ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №5

за курсом “Програмування”

студентки групи ПА-23-1

Мазур Віолети

кафедра комп’ютерних технологій, ДНУ

2023/2024

1.Постановка задачі

Індивідуальне завдання. Варіант 12

Задане натуральне число n, цілі числа a1, …, an. Залишити без змін послідовність a1, …, an, якщо її члени упорядковані за зростанням або за спаданням. В протилежному випадку отримати послідовність a1, …, am. (m < n), ще m є таким, що або a1 ≤ a2 ≤ … ≤ am та am > am+1, або a1 ≥ a2 ≥ … ≥ am та am < am+1.

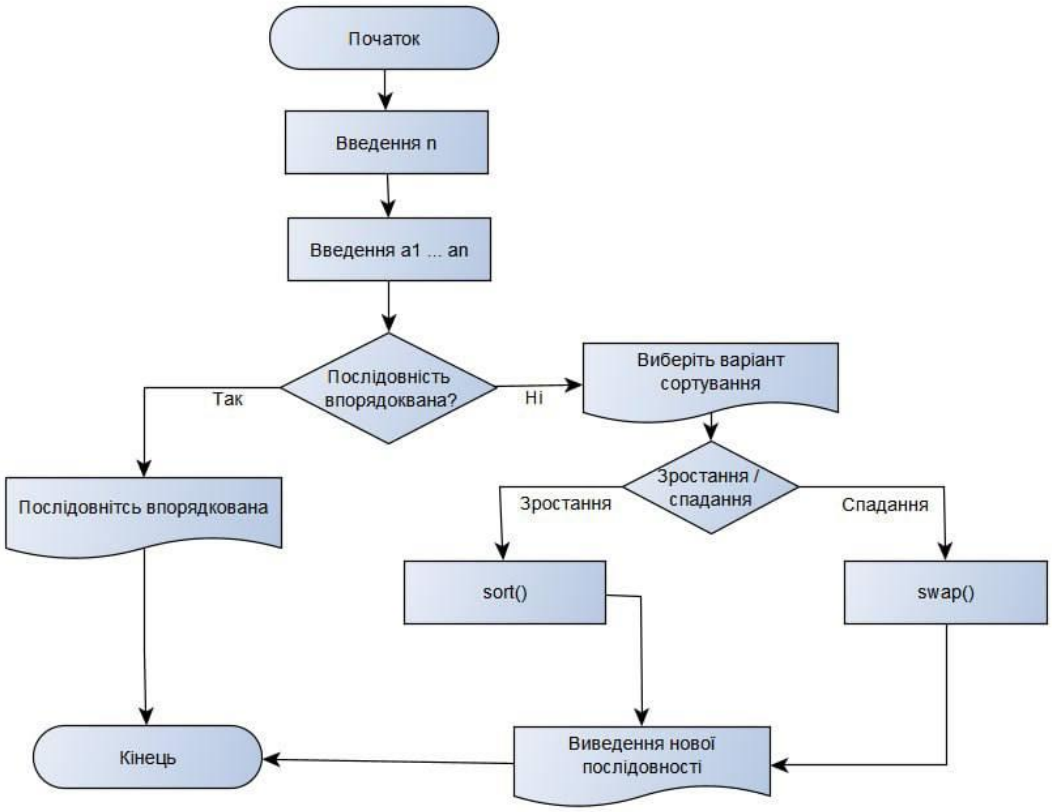
Не використовувати масиви.

2. Опис розв'язку + блок-схема

Опис розв'язку:

Цей код надає розв'язок до задачі, де задано натуральне число n та цілі числа a1, ..., an. Він перевіряє, чи послідовність впорядкована за зростанням або спаданням. Якщо послідовність є впорядкованою, то вона повертається без змін. Якщо послідовність не є впорядкованою, то вона модифікується, залишаючи лише підпослідовність до елементу зі зміною тенденції.

Блок-схема:



3. Вихідний текст програми розв'язку задачі (основні фрагменти з коментарями)

Код:

```
#include <iostream>

using namespace std;

class MyVector {
public:
    MyVector() : data(nullptr), size(0) {}

    void push_back(int value) {
        int* newData = new int[size + 1];
        for (int i = 0; i < size; ++i) {
            newData[i] = data[i];
```

```
        }
        newData[size] = value;
        delete[] data;
        data = newData;
        ++size;
    }

    int& operator[](int index) {
        return data[index];
    }

    int getSize() const {
        return size;
    }

    ~MyVector() {
        delete[] data;
    }

private:
    int* data;
    int size;
};

int main() {
    // Введення кількості чисел
    std::cout << "Введіть кількість чисел: ";
    int n;
```

```
std::cin >> n;

// Введення послідовності чисел
MyVector numbers;
std::cout << "Введіть послідовність чисел: ";
for (int i = 0; i < n; ++i) {
    int num;
    std::cin >> num;
    numbers.push_back(num);
}

// Перевірка чи послідовність вже впорядкована
bool isSortedAscending = true;
bool isSortedDescending = true;
for (int i = 1; i < numbers.getSize(); ++i) {
    if (numbers[i - 1] > numbers[i]) {
        isSortedAscending = false;
    }
    if (numbers[i - 1] < numbers[i]) {
        isSortedDescending = false;
    }
}

if (isSortedAscending || isSortedDescending) {
    std::cout << "Послідовність впорядкована." << std::endl;
    return 0;
}
```

```
// Вибір способу сортування
std::cout << "Оберіть спосіб сортування (1 - зростання, 2 - спадання): ";
int sortOption;
std::cin >> sortOption;

// Сортування у вибраному порядку
for (int i = 0; i < numbers.getSize(); ++i) {
    for (int j = 0; j < numbers.getSize() - i - 1; ++j) {
        if ((sortOption == 1 && numbers[j] > numbers[j + 1]) ||
            (sortOption == 2 && numbers[j] < numbers[j + 1])) {
            int temp = numbers[j];
            numbers[j] = numbers[j + 1];
            numbers[j + 1] = temp;
        }
    }
}

if (numbers.getSize() >= 2) {
    int lastIdx = numbers.getSize() - 1;
    int secondLastIdx = lastIdx - 1;
    if (sortOption == 1) {
        int temp = numbers[lastIdx];
        numbers[lastIdx] = numbers[secondLastIdx];
        numbers[secondLastIdx] = temp;
    } else {
        int temp = numbers[secondLastIdx];
        numbers[secondLastIdx] = numbers[lastIdx];
        numbers[lastIdx] = temp;
```

```
        }
    }

    // Виведення відсортованої та зміненої послідовності
    std::cout << "Відсортована послідовність: ";
    for (int i = 0; i < numbers.getSize(); ++i) {
        std::cout << numbers[i] << " ";
    }

    return 0;
}
```

4. Опис інтерфейсу програми (керівництво користувача)

Основний інтерфейс програми має наступну структуру:

1. Користувач вводить натуральне число n - кількість елементів послідовності.

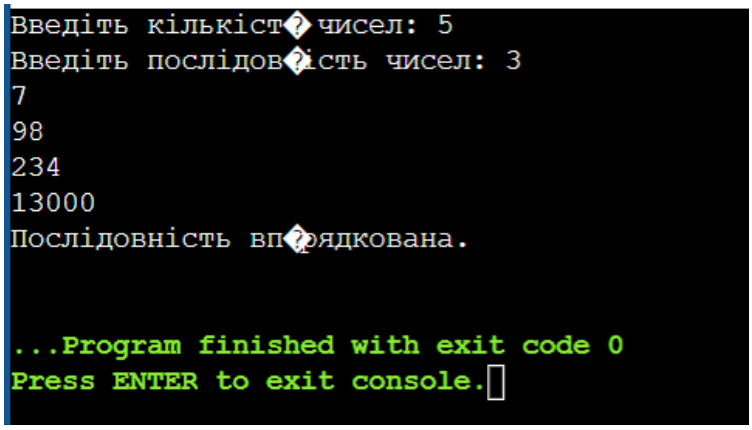
2. Користувач вводить цілі числа a1, ..., an, розділяючи їх пробілами.

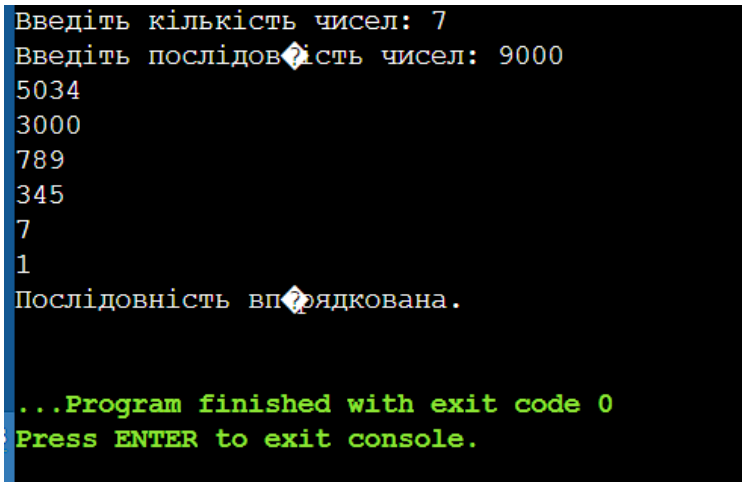
3. Програма перевіряє, чи є послідовність вже впорядкованою за зростанням або за спаданням. Якщо так, виводить повідомлення "Послідовність залишається без змін".

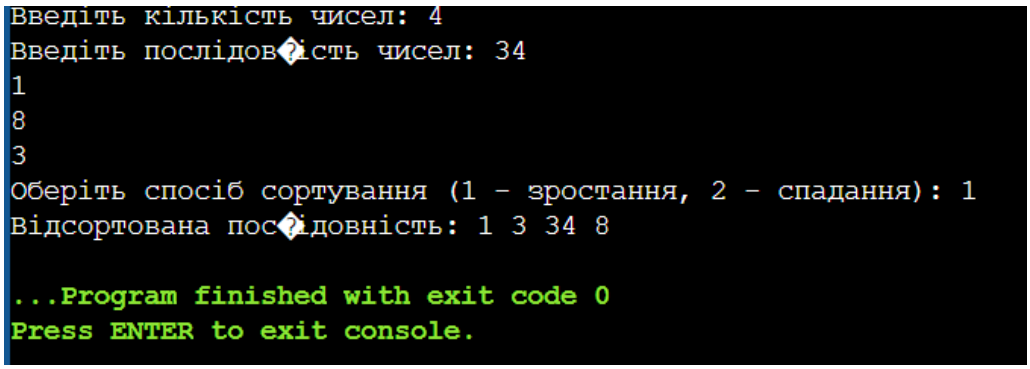
4. Якщо послідовність не є впорядкованою, програма знаходить підпослідовність a1, ..., am, де m < n, таку, що або a1 ≤ a2 ≤ ... ≤ am та am > am+1, або a1 ≥ a2 ≥ ... ≥ am та am < am+**1.**

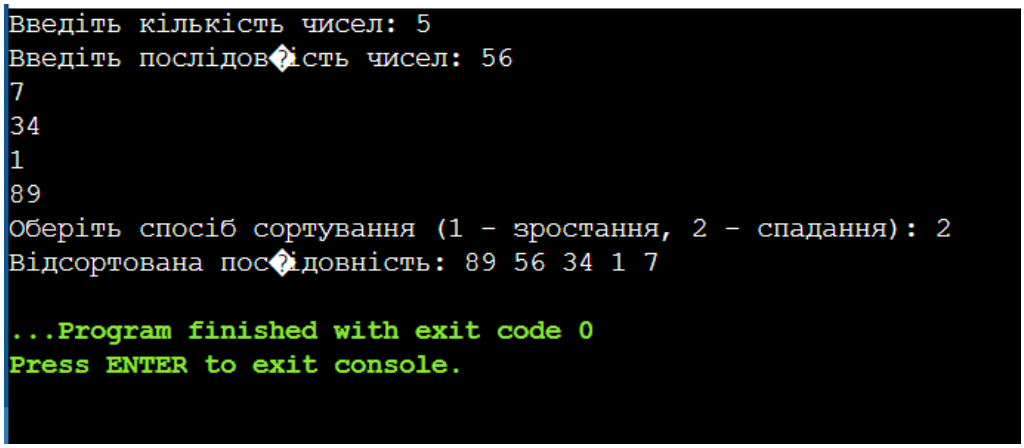
5. Програма виводить підпослідовність a1, ..., am.

5. Опис тестових прикладів









6. Аналіз помилок (опис усунення зауважень)

1. Помилка: Умова перевірки порядку сортування в коді неправильно сформульована. Опис усунення: Перевірте чи послідовність впорядкована за зростанням за допомогою "a[i] < a[i+1]" та за спаданням за допомогою "a[i] > a[i+1]". Якщо обидві перевірки повертають "false", то послідовність неупорядкована.

2. Помилка: Не визначена змінна "m" перед використанням у коді. Опис усунення: Спочатку визначте змінну "m" та ініціалізуйте її значенням, наприклад, 0.

3. Помилка: Умова перевірки напрямку послідовності в коді неправильно сформульована. Опис усунення: Перевірте чи послідовність має спадний напрямок за допомогою "am > am+1" або зростаючий напрямок за допомогою "am < am+1".

7. Висновки

У цій лабораторній роботі було розроблено консольну програму на мові програмування C++, яка включає одне індивідуальне завдання.

Завдання вимагало введення даних від користувача, обробки цих даних та виведення результату на екран. В процесі розробки, було важливо враховувати потенційні помилки, такі як некоректне введення даних, невірна обробка винятків та несправжня обробка умов. Додатково, блок-схема могла б бути використана для кращого розуміння логіки завдання та виявлення можливих помилок.

У разі виявлення помилок, важливо було їх виправити та перевірити програму на різних вхідних даних для впевненості в її коректності.

У цілому, ця лабораторна робота надала можливість поглибити розуміння основ програмування на мові C++, вирішити завдання та навчитися виявляти та виправляти можливі помилки в програмі.