**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Программирование»**

**Тема: Обработка текста**

Студент гр. 8383 Федоров И.А

Преподаватель Круглик А.Д

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на курсовую работу**

|  |
| --- |
| Студент Федоров И.А |
| Группа 8383 |
| Тема работы : Обработка текста |
| Исходные данные:  Программе на вход подается текст (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 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. Вывести все предложения в которых есть дата с текущим годом и месяцем. (дата формата DD/MM/YYYY). 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 даты.  * 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 функций. |

Предполагаемый объем пояснительной записки:

Не менее страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент Федоров И.А.

Преподаватель

**АННОТАЦИЯ**

При выполнении лабораторной работы на языке программирования C была написана программа, позволяющая работать с текстом и содержащая следующие функции:

* функция динамического считывания текста и записи его в динамический двумерный массив строк.
* функция вывода текста на экран.
* функция удаляющая повторяющиеся предложения.
* функция удаления предложений, все даты которых относятся к 19 веку.
* функция поиска предложений, содержащих дату текущего месяца и года.
* функция сортировки предложений по увеличению их минимальной даты.
* функция выводы минимальной и максимальной даты каждого предложения.
* функция очистки памяти при выходе из программы .

**СОДЕРЖАНИЕ**

**Оглавление**

**ВВЕДЕНИЕ** 4

**1. ОПИСАНИЕ ПРОГРАММЫ** 6

1.1 Подключенные библиотеки 6

1.2. Функция main 6

1.3 Функция memory\_text 8

1.4 Функция delete\_same 9

1.5 Функция min\_date\_and\_max\_date 10

1.6 Функция delete\_XIX\_century 11

1.7 Функция find\_date 12

1.8 Функция free\_Memory 13

1.9 Функция сomp 13

1.10 Функция output\_text 13

**2.ТЕСТИРОВАНИЕ ПРОГРАММЫ** 12

2.1 Тест функции find\_date 14

2.2. Тест функции delete\_XIX\_century 14

2.3 Тест функции min\_date\_and\_max\_date 15

2.4 Тест функции sort 16

**ЗАКЛЮЧЕНИЕ** 16

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 17

**ПРИЛОЖЕНИЕ** 18

***1. ОПИСАНИЕ ПРОГРАММЫ***

**1.1 Подключенные библиотеки**

Были подключены:

1. Заголовочный файл **<stdio.h>,** содержащий функции ввода и вывода
2. Заголовочный файл **<malloc.h>** , который содержит в себе функции, занимающиеся выделением памяти.
3. Заголовочный файл **<string.h>** , содержащий в себе функции для работы со строками
4. Заголовочный файл **<ctype.h>**, содержащий в себе функции для работы c отдельными символами.
5. Заголовочный файл **<stdlib.h>** ,содержащий в себе функции для работы с динамической памятью, для преобразования строки в число, функции для сортировки и поиска.

**1.2. Функция main**

Выделяется память для двумерного динамического массива (состоит из массива указателей на одномерные массивы строк) . Num\_sent – переменная, которая сначала будет хранить в себе индекс сохраняемого предложения, а потом число предложений в тексте. Выводится подсказка для пользователя, затем последовательно вызываются функции memory\_text , del\_same. Затем пользователю предлагаются доступные действия.

int main(){

printf("Введите текст , разделителями между словами является пробел, а между предложениями - точка\n");

int len\_text = 0;

char choice;

char opp;

char \*\*dynamic\_array = (char\*\*)malloc(sizeof(char\*));

int i = 0;

int q = 0;

int k, n , m = 0;

int num\_sent = 0;

dynamic\_array = memory\_text(dynamic\_array, &num\_sent);

dynamic\_array = delete\_same(dynamic\_array, &num\_sent);

while (q != 1) {

printf("\nВведите 1, ecли вы хотите вывести все предложения, в которых есть дата с текущем месяцем и годом.\n");

printf("\nВведите 2, ecли вы хотите отсортировать предложения по увеличению минимальной даты в них.\n");

printf("\nВведите 3, ecли вы хотите для каждого предложения вывести самую раннюю и позднюю дату.\n");

printf("\nВведите 4, ecли вы хотите удалить все предложения в которых все даты относятся к 19 веку\n");

printf("\nВведите 5, если вы хотите закончить работу программы.\n");

choice = getchar();

opp = getchar();

switch (choice) {

case '1':

find\_date(dynamic\_array, &num\_sent);

break;

case '2':

delete\_XIX\_century(dynamic\_array,&num\_sent);

output\_text(dynamic\_array, &num\_sent);

break;

case '3':

min\_date\_and\_max\_date(dynamic\_array,&num\_sent);

break;

case '4':

qsort(dynamic\_array, num\_sent, sizeof(char\*), comp);

output\_text(dynamic\_array, &num\_sent);

break;

case '5':

printf("%s\nGoodbye, User\n%s", GREEN, NONE);

free\_Memory(dynamic\_array, num\_sent);

q = 1;

break;

default:

printf("%sНеправильный ввод!%s", RED, NONE);

}

}

return 0; }

**1.3 Функция memory\_text**

Функция отвечает за сохранение текста в динамический двумерный массив . Str\_len – переменная, содержащая в себе значения для выделения и перевыделения памяти для предложения (По умолчанию = 12 , на 1 больше длины одной даты). Symb – переменная для посимвольного сохранения текста. Letters – переменная, содержащая в себе индекс символа в предложении. в цикле выделяется память для каждого предложения. Текст забивается в двумерный динамический массив посимвольно. При сохранении каждого символа проводится проверка на достаточность выделенной памяти, если индекс символа в предложении подходит к размеру выделенной памяти, происходит ее перевыделение. При сохранении символа ‘.’, выделяется память под очередное предложение и программа переходит к заполнению следующего предложения.

char\*\* memory\_text(char \*\*dynamic\_array, int \*num\_sent){

int str\_len = 12;

char symb;

int letters = 0;

while (symb != '\n'){

dynamic\_array[\*num\_sent] = (char\*)malloc(str\_len \* sizeof(char));

while (symb != '.' && symb != '\n'){

symb = getchar();

if (symb == '\n'){

break;

}

if (letters >= str\_len - 1){

str\_len += str\_len;

dynamic\_array[\*num\_sent]=(char\*)realloc(dynamic\_array[\*num\_sent],str\_len\*sizeof(char) ) ;

}

dynamic\_array[\*num\_sent][letters++] = symb;

}

if (symb == '\n'){

break;

}

symb = '\0';

dynamic\_array[\*num\_sent][letters] = symb;

\*num\_sent += 1 ;

dynamic\_array = (char\*\*)realloc(dynamic\_array, (\*num\_sent + 1) \* sizeof(char\*));

letters = 0;

str\_len = 12;

}

return dynamic\_array;

}

**1.4 Функция delete\_same**

Функция отвечает за удаление одинаковых предложений в исходном тексте. Count – переменная, считающая количество одинаковых символов в двух предложениях. Все предложения, начиная с первого, сравниваются посимвольно с предыдущими (при условии, что длины предложений равны, т.к в противном случае сравнение не требуется) и если количество одинаковых символов окажется равным количеству символов в предложении, повторное предложение удаляется сдвига предложений (указателей на них) и применения функции **realloc.** Переменная, содержащая количество предложений в тексте, уменьшается на 1 перед использования функции realloc. Переменная count обнуляется.

char\*\* delete\_same(char \*\*dynamic\_array, int \*num\_sent){

int count = 0;

for(int sentence1 = 0; sentence1 < \*num\_sent - 1; sentence1++){

for (int sentence2 = sentence1 + 1; sentence2 < \*num\_sent; sentence2++){

if (strlen(dynamic\_array[sentence1]) == strlen(dynamic\_array[sentence2]))

{

count = 0;

for(int j = 0; j < strlen(dynamic\_array[sentence1]); j++){

if (toupper(dynamic\_array[sentence1][j]) == toupper(dynamic\_array[sentence2][j])){

++count;

}

}

if (count == strlen(dynamic\_array[sentence1])){

free(dynamic\_array[sentence2]);

for (int l = sentence2; l < \*num\_sent; l++){

dynamic\_array[l] = dynamic\_array[l+1];

}

sentence2--;

(\*num\_sent)--;

dynamic\_array = (char\*\*)realloc(dynamic\_array,(\*num\_sent)\*sizeof(char\*));

}

}

}

}

return dynamic\_array;

}

**1.5 Функция min\_date\_and\_max\_date**

Функция для вывода минимальной и максимальной даты.

В функции используется структура для хранения дат:

typedef struct {

int year;

int month;

int day;

}date;

Переменная size – первоначальный размер массива dates, в котором будут содержаться даты. Предложения по очереди копируются в буфер.Е сли все условия по формату соблюдены, день, месяц и дата сохраняются в массивы, после чего преобразовываются в целочисленный тип и добавляются в массив структур.

Далее все даты предложения сортируются при помощи qsort и функции-компаратора compare:

int compare (const date \*a, const date \*b){

return a->year \* 10000 + a->month \* 100 + a->day - b->year \* 10000 - b-

>month \* 100 - b->day;

}

Далее выводятся минимальна и максимальная дата предложения, если же в предложении всего одная дата или их вообще нет, функция выводит сообщение.

**1.6 Функция delete\_XIX\_century**

Функция удаляет все предложения , в которых каждая дата относится к 19 веку . Count1 – переменная, запоминающая количество дат в предложени. Сount2 - переменная, запоминающая количество дат, относящихся к 19 веку. Если их значения совпадают, строки массива сдвигаются и функцией **realloc** удаляется предложение. По завершении работы с предложением память, выделенная под буфер, освобождается, а переменные count1 и count2 обнуляются.

char\*\* delete\_XIX\_century(char \*\*dynamic\_array, int \*num\_sent){

int count1 = 0;

int count2 = 0;

char \*bufer;

char \*instr;

char fstr[] = "/18";

for (int i = 0; i < \*num\_sent; i++){

bufer = (char\*)malloc((strlen(dynamic\_array[i]))\*sizeof(char));

memcpy(bufer, dynamic\_array[i], (strlen(dynamic\_array[i])+1));

char \*str = strtok(bufer, " "); /////" ,.\0"

while(str != NULL){

count1++;

instr = strstr(str,fstr);

if (instr != NULL) count2++;

str = strtok(NULL, " ");

}

if (count1 == count2){

for (int j = i; j < \*num\_sent - 1; j++){

dynamic\_array[j] = dynamic\_array[j+1];

}

i -= 1;

dynamic\_array[\*num\_sent - 1] = NULL;

\*num\_sent -= 1;

}

free(bufer);

count1 = 0;

count2 = 0;

}

return dynamic\_array;

}

**1.7 Функция find\_date**

Функция выводит предложение, если в нем содержится дата с текущем месяцем и годом. Count – переменная, считающая количество предложений с такой датой. Если после прохода по всем предложениям count равен нулю, то функция сообщает о том, что таких предложений найдено не было.

void find\_date(char \*\*dynamic\_array,int \*num\_sent)

{

char str[] = "12/2018";

char \*instr;

int count = 0;

for (int i = 0 ; i < \*num\_sent; i++)

{

instr = strstr(dynamic\_array[i],str);

if (instr != NULL){

count ++;

for (int j = 0; j < strlen(dynamic\_array[i]); j++)

{

printf("%c", dynamic\_array[i][j]);

}

printf("\n");

}

}

if (count==0) printf("Даты не нашлось"); }

**1.8 Функция free\_Memory**

Функция очищает память после завершения программы. Так как массив двумерный, то функция сначала удаляет каждое предложение, а затем удаляет массив указателей на эти предложения.

void free\_Memory(char \*\*dynamic\_array, int n){

for (int i = 0; i < n ; i++){

free(dynamic\_array[i]);

}

free(dynamic\_array);

}

**1.9 Функция comp**

Функция-компаратор для qsort. Функция находит минимальные даты в предложениях, и затем возвращает -1 0 или 1 в зависимости от результата сравнения.

int comp(const char\* a, const char\* b){

const char \*str1 = \*( const char \* const \*) a;

const char \*str2 = \*( const char \* const \*) b;

**1.10 output\_text**

Функция для вывода текста. Поочерёдно выводится каждое предложение. Если число предложений равно нулю (переменная num\_sent равна нулю), об этом выведется сообщение.

void output\_text(char \*\*dynamic\_array, int \*num\_sent) {

if ((\*num\_sent) == 0){

printf("%sПредложений нет :(%s", YELLOW, NONE);

printf("\n");

}

else {

for (int i = 0; i < \*num\_sent; i++)

{

if (dynamic\_array[i] != NULL){

printf("%s%s%s", GREEN, dynamic\_array[i], NONE);}

}

}

}

**2.ТЕСТИРОВАНИЕ ПРОГРАММЫ**

**2.1 Тест функции find\_date**

|  |  |
| --- | --- |
| Входные данные: | **22/11/1703 01/12/1809 11/08/2000. 21/11/0905 22/10/0801. 01/01/1555 11/12/1311 . 22/11/1900 22/12/1809 11/08/2000. 21/11/0905 22/10/0801 . 01/01/1555 11/12/1311. 22/11/1900 22/12/1809 11/08/2000. 21/11/0905 22/10/0801 . 01/01/1555 11/12/1311.** |
| **Вывод:** | **Даты не нашлось** |

|  |  |
| --- | --- |
| Входные данные: | *22/12/2018* **01/01/1555 11/12/1311 skjfgkf fgjk. 21/11/0905 22/10/0801 sfdsf . 01/01/1555 11/12/1311 . 22/11/1900 22/12/1809 11/08/2000 sdfdsfs. 21/11/0905 22/10/0801 . 01/01/1555 11/12/1311 dfsfdsf s. 22/11/1900 22/12/1809 11/08/2000. 21/11/0905 22/10/0801 . 01/01/1555 11/12/1311.** |
| **Вывод:** | 22/12/2018   01/01/1555  11/12/1311 skjfgkf fgjk. |

**2.2 Тест функции delete\_XIX\_century**

|  |  |
| --- | --- |
| Входные данные: | **01/01/1000 22/10/0901 23/11/1212.** *25/09/1812**13/03/1888* **12/01/0111 .** *13/04/1801 12/12/1812 29/11/1899***. 25/11/1111 22/03/0987.** |
| **Вывод:** | **01/01/1000 22/10/0901 23/11/1212.** *25/09/1812**13/03/1888* **12/01/0111 .25/11/1111 22/03/0987.** |

|  |  |
| --- | --- |
| Входные данные: | **01/01/1000 22/10/0901 23/11/1212.** *25/09/1812**13/03/1888* **12/01/0111ds fdsf .** *13/04/1801 12/12/1812 29/11/1****9****99***. 25/11/1111 fgsdfgggdg 22/03/0987.** |
| **Вывод:** | **01/01/1000 22/10/0901 23/11/1212.** *25/09/1812**13/03/1888* **12/01/0111 fsdfsdfdsf . fdgfdgfdg***13/04/1801 12/12/1812 29/11/1****9****99***. fdg25/11/1111 22/03/0987.** |

**2.3 Тест функции min\_date\_and\_max\_date**

|  |  |
| --- | --- |
| Входные данные: | 25/11/1900 13/01/0897 23/11/1911. 01/01/2000 23/11/1892 25/11/0101. 31/12/2018 21/11/2018 23/12/2011. |
| **Вывод:** | Минимальная и Максимальная дата для 1 -го предложения: 13/01/0897 23/11/1911  Минимальная и Максимальная дата для 2 -го предложения: 25/11/0101 01/01/2000  Минимальная и Максимальная дата для 3 -го предложения: 23/12/2011 31/12/2018 |

|  |  |
| --- | --- |
| Входные данные: | 25/11/2014. dfdsf dfdsf 234 sdfdsfsdf. |
| Вывод: | Дат в предложении нет или дата всего одна. |

**2.3 Тест сортировки**

|  |  |
| --- | --- |
| **Входные данные:** | 01/01/1000 ***22/10/0901*** 23/11/1212. *25/09/1812* *13/03/1888* ***12/01/0111*** . ***13/04/1801*** *12/12/1812 29/11/1899*. 25/11/1111 ***22/03/0987***. 01/01/1000 22/10/0901 23/11/1212. *25/09/1812* *13/03/1888* 12/01/0111 . *13/04/1801 12/12/1812 29/11/1899*. 25/11/1111 22/03/0987. |
| **Вывод:** | *25/09/1812* *13/03/1888* ***12/01/0111*** . 01/01/1000 ***22/10/0901*** 23/11/1212. 25/11/1111 ***22/03/0987***. . ***13/04/1801*** *12/12/1812 29/11/1899*. |

|  |  |
| --- | --- |
| **Входные данные:** | 25/11/1900 **13/01/0897** 23/11/1911. 01/01/2000 23/11/1892 **25/11/0101**. 31/12/2018 21/11/2018 **23/12/2011**. 25/09/1812 13/03/1888 **12/01/0111** .  12/12/1212 12/12/1212 . |
| **Вывод:** | 01/01/2000 23/11/1892 **25/11/0101.** 25/09/1812 13/03/1888 **12/01/0111** . 25/11/1900 **13/01/0897** 23/11/1911. 12/12/1212 12/12/1212 . |

**ЗАКЛЮЧЕНИЕ**

* ходе выполнения работы была создана программа для ввода, редактирования и вывода текста состоящего из дат. В

В ходе данной работы была написана программа по ввод и обработке текста состоящего из дата. Все подзадачи были реализованы в виде отдельных функций. Все сортировки были осуществлены с использованием функции стандартной библиотеки языка Си. После проведения тестов можно сделать вывод, что программа работает корректно.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Cpluplus.com. URL: http://www.cplusplus.com (дата обращения 18.12.2018)

[cppstudio](http://yandex.ru/clck/jsredir?bu=9x6e&from=yandex.ru%3Bsearch%2F%3Bweb%3B%3B&text=&etext=2006.rLBEjLHXw9vdM6C04DpdpKStninnDZADiYy40ETn3O0.a39854bdf5c82f507d3c990362bc14bc7420bd17&uuid=&state=PEtFfuTeVD4jaxywoSUvtHP1ewhXQEMzHZiDCUtHOY71mlnIME-qGyFnuwy8gZCu&&cst=AiuY0DBWFJ5fN_r-AEszkz6A4B_qVl2QiRpoBKkmyxyd4cz0m8SCoq5mTMXUpeSave4IgXwUAZQKJ9X0Bl7nbJ-QKgv-SgzkgP-LrZfMVoQxTor9YoTUmNtEJKmeBNN50EGpNtY04hD5fQdYhwL9uoWQGpLz415r012J8_zRg4Q_g7z6umL5AcX_uiCs7CJ3DfDZT-eWZg46_WFWqpxbwoaFe0dllCvon83NYSHih1LjhMiTdKwTZpd1rustf0tsi6tE1QvdBB3Ia_PKt0jHVaTaiIQGynkkiAxYZryPHWv1c6Ii_jTiEtGRz42EnezVW_-wY7uGhP7NpE8ybcQz4iqtlR7rwxzO3_YoU303KR1xBmsX5MAUyRculRNxSZBB_JCrNG-FEvg1gs7Fz68bF3xFlXahwDKluAmhmBm90LmVpiNtRPCWn154SGpTF7dB_ndhHYEu-Q_vUdZnb7XRLWHPWqo4UTv23mhj6HbVxdglyZoL5tAQXWEKzwn4eScuueLvea_JnjdsdVcVLT_oIKHvYQuBvj2mACYzfEfr4OQ_yP1O8ejy33_0T5xyT814KyH6usZv7--4YvZj9aJMc2fGOH42qWkIwZgo8zfIX5OX-HNonWZyeef7O1N4QHvWfCcHTQwM2B87UYWX_e5ccIpbXvIUef-O1hJADJiICbB1RZN-BQNjSK6e0Gft573oE1BG5Pkz38RT9usBoTxwO_WvHagrFsLs4aA5ThZoBJJKyl9cV1oralhdV_Elob84WWW0y8Df2YaM-3iX9DhHAw,,&data=UlNrNmk5WktYejR0eWJFYk1LdmtxbVlNLTNUZXh6aWFNT3hobFBVS3Rpa0pLZi01c24zZk5SQ25HVVNPUmhqUWZCekI2MVZrMTRxZFZ5MkNaNENIUndGeU5KU0dIYml0&sign=0beb91f7dce1bfdde35faaa7378c4ea6&keyno=0&b64e=2&ref=orjY4mGPRjk5boDnW0uvlrrd71vZw9kpfmwSlf_6ZUZJNqUmmGIlr_Es0MXInCFOJ_biCUjZBwY,&l10n=ru&rp=1&cts=1545256386882&mc=3.916126946588283&hdtime=8697). URL: http://cppstudio.com/ (дата обращения 17.12.2018)

Prog-cpp.ru. URL: https://prog-cpp.ru/ (дата обращения 17.12.2018)

**Приложение (код программы)**

**#include <malloc.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <ctype.h>**

**#define RED "\033[0;31m"**

**#define NONE "\033[0m"**

**#define GREEN "\033[0;32m"**

**#define MAX 1190000**

**#define YELLOW "\033[0;33m"**

**typedef struct {**

**int year;**

**int month;**

**int day;**

**}date;**

**int compare(const date \*a, const date \*b);**

**char\*\* memory\_text(char \*\*dynamic\_array, int \*num\_sent);**

**char\*\* delete\_same(char \*\*dynamic\_array, int \*num\_sent);**

**void find\_date(char \*\*dynamic\_array, int \*num\_sent);**

**char\*\* delete\_XIX\_century(char \*\*dynamic\_array, int \*num\_sent);**

**void min\_date\_and\_max\_date(char \*\*dynamic\_array, int \*num\_sent);**

**void output\_text(char \*\*dynamic\_array, int \*num\_sent);**

**char\*\* sort\_Sentence(char\*\* dynamic\_array, int\* num\_sent);**

**int comp(const char\* a, const char\* b);**

**int main(){**

**printf("Введите текст , разделителями между словами является пробел, а между предложениями - точка\n");**

**int len\_text = 0;**

**char choice;**

**char opp;**

**char \*\*dynamic\_array = (char\*\*)malloc(sizeof(char\*));**

**int i = 0;**

**int q = 0;**

**int k, n , m = 0;**

**int num\_sent = 0;**

**dynamic\_array = memory\_text(dynamic\_array, &num\_sent);**

**dynamic\_array = delete\_same(dynamic\_array, &num\_sent);**

**while (q != 1) {**

**printf("\nВведите 1, ecли вы хотите вывести все предложения, в которых есть дата с текущем месяцем и годом.\n");**

**printf("\nВведите 2, ecли вы хотите отсортировать предложения по увеличению минимальной даты в них.\n");**

**printf("\nВведите 3, ecли вы хотите для каждого предложения вывести самую раннюю и позднюю дату.\n");**

**printf("\nВведите 4, ecли вы хотите удалить все предложения в которых все даты относятся к 19 веку\n");**

**printf("\nВведите 5, если вы хотите закончить работу программы.\n");**

**choice = getchar();**

**opp = getchar();**

**switch (choice) {**

**case '1':**

**find\_date(dynamic\_array, &num\_sent);**

**break;**

**case '2':**

**delete\_XIX\_century(dynamic\_array,&num\_sent);**

**output\_text(dynamic\_array, &num\_sent);**

**break;**

**case '3':**

**min\_date\_and\_max\_date(dynamic\_array,&num\_sent);**

**break;**

**case '4':**

**qsort(dynamic\_array, num\_sent, sizeof(char\*), comp);**

**output\_text(dynamic\_array, &num\_sent);**

**break;**

**case '5':**

**printf("%s\nGoodbye, User\n%s", GREEN, NONE);**

**free\_Memory(dynamic\_array, num\_sent);**

**q = 1;**

**break;**

**default:**

**printf("%sНеправильный ввод!%s", RED, NONE);**

**}**

**}**

**return 0;**

**}**

**int compare(const date \*a, const date \*b){**

**return a->year \* 10000 + a->month \* 100 + a->day - b->year \* 10000 - b->month \* 100 - b->day;**

**}**

**char\*\* memory\_text(char \*\*dynamic\_array, int \*num\_sent){**

**int str\_len = 12;**

**char symb;**

**int letters = 0;**

**while (symb != '\n'){**

**dynamic\_array[\*num\_sent] = (char\*)malloc(str\_len \* sizeof(char));**

**while (symb != '.' && symb != '\n'){**

**symb = getchar();**

**if (symb == '\n'){**

**break;**

**}**

**if (letters >= str\_len - 1){**

**str\_len += str\_len;**

**dynamic\_array[\*num\_sent] = (char\*)realloc(dynamic\_array[\*num\_sent], str\_len \* sizeof(char));**

**}**

**dynamic\_array[\*num\_sent][letters++] = symb;**

**}**

**if (symb == '\n'){**

**break;**

**}**

**symb = '\0';**

**dynamic\_array[\*num\_sent][letters] = symb;**

**\*num\_sent += 1 ;**

**dynamic\_array = (char\*\*)realloc(dynamic\_array, (\*num\_sent + 1) \* sizeof(char\*));**

**letters = 0;**

**str\_len = 12;**

**}**

**return dynamic\_array;**

**}**

**char\*\* delete\_same(char \*\*dynamic\_array, int \*num\_sent){**

**int count = 0;**

**for(int sentence1 = 0; sentence1 < \*num\_sent - 1; sentence1++){**

**for (int sentence2 = sentence1 + 1; sentence2 < \*num\_sent; sentence2++){**

**if (strlen(dynamic\_array[sentence1]) == strlen(dynamic\_array[sentence2])){**

**count = 0;**

**for(int j = 0; j < strlen(dynamic\_array[sentence1]); j++){**

**if (toupper(dynamic\_array[sentence1][j]) == toupper(dynamic\_array[sentence2][j])){**

**++count;**

**}**

**}**

**if (count == strlen(dynamic\_array[sentence1])){**

**free(dynamic\_array[sentence2]);**

**for (int l = sentence2; l < \*num\_sent; l++){**

**dynamic\_array[l] = dynamic\_array[l+1];**

**}**

**sentence2--;**

**(\*num\_sent)--;**

**dynamic\_array = (char\*\*)realloc(dynamic\_array,(\*num\_sent)\*sizeof(char\*));**

**}**

**}**

**}**

**}**

**return dynamic\_array;**

**}**

**void output\_text(char \*\*dynamic\_array, int \*num\_sent)**

**{**

**if ((\*num\_sent) == 0){**

**printf("%sПредложений нет :(%s", YELLOW, NONE);**

**printf("\n");**

**}**

**else {**

**for (int i = 0; i < \*num\_sent; i++)**

**{**

**if (dynamic\_array[i] != NULL)**

**{**

**printf("%s%s%s", GREEN, dynamic\_array[i], NONE);**

**}**

**}**

**}**

**}**

**void find\_date(char \*\*dynamic\_array,int \*num\_sent)**

**{**

**char str[] = "12/2018";**

**char \*instr;**

**int count = 0;**

**for (int i = 0 ; i < \*num\_sent; i++)**

**{**

**instr = strstr(dynamic\_array[i],str);**

**if (instr != NULL){**

**count ++;**

**for (int j = 0; j < strlen(dynamic\_array[i]); j++)**

**{**

**printf("%c", dynamic\_array[i][j]);**

**}**

**printf("\n");**

**}**

**}**

**if (count==0) printf("Даты не нашлось");**

**}**

**char\*\* delete\_XIX\_century(char \*\*dynamic\_array, int \*num\_sent){**

**int count1 = 0;**

**int count2 = 0;**

**char \*bufer;**

**char \*instr;**

**char fstr[] = "/18";**

**for (int i = 0; i < \*num\_sent; i++){**

**bufer = (char\*)malloc((strlen(dynamic\_array[i]))\*sizeof(char));**

**memcpy(bufer, dynamic\_array[i], (strlen(dynamic\_array[i])+1));**

**char \*str = strtok(bufer, " "); /////" ,.\0"**

**while(str != NULL){**

**count1++;**

**instr = strstr(str,fstr);**

**if (instr != NULL) count2++;**

**str = strtok(NULL, " ");**

**}**

**if (count1 == count2){**

**for (int j = i; j < \*num\_sent - 1; j++){**

**dynamic\_array[j] = dynamic\_array[j+1];**

**}**

**i -= 1;**

**dynamic\_array[\*num\_sent - 1] = NULL;**

**\*num\_sent -= 1;**

**}**

**free(bufer);**

**count1 = 0;**

**count2 = 0;**

**}**

**return dynamic\_array;**

**}**

**void min\_date\_and\_max\_date(char \*\*dynamic\_array, int \*num\_sent)**

**{**

**char \*d;**

**char \*m;**

**char \*y;**

**int size = 10;**

**char \*bufer;**

**date \*dates = (date\*)malloc(size \* sizeof(date));**

**int index = 0;**

**for (int i = 0; i < \*num\_sent; i++)**

**{**

**bufer = (char\*)malloc((strlen(dynamic\_array[i])+1) \* sizeof(char));**

**memcpy(bufer, dynamic\_array[i], (strlen(dynamic\_array[i])+1) \* sizeof(char));**

**char \*str = strtok(bufer, " \0");**

**while(str != NULL)**

**{**

**if (strlen(str) >= 10) {**

**for (int j = 0; j < strlen(str); j++) {**

**if (isdigit(str[j]) && isdigit(str[j + 1]) && str[j + 2] == '/' && j + 9 < strlen(str)){**

**if (isdigit(str[j + 3]) && isdigit(str[j + 4]) && str[j + 5] == '/'){**

**if ( isdigit(str[j + 6]) && isdigit(str[j + 7]) && isdigit(str[j + 8]) && isdigit(str[j + 9])){**

**if (index >= size){**

**size += 10;**

**dates = (date\*)realloc(dates, size \* sizeof(date)); }**

**d = (char\*)malloc(2 \* sizeof(char));**

**d[0] = str[j]; d[1] = str[j + 1];**

**m = (char\*)malloc(2 \* sizeof(char));**

**m[0] = str[j + 3]; m[1] = str[j + 4];**

**y = (char\*)malloc(4 \* sizeof(char));**

**y[0] = str[j + 6]; y[1] = str[j + 7]; y[2] = str[j + 8] ; y[3] = str[j + 9];**

**dates[index].day = atoi(d);**

**dates[index].month = atoi(m);**

**dates[index].year = atoi(y);**

**free(d); free(m); free(y);**

**index += 1;**

**}**

**}**

**}**

**}**

**}**

**str = strtok(NULL, " \0");**

**}**

**free(bufer);**

**qsort(dates, index, sizeof(date),(int(\*)(const void \*, const void \*)) compare);**

**if (index == 0 || index == 1){**

**printf("%sВ предложении нет дат или дата всего одна\n%s", YELLOW, NONE);**

**}**

**else {**

**printf("%sМинимальная и Максимальная дата для %d -го предложения: %s",YELLOW,i+1,NONE);**

**for (int k = 0; k < index; k += (index - 1)){**

**if (dates[k].day < 10){**

**printf("%s0%s", GREEN, NONE);**

**}**

**printf("%s%d/%s", GREEN, dates[k].day, NONE);**

**if (dates[k].month < 10){**

**printf("%s0%s", GREEN, NONE);**

**}**

**printf("%s%d/%s", GREEN, dates[k].month, NONE);**

**if (dates[k].year < 1000){**

**printf("%s0%s", GREEN, NONE);**

**if (dates[k].year < 100){**

**printf("%s0%s", GREEN, NONE);**

**if (dates[k].year < 10){**

**printf("%s0%s", GREEN, NONE);**

**}**

**}**

**}**

**printf("%s%d%s\n", GREEN, dates[k].year, NONE);**

**}**

**}**

**index = 0;**

**}**

**free(dates);**

**}**

**int comp(const char\* a, const char\* b){**

**const char \*str1 = \*( const char \* const \*) a;**

**const char \*str2 = \*( const char \* const \*) b;**

**int mins\_1 = MAX;**

**int min1 = 0;**

**int mins\_2 = MAX;**

**int min2 = 0;**

**int x;**

**char y;**

**for (int i = 0; i < strlen(str1)-9; i++){**

**if (str1[i] == ' '){**

**i++;**

**}**

**min1 = (y=str1[i] - '0')\*10 + (y=str1[i+1] - '0') + (y=str1[i+3] - '0')\*100 + (y=str1[i+4] - '0')\*10 + (y=str1[i+6] - '0')\*120000 + (y=str1[i+7] - '0')\*10000 + (y = str1[i+8] - '0')\*1000 + (y = str1[i+9] - '0')\*200;**

**if (min1 <= mins\_1) {**

**mins\_1 = min1;**

**}**

**}**

**for (int k = 0; k < strlen(str2)-9; k++){**

**while (str2[k] == ' ') k++;**

**min2 = (y=str2[k] - '0')\*10 + (y=str2[k+1] - '0') + (y=str1[k+3] - '0')\*100 + (y=str2[k+4] - '0')\*10 + (y=str2[k+6] - '0')\*120000 + (y=str2[k+7] - '0')\*10000 + (y = str2[k+8] - '0')\*1000 + (y = str2[k+9] - '0')\*150;**

**if (min2 <= mins\_2) {**

**mins\_2 = min2;**

**}**

**}**

**if (mins\_1 > mins\_2){**

**return 1;**

**}**

**if (mins\_1 == mins\_2){**

**return 0;**

**}**

**if (mins\_1 < mins\_2){**

**return -1;**

**}**

**}**

**void free\_Memory(char \*\*dynamic\_array, int n){**

**for (int i = 0; i < n ; i++){**

**free(dynamic\_array[i]);**

**}**

**free(dynamic\_array);**

**}**