МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа со строками в языке Си

Студент гр. 1304 Новицкий М.Д.

Преподаватель Чайка К.В.

Санкт-Петербург

2021

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Исходные данные:

Текст, состоящая из дат. Даты формата DD/MM/YYYY, разделённые пробелом.

Ввод строки заканчивается символом переноса строки.

Содержание пояснительной записки:

Введение.

Основные теоретические положения.

Описание кода программы.

Заключение.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата получения задания: 15.07.2021

Дата сдачи реферата: 18.12.2021

Дата защиты реферата: 20.12.2021

АННОТАЦИЯ

Курсовая работа представляет из себя программу, предназначенную для работы с текстом (текст представляет собой предложения, разделенные точкой. Предложения - набор дат, разделенные пробелом, дата - подстрока вида “DD/MM/YYYY”. Длина текста и каждого предложения заранее не известна. Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним. Программа удаляет повторяющиеся

предложения и запрашивает у пользователя одну из 5 команд, 1,2,3,4 – для обработки текста и 5 – для выхода из программы. Код программы написан на языке программирования Си, запуск программы выполняется на операционных системах семейства Linux. При написании программы использовались управляющие конструкции и стандартные библиотеки языка программирования Си. Для ввода строки использовалась отдельная функции. Исходный код, показывающие корректную работу программы, и результаты тестирования представлены в приложениях.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – написание программы для считывания и редактирования строк.

Требуется самостоятельно (без использования стандартных и сторонних библиотек) реализовать следующие функции:

1. Функция принимающая текст и выводящая те предложения, в которых есть дата с нынешним годом и месяцем.
2. Функция принимающая текст и сортирующая его предложения по увеличению ранней даты в них.
3. Функция удаляющая предложения, в которых все даты относятся к 19 веку.
4. Функция выводящая для каждого предложения самую раннюю и самую позднюю дату.

Функции должны поддерживать защиту от выхода за границу буфера.

Для демонстрации работы функций, требуется написать диалоговую программу, которая должна предлагать пользователю выбрать тестируемую функцию, а после - запрашивать у пользователя текст, который надо передать в качестве аргумента. Одним из вариантов выбора следует предусмотреть завершение программы.

1. ОПИСАНИЕ КОДА ПРОГРАММЫ

1.1. Функция считывания строки

```
int read_arr(char* sen_sep, char*** txt){
    int a = 0;
    char str;
    int flag_sent = 1;
    int flag_txt = 1;
    int sent_count = 0;
    int char_count = 0;
    int string_length = 1000;
    while (1)
    {
        char_count = 0;
        (*txt) = realloc((*txt), sizeof(char*) *
(sent_count+1));
        if ((*txt) == NULL){
                        puts("Недостаточно памяти!");
                        exit(0);
                }
        (*txt)[sent_count] = malloc(sizeof(char) *
string_length);
        str = getchar();
        if (str == '\n')
        {
            a++;
            if (a==2)
                break;
        }
        else{
                (*txt)[sent_count][char_count] = str;
                a=0;
        }
```

```
        do
        {
            char_count++;
            str = getchar();
            if ((str == '\n')){
                a++;
                break;
            }
            else{
                a = 0;
            }
            (*txt)[sent_count][char_count] = str;
            flag_sent = (strchr(sen_sep,str) == NULL);
            if (char_count>=string_length - 2)
            {
                string_length = string_length + 500;
                (*txt)[sent_count] = (char*)
realloc((*txt)[sent_count], string_length);
                if ((*txt)[sent_count] == NULL){
                        puts("Недостаточно памяти!");
                        exit(0);
                }
            }
        }
        while(flag_sent == 1);
        if (a == 2)
            break;
        str=getchar();
        if (str == '\n')
                a++;
        else
                a = 0;
        (*txt)[sent_count][char_count+1] = ' ';
        (*txt)[sent_count][char_count+2] = '\0';
        sent_count++;
    }
    return sent_count;
}
```

Функция считывает строку посимвольно, находит количество предложений в тексте, а также построчно выделяет память. Если пользователь прервёт ввод, перейдя на новую строку два раза, то он прервёт ввод.

1.2. Функция удаления повторяющихся предложений.

```
int delete_sent(int sent_count, char*** txt){
    for(int i = 0; i<sent_count; i++){

            int j = i+1;

            while (j<sent_count){
                if (strcasecmp((*txt)[i],(*txt)[j])==0){

                    free((*txt)[j]);
                    sent_count--;
                    for (int k = j; k<=sent_count;k++){
                        (*txt)[k] = (*txt)[k+1];
                    }
                }
                else{
                    j++;
                }
            }
    }
    return sent_count;
}
```

Функция принимает на вход текст и кол-во предложений. Проверяет на наличие одинаковых предложений, и если такие есть, удаляет одно из них. Функция возвращает новое кол-во предложений

1.3. Функция выводящая предложения в которых есть дата с нынешним годом и месяцем.

```
 void print_sentenses_with_curent_date(int new_sent_count, char**
txt){
        time_t t = time(NULL);
        struct tm* timeinfo = localtime(&t);
        int d,m,y,f,i;
        for(i = 0 ; i<new_sent_count ; i++){
                f = 0;
for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m, &y) > 0 ; p +=
11){
    if(y == timeinfo->tm_year+1900  &&  m == timeinfo->tm_mon+1){
                f = 1;
                break;
                     }
                }
    if(f) printf("%s\n",txt[i]);
        }
}
```

Функция использует структуру timeinfo и функцию localtime библиотеки time.h, чтобы записать в неё нынешний год и месяц. После чего считывает все предложения текста посимвольно с шагом 11, так как дата формата DD/MM/YYYY, то на неё потребуется 10 символов плюс пробел, итого 11 символов. До считывания, инициализируется переменная f, для того, чтобы понять если в предложении дата с нынешним месяцем и годом. После считывания первое значение заносится в переменную дня, второе в переменную месяца, третье в переменную года. После чего идёт сравнение со значениями структуры timeinfo. Если значения равны, то f присваивается значение 1 после чего цикл завершается, далее идёт условие, if (f), то функция печатает это предложение и переходит к следующему.

1.4. Функция сортирующая предложения в тексте по возрастанию самой ранней даты в них.

```
void sort_sentenses(int new_sent_count,char*** txt){
        qsort(*txt,new_sent_count,sizeof(char*),cmp);
}
```

Функция принимает на вход текст и количество предложений. Далее сортировка происходит с помощью функции qsort библиотеки stdlib.h.

1.5. Компоратор для сортировки предложений.

Функция написана для сравнивания самой ранней даты в предложении. Так же как и в функции 1.3, мы пробегаемся по предложениям с шагом 11, и записываем соответствующие значения в переменный d,m,y. До считывания предложения заводится две переменные min1 и min2(которые равны 1000000000. Заводим мы такое значение, чтобы не выходить за пределы int) – значение самых ранних дат для двух предложений. После чего заводим переменную value, которая равна y*10000 + m*100 + d, а потом сравниваем её для первого предложения с min1, для второго я min2.Если value меньше min(1,2), то тогда мы записываем в min(1,2) значение value. После этого сравниваем min1 и min2. Если min1 больше min2, то компоратор вернёт значение 1 и поменяет два предложения местами. Если min2 больше min1, то всё останется на своих местах.

1.6. Функция, удаляющая все предложения, в которых все даты относятся к 19 веку.

```
int delete_sentenses_of_19th_century(int new_sent_count, char** txt){
        int d,m,y,f,i,j,count;
        char* t = " ";
        for(i = 0 ; i < new_sent_count;i++){
                count = 1;
                for(j = 0; j < strlen(txt[i]);j++){
                        if (strcasecmp(t,&txt[i][j]) == 0){
                        count++;
                        }
                }
                f = 0;
for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m, &y) > 0 ; p += 11){if(y < 1900 && y > 1800){
                          f++;
                        }
                }
                if(f == count){
                        free(txt[i]);
                        new_sent_count--;
                }
        }
        return(new_sent_count);
}
```

Функция принимает на вход текст и кол-во предложений. После чего пробегается по предложению, находит кол-во слов в нём, записывает их в переменную count. Потом ещё раз пробегается по строке и записывает по тому же принципу, что описан выше в функциях 1.3,1.5. Потом сравнивает значение года с 1900 и с 1800 годами, если год находится между 1900 и 1800 годом соответственно то к переменной f прибавляется 1.Если f = count, то мы удаляем это предложение и освобождаем память.

1.7.Функция печатающая для каждого предложения самую раннюю и самую позднюю даты.

```
void print_soon_and_late_date(int new_sent_count, char** txt){
        int d,y,m,i,j;
        int min,max;
        for(i = 0 ; i<new_sent_count ; i++){
                min = 1000000000;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value<min){
                                min = value;
                        }
                }
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value == min){
                                printf("%d/%d/%d ",d,m,y);
                                break;
                        }
                }
                max = 0;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value>max){
                                max = value;
                        }
                }
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
```

```
                        int value = y*10000 + m*100 + d;
                        if(value = max){
                                printf("%d/%d/%d\n",d,m,y);
                                break;
                        }
                }

        }
}
```

Так же как и в предыдущих функциях, функция получает текст и кол-во предложений, пробегается по этим предложениям, но теперь для одного предложения мы находим самую раннюю и позднюю дату, поэтому заводим переменную min (значением 1000000000), сравниваем её с каждым value, записываем в неё минимальный value, после чего ещё раз пробегаемся по предложению и сравниваем с каждым value, пока мы не найдём value равный min, когда мы его находим то печатаем значения в формате %d/%d/%d, DD/MM/YYYY (d,m,y). Так же делаем с max (значение равно 0), но только записываем в него максимальное значение. Потом так же печатаем его на экран, только с учётом пробела.

1.8.1.Функция печатающая текст на экран.

```
 void print(char** txt, int new_sent_count){
        char** s;


        for (s = txt; s < txt+new_sent_count; s++)
                printf("%s", *s);
        printf("\n");
}
```

Функция печатает текст на экран, пока не закончится цикл.

1.8.2Функция освобождающая память.

```
 void free_txt(int new_sent_count, char*** txt){
        for (int i =0;i<new_sent_count;i++)
        free((*txt)[i]);
    free(*txt);
}
```

Функция освобождающая память с помощью цикла.

1.8.3.Функция проверяющая правильность ввода.

```
     int error_check(int new_sent_count, char** txt,int f){
        int d,m,y,i,n;
        f = 0;
        for(i = 0 ; i<new_sent_count ; i++){
                n = i+1;
```

```
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        if(d<0 || m<0 || y<0){
                                f = 1;
                                puts("Значение года, месяца или дня
не могут быть отрицательными. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(m == 0 || m > 12){
                                f = 1;
                                puts("Колличество месяцев не может
быть больше 12 или равнять 0. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d == 0 || d > 31){
                                f = 1;
                                puts("Колличество дней в месяце не
может превышать 31. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d > 30 && (m!= 3 || m!=5 || m!=8 ||
m!=10)){
                                f = 1;
                                puts("Колличество дней в Апреле,
Июле, Сентябре, Ноябре ровняется 30. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
```

```
                        if(d == 29 && m == 3 && ((y%4>=0) &&
((y%100>=0) && (y%400>=0)) && (y%100==0))){
                                f = 1;
                                puts("Колличество дней в феврале
зависит от того,високосный ли год. \n");
                                printf("%d год не является
високосным. \n",y);
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d>29 && m == 3){
                                f = 1;
                                puts("Колличество дней в феврале не
может быть больше 29. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                }
        }
        return f;
}
```

ЗАКЛЮЧЕНИЕ

Для успешного достижения поставленной цели — написания программы для работы с текстом, соответствующей заданию курсовой работы, были выполнены соответствующие задачи:

1. Изучен теоретический материал по теме курсовой работы.
2. Разработан программный код.
3. Реализован программный код.
4. Проведено тестирование программы.

Исходный код программы представлен в приложении А, результаты тестирования - в приложении Б.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.

2. Основы программирования на языках Си и C++ [Электронный ресурс URL: http://cplusplus.com

3. Сайт CPPStudio - Функция localtime: преобразовывает секунды в текущую дату (cppstudio.com)

4. Сайт Stack Overflow - c++ - Как определить какая из данных дат ранняя, а какая поздняя? C++ - Stack Overflow на русском

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

int error_check(int new_sent_count, char** txt,int f);
void sort_sentenses(int new_sent_count, char*** txt);
int cmp(const void* a,const void* b);
void print_soon_and_late_date(int new_sent_count, char** txt);
int delete_sentenses_of_19th_century(int new_sent_count, char**
txt);
void print_sentenses_with_curent_date(int new_sent_count,char**
txt);
int delete_sent(int sent_count, char*** txt);
int read_arr(char* sent_sep, char*** txt);
void free_txt(int new_sent_count, char*** txt);
void print(char** text, int new_sent_count);
char main(){
        puts("Введите строку.Если вы желаете прекратить ввод,
введите пустую строку.");
        int comand = 0;
        int f = 0;
        char** txt = malloc(sizeof(char*));
        char* sent_sep = ".";
        int sent_count = read_arr(sent_sep, &txt);
        int new_sent_count = delete_sent(sent_count, &txt);
        error_check(new_sent_count, txt, f);
        switch(f){
                case 1:
                {
                        exit(0);
                        break;
                }
                case 0:
                {
                        print(txt, new_sent_count);
                        while(1){
```

```
                        puts("1 - Вывести все предложения в которых
есть дата с текущим годом и месяцем.");
                        puts("2 - Отсортировать предложения по
увеличению минимальной даты в них.");
                        puts("3 - Удалить все предложения в которых
все даты относятся к 19 веку.");
                        puts("4 - Для каждого предложения вывести
самую раннюю и позднюю дату.");
                        puts("5 - Выход.");
                        puts("\n");
                        scanf("%d",&comand);
                        switch(comand){
                                case 1 :
                                {       puts("\n");

print_sentenses_with_curent_date(new_sent_count,txt);
                                        puts("\n");
                                        break;
                                }
                                case 2:{
                                        puts("\n");

sort_sentenses(new_sent_count,&txt);
                                        print(txt,new_sent_count);
                                        puts("\n");
                                        break;
                                }
                                case 3:
                                {

delete_sentenses_of_19th_century(new_sent_count,txt);
                                        print(txt,new_sent_count);
                                        puts("\n");
                                        break;
                                }
                                case 4:
                                {       puts("\n");

print_soon_and_late_date(new_sent_count,txt);
                                        puts("\n");
```

```
                                        print(txt,new_sent_count);
                                        puts("\n");
                                        break;
                                }
                                case 5:
                                {
                                        return 0;
                                }
                                default:
                                return 0;
                                }
                        }
                }
        }
        free_txt(new_sent_count, &txt);
        return 0;
}
int read_arr(char* sen_sep, char*** txt){
    int a = 0;
    char str;
    int flag_sent = 1;
    int flag_txt = 1;
    int sent_count = 0;
    int char_count = 0;
    int string_length = 1000;
    while (1)
    {
        char_count = 0;
        (*txt) = realloc((*txt), sizeof(char*) * (sent_count+1));
        if ((*txt) == NULL){
                        puts("Недостаточно памяти!");
                        exit(0);
                }
        (*txt)[sent_count] = malloc(sizeof(char) * string_length);
        str = getchar();
        if (str == '\n')
        {
            a++;
            if (a==2)
```

```
                break;
        }
        else{
                (*txt)[sent_count][char_count] = str;
                a=0;
        }

        do
        {
            char_count++;
            str = getchar();
            if ((str == '\n')){
                a++;
                break;
            }
            else{
                a = 0;
            }
            (*txt)[sent_count][char_count] = str;
            flag_sent = (strchr(sen_sep,str) == NULL);
            if (char_count>=string_length - 2)
            {
                string_length = string_length + 500;
                (*txt)[sent_count] = (char*)
realloc((*txt)[sent_count], string_length);
                if ((*txt)[sent_count] == NULL){
                        puts("Недостаточно памяти!");
                        exit(0);
                }
            }
        }
        while(flag_sent == 1);
        if (a == 2)
            break;
        str=getchar();
        if (str == '\n')
                a++;
        else
                a = 0;
```

```
        (*txt)[sent_count][char_count+1] = ' ';
        (*txt)[sent_count][char_count+2] = '\0';
        sent_count++;
    }
    return sent_count;
}
int error_check(int new_sent_count, char** txt,int f){
        int d,m,y,i,n;
        f = 0;
        for(i = 0 ; i<new_sent_count ; i++){
                n = i+1;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        if(d<0 || m<0 || y<0){
                                f = 1;
                                puts("Значение года, месяца или дня
не могут быть отрицательными. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(m == 0 || m > 12){
                                f = 1;
                                puts("Колличество месяцев не может
быть больше 12 или равнять 0. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d == 0 || d > 31){
                                f = 1;
                                puts("Колличество дней в месяце не
может превышать 31. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d > 30 && (m!= 3 || m!=5 || m!=8 ||
m!=10)){
                                f = 1;
```

```
                                puts("Колличество дней в Апреле,
Июле, Сентябре, Ноябре ровняется 30. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d == 29 && m == 3 && ((y%4>=0) &&
((y%100>=0) && (y%400>=0)) && (y%100==0))){
                                f = 1;
                                puts("Колличество дней в феврале
зависит от того,високосный ли год. \n");
                                printf("%d год не является
високосным. \n",y);
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                        if(d>29 && m == 3){
                                f = 1;
                                puts("Колличество дней в феврале не
может быть больше 29. \n");
                                printf("Ошибка в предложении номер
%d. \n",n);
                                break;
                        }
                }
        }
        return f;
}
int delete_sent(int sent_count, char*** txt){
    for(int i = 0; i<sent_count; i++){

            int j = i+1;

            while (j<sent_count){
                if (strcasecmp((*txt)[i],(*txt)[j])==0){

                    free((*txt)[j]);
                    sent_count--;
                    for (int k = j; k<=sent_count;k++){
```

```
                        (*txt)[k] = (*txt)[k+1];
                    }
                }
                else{
                    j++;
                }
            }
    }
    return sent_count;
}
void print_sentenses_with_curent_date(int new_sent_count, char**
txt){
        time_t t = time(NULL);
        struct tm* timeinfo = localtime(&t);
        int d,m,y,f,i;
        for(i = 0 ; i<new_sent_count ; i++){
                f = 0;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        if(y == timeinfo->tm_year+1900  &&  m ==
timeinfo->tm_mon+1){
                          f = 1;
                          break;
                        }
                }
                if(f) printf("%s\n",txt[i]);
        }
}

int delete_sentenses_of_19th_century(int new_sent_count, char**
txt){
        int d,m,y,f,i,j,count;
        char* t = " ";
        for(i = 0 ; i < new_sent_count;i++){
                count = 1;
                for(j = 0; j < strlen(txt[i]);j++){
                        if (strcasecmp(t,&txt[i][j]) == 0){
                        count++;
                        }
                }
```

```
                f = 0;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        if(y < 1900 && y > 1800){
                          f++;
                        }
                }
                if(f == count){
                        free(txt[i]);
                        new_sent_count--;
                }
        }
        return(new_sent_count);
}
void print_soon_and_late_date(int new_sent_count, char** txt){
        int d,y,m,i,j;
        int min,max;
        for(i = 0 ; i<new_sent_count ; i++){
                min = 1000000000;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value<min){
                                min = value;
                        }
                }
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value == min){
                                printf("%d/%d/%d ",d,m,y);
                                break;
                        }
                }
                max = 0;
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value>max){
                                max = value;
```

```
                        }
                }
                for(char* p = txt[i]; sscanf(p, "%d/%d/%d", &d, &m,
&y) > 0 ; p += 11){
                        int value = y*10000 + m*100 + d;
                        if(value = max){
                                printf("%d/%d/%d\n",d,m,y);
                                break;
                        }
                }

        }
}
void sort_sentenses(int new_sent_count,char*** txt){
        qsort(*txt,new_sent_count,sizeof(char*),cmp);
}
int cmp(const void* a,const void* b){
        int d,m,y,i;
        int min1,min2;
        char **s1 = (char**) a;
        char **s2 = (char**) b;
        min1 = 1000000000;
        min2 = 1000000000;
        for(char* p = *s1; sscanf(p, "%d/%d/%d", &d, &m, &y) > 0 ;
p += 11){
                int value = y*10000 + m*100 + d;
                if(value<min1){
                        min1=value;
                }
        }
        for(char* p = *s2; sscanf(p, "%d/%d/%d", &d, &m, &y) > 0 ;
p += 11){
                int value = y*10000 + m*100 + d;
                if(value<min1){
                        min2=value;
                }
        }
        if(min1>min2){
                return 1;
```

```
        }
        if(min2>min1){
                return -1;
        }
        if(min1=min2){
                return 0;
        }
}

void free_txt(int new_sent_count, char*** txt){
        for (int i =0;i<new_sent_count;i++)
        free((*txt)[i]);
    free(*txt);
}
void print(char** txt, int new_sent_count){
        char** s;

        for (s = txt; s < txt+new_sent_count; s++)
                printf("%s", *s);
        printf("\n");
}
```

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТА ТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

```
Введите строку.Если вы желаете прекратить ввод, введите пустую строку.
20/12/2003 12/12/2021. 20/12/2003 20/12/2005.

20/12/2003 12/12/2021. 20/12/2003 20/12/2005.
1 - Вывести все предложения в которых есть дата с текущим годом и месяцем.
2 - Отсортировать предложения по увеличению минимальной даты в них.
3 - Удалить все предложения в которых все даты относятся к 19 веку.
4 - Для каждого предложения вывести самую раннюю и позднюю дату.
5 - Выход.


1



20/12/2003 12/12/2021.


1 - Вывести все предложения в которых есть дата с текущим годом и месяцем.
2 - Отсортировать предложения по увеличению минимальной даты в них.
3 - Удалить все предложения в которых все даты относятся к 19 веку.
4 - Для каждого предложения вывести самую раннюю и позднюю дату.
5 - Выход.
```

```
Введите строку.Если вы желаете прекратить ввод, введите пустую строку.
20/12/2003 12/12/2021. 20/12/2003 20/12/2005.

20/12/2003 12/12/2021. 20/12/2003 20/12/2005.
1 - Вывести все предложения в которых есть дата с текущим годом и месяцем.
2 - Отсортировать предложения по увеличению минимальной даты в них.
3 - Удалить все предложения в которых все даты относятся к 19 веку.
4 - Для каждого предложения вывести самую раннюю и позднюю дату.
5 - Выход.
```