Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 2**

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

***з розділу***: «Лінійні та розгалужені алгоритми. Умовні оператори. Константи, змінні»

***Виконала:***

студент групи ШІ-12

Виклюк Яна Ярославівна

# **Тема роботи:**

Операції та їх види. Лінійні та розгалужені алгоритми у С++. Умовні оператори, їх види та використання. Константи та змінні.

# **Мета роботи:**

Ознайомитись з операціями та їх видами у С++. Ознайомитись з лінійними та розгалуженими алгоритмами. Ознайомитись та навчитися використовувати умовні оператори та їх види, константи та змінні.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Операції.
* Тема №2: Лінійні та розгалужені алгоритми у С++.
* Тема №3: Умовні оператори, їх види та використання.
* Тема №3: Константи та змінні.
* Тема №5: Цикл for (для self practice).

1. Індивідуальний план опрацювання теорії:

* Тема №1: Операції.
  + Джерела Інформації:
    - https://acode.com.ua/urok-42-aryfmetychni-operatory/
  + Що опрацьовано:
    - Опрацьовано операції та їх види.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 20.10
* Звершення опрацювання теми: 21.10
* Тема №2 : Лінійні та розгалужені алгоритми у С++.
  + Джерела Інформації
    - http://cpp.dp.ua/ponyattya-algoritmu
  + Що опрацьовано:
    - Опрацьовано поняття алгоритму, основні типи алгоритмів
  + Статус: Ознайомлений
  + Початок опрацювання теми: 21.10
  + Звершення опрацювання теми: 21.10
* Тема №3: Умовні оператори, їх види та використання.
  + Джерела Інформації:
    - https://acode.com.ua/urok-67-operatory-umovnogo-rozgaluzhennya-if-else/
    - <https://acode.com.ua/urok-68-operator-switch/>
  + Що опрацьовано:
    - Опрацьовано умовні оператори ( if-else, switch case)
  + Статус: Ознайомлений
  + Початок опрацювання теми: 23.10
  + Звершення опрацювання теми: 24.10
* Тема №4: Константи та змінні.
  + Джерела Інформації:
    - https://purecodecpp.com/uk/archives/165
  + Що опрацьовано:
    - Опрацьовано константи та змінні, їх оголошення та ініціалізація
  + Статус: Ознайомлений
  + Початок опрацювання теми: 24.10
  + Звершення опрацювання теми: 25.10
* Тема №5 Цикл for.
  + Джерела Інформації:
    - https://acode.com.ua/urok-72-tsykl-for/
    - http://cpp.dp.ua/operatory-tsyklu/
  + Що опрацьовано:
    - Опрацьовано роботу циклу for
  + Статус: Ознайомлений
  + Початок опрацювання теми: 27.10
  + Звершення опрацювання теми: 28.10

# **Виконання роботи:**

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

Завдання №1 VNS Lab 1 Task 1

* Варіант №18
* Деталі завдання
* Обчислити значення виразу при різних дійсних типах даних (float й double),  використовуючи бібліотеку для математичних функцій <cmath> для піднесення числа або многочлена до степеня.
* Важливі деталі для врахування в імплементації програми
* Зрозуміти принцип роботи з функцією підняття до степеня pow, проаналізувати чому виходять різні результати при double і float.

Завдання №2 VNS Lab 1 Task 2

* Варіант №18
* Деталі завдання
* Обчислити та порівняти числа (деякі з яких мають постфіксні операції), котрі вводить користувач, вивести відповідні результати обчислення та порівняння
* Важливі деталі для врахування в імплементації програми
* Розуміти принцип роботи постфіксної операції(її відмінність від префіксної)

Завдання №3 Algotester Lab 1 Task 1

* Варіант №2
* Деталі завдання
* Потрібно визначити чи стiл буде стояти паралельно площинi пiдлоги та довжина найменшої нiжки не буде рiвна нулю, знаючи, що стіл перевернеться якщо hmax >= 2 ∗ hmin.
* Важливі деталі для врахування в імплементації програми
* Треба врахувати, що стіл може перевернутись мiж початком та кiнцем вiдпилювання.

Завдання №4 Class Practice Task

* Варіант завдання відсутній
* Деталі завдання
* Створити простий порадник щодо погоди. Користувач вводить поточні погодні умови, а програма видає рекомендації щодо активності на основі погоди.

Завдання №5 Self Practice Task (Марічка і печиво)

* Варіант завдання відсутній
* Умова

Зібралися Зеник і Марічка разом з пластунами в похід. Похід — серйозна справа. Потрібно запастись продуктами харчування та розподілити їх споживання по днях так, щоб всім вистачило. Цього разу Зеник слідкує за тим, щоб печива вистачило аж до останнього дня походу. Зеник чітко знає, скільки пачок печива повинно залишитись кожного дня, і щовечора перераховує їх. Якщо Зеник побачить, що залишилось менше пачок, ніж повинно залишитись за його розрахунками, він неодмінно знайде того, хто з’їв забагато печива, і покарає його.

Марічка дуже любить печиво. Сьогодні, коли всі пластуни покинуть свої намети і підуть купатися в річку, Марічка планує непомітно з’їсти трохи печива. Звісно, Марічка не хоче бути покараною і дуже боїться, щоб Зеник не помітив пропажу.

Марічка підгледіла, скільки пачок печива є в рюкзаку Зеника. Також вона знає, скільки штук в кожній пачці. Марічці не терпиться дізнатися, скільки ж печива вона зможе з’їсти так, щоб Зеник не помітив. Зеник помітить пропажу печива з деякої пачки тоді і тільки тоді, коли Марічка повністю спустошить її.

Вхідні дані

У першому рядку задано одне натуральне число n — кількість пачок печива.

У другому рядку задано n натуральних чисел ai — кількість штук печива в i-й пачці.

* Деталі завдання
* Вивести максимальну кількість штук печива, яку зможе з’їсти Марічка так, щоб Зеник не помітив цього.

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

Програма №1 VNS Lab 1 Task 1

* Блок-схема

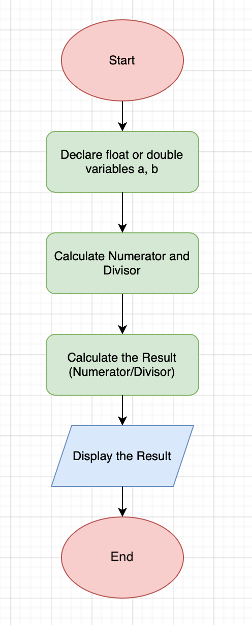


Рисунок Блок-схема до програми №1

* Планований час на реалізацію 20 хв
* Важливі деталі для врахування в імплементації
* Уважно поставити дужки, щоб обчислення були виконані правильно. Чим більше проміжних змінних, тим менш точним є кінцевий результат, адже кожна проміжна змінна має неточність.

Програма №2 VNS Lab 1 Task 2

* Блок-схема

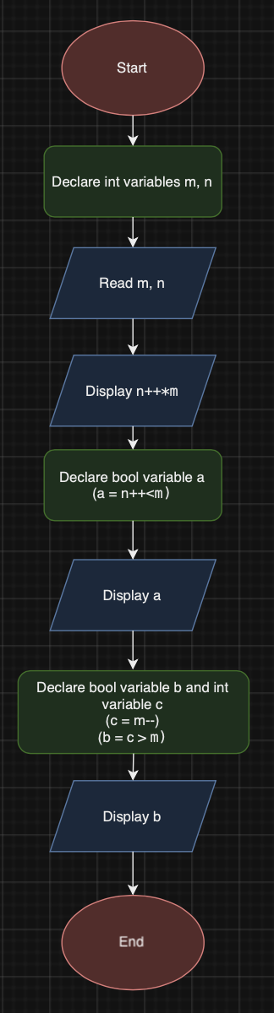


Рисунок Блок-схема до програми №2

* Планований час на реалізацію 20 хв
* Важливі деталі для врахування в імплементації
* Уважно слідкувати за змінними під час виконання операцій, враховуючи постфіксні операції

Програма №3 Algotester Lab 1 Task 1

* Блок-схема

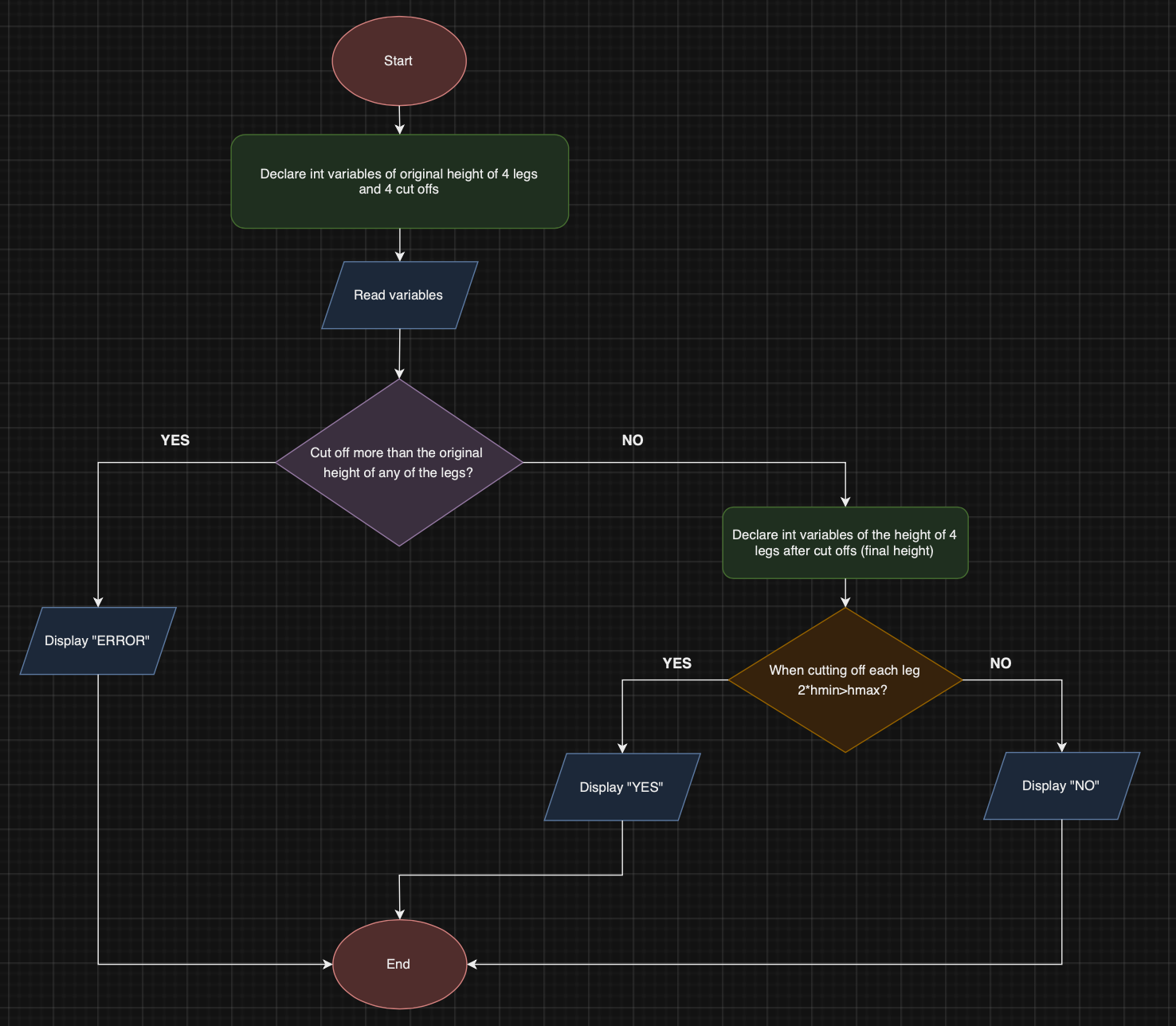


Рисунок Блок-схема до програми №3

* Планований час на реалізацію 90 хв
* Важливі деталі для врахування в імплементації
* Правильно ввести умови за яких стіл не перевернеться, перевіривши та порівнявши з уже відрізаними ніжками кожну невідрізану після кожного відрізання.

Програма №4 Class Practice Task

* Блок-схема

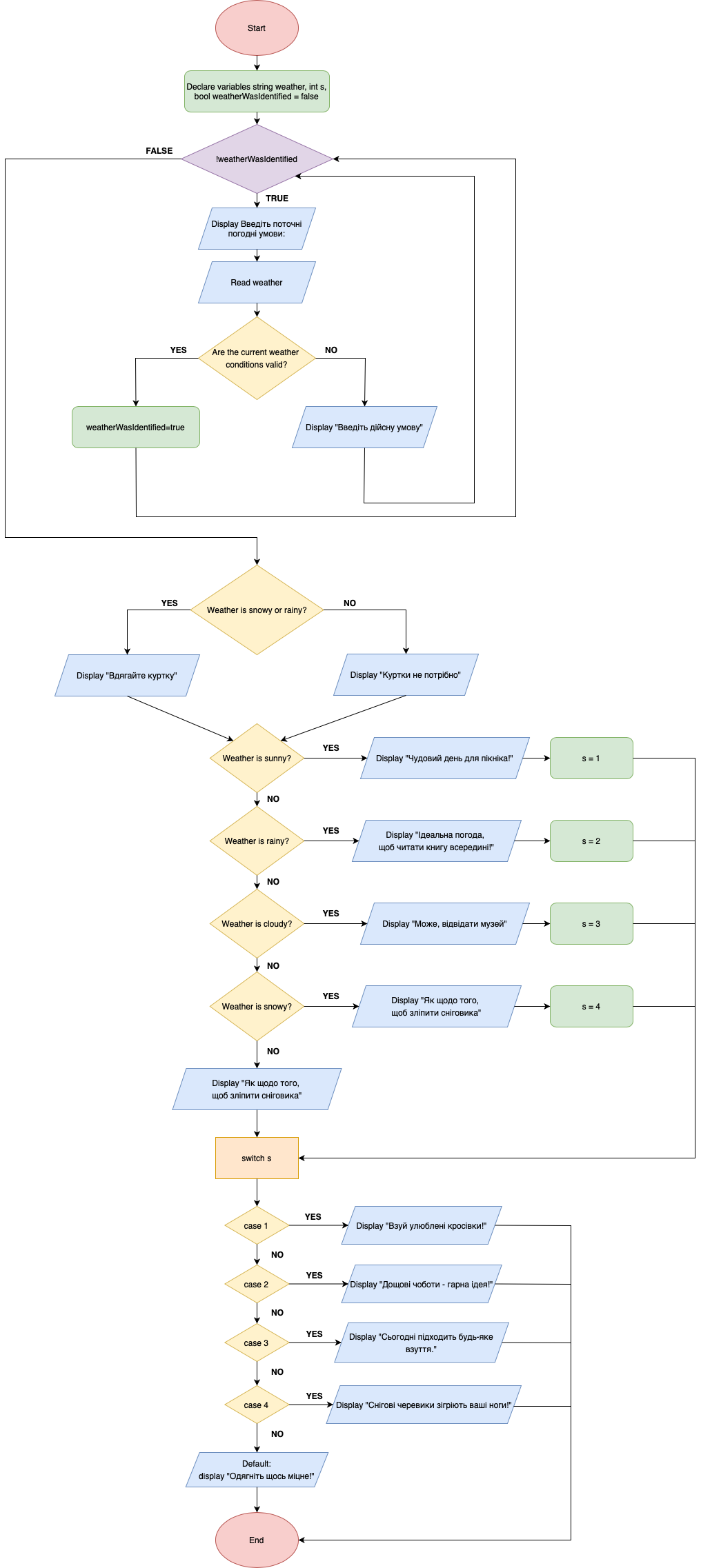


Рисунок Блок-схема до програми №4

* Планований час на реалізацію 45 хв
* Важливі деталі для врахування в імплементації
* Ввести додаткову int змінну для використання switch case.

Програма №5 Self Practice Task (Марічка і печиво)

* Блок-схема

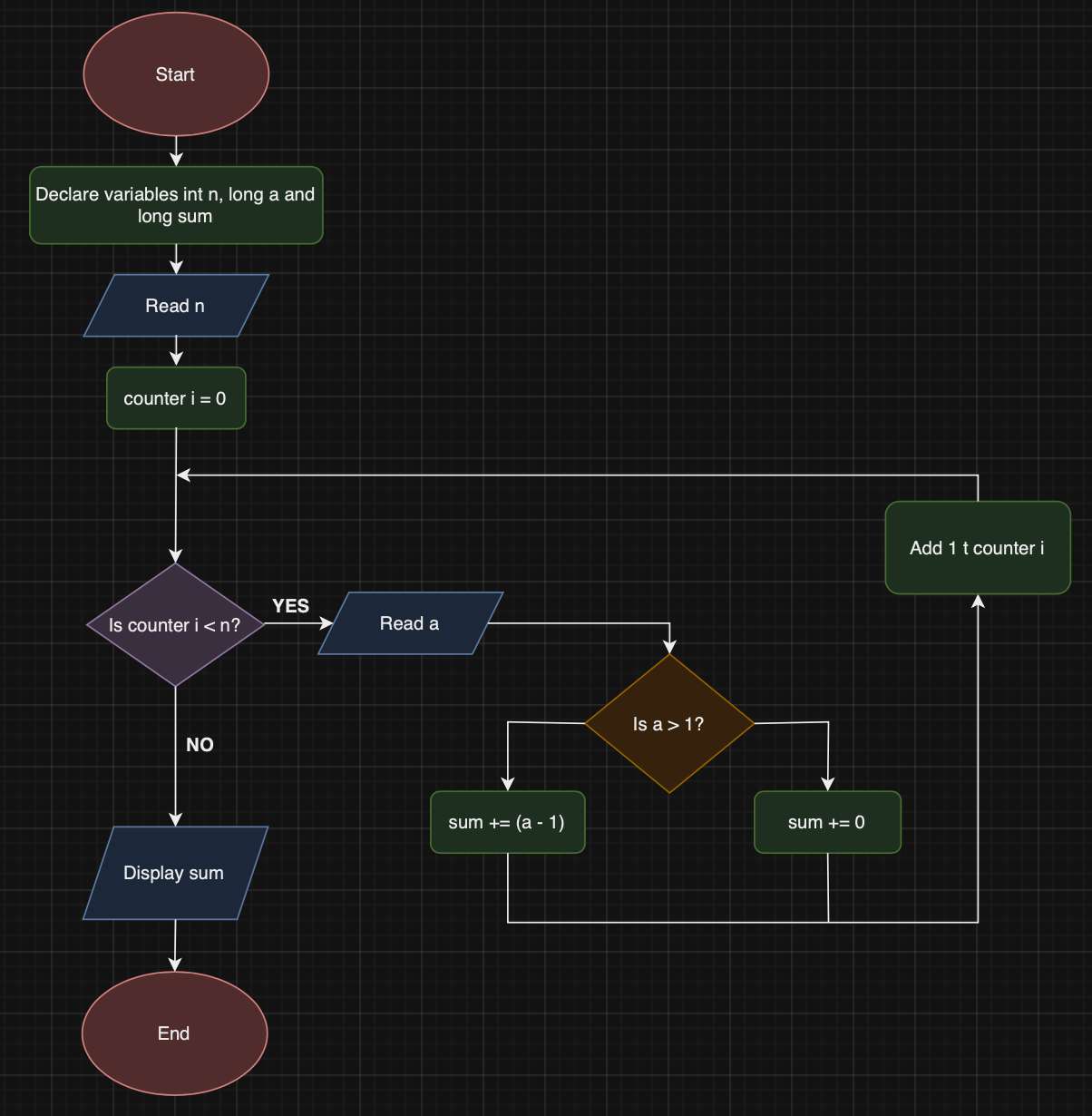


Рисунок Блок-схема до програми №5

* Планований час на реалізацію 40 хв
* Важливі деталі для врахування в імплементації
* Врахувати обмеження змінних

## **3. Конфігурація середовища до виконання завдань:**

Завдання №1 VNS Lab 1 Task 1

Деталі по конфігурації середовища

Потрібно підключити лише бібліотеку <cmath>, котра містить більшість математичних функцій.

Завдання №2 VNS Lab 1 Task 2

Деталі по конфігурації середовища

Достатньо базових розширень та налаштувань Visual Studio Code, C++, нічого додаткового не потрібно.

Завдання №3 Algotester Lab 1 Task 1

Деталі по конфігурації середовища

Достатньо базових розширень та налаштувань Visual Studio Code, C++, нічого додаткового не потрібно.

Завдання №4 Class Practice Task

Деталі по конфігурації середовища

Для використання рядків типу string потрібно підключити заголовковий файл <string>.

Завдання №5 Self Practice Task (Марічка і печиво)

Деталі по конфігурації середовища

Достатньо базових розширень та налаштувань Visual Studio Code, C++, нічого додаткового не потрібно

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

Завдання №1 VNS Lab 1 Task 1

Деталі по програмі

Потрібно обчислити значення заданого виразу, що містить операції піднесення до степеня. Щоб виконати цю операцію необхідно підключити бібліотеку <cmath>, у якій є функція pow(), котра і виконує операцію піднесення до степеня.

#include <iostream>

#include <cmath>

using namespace std;

float a = 1000;

float b = 0.0001;

float Numeratorf = pow((a-b),3) - pow(a,3);

float Divisorf = pow (b,3) - 3 \* a \* pow(b,2) - 3 \* pow(a,2) \* b;

float Resultf = Numeratorf/Divisorf;

double ad = 1000;

double bd = 0.0001;

double Numeratord = pow((ad-bd),3) - pow(ad,3);

double Divisord = pow (bd,3) - 3 \* ad \* pow(bd,2) - 3 \* pow(ad,2) \* bd;

double Resultd = Numeratord/Divisord;

int main()

{

cout << Resultf << endl;

cout << Resultd << endl;

return 0;

}

Блок з кодом до завдання №1

Завдання №2 VNS Lab 1 Task 2

Деталі по програмі

Обчислити та порівняти числа (деякі з яких мають постфіксні операції), котрі вводить користувач, вивести відповідні результати обчислення та порівняння. Потрібно розуміти як працюють постфіксні операції та їхня різниця від префіксних.

#include <iostream>

using namespace std;

int m;

int n;

int main()

{

cin >> m;

cin >> n;

cout << n++\*m << endl;

bool a = n++<m;

cout << a << endl;

int c = m--;

bool b = c > m;

cout << b << endl;

return 0;

}

Блок з кодом до завдання №2

Завдання №3 Algotester Lab 1 Task 1

Деталі по програмі

Потрібно визначити чи стiл буде стояти паралельно площинi пiдлоги, тобто 2 ∗ hmin має бути більше за hmax. Ми відрізаємо ніжки по черзі, тож зробивши перше відрізання порівнюємо відрізану ніжку з усіма цілими, тобто з трьома(якась з них буде максимальною, тобто або перша, або друга, або третя), потім виконуємо те саме при другому відрізанні, та порівнюємо другу відрізану(котра може бути мінімальною) з двома цілими, потім аналогічно відрізаємо третю ніжку та порівнюємо її з останньою цілою. Для того щоб стіл не перевернувся умови для кожного відрізання мають бути виконані водночас. Якщо довжина відрізання буде більшою а початкову довжину ніжки, то програма видасть помилку, в інших випадках стіл перевернеться.

#include <iostream>

using namespace std;

int OriginalHeight1, OriginalHeight2, OriginalHeight3, OriginalHeight4;

int Cut1, Cut2, Cut3, Cut4;

int main ()

{

cin >> OriginalHeight1 >> OriginalHeight2 >> OriginalHeight3 >> OriginalHeight4;

cin >> Cut1 >> Cut2 >> Cut3 >> Cut4;

if (Cut1>OriginalHeight1 || Cut2>OriginalHeight2 || Cut3>OriginalHeight3 || Cut4>OriginalHeight4)

{

cout << "ERROR" << endl;

}

else

{

int FinalHeight1 = OriginalHeight1 - Cut1;

int FinalHeight2 = OriginalHeight2 - Cut2;

int FinalHeight3 = OriginalHeight3 - Cut3;

int FinalHeight4 = OriginalHeight4 - Cut4;

if ((2\*FinalHeight1>OriginalHeight2 || 2\*FinalHeight1>OriginalHeight3 || 2\*FinalHeight1>OriginalHeight4) && (2\*FinalHeight2>OriginalHeight3 || 2\*FinalHeight2>OriginalHeight4) && 2\*FinalHeight3>OriginalHeight4)

{

cout << "YES" << endl;

}

else

{

cout << "NO" << endl;

}

}

return 0;

}

Блок з кодом до завдання №3

Завдання №4 Class Practice Task

Деталі по програмі

Використати умовні оператори if-else, else if, switch case для створення порадника погоди. Якщо користувач вводить погоду незадану у програмі, то вона буде просити ввести дійсні погодні умови доти, доки користувач цього не зробить. Це виконано за допомогою циклу while.

#include <iostream>

#include <string>

using namespace std;

string weather;

int s;

int main()

{

bool weatherWasIdentified = false;

while(!weatherWasIdentified)

{

cout << "Введіть поточні погодні умови:" << endl;

cin >> weather;

if(weather=="sunny" || weather=="rainy" || weather=="cloudy" || weather=="snowy" || weather=="windy")

{

weatherWasIdentified=true;

}

else

{

cout << "Введіть дійсну умову" << endl;

}

}

if (weather=="snowy" || weather=="rainy")

{

cout << "Вдягайте куртку" << endl;

}

else

{

cout << "Куртки не потрібно" << endl;

}

if (weather=="sunny")

{

cout << "Чудовий день для пікніка!" << endl;

s = 1;

}

else if (weather=="rainy")

{

cout << "Ідеальна погода, щоб читати книгу всередині!" << endl;

s = 2;

}

else if (weather=="cloudy")

{

cout << "Може, відвідати музей" << endl;

s = 3;

}

else if (weather=="snowy")

{

cout << "Як щодо того, щоб зліпити сніговика?" << endl;

s = 4;

}

else

{

cout << "Запустіть повітряного змія, якщо він у вас є!" << endl;

}

switch(s)

{

case 1:

cout << "Взуй улюблені кросівки!" << endl;

break;

case 2 :

cout << "Дощові чоботи - гарна ідея!" << endl;

break;

case 3 :

cout << "Сьогодні підходить будь-яке взуття." << endl;

break;

case 4 :

cout << "Снігові черевики зігріють ваші ноги!" << endl;

break;

default:

cout << "Одягніть щось міцне!" << endl;

break;

}

return 0;

}

Блок з кодом до завдання №4

Завдання №5 Self Practice Task (Марічка і печиво)

Деталі по програмі

Якщо кількість печива у пачці буде більшою за одиницю, то до загальної кількості печива котре може з’їсти Марічка додаємо кількість печива на 1 меншу, адже Зеник не помітить цього, якщо хоча б 1 печиво залишиться у пачці, у іншому випадку сума не змінюється, адже Марічка не може з’їсти печиво непоміченою. Перевіривши кількість печива у кожній пачці, та виконавши обчислення можемо вивести максимальну кількість печива, котре може з’їсти Марічка.

#include <iostream>

using namespace std;

int n;

long sum;

long a;

int main()

{

cin >> n;

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

{

cin >> a;

if (a > 1)

{

sum += (a - 1);

}

else

{

sum += 0;

}

}

cout << sum << endl;

return 0;

}

Блок з кодом до завдання №5

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

Завдання №1 VNS Lab 1 Task 1

Деталі по виконанню і тестуванню програми

При виконанні цієї програми результати з типами даних float та double відрізняються, адже на float виділяється 4 байти пам’яті, а на double 8 байтів, тобто у 2 рази більше. Результат при застосуванні double буде точніший, проте займатиме більше пам’яті. Також кількість проміжних змінних впливає на точність результату, чим їх менше тим точнішим є результат.



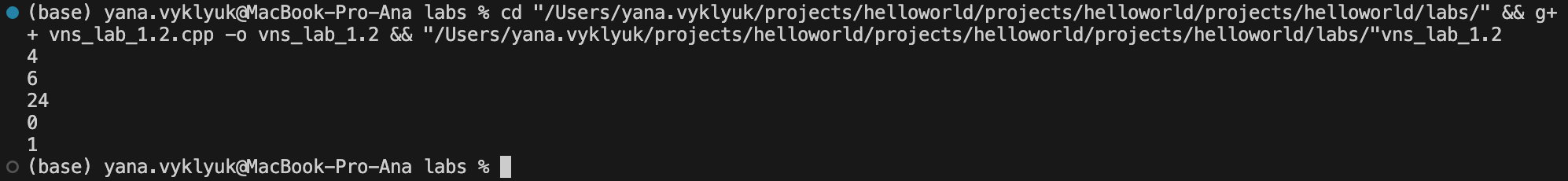
Блок з виконанням та тестуванням програми до завдання №1

Час затрачений на виконання завдання 25 хв.

Завдання №2 VNS Lab 1 Task 2

Деталі по виконанню і тестуванню програми

Першим виведеним результатом n++\*m є 24, що дорівнює n\*m, проте після виконання цієї операції значення n збільшилось на 1, тож при виконанні уже наступної операції n++<m, буде використовуватись n збільшене на одиницю, після виконання порівняння n знову збільшиться на 1. В останньому виразі m-- >m завжди буде виводиттись 1, тобто твердження буде правдивим, бо m зменшене на 1 буде знаходитись у правій частині нерівності, тож ліва буде більшою.



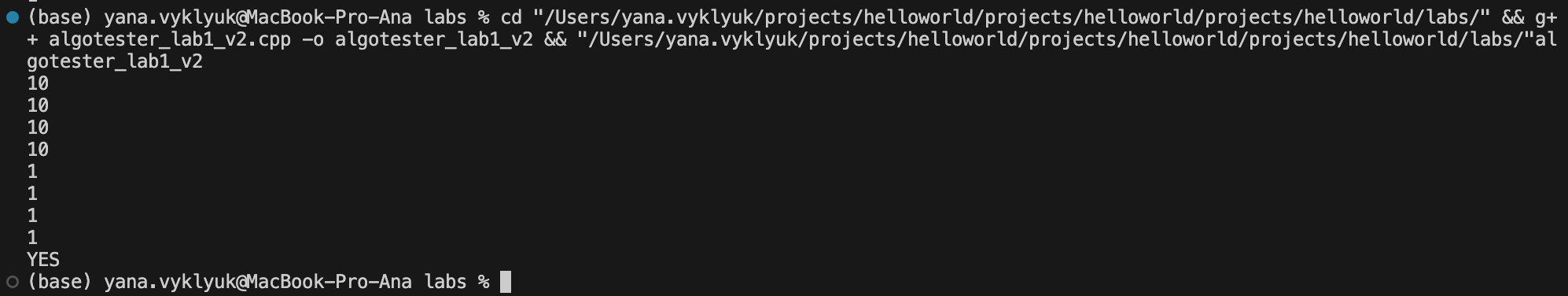
Блок з виконанням та тестуванням програми до завдання №2

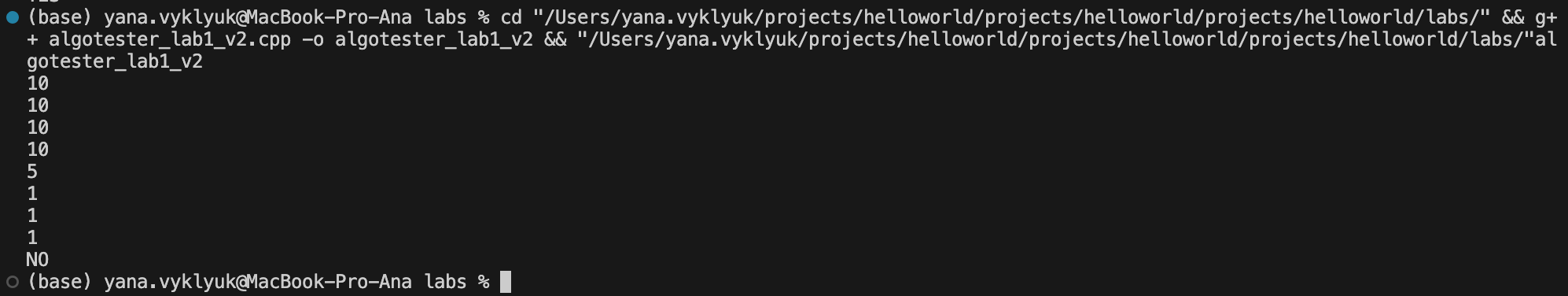
Час затрачений на виконання завдання 15 хв

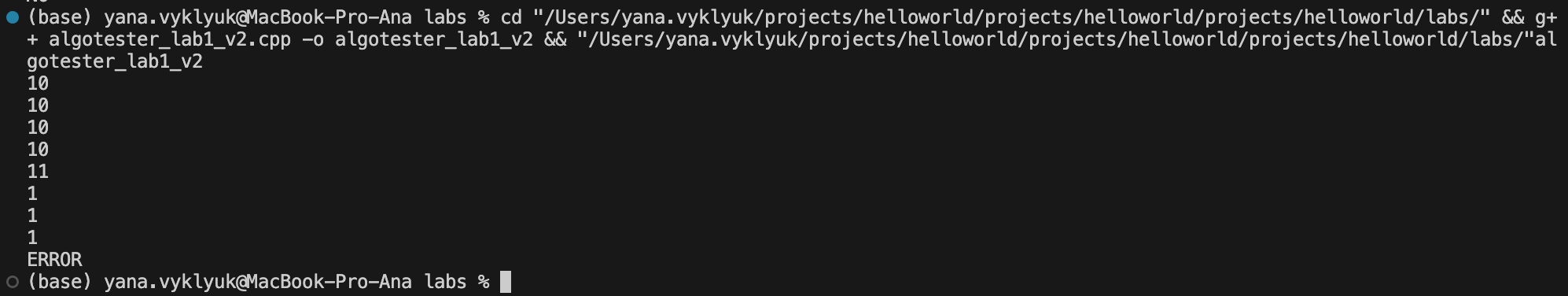
Завдання №3 Algotester Lab 1 Task 1

Деталі по виконанню і тестуванню програми

Перший приклад задовольняє умову 2 ∗ hmin має бути більше за hmax для кожного відрізання, тому виводиться “YES”. Другий приклад не задовольняє умову, адже 2 ∗ hmin дорівнює hmax, тож стіл перевернеться, виводиться “NO”. У третьому прикладі довжина відрізання довша за початкову довжину ніжки, тому виводиться “ERROR”, і програма не виконується далі.







Блок з виконанням та тестуванням програми до завдання №3

Час затрачений на виконання завдання 90 хв

Завдання №4 Class Practice Task

Деталі по виконанню і тестуванню програми

При введені незаданих погодних умов програма просить ввести дійсні умови. Ввівши останні, залежно від погоди програма виводить чи варто вдягати куртку, чим можна зайнятись та що взути.



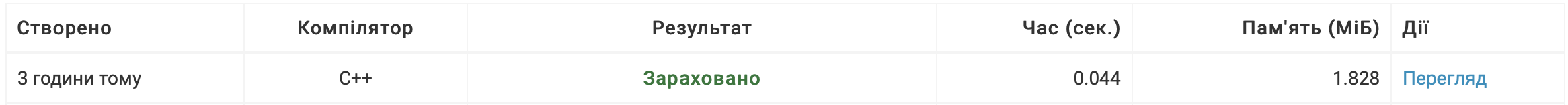
Блок з виконанням та тестуванням програми до завдання №4

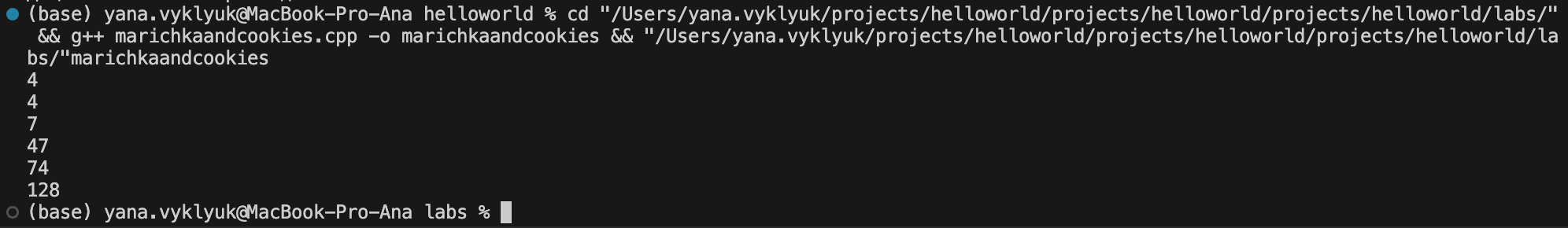
Час затрачений на виконання завдання 50 хв.

Завдання №5 Self Practice Task (Марічка і печиво)

Деталі по виконанню і тестуванню програми

Код програми загружено на Algotester. Програма виводить максимальну кількість печива, котру може з’їсти Марічка, щоб Зеник не помітив цього.





Блок з виконанням та тестуванням програми до завдання №5

Час затрачений на виконання завдання 40 хв.

# **Висновки:**

Ознайомлено з операціями та їх видами у С++. Використано постфіксні операції у лабораторній роботі. Ознайомлено з лінійними та розгалуженими алгоритмами. Ознайомлено та використано умовні оператори, а саме if else, else if, switch case. Ознайомлено з особливостями роботи та синтаксису кожного умовного оператора. Використано бібліотеку <cmath> та функцію pow(), тобто піднесення до степеня, для виконання обчислень. Ознайомлено та використано цикли for та while, для виконання завдань. Поглиблено знання з теми змінних та констант.