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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текстовых данных

Студент гр. 1304 _____ Лобанов Е.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Лобанов Е.А.

Группа 1304

Тема работы: обработка текстовых данных

Исходные данные:
Текст произвольной длины, разделённый по предложениям точкой.

Содержание пояснительной записки:
«Содержание», «Введение», «Разработка кода», «Техническое задание», «Выделение подзадач», «Ввод текста», «Функции обработки текста», «Функция main()», «Тестирование», «Инструкция», «Заключение», «Список использованных источников», «Приложение А. Исходный код программы».

Предполагаемый объем пояснительной записки:
Не менее 12 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 21.12.2021

Дата защиты реферата: 23.12.2021

Студент _____ Лобанов Е.А.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

Разработана программа, осуществляющая считывание текста – набора латинских букв и цифр и сохраняющая данный текст как набор предложений в динамическую память. Далее программа удаляет все появляющиеся повторно предложения и, с указания пользователя, выполняет одно из четырёх действий, с поддержкой последовательного выполнения этих функций и выхода из программы. При написании программы были использованы стандартные библиотеки языка Си.

SUMMARY

The program was developed that reads text – a set of Latin letters and numbers and saves it as set of sentences into dynamic memory. Then the program deletes all repeatable sentences and proceeds to execute one of the four actions by the choice of a user with support of sequential execution and exit from program. C standard library functions were used in development.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью этой работы является изучение стандартных функций языка Си для работы с текстом и создание программы, обрабатывающей введённый текст согласно первичной обработке и командам пользователя.

Для достижения этой цели требуется решить следующие задачи:

1. Изучение функций стандартной библиотеки для работы с текстом
2. Изучение работы динамической памяти и способов взаимодействий с ней для хранения текста
3. Разработка кода программы
4. Компиляция программы
5. Отладка и тестирование программы
6. Разработка пользовательской инструкции

1. РАЗРАБОТКА КОДА

1.1. Техническое задание

Вариант 4

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. Найти во всем тексте даты записанные в подстроке вида “d<day>m<month>y<year>” и вывести все даты по возрастанию в формате “DD:MM:YYYY:. Пример даты в тексте, “d14m03y0988”.

2. Удалить все предложения, в которых количество слов нечетно.

3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.

4. Вывести все предложения в которых нет заглавных букв.

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.

1.2. Выделение подзадач

Таким образом, мы можем выделить следующие подзадачи:

- Реализация функции считывания и разбиения текста на отдельные предложения и запись их в динамические массивы
- Удаление повторяющихся предложений
- Вывод подсказки для пользователя и ожидание ввода команды внутри оператора switch
- Реализация действия «1» по поиску дат в тексте по определённому формату и вывод этих дат в порядке возрастания
- Реализация подфункции по сравнению дат по возрастанию для передачи к функции qsort()
- Реализация действия «2» по удалению предложений с нечетным количеством слов
- Реализация действия «3» по преобразованию всех слов, в которых нет цифр так, чтобы все буквы кроме последней были прописными
- Реализация действия «4» по выводу всех предложений, в которых нет заглавных букв
- Реализация функции вывода текста
- Реализация освобождения всей занятой динамически выделенной памяти во избежание утечки памяти

1.3. Ввод текста

Ввод текста осуществляется с помощью функции read_text(), принимающей на вход ссылку на выделенный динамически массив. Затем, функция изменяет размер ранее выделенной памяти и, если выделить память не удалось, выводит сообщение об ошибке и завершает выполнение программы. Далее, программа выделяет память для массива для нового предложения внутри первого массива, считывает символ, и, если он не равен символу перевода строки, записывает его в массив. Если вводится символ точки, то программа завершает ввод предложения, переходит на обработку нового предложение и начинает цикл заново. Если набирается два символа перевода строки подряд, то функция завершает ввод текста.

1.4. Функции обработки текста

Функция format_text() получает на вход ссылку на массив предложений находит в введённом тексте предложения, которые встречаются два или более раз, и удаляет все дубликаты оригинального предложения. Сравнение производится посимвольно, без учёта регистра. Удаление предложения производится освобождением выделенной для данного предложения памяти и смещением всех последующих предложений. Функция возвращает количество предложений в тексте.

Действие «1» производится функцией print_dates(). Функция получает на вход ссылку на массив предложений и количество предложений. Затем для каждого предложения он считает его длину и осуществляет проход по этому же самому предложению в поиске строки формата d<DD>m<MM>y<YYYY>. Если такая строка нашлась, то он выделяет динамическую память для отдельного хранения даты, месяца и года. Далее функция осуществляет сортировку qsort() с использованием побочной функции cmp(), которая сортирует эти даты в порядке возрастания, и выводит эти даты в формате DD:MM:YYYY. Если дат в тексте не нашлось, то выводится строка, оповещающая об этом. Использованная память для хранения этих дат очищается после выполнения программы.

Действие «2» производится функцией odd_count(). Функция получает на вход ссылку на массив предложений и количество предложений. Затем функция производит подсчёт количества пробелов в каждом предложении, и, если это количество чётно, то оно удаляет это предложение по способу, аналогичному в функции format_text(). Функция возвращает количество предложений после преобразования.

Действие «3» производится функцией upper_case(). Функция получает на вход ссылку на массив предложений и количество предложений. Затем функция производит проход по каждому слову в предложении. Если в данном слове не было обнаружено цифр, то функция переводит все буквы в верхний регистр, а затем опускает последнюю букву с конца. Функция возвращает количество предложений после преобразования.

Действие «4» производится функцией lower_case(). Функция получает на вход ссылку на массив предложений и количество предложений. Затем функция производит проход по предложению, и, если в предложении нет заглавных букв, выводит это предложение. Если не нашлось ни одного предложения с заглавными буквами, то выводится строка, оповещающая об этом.

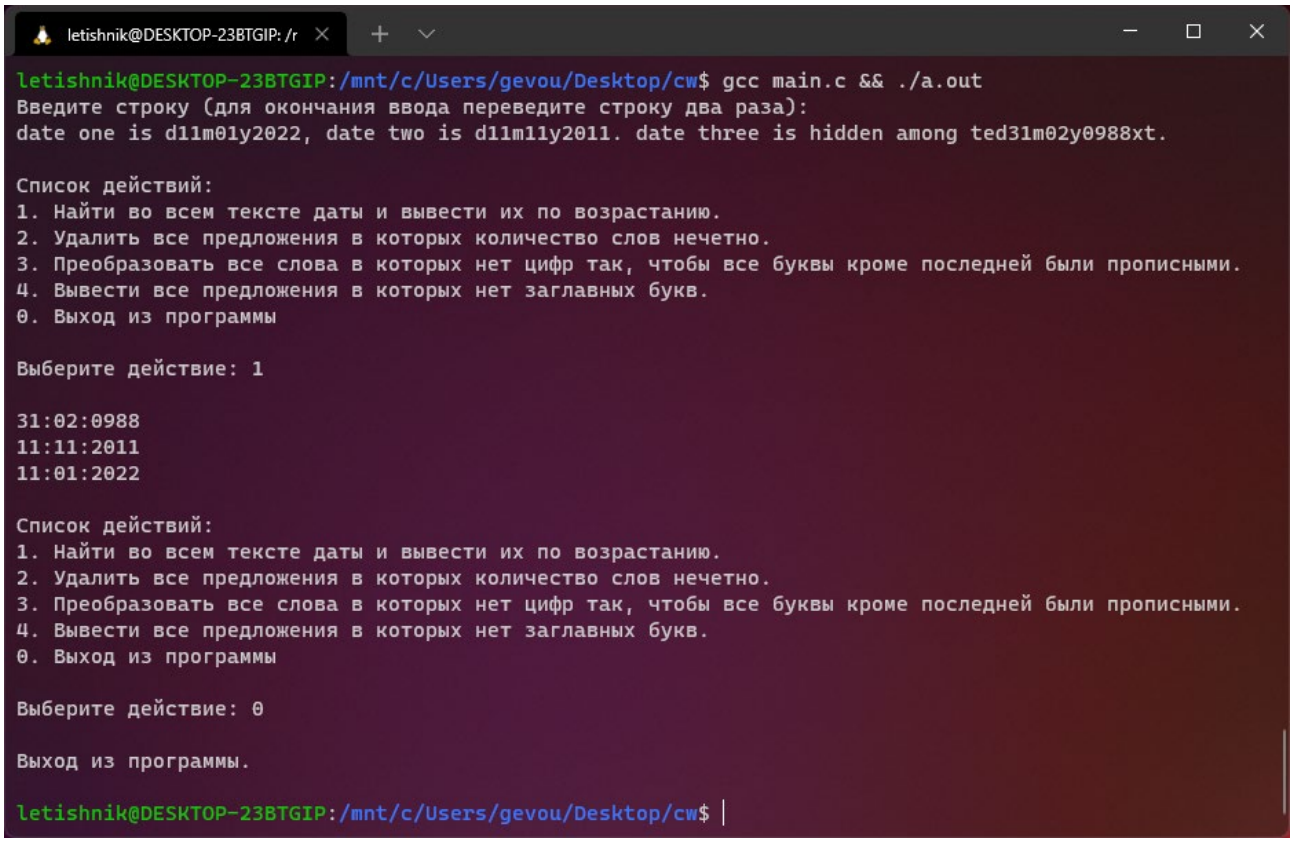
Функция print_text() выводит текст по каждому предложению отдельно.

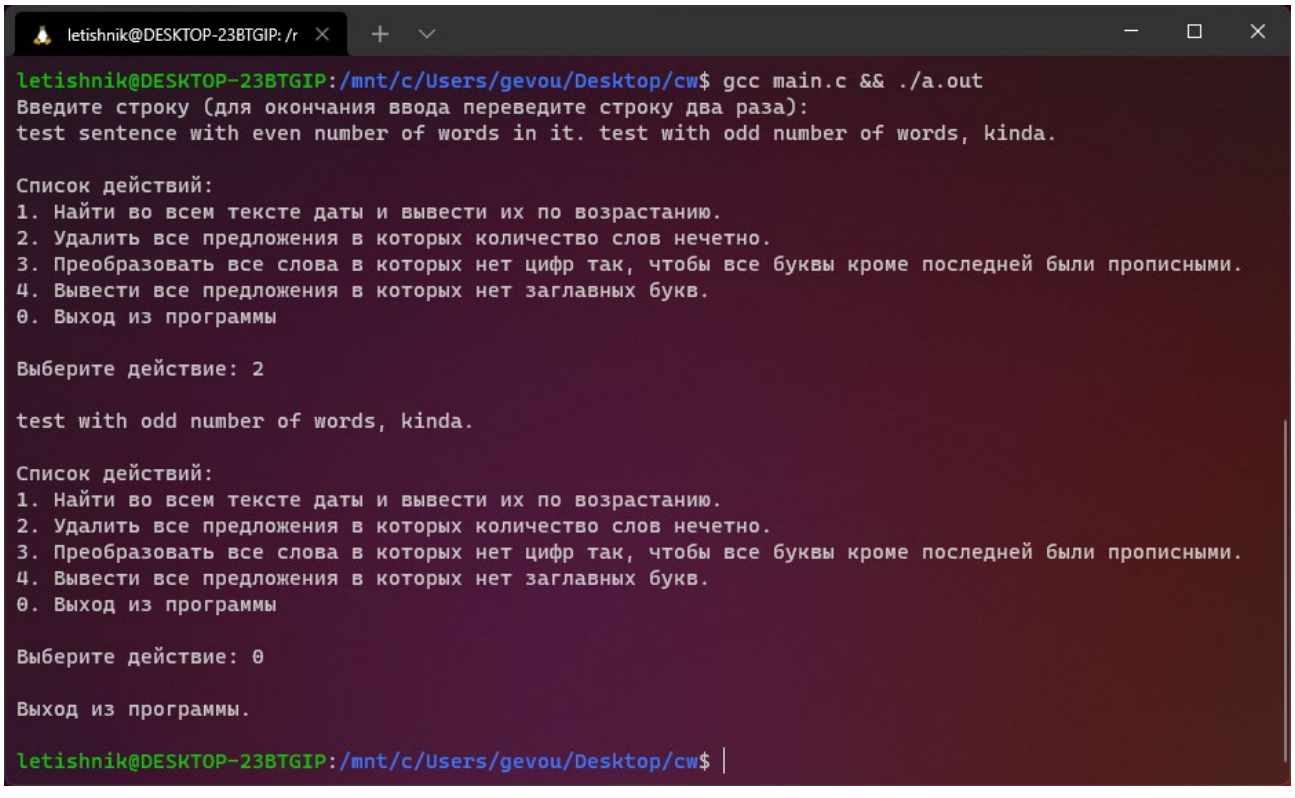
Функция free_text() освобождает всю выделенную динамически под текст память.

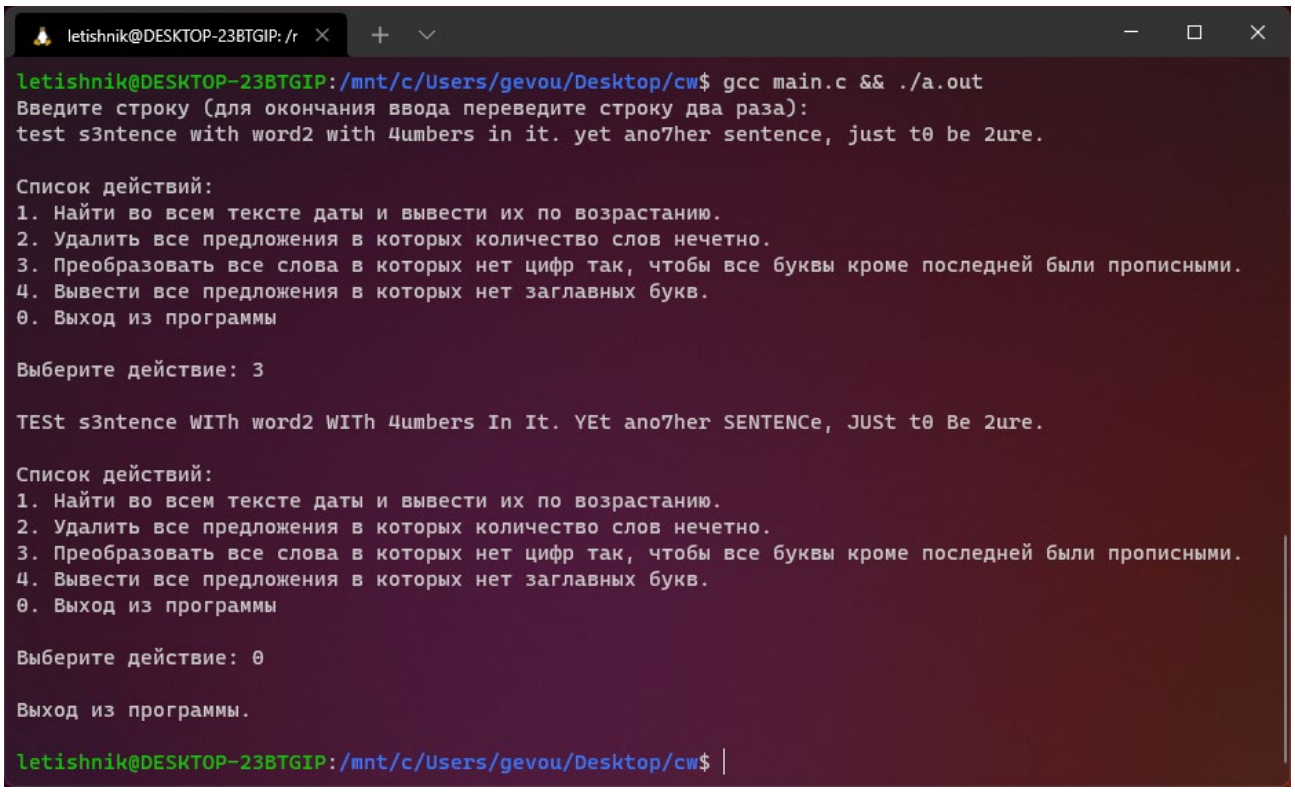
1.5. Функция main()

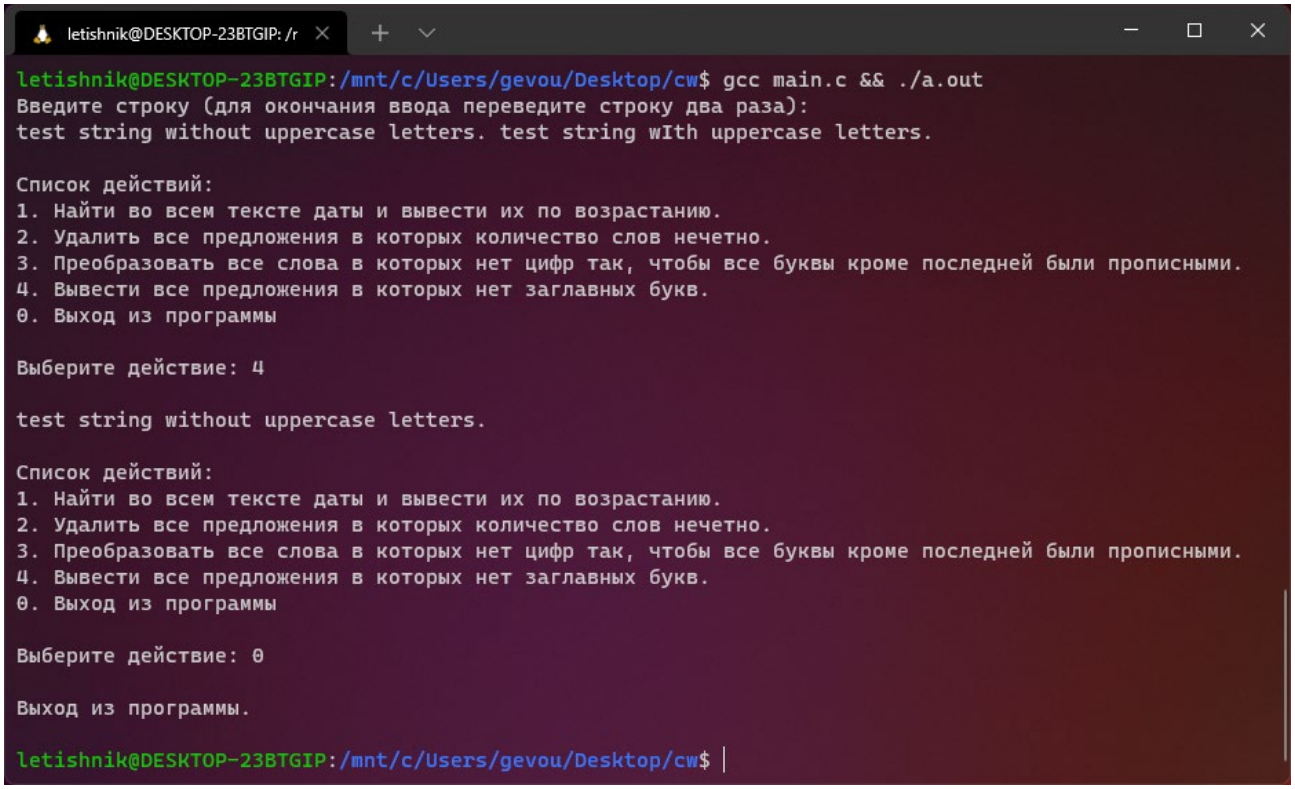
Функция main() является основной функцией программы, которая выделяет память под текст, осуществляет вызов функции по считыванию и форматированию строки, а также считывает выбор пользователя и вызов оператора switch, с переходом к соответствующей функции. В конце, программа вызывает функцию по освобождению выделенной памяти и завершает выполнение программы.

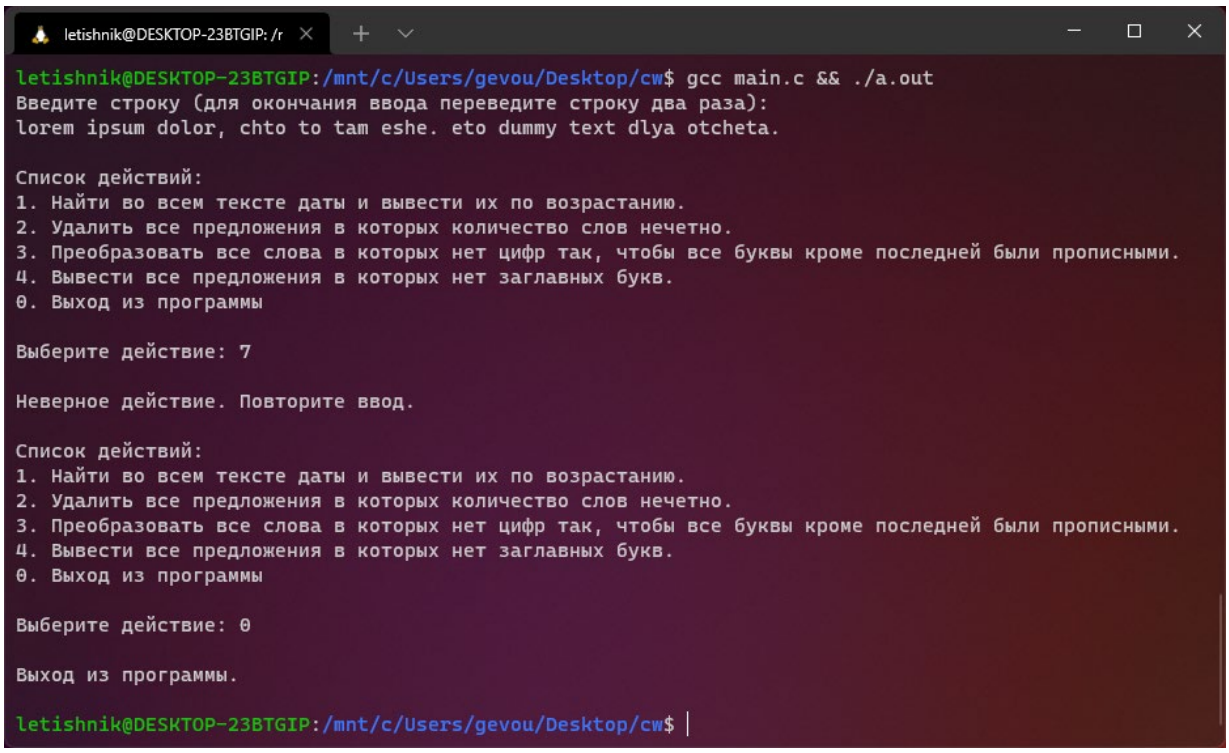
2. ТЕСТИРОВАНИЕ






letishnik@DESKTOP-23BTGIP: /r
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$ gcc main.c && ./a.out
Введите строку (для окончания ввода переведите строку два раза):
test s3ntence with word2 with 4umbers in it. yet ano7her sentence, just t0 be 2ure.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 3
TESt s3ntence WITh word2 WITh 4umbers In It. YEt ano7her SENTENCe, JUSt t0 Be 2ure.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 0
Выход из программы.
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$


letishnik@DESKTOP-23BTGIP: /r
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$ gcc main.c && ./a.out
Введите строку (для окончания ввода переведите строку два раза):
test string without uppercase letters. test string wIth uppercase letters.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 4
test string without uppercase letters.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 0
Выход из программы.
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$


letishnik@DESKTOP-23BTGIP: /r
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$ gcc main.c && ./a.out
Введите строку (для окончания ввода переведите строку два раза):
lorem ipsum dolor, chto to tam eshe. eto dummy text dlya otcheta.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 7
Неверное действие. Повторите ввод.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 0
Выход из программы.
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$


letishnik@DESKTOP-23BTGIP: /r
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$ gcc main.c && ./a.out
Введите строку (для окончания ввода переведите строку два раза):
Test sentence with uppercase 1etters and number. test string without uppercase 1etters and num9er.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 3
TESt SENTENCe WITh UPPERCASe 1etters ANd NUMBEr. TESt STRINg WITHOUt UPPERCASe 1etters ANd num9er.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 2
TESt STRINg WITHOUt UPPERCASe 1etters ANd num9er.
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 4
В тексте нет предложений, в которых нет заглавных букв!
Список действий:
1. Найти во всем тексте даты и вывести их по возрастанию.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.
0. Выход из программы
Выберите действие: 0
Выход из программы.
letishnik@DESKTOP-23BTGIP:/mnt/c/Users/gevou/Desktop/cw$

3. ИНСТРУКЦИЯ

1. Запустить терминал из папки с исходным файлом и выполнить команду gcc main.c для компиляции программы

2. Запустить получившийся файл с помощью команды ./a.out

3. Ввести текст, и завершить ввод текста двумя переводами строки

4. Ввести номер желаемого действия и нажать Enter

5. Возможен ввод нового действия или выход из программы посредством ввода команды «0»

ЗАКЛЮЧЕНИЕ

Изучены новые средства работы с текстом на языке Си. Разработана программа согласно техническому заданию, которая производит работу, форматирование и вывод текста.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.The C Programming Language / Brian W. Kernigan, Dennis M. Ritchie Second edition, 1988. 288 с.

2. The C++ Resources network [Электронный ресурс]
URL: http://cplusplus.com

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

int read_text(char*** text);
int format_text(int sentences, char*** text);
int cmp(const void* p1, const void* p2);
void print_dates(int sentences, char*** text);
int odd_count(int sentences, char*** text);
int upper_case(int sentence, char*** text);
void lower_case(int sentence, char*** text);
void print_text(int sentence, char*** text);
void free_text(int sentence, char*** text);

int main(){
    printf("Введите строку (для окончания ввода переведите строку два
раза):\n");
    char** text = malloc(sizeof(char*));
    int choice, end = 0;
    int text_len = read_text(&text);
    text_len = format_text(text_len, &text);
    while (end == 0){
        printf("Список действий:\n"
                "1. Найти во всем тексте даты и вывести их по
возрастанию.\n"
                "2. Удалить все предложения в которых количество слов
нечетно.\n"
                "3. Преобразовать все слова в которых нет цифр так, чтобы
все буквы кроме последней были прописными.\n"
                "4. Вывести все предложения в которых нет заглавных
букв.\n"
                "0. Выход из программы\n\n"
                "Выберите действие: ");
        scanf("%d", &choice);
        printf("\n");
```

```
        switch (choice){
            case 1:
                print_dates(text_len, &text);
                break;
            case 2:
                text_len = odd_count(text_len, &text);
                print_text(text_len, &text);
                break;
            case 3:
                text_len = upper_case(text_len, &text);
                print_text(text_len, &text);
                break;
            case 4:
                lower_case(text_len, &text);
                break;
            case 0:
                printf("Выход из программы.\n");
                end = 1;
                break;
            default:
                printf("Неверное действие. Повторите ввод.\n");
                break;
        }
        printf("\n");
    }
    free_text(text_len, &text);
    return 0;
}

int read_text(char*** text){
    int new_lines_count = 0, passed = 1, sentences = 0, count = 0,
str_length = 1000;
    char c;
    while (1){
        count = 0;
        (*text) = realloc((*text), sizeof(char*) * (sentences+1));
        if ((*text) == NULL){
            puts("Недостаточно памяти!");
```

```
            exit(0);
        }
        (*text)[sentences] = malloc(sizeof(char) * str_length);
        c = getchar();
        if (c == '\n'){
            new_lines_count++;
            if (new_lines_count==2)
                break;
        }
        else{
            (*text)[sentences][count] = c;
            new_lines_count=0;
        }
        do{
            count++;
            c = getchar();
            if (c == '\n'){
                new_lines_count++;
                break;
            }
            else
                new_lines_count = 0;
            (*text)[sentences][count] = c;
            passed = (strchr(".", c) == NULL);
            if (count >= str_length - 2){
                str_length = str_length + 500;
                (*text)[sentences] = (char*) realloc((*text)[sentences],
str_length);
                if ((*text)[sentences] == NULL){
                        puts("Недостаточно памяти!");
                        exit(0);
                }
            }
        } while(passed);
        if (new_lines_count == 2)
            break;
        c=getchar();
        if (c == '\n')
```

```
                new_lines_count++;
        else
                new_lines_count = 0;
        (*text)[sentences][count+1] = ' ';
        (*text)[sentences][count+2] = '\0';
        sentences++;
    }
    return sentences;
}

int format_text(int sentences, char*** text){
    for(int i = 0; i < sentences; i++){
            int j = i + 1;
            while (j < sentences){
                if (strcasecmp((*text)[i], (*text)[j]) == 0){
                    free((*text)[j]);
                    sentences--;
                    for (int k = j; k <= sentences; k++){
                        (*text)[k] = (*text)[k + 1];
                    }
                }
                else
                    j++;
            }
    }
    return sentences;
}

int cmp(const void* p1, const void* p2){
    const int *a = *(const int**)p1;
    const int *b = *(const int**)p2;
    if (a[2] == b[2]){
        if (a[1] == b[1])
            return a[0] - b[0];
        else
            return a[1] - b[1];
    }
    else
```

```
        return a[2] - b[2];
}

void print_dates(int sentences, char*** text){    // task 1
    int sentence_len, items = -1, j, flag = 0;
    int flag1 = 0, flag2 = 0, flag3 = 0, flag4 = 0, flag5 = 0;
    int** dates = malloc(sizeof(int*));
    char temp[11];
    for (int i = 0; i < sentences; i++){
        sentence_len = 0;
        for (int j = 0; (*text)[i][j]; j++)
            sentence_len++;
        for (j = 0; j <= sentence_len-10; j++){
            if ((*text)[i][j] == 'd' && (*text)[i][j+3] == 'm' &&
(*text)[i][j+6] == 'y' &&
                isdigit((*text)[i][j+1]) && isdigit((*text)[i][j+2]) &&
                isdigit((*text)[i][j+4]) && isdigit((*text)[i][j+5]) &&
                isdigit((*text)[i][j+7]) && isdigit((*text)[i][j+8]) &&
isdigit((*text)[i][j+9]) && isdigit((*text)[i][j+10])){
                    flag = 1;
                    items++;
                    dates = realloc(dates, sizeof(int*) * (items+1));
                    if (dates == NULL){
                        puts("Недостаточно памяти!");
                        exit(0);
                    }
                    for (int g = 0; g < 11; g++)
                        temp[g] = (*text)[i][j+g];
                    dates[items] = calloc(3, sizeof(int));
                    sscanf(temp, "d%dm%dy%d", &dates[items][0],
&dates[items][1], &dates[items][2]);
            }
        }
    }
    if (flag){
        qsort(dates, items+1, sizeof(int*), cmp);
        for (int i = 0; i <= items; i++){
            if (dates[i][0] < 10)
```

```
                flag1 = 1;
            if (dates[i][1] < 10)
                flag2 = 1;
            if (dates[i][2] < 1000)
                flag3 = 1;
            if (dates[i][2] < 100)
                flag4 = 1;
            if (dates[i][2] < 10)
                flag5 = 1;
            printf("%c%d:%c%d:%c%c%c%d\n", '0'*flag1, dates[i][0],
'0'*flag2, dates[i][1], '0'*flag3, '0'*flag4, '0'*flag5, dates[i][2]);
            flag1 = 0; flag2 = 0; flag3 = 0; flag4 = 0; flag5 = 0;
        }
        for(int i = 0; i < items; i++)
            free((dates)[i]);
    }
    else
        puts("В тексте не найдено дат!");
    free(dates);
}

int odd_count(int sentences, char*** text){       // task 2
    int count;
    for (int i = 0; i < sentences; i++){
        count = 0;
        for (int j = 0; (*text)[i][j]; j++){
            if ((*text)[i][j] == ' ')
                count++;
        }
        if (count % 2){
            free((*text)[i]);
            sentences--;
            for (int k = i; k <= sentences; k++){
                (*text)[k] = (*text)[k + 1];
            }
        }
    }
    return sentences;
```

```
}

int upper_case(int sentence, char*** text){     // task 3
    int word_len, num_count;
    char c;
    for (int i = 0; i < sentence; i++){
        word_len = 0;
        num_count = 0;
        for (int j = 0; (*text)[i][j]; j++){
            if ((*text)[i][j] != ' ')
                word_len++;
                if (isdigit((*text)[i][j])){
                    num_count++;
                }
            if ((*text)[i][j] == ' '){
                if (num_count == 0){
                    for (int k = j-word_len; k < j; k++){
                            (*text)[i][k] = toupper((*text)[i][k]);
                    }
                    for (int k = j - 1; k >= j-word_len; k--){
                        if (isalpha((*text)[i][k])){
                            (*text)[i][k] = tolower((*text)[i][k]);
                            break;
                        }
                    }
                }
                word_len = 0;
                num_count = 0;
            }
        }
    }
    return sentence;
}

void lower_case(int sentence, char*** text){     // task 4
    int failed, flag = 0;
    for (int i = 0; i < sentence; i++){
        failed = 0;
```

```
        for (int j = 0; (*text)[i][j]; j++){
            if (isupper((*text)[i][j]))
                failed = 1;
        }
        if (failed == 0){
            printf("%s", (*text)[i]);
        } else
            flag++;
    }
    if (flag == sentence)
        printf("В тексте нет предложений, в которых нет заглавных
букв!");
    printf("\n");
}

void print_text(int sentence, char*** text){
    for (int i = 0; i < sentence; i++)
        printf("%s", (*text)[i]);
    printf("\n");
}

void free_text(int sentence, char*** text){
    for(int i = 0; i < sentence; i++)
           free((*text)[i]);
     free(*text);
}
```