Міністерство освіти і науки України
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Обчислювальна техніка та програмування»

# ЗВІТ
Про ввиконання лабораторної роботи №13
«Строки»

Виконавець:
студент гр. КІТ-120В
Олексієнко Микита

Харків 2020

# Лабораторна робота №13. Строки

# 1.Вимоги

## 1.1 Розробник

Олексієнко Микита Віталійович

студент групи КІТ-120В

20.12.2020

### 1.2 Загальне завдання

Реалізувати програму відповідно до індивідуального завдання.

### 1.3 Індивідуальне завдання

Вирахувати для тексту частотну таблицю: для кожного символу визначити його частоту появи у тексті ( число таких символів у тексті ділене на загальне число символів у тексті ).

# 2. Виконання роботи

### 2.1 Функціональне призначення

Програма призначена для того щоб обчислити для тексту частотну таблицю

### 2.2 Створення репозиторію

Створивши репозиторій на github, клоную його та створюю всі файли за вимогами структури проекту.

### 2.3 Написання коду програми

```
#include <stdio.h>
#include <stdlib.h>
int result ( char *input, char **cx, float **fx)
{
    int n = 0;
    char *p = input;
    int k;
    int l = 0;
```

```
    char *c = (char *)calloc( 1, sizeof(*c));          // создание динамического массива неповторяющихся символов
    float *f = (float *)calloc( 1, sizeof(*f));        // создание динамического массива частот повторений
    while ( *p != '\0' )                               // цикл прохождения по входному массиву
    {
        char h = *p++;
        l++;
        if ( n == 0 )                                  // обработка первого символа
        {
            k = 0;
            c[k] = h;
            f[k] = 1;
            n++;
        }
        else
        {
            if ( h == c[k] )                           // если текущий символ входного массива равен текущему символу выходного массива
            {
                f[k]++;                                // увеличиваем счетчик кол-ва повторений этого символа
            }
            else
            {
                int i;
                for ( i = 0; i < n ; i++ )             // в противном случае, ищем символ в выходном массиве
                {
                    if ( c[i] == h ) break;
                }
```

```
                if ( i == n )                           // если символ не найден, то создаем в выходном массиве соответствующий элемент
                {
                        k = n++;
                        c = (char *)realloc( c, n * sizeof(*c));
                        f = (float *)realloc( f, n * sizeof(*f));
                        c[k] = h;
                        f[k] = 1;
                }
                else                                    // если символ найден, то увеличиваем счетчик повторений
                {
                        k = i;                          // данная позиция в выходном массиве становиться текущей
                        f[k]++;
                }
            }
        }
    }
    for ( int i = 0; i < n ; i++ )                      // преобразования кол-ва повторений символов в частоту
    {
        f[i] /= l;
    }
    *cx = c;
    *fx = f;
    return n;
}
int main ()
{
    char input[] = "hi, my name is Nikita";
    char *c;
```

```
    float *f;
    int n;
    n = result( input, &c, &f);
    printf("%d\n", n );
    for ( int i = 0; i < n; i++)
    {
        printf("%c - %f\n", c[i], f[i] );
    }

    return 0;
}
```

## 2.4. Зробимо опис функції:

### Опис функцій

#### ◆ main()

```
int main ( )
```

*Функція main*

Послідовність дій:

- оголошеня змінних

**Аргументи**

**input[]** - вхідний масив
**c** - символ, який півторюється
**f** - частота появи в тексті
**n** - кількість символів (повторення не вважаємо)

**Повертає**

успішний код повернення з програми (0)

Граф всіх викликів цієї функції:

#### ◆ result()

```
int result ( char *  input,
             char ** cx,
             float ** fx
           )
```

*Функція result*

Послідовність дій:

- оголошеня змінних

**Аргументи**

**n** - кількість символів (повторення не вважаємо)
**p** - показчик на поточний символ вхідного масиву
**k** - індекс поточного символу в вихідному масиві
**l** - загальна кількість символів у вхідному масиві
**c** - показчик на масив символів, що не повторюються
**f** - показчик на масив, що містить частоту повторення символів

- Для кожного символу з вхідного рядка створюється запис в вихідному масиві містить сам символ і кількість його повторень,
- а якщо такий запис існує, то кількість повторень збільшується на одиницю.
- Таблиця символів і кількість їх повторень зберігається в двох роздільних динамічних масивах.
- В кінці функції кількість повторень символів перетворюється в частоту.

**Повертає**

кількість елементів у вихідному масиві (n)

Рисунок 1 — Опис функціїї

## 2.5 Перевірка правильності роботи програми за допомогою Nemiver:

```
123  int main ()
124  {
125      char input[] = "hi, my name is Nikita";
126
127      char *c;
128      float *f;
129
130      int n;
131
132      n = result( input, &c, &f);
133
134      printf("%d\n", n );
135      for ( int i = 0; i < n; i++)
136      {
137          printf("%c - %f\n", c[i], f[i]  );
138      }
139
140      return 0;
141  }
142
143
144
145
146
```

| ID по | Переменная | Значение | Тип |
|---|---|---|---|
| | ▾ Локальные переменные | | |
| | ▸ input | [22] | char [22] |
| | ▾ c | 0x5555555592a0 "hi, mynaesNkt" | char * |
| | *c | 104 'h' | char |
| | ▾ f | 0x5555555592c0 | float * |
| | *f | 0.0476190485 | float |
| | n | 13 | int |
| | Параметры функции | | |

Рисунок 2 — Код у Nemiver.

## 2.6. Блок-схема  моєї програми виглядає наступним чином:

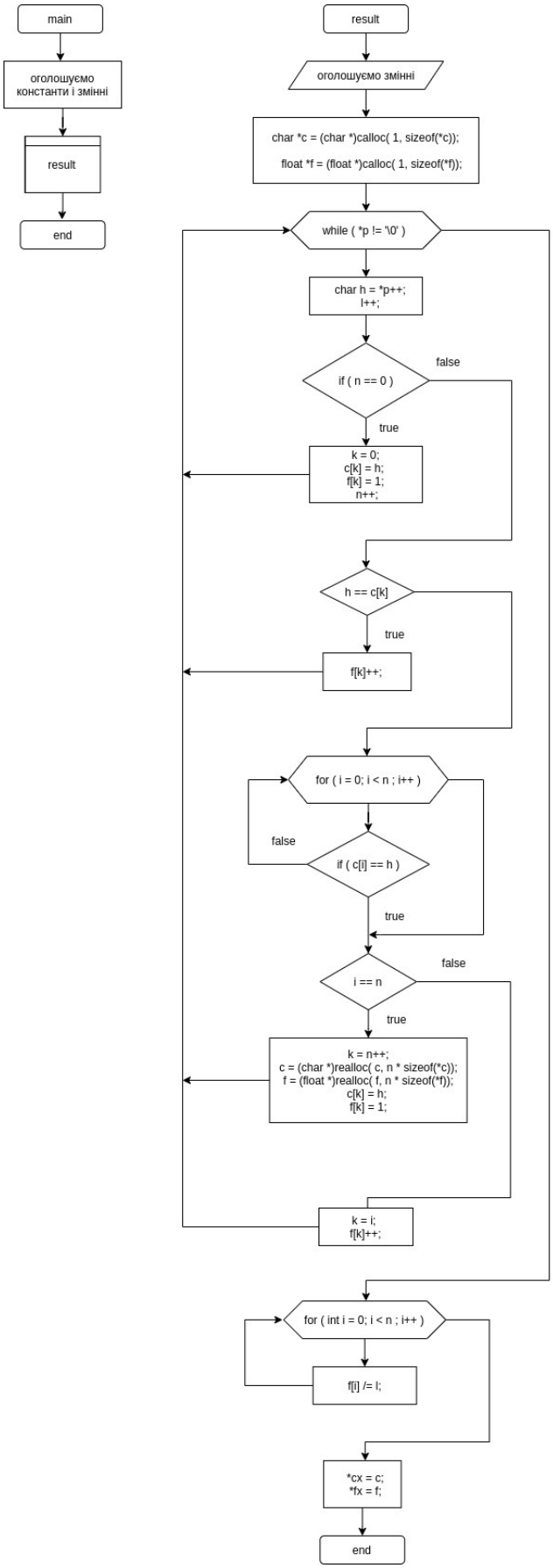


Рисунок 3 — Блок-схема

**Висновок:**

*Я створив програму, яка обчисляє для тексту частотну таблицю*

***Контрольні питання:***

*1. За допомогою функції strcat.*

*2. За допомогою функції strstr.*

*3. Масив символів. Кінцем строки вважається перший спеціальный нуль-термінатор.*

*4. За допомогою функції strcmp.*

*5. За допомогою функції strcpy.*