

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения ЭВМ

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Обработка строк на языке Си

Студент гр. 0382

Гудов Н.Р.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2020

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Гудов Н.Р.

Группа 0382

Тема работы: Обработка строк на языке Си

Исходные данные:

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой. Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр. Длина текста и каждого предложения заранее не известна. Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

Содержание пояснительной записки:

Перечисляются требуемые разделы пояснительной записки (обязательны разделы «Содержание», «Введение», «Заключение», «Список использованных источников»)

Предполагаемый объем пояснительной записки:

Не менее 00 страниц.

Дата выдачи задания: 24.11.2020

Дата сдачи реферата: 25.12.2020

Дата защиты реферата: 26.12.2020

Студент

Гудов Н.Р.

Преподаватель

Жангиров Т.Р.

АННОТАЦИЯ

Была реализована обработка текста заранее неизвестной длины. Для этого использовались динамические массивы и стандартные библиотеки языка с содержащимися в них функциями. Было реализовано меню, позволяющее пользователю выбирать некоторое действие по обработке текста. При желании выбрать функцию, не покрывающую возможности программы, пользователю напомнят о прочтении всплывающего перечня действий.

СОДЕРЖАНИЕ

Введение	4
1. Цель и задание	
1.1. Цель	6
1.2. Задание	
2. Реализация программы	
2.1. Решение первой подзадачи Ввод текста	7
2.2. Обработка текста	
2.3. Первая функция	8
2.4. Вторая функция	
2.5. Третья функция	
2.6. Четвертая функция	
 Заключение	 10
Список использованных источников	11
Приложение А. Работа программы	12

ВВЕДЕНИЕ

Необходимо создать программу, производящую обработку текста по выбранному пользователем сценарию. Реализация программы происходила с помощью использования динамической памяти и стандартных библиотек языка.

Создана программа, считывающая набор символов и работающая с ними как с текстом. Связь между программой и пользователем производится через консоль.

1. ЦЕЛЬ РАБОТЫ, ЗАДАНИЕ

1.1. Цель работы.

Цель работы: создать программу, производящую обработку данных, введенных пользователем.

Для этого требуется:

Реализовать хранение данных.

Реализовать обработку данных.

1.2. Задание курсовой работы

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой. Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра).

Далее, программа должна запрашивать у пользователя одно из следующих доступных действий (программа должна печатать для этого подсказку. Также следует предусмотреть возможность выхода из программы):

- 1) Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра). Например, строка “abcdba” должна принять вида “cd”.
- 2) Отсортировать все слова в предложении в лексикографическом порядке.
- 3) Удалить все предложения, в которых хотя бы одного слово является палиндромом.
- 4) Вывести все предложения в которых есть слово “HiddenAgent” и которое не является первым словом.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1. Решение первой подзадачи-Ввод текста

Пользователь вводит текст. Для этого имеется функция `get_text`, которая динамически выделяет память, после чего обращается к еще одной схожей функции `get_sentence`. Эта функция считывает значения, пока не встретится точка. Обе функции способны расширять объем выделенной памяти. Количество предложений записывается в переменную `size`. Запись прекращается после получения на вход символа `\n`.

После этого следует первичная обработка полученного текста, необходимо удалить дублирующие предложения. Функция `Remove` проверяет длины потенциально одинаковых предложений. Если проверка определяет одинаковые предложения, то переменные, такие как `trigger`, меняют свое значение, тем самым, не давая предложению добавиться в обновленный текст. Предложения сравниваются в нижнем регистре. Одновременно идет подсчет предложений, входящих в новый текст.

2.2. Обработка текста

После ввода, пользователю выводится текст без дублирующих предложений, после чего предлагается выбрать одно из доступных действий по его обработке. С помощью `scanf` и `switch` считываются, и выполняются инструкции. Вывод всех функция, кроме второй и последней реализован через отдельную функцию `menu`. Результат второй функции в силу моей некомпетентности выводится сразу. Последняя функция предлагает досрочно завершить программу и не нуждается в специальном выводе.

2.3. Первая функция

Функция FLalpha выполняет сравнение символов по краям предложений. До тех пор, пока символы с обоих концов предложения одинаковые на каждой итерации переменная с увеличивает значения на один. Таким образом считается количество позиций, на которые надо сместиться влево. Смещение организовано через цикл, количество итераций считается через доступные значения о смещении и длине предложения. Так как смещения происходило без символа точки и конца строки, они приписываются отдельно после цикла. Возвращает все тот же текст.

2.4. Вторая функция

Функция сортировки слов внутри предложений в лексикографическом порядке происходит следующим образом. С помощью strtok дробим предложение на слова. Слова записываем в новый массив строк. Затем с помощью qsort и специально созданной для этого небольшой функции sortstr размещаем слова внутри массива в нужном порядке, для сравнения используем strcmp. После выводим новое предложение прямо из этой функции. С помощью цикла выводятся и оставшиеся предложения, обработанные по этой же схеме.

2.5. Третья функция

Функция Palindrom реализована схоже с предыдущей. В ней тоже дробятся целые предложения. Каждое слово проверяется еще одной функцией isPalindrom, сравнивающей идущие друг на друга с концов слова символы. Предложения, прошедшие проверку, добавляются в еще одну динамически выделенную область памяти.

2.6. Четвертая функция

С помощью стандартной функции strstr получаем указатель на вхождение первого символа нужной подстроки, и если такой существует, сравниваем его с указателем на имеющуюся область памяти, тем самым проверяя, что

предложение не начинается с нужной подстроки. Прошедшие проверку предложения собираем вместе.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было создано консольное приложение для обработки текста, исходя из требований пользователя.

Программа может в автоматическом порядке удалять дублируемые предложения. Начинать предложения с разных букв. Сортировать предложения в лексикографическом порядке. Избавляться от предложений с палиндромами.

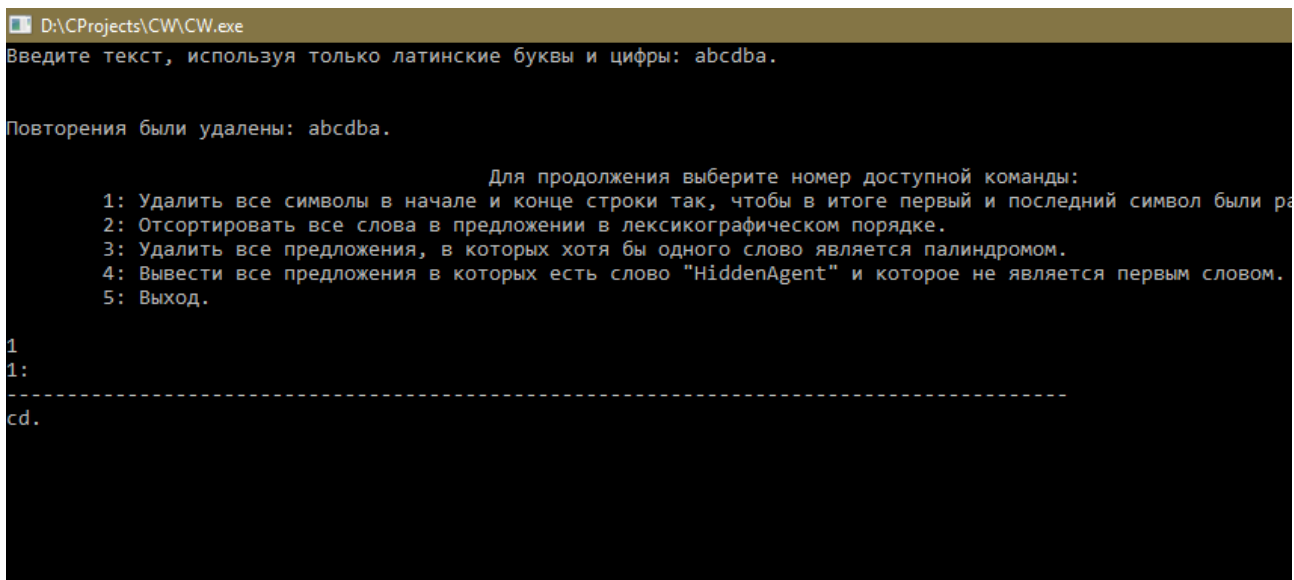
Результат работы соответствует поставленным требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Некоторые источники сети Интернет.

ПРИЛОЖЕНИЕ А

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ



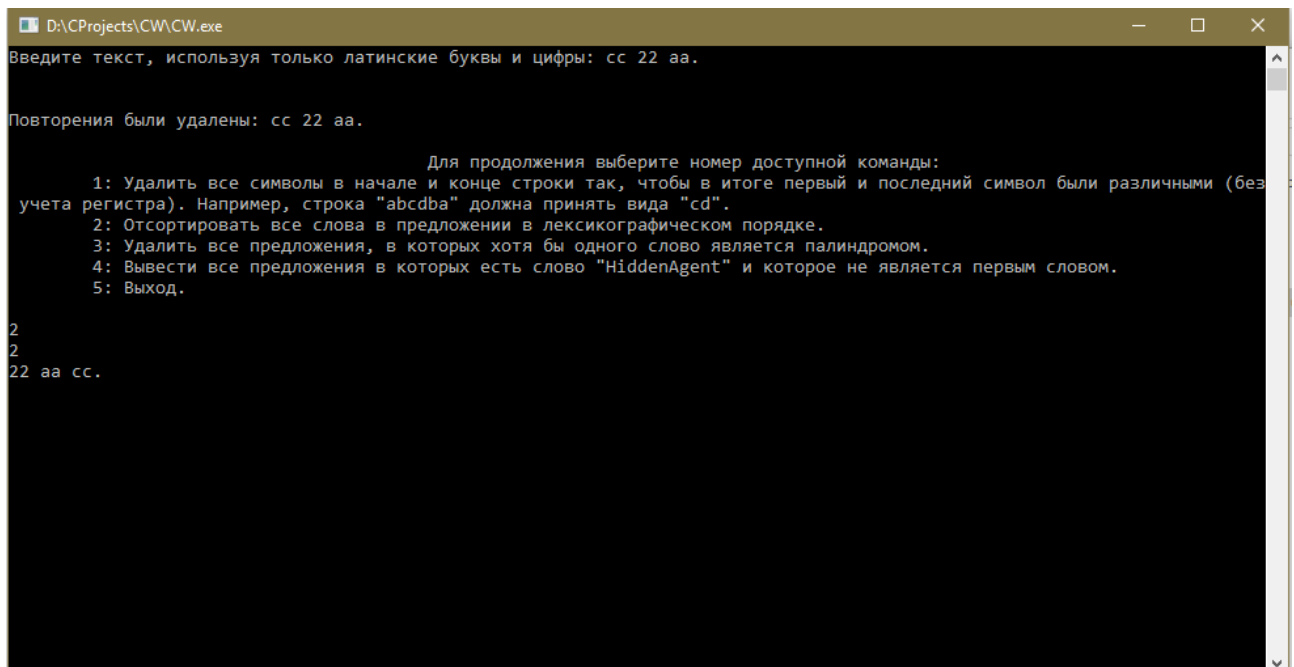
```
D:\CProjects\CW\CW.exe
Введите текст, используя только латинские буквы и цифры: abcdba.

Повторения были удалены: abcdba.

        Для продолжения выберите номер доступной команды:
1: Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были ра
2: Отсортировать все слова в предложении в лексикографическом порядке.
3: Удалить все предложения, в которых хотя бы одного слово является палиндромом.
4: Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом.
5: Выход.

1
1:
-----
cd.
```

Работа программы, при выборе в меню пункта 1



```
D:\CProjects\CW\CW.exe
Введите текст, используя только латинские буквы и цифры: сс 22 аа.

Повторения были удалены: сс 22 аа.

        Для продолжения выберите номер доступной команды:
1: Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без
учета регистра). Например, строка "abcdba" должна принять вида "cd".
2: Отсортировать все слова в предложении в лексикографическом порядке.
3: Удалить все предложения, в которых хотя бы одного слово является палиндромом.
4: Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом.
5: Выход.

2
2
22 аа сс.
```

Вторая операция-лексикографический порядок.

```
D:\CProjects\CW\CW.exe
Введите текст, используя только латинские буквы и цифры: ada fw. awdadfffaw.

Повторения были удалены: ada fw. awdadfffaw.

Для продолжения выберите номер доступной команды:
1: Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра). Например, строка "abcdba" должна принять вида "cd".
2: Отсортировать все слова в предложении в лексикографическом порядке.
3: Удалить все предложения, в которых хотя бы одного слово является палиндромом.
4: Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом.
5: Выход.

3
3:
-----
awdadfffaw
```

Функция 3

```
D:\CProjects\CW\CW.exe
Введите текст, используя только латинские буквы и цифры: HiddenAgent RFSS. FAWF HiddenAgent. Afff.

Повторения были удалены: HiddenAgent RFSS. FAWF HiddenAgent. Afff.

Для продолжения выберите номер доступной команды:
1: Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра). Например, строка "abcdba" должна принять вида "cd".
2: Отсортировать все слова в предложении в лексикографическом порядке.
3: Удалить все предложения, в которых хотя бы одного слово является палиндромом.
4: Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом.
5: Выход.

4
4:
-----
FAWF HiddenAgent.
```

Четвертый пункт меню

```
D:\CProjects\CW\CW.exe
Введите текст, используя только латинские буквы и цифры: dawd.

Повторения были удалены: dawd.

Для продолжения выберите номер доступной команды:
1: Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра). Например, строка "abcdba" должна принять вида "cd".
2: Отсортировать все слова в предложении в лексикографическом порядке.
3: Удалить все предложения, в которых хотя бы одного слово является палиндромом.
4: Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом.
5: Выход.

5
Exit
```

Выход

```
D:\CProjects\CW\CW.exe
Введите текст, используя только латинские буквы и цифры: adwa.

Повторения были удалены: adwa.

Для продолжения выберите номер доступной команды:
1: Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра). Например, строка "abcdba" должна принять вида "cd".
2: Отсортировать все слова в предложении в лексикографическом порядке.
3: Удалить все предложения, в которых хотя бы одного слово является палиндромом.
4: Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом.
5: Выход.

6
Выберите что-нибудь из предложенного списка!!
```

Неправильный ввод.

ПРИЛОЖЕНИЕ В

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

Файл main.c

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>
#define TEXTSIZE 10
#define SENTSIZE 50
#define TRUE 1
#define FALSE 0

char* get_sentence()
{
    int ssize=SENTSIZE;
    char* sentence=malloc(ssize*sizeof(char));
    char symbol;
    int i=0;
    while(TRUE)
    {
        symbol=getchar();
        sentence[i++]=symbol;
        if (i==ssize)
        {
            ssize+=20;
            sentence=(realloc(sentence, ssize*sizeof(char)));
        }
        if (symbol=='.') break;
    }
    sentence[i]='\0';
    return sentence;
}

char** get_text(int* size)
{
    int tsize=TEXTSIZE;
    char** text=malloc(TEXTSIZE*sizeof(char*));
    char* sentence;
    int i=0;
    while(TRUE)
    {
        sentence=get_sentence();
        text[i++]=sentence;
        if(getchar() == '\n') break;
        if(i==tsize)
        {
            tsize+=5;
            text=realloc(text, tsize*sizeof(char*));
        }
    }
    *size=i;
    return text;
}

char** HiddenAgent(char** text, int* size)
{
    int c=0;
```

```

char** text2=malloc(*size*sizeof(char*));
for(int i=0; i<*size; i++)
{
    char* p=strstr(text[i],"HiddenAgent");
    if ((p != &text[i][0]) && (p !=NULL))
    {
        text2[c++]=text[i];
    }
}
*size=c;
return text2;
}

char** Remover(char** text, int* size)
{
    int c=0;
    int trigger, d;
    char** text2=malloc(*size*sizeof(char*));
    for(int i=0; i< *size; i++)
    {
        d=0;
        for(int j=0; j<i; j++)
        {
            if(strlen(text[i])==strlen(text[j]))
            {
                trigger=1;
                for(int k=0; k<strlen(text[i]); k++)
                {
                    if(tolower(text[j][k])!=tolower(text[i][k]))
                    {
                        trigger=0;
                        break;
                    }
                }
                d+=trigger;
            }
        }
        if(d==0)
        {
            text2[c++]=text[i];
        }
    }
    *size=c;
    return text2;
}

char** FLalpha(char** text, int* size)
{
    for (int i =0; i< *size; i++)
    {
        int c=0;
        int l=strlen(text[i])-1;
        for (int j=0; j<(l/2); j++)
        {
            if(tolower(text[i][j])==tolower(text[i][l-j-1]))
            {
                c+=1;
            }
            else break;
        }
        int k;
    }
}

```



```

        for(k=0; k<1-2*c; k++)
        {
            text[i][k]=text[i][k+c];
        }
        text[i][k]='.';
        text[i][k+1]='\0';
    }
    return text;
}

int isPalindrom(char* str)
{
    int i=0, j=strlen(str)-1;
    while (i<j)
    {
        if(str[i++]!=str[j--])
        {
            return FALSE;
        }
    }
    return TRUE;
}

char** Palindrom(char** text, int* size)
{
    int c=0;
    char** text2=malloc(*size*sizeof(char*));
    char* sep=" .,";
    for (int i =0; i< *size; i++)
    {
        int trigger=1;
        char *word = strtok(text[i], sep);
        while(word != NULL)
        {
            char* ptext=text[i];
            if (isPalindrom(ptext))
            {
                trigger=0;
                break;
            }
            word=strtok(NULL, sep);
        }
        if (trigger==1)
        {
            text2[c++]=text[i];
        }
    }
    *size=c;
    return text2;
}

int sortstr(const void *a, const void *b)
{
    char** x=(char**)a;
    char** y=(char**)b;
    return strcmp(*x, *y);
}

char** Sort(char** text, int* size)
{
    char** text2=malloc(*size*sizeof(char*));
    char *arr[*size];
    int i;

```

```

    for (i = 0; i < *size; i++)
    {
        arr[i] = (char*) malloc(25);
    }
    char* sep=" .,";
    for (int i =0; i< *size; i++)
    {
        int j=0;
        char *word = strtok(text[i], sep);
        while (word != NULL)
        {
            arr[j++]=word;
            word = strtok(NULL, sep);
        }
        qsort(arr, j, sizeof(char*), sortstr);
        for (int k=0; k<j; k++)
        {
            printf("%s",arr[k]);
            if (k==j-1){printf(". ");}
            else{printf(" ");}
        }
    }
    return text2;
}

void menu(int sw, char*** text, int size, char** (*f)(char**, int*))
{
    wprintf(L"%d:\n",sw);
    wprintf(L"-----\n");
    -----\n");
    char** text2 = f(*text, &size);
    for(int i = 0; i < size; i++)
    {
        printf("%s ", text2[i]);
        free(text2[i]);
    }
    for(int i = 0; i < size; i++) free((*text)[i]);
    free(text2); free(*text);
}

int main() {
    setlocale(LC_ALL, "");
    wprintf(L"Введите текст, используя только латинские буквы и цифры: ");

    int size;
    char **text;

    text = get_text(&size);
    char **tmp = Remover(text, &size);
    free(text);
    text = tmp;
    wprintf(L"\n\nПовторения были удалены: ");
    for (int i = 0; i < size; i++) printf("%s ", text[i]);
    printf("\n");
    char **temp = text;

    wprintf(L"\n\t\t\t\t\tДля продолжения выберите номер доступной команды:\n");
    wprintf(L"\t1: Удалить все символы в начале и конце строки так, чтобы в
итоге первый и последний символ были различными (без учета регистра). Например,
строка "abcdba" должна принять вида "cd".\n");
    wprintf(L"\t2: Отсортировать все слова в предложении в лексикографическом
порядке.\n");

```

```

    wprintf(L"\t3: Удалить все предложения, в которых хотя бы одного слово
является палиндромом.\n");
    wprintf(L"\t4: Вывести все предложения в которых есть слово "HiddenAgent" и
которое не является первым словом.\n");
    wprintf(L"\t5: Выход.\n\n");

    int sw;
    scanf("%d", &sw);
    switch (sw)
    {

        case 1: menu(1, &temp, size, FLalpha); getch(); break;
        case 2: wprintf(L"%d\n",sw); Sort(temp, &size); getch(); break;
        case 3: menu(3, &temp, size, Palindrom); getch(); break;
        case 4: menu(4, &temp, size, HiddenAgent); getch(); break;
        case 5: wprintf(L"Exit\n"); getch(); break;
        default: wprintf(L"Выберите что-нибудь из предложенного
списка!!\n"); getch(); break;

    }
    return 0;
}

```