МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Основи програмування»**

**на тему: «** **Одновимірні**[**масиви**](http://lider.diit.edu.ua/mod/lesson/view.php?id=40269)**. Алгоритми сортування для одновимірних масивів.**[**Функції**](http://lider.diit.edu.ua/mod/lesson/view.php?id=55933)**.»**

Виконав: студент гр. ПЗ2011

Кулик Сергій Вадимович

Прийняла: ас. каф. КІТ

Нежуміра О. І.

Дніпро, 2020

**Тема.** Одновимірні [масиви](http://lider.diit.edu.ua/mod/lesson/view.php?id=40269" \o "Масиви). Алгоритми сортування для одновимірних масивів. [Функції](http://lider.diit.edu.ua/mod/lesson/view.php?id=55933" \o "Функції).

**Мета.** 1. Отримати навички роботи з одновимірними масивами в мові С++.

          2. Навчитися розробляти для одновимірних масивів алгоритми:

              - обробки елементів;

              - сортування.

          3. Отримати навички розробки і використання функцій.

1. Постановка задачі

Розробити програму по вибраному варіанту.  
Вимоги до програми:  
- вхідні дані вводити з клавіатури;  
- вхідні дані перевіряти на коректність (діапазон значень, захист від дурня);  
- результати роботи програми вивести на екран.  
Вимоги до тексту програми:  
- коментарі щодо призначення програми, її вхідних та вихідних даних;  
- коментарі щодо призначення блоків програми;  
- самодокументуємий код: назви (ідентифікатори) змінних і функцій мають назви, що відповідають їх суті.

Індивідуальне завдання: Є масив координат точок на осі Х. Знайти координату точки, найближчої до заданої точки.  
Сортування масиву за методом " Сортування парне-непарне"

## Зовнішні специфікації

Вхідні дані: Елементи масиву

Формат вхідних даних відображений у таблиці 1

## Таблиця 1

**Формат вхідних даних**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Елементи масиву | - | Дійсні числа | 4 або -9,3 |
| 2 | Задана точка | N | Дійсне число | 4 або -9,3 |

Вихідні дані: упорядкований масив, найближчий елемент до заданого числа

Формат вихідних даних відображений у таблиці 2

**Формат вихідних даних**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Відсортований  масив | - | Дійсні числа | 8 або -9,3 |
| 2 | Найближчий елемент массива до заданого числа | - | Дійсне число | 8 або -9,3 |

3. Функціональні вимоги до програми

Програма повинна реалізовувати такі дії:

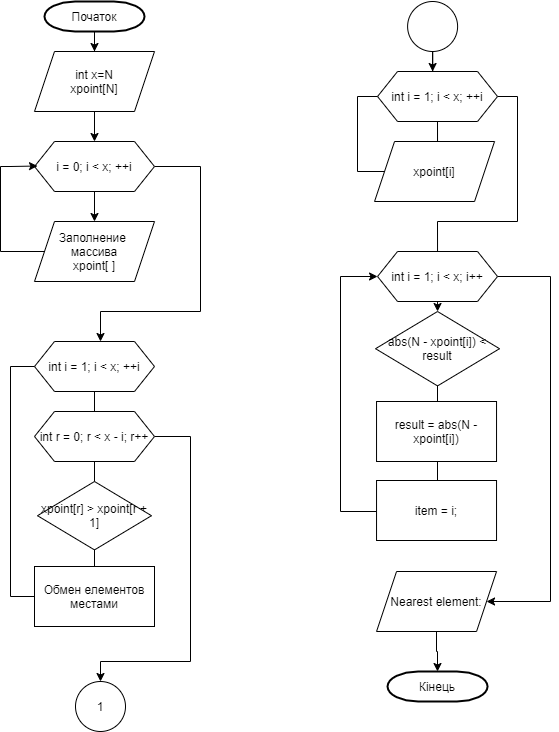
* Введення користувачем елементів масиву.
* Вивід упорядковування масиву.
* Знаходження у масиві найближчого числа до заданого.
* Вивід результату.

4. Тести до алгоритму програми

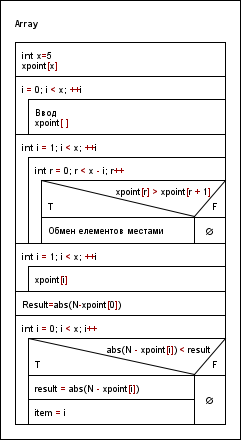
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Вхідні дані | | Очікувані результати |
| 1 | Довільні числа в масиві | 6, 4, 8, 2, 1,-2, 0, 6.3, -4, -8 | | -8, -4, -2, 0, 1, 2, 4, 6, 6.3, 8 |
| 2 | Довільні числа з повторенням | 6, 6, 3, 2, 2, 8, 4, -4, -4, -1 | | -1, -4, -4, 2, 2, 4, 3, 6, 6, 8 |
| 3 | Відсортований масив | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 | | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 |
| 4 | Відсортований масив в зворотному напрямку | 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 | | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 |
| 5 | Число N дорівнює одному із елементів массиву | N | array | Найближчий елемент 8 |
| N=8 | 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 |
| 6 | Число не N дорівнює одному із елементів массиву | N | Array | Найближчий елемент 10 |
| N=12 | 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 |

5. Алгоритми

5.1 Блок-схема



5.2 Діаграмма Нассі-Шнайдермана



6. Текст программи

/\*/\*Є масив координат точок на осі Х. Знайти координату точки, найближчої до заданої точки.\*/

#include <iostream>

#include <clocale>

using namespace std;

void restore\_stream(std::istream& stream)

{

// очистити прапор стану потоку (робимо його рівним goodbit)

stream.clear();

// видалити з буфера введення непрочитані через виникнення помилки символи

stream.ignore(numeric\_limits<streamsize>::max(), '\n');

}

//Функція "Захист від дурня"

void try\_again()

{

if (cin.fail())

{

while (cin.fail())

{

cout << "Sorry, not currently . Try again!" << endl;

restore\_stream(cin);

}

}

}

int main()

{

const int x = 10;

float xpoint[x];

cout << "Enter points that are located along the X axis" << endl;

//Заполнения массива с клавиатуры

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

{

cout << "[ " << i << " ] = ";

cin >> xpoint[i];

//Контроль вводу даних

if (cin.fail())

{

while (cin.fail())

{

cout<<"Sorry, not currently . Try again!"<<endl;

restore\_stream(cin);

{

--i;

}

}

}

}

//Сортировка массива методом пузырька

for (int i = 1; i < x; ++i)

{

for (int r = 0; r < x - i; r++)

{

if (xpoint[r] > xpoint[r + 1])

{

// Обмен местами

float temp = xpoint[r];

xpoint[r] = xpoint[r + 1];

xpoint[r + 1] = temp;

}

}

}

//Вывод отсортированого массива

cout << "sorted array: ";

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

{

cout << xpoint[i] << " ";

}

cout << endl;

//Задання точки

int N = 0, item = 0;

cout << "Set point:" << endl;

cin >> N;

if (cin.fail())

{

while (cin.fail())

{

//Захист від дурня

cout << "Sorry, not currently . Try again!" << endl;

restore\_stream(cin);

cin >> N;

}

}

//Пошук найбилщого елементу до заданого числа

int result = abs(N - xpoint[0]);

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

{

if (abs(N - xpoint[i]) < result)

{

result = abs(N - xpoint[i]);

item = i;

}

}

cout << "Nearest element: " << xpoint[item] << endl;

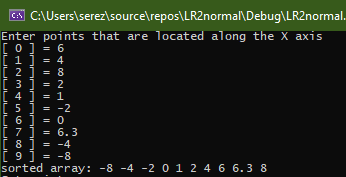
return 0;

}

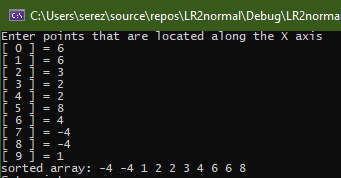
}

7. Результати тестувань:

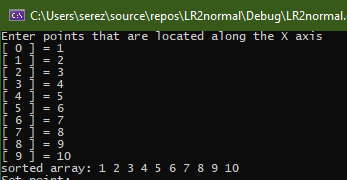
Вхідні дані: 6, 4, 8, 2, 1,-2, 0, 6.3, -4, -8



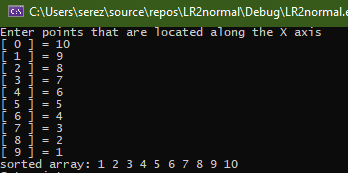
Довільні числа з повторенням: 6, 6, 3, 2, 2, 8, 4, -4, -4, -1



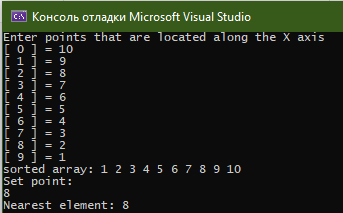
Масив відсортований : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



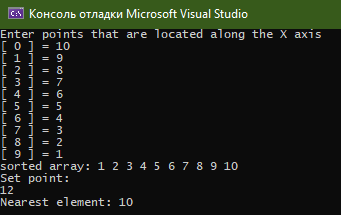
Масив відсортованийв зворотньому напрямку: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1



Число N дорівнює одному із елементів массиву



Число N не дорівнює одному із елементів массиву



Аналіз результатів тестування

Результати перевірки збігаються з результатами роботи программи. Массиви сортуються правильно. Пошук елементів також працює корректно. У программі передбаченаперевірка вхідних данних, тому тест на «захист від дурня» не проводився.

Висновок

Під час виконання лабараторної роботи, познайомився з одновимірними масивами. Для своєї програми використав алгоритм сортування бульбашкою, хоча існують більш швидкі алгоритми сортування. Для заповнення масиву елементами використовував цикл. Розмір масиву повинен бути відомий до компіляції програми. Використовував цикл для виведення відсортованого масива на екран. Для захисту від дурня використав виклик функції. Якщо користувач вводить некорректні дані, то программа показує відповідне повідомлення, та просить ввести дані ще раз.