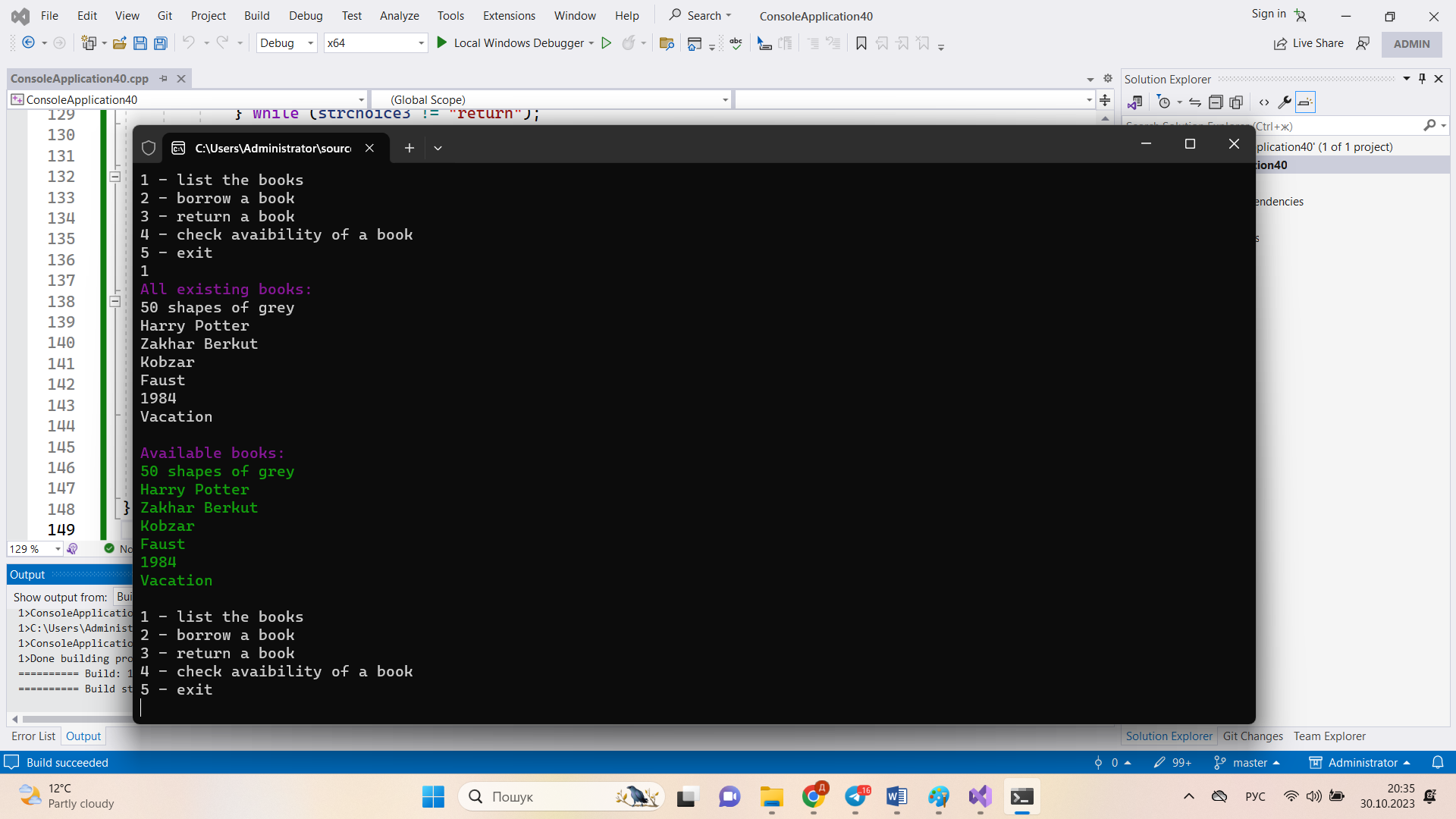
            goto tryagain0;

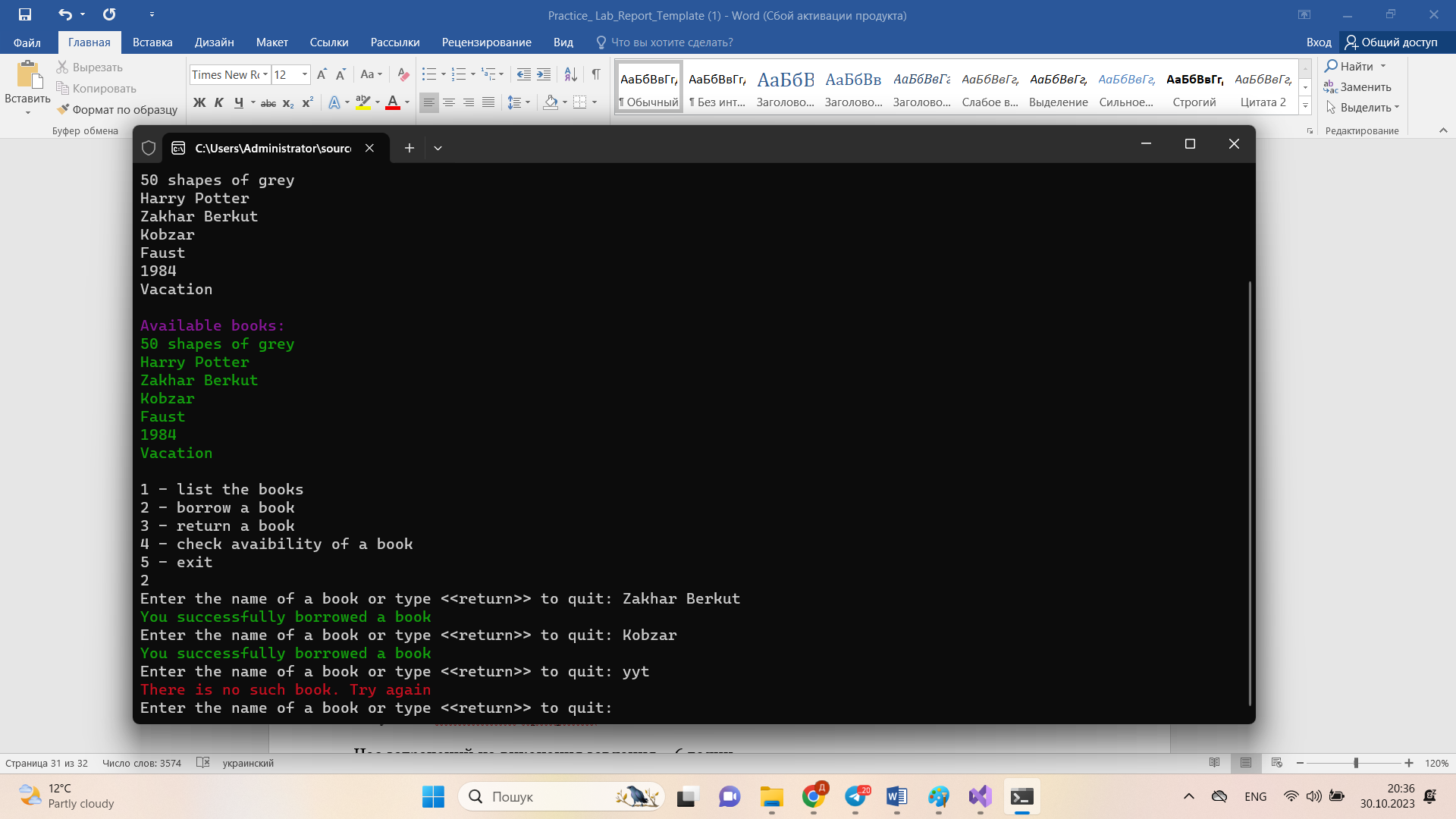
        }

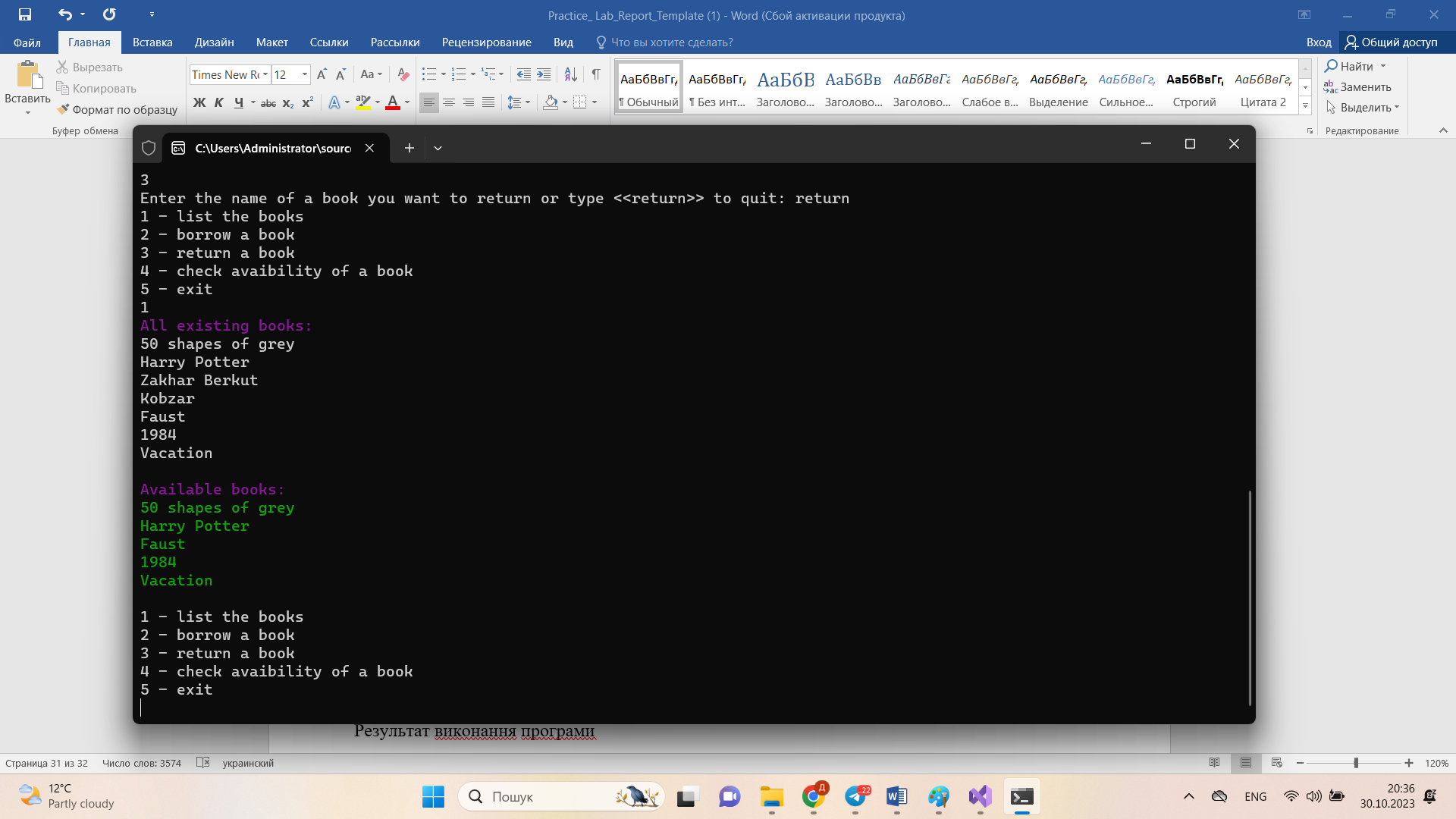
    }

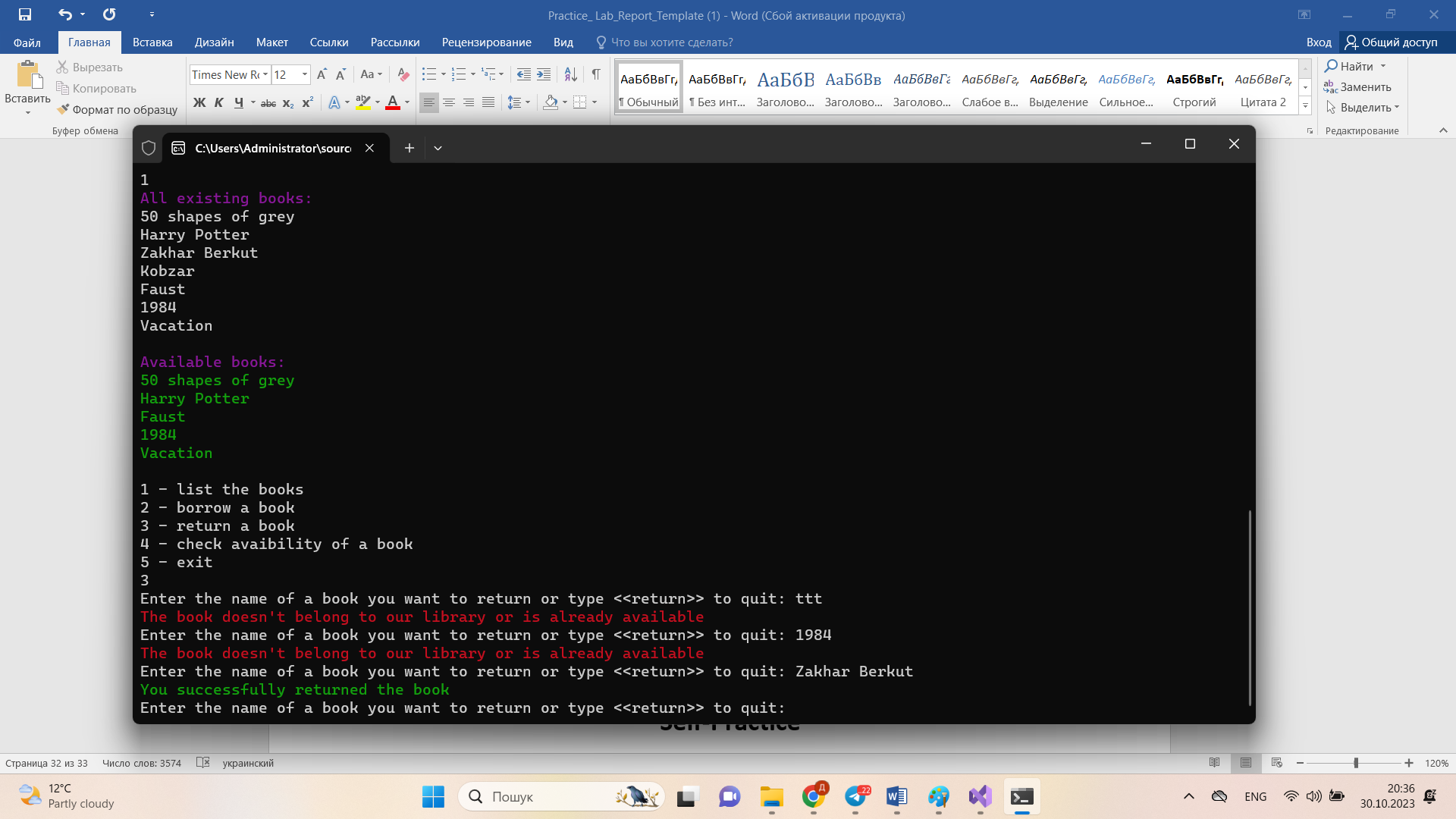
}

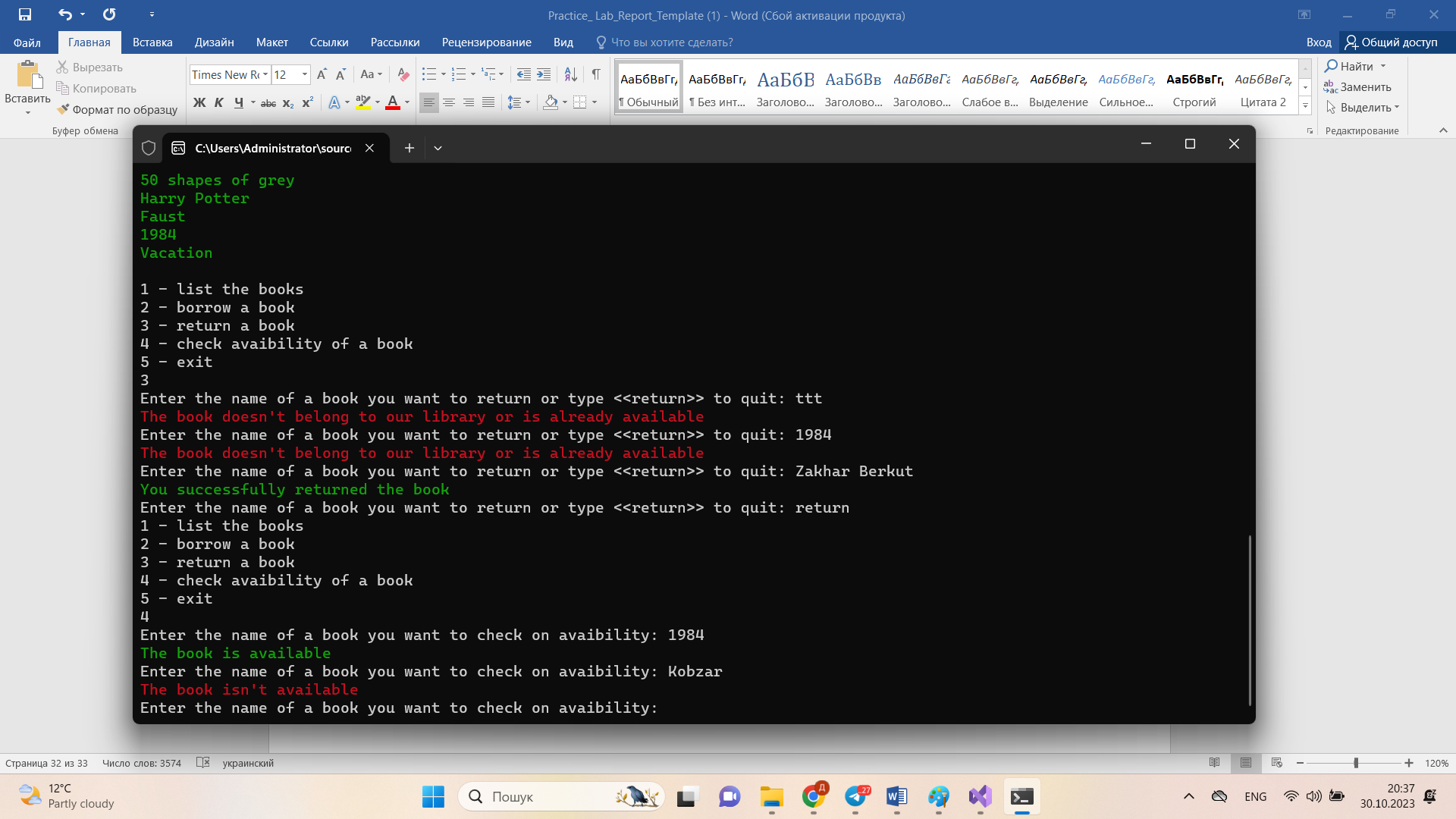
## **3. Результати виконання завдань, тестування та фактично затрачений час:**

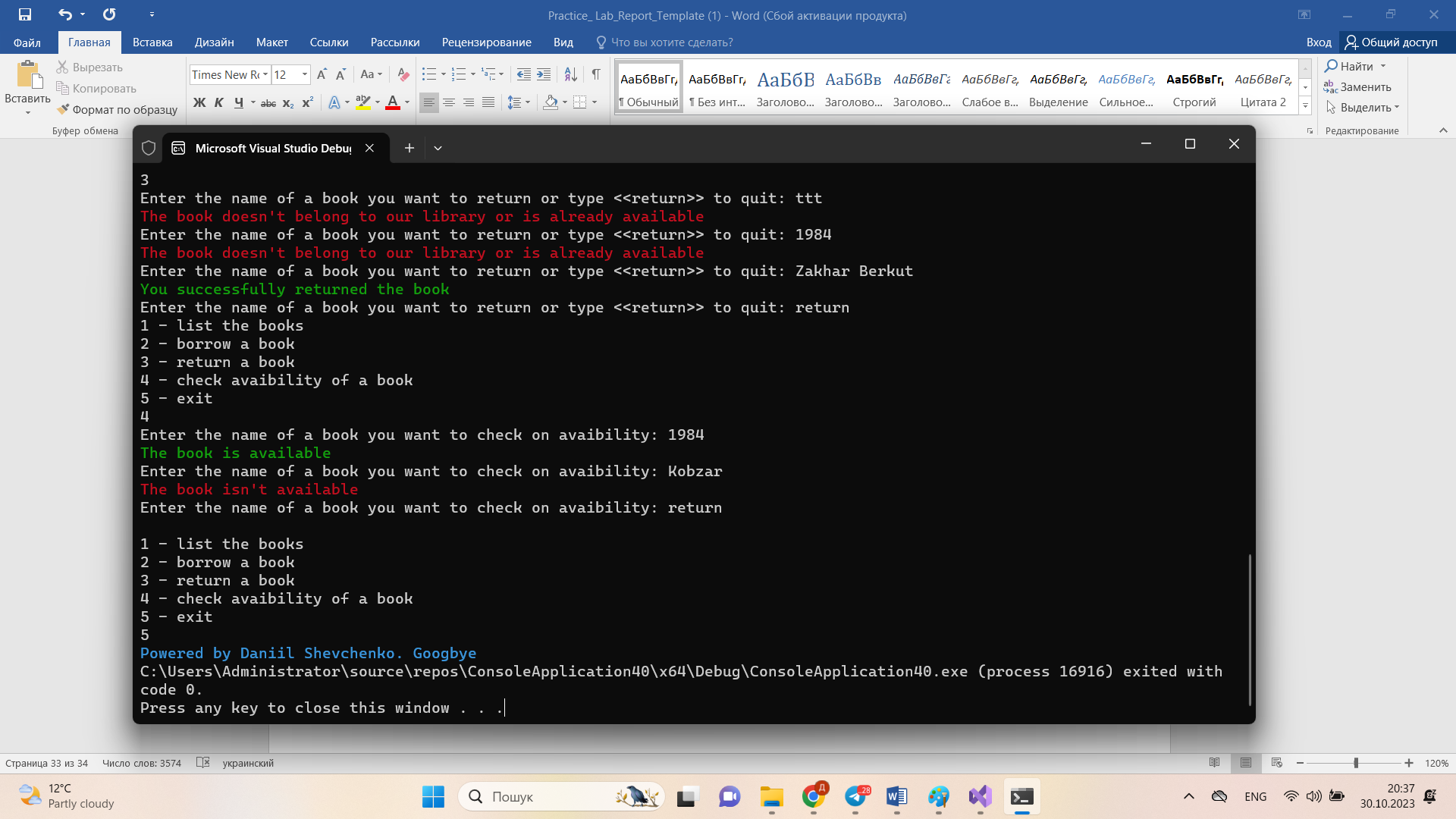












Результат виконання програми

Час затрачений на виконання завдання – 6 годин  
**Self-Practice(Algotester Lab3V1)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

# Lab 3v1

*Limits: 1 sec., 256 MiB*

Ви з’явились у світі під назвою Атод посеред Пустелі Безправ’я. Так сталось, що Ви попали саме в той час і місце, де ведеться битва між чаклункою Ліною і темними силами, які хочуть знищити цей світ. На жаль, трапилась халепа, бо деякі слова із книги чар були пошкоджені під час битви. Одне таке слово можна відновити виконавши ритуал зцілення над пошкодженими буквами. Ритуал зцілення можна виконати на всіх **підряд** розташованих **пошкоджених** буквах. Вам не залишається нічого іншого як допомогти Ліні відновити ці слова і сказати скільки мінімально треба провести таких ритуалів, щоб прочитати одне з наймогутніших у цьому світі заклять - Поневолення Дракона!

# Input

У першому рядку N - кількість рядків у заклятті.

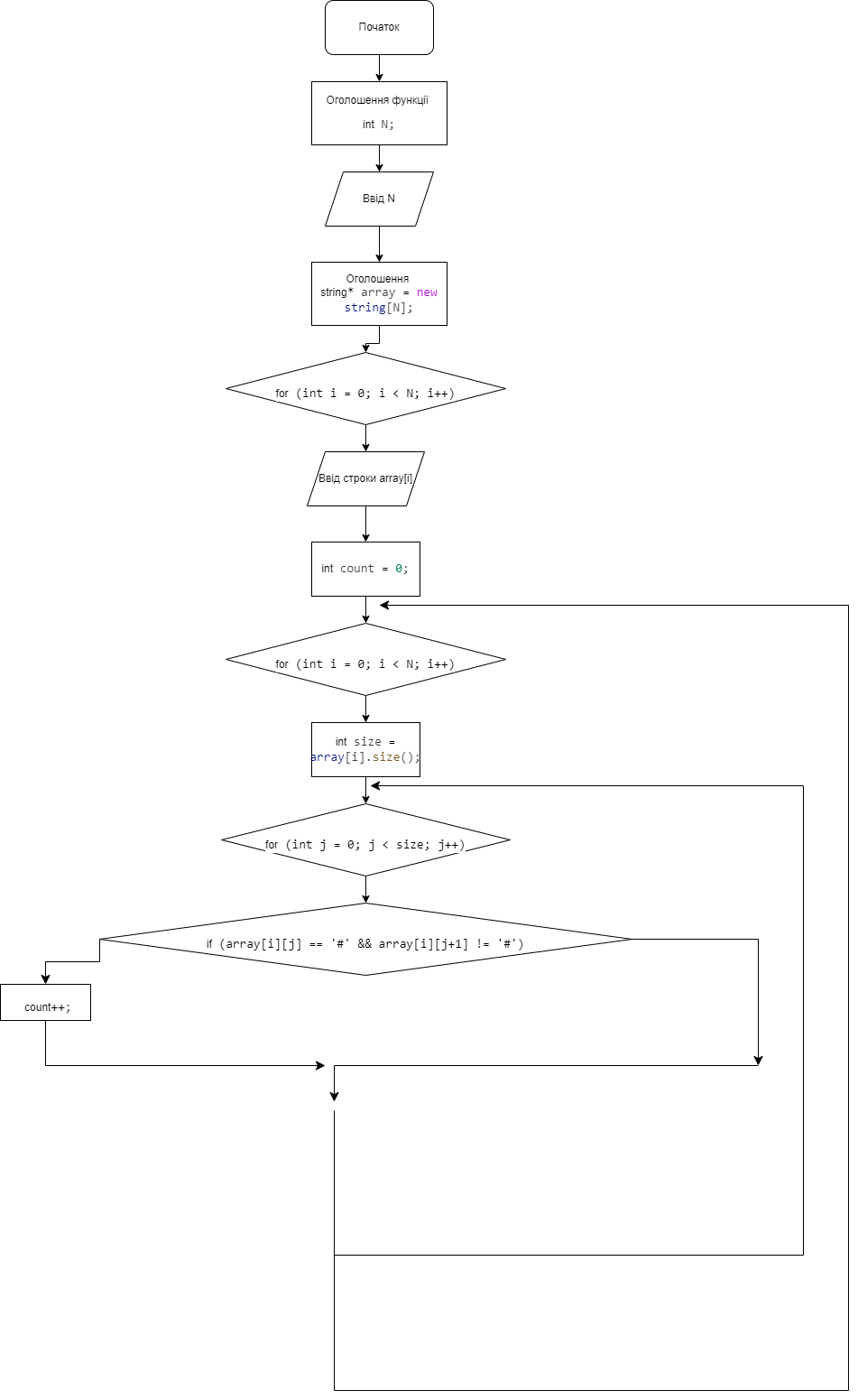
В наступних N рядкях - набір слів w1,…,wM розділених пробілами, де кожне слово може містити малі латинські літери та символ #, який позначає пошкоджену букву.

# Output

Єдине ціле число - мінімальна кількість ритуалів, які потрібно провести, щоб відновити закляття.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма Self-Practice



* Планований час на реалізацію – 30 хвилин

## **3. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

int main()

{

    int N;

    cin >> N;

    string\* array = new string[N];

    cin.ignore(30000, '\n');

    for (int i = 0; i < N; i++)

    {

        getline(cin, array[i]);

    }

    int count = 0;

    for (int i = 0; i < N; i++)

    {

        int size = array[i].size();

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

        {

            if (array[i][j] == '#' && array[i][j+1] != '#')

            {

                count++;

            }

        }

    }

    cout << count;

}

## **4. Результати виконання завдань, тестування та фактично затрачений час:**



Час затрачений на виконання завдання – 40 хвилин

**Self-Practice(Задача з Algotester)**

## **1. Опрацювання завдання та вимог до програм та середовища:**

# Депутатські гроші

*Обмеження: 2 сек., 256 МіБ*

Часто-густо громадяни намагаються з’ясувати, наскільки багатими є депутати. Дехто вірить, що матеріальні статки окремих депутатів є необмеженими.

Тож уявіть собі депутата, у якого є необмежена кількість купюр усіх номіналів (1, 2, 5, 10, 20, 50, 100, 200 та 500 гривень). Він хоче придбати подарунок для своєї тещі, що коштує n гривень.

Незважаючи на те, що наш герой-олігарх є нескінченно багатим, він також є нескінченно скупим та педантичним. Саме тому він хоче оплатити покупку готівкою без решти. Яка мінімальна кількість купюр йому для цього знадобиться?

# Вхідні дані

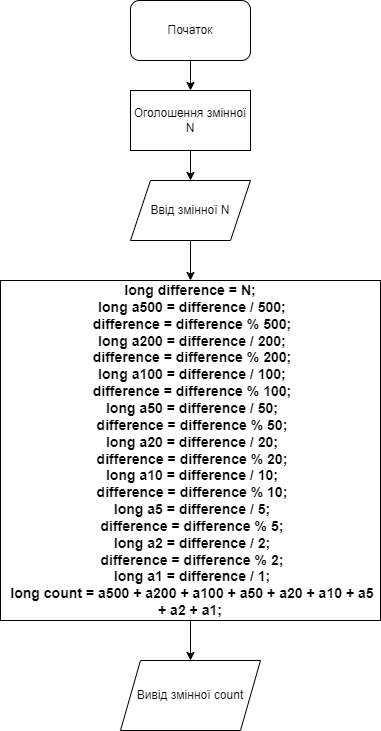
У єдиному рядку задано одне натуральне число n� — вартість подарунку.

# Вихідні дані

У єдиному рядку виведіть одне ціле число — мінімальну кількість купюр, що необхідна для покупки подарунка.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма Self-Practice



## **3. Код програм з посиланням на зовнішні ресурси:**

#include <iostream>

using namespace std;

int main()

{

long N;

cin >> N;

long difference = N;

long a500 = difference / 500;

difference = difference % 500;

long a200 = difference / 200;

difference = difference % 200;

long a100 = difference / 100;

difference = difference % 100;

long a50 = difference / 50;

difference = difference % 50;

long a20 = difference / 20;

difference = difference % 20;

long a10 = difference / 10;

difference = difference % 10;

long a5 = difference / 5;

difference = difference % 5;

long a2 = difference / 2;

difference = difference % 2;

long a1 = difference / 1;

long count = a500 + a200 + a100 + a50 + a20 + a10 + a5 + a2 + a1;

cout << count;

}

## **4. Результати виконання завдань, тестування та фактично затрачений час:**

# 

# **Висновки:**

Отже, в ході роботи над епіком 3 я навчився працювати з різними типами циклів, зокрема while, do while, for i foreach, зрозумів різницю між ними та випадки, в яких їх доцільно застосовувати. Також я навчився працювати з функціями, ознайомився з поняттям параметрів та аргументів функцій, різними типами повернень(в тому числі void), зрозумів різницю між передачою аргументів функцій за значенням, посиланням та адресою. Окремо я ознайомився з поняттям еліпсису і як його використовувати для задання функцій з нефіксованою кількістю аргументів, а також ознайомився з рекурсивними функціями.

Pull-Request: https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/498