Міністрество освіти і науки України
Національний технічний унівеститет
«Харківський політехнічний інституту»
Кафедра «Обчислювальна техніка та програмування»

**ЗВІТ**
Про виконання лабораторної роботи № 10
«Вступ до документації коду»

Керівник: викладач
Бульба С. С.

Виконавець: студент гр. КІТ-120є
Куліш П.П.

Харків 2020

**Лабораторна робота № 10. Вступ до документації проекту.**

# 1 Вимоги

## 1.1 Розробник

- Куліш Павло Павлович;
- Студент групи КІТ-120є;
- 01-груд-2020.

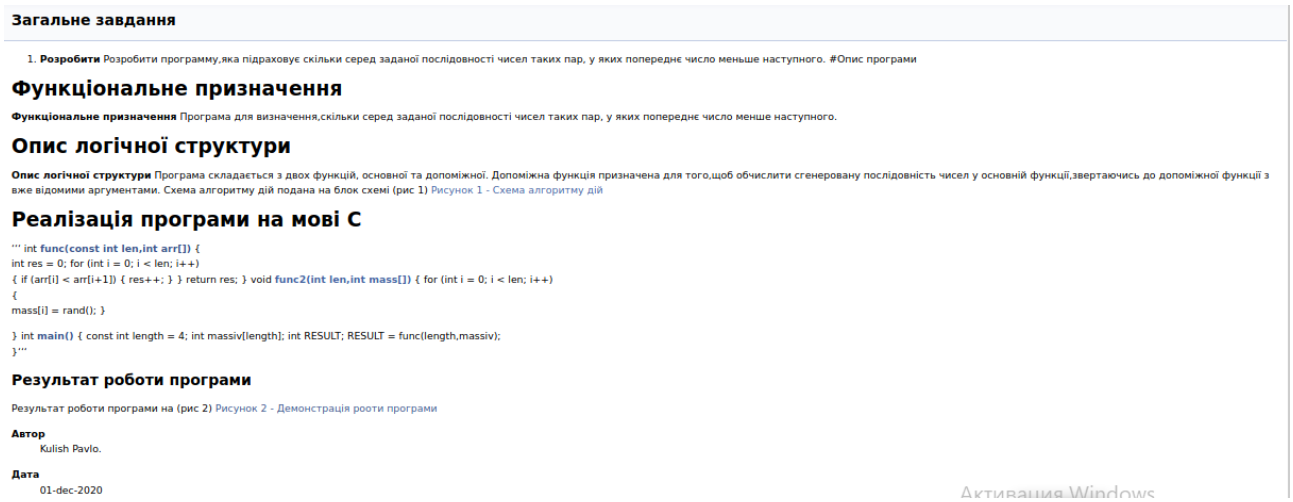
## 1.2 Загальне завдання

Розробити повноцінний звіт для лабораторної роботи, що присвячена функціям у двох форматів :

- Markdown
- doc Формат

# 2 Опис програм

**2.1.1** Опис завдання № 1 засобами Doxygen & MarkDown (рис 1), опис функцій та детальний опис (рис 2), схема алгоритму дій (рис 3).



**Загальне завдання**

1. **Розробити** Розробити программу,яка підраховує скільки серед заданої послідовності чисел таких пар, у яких попереднє число меньше наступного. #Опис програми

**Функціональне призначення**

**Функціональне призначення** Програма для визначення,скільки серед заданої послідовності чисел таких пар, у яких попереднє число менше наступного.

**Опис логічної структури**

**Опис логічної структури** Програма складається з двох функцій, основної та допоміжної. Допоміжна функція призначена для того,щоб обчислити сгенеровану послідовність чисел у основній функції,звертаючись до допоміжної функції з вже відомими аргументами. Схема алгоритму дій подана на блок схемі (рис 1) Рисунок 1 - Схема алгоритму дій

**Реалізація програми на мові С**

```
''' int func(const int len,int arr[]) {
int res = 0; for (int i = 0; i < len; i++)
{ if (arr[i] < arr[i+1]) { res++; } } return res; } void func2(int len,int mass[]) { for (int i = 0; i < len; i++)
{
mass[i] = rand(); }

} int main() { const int length = 4; int massiv[length]; int RESULT; RESULT = func(length,massiv);
}'''
```

**Результат роботи програми**

Результат роботи програми на (рис 2) Рисунок 2 - Демонстрація рооти програми

**Автор**
Kulish Pavlo.

**Дата**
01-dec-2020



Рисунок 1 – Опис завдання № 1 засобами Doxygen & Markdown

## Функції

int **func** (const int len, int arr[])

@func Головна функція, призначення якої підрахувати результат Детальніше...

void **func2** (int len, int mass[])

@func2 Функція псевдовипадкового генератору чисел rand() Детальніше...

int **main** ()

Основна функція, у якої йде звернення до функції @func. Детальніше...

## Детальний опис

Програма призначена для знаходження пар чисел, де попереднє число меньше наступного.

**Автор**

Kulish Pavlo

**Повертає**

Змінну з кількістю пар таких чисел.

**Версія**

0.1 date 23.11.2020

Рисунок 2 – Детальний опис, та опис функцій

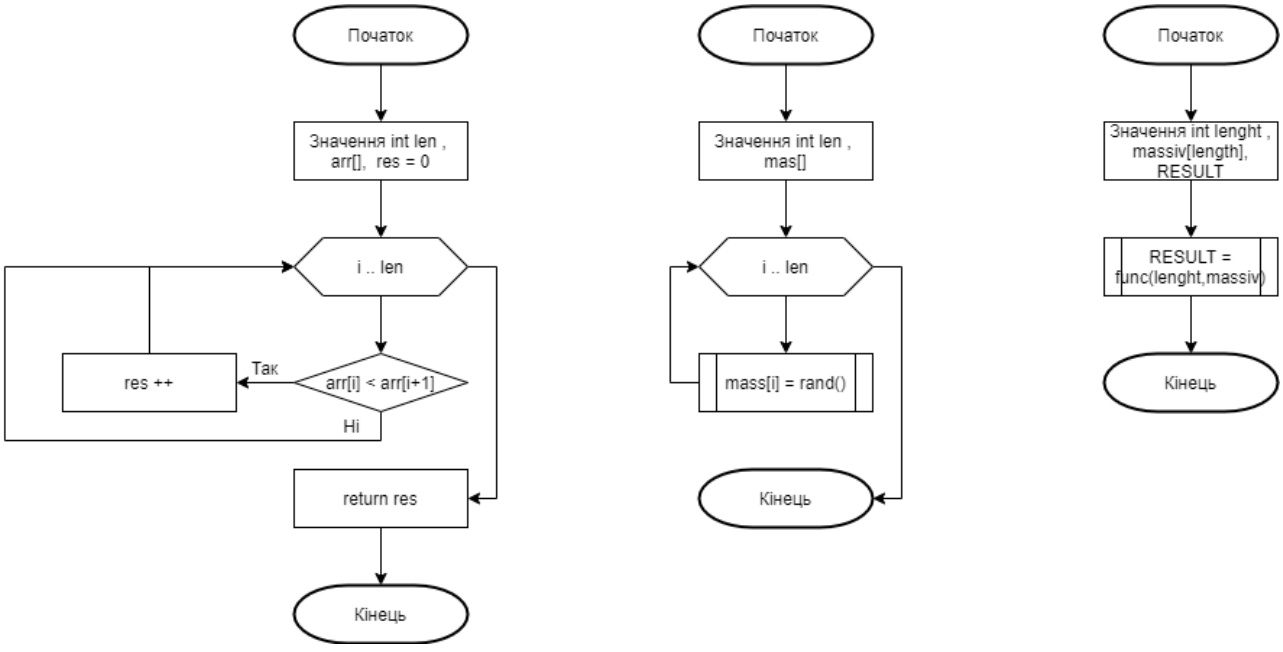



Рисунок 3 – Схема алгоритму дій

### 2.1.2 Реалізація програми на мові С ( рис 4) та дебагер (рис 5)

```
1
2  int func(const int len,int arr[])              // Головна функція, призначення якої підрахувати результат
3  {                                              // за умови, якщо попередній елемент массиву меньше наступного,
4          int res = 0;                           // то до кінцевого результату додається 1
5          for (int i = 0; i < len; i++)
6          {
7                  if (arr[i] < arr[i+1])
8                  {
9                          res++;
10                 }
11         }
12         return res;
13 }
14
15 int main()
16 {
17         const int length = 4;
18         int massiv[length];
19         int RESULT;
20
21         for (int i = 0; i < length; i++) // Псевдовипадковий генератор чисел rand()
22         {                                // заповнює масив довжиною length
23                 massiv[i] = rand();
24         }
25
26         RESULT = func(length,massiv);    // Кінцевий результат, обичислений за допомогою звернення
27 }                                        // До функції , яка використовує довжину массиву,
28                                         // та массив,для обчислення RESULT
29
```

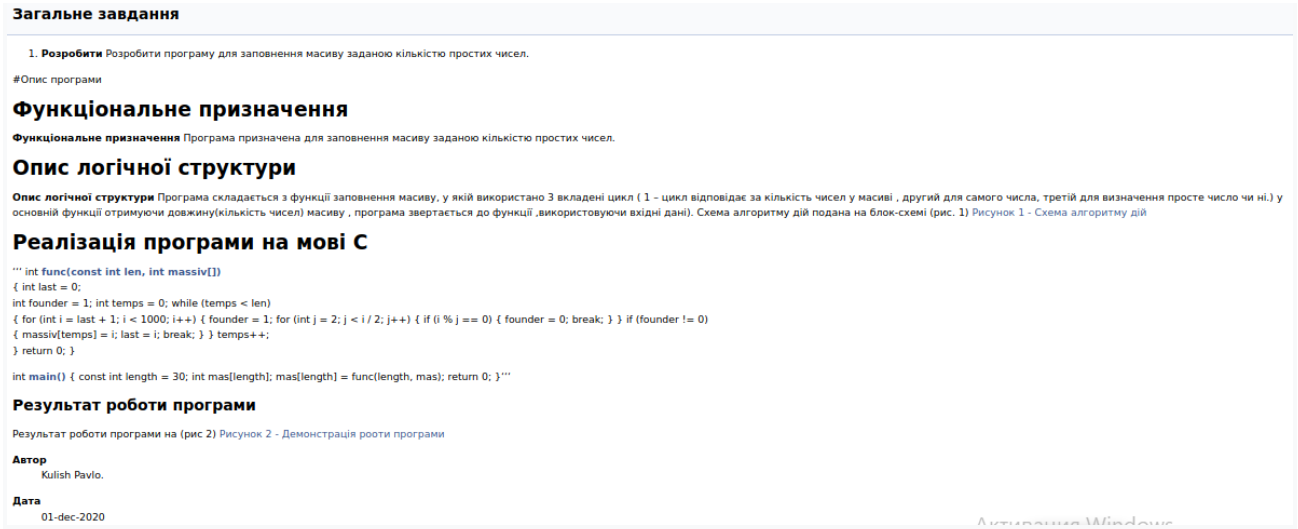
Рисунок 4 – Реалізована програма

| Переменная | Значение | Тип |
|---|---|---|
| ▼ Локальные переменные | | |
| length | 4 | const int |
| ▶ massiv | [1431655201] | int [93824992236321] |
| RESULT | 2 | int |
| Параметры функции | | |

Рисунок 5 – Демонстрація виконання через дебагер

### 2.2.1 Опис програми

Опис завдання № 2 засобами Doxygen&MarkDown (рис 6), опис функцій та детальний опис (рис 7,8), схема алгоритму дій (рис 9).



**Загальне завдання**

1. **Розробити** Розробити програму для заповнення масиву заданою кількістю простих чисел.

#Опис програми

## Функціональне призначення

**Функціональне призначення** Програма призначена для заповнення масиву заданою кількістю простих чисел.

## Опис логічної структури

**Опис логічної структури** Програма складається з функції заповнення масиву, у якій використано 3 вкладені цикл ( 1 – цикл відповідає за кількість чисел у масиві , другий для самого числа, третій для визначення просте число чи ні.) у основній функції отримуючи довжину(кількість чисел) масиву , програма звертається до функції ,використовуючи вхідні дані). Схема алгоритму дій подана на блок-схемі (рис. 1) Рисунок 1 - Схема алгоритму дій

## Реалізація програми на мові С

```
int func(const int len, int massiv[])
{ int last = 0;
int founder = 1; int temps = 0; while (temps < len)
{ for (int i = last + 1; i < 1000; i++) { founder = 1; for (int j = 2; j < i / 2; j++) { if (i % j == 0) { founder = 0; break; } } if (founder != 0)
{ massiv[temps] = i; last = i; break; } } temps++;
} return 0; }

int main() { const int length = 30; int mas[length]; mas[length] = func(length, mas); return 0; }
```

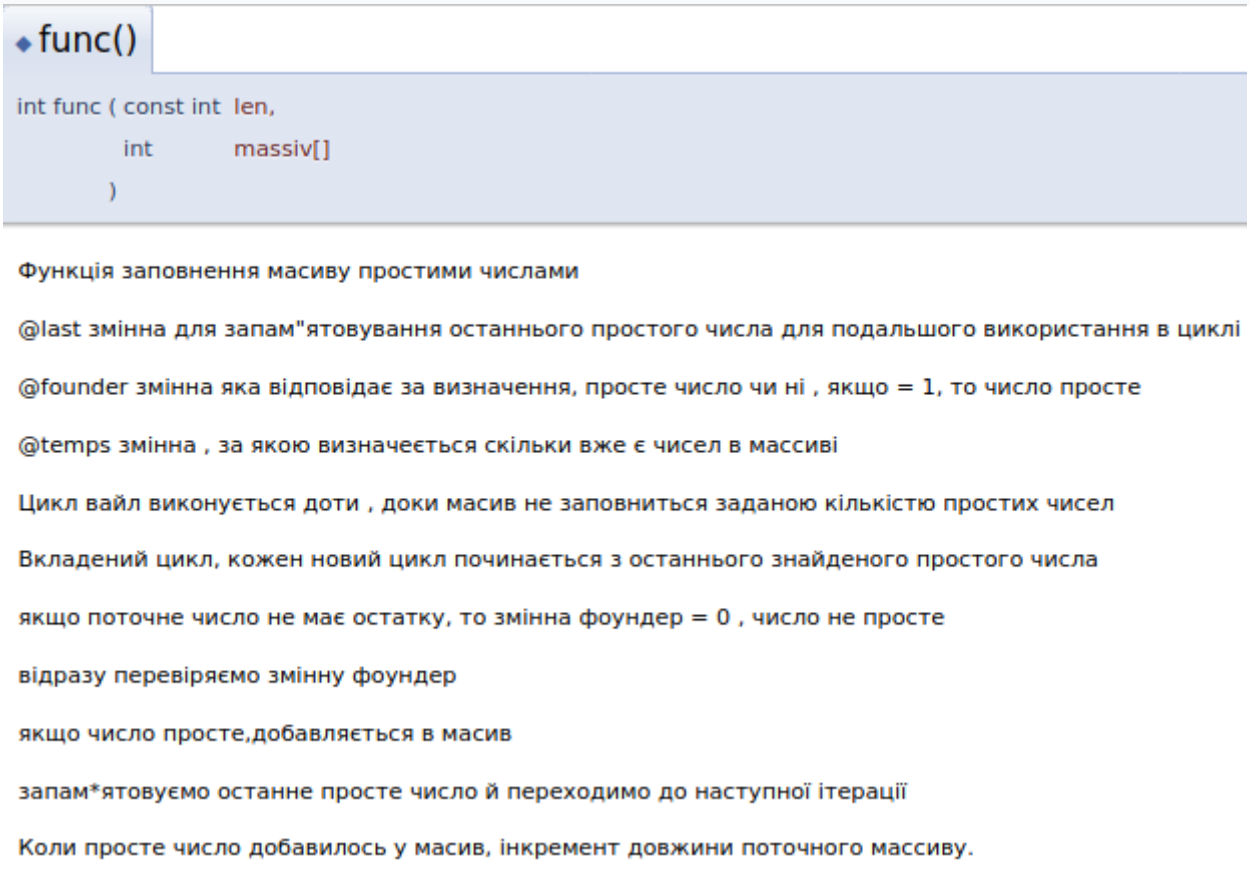
## Результат роботи програми

Результат роботи програми на (рис 2) Рисунок 2 - Демонстрація рооти програми

**Автор**
Kulish Pavlo.

**Дата**
01-dec-2020

Рисунок 6 - Опис завдання № 2 засобами Doxygen & MarkDown



### func()

```
int func ( const int  len,
           int        massiv[]
         )
```

Функція заповнення масиву простими числами

@last змінна для запам"ятовування останнього простого числа для подальшого використання в циклі

@founder змінна яка відповідає за визначення, просте число чи ні , якщо = 1, то число просте

@temps змінна , за якою визначеється скільки вже є чисел в массиві

Цикл вайл виконується дотии , доки масив не заповниться заданою кількістю простих чисел

Вкладений цикл, кожен новий цикл починається з останнього знайденого простого числа

якщо поточне число не має остатку, то змінна фоундер = 0 , число не просте

відразу перевіряємо змінну фоундер

якщо число просте,добавляється в масив

запам*ятовуємо останне просте число й переходимо до наступної ітерації

Коли просте число добавилось у масив, інкремент довжини поточного массиву.

Рисунок 7 – Опис функції func

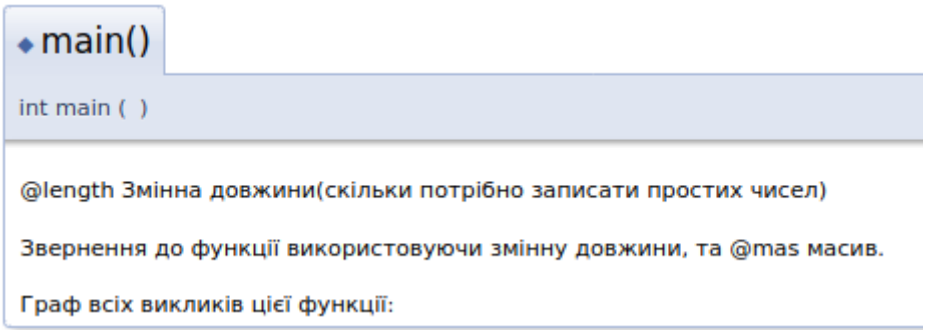



Рисунок 8 – Опис функції main

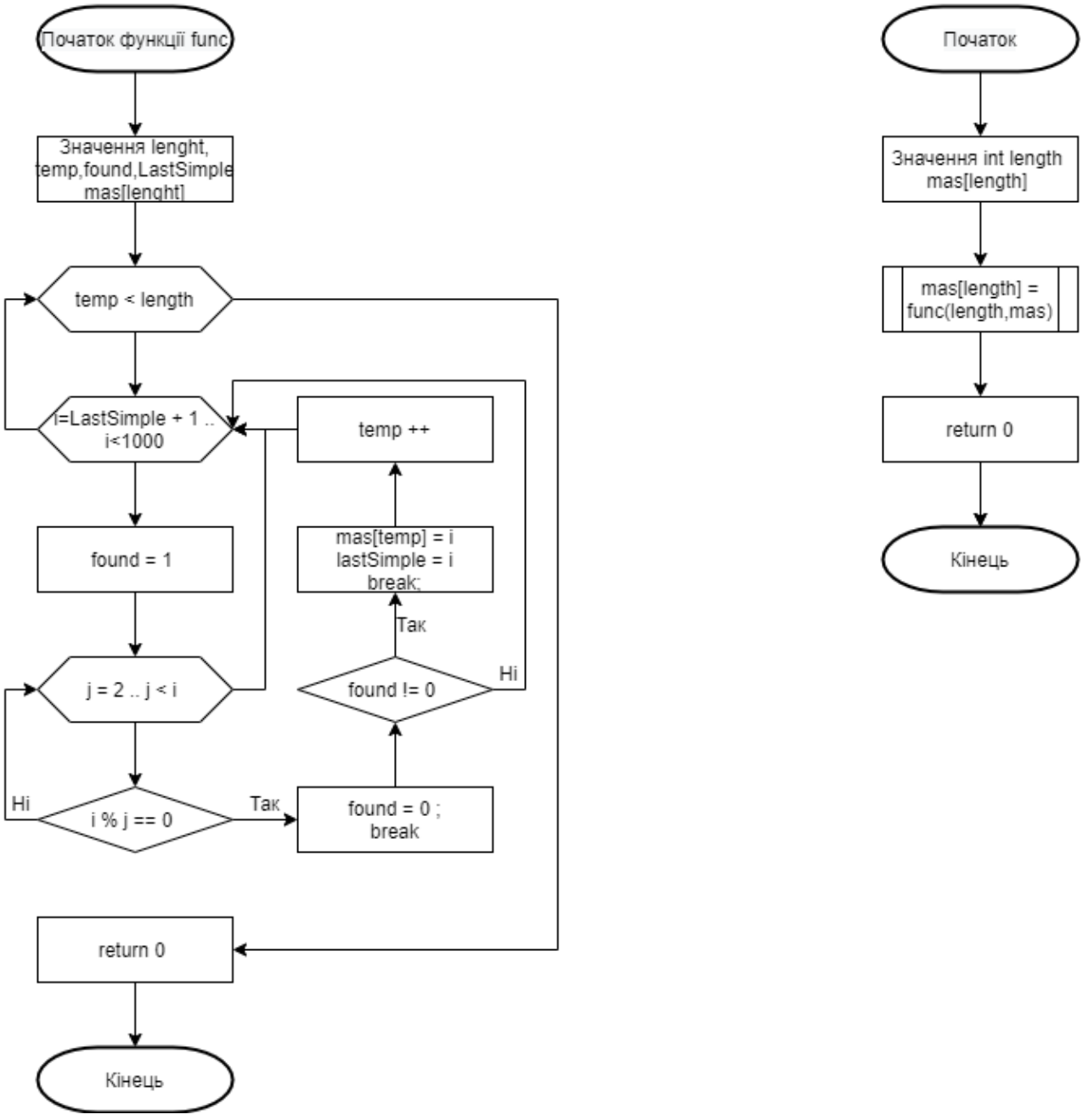



Рисунок 9 – Схема алгоритму дій

## 2.2.2 Реалізація програми на мові С (рис 10,11) та дебагер (рис 12)

```
int func(const int len, int massiv[])                // функція заповнення масиву заданою кількістю простих чисел
{
        int last = 0;                                // змінна для запам"ятовування останнього простого числа для подальшого використання в циклі
        int founder = 1;                             // змінна яка відповідає за визначення, просте число чи ні , якщо = 1, то число просте
        int temps = 0;                               // змінна , за якою визначеється скільки вже є чисел в массиві
        while (temps < len)                          // Цикл вайл виконується доти , доки масив не заповниться заданою кількістю простих чисел
        {
                for (int i = last + 1; i < 1000; i++) // Вкладений цикл, кожен новий цикл починається з останнього знайденого простого числа
                {
                        founder = 1;
                        for (int j = 2; j < i / 2; j++)
                        {
                                if (i % j == 0)      // якщо поточне число не має остатку, то змінна фоундер = 0 , число не просте
                                {
                                        founder = 0;
                                        break;
                                }
                        }
                        if (founder != 0)            // відразу перевіряємо змінну фоундер ,
                        {
                                massiv[temps] = i;   // якщо число просте,добавляється в масив,запам. останнє
                                                     // й переходимо до наступної ітерації
                                last = i;
                                break;
                        }
                }
                temps++;                             // Коли просте число добавилось у масив, інкремент довжини поточного массиву.
        }
        return 0;
}
```

Рисунок 10 – Функція , до якої йде звернення

```
int main()
{
        const int length = 30;
        int mas[length];

        mas[length] = func(length, mas);             // Звернення до функції використовуючи змінну довжини, та масив.
        return 0;
}
```

Рисунок 11 – Основна функція

| Переменная | Значение | Тип |
|---|---|---|
| ▼ Локальные переменные | | |
| length | 30 | const int |
| ▼ mas | [30] | int [30] |
| 0 | 1 | int |
| 1 | 2 | int |
| 2 | 3 | int |
| 3 | 4 | int |
| 4 | 5 | int |
| 5 | 7 | int |
| 6 | 11 | int |
| 7 | 13 | int |
| 8 | 17 | int |
| 9 | 19 | int |
| 10 | 23 | int |
| 11 | 29 | int |
| 12 | 31 | int |
| 13 | 37 | int |
| 14 | 41 | int |
| 15 | 43 | int |

Рисунок 12 – Демонстрація виконання через дебагер

### 2.3.1 Опис програми

Опис завдання № 3 засобами Doxygen&MarkDown (рис 13), опис функцій та детальний опис (рис 14,15), та схема алгоритму дій (рис 16).

**Загальне завдання**

1. **Розробити** Розробити програму для підраховування кількості слів у заданому масиві з будь-яким проміжком(кількістю пробілів)

#Опис програми

## Функціональне призначення

**Функціональне призначення** Програма призначена для підраховування кількості слів з будь-яким проміжком ( будь-якою кількістю пробілів між словами ) .

## Опис логічної структури

**Опис логічної структури** Є заданий масив зі словами , програма звертається до функції використовуючи цей масив . Функція складається з циклу вайл , у якому йде перебір кожного індексу , використовуючи умовні оператори ,програма додає до кількості слів 1 , поки йде слово, як тільки воно закінчилося , йде перевірка наступного індексу , й так, поки не перевіриться весь масив. Повертає результат кількості слів. Схема алгоритму дій подана на блок-схемі (рис. 1) Рисунок 1 - Схема алгоритму дій

## Реалізація програми на мові С

```
''' char func(const char massiv[])
{ int result = 0;
int length = 0;
int found = 0;
while(massiv[length] !=0) { if(massiv[length] !=' ') {
if (found == 0) { result ++; } found = 1; } else { found = 0; } length++; } return result; }

int main()
{ const char mas[] = " Hi gitler x x x "; int RES = func(mas);
}'''
```

## Результат роботи програми

Результат роботи програми на (рис 2) Рисунок 2 - Демонстрація роботи програми

**Автор**
Kulish Pavlo.

**Дата**
01-dec-2020



Рисунок 13 – Опис завдання № 3 засобами Doxygen&MarkDown

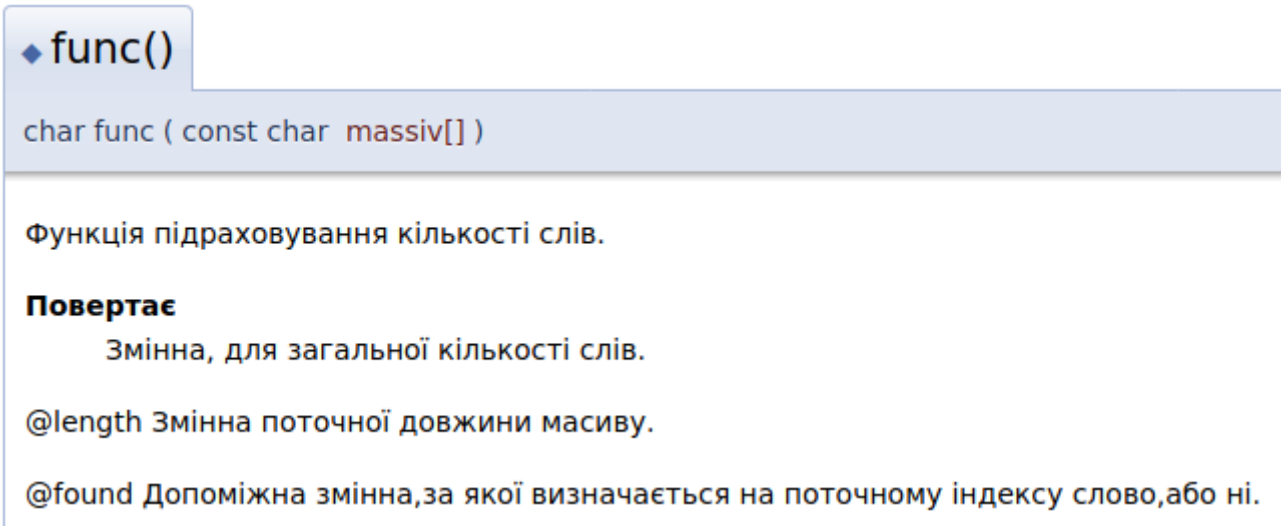



Рисунок 14 – Опис функції func

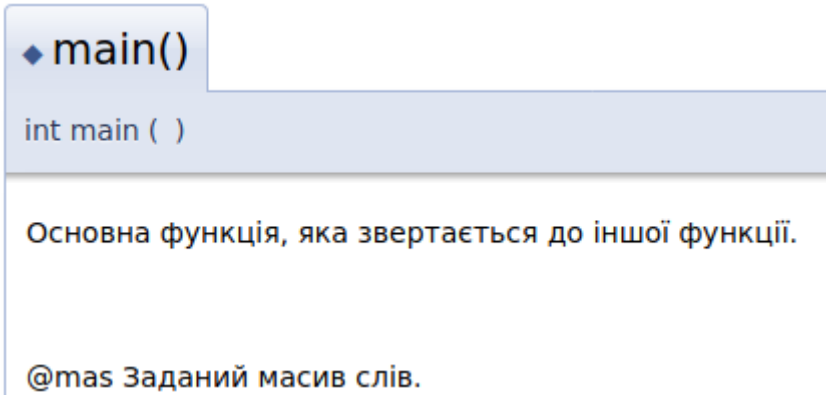
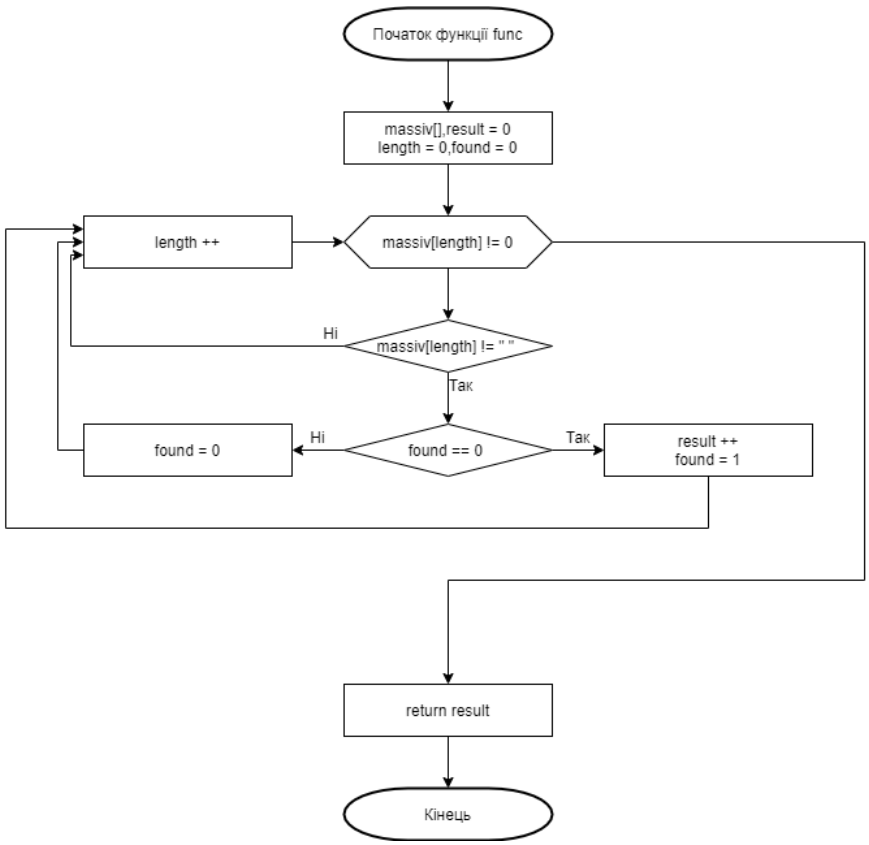



Рисунок 15 – Опис функції main



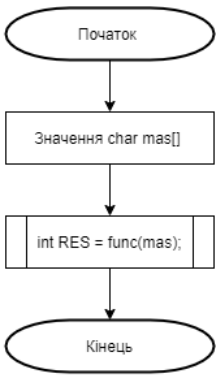




Рисунок 16 – Схема алгоритму дій

### 2.3.3 Реалізація програми на мові С (рис 17) та дебагер (рис 18)

```c
char func(const char massiv[])
{
        int result = 0;                                 // Змінна, для підрахування кількості слів
        int length = 0;                                 // З якого індексу починаємо
        int found = 0;                                  // Допоміжна змінна,для визначення слово,чи ні
        while(massiv[length] !=0)                       // Основний цикл , у якому через умовні оператори
        {
                if(massiv[length] !=' ')                // Йде перевірка кожного індексу масиву,поки не наткнемось
                {                                       // На проміжок,доти йде слово, як тільки проміжок почався,
                        if (found == 0)                 // інкрементуємо результат , присвоюємо змінній фоунд 1.
                        {
                                result ++;
                        }
                        found = 1;
                }
                else
                {
                        found = 0;
                }
                length++;
        }
        return result;                                  // Повернення кінцевого результату після виконання функції
}


int main()
{
        const char mas[] = " Hi gitler x x x ";
        int RES = func(mas);                            // Звернення до функції, використовуючи заданий масив
}                                                       // й отримання результату кількості слів.
```

Рисунок 17 – Реалізація програми на мові С

```
const char mas[] = " Hi gitler x x x ";
```

| Переменная | Значение | Тип |
|---|---|---|
| ▼ Локальные переменные | | |
| ▶ mas | [18] | const char [18] |
| RES | 5 | int |

Рисунок 18 – Результат виконання через дебагер

### 3. Структура проекту (рис 19)


Рисунок 19 – Структура проекту

### Висновки

Було набуто досвіду у роботі з Markdown документацією проекту, для завдань, які були зроблені у попередній роботі була розроблена MarkDown документація.