**Лабораторна робота №10. Документація проекту.**

**1. Вимоги**

**1.1 Розробник:**

* Клименко Юрій;
* студент групи КІТ 120-а;
* 18-грудня-2020.

**1.2 Загальне завдання**:

Розробити повноцінний звіт для лабораторної роботи "Функції".

**1.3 Перевірив:**

асист. Челак Віктор Володимирович.

**2. Загальний опис проекту.**

Проект складається з:

* папки **src** з 3 кодами
* Doxyfile
* Makefile
* Readme

**Структура проекту:**

├── Doxyfile

├── Makefile

├── README.md

└── src

├── lab05\_05.c

├── lab06\_06.с

└── lab07\_01.c

**3. Опис коду lab05\_05.c**

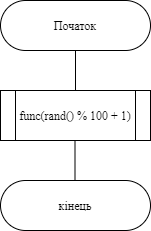
**3.1** Функціональне призначення.

Програма отримує корінь числа без допомоги зовнішніх бібліотек. Результат зберігається в змінній **root\_num**. Демонстрація результату передбачає покрокове виконання програми.

**3.2** Опис логічної структури.

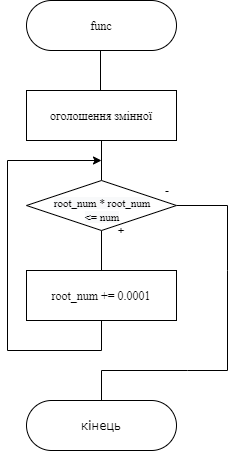
Спочатку ми задаємо число за допомогою генератора псевдовипадкових чисел. Потім ми оголошуємо функцію **func**, котра і знаходить нам корінь цього числа, поступово збільшуючи наш "корінь" на *0.0001* і перемножує його само на себе, доки ми не отримаємо число, задане в **main()**.

Головна функція **main()**. Схема алгоритму подана на рисунку 1.



**Рисунок 1** - схема алгоритму головної функції **main().**

Функція **func**, що знаходить корінь до заданого числа в головній функції. Схема алгоритму зображена на 2 рисунку.



**Рисунок** **2** - схема алгоритму функції **func**.

**3.3** Результат виконання програми.

Використавши debug можно перевірити програму на дієздатність. Результат на рисунках 3-4.



**Рисунок 3** - задане число генератором псевдовипадкових чисел.



**Рисунок** **4** - отриманий корінь.

**4. Опис коду lab06\_06.c**

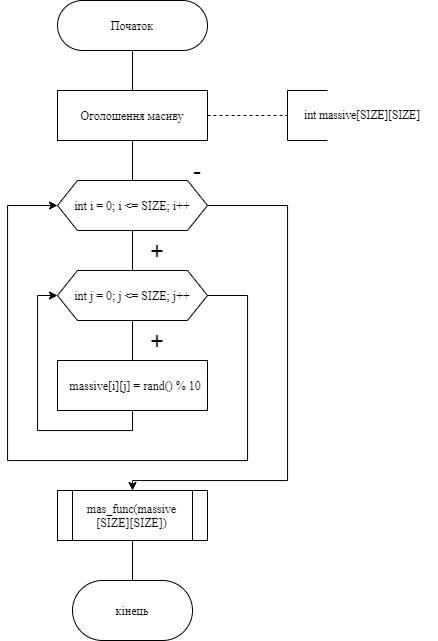
**4.1** Функціональне призначення.

Програма перемножує матрицю саму на себе. Результат зберігається в масиві **new\_massive**. Демонстрація результату передбачає покрокове виконання програми.

**4.2** Опис логічної структури програми.

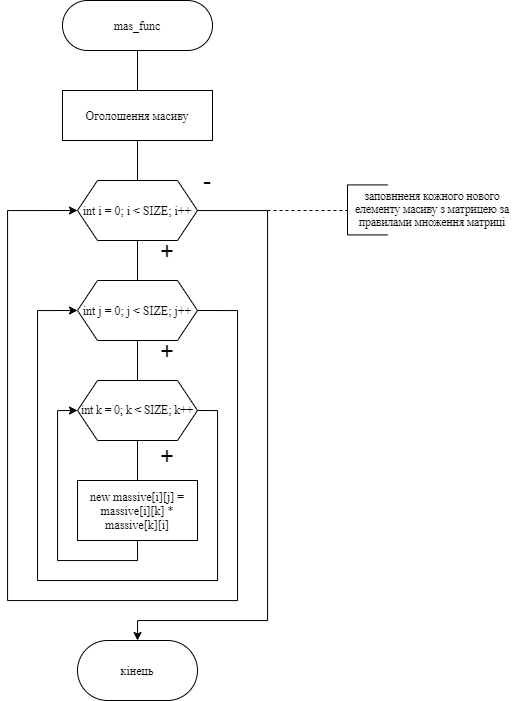
Спочатку ми оголошуємо масив з матрицею, яку будемо перемножувати саму на себе. Задаємо псевдовипадкові числа цій матриці. Потім за допомогою функції **mas\_func** ми виконуємо пермноження матриці саму на себе за правилами множення матриць.

*Головна функція* **main().** Схема алгоритму подана на рисунку 5.



**Рисунок 5** - cхема алгоритму головної функції **main()**.

Функція **mas\_func**, що заповнює кожний новий елемент масиву з матрицею за правилами множення матриці. Схема алгоритму на 6 рисунку.



**Рисунок 6** - схема алгоритму функції **mas\_func**.

**4.3** Результат виконання програми.

Використавши **debug** можно перевірити програму на дієздатність. Результат на рисунку 7.



**Рисунок 7** - результат виконання програми у **debug**.

**5. Опис коду lab07\_01.c.**

**5.1** Функціональне призначення.

Програма визначає кількість пар у послідовності, де перше число менше наступного за допомогою варіативної функції. Результат зберігається в **amount\_numerator**.

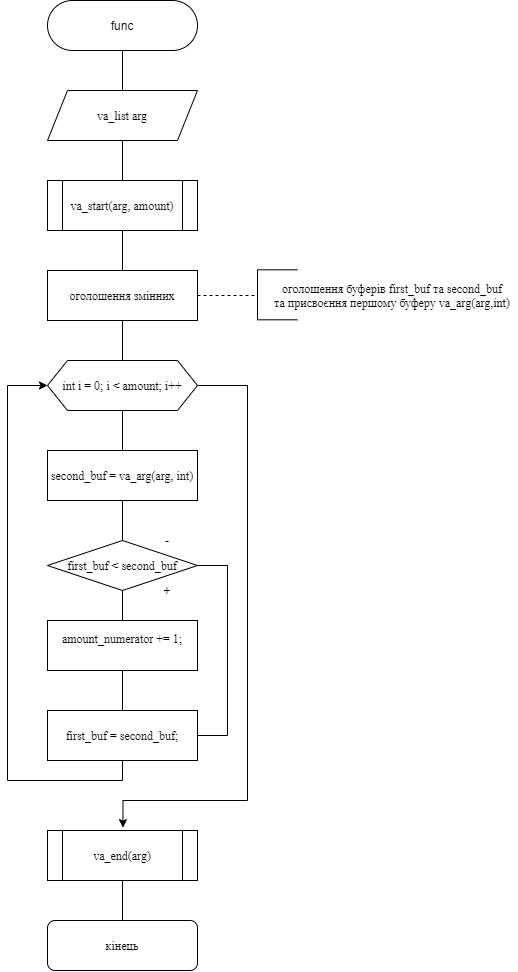
**5.2** Опис логічної структури програми.

Оголошуємо головну функцію **main()**, у котрій викликаємо функцію **func**, де задаємо кількість чисел в послідовності та саму послідовність(5 - чисел в послідовності; 1, 2, 3, 4, 5 - сама послідовність). Далі **func** сортує цю послідовність та знаходить кількість пар, де перше число менше наступного.

*Головна функція* **main().**Схема алгоритму подана на рисунку 8.

**Рисунок 8** - cхема алгоритму головної функції **main()**.

Функція **func**, котра сортує послідовність та знаходить кількість пар, де перше число менше наступного. Схема алгоритму зображена



**Рисунок 9** - схема алгоритму функції **func**.

**5.3** Результат виконання програми.

Використавши **debug** можно перевірити програму на дієздатність. Результат на рисунку 10.



**Рисунок 10** - результат виконання програми у **debug**.

**Висновок**

При виконанні лабораторної роботи №10 було отримано досвід у розробці документації формату **markdown**.