Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт 4**

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

***з розділу***: «Епік 4: Прості структури даних. Одновимірні масиви. Двовимірні Масиви. Алгоритми обробки»

***Виконав:***

студент групи ШІ-13

Юнко Дмитро Богданович

# **Тема роботи:**

# Робота із простими структурами даних та алгоритмами обробки. Робота з одновимірними та двовимірними масивами.

# **Мета роботи:**

# Мета полягає в освоєнні простих структур даних та алгритмів обробки масивів з використанням відповідних функцій. Навчитися передавати масиви як параметри функцій

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

- Тема №1: Прості структури даних

- Тема №2: Одновимірні масиви

- Тема №3: Двовимірні масиви

- Тема №4: Масиви як параметри функцій

- Тема №5: Стандартні операції обробки масивів

- Тема №6: Базові алгоритми сортування масивів

1. Індивідуальний план опрацювання теорії:

Тема №1: Прості структури даних

* + Джерела Інформації
    - Стаття: [Data Structures Tutorial - GeeksforGeeks](https://www.geeksforgeeks.org/data-structures/)
    - Курс: Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Визначення та класифікація структур даних
    - Основні операції на структурах даних
    - Різні лінійні структури, такі як масив, рядок, зв’язаний список, стек, черга.
    - Реалізація популярних типів структур даних та які їх переваги та недоліки.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 28.11.2023
  + Звершення опрацювання теми: 28.11.2023
* Тема №2: Одновимірні масиви
  + Джерела Інформації:
    - Стаття: [C++ One-Dimensional Array (codescracker.com)](https://codescracker.com/cpp/cpp-one-dimensional-arrays.htm)
    - Стаття: [C++ Arrays - GeeksforGeeks](https://www.geeksforgeeks.org/cpp-arrays/)
    - Курс: Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - що таке одновимірний масив, як він зберігається в пам’яті комп’ютера, та як його оголосити, вказавши тип даних, назву та розмір масиву
    - як присвоїти значення елементам одновимірного масиву під час оголошення або після нього
    - як отримати доступ до елементів одновимірного масиву, використовуючи їх індекси
    - як отримати розмір масиву
  + Статус: Ознайомлений
  + Початок опрацювання теми: 28.11.2023
  + Звершення опрацювання теми: 28.11.2023
* Тема №3: Двовимірні масиви.
  + Джерела Інформації:
    - Стаття: [Two Dimensional Array in C++ | DigitalOcean](https://www.digitalocean.com/community/tutorials/two-dimensional-array-in-c-plus-plus)
    - Стаття: [C++ Multidimensional Array - GeeksforGeeks](https://www.geeksforgeeks.org/cpp-multidimensional-array/)
    - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - що таке двовимірний масив, як він зберігається в пам’яті комп’ютера, та як його оголосити, вказавши тип даних, назву та розміри обох вимірів
    - Ініціалізація та доступ до елементів двовимірних масивів в C++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 28.11.2023
  + Звершення опрацювання теми: 28.11.2023
* Тема №4: Масиви як параметри функцій
  + Джерела Інформації:
    - Стаття: [C++ One-Dimensional Array (codescracker.com)](https://codescracker.com/cpp/cpp-one-dimensional-arrays.htm)
    - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Передача масивів
    - Передача багатовимірних масивів
  + Статус: Ознайомлений
  + Початок опрацювання теми: 29.11.2023
  + Звершення опрацювання теми: 29.11.2023
* Тема №5: Стандартні операції обробки масивів
  + Джерела Інформації:
    - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Операції злиття, розділення,
    - Виведення, вставка, видалення, пошук, сортування, і оновлення елементів масиву
  + Статус: Ознайомлений
  + Початок опрацювання теми: 29.11.2023
  + Звершення опрацювання теми: 29.11.2023

Тема №6: Базові алгоритми сортування масивів

* + Джерела Інформації:
    - Курс: [Sorting Algorithms - GeeksforGeeks](https://www.geeksforgeeks.org/sorting-algorithms/)
    - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - які види сортування існують
    - як використовувати різні алгоритми сортування, такі як сортування бульбашкою, сортування вибором, сортування вставками, сортування злиттям, швидке сортування
    - як аналізувати складність різних алгоритмів сортування, таких як часова складність, просторова складність, стабільність, і адаптивність
  + Статус: Ознайомлений
  + Початок опрацювання теми: 29.11.2023
  + Звершення опрацювання теми: 29.11.2023

# **Виконання роботи:**

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

**Завдання №1 Опрацювання теорії**

* Деталі завдання: опрацювання різноманітних матеріалів, аналіз та дослідження відео, статей, книг на задані теми. Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Проектування за допомогою Draw.io(складання блок-схем)**

* Деталі завдання:

Побудова блок-схем до завдань 3-9.

#### Завдання №3 VNS Lab 4 - Task 1

* Варіант завдання: **20**
* Деталі завдання: 20.

1) Реалізувати з використанням масиву двонаправлене кільце (перегляд

можливий в обидва боки, від останнього елемента можна перейти до

першого).

2) Роздрукувати отриманий масив, починаючи з К-ого елемента і до К-1 (по

кільцю вліво).

3) Додати в кільце після елементів з індексами кратними 5 елементи, які

дорівнюють 0.

4) Роздрукувати отриманий масив, починаючи з К-ого елемента (і до К+1 по

кільцю вправо).

#### Завдання №4 VNS Lab 5 - Task 1

* Варіант завдання: **20**
* Деталі завдання: Знайти мінімальний з неповторюваних елементів двовимірного масиву.

#### Завдання №5 Algotester Lab 2

Варіант завдання: 1

Деталі завдання: : Є дорога, представлена як масив чисел. Завдання: знайти мінімальну можливу втому після видалення одного числа. Втому визначає різниця між максимальним та мінімальним елементом.

Можна видалити одне число з масиву, щоб мінімізувати втому.

* Важливі деталі: Використання алгоритму bubble sort для сортування масиву дорога

#### Завдання №6 Algotester Lab 3

* Варіант завдання: 3
* Деталі завдання: зробити компресію стрічки, тобто якщо якась буква йде більше одного разу підряд у стрічці замінити її на букву + кількість входжень підряд.

#### Завдання №7 Class Practice

* Варіант завдання: **-**
* Деталі завдання: Реалізувати програму, яка перевіряє, чи дане слово чи число є паліндромом за допомогою рекурсії.
* Важливі деталі: використати рекурсію для перевірки чи слово/набір чисел є паліндромом. Використати перевантаження функції для того

#### Завдання №8 Self-Practice | Class Practice using do-while

* Деталі завдання: Вирішення Class Practice використовуючи do-while

#### Завдання №9 Self-Practice | Algotester Lab 2 using library vector

* Деталі завдання:  вирішення Algotester Lab 2 використовуючи бібліотеку vector

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

#### Програма № 3 VNS Lab 4 - Task 1

* Блок-схема
* Планований час на реалізацію: 15хв

використання бібліотеки cmath

#### Програма №4 VNS Lab 5 - Task 1

* Планований час на реалізацію: 60хв

#### Програма №5 Algotester Lab 2

* Блок-схема

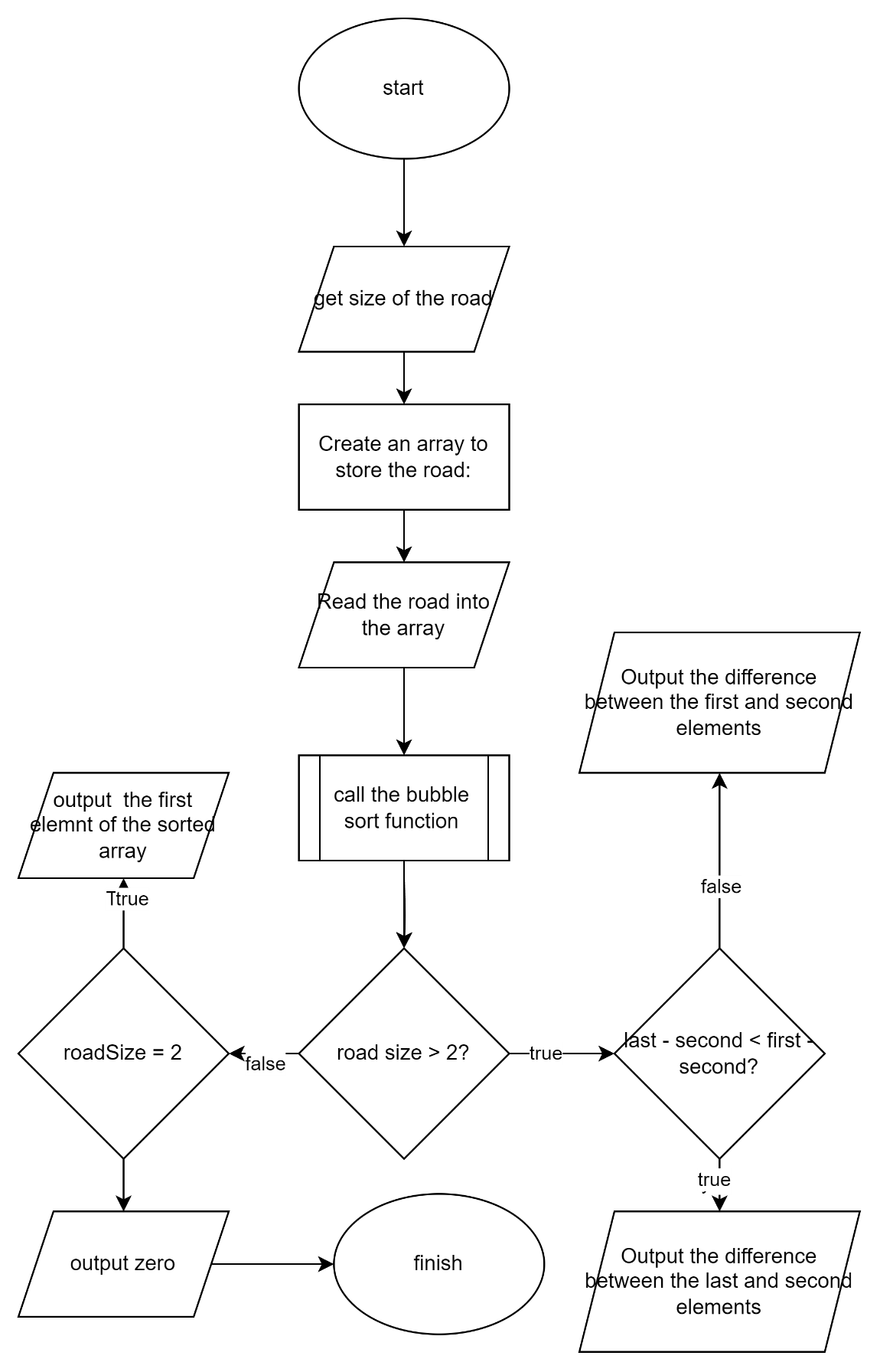


Рисунок 1 Algotester lab 2

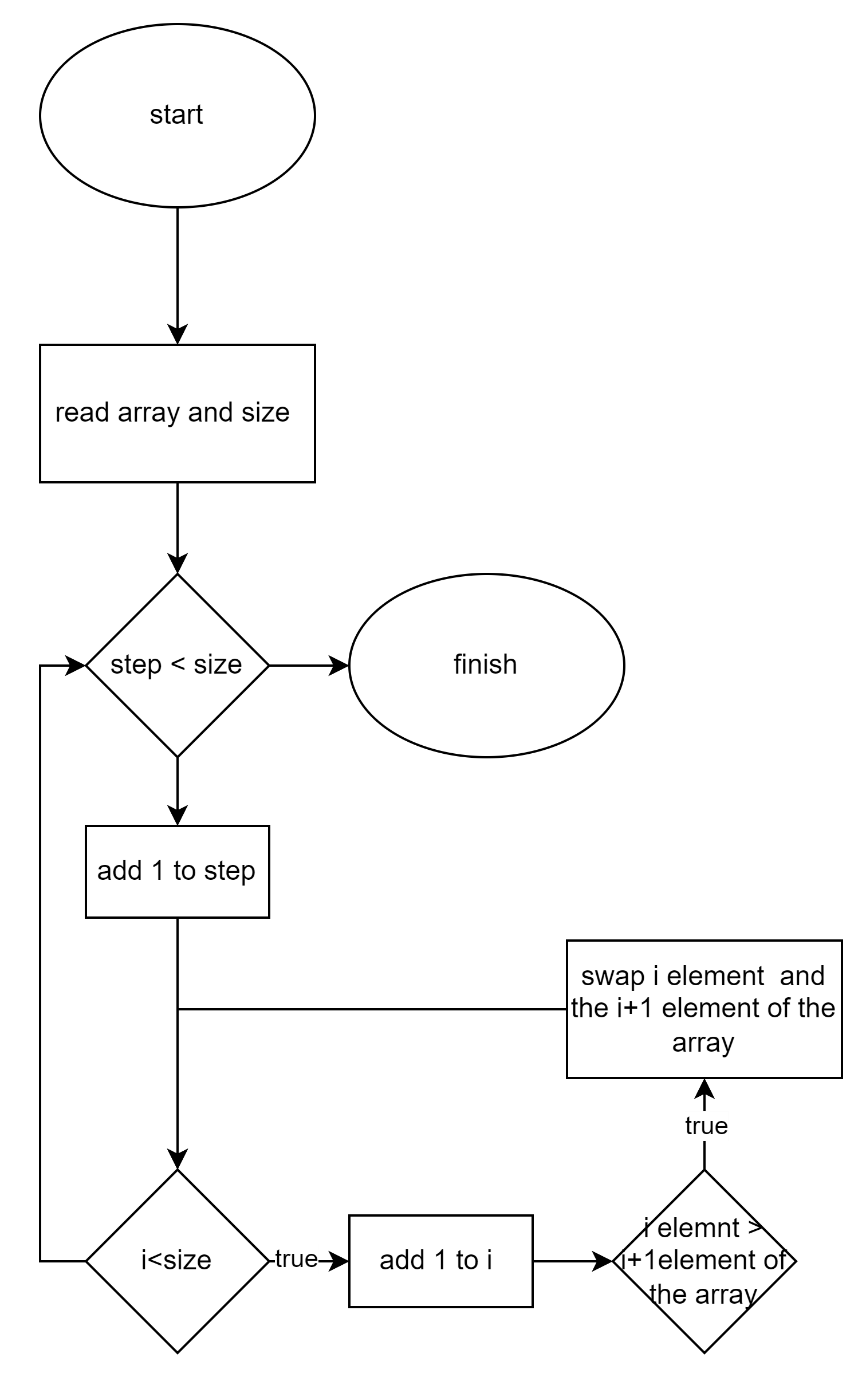


Рисунок 2 Bubble sort function

* Планований час на реалізацію 15хв

#### Програма №6 Algotester Lab 3

* Планований час на реалізацію 45 хв

#### Програма №6 Algotester Lab 3

* Планований час на реалізацію 30 хв

#### Програма №8 Self Practice Algotester lab 2 using merge sort:

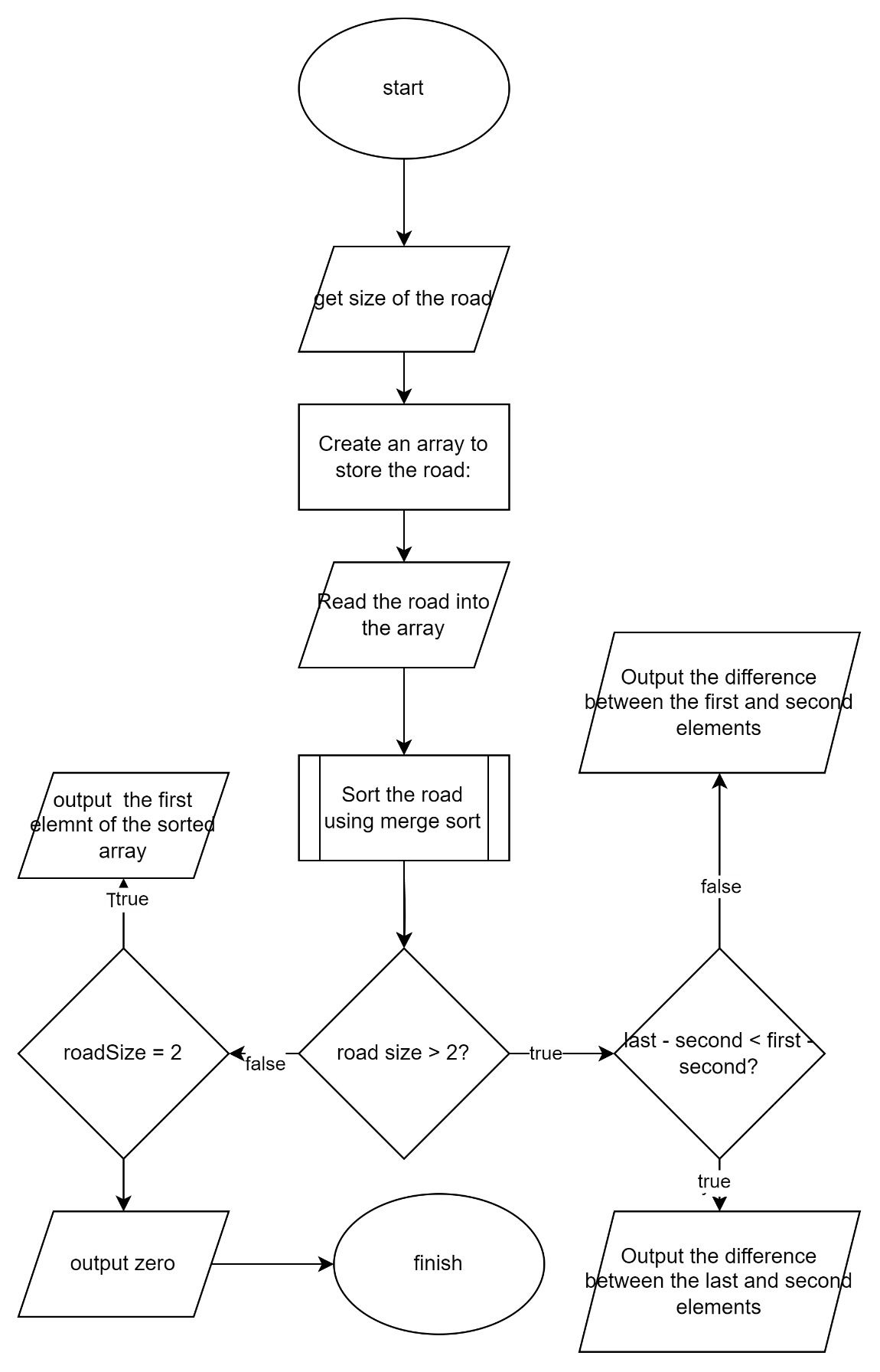


Рисунок 3 Algotester lab 2

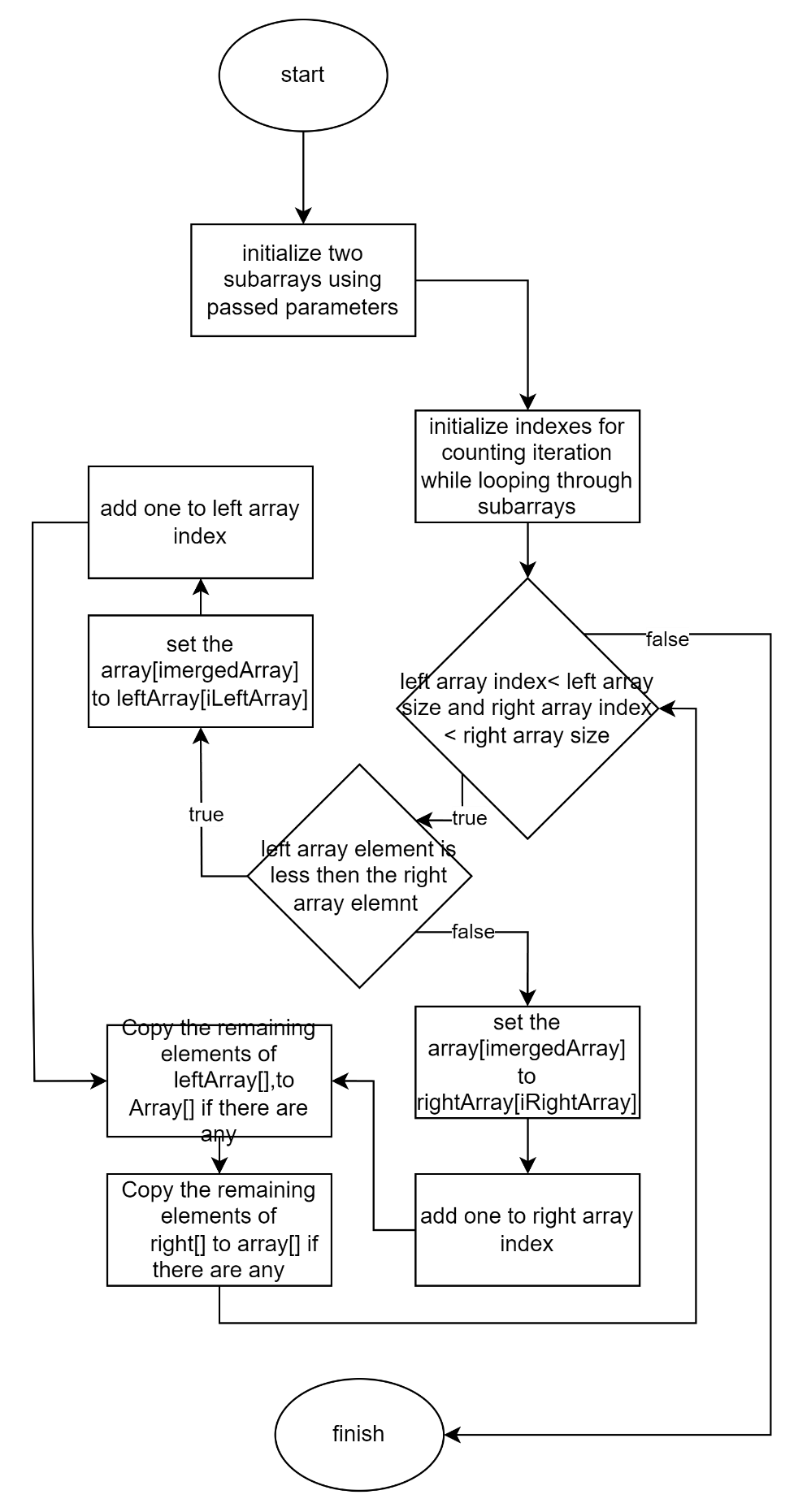


Рисунок 4 Merge function

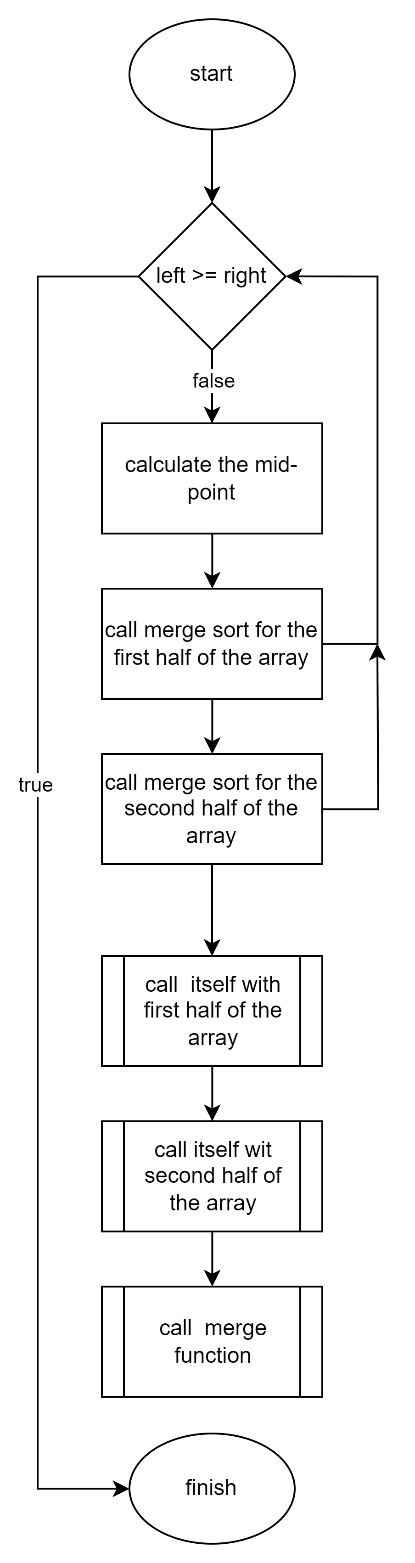


Рисунок 5 Merge sort function

* Планований час на реалізацію: 80хв

#### Програма №6 Algotester Lab 2 using selection sort

* Планований час на реалізацію 15 хв

## **4. Код програм з посиланням на зовнішні ресурси:** [https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/230/files#diff-097297c805c8250774272aa6616b4bbbdd5bd20fde5f2b0cf8b246d98169163d](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/230/files%23diff-097297c805c8250774272aa6616b4bbbdd5bd20fde5f2b0cf8b246d98169163d)

#include <iostream>

using namespace std;

struct Node

    {

        int value;

        Node\* next;

        Node\* previous;

    };

    class LinkedList {

        private:

        Node\* head;

        Node\* end;

        public:

        void insert(int value) {

                Node\* newNode = new Node;

                newNode->value = value;

                newNode->next = nullptr;

                 if(head==nullptr){

                    head = newNode;

                    end = newNode;

                    newNode->next = newNode;

                    newNode->previous = newNode;

                } else {

                    newNode->next = head;

                    newNode->previous = end;

                    end->next = newNode;

                    head->previous = newNode;

                    end = newNode;

                }

            }

            LinkedList(int size) {

                head = nullptr;

                end = nullptr;

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

            {

                insert(i+1);

            }

            }

            ~LinkedList() {

                Node\* temp;

                while(head != nullptr){

                    temp= head;

                    head = head->next;

                    delete temp;

                }

            }

            void insertZeroAfterFive() {

                Node\* temp = head;

                int index = 0;

                do {

                    if (index % 5 == 0) {

                        Node\* newNode = new Node;

                        newNode->value = 0;

                        newNode->next = temp->next;

                        newNode->previous = temp;

                        temp->next->previous = newNode;

                        temp->next = newNode;

                        temp = newNode->next;

                        index++;

                    } else {

                        temp = temp->next;

                    }

                    index++;

                } while (temp != head);

            }

            void displayFromKToKMinusOne(int K) {

                Node\* temp = head;

                for (int i = 0; i < K; i++) {

                    temp = temp->next;

                }

                for (int i = 0; i < 2; i++) {

                    cout << temp->value << " ";

                    temp = temp->previous;

                }

                cout << endl;

            }

            void displayFromKToKPlusOne(int K) {

                Node\* temp = head;

                for (int i = 0; i < K; i++) {

                    temp = temp->next;

                }

                for (int i = 0; i < 2; i++) {

                    cout << temp->value << " ";

                    temp = temp->next;

                }

                cout << endl;

            }

             void display() {

            Node\* temp = head;

            while (temp != end) {

                cout << temp->value << " ";

                temp = temp->next;

            }

            // cout << temp->value<<" "<<temp -> previous->value;

            cout << endl;

        }

    };

int main(int argc, char const \*argv[])

{

    LinkedList List{10};

    List.display();

    List.displayFromKToKPlusOne(5);

    List.displayFromKToKMinusOne(5);

    List.insertZeroAfterFive();

    List.display();

    return 0;

}

###### Код до програми № 3 VNS Lab 4 - Task 1

#include <iostream>

#include <vector>

using namespace std;

    int jlength{0};

    int klength{0};

int displayArr ( int \*arr) {

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

    {   cout<< "{";

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

        {

            cout<<" "<<\*(arr + j \* klength + k);

        }

        cout <<"}"<<endl;

    }

}

int findMin (vector<int>& uniqAr){

    int min{0};

    for (vector<int>::iterator it = uniqAr.begin(); it != uniqAr.end(); it++)

    {

        if (it==uniqAr.begin())

        {

            min=\*it;

        }else if(\*it <min){

            min=\*it;

        }

    }

    cout<< min<<endl;

}

    vector<int> findUnique (int \*arr) {

        vector<int> uniqAr;

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

    {

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

        {

            int curr = \*(arr + j \* klength + k);

            bool isUnique = true;

            for (int i = 0; i < jlength\*klength; i++)

            {

                if(i== j \* klength + k)

                    {

                        continue;

                    }

                   if(curr == \*(arr + i) )

                   {

                    isUnique=false;

                    break;

                    }

            }

            if(isUnique)

                {

                    uniqAr.push\_back(curr);

                }

        }

    }

    findMin(uniqAr);

    return uniqAr;

    }

int main(int argc, char const \*argv[])

{

    cout<< "Size of 2d array:";

    cin>> jlength;

    cin>> klength;

    int array[jlength \* klength];

    for (size\_t j = 0; j < jlength; j++)

    {

        cout<< "enter "<<j+1 <<" array";

        for (size\_t k = 0; k < klength; k++)

        {

            cin >> \*(array + j \* klength + k);

        }

        cout<<endl;

    }

    cout<<endl;

    displayArr(array);

    findUnique(array);

    return 0;

}

###### Код до програми № 4 VNS Lab 5 - Task 1 [https://github.com/artificial-intelligence department/ai\_programming\_playground/pull/494](https://github.com/artificial-intelligence%20department/ai_programming_playground/pull/494)

#include <iostream>

using namespace std;

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void sort(int array[], int size) {

for (int step = 0; step < size - 1; step++) {

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

if (array[i] > array[i+1]){

swap(&array[i], &array[i+1]);

}

}

}

}

int main()

{

int roadSize {0};

cin >> roadSize;

int road[roadSize] {0};

int min {0};

for (size\_t i = 0; i < roadSize; i++)

{

cin >> road[i];

}

sort(road, roadSize);

if(roadSize > 2){

if(road[roadSize-1] - road[1] < road[roadSize-2] - road[0]){

cout << road[roadSize-1] - road[1] <<endl;

}else{

cout << road[roadSize-2] - road[0] <<endl;

}

}else if (roadSize == 2 ){

cout << road[0];

}else{

cout << 0;

}

return 0;

}

###### Код до програми № 5 Algotester Lab 2

#include <string>

#include <iostream>

using namespace std;

int main(int argc, char const \*argv[])

{

string s {""};

string compressedS {""};

cin >> s;

int repNum {1};

for (const char\* c = s.c\_str(); \*c; ++c)

{

if(\*c == \*(c+1)){

repNum ++;

}else{

char prevC = \*(c);

compressedS.append(1,prevC);

if (repNum>1)

{

compressedS.append(to\_string(repNum));

}

repNum = 1;

}

}

cout << compressedS;

return 0;

}

###### Код до програми № 6 Algotester Lab 3

#include <iostream>

#include <string>

#include <vector>

#include <sstream>

using namespace std;

bool isPalindrome (string &n){

static int i {0};

if(n[i]== n[n.length()-(i+1)] && i == n.length()-1 ){

return true;

}else if( i == n.length()-1){

return false;

}

i++;

isPalindrome(n);

}

bool isPalindrome (vector<int> &n){

static int i {0};

if(n[i]== n[n.size()-(i+1)] && i == n.size()-1 ){

return true;

}else if( i == n.size()-1){

return false;

}

i++;

isPalindrome(n);

}

int main(int argc, const char\*\* argv) {

string word;

getline(cin,word);

string numbsLine;

vector<int> numbs;

int numb {0};

getline(cin, numbsLine);

stringstream nsL(numbsLine);

while (nsL >> numb){

numbs.push\_back(numb);

}

cout<< isPalindrome(word);

cout<< isPalindrome(numbs);

return 0;

}

###### Код до програми № 7 Class Practice

#include <iostream>

using namespace std;

void merge(int array[],int left,int right, int mid ) {

  int leftArraySize = mid - left + 1;

  int rightArraySize = right - mid;

  int leftArray [leftArraySize] {0};

  int rightArray [rightArraySize] {0};

 for (size\_t i = 0; i < leftArraySize; i++){

      leftArray[i] = array[left + i];

    }

    for (size\_t j = 0; j < rightArraySize; j++){

        rightArray[j] = array[mid + 1 + j];

    }

    int iLeftArray = 0;

    int iRightArray = 0;

    int imergedArray = left;

    while (iLeftArray < leftArraySize && iRightArray < rightArraySize){

      if (leftArray[iLeftArray]<= rightArray[iRightArray]) {

            array[imergedArray] = leftArray[iLeftArray];

            iLeftArray++;

        }

        else {

            array[imergedArray]= rightArray[iRightArray];

            iRightArray++;

        }

        imergedArray++;

    }

    while (iLeftArray < leftArraySize) {

        array[imergedArray]= leftArray[iLeftArray];

        iLeftArray++;

        imergedArray++;

    }

    while (iRightArray < rightArraySize) {

        array[imergedArray]= rightArray[iRightArray];

        iRightArray++;

        imergedArray++;

    }

}

void mergeSort(int array[], int left, int right)

{

    if (left >= right)

        return;

    int mid = left + (right - left) / 2;

    mergeSort(array, left, mid);

    mergeSort(array, mid + 1, right);

    merge(array, left, right, mid);

}

int main()

{

    int roadSize {0};

    cin >> roadSize;

    int road[roadSize] {0};

    int min {0};

    for (size\_t i = 0; i < roadSize; i++)

    {

        cin >> road[i];

    }

    mergeSort(road, 0, roadSize-1);

    if(roadSize > 2){

      if(road[roadSize-1] - road[1] < road[roadSize-2] - road[0]){

        cout << road[roadSize-1] - road[1] <<endl;

      }else{

        cout << road[roadSize-2] - road[0] <<endl;

      }

    }else if (roadSize == 2 ){

      cout << road[0];

    }else{

      cout << 0;

    }

    return 0;

}

###### Код до програми № 8 Self-Practice Algotester Lab 2 using merge sort

#include <iostream>

using namespace std;

void swap(int \*a, int \*b) {

  int temp = \*a;

  \*a = \*b;

  \*b = temp;

}

void sort(int array[], int size) {

  for (int step = 0; step < size - 1; step++) {

    int minId = step;

    for (int i = step + 1; i < size; i++) {

      if (array[i] < array[minId])

      minId = i;

    }

    swap(&array[minId], &array[step]);

  }

}

int main()

{

    int roadSize {0};

    cin >> roadSize;

    int road[roadSize] {0};

    int min {0};

    for (size\_t i = 0; i < roadSize; i++)

    {

        cin >> road[i];

    }

    sort(road, roadSize);

    if(roadSize > 2){

      if(road[roadSize-1] - road[1] < road[roadSize-2] - road[0]){

        cout << road[roadSize-1] - road[1] <<endl;

      }else{

        cout << road[roadSize-2] - road[0] <<endl;

      }

    }else if (roadSize == 2 ){

      cout << road[0];

    }else{

      cout << 0;

    }

    return 0;

}

###### Код до програми № 9 Self-Practice Algotester Lab 2 using seletion sort

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

#### Завдання №3 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_4\output> & .\'vns\_lab\_4\_task\_1\_variant\_20\_dmytro\_yunko.exe'

1 2 3 4 5 6 7 8 9 10

6 7

6 5

1 0 2 3 4 5 0 6 7 8 9 0 10

VNS Lab 4 - Task 1

#### Завдання №4 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\epic\_04\output> & .\'vns\_lab5\_task\_1.exe'

Size of 2d array: 3 3

enter 1 array1 2 3

enter 2 array1 2 3

enter 3 array5 4 8

{ 1 2 3}

{ 1 2 3}

{ 5 4 8}

4

PS D:\ED\University\C++\epic\_04\output> & .\'vns\_lab5\_task\_1.exe'

Size of 2d array:3 3

enter 1 array1 2 3

enter 2 array6 8 9

enter 3 array1 2 3

{ 1 2 3}

{ 6 8 9}

{ 1 2 3}

6

VNS Lab 5 - Task 1

#### Завдання №5 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\epic\_04\output> & .\'alg\_lab\_02\_bbl.exe'

5

1 2 2 4 4

2

PS D:\ED\University\C++\epic\_04\output> & .\'alg\_lab\_02\_bbl.exe'

5

1 2 2 2 4

1

Algotester Lab 2

#### Завдання №6 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\epic\_04\output> & .\'algo\_lab\_3v3.exe'

AAAABBBCQQQQ

A4B3CQ4

PS D:\ED\University\C++\epic\_04\output> & .\'algo\_lab\_3v3.exe'

AA

A2

PS D:\ED\University\C++\epic\_04\output> & .\'algo\_lab\_3v3.exe'

A

A

Algotester Lab 3

#### Завдання №7 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\epic\_04\output> & .\'practice.exe'

radar

1 2 3 2 1

11

Class Practice

#### Завдання №8 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\epic\_04\output> & .\'alg\_lab\_02\_mrg.exe'

5

1 2 2 4 4

2

PS D:\ED\University\C++\epic\_04\output> & .\'alg\_lab\_02\_mrg.exe'

5

1 2 2 2 4

1

Self-Practice Algotester Lab 2 merge sort

#### Завдання №9 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\epic\_04\output> & .\'alg\_lab\_02\_slct.exe'

5

1 2 2 4 4

2

PS D:\ED\University\C++\epic\_04\output> & .\'alg\_lab\_02\_slctl.exe'

5

1 2 2 2 4

1

Self-Practice Algotester Lab 2 selection sort

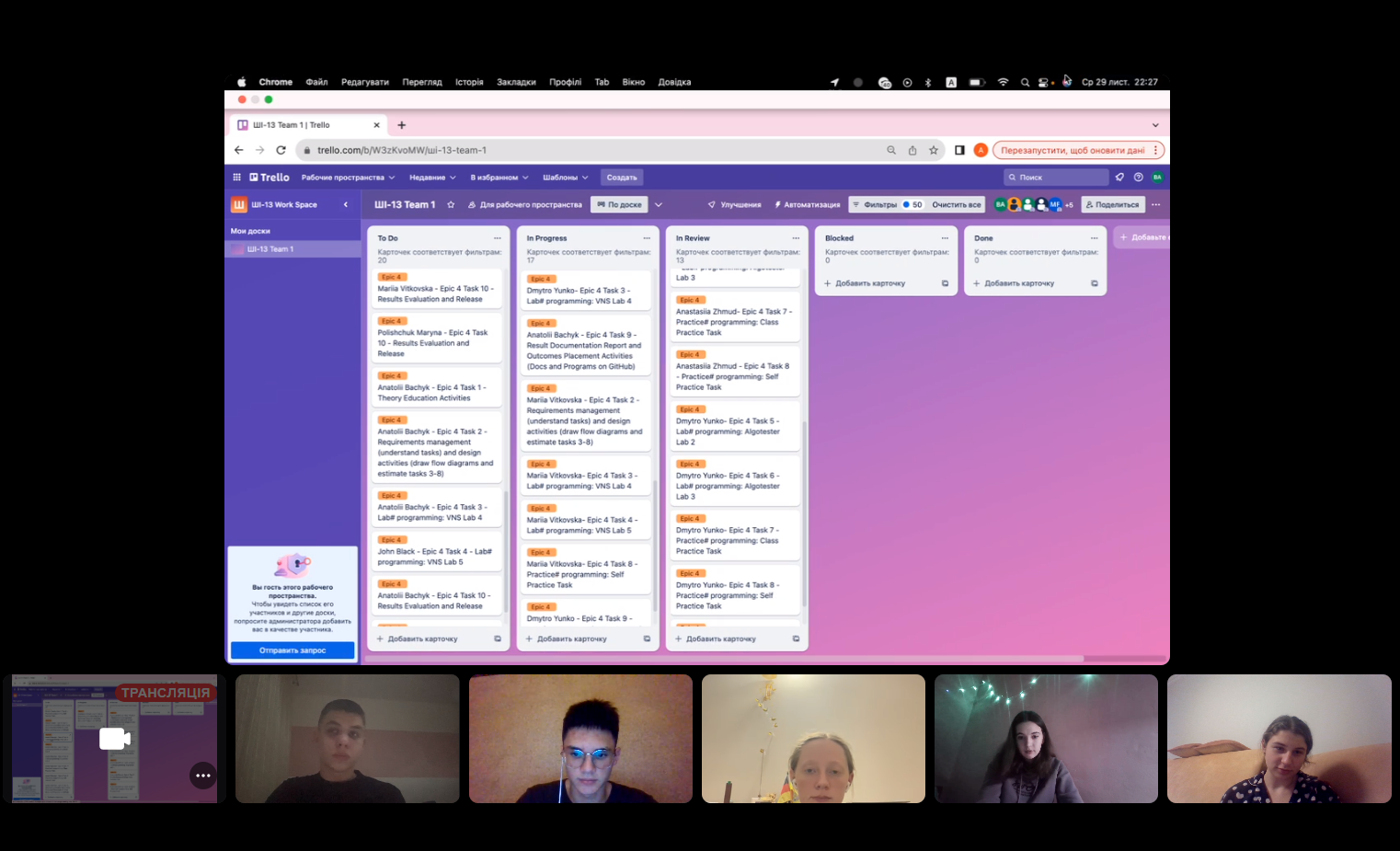


Рисунок 6 team meeting

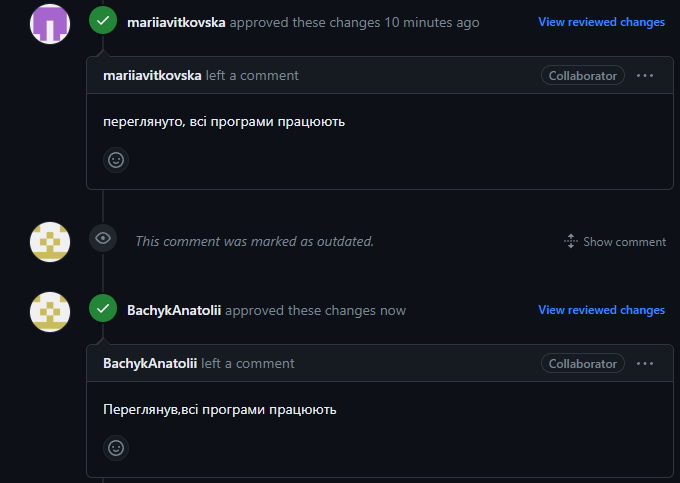


Рисунок ПР звіт

Час затрачений на виконання завдання: 2дн

# **Висновки:**

У цій лабораторній роботі я вивчив основні поняття, принципи, і методи роботи з простими структурами даних, такими як масиви, і використав їх для розв’язання різних задач програмування. Я демонстрував свої навички в оголошенні, ініціалізації, доступі, передачі, і сортуванні одновимірних і двовимірних масивів в C++. Я також аналізував складність різних алгоритмів сортування, і порівнював їх ефективність.

Мої результати показали, що структури даних, такі як масиви, є корисними інструментами для зберігання і обробки даних, які мають певну структуру і розмір. Масиви дозволяють швидко і легко отримувати доступ до елементів за їх індексами, і виконувати різні операції над ними. Масиви також можна передавати до функцій, що полегшує розбиття програми на менші та більш зрозумілі частини. Однак, масиви мають і свої недоліки, такі як статичний розмір, великий обсяг пам’яті, і складність роботи з багатовимірними масивами.

Я використовував різні алгоритми сортування, щоб впорядкувати масиви за зростанням або спаданням. Я порівнював їх часову і просторову складність, і визначав, який алгоритм є найкращим для певної ситуації. Я виявив, що сортування бульбашкою і сортування вибором є найпростішими, але найповільнішими алгоритмами, які мають квадратичну часову складність. Сортування злиттям і швидке сортування є складнішими, але найшвидшими алгоритмами, які мають логарифмічну часову складність. Однак, вони вимагають більше пам’яті, і не є стабільними, тобто не зберігають відносний порядок однакових елементів.

За допомогою цієї лабораторної роботи я розвинув свої знання і навички в роботі з простими структурами даних, і навчився застосовувати їх до різних задач програмування. Я також покращив своє розуміння алгоритмів сортування, і навчився аналізувати їх ефективність. Ці навички є важливими для розвитку мого мислення та логіки, і допоможуть мені в майбутніх проектах.