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

1. Вимоги

1.1 Розробник

- Бельчинська Катерина Юріївна;
- студентка групи КІТ-320;
- 04-dec-2020.

1.2 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.

1.3 Індивідуальне завдання

Визначити, чи є ціле 6-значне число “щасливим” квитком (сума першої половини чисел номера дорівнює сумі другого).

2. Опис роботи

2.1 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.

2.2 Опис логічної структури для завдання 3 з лабораторної роботи № 5

Функція 'main' генерує задане число та викликає функцію для обчислення результату. Схема алгоритму функції наведена на рис. 1.

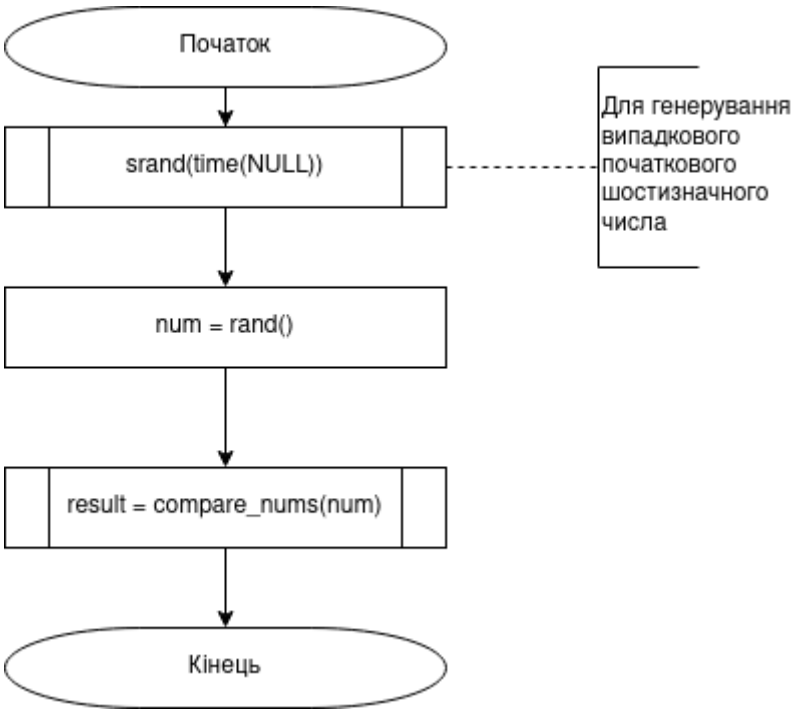

Рис. 1. Схема алгоритму функції main (lab05)

Функція 'compare_nums' сумує перші та останні три цифри між собою відповідно та порівнює отримані значення між собою. Схема алгоритму функції наведена на рис. 2.

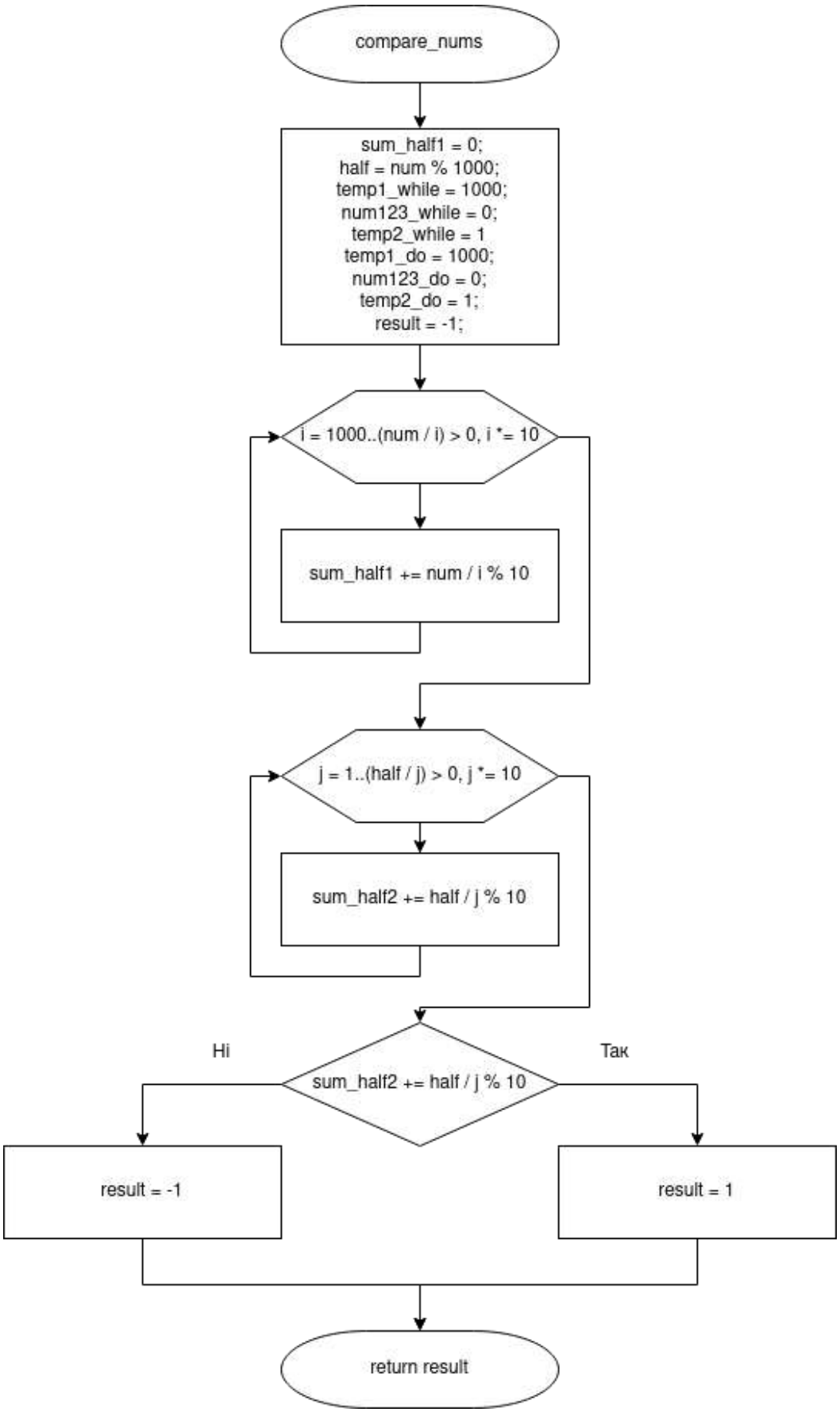

Рис. 2. Схема алгоритму функції compareNums

2.3 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;

	}
```

2.4 Опис логічної структури для завдання 3 з лабораторної роботи № 6

Функція 'main' - визначається задане число, ініціалізація результуючого масиву, виклик функції, яка заповнює цей масив, виклик варіативної функції, яка обчислює кількість пар, у яких перше число менше наступного. Схема алгоритму функції наведена на рис. 3.

Функція 'copyOne' переписує число або розряд у результуючий масив. Схема алгоритму функції наведена на рис. 4.

Функція 'copyNum' оприділяє, яку цифру треба записати у результуючий масив. Схема алгоритму функції наведена на рис. 5.

Функція 'fillStr' заповнює результуючий масив, оприділяючи перед цим розряд заданого числа. Схема алгоритму функції наведена на рис. 6.

Функція 'compareNum' обчислює кількість пар, де перше число менше наступного. Схема алгоритму функції наведена на рис. 7.

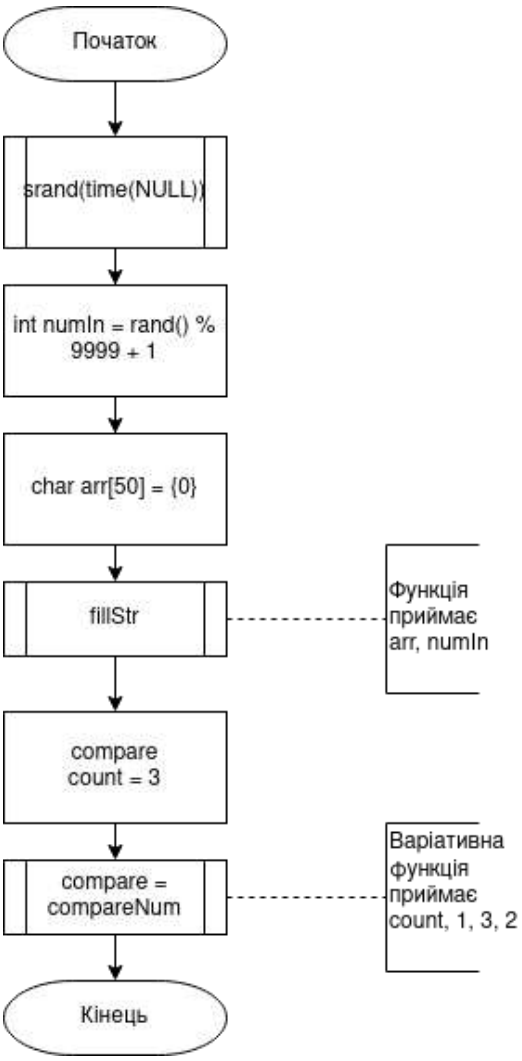

Рис. 3. Схема алгоритму функції main (lab06)

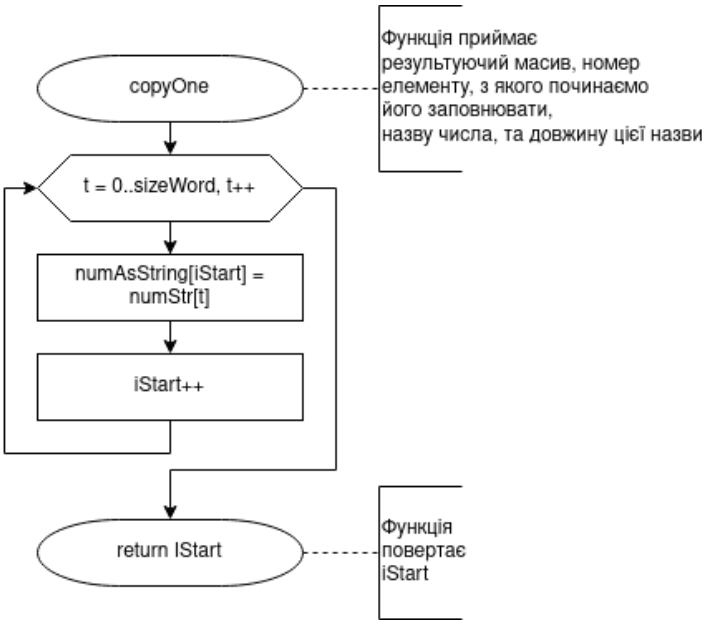

Рис. 4. Схема алгоритму функції copyOne

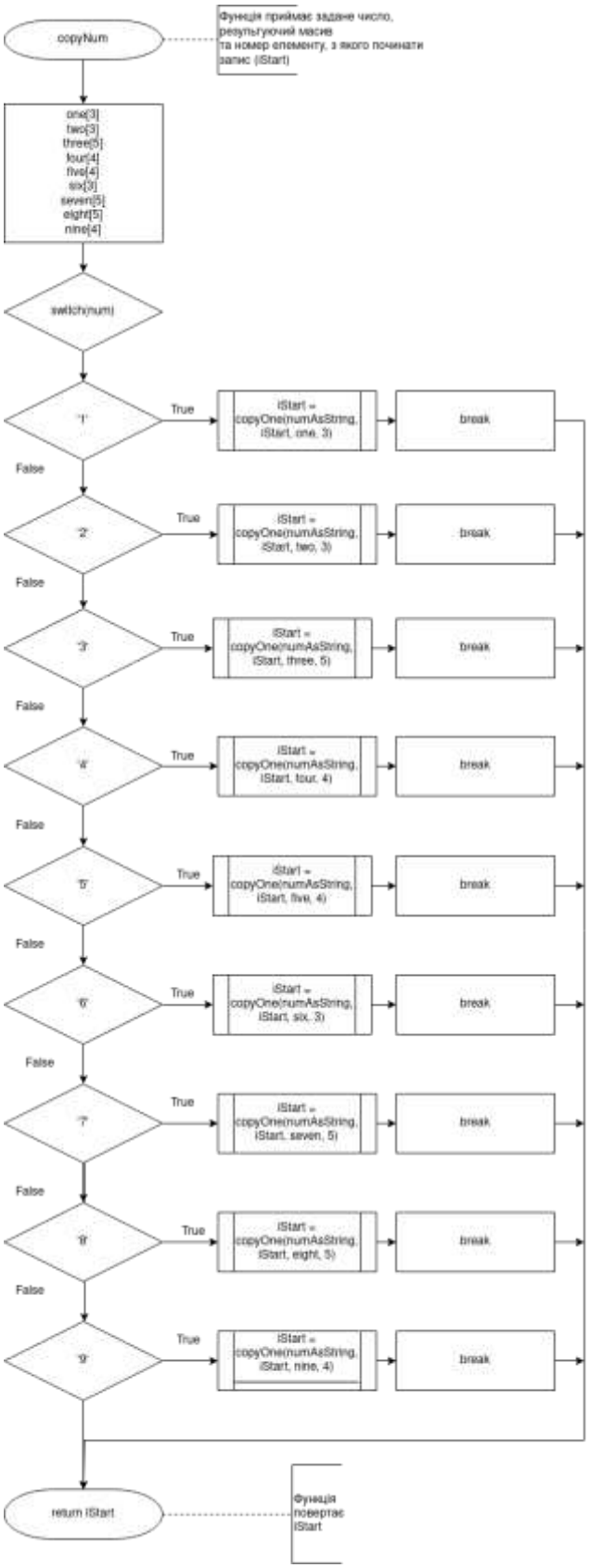

Рис. 5. Схема алгоритму функції copyNum

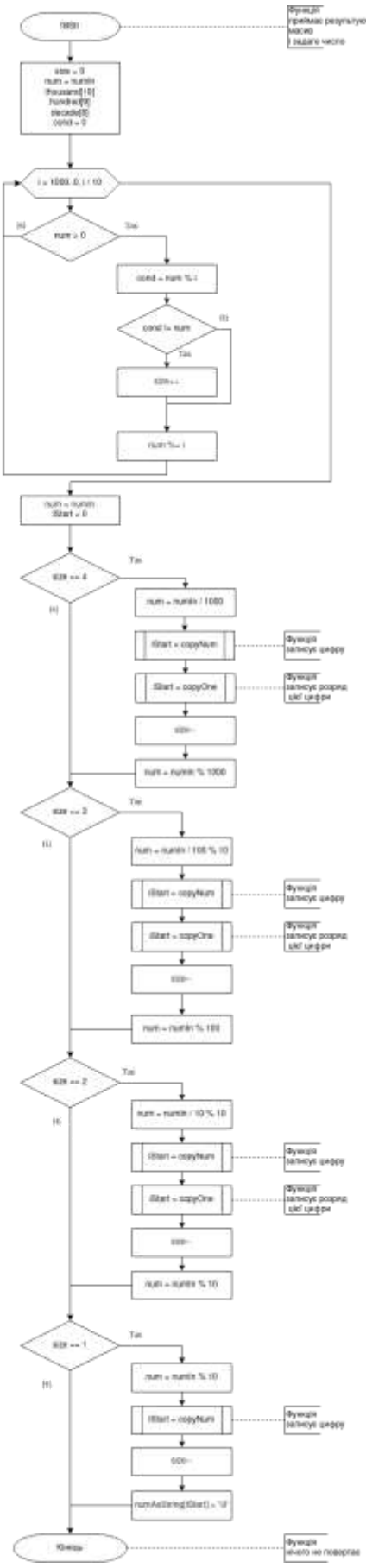

Рис. 6. Схема алгоритму функції fillStr

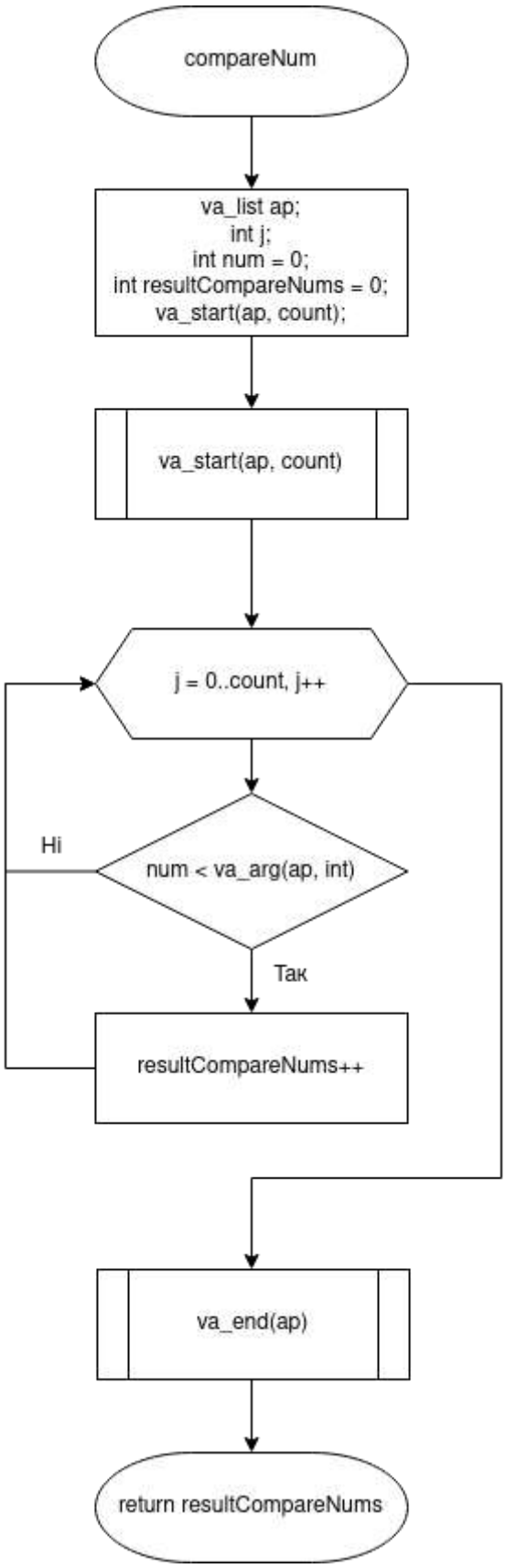

Рис. 7. Схема алгоритму функції compareNum

2.5 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
```

3. Варіанти використання

Для завдання 3 з лабораторної роботи № 5.

У відлагоджувачі nemiver викликаємо у функції main три функції, які визначали «щасливе» задане число чи ні для циклів for, while do та do while. Викликаємо функцію для числа, обчисленого генератором псевдовипадкових чисел rand(). На рис. 8 наведено приклад роботи програми, якщо число «не щасливе». На рис. 9 наведено приклад роботи програми, якщо число «щасливе».

Рис. 8. Результат роботи програми, якщо число «не щасливе»

Рис. 9. Результат роботи програми, якщо число «щасливе»

Для завдання 3 з лабораторної роботи № 6.

У відлагоджувачі nemiver за допомогою точок зупинки заходимо у функцію, яка заповнює результуючий масив заданим числом, перетвореним у строку, задане число генерується за допомогою генератора псевдовипадкових чисел rand(). На рис. 10-12 наведено приклади роботи програми для різних початкових даних.

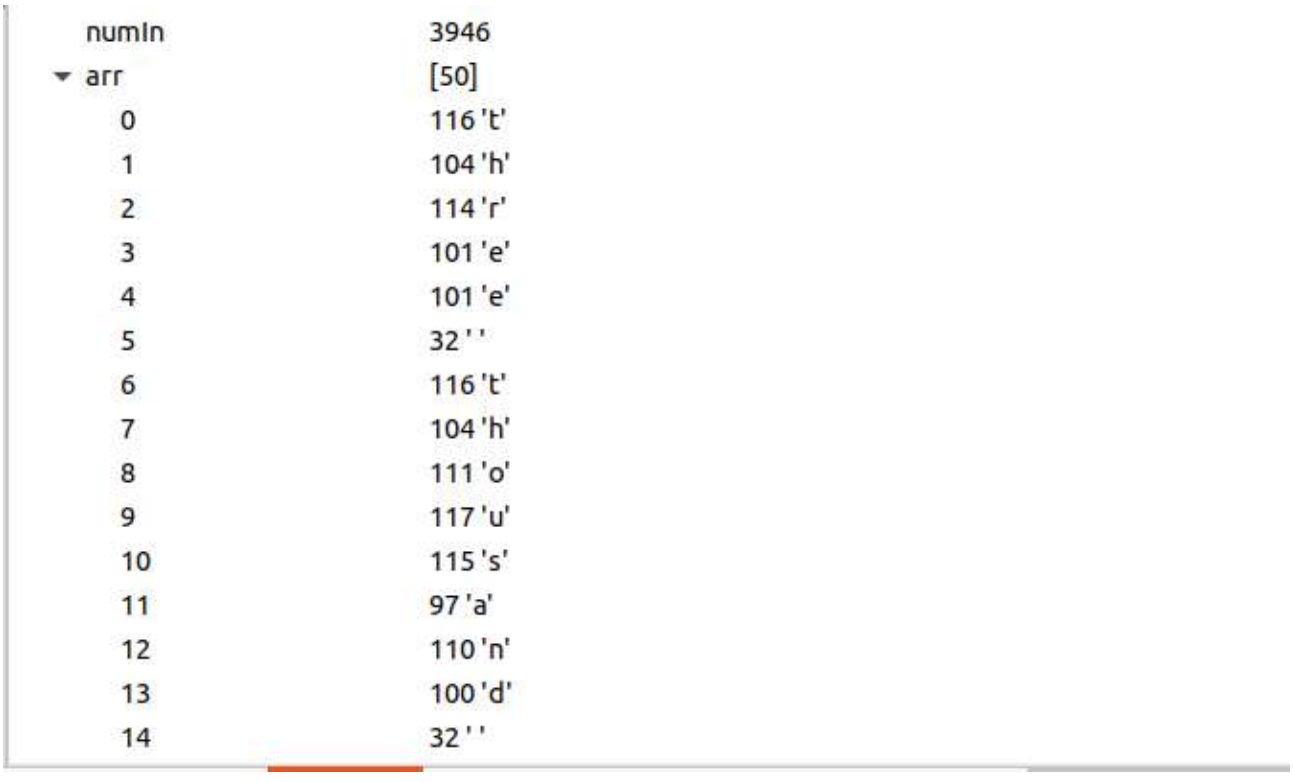

Рис. 10. Результат роботи програми для числа 3946

numIn	1762
arr	[50]
0	111 'o'
1	110 'n'
2	101 'e'
3	32 ' '
4	116 't'
5	104 'h'
6	111 'o'
7	117 'u'
8	115 's'
9	97 'a'
10	110 'n'
11	100 'd'
12	32 ' '
13	115 's'
14	101 'e'

Рис. 11. Результат роботи програми для числа 1762

numIn	477	int
arr	[50]	char [50]
0	102 'f'	char
1	111 'o'	char
2	117 'u'	char
3	114 'r'	char
4	32 ' '	char
5	104 'h'	char
6	117 'u'	char
7	110 'n'	char
8	100 'd'	char
9	114 'r'	char
10	101 'e'	char
11	100 'd'	char
12	32 ' '	char
13	115 's'	char
14	101 'e'	char
15	118 'v'	char
16	101 'e'	char

Рис. 12. Результат роботи програми для числа 477

Висновки

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 менші.