# Лабораторна робота № 13. Строки (Null-terminated C Strings)

# 1 Вимоги

## 1.1 Розробник

- Радєвич Владислав Романович;
- студент групи КІТ – 320;
- 22.12.2020 р.

## 1.2 Загальне завдання

Розробити програми, умови яких надано у лабораторному практикуму. Мною було взято умови для розробки з розділу на оцінку «відмінно».

## 1.3 Індивідуальне завдання

Зробити звіт за обраним мною варіантом. На даний момент це завдання номер 3.

# 2 Опис програми

## 2.1 Функціональне призначення

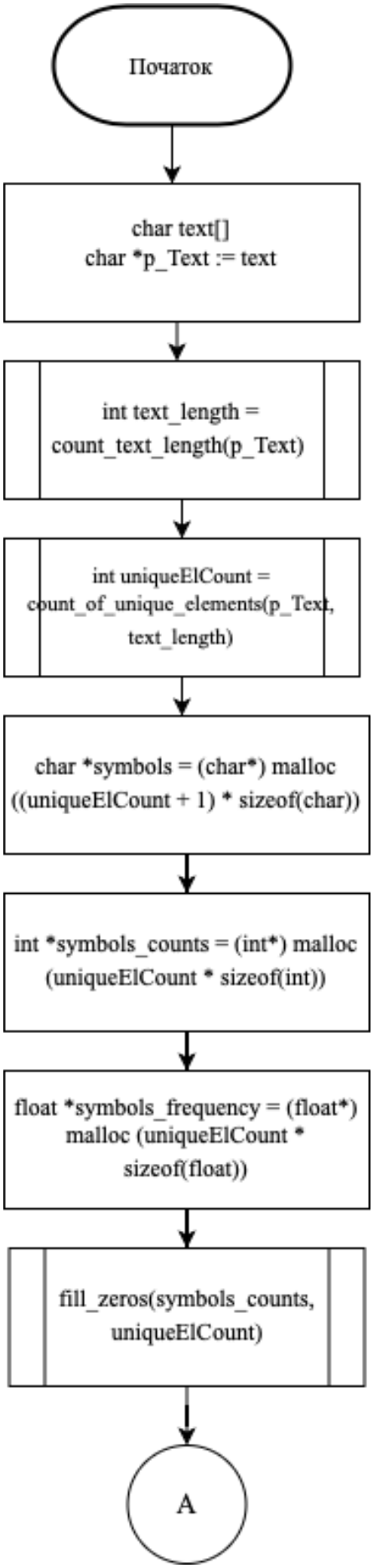
Програма призначена вирахування частотної таблиці символі у заданому тексті за допомогою функціям роботи зі рядками.

## 2.2 Опис логічної структури

### Основна функція

`int main`

*Призначення*: головна функція
*Схема алгоритму функції* подана на рис. 1
*Опис роботи*: задає розмір масивам, змінним, виділяючи їм пам'ять та звільня її. Викликає функції count_text_length, count_of_unique_elements, fill_zeros, get_symbols, get_symbols_counts, get_symbols_frequencies.


Початок
char text[]
char *p_Text := text
int text_length =
count_text_length(p_Text)
int uniqueElCount =
count_of_unique_elements(p_Text,
text_length)
char *symbols = (char*) malloc
((uniqueElCount + 1) * sizeof(char))
int *symbols_counts = (int*) malloc
(uniqueElCount * sizeof(int))
float *symbols_frequency = (float*)
malloc (uniqueElCount *
sizeof(float))
fill_zeros(symbols_counts,
uniqueElCount)
A

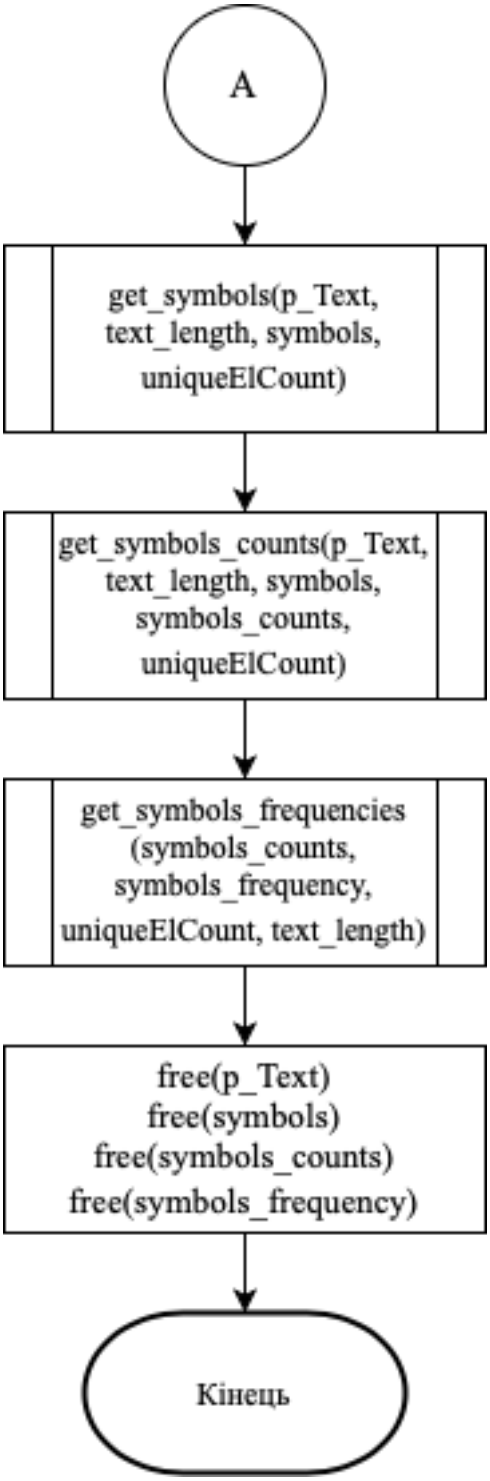



Рисунок 1 – схема алгоритму функції main

## Функція знаходження кількості слів

```
int count_text_length(char* arr);
```

*Призначення*: знаходження кількості слів у заданому тексті.
*Схема алгоритму функції* подана на рис. 2

*Опис роботи*: функція розділяє заданий текст на окремі слова, та рахує их.

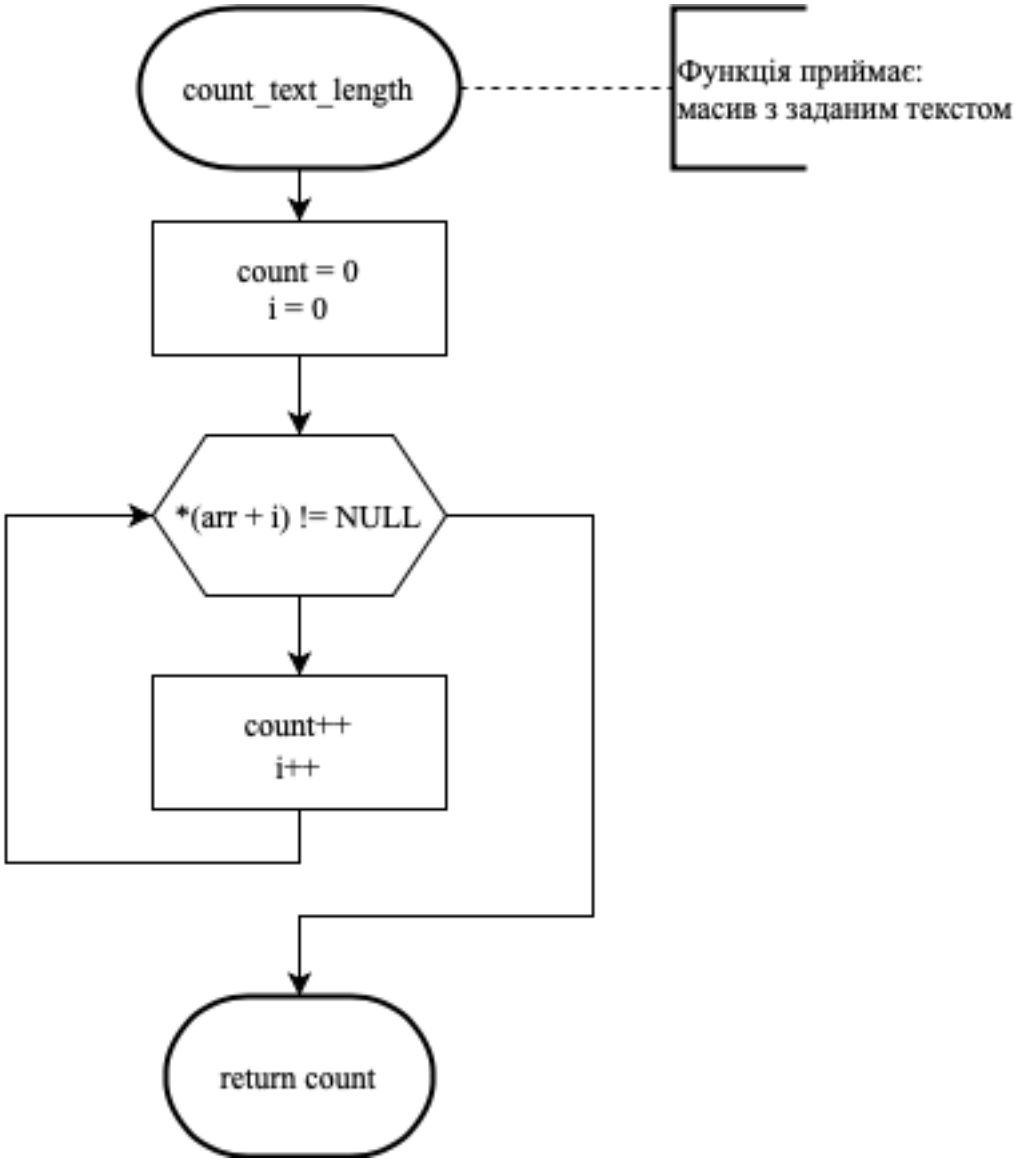



Рисунок 2 – схема алгоритму функції count_text_length

## Функція знаходження кількість символів, які дублюються та знаходить довжину рядка унікальних символів

```
int count_of_unique_elements(char* arr, int size);
```

*Призначення*: знаходження всіх символів, що дублюються в заданому рядку та знаходження кількості унікальних символів, що є результатом функції.

*Схема алгоритму функції* подана на рис. 3

*Опис роботи*: функція знаходе всі символи, що дублюються в заданому рядку та знаходить кількості унікальних символів, що є результатом функції.

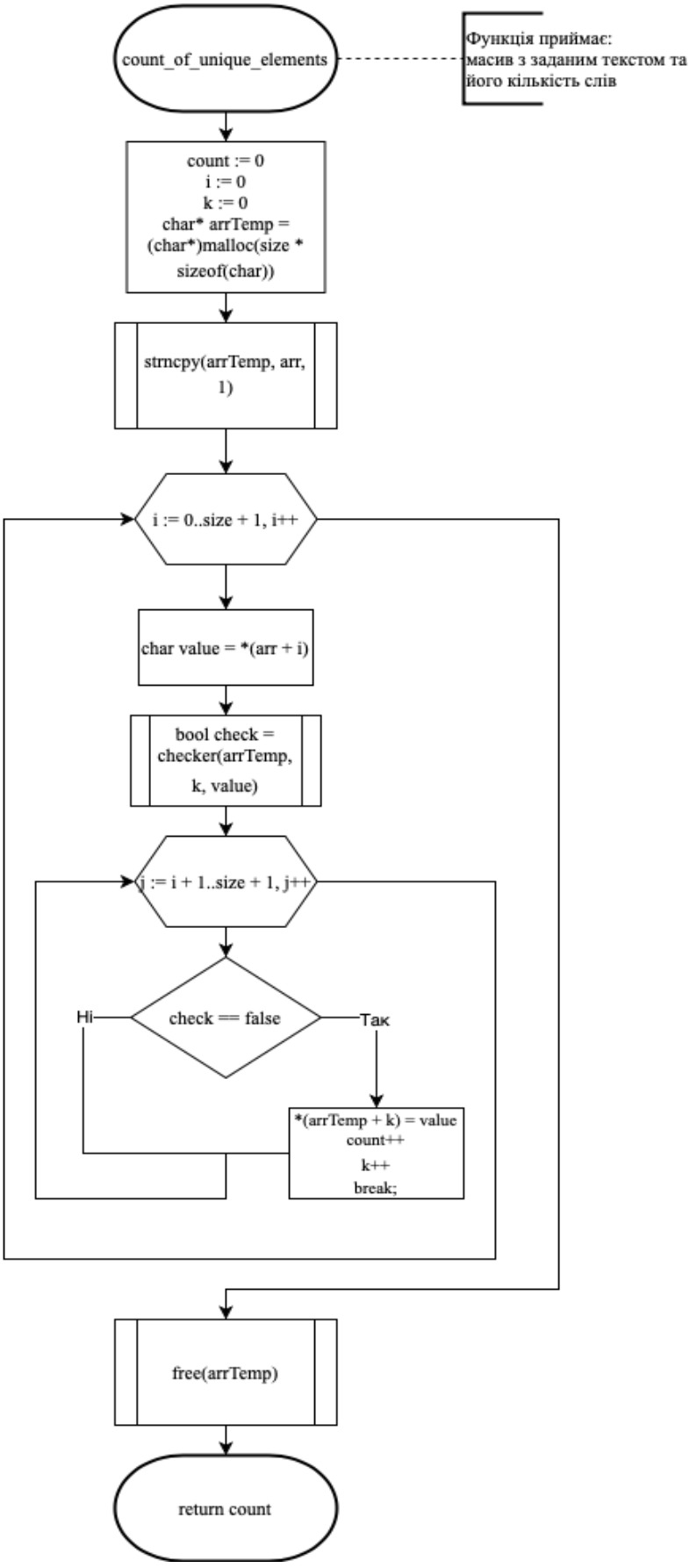



Рисунок 3 – схема count_of_unique_elements

## Допоміжна функція

```
bool checker(char *arr, int currentIndex, char value);
```

*Призначення*: знаходження символів, що дублюються.

*Схема алгоритму функції* подана на рис. 4

*Опис роботи*: функція перевіряє кожен елемента масиву на повтор: результатом функції є логічний тип, які набуває значення «так» або «ні».

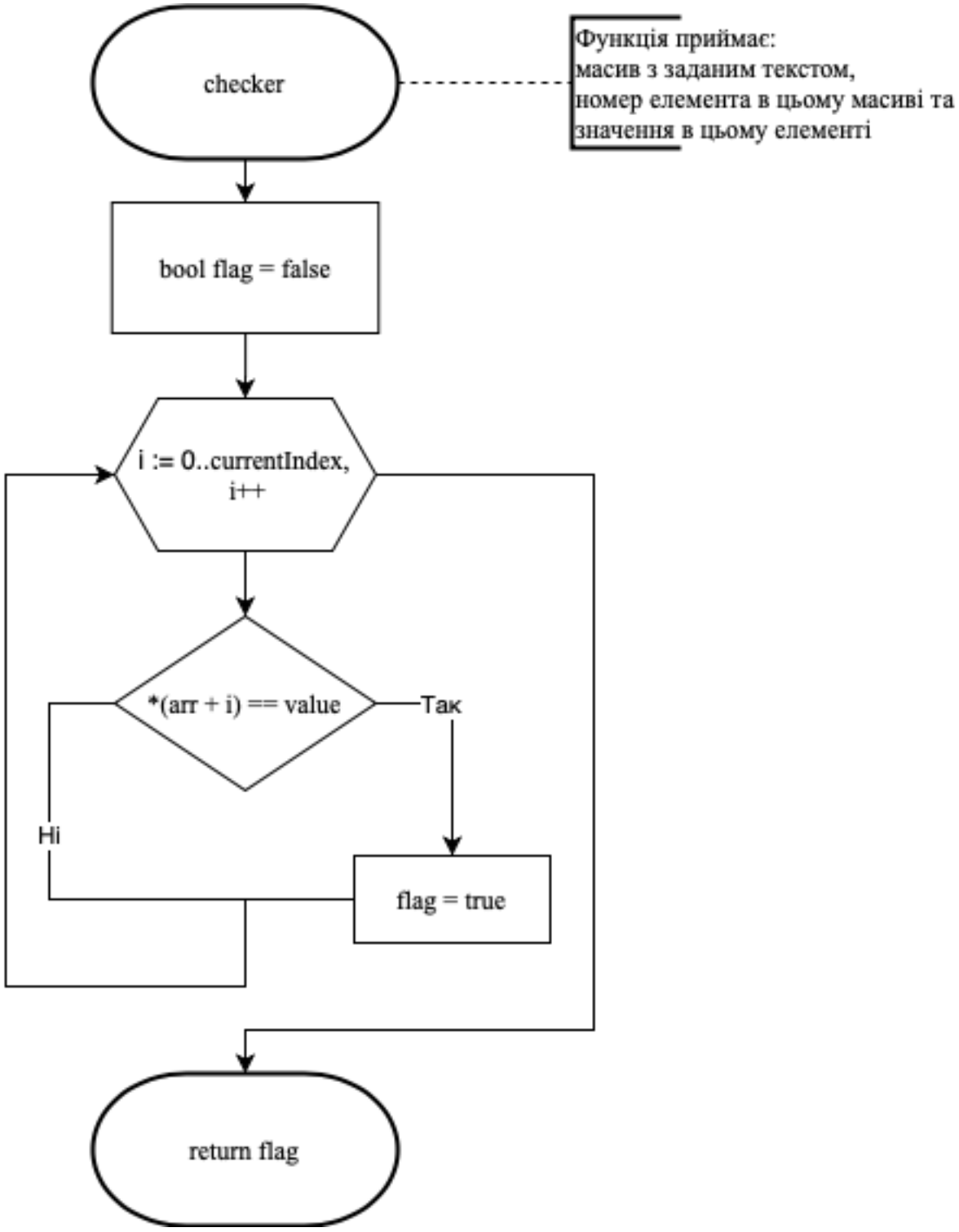



Рисунок 4 – схема алгоритму функції checker

## Функція очищення масиву

```
void fill_zeros(int *arr, int size);
```

*Призначення*: очистити масив, заповнюючи його нулями.
*Схема алгоритму функції* подана на рис. 5

*Опис роботи*: функція присвоює кожному елементу масиву значення 0.

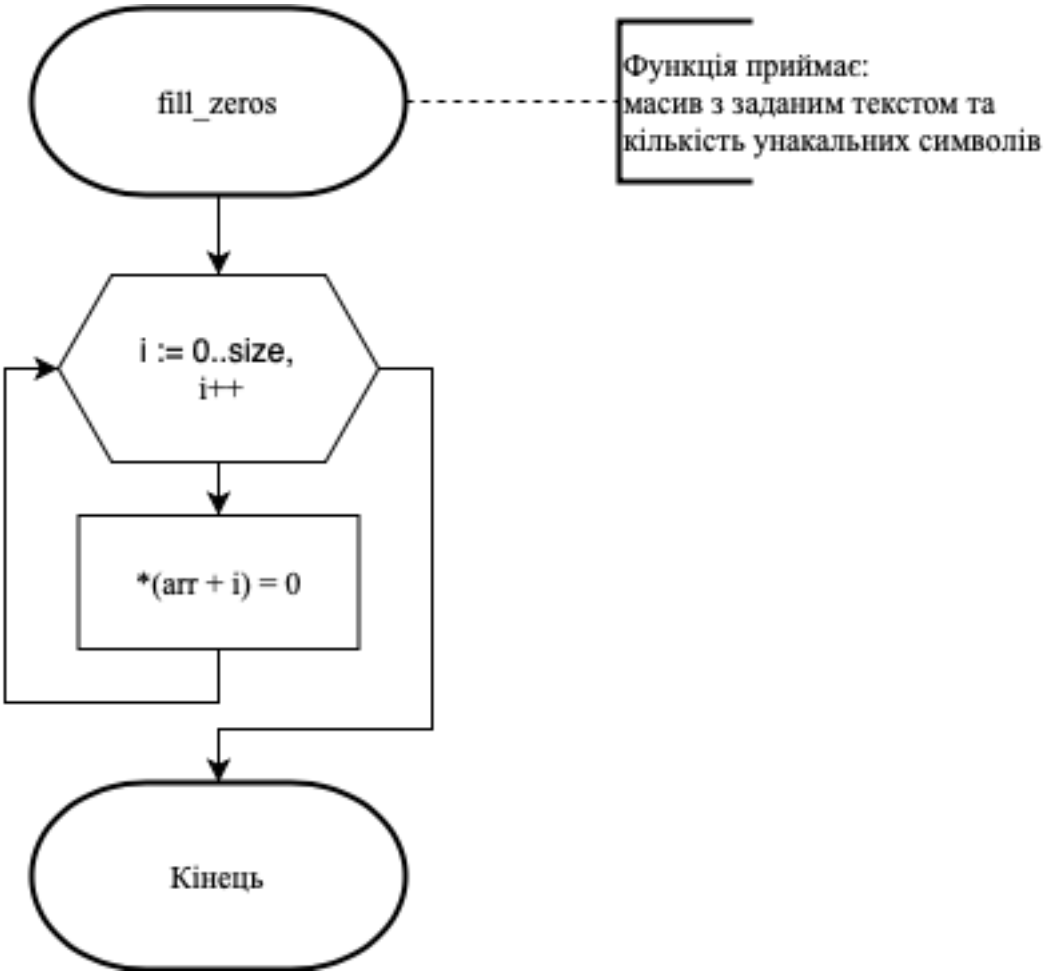



Рисунок 5 – схема алгоритму функції fill_zeros

## Функція виділення унікальних символів

```
void get_symbols(char *strIn, int sizeIn, char *symbols);
```

*Призначення*: виділення перших унікальних символів зі заданого тексту.

*Схема алгоритму функції* подана на рис. 6

*Опис роботи*: функція виділяє унікальні символи з заданого тексту за переписує їх в окремий масив.

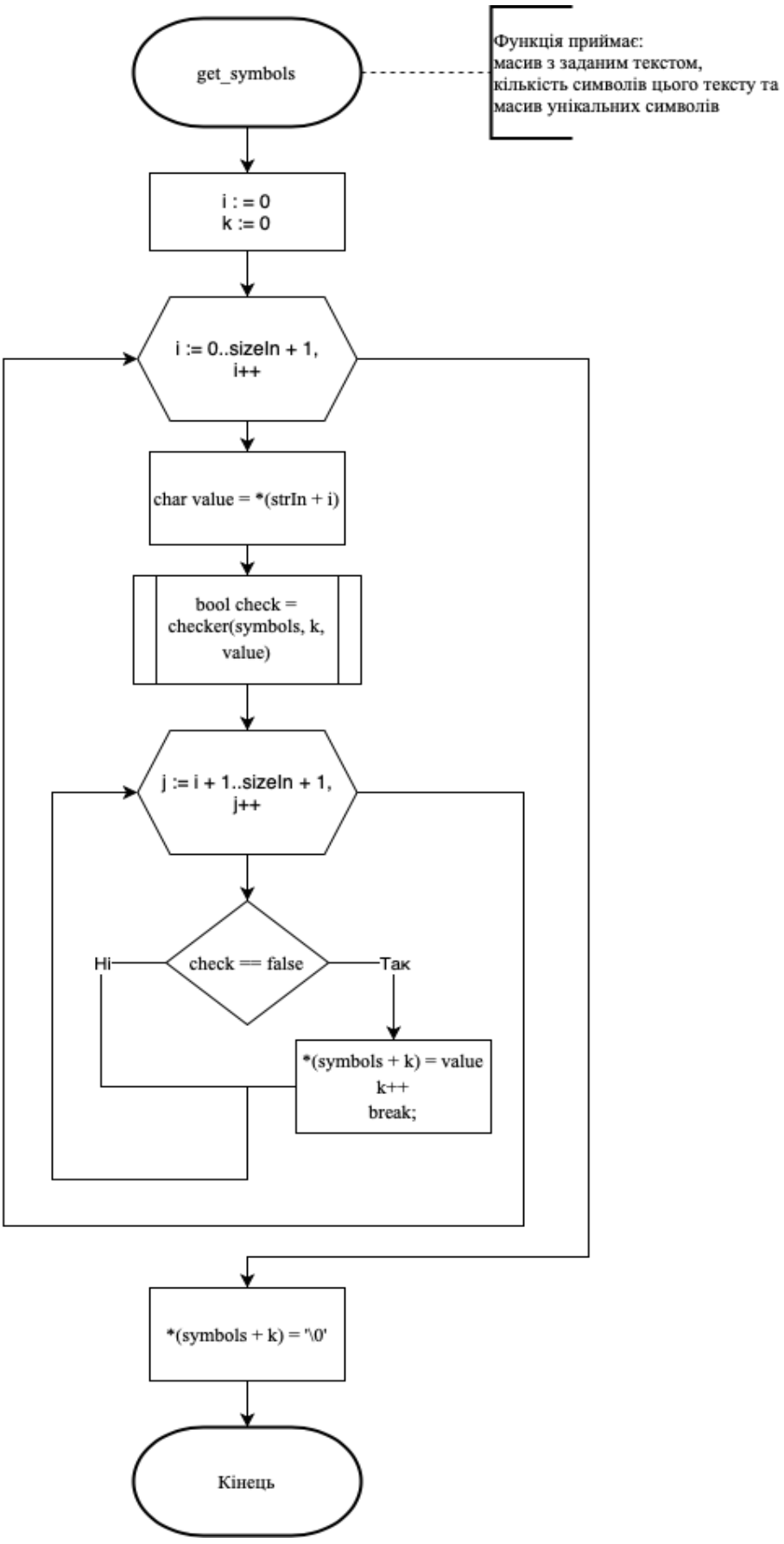



Рисунок 6 – схема алгоритму функції get_symbols

## Функція присвоєння кількості

```
void get_symbols_counts(char* strIn, int sizeIn, char* elements, int*
elCounts, int sizeOut);
```

*Призначення*: визначити кількість символів, що повторюються та записати це значення в окремий масив.

*Схема алгоритму функції* подана на рис. 7

*Опис роботи*: функція присвоює кожному унікальному символу його кількість повторення до переписання тексту в один рядок тільки з унікальними символами.

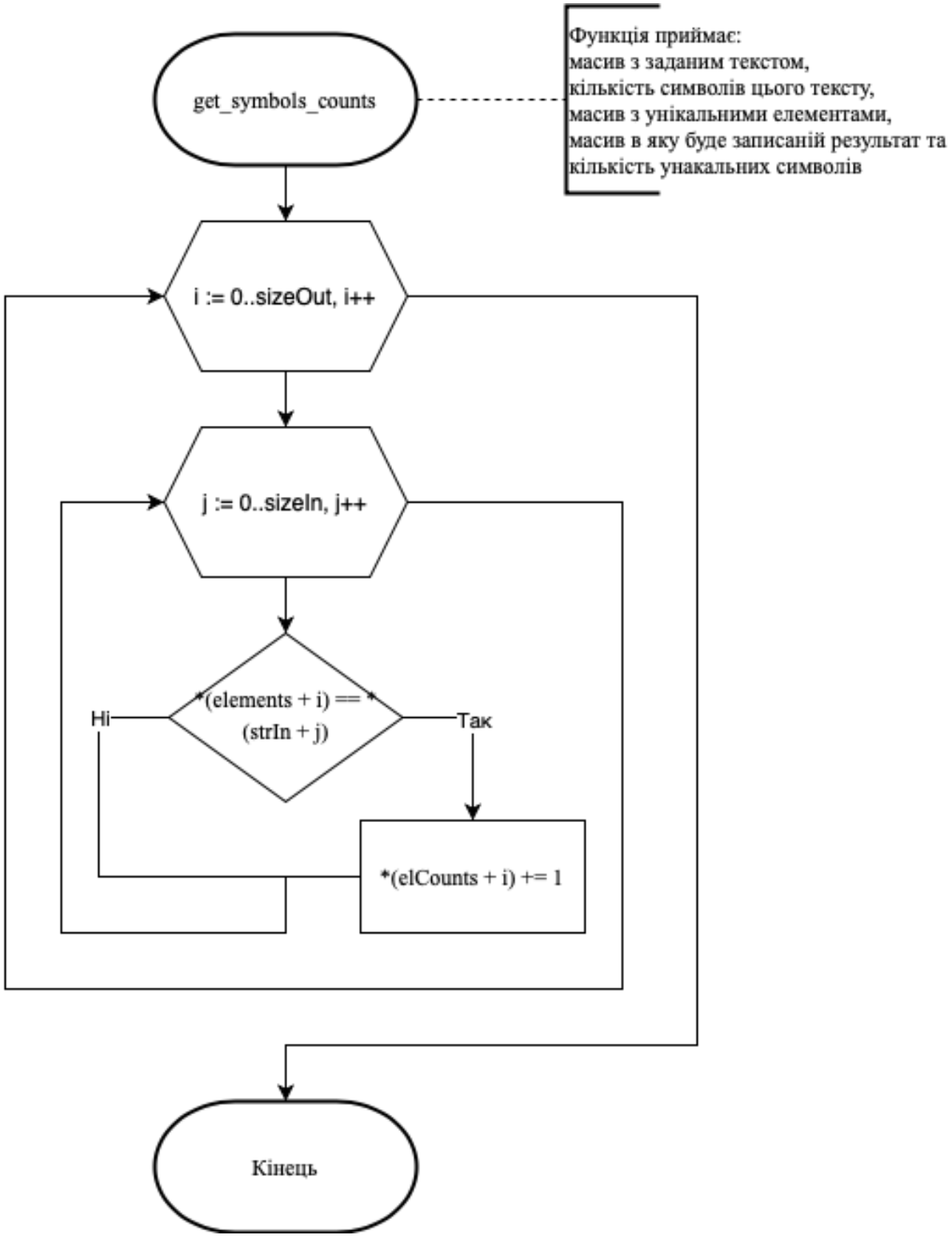



Рисунок 7 – схема алгоритму функції get_symbols_counts

## Функція визначення частоти

```
void get_symbols_frequencies(int* elCounts, float* elFreaqs, int size, int
totalCount);
```

*Призначення*: визначити частоту символів, що повторюються та записати це значення в окремий масив.

*Схема алгоритму функції* подана на рис. 8

*Опис роботи*: функція присвоює кожному унікальному символу його частота повторення до переписання тексту в один рядок тільки з унікальними символами.

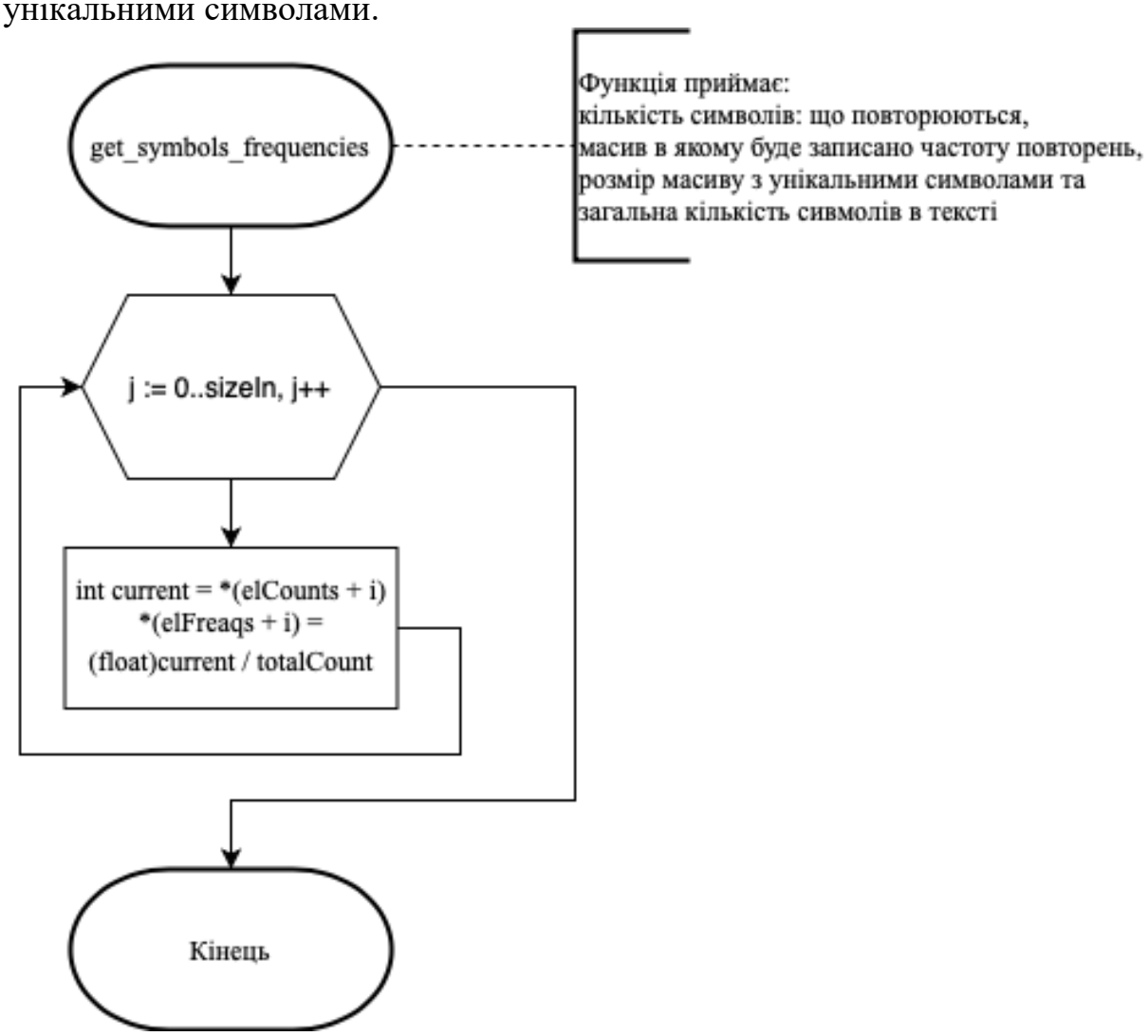



Рисунок 8 – схема алгоритму функції get_symbols_frequencies

## Структура проекту

```
.
├── 1
│   ├── README.md
│   └── src
│       ├── lib.c
│       ├── lib.h
│       └── main.c
├── 2
│   ├── README.md
│   └── src
│       ├── lib.c
│       ├── lib.h
```

```
│       └── main.c
├── 3
│   ├── README.md
│   └── src
│       ├── lib.c
│       ├── lib.h
│       └── main.c
├── doc
│   ├── assets
│   │   ├── 9.png
│   │   ├── 10.png
│   │   ├── 11.png
│   │   ├── checker.png
│   │   ├── count_of_unique_elements.png
│   │   ├── count_text_length.png
│   │   ├── fill_zeros.png
│   │   ├── flowchart_main(1).png
│   │   ├── flowchart_main(2).png
│   │   ├── get_symbols_counts.png
│   │   ├── get_symbols_frequencies.png
│   │   └── get_symbols.png
│   ├── Lab13.md
│   └── Radievych13.pdf
├── Doxyfile
├── Makefile
└── README.md
```

# 3 Варіанти використання

Цю програму можна використовувати за для знаходження частоти повторення символів з заданому тексті.

Результат роботи з doxygen продемонстровано на рисунку 9, рисунку 10 та рисунку 11.

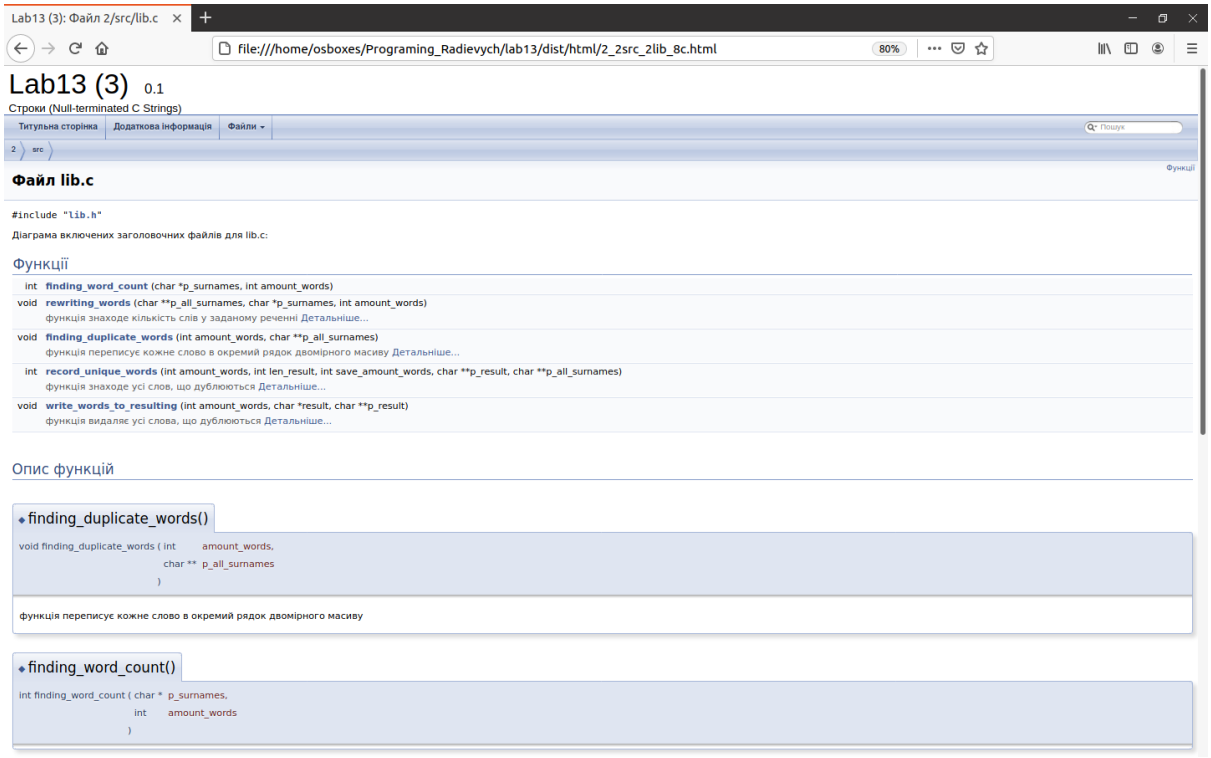



Рисунок 9 – робота з doxygen

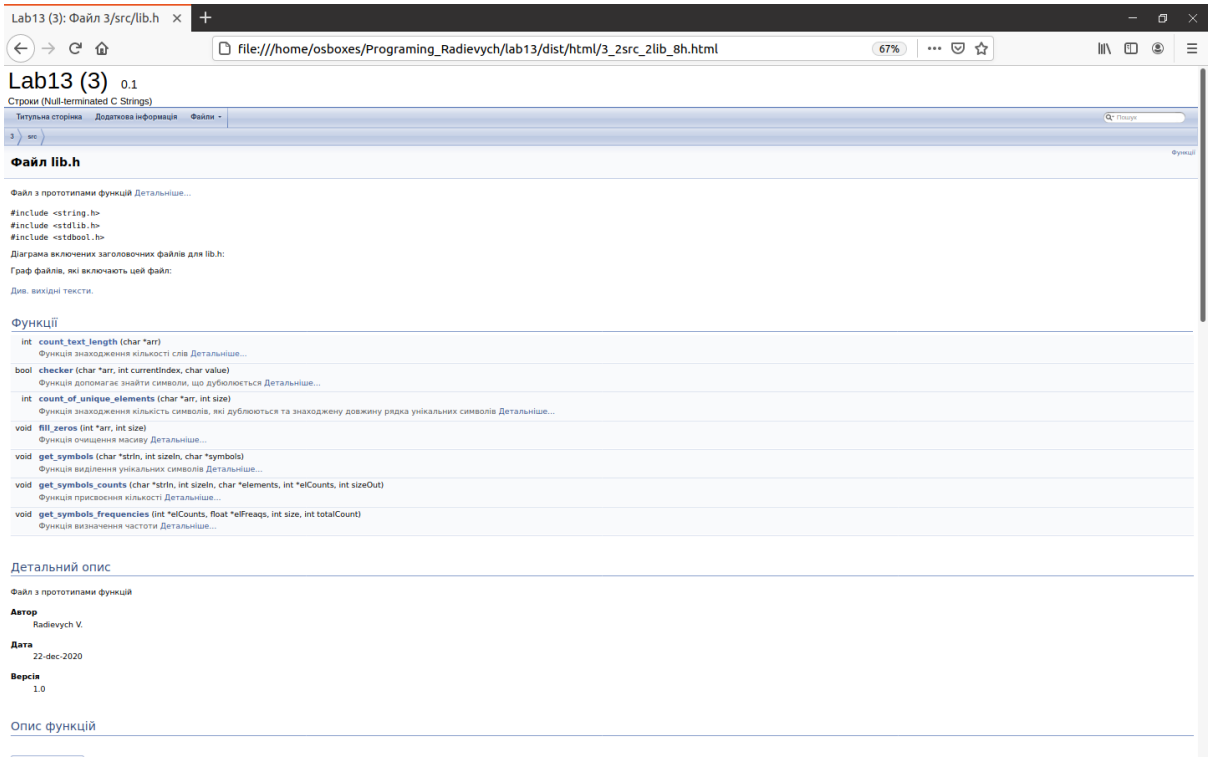



Рисунок 10 – робота з doxygen

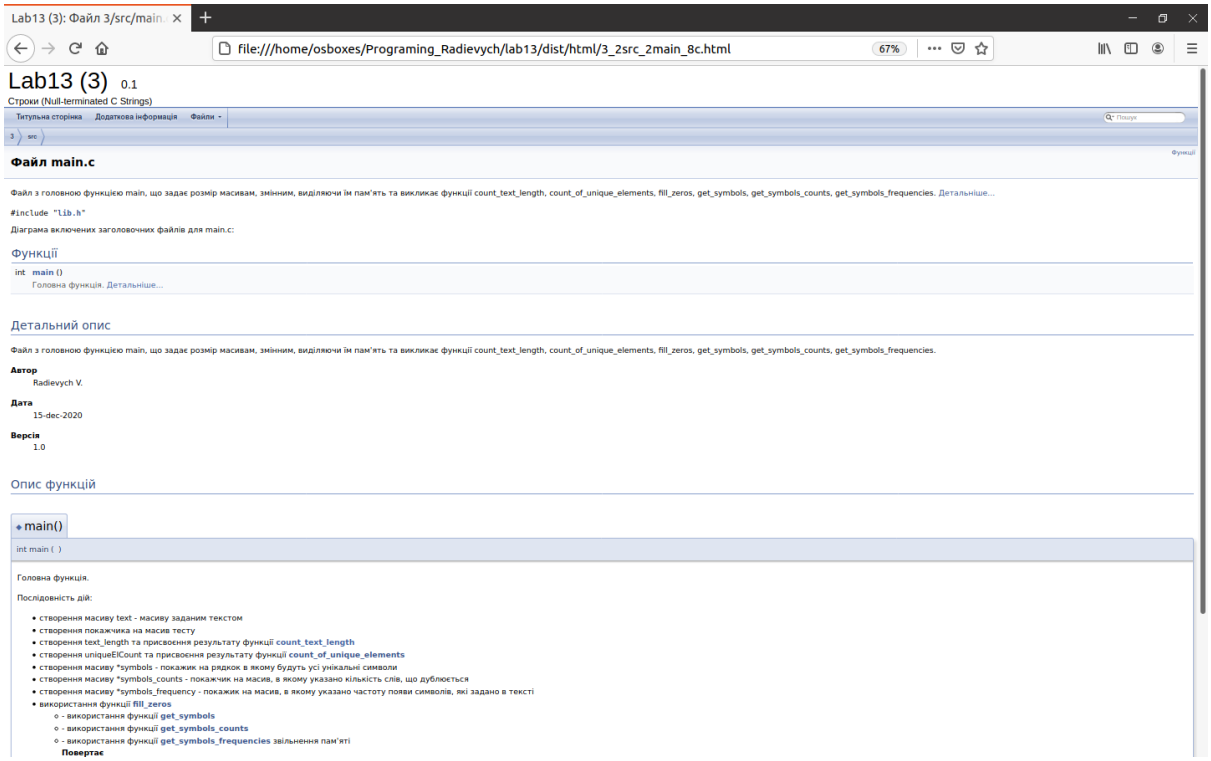



Рисунок 11 – робота з doxygen

# Висновок

При виконанні даної лабораторної роботи я закріпив набуті мною навички, створення програми, використовуючи функції роботи з рядками.

Посилання на GitHub, де знаходяться усі програми:

https://github.com/KotKHPI/Programming_Radievych