Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №4

«Работа со строками»

Студент: Гатченко А.С.

Группа: Б22-525

Преподаватель: Половнева Ю. А.

Москва – 2022

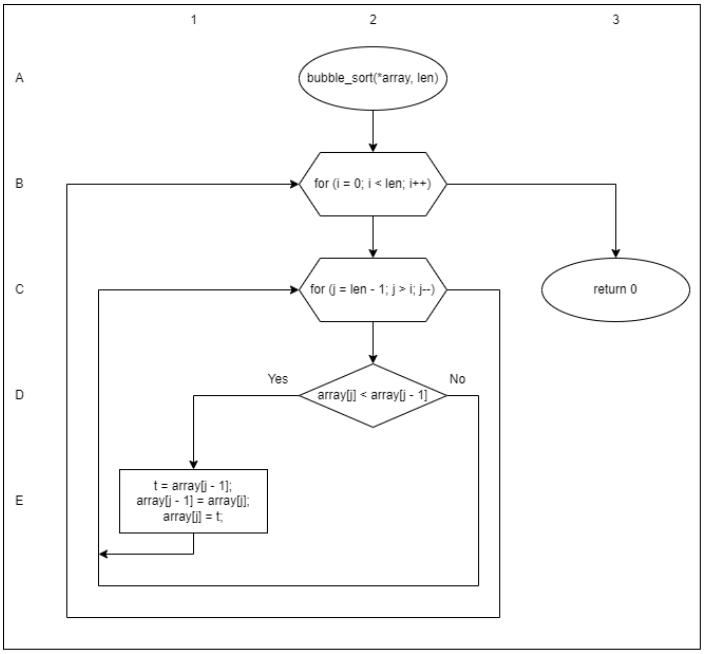
1. Формулировка индивидуального задания

Упорядочить слова в строке по возрастанию длины.

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовались встроенные типы данных int, предназначенные для работы с целыми числами, а также указатели на символы и на массивы символов.

3. Описание использованного алгоритма



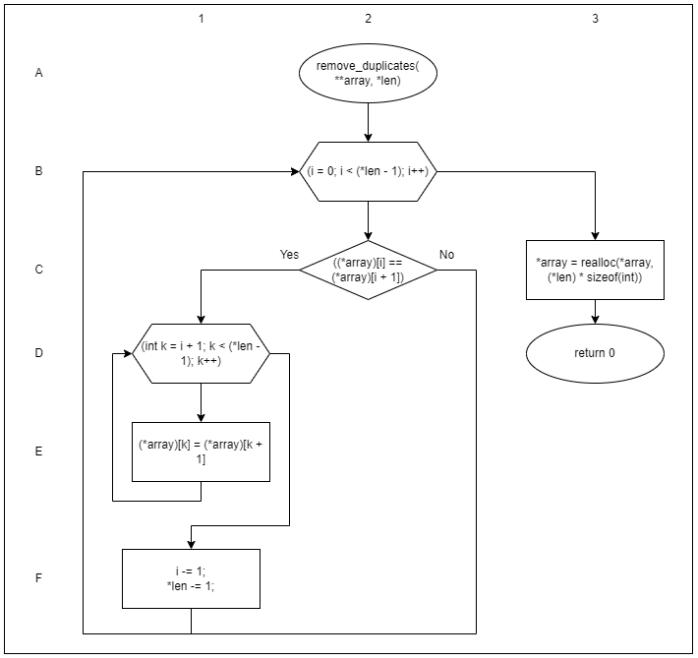
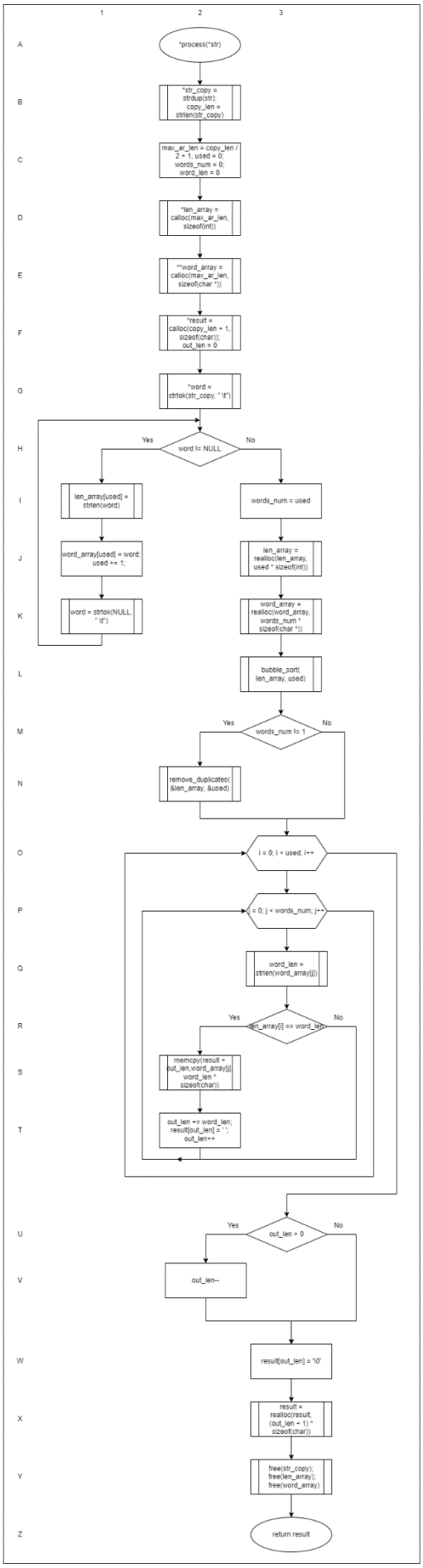


Рис. 1: Блок-схема алгоритма работы функции `bubble_sort()`

Рис. 2: Блок-схема алгоритма работы функции `remove_duplicates()`



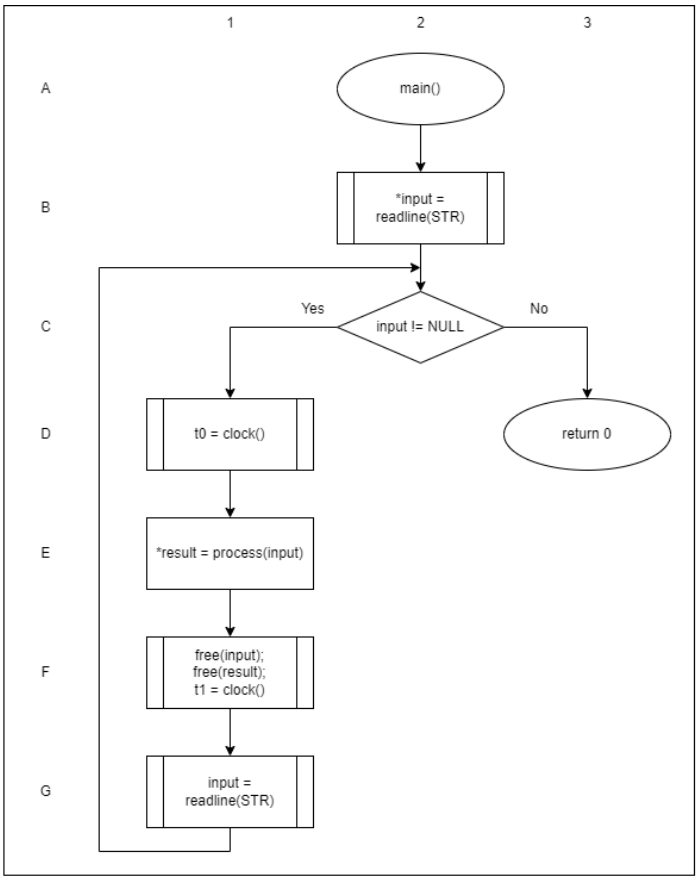
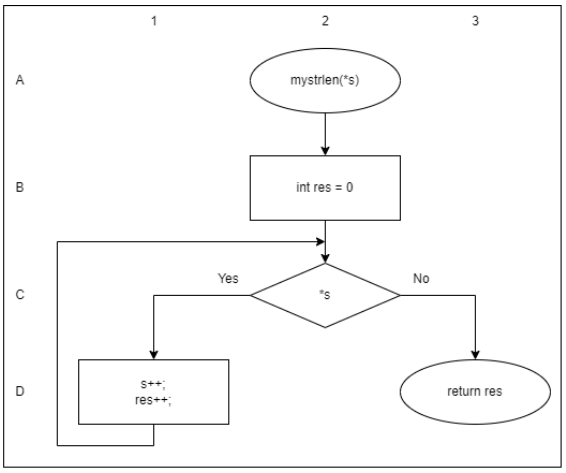
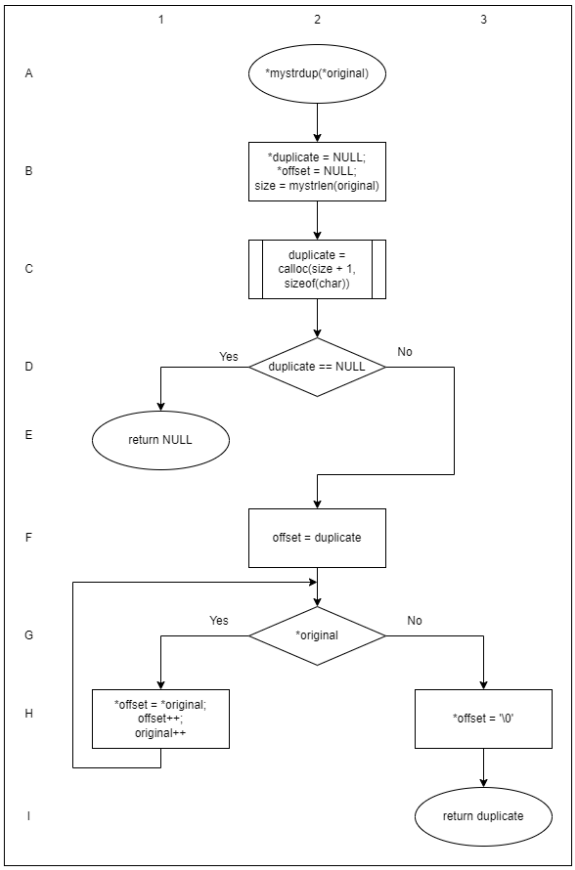


Рис. 3: Блок-схема алгоритма работы функции `process()`

Рис. 4: Блок-схема алгоритма работы функции `main()`





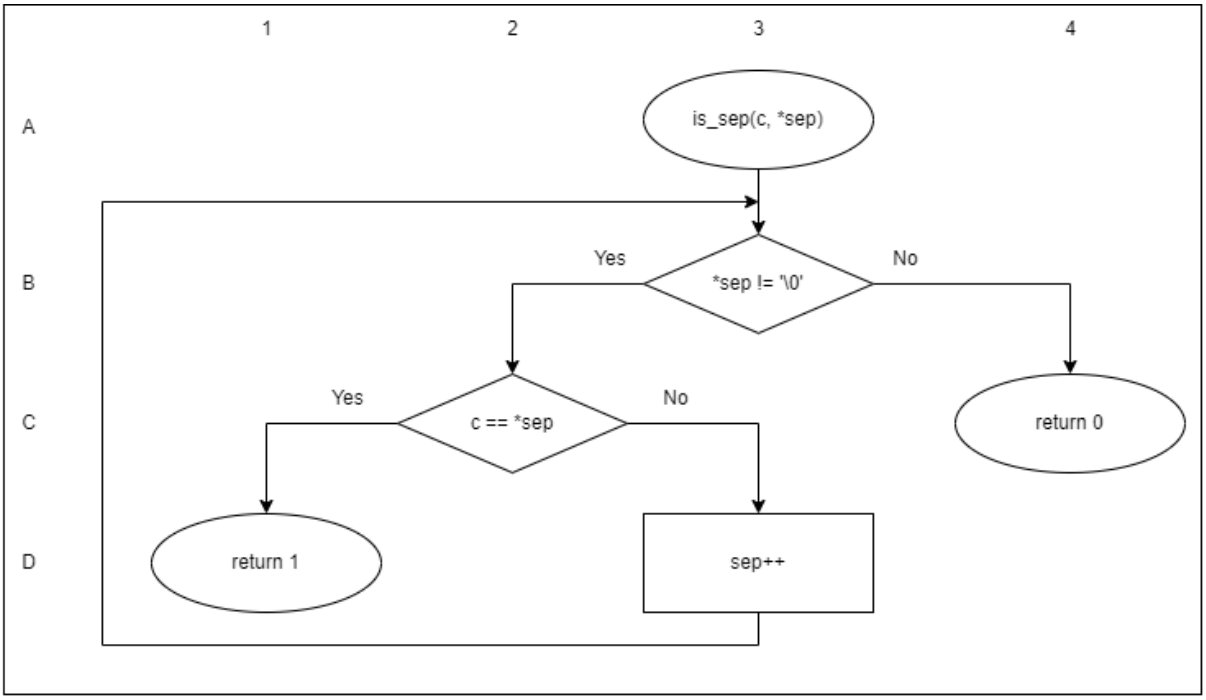
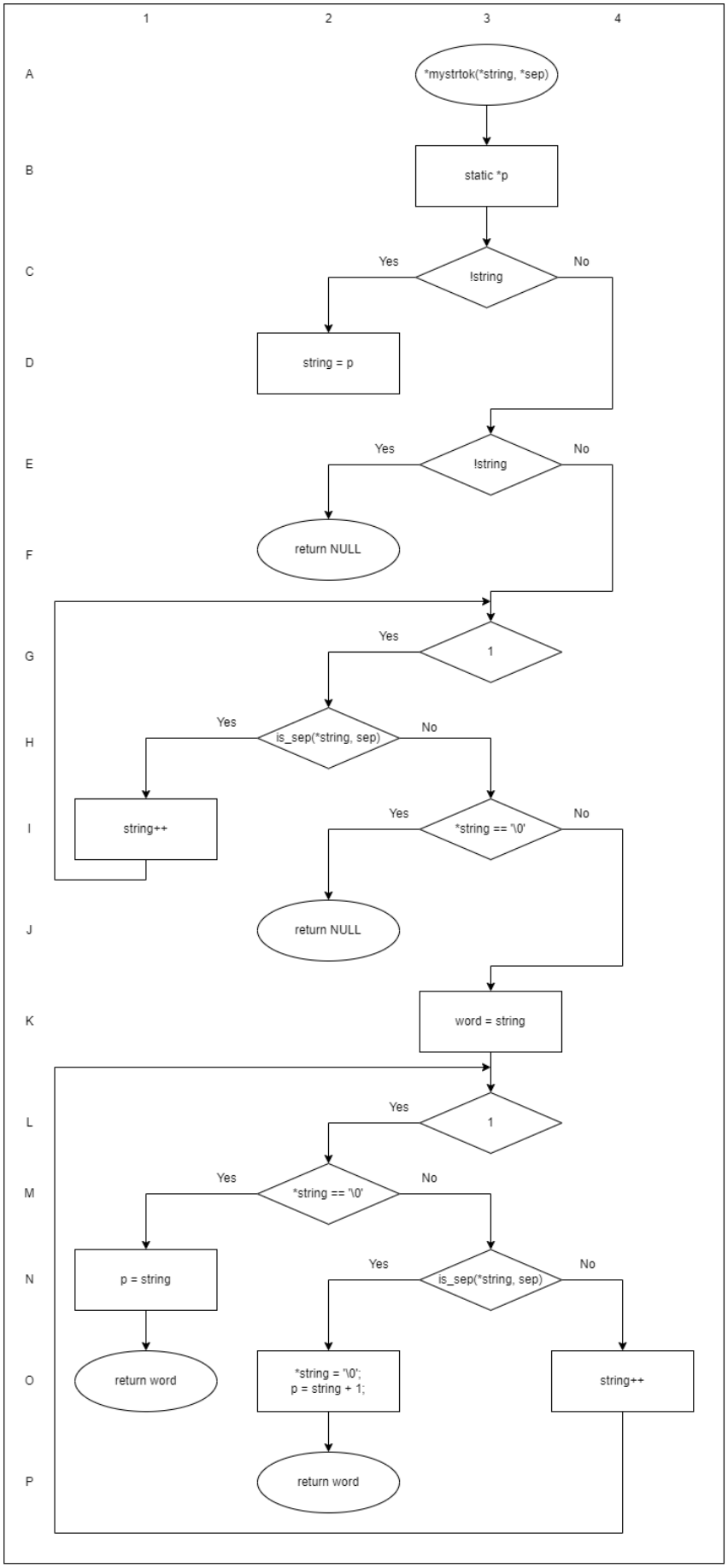


Рис. 5: Блок-схема алгоритма работы функции `mystrlen()`

Рис. 6: Блок-схема алгоритма работы функции `mystrdup()`

Рис. 7: Блок-схема алгоритма работы функции `is_sep()`



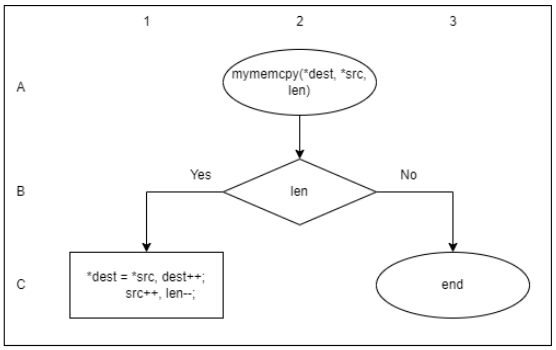


Рис. 8: Блок-схема алгоритма работы функции `mystrtok()`

Рис. 9: Блок-схема алгоритма работы функции `mymemcpy()`

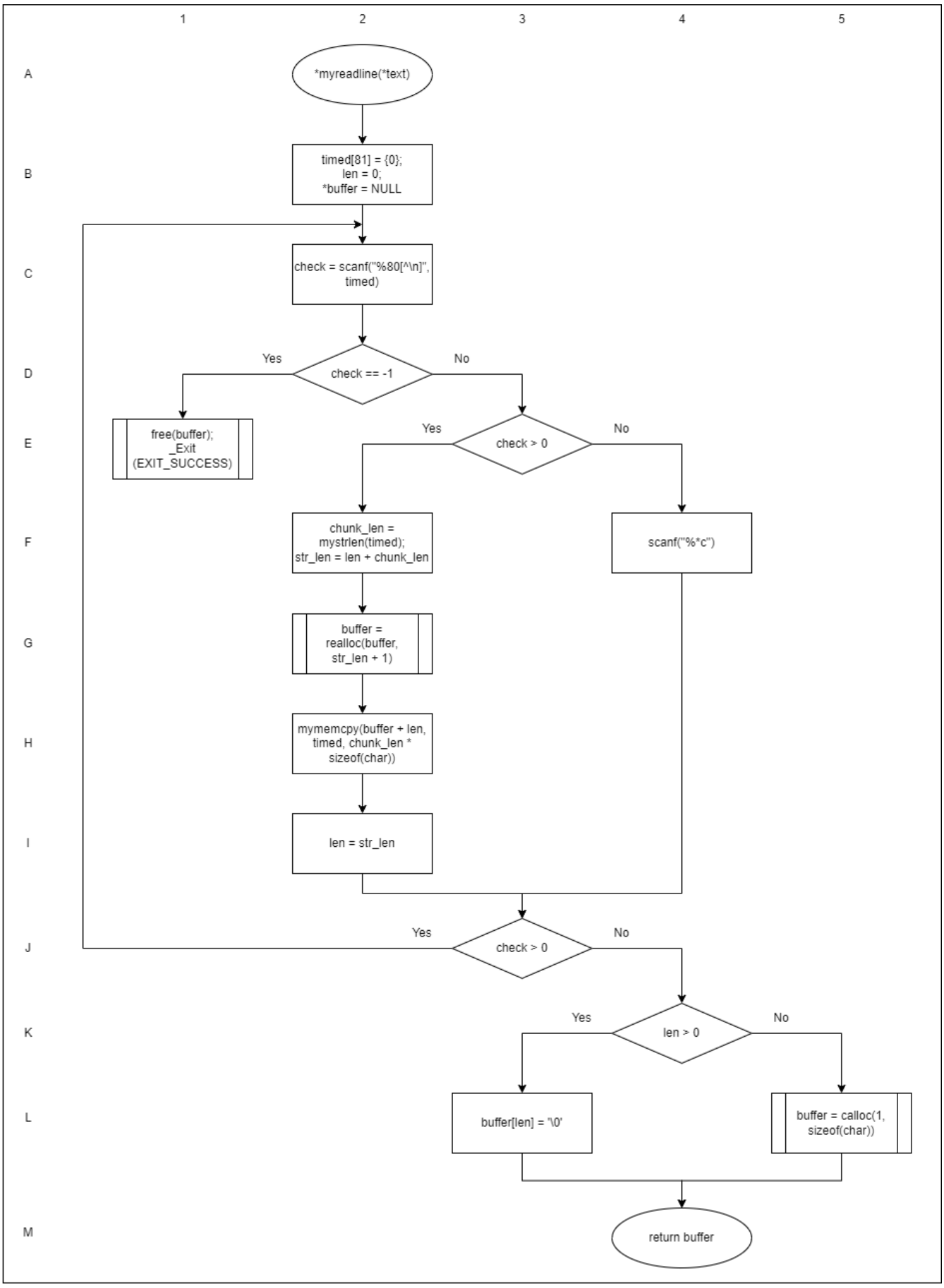


Рис. 10: Блок-схема алгоритма работы функции `myreadline()`

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы lab4 (1)

1) Файл: main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <readline/readline.h>
#include <string.h>
#include <time.h>

#define STR "Your input: "

int bubble_sort(int *array, int len) {
    for (int i = 0; i < len; i++) {
        for (int j = len - 1; j > i; j--) {
            if (array[j] < array[j - 1]) {
                int t = array[j - 1];
                array[j - 1] = array[j];
                array[j] = t;
            }
        }
    }
    return 0;
}

int remove_duplicates(int **array, int *len){
    for (int i = 0; i < (*len - 1); i++){
        if ((*array)[i] == (*array)[i + 1]){
            for (int k = i + 1; k < (*len - 1); k++){
                (*array)[k] = (*array)[k + 1];
            }
            i -= 1;
            *len -= 1;
        }
    }

    *array = realloc(*array, (*len) * sizeof(int));

    return 0;
}

char *process(const char *str) {
    char *str_copy = strdup(str);
    int copy_len = strlen(str_copy);
    int max_ar_len = copy_len / 2 + 1;
    int used = 0;
    int words_num = 0;
    int word_len = 0;
    int *len_array = calloc(max_ar_len, sizeof(int));
    char **word_array = calloc(max_ar_len, sizeof(char *));
    char *result = calloc(copy_len + 1, sizeof(char));
    int out_len = 0;
    char *word = strtok(str_copy, " \t");
    while (word != NULL) {
        len_array[used] = strlen(word);
        word_array[used] = word;
        used += 1;
        word = strtok(NULL, " \t");
    }
    words_num = used;
    len_array = realloc(len_array, used * sizeof(int));
    word_array = realloc(word_array, words_num * sizeof(char *));
```

```
    bubble_sort(len_array, used);
    if (words_num != 1){
        remove_duplicates(&len_array, &used);
    }

    for (int i = 0; i < used; i++){
        for (int j = 0; j < words_num; j++){
            word_len = strlen(word_array[j]);
            if (len_array[i] == word_len){
                memcpy(result + out_len, word_array[j], word_len *
sizeof(char));
                out_len += word_len;
                result[out_len] = ' ';
                out_len++;
            }
        }
    }

    if (out_len > 0) {
        out_len--;
    }

    result[out_len] = '\0';
    result = realloc(result, (out_len + 1) * sizeof(char));
    free(str_copy);
    free(len_array);
    free(word_array);

    return result;
}

int main() {
    char *input = readline(STR);
    while (input != NULL) {
       clock_t t0 = clock();
        printf("\"%s\"\n", input);
        char *result = process(input);
        printf("\"%s\"\n", result);
        free(input);
        free(result);
        clock_t t1 = clock();
        printf("%lf\n", (double)(t1 - t0)  / CLOCKS_PER_SEC);
        input = readline(STR);
    }
    return 0;
}
```

Листинг 2: Исходные коды программы lab4 (2)

1) Файл: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "read.h"

#define STR "Your input: "

int bubble_sort(int *array, int len){
    for (int i = 0; i < len; i++) {
        for (int j = len - 1; j > i; j--) {
            if (array[j] < array[j - 1]) {
                int t = array[j - 1];
```

```
                array[j - 1] = array[j];
                array[j] = t;
            }
        }
    }
    return 0;
}

int remove_duplicates(int **array, int *len){
    for (int i = 0; i < (*len - 1); i++){
        if ((*array)[i] == (*array)[i + 1]){
            for (int k = i + 1; k < (*len - 1); k++){
                (*array)[k] = (*array)[k + 1];
            }
            i -= 1;
            *len -= 1;
        }
    }

    *array = realloc(*array, (*len) * sizeof(int));

    return 0;
}

char *process(const char *str){
    char *str_copy = mystrdup(str);
    int copy_len = mystrlen(str_copy);
    int max_ar_len = copy_len / 2 + 1;
    int used = 0;
    int words_num = 0;
    int word_len = 0;
    int *len_array = calloc(max_ar_len, sizeof(int));
    char **word_array = calloc(max_ar_len, sizeof(char *));
    char *result = calloc(copy_len + 1, sizeof(char));
    int out_len = 0;
    char *word = mystrtok(str_copy, " \t");
    while (word != NULL){
        len_array[used] = mystrlen(word);
        word_array[used] = word;
        used += 1;
        word = mystrtok(NULL, " \t");
    }
    words_num = used;
    len_array = realloc(len_array, used * sizeof(int));
    word_array = realloc(word_array, words_num * sizeof(char *));
    bubble_sort(len_array, used);
    if (words_num != 1){
        remove_duplicates(&len_array, &used);
    }

    for (int i = 0; i < used; i++){
        for (int j = 0; j < words_num; j++){
            word_len = mystrlen(word_array[j]);
            if (len_array[i] == word_len){
                mymemcpy(result + out_len, word_array[j], word_len *
sizeof(char));
                out_len += word_len;
                result[out_len] = ' ';
                out_len++;
            }
        }
    }
```

```
    if (out_len > 0){
        out_len--;
    }

    result[out_len] = '\0';
    result = realloc(result, (out_len + 1) * sizeof(char));
    free(str_copy);
    free(len_array);
    free(word_array);

    return result;
}

int main(){
    char *input = myreadline(STR);
    while (input != NULL){
       clock_t t0 = clock();
        printf("\"%s\"\n", input);
        char *result = process(input);
        printf("\"%s\"\n", result);
        free(input);
        free(result);
        clock_t t1 = clock();
        printf("%lf\n", (double)(t1 - t0) / CLOCKS_PER_SEC);
        input = myreadline(STR);
       }
    return 0;
}
```

2) Файл: read.c

```
#include <stdio.h>
#include <stdlib.h>

size_t mystrlen(const char *s) {
    int res = 0;
    while (*s) {
        s++;
        res++;
    }
    return res;
}

char *mystrdup(const char *original){
   char *duplicate = NULL;
   char *offset = NULL;
   int size = mystrlen(original);

   duplicate = calloc(size + 1, sizeof(char));

   if (duplicate == NULL){
      return NULL;
   }

   offset = duplicate;    // чтоб сохранить исходный указатель в duplicate

   while (*original){
      *offset = *original;
      offset++;
      original++;
   }
   *offset = '\0';
```

```
    return duplicate;
}

int is_sep(char c, const char *sep){
    while(*sep != '\0'){
        if(c == *sep){
            return 1;
        }
            sep++;
    }
    return 0;
}

char *mystrtok(char *string, const char *sep){
    static char *p;
     char *word;

    if (!string){
         string = p;
    }

    if (!string){
        return NULL;
    }

    while (1){
        if (is_sep(*string, sep)){ // Если нет разделителей в начале, то break
            string++;
            continue;
        }
        if (*string == '\0'){
            return NULL;
        }
        break;
    }

    word = string;

    while (1){
        if (*string == '\0'){ // Конец строки
            p = string;
            return word;
        }
        if (is_sep(*string, sep)){
            *string = '\0';
            p = string + 1;
            return word;
        }
        string++;
    }
}

void mymemcpy (char *dest, const char *src, size_t len){
     while (len){
         *dest = *src;
         dest++;
         src++;
         len--;
     }
}

char *myreadline(const char *text){
    printf("%s", text);
```

```
   char timed[81] = {0};
   int len = 0;
   int check;
   char *buffer = NULL;

   do {
      check = scanf("%80[^\n]", timed);

      if (check == -1){
           free(buffer);
           printf("Program has been stopped\n");
           _Exit (EXIT_SUCCESS);
       }
      else if (check > 0){
         int chunk_len = mystrlen(timed);
         int str_len = len + chunk_len;
         buffer = realloc(buffer, str_len + 1);
         mymemcpy(buffer + len, timed, chunk_len * sizeof(char));
         len = str_len;
      }
       else {
            scanf("%*c");
        }
   }
   while (check > 0);
   if (len > 0){
      buffer[len] = '\0';
   }
   else {
        buffer = calloc(1, sizeof(char));
    }
    return buffer;
}
```

3) Файл: read.h

```
#ifndef READ
#define READ
#include <stddef.h>

size_t mystrlen(const char *s);
char *mystrdup(const char *original);
char *mystrtok(char *string, const char *sep);
char *mymemcpy (char *dest, const char *src, size_t len);
char *myreadline(const char *text);

#endif
```

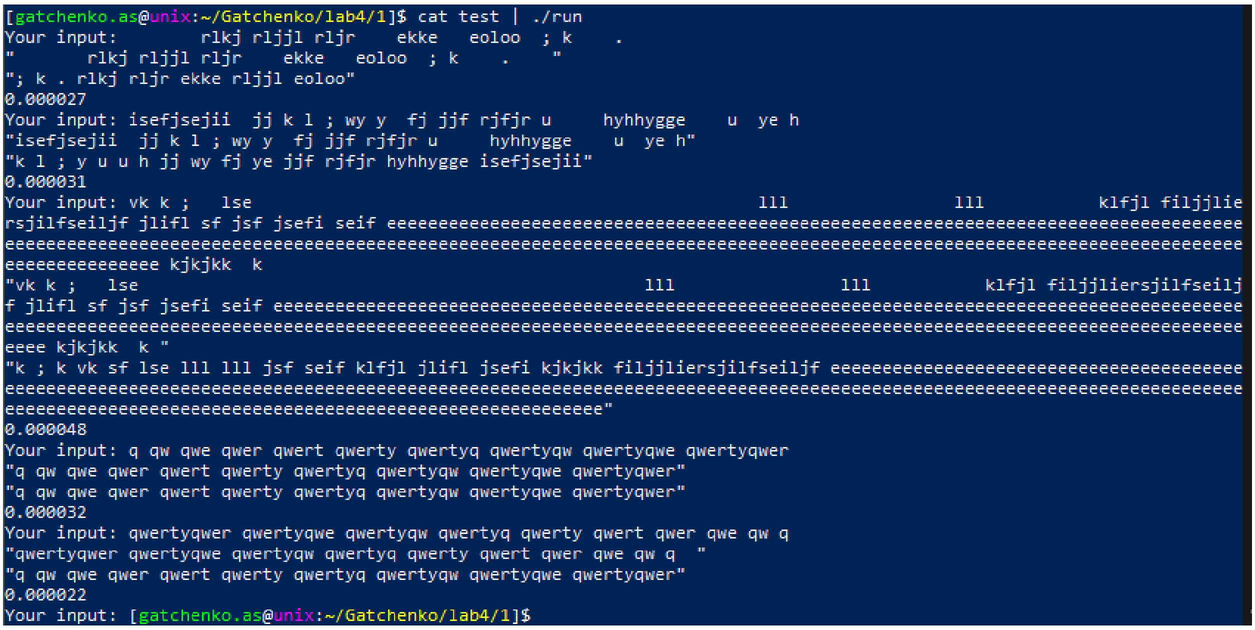
5. Описание тестовых примеров

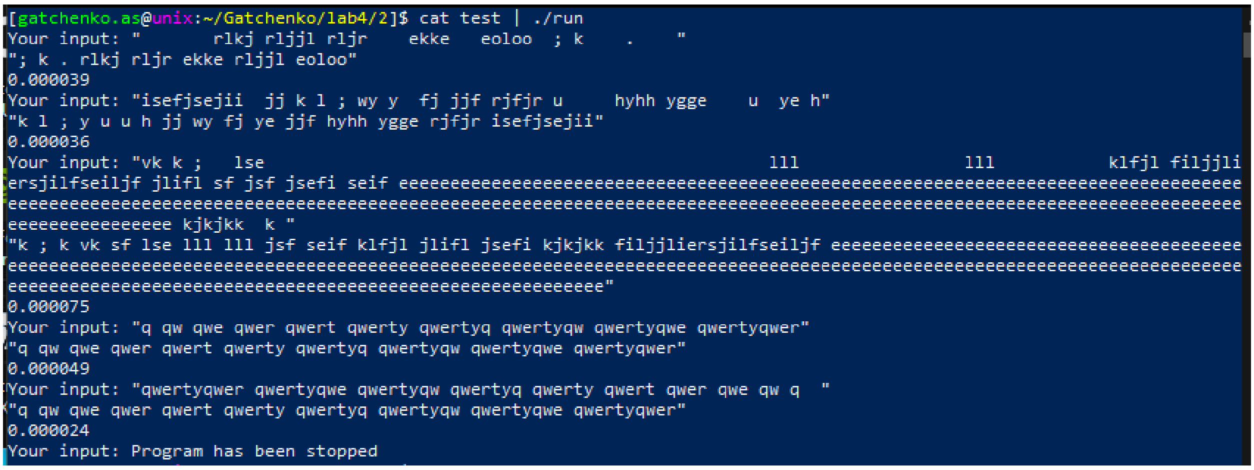
Таблица 1: Тестовые примеры lab4 (1) и lab4 (2)

Введенная строка	Строка после обработки	Время выполнения в lab4 (1) (сек)	Время выполнения в lab4 (2) (сек)
« rlkj rljjl rljr ekke eoloo ; k . »	«; k . rlkj rljr ekke rljjl eoloo»	0.000027	0.000039

«vk k ; lse lll lll klfjl filjjliersjilfseiljf jlifl sf jsf jsefi seif eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeeeeeee ee kjkjkk k»	«k ; k vk sf lse lll lll jsf seif klfjl jlifl jsefi kjkjkk filjjliersjilfseiljf eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeeeee eeeeeeeeeeeeeeeeee»	0.000048	0.000075
«q qw qwe qwer qwert qwerty qwertyq qwertyqw qwertyqwe qwertyqwer»	«q qw qwe qwer qwert qwerty qwertyq qwertyqw qwertyqwe qwertyqwer»	0.000032	0.000049
«qwertyqwer qwertyqwe qwertyqw qwertyq qwerty qwert qwer qwe qw q »	«q qw qwe qwer qwert qwerty qwertyq qwertyqw qwertyqwe qwertyqwer»	0.000022	0.000024

6. Скриншоты





```
[gatchenko.as@unix:~/Gatchenko/lab4/2]$ cat test | ./run
Your input: "       rlkj rljjl rljr    ekke   eoloo  ; k    .    "
"; k . rlkj rljr ekke rljjl eoloo"
0.000039
Your input: "isefjsejii  jj k l ; wy y  fj jjf rjfjr u     hyhh ygge    u  ye h"
"k l ; y u u h jj wy fj ye jjf hyhh ygge rjfjr isefjsejii"
0.000036
Your input: "vk k ;   lse                                              lll                lll           klfjl filjjli
ersjilfseiljf jlifl sf jsf jsefi seif eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeee kjkjkk  k "
"k ; k vk sf lse lll lll jsf seif klfjl jlifl jsefi kjkjkk filjjliersjilfseiljf eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee"
0.000075
Your input: "q qw qwe qwer qwert qwerty qwertyq qwertyqw qwertyqwe qwertyqwer"
"q qw qwe qwer qwert qwerty qwertyq qwertyqw qwertyqwe qwertyqwer"
0.000049
Your input: "qwertyqwer qwertyqwe qwertyqw qwertyq qwerty qwert qwer qwe qw q  "
"q qw qwe qwer qwert qwerty qwertyq qwertyqw qwertyqwe qwertyqwer"
0.000024
Your input: Program has been stopped
```

Рис. 11: Запуск программы lab4 (1)

Рис. 12: Запуск программы lab4 (2)

7. Выводы

В ходе выполнения данной работы на примере программы, выполняющей обработку строк, были рассмотрены базовые принципы работы построения программ на языке С и обработки строк:

1. Организация ввода/вывода, а также проверка корректности ввода.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными и дробными операндами.
5. Использование циклов и условий.
6. Использование указателей (параметров) на символы, а также на массивы символов.
7. Разбиение программы на несколько файлов.
8. Осуществление ввода с помощью сторонней библиотеки GNU readline.
9. Работа с памятью.