Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

ДО РОЗРАХУНКОВОЇ РОБОТИ №1
З КУРСУ ОСНОВИ ПРОГРАМУВАННЯ.

Виконав:
Студент групи ШІ-11
Бубельник Юрій Олегович

Львів 2024

Мета роботи:

Одержати практичні навички в розробці і дослідженні алгоритмів розв’язання задач.

Теоретичні відомості:

Виконання роботи:

Завдання №1 – VNS Practice Work Task 1 v25

Задача:

Розробити лінійний алгоритм для розв’язання задачі.

Варіант 25. $a = y + \dfrac{x}{y^3 + \left|\dfrac{x^2}{y + \sqrt[3]{x^2}}\right|}$; $b = 1 + tg^2(x/2)$, де x=1,23; y=0,79.

Завдання №2 – VNS Practice Work Task 2 v1

Задача:

Розробити алгоритм, що розгалужується для
розв’язання задачі номер якої відповідає порядковому номеру студента в журналі викладача

Варіант 1. $y = \begin{cases} ax + b\cos x, & x < 0{,}5, \\ bx^2 + c\sin 2x, & 0{,}5 \le x < 1; \end{cases}$

де $x \in [0,2]$; $h_x = 0{,}1$; $a = 0{,}75$; $b = 1{,}19$; $c = -2{,}5$.

Завдання №3 – VNS Practice Work Task 3 v19

Варіант 19. Обчислює площу трикутника, якщо відомі координати його кутів. Нижче приведений вид екрану під час виконання програми, що рекомендується (дані, введені користувачем, які вводяться напівжирним шрифтом).

Обчислення площі трикутника.
Введіть координати кутів
(числа розділяйте пропуском):
x1,y1 > **-2 5**
x2,y2 > **1 7**
x3,y3 > **5 -3**
Площа трикутника: 23.56 кв.см.

Обчислення вартості покупки, що складається з набора зошитів і олівців.

Додатково реалізував запис даних у файл та вивід цих даних у консоль, за вибором користувача.

Завдання №4 – VNS Practice Work Task 4 v16

Варіант 16. Скласти програму, яка генерує послідовності з 10 випадкових чисел в діапазоні від 1 до 10, виводить ці числа на екран і обчислює їх середнє арифметичне.

Algotester Practice Work

Завдання №5 Algotester Lab 1v1

Задача:

У вашого персонажа є H хітпойнтів та M мани.
Персонаж 3 рази використає закляття, кожне з яких може використати хітпойнти та ману одночасно.
Якщо якесь закляття забирає і хітпойнти і ману - ваш персонаж програє, отже для виграшу треба використовувати при одному заклинанні АБО хітпойнти, АБО ману.
Якщо в кінці персонаж буде мати додатню кількість хітпойнтів та мани (H,M>0H,M>0) - він виграє, в іншому випадку програє.
Ваше завдання у випадку виграшу персонажа вивести YES, вивести NO у іншому випадку.

Input

2 цілих числа H та M - хітпойнти та мана персонажа

3 рядки по 2 цілих числа, h_i та m_i - кількість хітпойнтів та мани, які ваш персонаж потратить за хід на i заклинання

Output

YES - якщо ваш персонаж виграє

NO - у всіх інших випадках

Constraints

$1 \leq H \leq 10^{12}$

$1 \leq M \leq 10^{12}$

$0 \leq h_i \leq 10^{12}$

$0 \leq m_i \leq 10^{12}$

Завдання №6 Algotester Lab 2v1

Задача:

У вас є дорога, яка виглядає як NN чисел.
Після того як ви по ній пройдете - вашу втому можна визначити як різницю максимального та мінімального елементу.
Ви хочете мінімізувати втому, але все що ви можете зробити - викинути одне число з дороги, тобто забрати його з масиву.
В результаті цієї дії, яку мінімальну втому ви можете отримати в кінці дороги?

Input

У першому рядку ціле число N - кількість чисел

У другому рядку масив r, який складається з N цілих чисел

Output

Єдине ціле число m - мінімальна втома, яку можна отримати

Constraints

$1 \leq N \leq 10^5$

$0 \leq r_i \leq 10^5$

Завдання №7 Algotester Lab 5v3

Задача:

У вас є карта гори розміром N×M.
Також ви знаєте координати {x,y}, у яких знаходиться вершина гори.
Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.
Клітинкі які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

Input

У першому рядку 2 числа N та M - розміри карти

у другому рядку 2 числа x та y - координати піку гори

Output

N рядків по M елементів в рядку через пробіл - висоти карти.

Constraints

$1 \leq N, M \leq 10^3$

$1 \leq x \leq N$

$1 \leq y \leq M$

Завдання №8 Algotester Lab 3v3

Задача:

Вам дана стрічка s.

Ваше завдання зробити компресію стрічки, тобто якщо якась буква йде більше одного разу підряд у стрічці замінити її на букву + кількість входжень підряд.

Input

У першому рядку стрічка S

Output

Стрічка $S_{compressed}$

Constraints

$1 \leq |S| \leq 10^5$

Код програм з посиланням на зовнішні ресурси:

Завдання №1:

VNS Practice Work Task 1 v25

vns_practice_work_1_task_yurii_bubelnyk.cpp > main()

```cpp
#include <iostream>
#include <cmath> // підключив бібліотеку для pow() та tan()
using namespace std;

int main(){

    double x = 1.23; // ініціалізація x
    double y = 0.79; // ініціалізація y

    double a = y + x / (pow(y, 3) + abs(pow(x, 2)/(y + pow(x, 2.0/3.0)))); // формула для a
    double tg = tan( x / 2 ); // обрахунок тангенса
    double b = 1 + pow(tg, 2); // формула для b

    cout << "Value of a = " << a <<  "\t\tValue of b = " << b << endl << endl; // вивід значень
    return 0;
}
```

Завдання №2:

VNS Practice Work Task 2 v1

vns_practice_work_2_task_yurii_bubelnyk.cpp > calculate(double, double, double, double, double, double)

```cpp
#include <iostream>
#include <cmath> // для cos() і sin()

using namespace std;

// Рекурсивна функція
void calculate(double x, double x_end, double hx, double a, double b, double c);

int main() {
    double a = 0.75;
    double b = 1.19;
    double c = -2.5;
    // Значення для X
    double x_start = 0.0;
    double x_end = 2.0;
    double hx = 0.1;

    // Виклик рекурсивної функції
    calculate(x_start, x_end, hx, a, b, c);

    return 0;
}

void calculate(double x, double x_end, double hx, double a, double b, double c) {
    if (x > x_end) { // коли X вийде за межі тоді закінчиться функція
        return;
    }

    double y; // Змінна для збереження обрахунків
    //Перевірка меж
    if (x < 0.5) {
        y = a * x + b; // Випадок для x < 0.5
    } else if (x >= 0.5 && x < 1.0) {
        y = cos(b * x) + c; // Випадок для 0.5 <= x < 1.0
    } else {
        y = sin(2 * x) + b * c; // Випадок для 1.0 <= x <= 2.0
    }

    // Вивід результатів
    cout << "x = " << x << ", y = " << y << endl;

    // Рекурся, функція викликається з наступним кроком X
    calculate(x + hx, x_end, hx, a, b, c);
}
```

Завдання №3:

VNS Practice Work Task 3 v19

vns_practice_work_2_task_yurii_bubelnyk.cpp | vns_practice_work_3_task_yurii_bubelnyki.cpp | Work3.txt

vns_practice_work_3_task_yurii_bubelnyki.cpp > main()

```cpp
#include <iostream>
#include <cmath> // підключив бібліотеку для abs()
#include <iomanip>  // Для встановлення точності виведення
#include <cstdio> // Для fopen, fprintf, fclose

using namespace std;

const int BUFFER = 256;

double Square(double x1, double y1, double x2, double y2, double x3, double y3);

bool IsValid(double x1, double y1, double x2, double y2, double x3, double y3);

double calculateValue(int numNotebooks, int numPencils, double priceNotebook, double pricePencil);

void CreateFile(const char* filename, double (*Func)(int, int, double, double), int numNotebooks, int numPencils, double priceNotebook, double pricePencil);

void ShowFile(const char* filename);

int main() {
    const char* filename = "Work3.txt";
    double x1, y1, x2, y2, x3, y3 = 0;

    do {
        cout << "x1,y1 > " << endl;
        cin >> x1 >> y1;
        cout << "x2,y2 > " << endl;
        cin >> x2 >> y2;
        cout << "x3,y3 > " << endl;
        cin >> x3 >> y3;

        // Перевіряємо, чи введено правильні координати
        if (IsValid(x1, y1, x2, y2, x3, y3)) {
            break; // Виходимо з циклу, якщо координати правильні
        } else {
            cout << "Invalid coordinates! Points cannot be the same or form a degenerate triangle. Please try again." << endl;
        }

    } while (true);  // Повторюється, поки користувач не введе коректні координати

    // Встановлюємо точність до двох знаків після коми
    cout << fixed << setprecision(2);
    cout << "Area of the triangle: " << Square(x1, y1, x2, y2, x3, y3) << " sq.cm." << endl;

    // Введення даних для покупки
    int numNotebooks, numPencils;
    double priceNotebook, pricePencil;

    cout << "\nEnter number of notebooks: ";
    cin >> numNotebooks;
    cout << "Enter price per notebook: ";
    cin >> priceNotebook;

    cout << "Enter number of pencils: ";
    cin >> numPencils;
    cout << "Enter price per pencil: ";
    cin >> pricePencil;
```

```
    cout << "Enter price per pencil: ";
    cin >> pricePencil;

    cout << "Total cost of purchase: " << calculateValue(numNotebooks, numPencils, priceNotebook, pricePencil) << " UAH.\n";
    int choice; // для запису вибору
    // options для користувача (-_0)
    do{
        cout << "Ente choice:\n\t 1)Create the file;\n\t 2)Show the file;\n\t 3)Exit.\n";
        cin >> choice;
        switch (choice)
        {
        case 1:
            CreateFile(filename, calculateValue, numNotebooks, numPencils, priceNotebook, pricePencil);
            break;
        case 2:
                ShowFile(filename);
            break;
        case 3:
            cout << "Exit -_-" << endl;
        default:
            break;
        }
    } while (choice != 3);

    return 0;
}

double Square(double x1, double y1, double x2, double y2, double x3, double y3) {
    // Обчислення площі трикутника через координати
    double square = 0;
    //  обчислюємо площу трикутника за допомогою координат за формулою площі через координати вершин.
    square = 0.5 * ((x1 * y2 + x2 * y3 + x3 * y1) - (y1 * x2 + y2 * x3 + y3 * x1));
    return abs(square);  // Повертаємо абсолютне значення площі
}

bool IsValid(double x1, double y1, double x2, double y2, double x3, double y3) {
    // Перевірка, чи всі точки різні, щоб утворити трикутник
    if ((x1 == x2 && y1 == y2) || (x2 == x3 && y2 == y3) || (x1 == x3 && y1 == y3)) {
        return false;  // Точки не можуть бути однаковими
    }
    // Перевірка на вироджений трикутник (сума двох сторін повинна бути більшою за третю)
    double side1 = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    double side2 = sqrt((x3 - x2) * (x3 - x2) + (y3 - y2) * (y3 - y2));
    double side3 = sqrt((x3 - x1) * (x3 - x1) + (y3 - y1) * (y3 - y1));

    if (side1 + side2 <= side3 || side2 + side3 <= side1 || side3 + side1 <= side2) {
        return false;  // Якщо три сторони не можуть утворити трикутник
    }

    return true;  // Координати правильні для утворення трикутника
}

double calculateValue(int numNotebooks, int numPencils, double priceNotebook, double pricePencil) {
    double sum = numNotebooks * priceNotebook + numPencils * pricePencil;
    return sum;
}
```

```
// передаємо у функцію CreateFile функція calculateValue як pointer
void CreateFile(const char* filename, double (*Func)(int, int, double, double), int numNotebooks, int numPencils, double priceNotebook, double pricePencil){
    FILE* file = fopen(filename, "w"); // Відкриваємо файл для запису
    if (file == nullptr) {
        cerr << "Error while opening th file: " << filename << endl;
        return;
    }

    int result = Func(numNotebooks, numPencils, priceNotebook, pricePencil);

    // Записуємо результат у файл
    fprintf(file, "The total value is: %d\n", result);

    fclose(file); // Закриваємо файл
    cout << "Result has been written to the file: " << filename << endl;
}
// вивід даних з фалу в термінал
void ShowFile(const char* filename){
    FILE* file = fopen(filename, "r"); // Відкриваємо файл для запису
    if (file == nullptr) {
        cerr << "Error while opening th file: " << filename << endl;
        return;
    }

    char buffer[BUFFER];
    cout << "From the file)))" << endl;
    while (fgets(buffer, BUFFER, file) != nullptr) {
        cout << buffer; // Виводимо зчитаний рядок
    }

    fclose(file);
}
```

Завдання №4:

VNS Practice Work Task 4 v16

vns_practice_work_4_task_yurii_bubelnyki.cpp > main()

```cpp
#include <iostream>
#include <ctime> //підключив бібліотеку для srand та rand

using namespace std;

const int SIZE = 10; //задаю розмір масив

void Initialize(int arr[]);

int main(){
    srand(time(NULL)); //для генерації псевдовипадкових чисел при повторному запуску програми

    int arr[SIZE]; // створюю масив типу int

    Initialize(arr); // викликаю функцію для обробки масиву

    return 0;
}
// функція для обрахунку середнього арифметичного послідовності з 10 числе в діапазоні від 1 до 10
void Initialize(int arr[]){

    int sum = 0; // ініціалізація змінної для запису суми послідовості

    for (int i = 0; i < SIZE; i++){
        arr[i] = rand() % 10 + 1; //  обмежую діапазон від [1, 10]
        sum += arr[i]; // сумую значення послідовності
        cout << arr[i] << " "; //  вивід значенньь послідовності
    }

    cout << "\nAverage: " << static_cast<double>(sum) / SIZE << endl; //переведення типу int в double для обчислення середнього арифметичного
}
```

Завдання №5:

Algotester Lab 1v1

Lab-1v1.cpp > main()

```cpp
#include <iostream>
using namespace std;

int main() {
    long long H, M; // long long бо межі 1e12
    cin >> H >> M;

    if (H < 1 || H > 1e12 || M < 1 || M > 1e12) // перевірка умови
        return 1;
    // заповнення значень
    for (int i = 0; i < 3; i++) {
        long long hi, mi;
        cin >> hi >> mi;
    // перевірка умови чи не виходидь за межі необхідна кількість ресурсів на заклинання
        if (hi < 0 || hi > 1e12 || mi < 0 || mi > 1e12)
            return 1;
    // якщо витрачається одразу два показника то програш
        if (hi > 0 && mi > 0) {
            cout << "NO" << endl;
            return 0;
        }
        // віднімання ресурсів після заклинання
        if (hi > 0) {
            H -= hi;
        } else if (mi > 0) {
            M -= mi;
        }
    // перевірка на кількість ресурсів
        if (H <= 0 || M <= 0) {
            cout << "NO" << endl;
            return 0;
        }
    }


    cout << "YES" << endl;
    return 0;
}
```

Завдання №6:

Algotester Lab 2v1

```cpp
#include <iostream>
#include <vector> // для списку розмір якого визначає користувач
using namespace std;

int main() {
    int N;
    cin >> N;

    if (N > 1e5 || N < 1) {
        return 1;
    }

    if (N <= 2) {
        cout << 0 << endl;
        return 0;
    }

    vector<int> r(N); // список з N елементів

    for (int i = 0; i < N; i++) {
        cin >> r[i];
        if (r[i] > 1e5 || r[i] < 0) {
            return 1;
        }
    }
    // запис початкових значень min та max
    int max = r[0], min = r[0];
    int maxIndex = 0, minIndex = 0;

    for (int i = 1; i < N; i++) {
        if (r[i] > max) {
            max = r[i];
            maxIndex = i;
        }
        if (r[i] < min) {
            min = r[i];
            minIndex = i;
        }
    }
    // нові значення max та min
    int newMax = (maxIndex == 0) ? r[1] : r[0];
    int newMin = (minIndex == 0) ? r[1] : r[0];
    // цикл для знаходження новий значень max та min
    for (int i = 0; i < N; i++) {
        if (i != maxIndex) {
            if (r[i] > newMax) {
                newMax = r[i];
            }
        }
        if (i != minIndex) {
            if (r[i] < newMin) {
                newMin = r[i];
            }
        }
    }
    // різниця між парами елемментів
    int rMax = newMax - min;
    int rMin = max - newMin;

    if (rMax < rMin) {
        cout << rMax << endl;
    } else {
        cout << rMin << endl;
    }

    return 0;
}
```

Завдання №7:

Algotester Lab 3v1

Lab-3v1.cpp > main()

```cpp
#include <iostream>
#include <string> // для string, length, to_string

using namespace std;

int main() {

    string s ,sc; // створення string для запису рядка
    cin >> s;
    if(s.length() < 1 || s.length() > 1e5) return 1; // перевірка меж

    int count = 1;// лічильник для запису кількості повторюваних символів

    for(int i = 0; i < s.length(); i++)
    {
        if(i + 1 < s.length() && s[i] == s[i + 1])
        {
            count++;
        }else{
            sc += s[i]; // додавання символів
            if(count == 1) continue;
            sc += to_string(count);  // запис числа як символ
            count = 1;
        }
    }

    cout << sc << endl; // вивід рядка

    return 0;
}
```

Завдання №8:

Algotester Lab 5v3

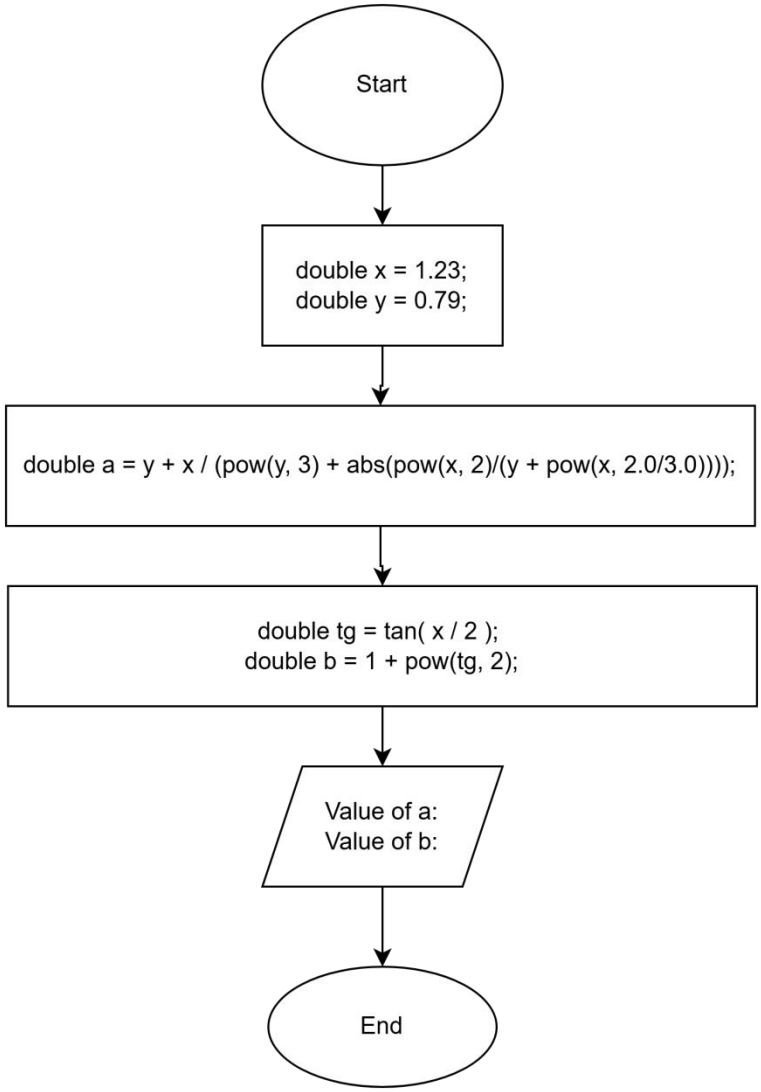
Lab-5v3.cpp > ...

```cpp
#include <iostream>
#include <vector>
#include <cstdlib> // Для abs()

using namespace std;

int main() {

    int N, M; // ініціалізація змінних типу int для умови
    cin >> N >> M;


    if (N < 1 || N > 1000 || M < 1 || M > 1000) return 1; // перевірка меж

    int x, y;
    cin >> x >> y;

    if (x < 1 || x > N || y < 1 || y > M) return 1; // перевірка меж
    // змінюю індиксація щоб починати з початку координат
    x -= 1;
    y -= 1;
    // змінна для відстані
    int max_distance = 0;
    // цикл для ініціалізації кожної точки або координати
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            int distance = abs(i - x) + abs(j - y); // використав манхетенську форму запису відстані
            max_distance = (max_distance < distance) ? distance : max_distance; // тернарний оператор для зміни відстані
        }
    }
    // цикл для виведення значень
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            int distance = abs(i - x) + abs(j - y); // знову обчислюю довжину
            cout << max_distance - distance << " "; // виводжу різницю як значення висоти гори
        }
        cout << endl;
    }


    return 0;
}
```

Дизайн та планована оцінка часу виконання завдань:

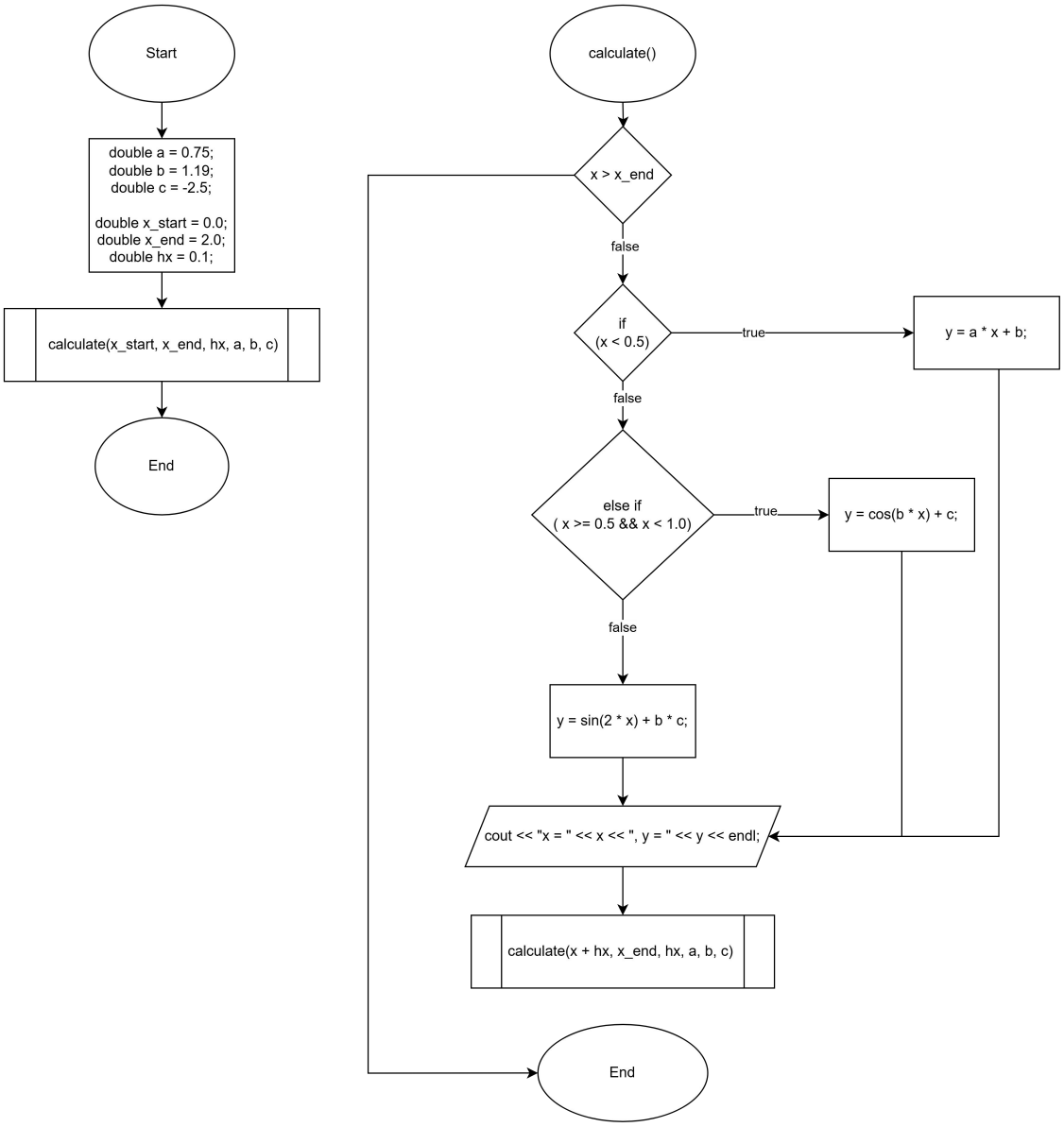
Завдання №1 – VNS Practice Work Task 1 v25

Планований час виконання 10хвилин.



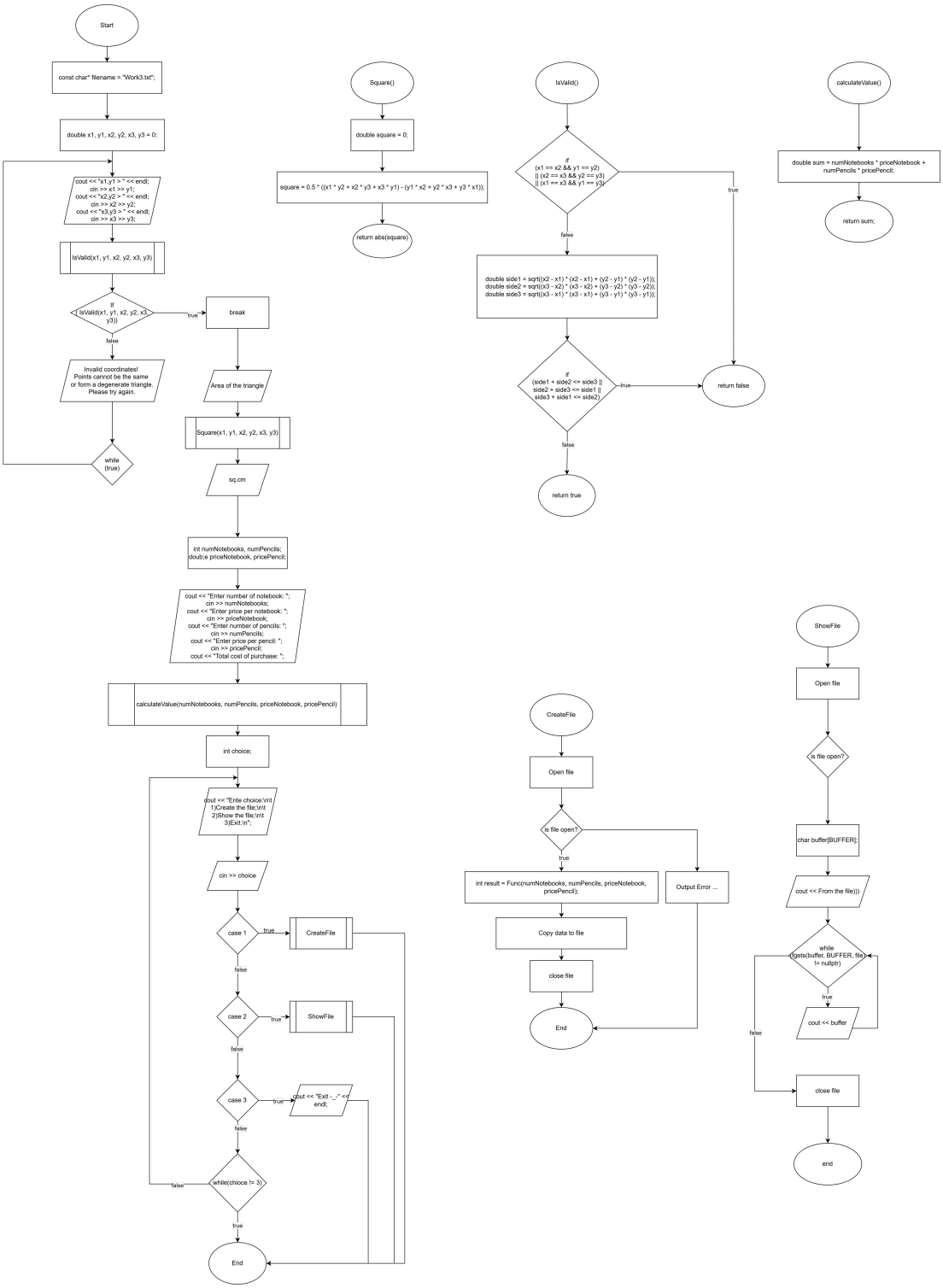
Завдання №2 – VNS Practice Work Task 2 v1

Планований час виконання 15хвилин.



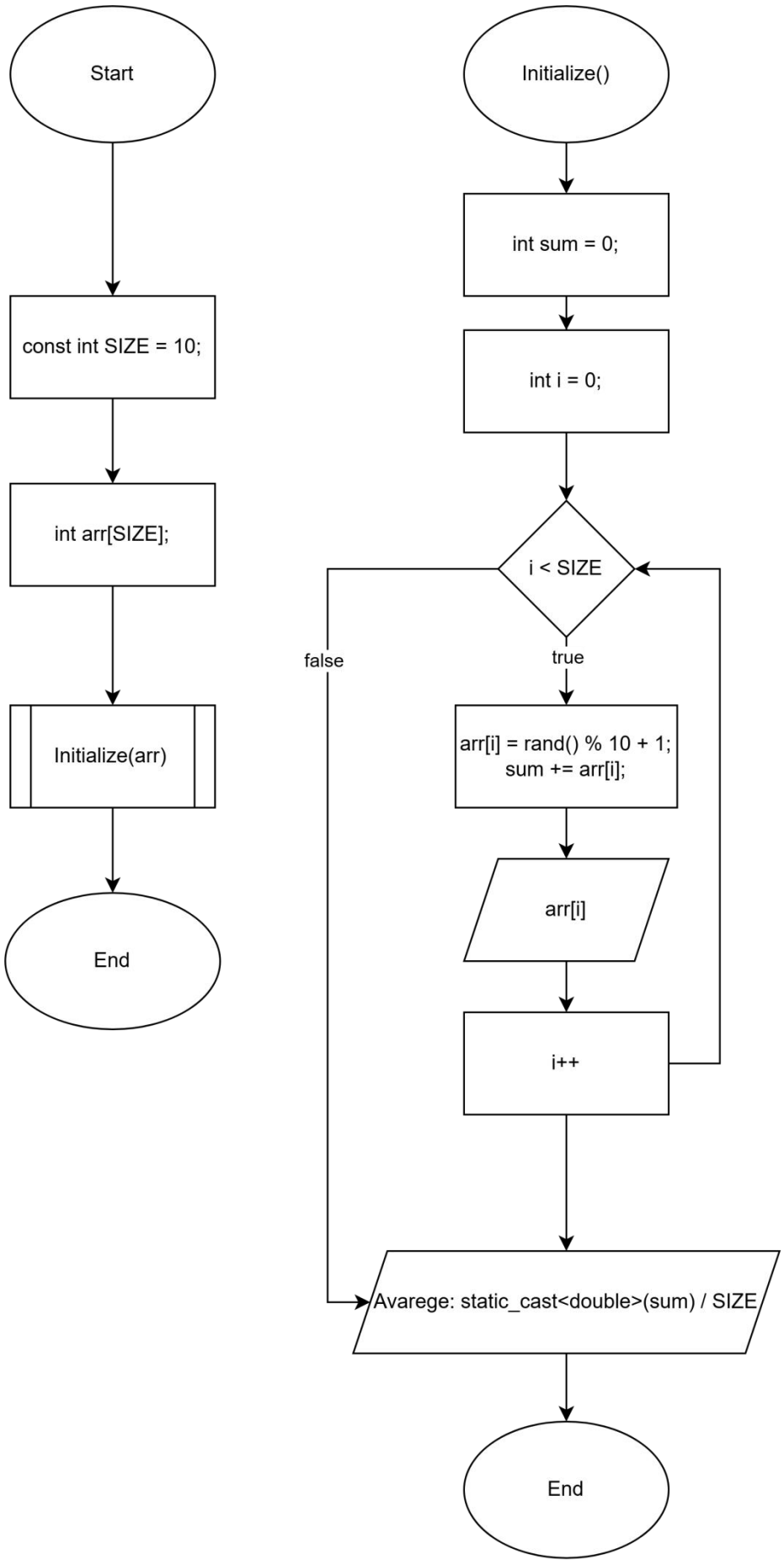
Завдання №3 – VNS Practice Work Task 3 v19

Планований час виконання 1 година.



Завдання №4 – VNS Practice Work Task 4 v16

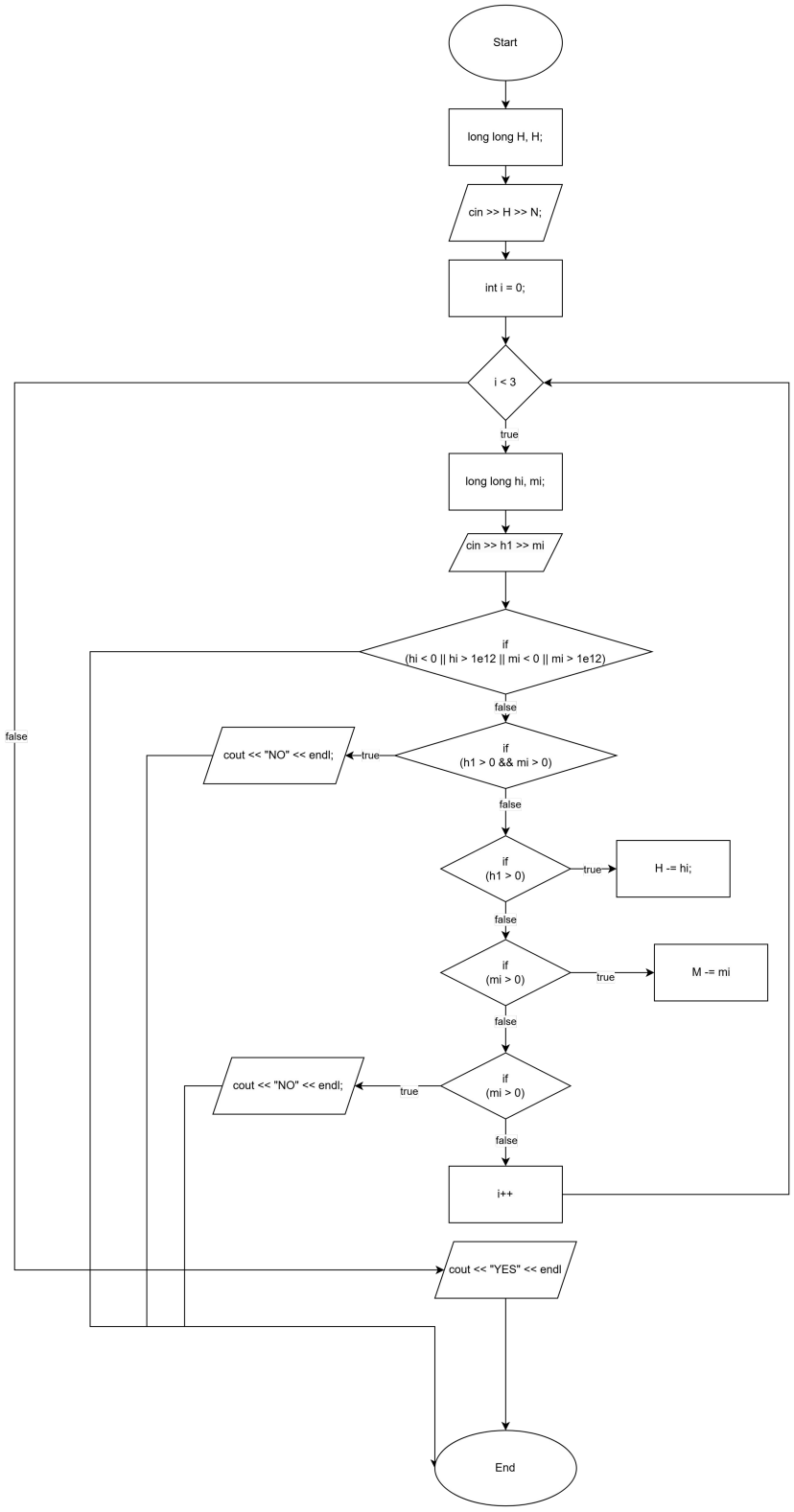
Планований час виконання 15хвилин.


Start
const int SIZE = 10;
int arr[SIZE];
Initialize(arr)
End
Initialize()
int sum = 0;
int i = 0;
i < SIZE
false
true
arr[i] = rand() % 10 + 1;
sum += arr[i];
arr[i]
i++
Avarege: static_cast<double>(sum) / SIZE
End

Завдання №5:

Algotester Lab 1v1

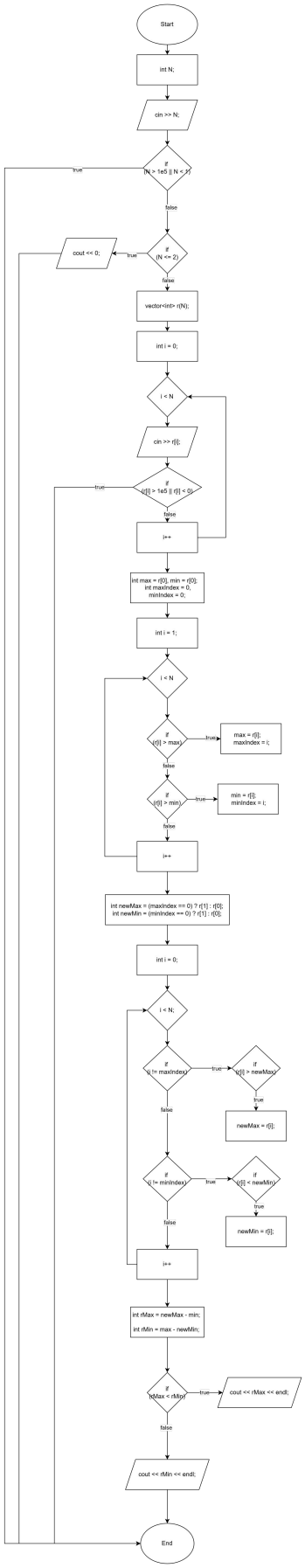
Планований час виконання 15хвилин.



Завдання №6:

Algotester Lab 2v1

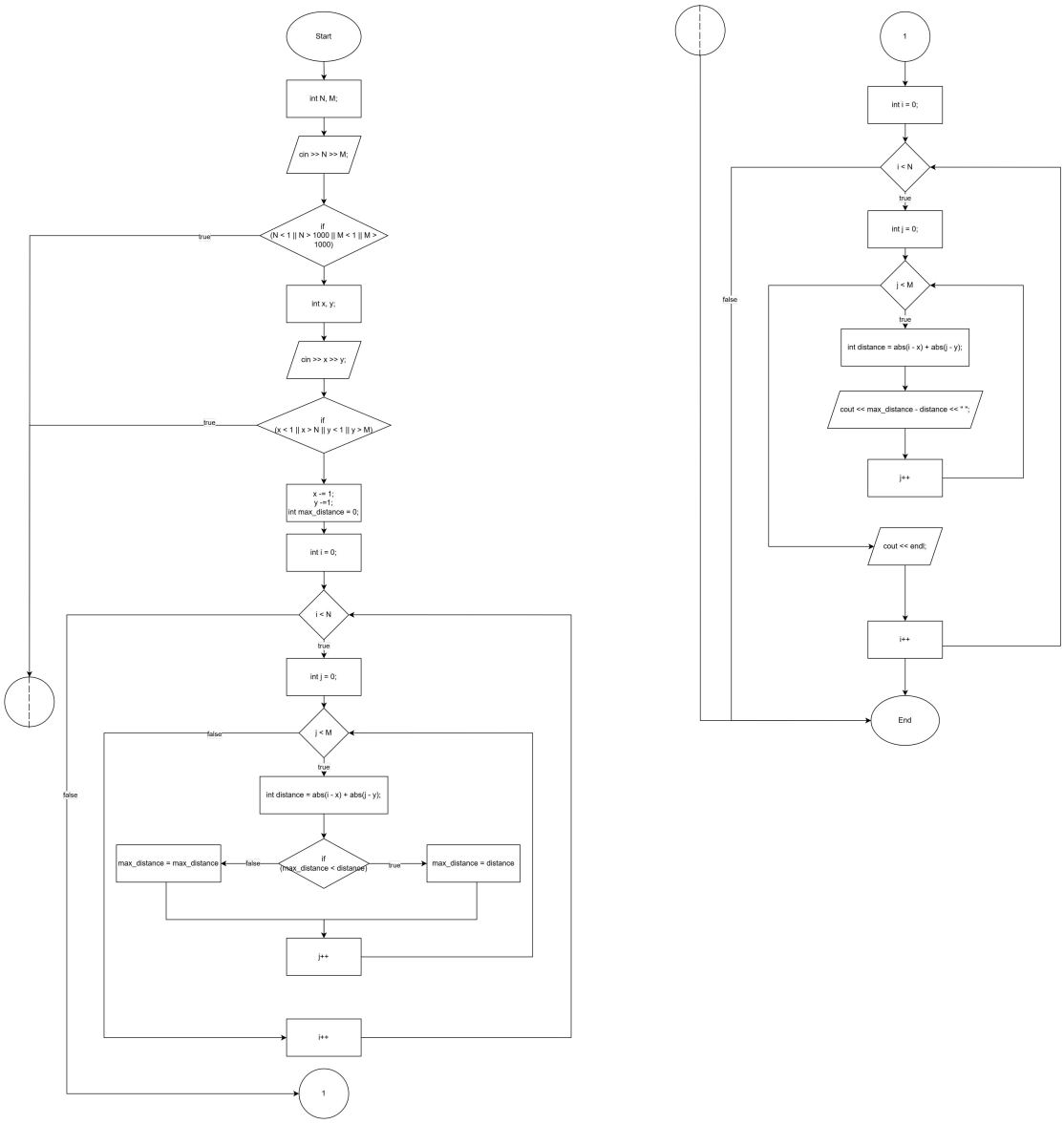
Планований час виконання 40хвилин.



Завдання №7

Algotester Lab 5v3

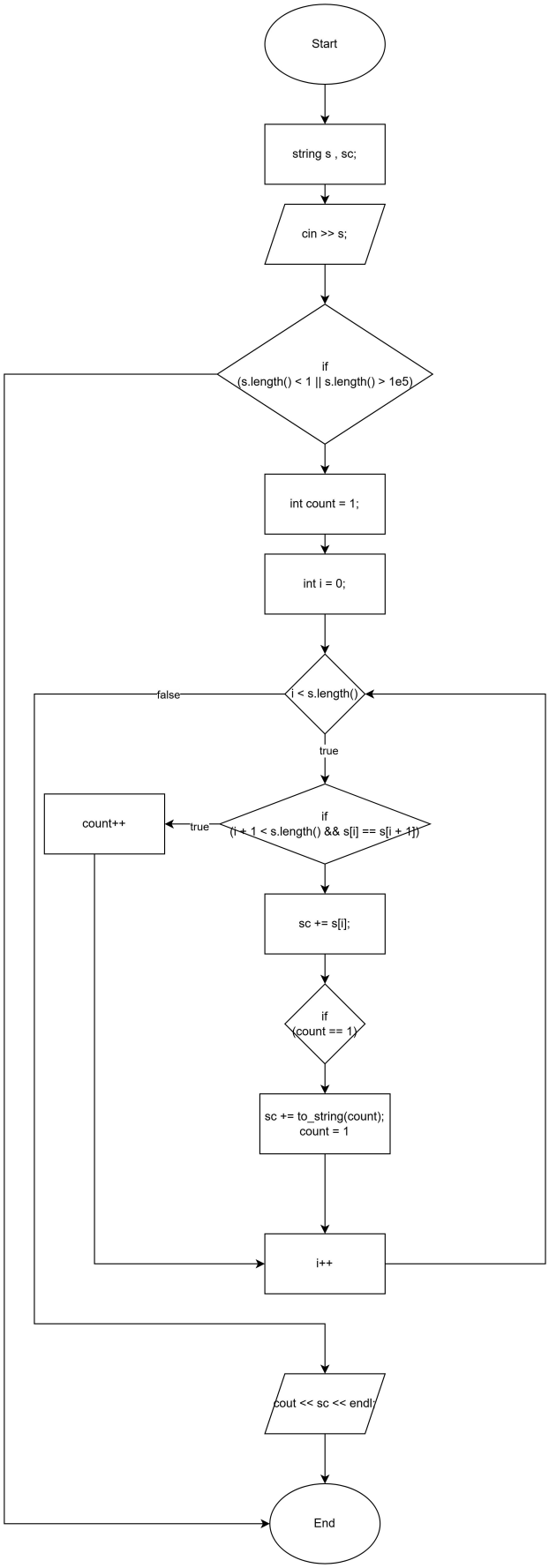
Планований час виконання 2години.



Завдання №8

Algotester Lab 3v3

Планований час виконання 1години.



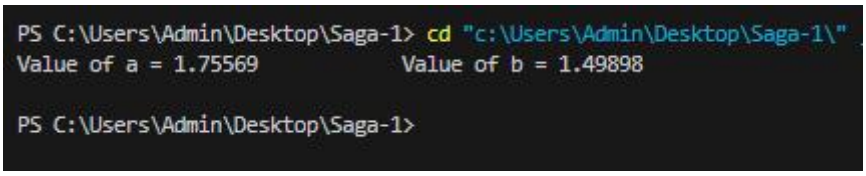
Код програми з посиланням на зовнішні ресурси

[Завдання №1 – VNS Practice Work Task 1 v25]
[Завдання №2 – VNS Practice Work Task 2 v1]
[Завдання №3 – VNS Practice Work Task 3 v19]
[Завдання №4 – VNS Practice Work Task 4 v16]
[Завдання №5 Algotester Lab 1v1]
[Завдання №6 Algotester Lab 2v1]
[Завдання №7 Algotester Lab 5v3]
[Завдання №8 Algotester Lab 3v3]

Результати виконаних завдань, тестування та фактично затрачений час

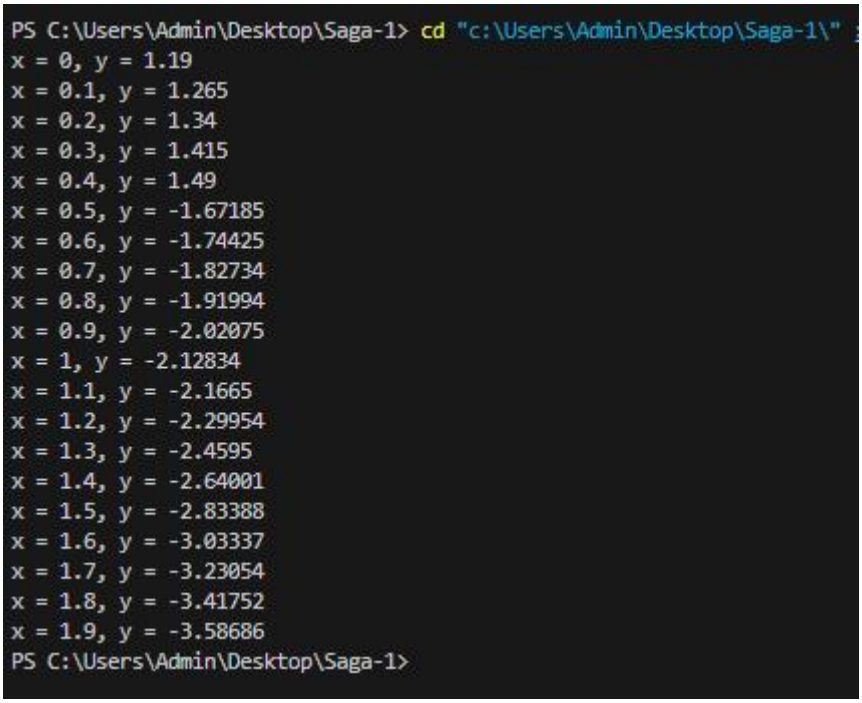
Завдання №1 – VNS Practice Work Task 1 v25

~10 хвилин



Завдання №2 – VNS Practice Work Task 2 v1

~15 хвилин



Завдання №3 – VNS Practice Work Task 3 v19

~1 година

```
PS C:\Users\Admin\Desktop\Saga-1> cd "c:\Users\Admin\Desktop\Saga-1\" ; if ($?) {
x1,y1 >
2 2
x2,y2 >
3 7
x3,y3 >
7 9
Area of the triangle: 9.00 sq.cm.

Enter number of notebooks: 10
Enter price per notebook: 10
Enter number of pencils: 10
Enter price per pencil: 10
Total cost of purchase: 200.00 UAH.
Ente choice:
        1)Create the file;
        2)Show the file;
        3)Exit.
1
Result has been written to the file: Work3.txt
Ente choice:
        1)Create the file;
        2)Show the file;
        3)Exit.
2
From the file)))
The total value is: 200
Ente choice:
        1)Create the file;
        2)Show the file;
        3)Exit.
3
Exit -_-
PS C:\Users\Admin\Desktop\Saga-1>
```

Завдання №4 – VNS Practice Work Task 4 v16

~25 хвилин

```
PS C:\Users\Admin\Desktop\Saga-1> cd "
7 1 9 6 9 2 3 3 3 3
Average: 4.6
PS C:\Users\Admin\Desktop\Saga-1>
```

Завдання №5 Algotester Lab 1v1

~10 хвилин

Created	Compiler	Result	Time (sec.)	Memory (MiB)
3 days ago	C++ 23	Accepted	0.003	1.207

Завдання №6 Algotester Lab 2v1

~15 хвилин

Created	Compiler	Result	Time (sec.)	Memory (MiB)
a day ago	C++ 23	Accepted	0.003	1.414

Завдання №7 Algotester Lab 5v3

~2 години хвилин

Created	Compiler	Result	Time (sec.)	Memory (MiB)
2 days ago	C++ 23	Accepted	0.099	4.531

Завдання №8 Algotester Lab 3v3
~30 хвилин

Created	Compiler	Result	Time (sec.)	Memory (MiB)
2 days ago	C++ 23	Accepted	0.003	1.570

Висновки:

Я покращив свої навички, практикуючись на написанні програм необхідних для розрахункової, а також структурував весь пройдений матеріал.

Посилання на pull request