**Лабораторна робота №8. Вступ до блок-схем алгоритмів. №9. Вступ до документації коду (частина 1). №10. Вступ до локументації проекту.**

1. **Вимоги**
   1. Розробник

### Бельчинська Катерина Юріївна

### студентка групи КІТ-320

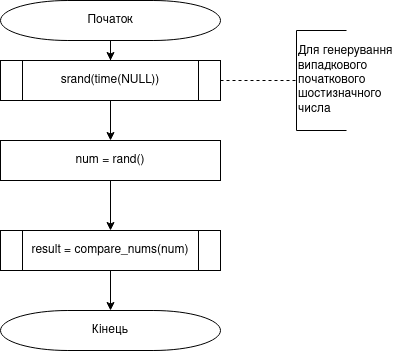
### 0-dec-2020

* 1. **Загальне завдання**
* Переробити програми, що були розроблені під час лабораторних робіт з тем “Масиви” та “Цикли” таким чином, щоб для обчислення результату використовувалися функції.
* Реалізувати функцію з варіативною кількістю аргументів.
  1. **Індивідуальне завдання**
* Визначити, чи є ціле 6-значне число “щасливим” квитком (сума першої половини чисел номера дорівнює сумі другого).

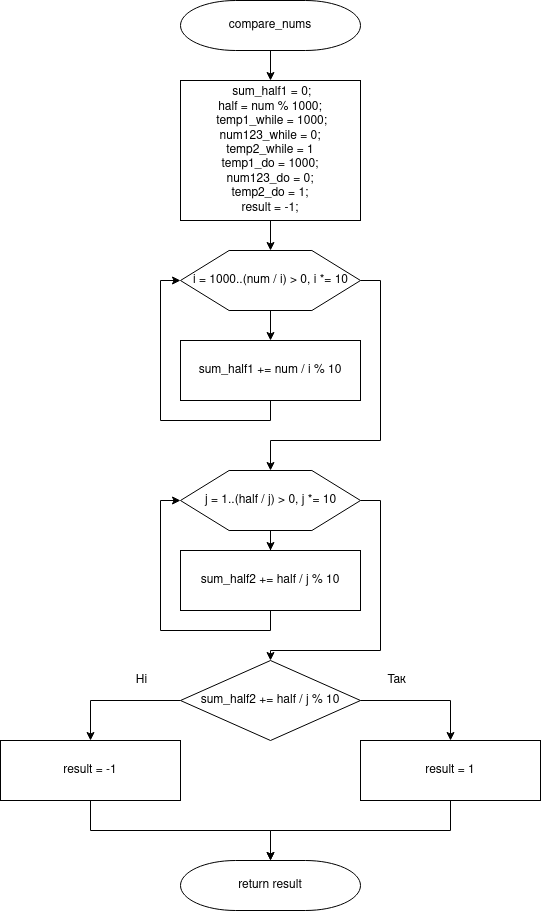
1. **Опис роботи**
   1. **Функціональне призначення**

* Виконування дій програми за допомогою функцій, для спрощення загального завдання програми. Функція допомагає «розбити» велику задачу на підзадачі, у функції main лише зсилатися на їх опис.
  1. **Опис логічної структури для завдання 3 з лабораторної роботи № 5**

Функція 'main' генерує задане число та викликає функцію для обчислення результату. Схема алгоритму функції:



Функція 'compare\_nums' сумує перші та останні три цифри між собою відповідно та порівнює отримані значення між собою. Схема алгоритму функції:



* 1. **Важливі елементи програми**

Виділяємо та сумуємо між собою перші три цифри числа:

int compare\_nums(int num) {

int sum\_half1 = 0;

for (int i = 1000; (num / i) > 0; i \*= 10) {

sum\_half1 += num / i % 10;

}

Виділяємо та сумуємо між собою останні три цифри числа:

int half = num % 1000;

int sum\_half2 = 0;

for (int j = 1; (half / j) > 0; j \*= 10) {

sum\_half2 += half / j % 10;

}

Порівняння обох значень

if (sum\_half1 == sum\_half2 && num123\_while == num456\_while && num123\_do == num456\_do) {

result = 1;

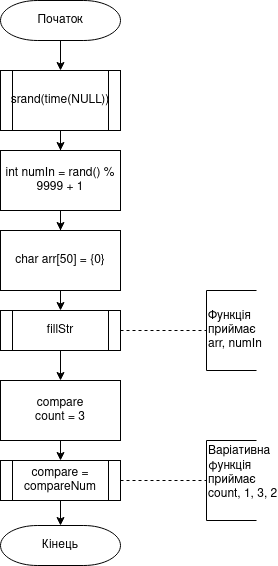
} else {

result = -1;

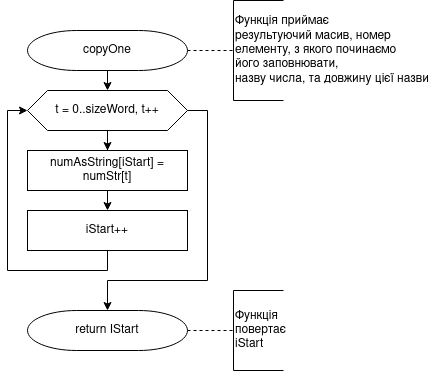
}

* 1. **Опис логічної структури для завдання 3 з лабораторної роботи № 6**

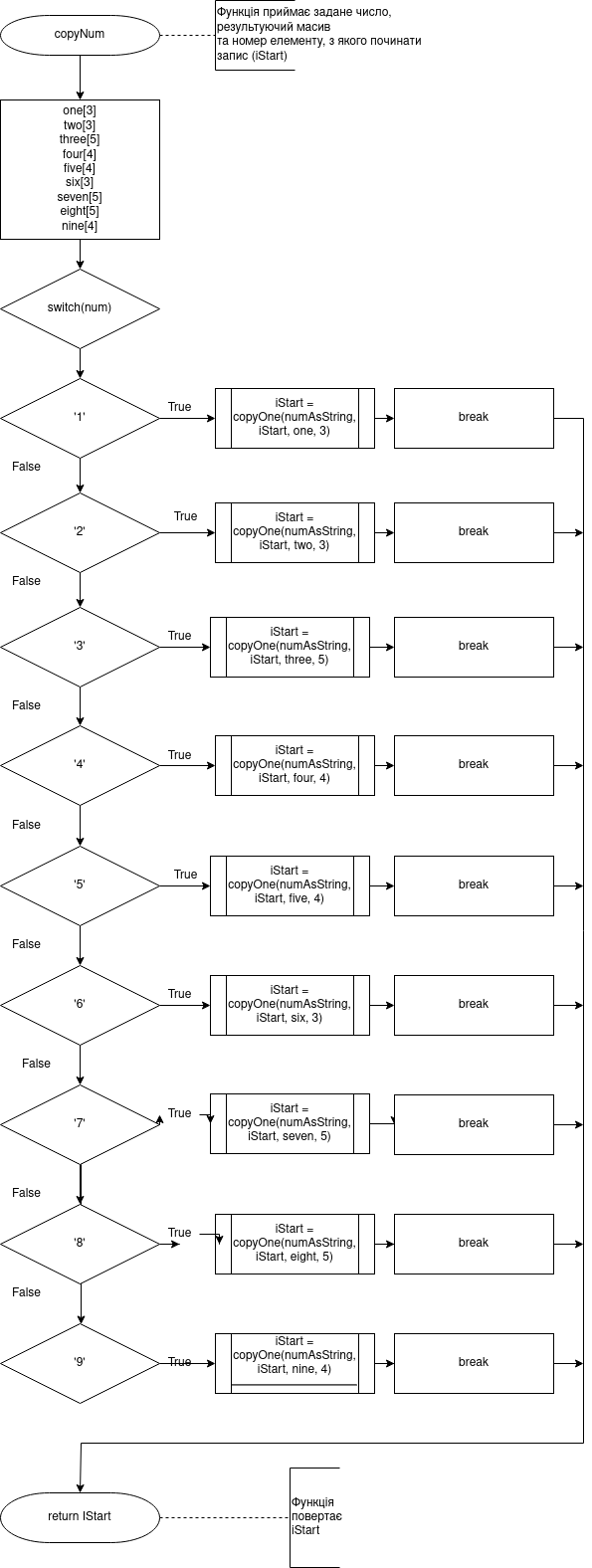
Функція 'main' оприділяємо задане число, інціалізація результуючого масиву, виклик функції, яка заповнює цей масив, виклик варітивної функції, яка обчислює кількість пар, у яких перше число менше наступного.



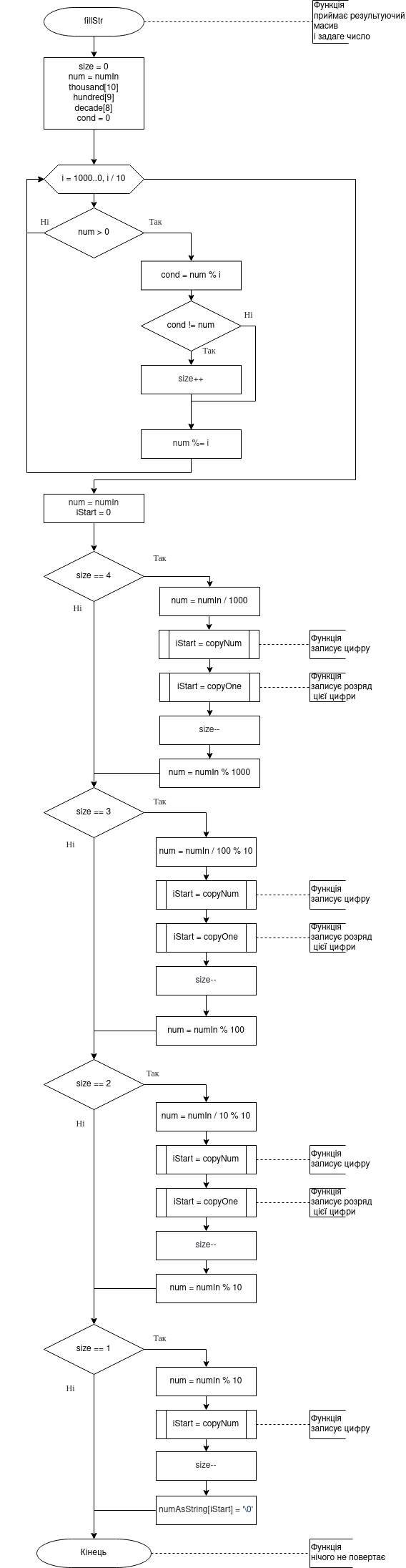
Функція 'copyOne' переписує число або розряд у результуючий масив. Схема алгоритму функції:



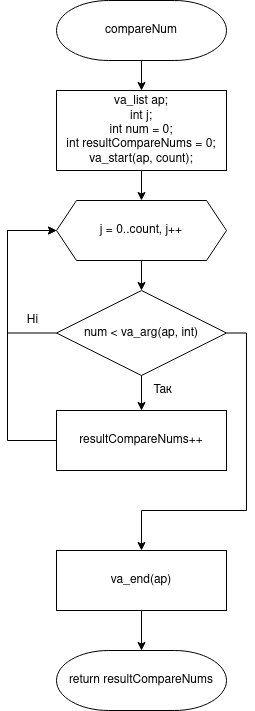
Функція 'copyNum' оприділяє, яку цифру треба записати у результуючий масив. Схема алгоритму функції:



Функція 'fillStr' заповнює результуючий масив, оприділяючи перед цим розряд заданого числа. Схема алгоритму функції:



Функція 'compareNum' обчислює кількість пар, де перше число менше наступного. Схема алгоритму функції:

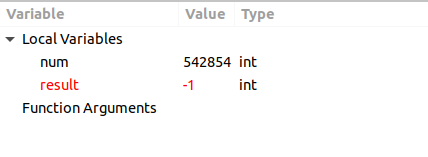


* 1. **Варіанти використання**
* Для завдання 3 з лабораторної роботи № 5
  + у відладнику nemiver викликаємо у функції

main три функції, які оприділяли «щасливе» задане число чи ні для циклів for, while do та do while.

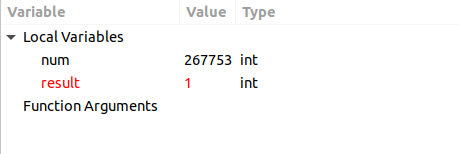
* + Викликаємо функцію для числа, обчисленого генератором псевдовипадкових чисел rand():

число «не щасливе»:

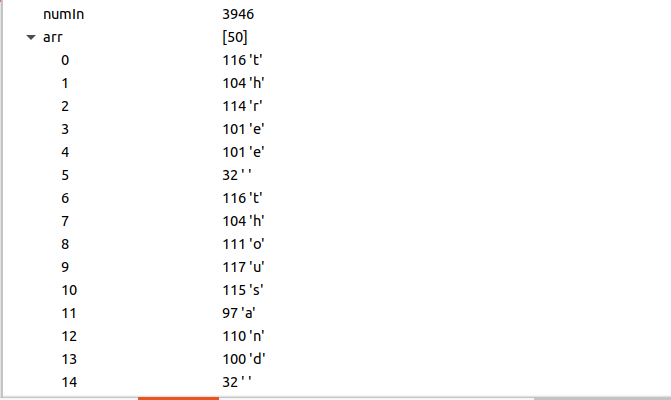
****

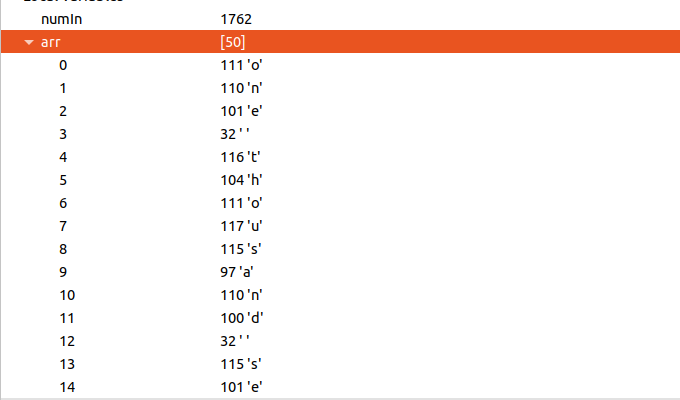
****

число «щасливе»:



* Для завдання 3 з лабораторної роботи № 6
  + у відладнику nemiver за допомогою точок зупину заходимо у функцію, яка заповнює результуючий масив заданим числом, перетвореним у строкузадане число генерується за допомогою генератора псевдовипадкових чисел rand():







* Структура проекту:

.

├── doc

│   ├── assets

│   │   ├── lab05\_false2.png

│   │   ├── lab05\_false.png

│   │   ├── lab05\_task3\_compare\_nums.png

│   │   ├── lab05\_task3\_main.png

│   │   ├── lab05\_true.png

│   │   ├── lab06(1).png

│   │   ├── lab06(2).png

│   │   ├── lab06(3).png

│   │   ├── lab06\_task3\_compareNum.png

│   │   ├── lab06\_task3\_copyNum.png

│   │   ├── lab06\_task3\_copyOne.png

│   │   ├── lab06\_task3\_fillStr.png

│   │   └── lab06\_task3\_main.png

│   ├── lab08\_09\_10.docx

│   ├── lab08\_09\_10.md

│   └── lab08\_09\_10.pdf

├── Doxyfile

├── lab05\_task3

│   ├── README.md

│   └── src

│   └── main.c

├── lab06\_task3

│   ├── README.md

│   └── src

│   └── main.c

├── Makefile

└── README.md

**Висновок**

В ході даної лабораторної роботи, я навчилася використовувати функцкції, які повертають і не повертають результат, задля уникання повторів в коді; «спрощувати» загальне завдання,розбиваючи його на меньші