**Лабораторна робота №11. Вступ до показчиків**

**1 Вимоги**

**1.1 Розробник**

* Михневич Владислав Вікторович
* Студент 1-го курсу
* Групи КІТ-120а

**1.2 Загальне завдання**

Розробити програми, що вирішують завдання з використанням показчиків.

**1.3 Індивідуальне завдання**

* Реалізувати програму, що записує елементи головної діагоналі в одновимірний масив і упорядковує цей масив за зростанням.

**2. Опис програми**

**2.1 Функціональне призначення**

Програма записує елементи головної діагоналі двовимірного квадратного масиву array у масив main\_diagonal\_elements. Далі елементи масиву main\_diagonal\_elements упорядковуються за зростанням.

**2.2 Опис логічної структури програми**

Для заповнення масиву main\_diagonal\_array викликаємо функцію fill\_array\_\_diagonal\_num, яка приймає параметрами розміри двовимірного масиву, показчик на двовимірний масив, показчик на масив з елементами головної діагоналі. Сортуємо масив за допомогою функції buble\_sort, яка приймає параметрами розмір масиву та показчик на масив.

**Функція заповнення масиву елементами головної діагоналі двовимірного масиву:**

int fill\_array\_\_diagonal\_num

* *Призначення*: заповнює масив елементами головної діагоналі двовимірного масиву.
* *Опис роботи*: функція записує у i елемент масиву з елементами головної діагоналі, елементи двовимірного масиву з рівними індексами.Схема алгоритму функції\_ подана на рис. 1.

.

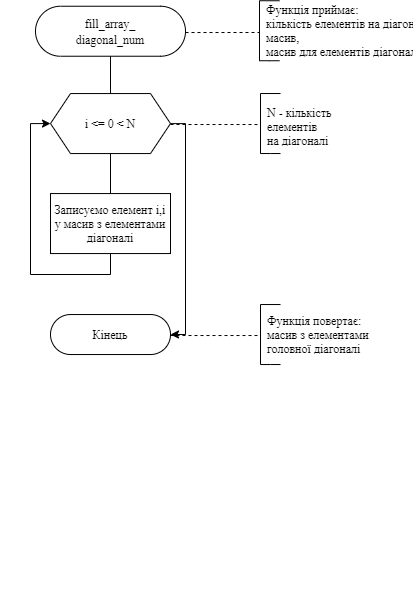


Рисунок 1 — Схема алгоритму функції fill\_array

**Функція сортування**

void buble\_sort

* *Призначення*: сортує масив за зростанням бульбашковим методом.
* *Опис роботи:* функція змінює місцями елементи, якщо елемент зліва більший за елемент справа. Схема алгоритму функції подана на рис. 2.

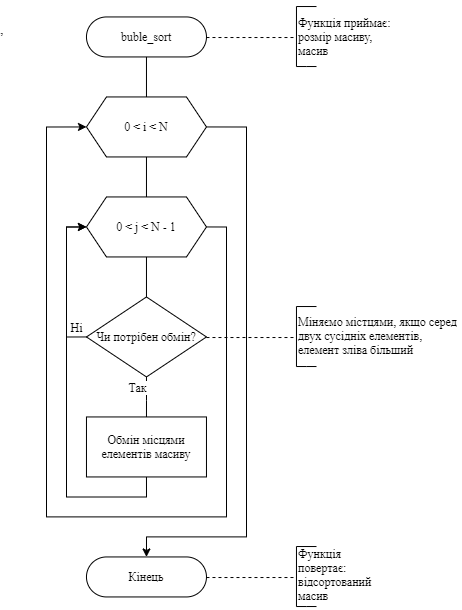


Рисунок 2 - схема алгоритму функції сортування

**Основна функція**

int main

* *Призначення:* головна функція
* *Опис роботи:* функція заповнює двовимірний масив випадковими елементами за допомогою функції fill\_array, заповнює функцію з елементами головної діагоналі за допомогою функції fill\_array\_\_diagonal\_num, і робить сортування масиву за допомогою функції buble\_sort. Схема алгоритму функції подана на рис. 3.

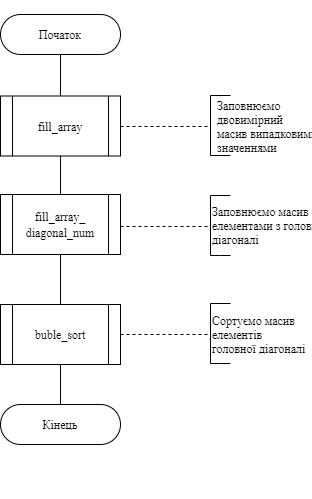


Рисунок 3 — Схема алгоритму функції main

**Структура проекту**

├── doc

│   ├── assets

│   │   ├── buble\_sort.png

│   │   ├── fill\_array\_diagonal\_num.png

│   │   ├── fill\_array.png

│   │   ├── get\_random\_num.png

│   │   └── main.png

│   ├── lab\_11.docx

│   ├── lab\_11.md

│   └── lab\_11.pdf

├── Doxyfile

├── Makefile

└── src

├── main.c

└── Task\_1-3

├── Task\_1.c

├── Task\_2.c

└── Task\_3.c

**2.3 Важливі фрагменти програми**

* Початкові дані. Константи

#define SIZE 5

* Запис елементів головної діагоналі

for (int i = 0; i < side\_size; i++)

{

main\_diagonal\_array[i] = square\_array[i][i];

}

* Зміна місцями елементів під час сортування

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

array[j + 1] = array[j] - array[j + 1];

array[j] -= array[j + 1];

**3. Варіанти використання**

Для демонстрації результатів використовується покрокове виконання програми та інші засоби налагодження відлагодника nemiver. Нижче наводиться послідовність дій запуску програми у режимі відлагодження.

*Крок 1* (див. рис. 4). Знаходячись в основній процедурі, досліджуємо стан змінних, в тому числі констант.

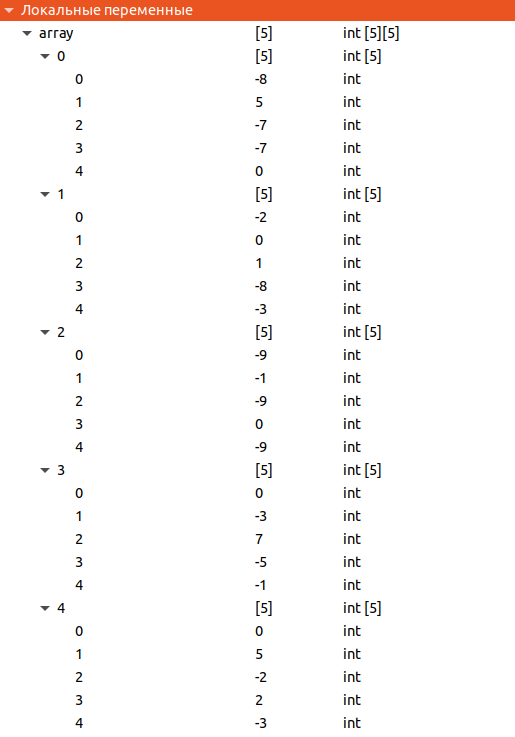


Рисунок 4 — значення змінних при запуску програми

*Крок 2* (див. рис. 5). Дослідження стану змінних після виконання функції сортування.

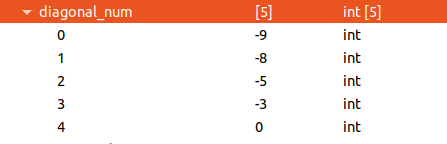


Рисунок 5 - значення змінних після сортування

**Висновки**

На цій лабораторній роботі ми навчилися створювати показчик на масив, навчилися здійснювати виділення та звільнення пам'яті, та чому її потрібно звільнювати, а також вивчили операції які можуть застосовуватись до покажчиків.