ЛАБОРАТОРНА РОБОТА № 12. ВЗАЄМОДІЯ З КОРИСТУВАЧЕМ ШЛЯХОМ МЕХАНІЗМУ ВВЕДЕННЯ/ВИВЕДЕННЯ.

ЛАБОРАТОРНА РОБОТА № 14. ВЗАЄМОДІЯ З ФАЙЛАМИ.

1. ВИМОГИ
   1.1. **Розробник**

- Бельчинська Катерина Юріївна;
- студентка групи КІТ-320;
- 2021.

1.2. **Індивідуальне завдання (лабораторна робота №12)**

Програму, яка була розроблена у лабораторній роботі з покажчиками змінити так, щоб:

- початкові дані вводилися з клавіатури;
- видача результуючих даних провадилася у консоль;
- при старті програми виводилась інформація об авторі, номері лабораторної роботи;
- при запиті даних, користувач отримав повідомлення, що від нього очікують.

Продемонструвати взаємодію з користувачем шляхом використання функцій:

- *printf()* та *scanf();*
- *gets(), getc()* та *puts(), putc();*
- *write(), read()*

1.3. **Індивідуальне завдання (лабораторна робота №14)**

Програму, яка була розроблена в лабораторній роботі зі строками, змінити так, щоб:

- початкові дані вводилися з файлу;
- видача результуючих даних провадилася не тільки у консоль, але й у файл;
- ім’я вхідного та результуючого файлу повинно бути отримано від користувача;
- при запиті даних, користувач отримав повідомлення, що від нього очікують.

2. ОПИС ПРОГРАМИ (ЛАБОРАТОРНА РОБОТА №12)

2.1. Функціональне призначення

Покажчики доцільно використовувати для працювання не з копією елементу, а безпосередньо з самим елементом, за допомогою його адреси знаходження у пам'яті.

2.2. Опис логічної структури

Функція 'main' виділяє пам'ять для заданого і результуючого масиву, викликає усі функції для заповнення заданного і результуючого масивів, визначення найбільшої незменшуваної послідовності. Схема алгоритму функції наведена на рис. 1.

Функція 'fillArrOne' заповнює заданий масив псевдовипадковими числами. Схема алгоритму функції наведена на рис. 2.

Функція 'countOfIncreasingSequences' рахує кількість незменшуваних послідовностей. Схема алгоритму функції наведена на рис. 3.

Функція 'findMaxIncreasingSequence' шукає найдовшу ділянку. Схема алгоритму функції наведена на рис. 4.

Функція 'fillArrayOut' заповнює результуючий масив. Схема алгоритму функції наведена на рис. 5.

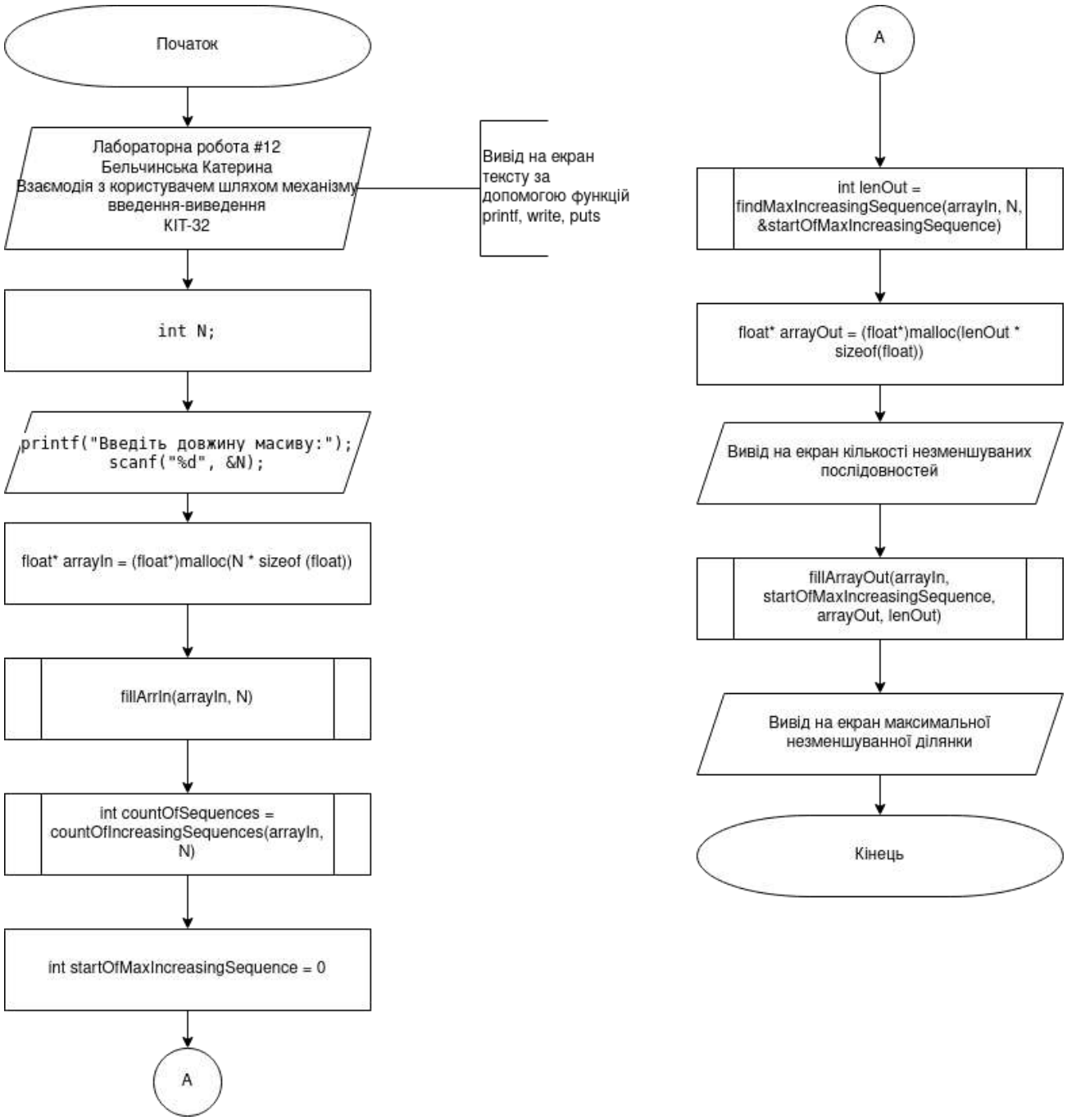


Рис.1. Схема алгоритму функції main

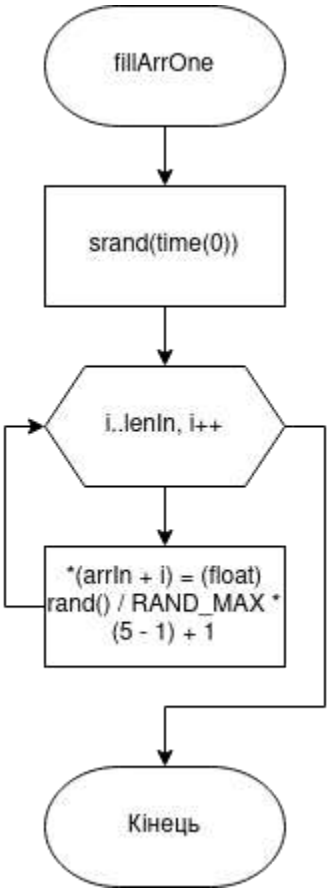


Рис.2. Схема алгоритму функції fillArrOne

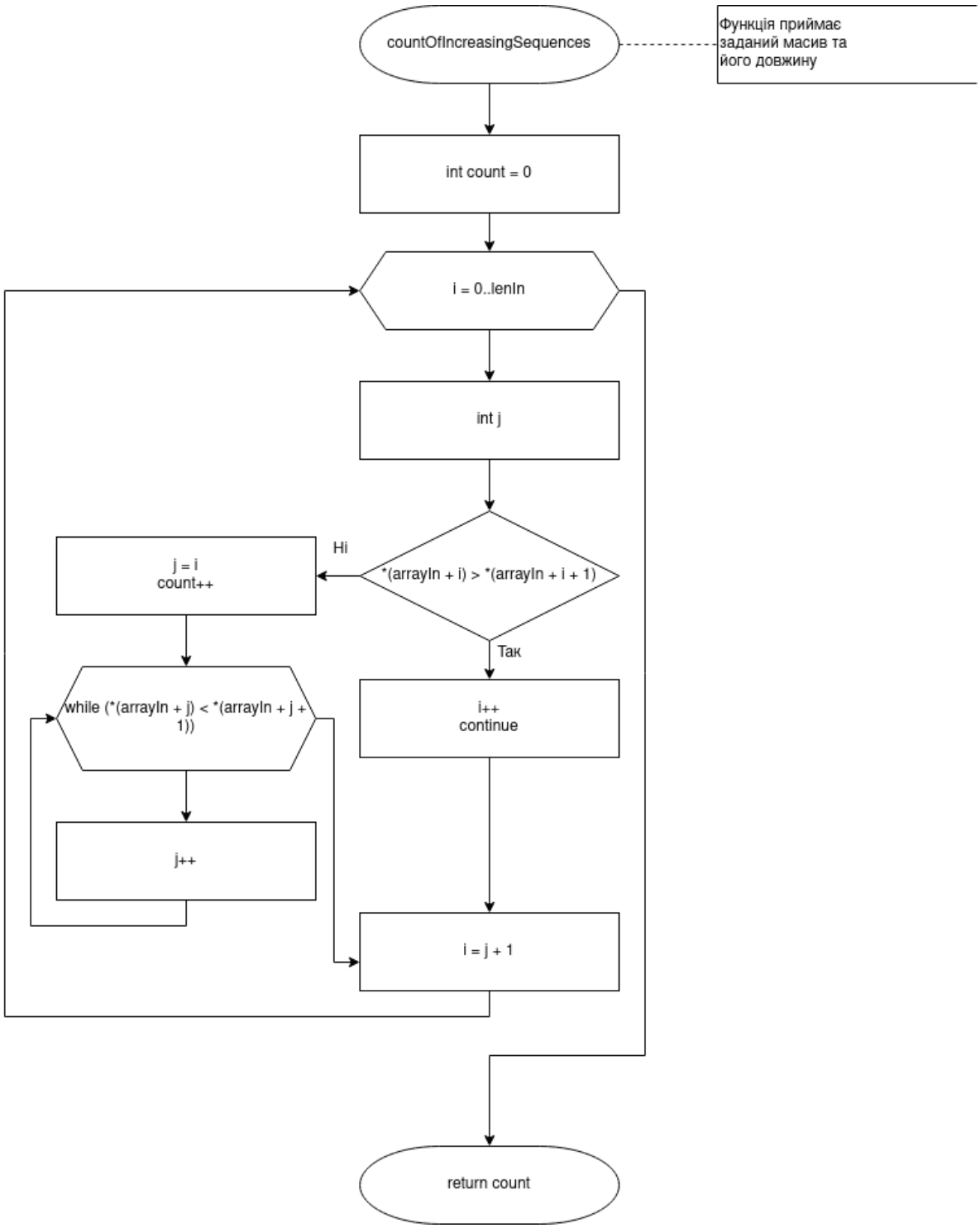


Рис.3. Схема алгоритму функції countOfIncreasingSequences

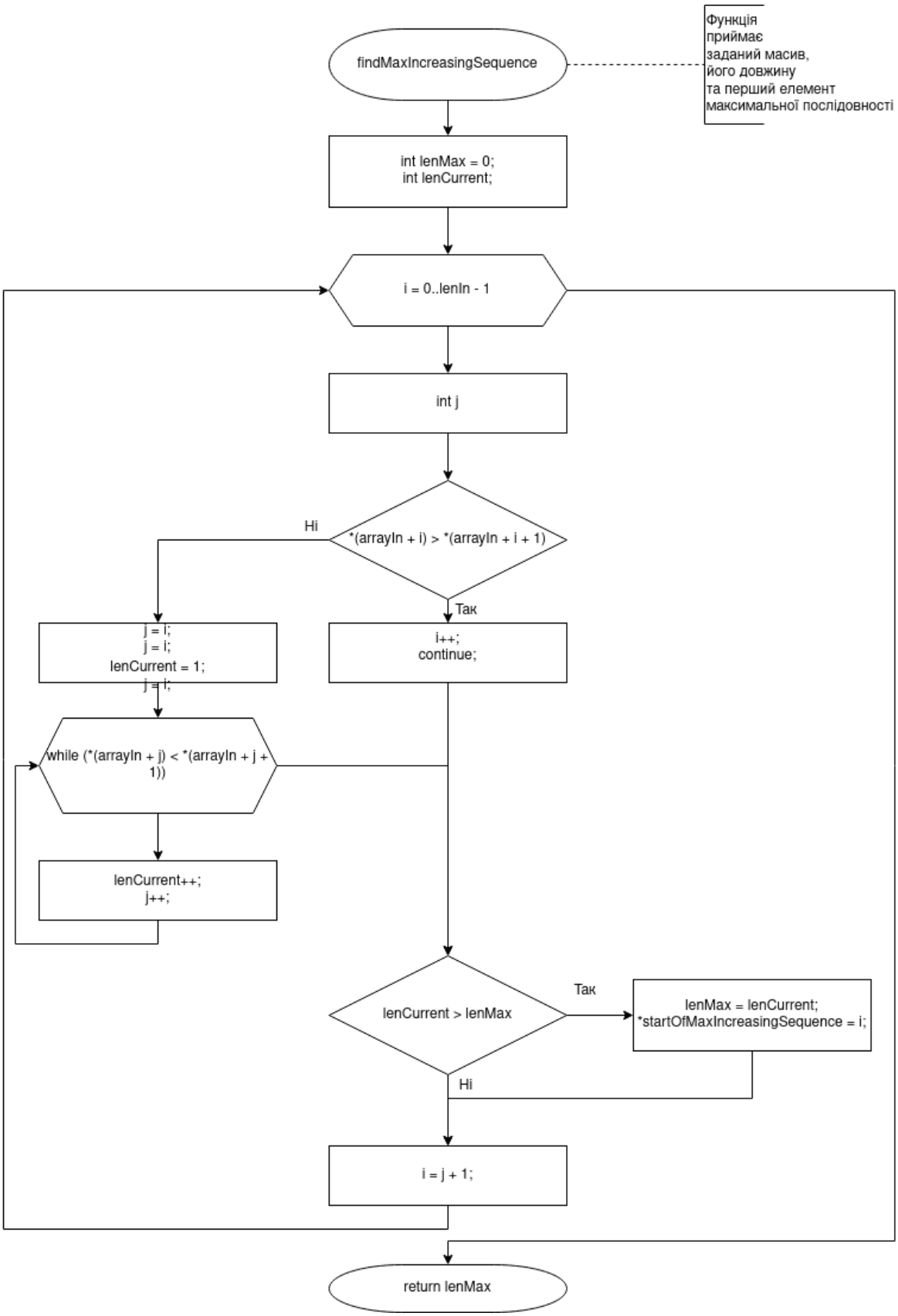


Рис.4. Схема алгоритму функції findMaxIncreasingSequence

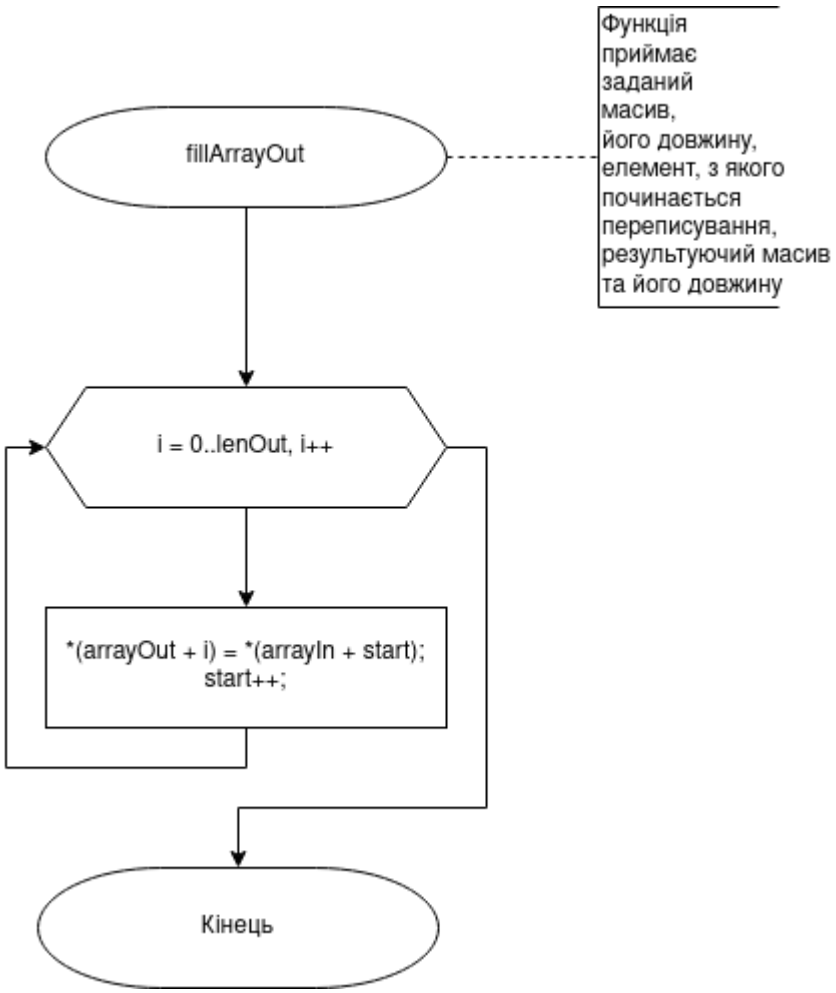


Рис.5. Схема алгоритму функції fillArrayOut

2.3. Структура проекту

```
.
├── doc
│   ├── assets
│   │   ├── countOfIncreasingSequences.png
│   │   ├── doxygen_lib.c.png
│   │   ├── doxygen_main.c.png
│   │   ├── doxygen_mainpage.png
│   │   ├── fillArrayOut.png
│   │   ├── fillArrayOut_.png
│   │   ├── fillArrOne.png
│   │   ├── findMaxIncreasingSequence.png
│   │   └── main.png
│   ├── lab12_14.pdf
│   └── lab12_14.docx
```

```
├── Doxyfile
├── Makefile
├── README.md
└── src
    ├── lib.c
    ├── lib.h
    └── main.c
```

2.4. Генерування Doxygen-документації

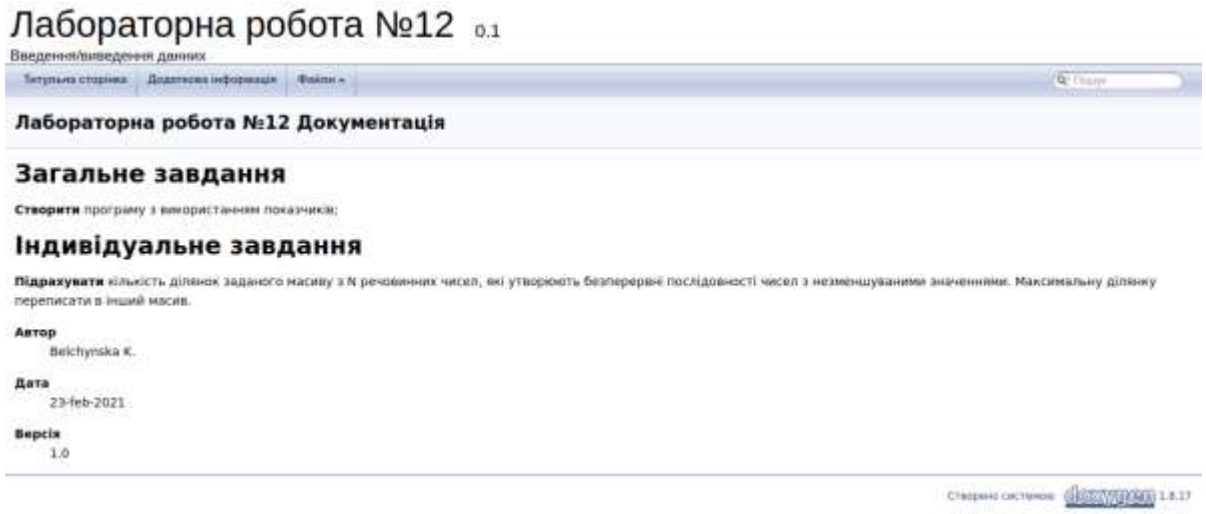


Рис.6 Титульна сторінка Doxygen.

Рис.7 Коментарі до функції main.

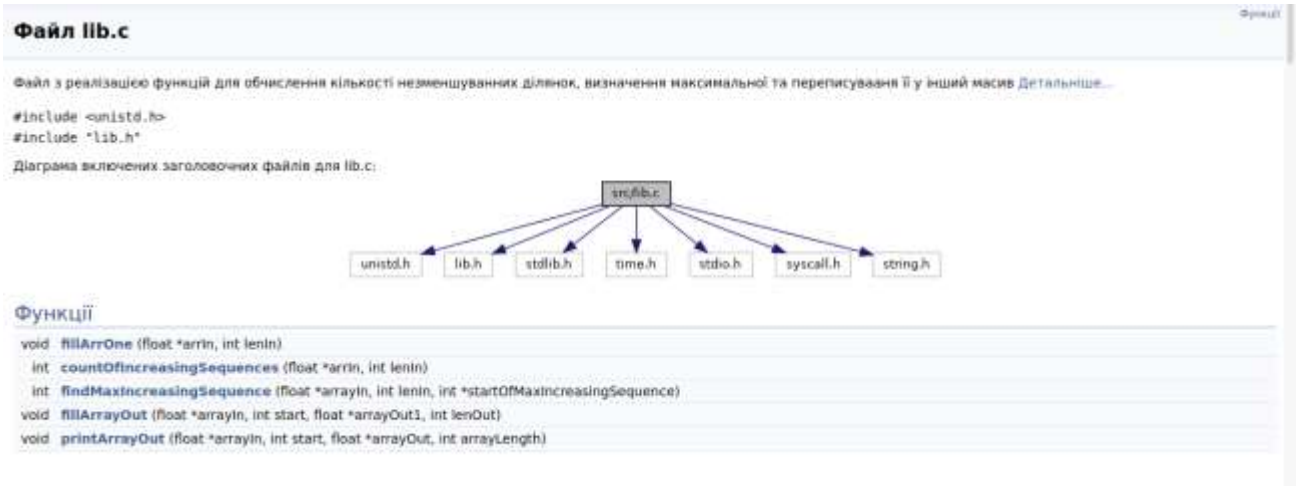


Рис. 8. Коментарі до функцій файлу lib.c

2.5. Перевірка на утечки пам’яті за допомогою Valgrind:

```
==7495== HEAP SUMMARY:
==7495==     in use at exit: 0 bytes in 0 blocks
==7495==   total heap usage: 4 allocs, 4 frees, 2,076 bytes allocated
==7495==
==7495== All heap blocks were freed -- no leaks are possible
==7495==
==7495== For lists of detected and suppressed errors, rerun with: -s
==7495== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Рис. 9. Перевірка на утечки пам’яті

3. ОПИС ПРОГРАМИ (ЛАБОРАТОРНА РОБОТА №14)

3.1. Функціональне призначення

Програму доцільно використовувати для розрахування частоти появи у даному тексті конкретного символу.

3.2. Опис логічної структури

Функція 'main' виділяє пам'ять для заданого і результуючого масиву, викликає усі функції для обчислення частоти. Схема алгоритму функції наведена на рис. 10.

Функція listdir обчислює та виводитть структуру файлів та каталогів та її розмір. Схема алгоритму функції наведена на рис. 11.

Функція readFromFile зчитує початкову інформацію з файлу. Схема алгоритму функції наведена на рис. 12.

Функція writeToFile записує результат обчислення у файл та виводить в консоль. Схема алгоритму функції наведена на рис. 13.

Функція countTextLength обчислює довжину заданого масиву. Схема алгоритму функції наведена на рис. 14.

Функція checker перевіряє кожен елемент на повтори. Схема алгоритму функції наведена на рис. 15.

Функція countOfUniqueElements обчислює кількість унікальних елементів. Схема алгоритму функції наведена на рис. 16.

Функція getsymbols переписує унікальні елементи в масив. Схема алгоритму функції наведена на рис. 17.

Функція getSymbolsCounts отримує кількість повторів кожного елементу. Схема алгоритму функції наведена на рис. 18.

Функція fillZeros ініціолізує результуючий масив. Схема алгоритму функції наведена на рис. 19.

Функція getSymbolsFrequencies вираховує та записує в масив частоту появи кожного елементу. Схема алгоритму функції наведена на рис. 20.

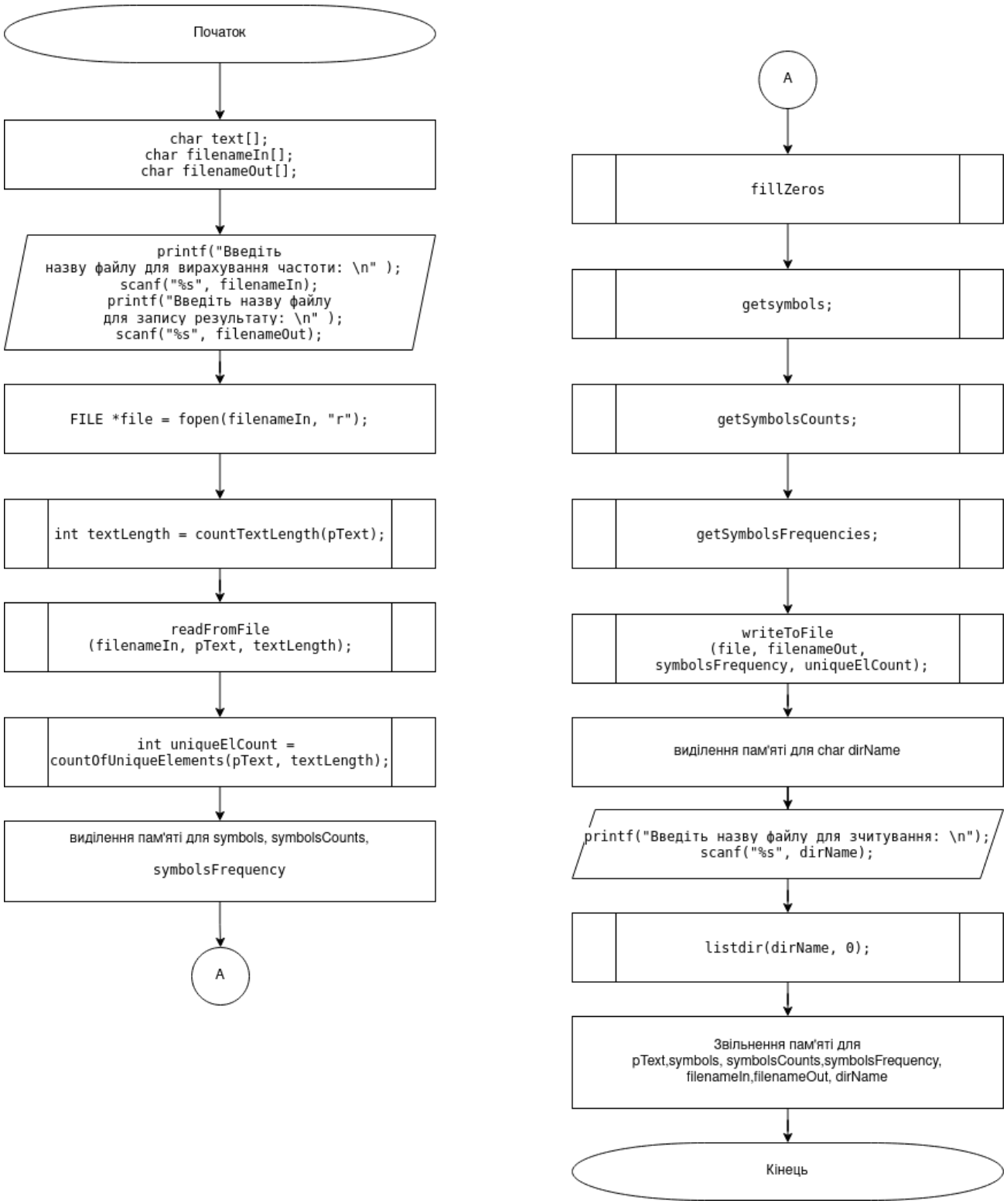


Рис.10. Схема алгоритму функції main

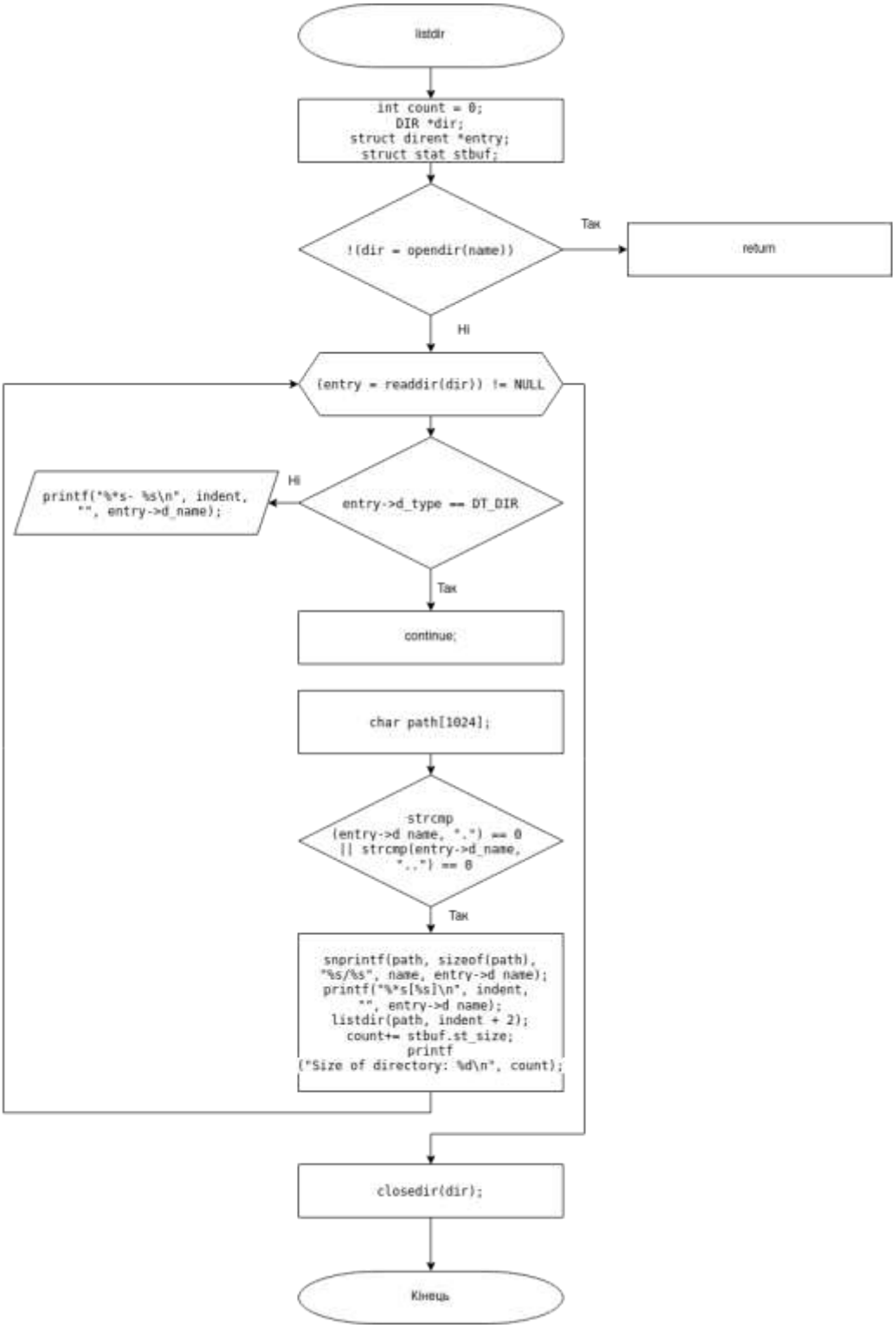


Рис.11. Схема алгоритму функції listdir

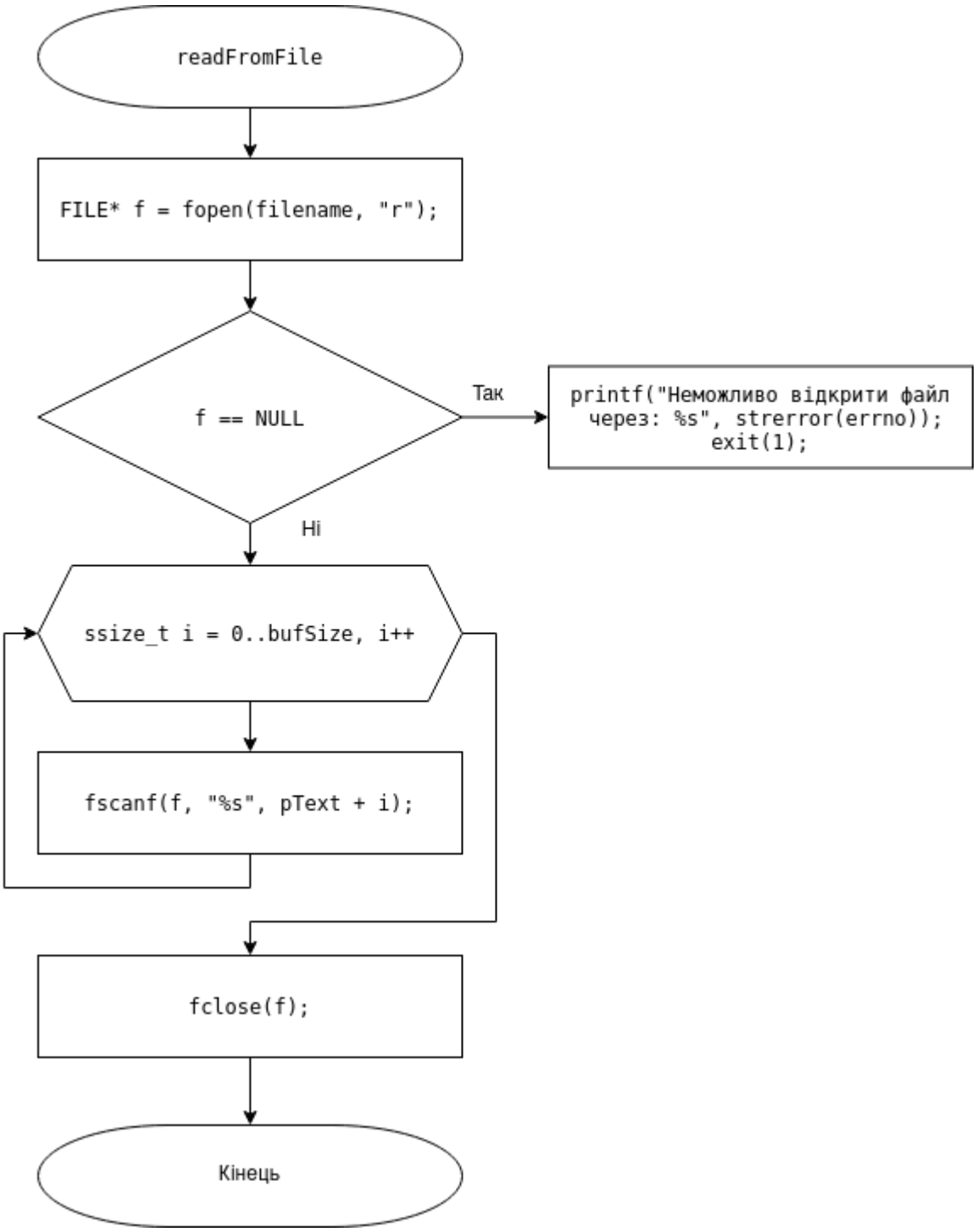


Рис.12. Схема алгоритму функції readFromFile

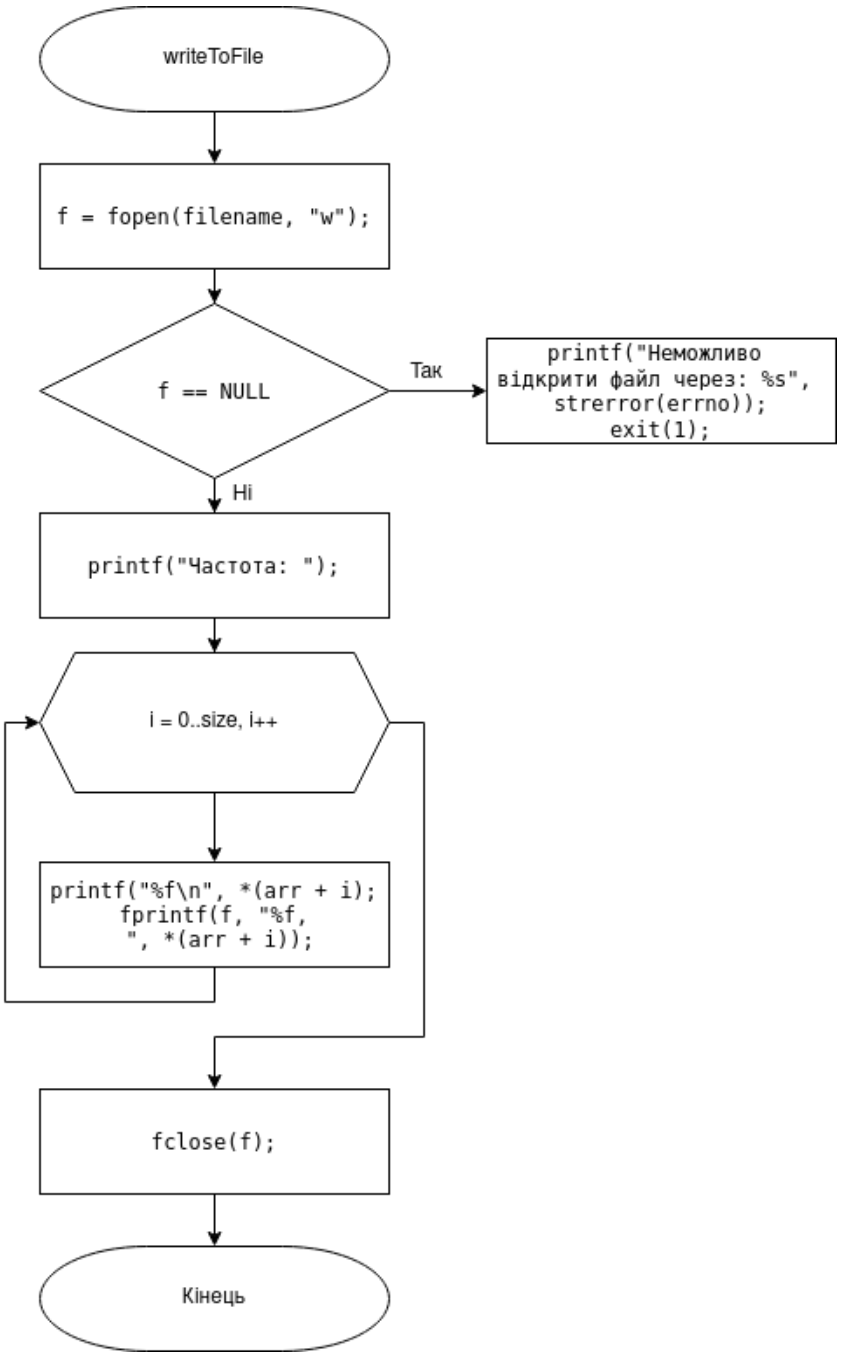


Рис.13. Схема алгоритму функції writeToFile

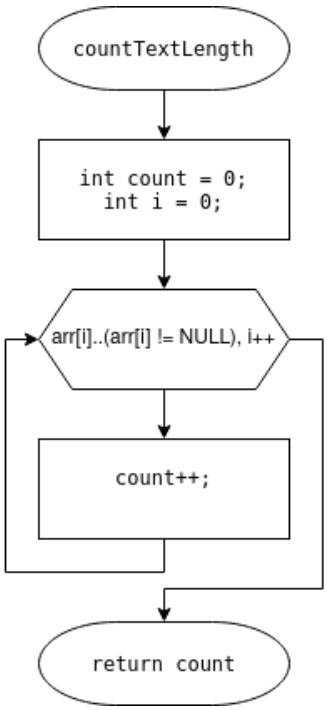


Рис.14. Схема алгоритму функції CountTextLength.

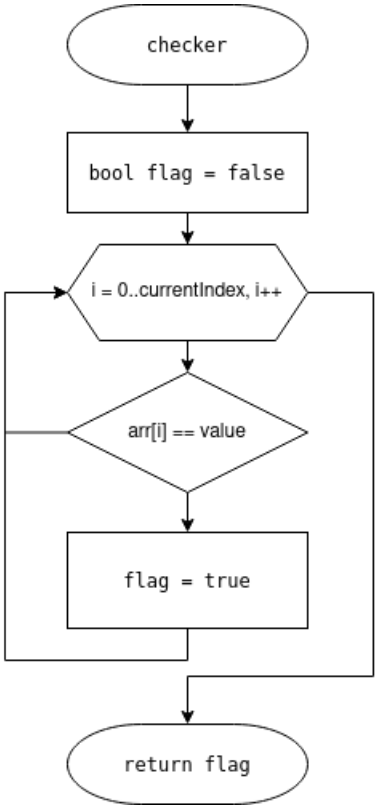


Рис.15. Схема алгоритму функції checker

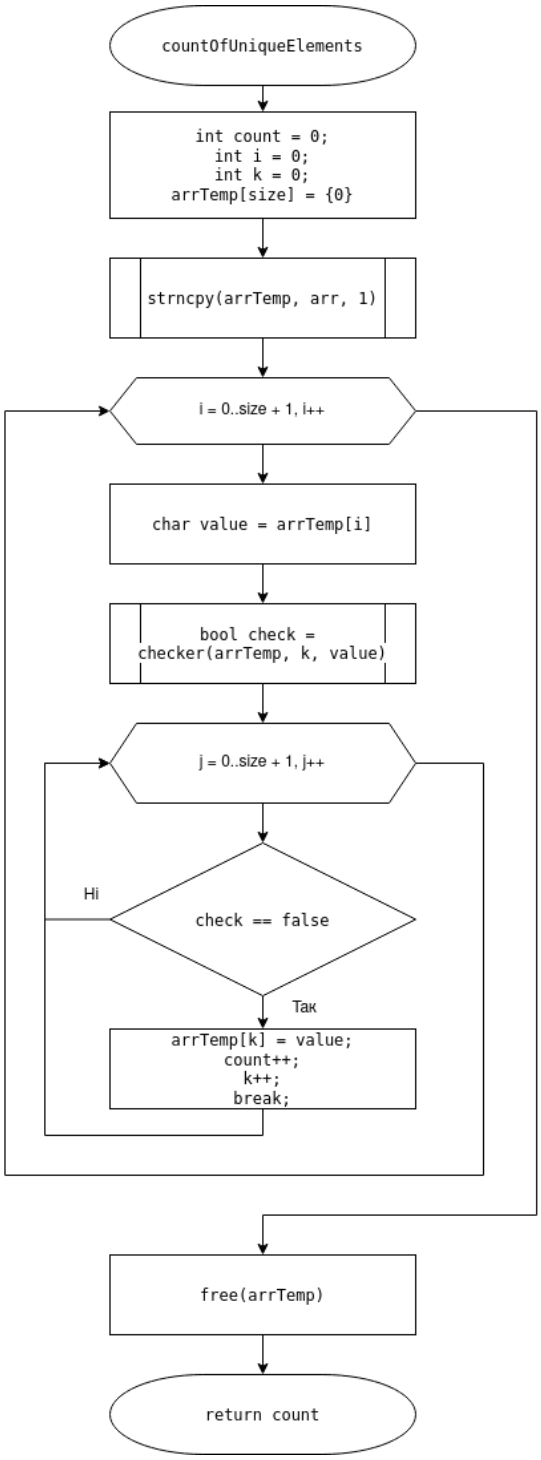


Рис.16. Схема алгоритму функції CountOfUniqueElements.

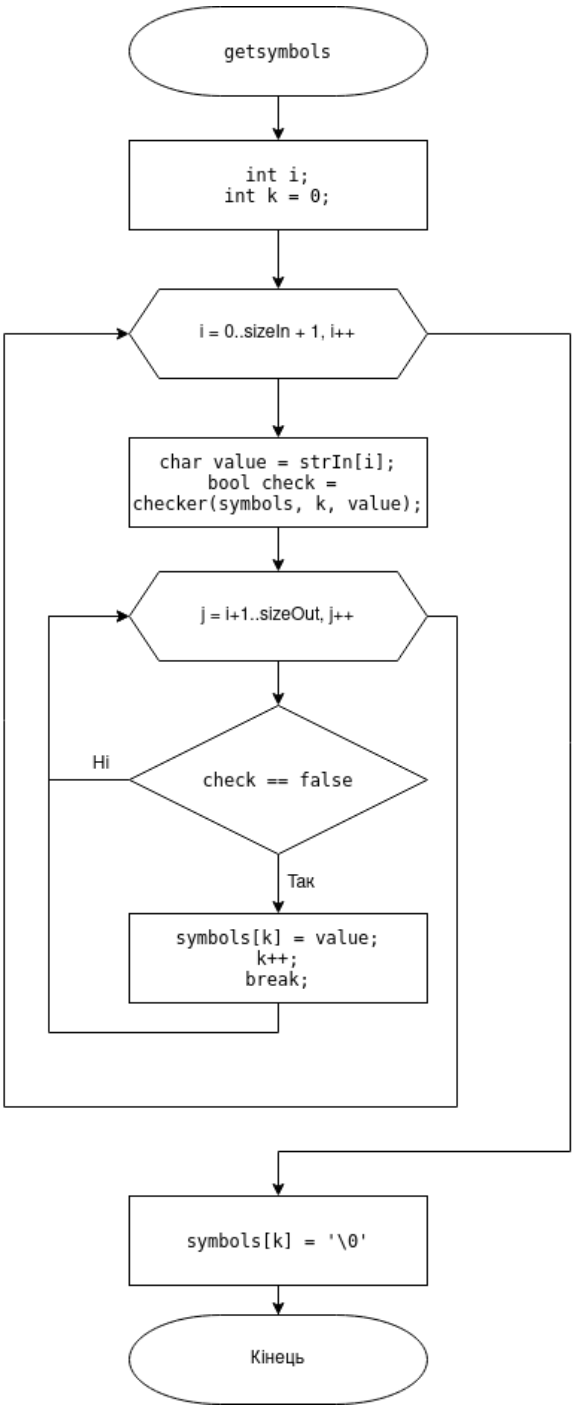


Рис.17. Схема алгоритму функції getsymbols.

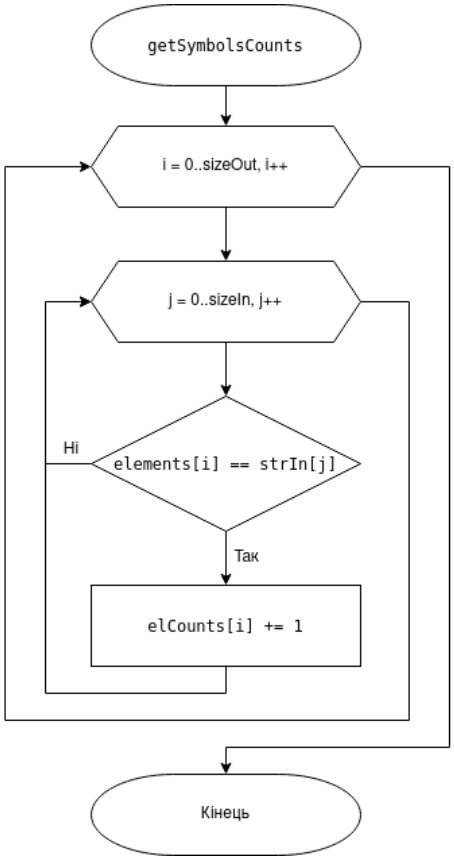


Рис.18. Схема алгоритму функції getSymbolsCounts.

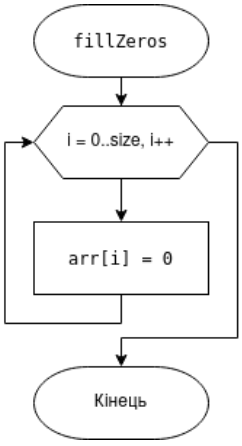


Рис.19. Схема алгоритму функції filZeros.

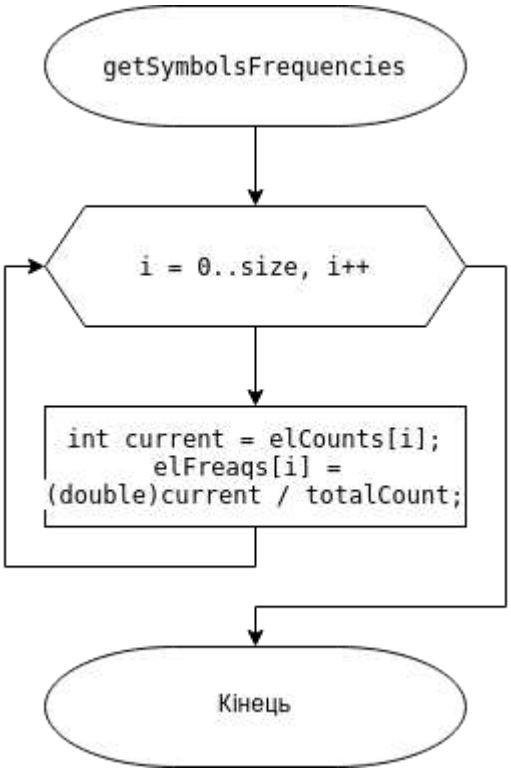


Рис.20. Схема алгоритму функції getSymbolFrequencies.

3.3. Структура проекту

```
.
├── dist
│   └── main3.bin
├── doc
│   ├── assets
│   │   ├── checker.png
│   │   ├── countOfUniqueElements.png
│   │   ├── countTextLength.png
│   │   ├── Doxygen1.png
│   │   ├── Doxygen2.png
│   │   ├── fillZeros.png
│   │   ├── frecuencyInFile.png
│   │   ├── frequency.png
│   │   ├── getSymbolsCounts.png
│   │   ├── getSymbolsFrequencies.png
│   │   ├── getsymbols.png
│   │   ├── listdir.png
│   │   ├── main.c.png
│   │   ├── readFromFile.png
│   │   ├── text.png
│   │   ├── tree.png
│   │   └── writeToFile.png
│   └── lab12_14.docx
├── Makefile
└── src
    ├── lib.c
    ├── lib.h
    └── main.c
```

3.4. Генерування Doxygen-документації

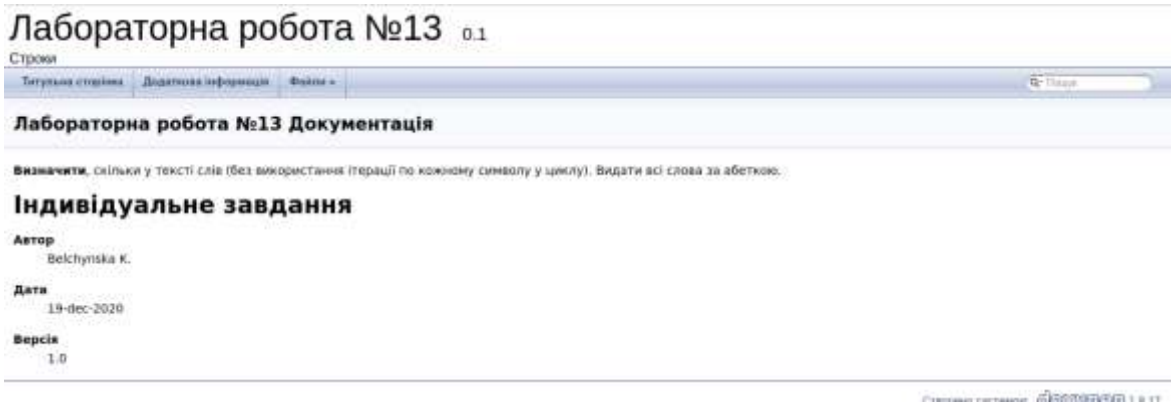


Рис. 16. Титульна сторінка Doxygen

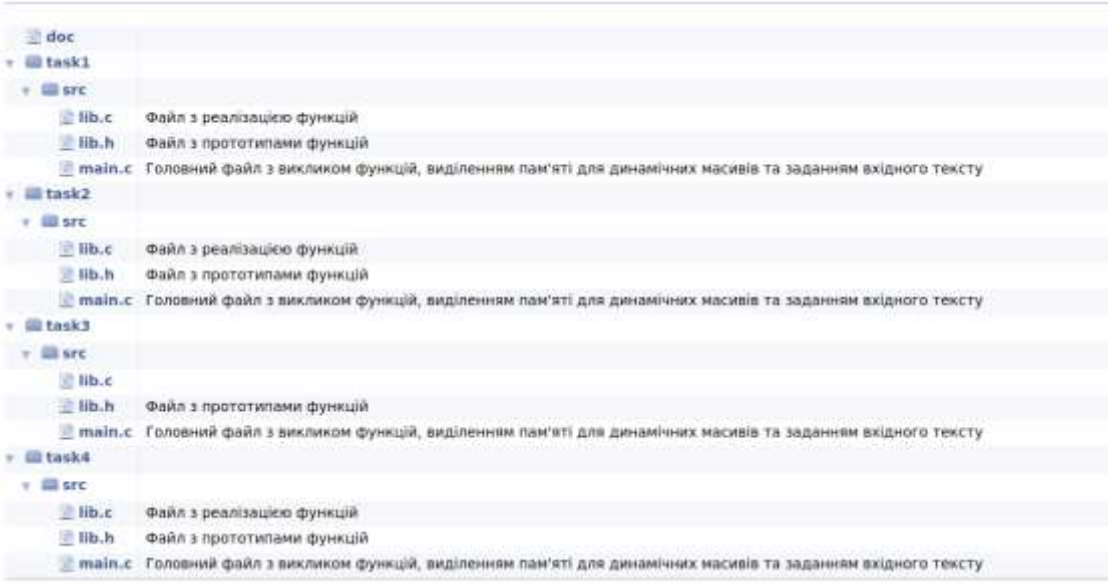
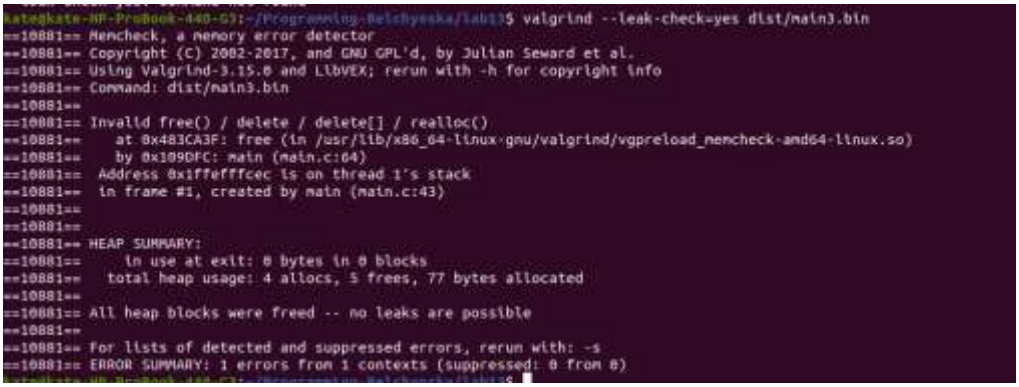


Рис. 21. Структура файлів в Doxygen.

3.5. Перевірка на утечки пам’яті за допомогою Valgrind:

Рис. 22. Перевірка на утечки пам’яті



4. ВАРІАНТИ ВИКОРИСТАННЯ (ЛАБОРАТОРНА РОБОТА №12)

   Програма виводить на екран послідовність речовинних чисел та довжину найбльшої послідовності:

```
Результуючий масив:

Результуючий масив:
122.000000
3.211100
3.222200

Process finished with exit code 0
```

Рис. 23. Вивід результату

5. ВАРІАНТИ ВИКОРИСТАННЯ (ЛАБОРАТОРНА РОБОТА №14)

   Програма зчитує інформацію з файлу та виводить результат у файл та консоль.

```
1 sfhossp'"jpo
```

Рис.24. Вхідний текст у файлі

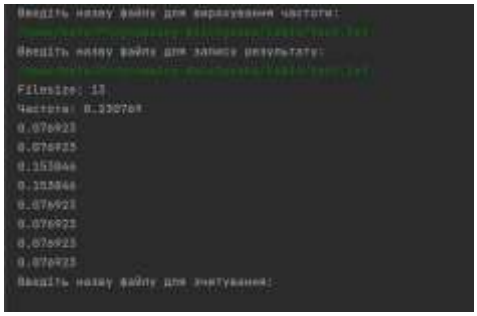


Рис. 25. Вивід результату обчислення у консолью

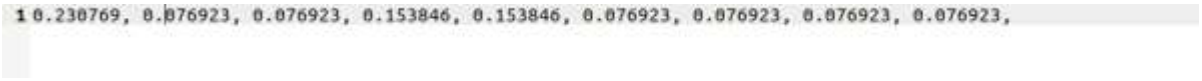


Рис.26. Вивід результату у файл

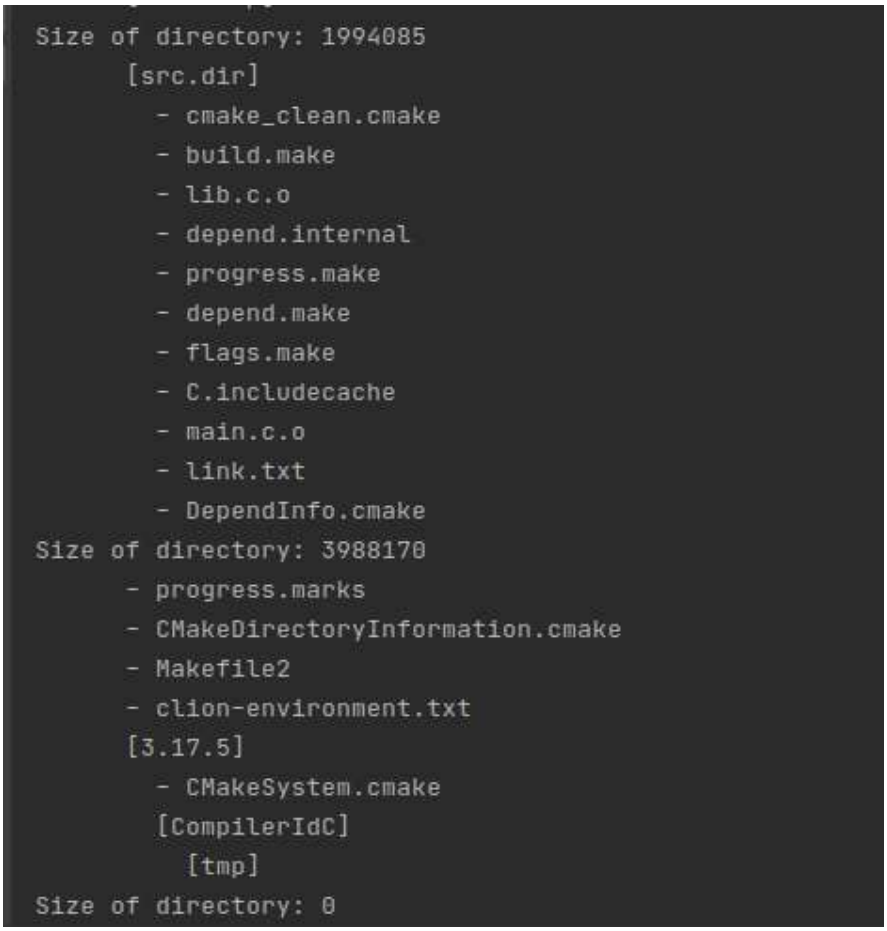


Рис.27. Вивід структури файлів як утиліта tree та вивід розміру файлів

ВИСНОВКИ

Лабораторна робота №12: в ході даної лабораторноґ роботи були використані функції для неформатованого низькорівневого та високорівневого вводу/виводу та форматованого вводу/виводу.

Лабораторна робота №14: в ході даної лабораторної роботи було досліджено роботу з файлами, викистання початкових даних з файлу та вивід результату у файл.