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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текста на языке Си

Студент гр. 3384 _____ Козьмин Н.В.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2023

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Козьмин Н.В.

Группа 3384

Тема работы: Обработка текста на языке Си

Исходные данные:
Вывод программы должен быть произведен в стандартный поток вывода: *stdout*. Ввод данных в программе в стандартный поток ввода: *stdin*. В случае использования *Makefile* название исполняемого файла должно быть: *cw*.
После вывода информацию о варианте курсовой работе программа ожидает ввода пользователем числа – номера команды:

- 0 - вывод текста после первичной обязательной обработки.
- 1 - вывод всех предложений, в которых есть слово “*banana*” или “*apple*”. Слово “*banana*” должно быть выделено желтым цветом, а слово “*apple*” зеленым.
- 2 - преобразование предложений так, чтобы в слове все буквы были строчные, а все цифры были заменены буквой ‘D’.
- 3 - удаление всех предложений, длина которых больше 15.
- 4 - сортировка предложений по возрастанию произведений цифр в них.
- 5 - вывод справки о функциях, которые реализует программа.

В случае вызова справки (опция 5) текст на вход подаваться не должен, во всех остальных случаях после выбора опции должен быть считан текст.

Признаком конца текста считается два подряд идущих символа переноса строки ‘\n’. После каждой из функций нужно вывести результат работы программы и завершить программу.

В случае ошибки и невозможности выполнить функцию по какой-либо причине, нужно вывести строку:

```
Error: <причина ошибки>
```

Каждое предложение должно выводиться в отдельной строке, пустых строк быть не должно. 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).

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.

Все символы пробелов и подобных им (например, символы табуляции) в начале предложения должны быть удалены.

Все остальные символы должны остаться в сохранности. Например, если в середине предложения между словами стоит табуляция, то и при выводе там тоже должна стоять табуляция.

Содержание пояснительной записки:
«Содержание», «Введение», «Файл с главной функцией», «Встроенные файлы», «Мейкфайл», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 20 страниц.

Дата выдачи задания: 16.11.2023

Дата сдачи реферата: 20.12.2023

Дата защиты реферата: 22.12.2023

Студент гр. 3384 _____ Козьмин Н.В.

Преподаватель _____ Глазунов С.А.

АННОТАЦИЯ

В проделанной работе описана программа, реализованная на языке Си, которая предназначена для работы с текстом. Функционал программы заключается в том, что после ввода команды от 0 до 4 и текста или ввода команды 5 выводится соответствующий результат (описанный в задании). При вводе некорректных данных выводится ошибка, которая объясняет проблему с тем, что получено. Для разработки использовались методы, изученные в течении семестра. После написания кода программы, он был проверен. Тесты и их результаты см. в приложении Б. Разработанный программный код см. в приложении А. Скриншоты с успешными запусками прилагаются.

SUMMARY

The work done describes a program implemented in the C language, which is designed to work with text. The functionality of the program is that after entering a command from 0 to 4 and text or entering command 5, the corresponding result (described in the task) is displayed. If you enter incorrect data, an error is displayed that explains the problem with what was received. Methods learned during the semester were used for development. After writing the program code, it was tested. Tests and their results can be found in Appendix B. The developed program code can be found in Appendix A. Screenshots of successful runs are attached.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является разработка программы на языке Си, обрабатывающую текст по определённым правилам. Для достижения поставленной цели требуется решить следующие задачи:

1. Создать файл с главной функцией *main*, в которой через *switch-case* будут выполнятся нужные функции.
2. Реализовать *Makefile* для сборки и тестирования программы с постепенным его обновлением.
3. Разработать функции:
    a. Для получения команды.
    b. Для чтения текста.
    c. Для первичной обязательной обработки.
    d. Для вывода текста.
    e. Для освобождения памяти из-под текста.
    f. Для выполнения команд, требуемых по условию задания.

1. ФАЙЛ С ГЛАВНОЙ ФУНКЦИЕЙ

После включения необходимых заголовочных файлов. Первым делом, выводится информация о варианте и авторе. Затем объявляется указатель *sentences* на массив указателей для типа *char.* Также объявляется *cnt_sentences* с типом *size_t*, так как этот тип используется для больших положительных целых чисел. Далее *switch* проверяет вывод функции, получающей команду, getcommand, которую предстоит реализовать, как и последующие.

В кейсах для всех команд, кроме пятой, сначала выполняется функция readtext, читающая текст и принимающая ссылку на указатель. Возвращает она количество предложений, которое записывается в *cnt_sentences*. Такой способ выбран, вместо передачи ссылки на *cnt_sentences*, так как похожим образом работает *scanf.*

Если текст не считан, значит *sentences* имеет значение *NULL*, что приводит условная конструкция к выходу из программы. Иначе выполнение программы продолжается. Выполняется функция *preprocessing*, которая принимает ссылку на предложения и их количество. После чего выполняются отдельные функции для разных кейсов. Завершаются команды 0-4 тем, что освобождают пямять функцией *freetext(&sentences, &cnt_sentences)*.

Команда 0 выполняет функцию *printtext(&sentences, &cnt_sentences),* печатающую текст. Команда 1 выполняет *bananapplesearch(&sentences, &cnt_sentences)* с встроенным выводом предложений, так как следует сделать выборку, а не печатать все предложения. Команда 2 выполняет *convertnumsandalpha(&sentences, &cnt_sentences)* и вызывает *printtext*. Команда 3 запускает *siftedoutput(&sentences, &cnt_sentences)* и выводит предложения по аналогии с командой 1. Команда 4 вызывает функции *sorttextnums(&sentences, &cnt_sentences)* и *printtext.* Команда 5 выполняет проинициализированную функцию *printhelp()*. Если задана другая команда (через ‘\n’) – выводится ошибка.

2. ВСТРОЕННЫЕ ФАЙЛЫ

2.1. Получение команды

Функция getcommand реализована с помощью получения двух символов, первого – номера команды, так как все они по условию задания однозначные, и второго – символа перевода строки. Однако требуется, чтобы команда была числом, поэтому из её символа вычитается символ равный нулю, что выполняет поставленную задачу.

2.2. Считывание текста

Как уже ранее было сказано, функция readtext принимает указатель на указатель с предложениями. Сначала он обнуляется, потом инициализируются другие переменные. Из них стоит обратить внимание на *cnt_line_breaks* = 1: оно считает текущее количество переводов строк, засчитывая уже введённое.

Далее циклично считывается символ в *cur_char*. И в первую очередь идёт проверка на количество переносов строк. Если оно обновляется на 2, то происходит при наличии предложения его записывание (которое будет описано позже) и добавление точки, а потом выход из цикла. Если же *cur_char != ‘\n’*, то счётчик, подряд идущих, разделителей обнуляется.

Затем идёт нативная проверка на неожиданные символы, а после проверка флага *issentence*, который хранит информацию о том “находимся ли мы сейчас в предложении”.

Во-первых, если *issentence == 0*, то следует пропустить пробельные символы и точки (пустые предложения) с текущей итерацией.

Во-вторых, если *(issentence && cur_char == '.'),* то следует записать предложение. Перевыделить память для него, записать последний символ (‘.’) и добавить ‘\0’, обнулить нужные счётчики и пропустить дальнейшие шаги.

В-третьих, если программа дошла до этого этапа, остаётся записать символ в текущее предложение. (Если *!issentence,* то нужно выделить память и определить в ноль указатель на новое предложение). Перевыделяем память для текущего предложения и записываем символ в новую “ячейку”.

2.3. Освобождение памяти

Функция *freetext* также, как и функции команд 1-4, получает ссылки на текст и на количество предложений в нём. После чего функция рекурсивно очищает память, начиная с выделенной под предложения, и заканчивая памятью, выделенной под хранение массива указателей на предложения.

2.4. Поиск бананов и яблок

Функция *bananapplesearch* начинается с того, что объявляются переменные *isnewword* и *isprint*, хранящие информацию о том “находимся ли мы сейчас в новом слове” и “нужно ли печатать текущее предложение” соответственно. Выполнение осуществляется проходом в циклах по символам предложений и реализацией следующего алгоритма: Если *issentence* истинно, символы начиная с текущего равны “*banana*”, и последующий символ равен запятой, точке, либо пробелу, то печатаются предыдущие символы, если нужно, и “*banana*” с выделением необходимым цветом, а также обновляются флаги. Аналогичный алгоритм проделывается для “*apple*”. Иначе просто обновляем *isnewword* и печатаем текущий символ, если нужно.

2.5. Перевод букв в нижний регистр и замена цифр

Функция *convertnumsandalpha* проходит по символам предложений и меняет по условию задания символы с помощью функций встроенных в стандартную библиотеку.

2.6. Удаление больших предложений

Функция *siftedoutput* печатает предложения, длина которых меньше или равна 15.

2.7. Сортировка предложений

Функция sorttextnums использует встроенную быструю сортировку (*qsort*) и функцию сравнения предложений *cmpsntncs,* принимающую ссылки на два объекта, и приводящую их к ссылкам на предложения. Предложения она сравнивает следующим образом: Находится сумма цифр первого предложения, находится сумма цифр второго предложения и из первой вычитается вторая. Так как, если первая больше второй результат будет положительный, если равны – нулевой и если первая меньше второй – отрицательный, требования к функции для сравнения в *qsort* будут выполнятся.

3. МЕЙКФАЙЛ

Первая цель *all* содержит только зависимость *cw*, так как это позволяет не производить линковку без необходимости (работает только на Linux, так как на Windows нужна зависимость cw.exe). В цели *cw* указываются все объектные файлы и производится из них сборка исполняемого файла. Затем для каждого объектного файла прописывается своя цель с зависимыми исходными и заголовочными файлами. Дополнительно прописывается цель clear (для Linux), чтобы удалить объектные файлы командой make clear.

ЗАКЛЮЧЕНИЕ

В проделанной работе была разработана программа на языке Си, обрабатывающая текст по всем требуемым правилам. Все поставленные задачи были выполнены, а также был проведён глубокий анализ написанного кода. Эти действия подвели итог программированию на языке Си в первом семестре и помогли закрепить полученные знания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стандартная библиотека языка Си. URL: https://ru.wikipedia.org/wiki/Стандартная_библиотека_языка_Си (дата обращения: 20.12.2023).
2. Основы программирования на языках Си и C++ для начинающих. URL: http://cppstudio.com/ (дата обращения: 20.12.2023)
3. StackOwerflow URL: https://stackoverflow.com/questions/ (дата обращения: 20.12.2023)
4. Оформление README.md: https://gist.github.com/alinastorm/8a04cdbc36be9c051a66f90ae6d6df35 (дата обращения 20.12.2023)

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```
#include <stdio.h>
#include "getcommand.h"
#include "readtext.h"
#include "preprocessing.h"
#include "printtext.h"
#include "freetext.h"
#include "bananapplesearch.h"
#include "convertnumsandalpha.h"
#include "siftedoutput.h"
#include "sorttextnums.h"


void printhelp(){
    puts(
        "Help about the functions that the program implements:\n"
        "0 - primary mandatory processing (deleting duplicate sentences
without taking into account case)\n"
        "1 - all sentences containing the words \033[33mbanana\033[0m
or \033[32mapple\033[0m\n"
        "2 - converting all letters to lowercase and replacing all
numbers with the letter 'D'\n"
        "3 - deleting all sentences with a length greater than 15\n"
        "4 - sorting sentences in ascending order of the products of all
numbers in them"
    );
}


int main(){
    printf("Course work for option 4.12, created by Nikita Kozmin\n");
    char **sentences;
    size_t cnt_sentences;
    switch (getcommand()){
    case 0:
        cnt_sentences = readtext(&sentences);
        if (!sentences) return 0;
        preprocessing(&sentences, &cnt_sentences);
        printtext(&sentences, &cnt_sentences);
        freetext(&sentences, &cnt_sentences);
        return 0;
    case 1:
        cnt_sentences = readtext(&sentences);
        if (!sentences) return 0;
        preprocessing(&sentences, &cnt_sentences);
        bananapplesearch(&sentences, &cnt_sentences);
        freetext(&sentences, &cnt_sentences);
        return 0;
    case 2:
        cnt_sentences = readtext(&sentences);
        if (!sentences) return 0;
```

```
        preprocessing(&sentences, &cnt_sentences);
        convertnumsandalpha(&sentences, &cnt_sentences);
        printtext(&sentences, &cnt_sentences);
        freetext(&sentences, &cnt_sentences);
        return 0;
    case 3:
        cnt_sentences = readtext(&sentences);
        if (!sentences) return 0;
        preprocessing(&sentences, &cnt_sentences);
        siftedoutput(&sentences, &cnt_sentences);
        freetext(&sentences, &cnt_sentences);
        return 0;
    case 4:
        cnt_sentences = readtext(&sentences);
        if (!sentences) return 0;
        preprocessing(&sentences, &cnt_sentences);
        sorttextnums(&sentences, &cnt_sentences);
        printtext(&sentences, &cnt_sentences);
        freetext(&sentences, &cnt_sentences);
        return 0;
    case 5:
        printhelp();
        return 0;
    default:
        printf("Error: Unexpected command\n");
        return 0;
    }
    return 0;
}
```

Название файла: getcommand.h

```
#ifndef getcommand
int getcommand();
#endif
```

Название файла: getcommand.c

```
#ifndef stdio
#include <stdio.h>
#endif

#include "getcommand.h"


int getcommand(){
    int command = getchar();
    if (getchar() != '\n') return -1;
    return command - '0';
}
```

Название файла: readtext.h

```
#ifndef readtext
size_t readtext(char ***sentences);
#endif
```

Название файла: readtext.c

```
#ifndef stdio
#include <stdio.h>
#endif
#ifndef stdlib
#include <stdlib.h>
#endif
#ifndef ctype
#include <ctype.h>
#endif

#include "readtext.h"


size_t readtext(char ***sentences){
    *sentences = NULL;
    size_t cnt_sentences = 0;
    size_t cnt_chars = 0;
    short cnt_line_breaks = 1;
    short issentence = 0;
    char cur_char;

    while (1){
        cur_char = getchar();

        // Проверка на переносы строки
        if (cur_char == '\n'){
            cnt_line_breaks += 1;
            if (cnt_line_breaks == 2){
                if (issentence){
                    (*sentences)[cnt_sentences-1]                    =
realloc((*sentences)[cnt_sentences-1], (cnt_chars+1)*sizeof(char));
                    (*sentences)[cnt_sentences-1][cnt_chars-1] = '.';
                    (*sentences)[cnt_sentences-1][cnt_chars] = '\0';
                    issentence = 0;
                    cnt_chars = 0;
                }
                break;
            }
        }
        else{
            cnt_line_breaks = 0;
        }

        // Проверка на неожиданные символы
        if (!isalnum(cur_char) && cur_char != '.' && cur_char != ','
&& !isspace(cur_char)){
            printf("Error: Unexpected symbol\n");
            *sentences = NULL;
            return 0;
        }

        // Удаление первых пробельных символов и пустых предложений
        if (!issentence && (isspace(cur_char) || cur_char == '.'))
continue;

        // Запись предложения
```

```
        if (issentence && cur_char == '.'){
            cnt_chars++;
            (*sentences)[cnt_sentences-1]                                    =
realloc((*sentences)[cnt_sentences-1], (cnt_chars+1)*sizeof(char));
            (*sentences)[cnt_sentences-1][cnt_chars-1] = cur_char;
            (*sentences)[cnt_sentences-1][cnt_chars] = '\0';
            issentence = 0;
            cnt_chars = 0;
            continue;
        }

        // Запись символа
        if (!issentence){
            cnt_sentences++;
            (*sentences)            =            realloc((*sentences),
cnt_sentences*sizeof(char*));
            (*sentences)[cnt_sentences-1] = NULL;
            issentence = 1;
        }
        cnt_chars++;
        (*sentences)[cnt_sentences-1]                                        =
realloc((*sentences)[cnt_sentences-1], (cnt_chars+1)*sizeof(char));
        (*sentences)[cnt_sentences-1][cnt_chars-1] = cur_char;
    }

    if (!*sentences) printf("Error: Text is missing\n");
    return cnt_sentences;
}
```

Название файла: preprocessing.h

```
#ifndef preprocessing
size_t preprocessing(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: preprocessing.c

```
#ifndef stdlib
#include <stdlib.h>
#endif
#ifndef string
#include <string.h>
#endif
#ifndef ctype
#include <ctype.h>
#endif

#include "preprocessing.h"


size_t preprocessing(char ***sentences, size_t *cnt_sentences){
    char *fst = NULL;  // Первое предложение
    char *scd = NULL;  // Второе предложение
    size_t len_fst;
    size_t len_scd;

    // Проход по первым предложениям
    for (size_t i = 0; i < (*cnt_sentences)-1; i++){
```

```
        len_fst = strlen((*sentences)[i]);
        fst = realloc(fst, sizeof(char)*(len_fst + 1));
        for (size_t k=0; k < (len_fst+1); k++){
            fst[k] = tolower((*sentences)[i][k]);
        }

        // Проход по каждому второму предложению
        for (size_t j = i+1; j < (*cnt_sentences); j++){
            len_scd = strlen((*sentences)[j]);
            if (len_fst == len_scd){
                scd = realloc(scd, sizeof(char)*(len_scd + 1));
                for (size_t k=0; k < (len_scd+1); k++){
                    scd[k] = tolower((*sentences)[j][k]);
                }

                // Удаление второго предложения при равенстве с первым
                if (strcmp(fst, scd) == 0){
                    for (size_t k = j+1; k < (*cnt_sentences); k++){
                        (*sentences)[k-1] = (*sentences)[k];
                    }
                    (*cnt_sentences)--;
                    (*sentences)        =        realloc((*sentences),
(*cnt_sentences)*sizeof(char*));
                }
            }
        }
    }
}
```

Название файла: printtext.h

```
#ifndef printtext
void printtext(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: printtext.c

```
#ifndef stdio
#include <stdio.h>
#endif

#include "printtext.h"


void printtext(char ***sentences, size_t *cnt_sentences){
    for (size_t i = 0; i < (*cnt_sentences); i++){
        printf("%s\n", (*sentences)[i]);
    }
}
```

Название файла: freetext.h

```
#ifndef freetext
void freetext(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: freetext.c

```
#ifndef stdlib
#include <stdlib.h>
#endif

#include "freetext.h"


void freetext(char ***sentences, size_t *cnt_sentences){
    for (size_t i=0; i < *cnt_sentences; i++){
        free((*sentences)[i]);
    }
    free(*sentences);
}
```

Название файла: bananapplesearch.h

```
#ifndef bananapplesearch
void bananapplesearch(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: bananapplesearch.c

```
#ifndef string
#include <string.h>
#endif
#ifndef stdio
#include <stdio.h>
#endif

#include "bananapplesearch.h"

#define RESET   "\033[0m"
#define YELLOW  "\033[33m"
#define GREEN   "\033[32m"


void bananapplesearch(char ***sentences, size_t *cnt_sentences){
    int isnewword;
    int isprint;

    // Проход по предложениям
    for (size_t i=0; i < *cnt_sentences; i++){
        isnewword = 1;
        isprint = 0;

        // Проход по символам предложений
        for (size_t j = 0; (*sentences)[i][j];){
            if (isnewword && !strncmp(&((*sentences)[i][j]), "banana",
6) &&
                    (       (*sentences)[i][j+6]==' '       ||
(*sentences)[i][j+6]==',' || (*sentences)[i][j+6]=='.' )){
                if (!isprint){
                    for (size_t k = 0; k < j; k++){
                        printf("%c", (*sentences)[i][k]);
                    }
                }
```

```
                isnewword = 0;
                isprint = 1;
                printf("%s", YELLOW);
                for (unsigned short k = 0; k < 6; k++){
                    printf("%c", (*sentences)[i][j]);
                    j++;
                }
                printf("%s", RESET);
            }
            else  if  (isnewword  &&  !strncmp(&((*sentences)[i][j]),
"apple", 5) &&
                    (       (*sentences)[i][j+5]==' '       ||
(*sentences)[i][j+5]==',' || (*sentences)[i][j+5]=='.' )){
                if (!isprint){
                    for (size_t k = 0; k < j; k++){
                        printf("%c", (*sentences)[i][k]);
                    }
                }
                isnewword = 0;
                isprint = 1;
                printf("%s", GREEN);
                for (unsigned short k = 0; k < 5; k++){
                    printf("%c", (*sentences)[i][j]);
                    j++;
                }
                printf("%s", RESET);
            }
            else{
                if        (((*sentences)[i][j]==' '       ||
(*sentences)[i][j]==',')) isnewword = 1;
                else isnewword = 0;
                if (isprint) printf("%c", (*sentences)[i][j]);
                j++;
            }
        }
        if (isprint) printf("\n");
    }
}
```

Название файла: convertnumsandalpha.h

```
#ifndef convertnumsandalpha
void convertnumsandalpha(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: convertnumsandalpha.c

```
#ifndef stddef
#include <stddef.h>
#endif
#ifndef ctype
#include <ctype.h>
#endif

#include "convertnumsandalpha.h"


void convertnumsandalpha(char ***sentences, size_t *cnt_sentences){
```

```
    for (size_t i=0; i < *cnt_sentences; i++){
        for (size_t j = 0; (*sentences)[i][j]; j++){
           if (isdigit((*sentences)[i][j])){
                (*sentences)[i][j] = 'D';
           }
            else{
                (*sentences)[i][j] = tolower((*sentences)[i][j]);
            }
         }
    }
}
```

Название файла: siftedoutput.h

```
#ifndef siftedoutput
void siftedoutput(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: siftedoutput.c

```
#ifndef stdio
#include <stdio.h>
#endif
#ifndef string
#include <string.h>
#endif

#include "siftedoutput.h"


void siftedoutput(char ***sentences, size_t *cnt_sentences){
    for (size_t i=0; i < *cnt_sentences; i++){
        if (strlen((*sentences)[i]) <= 15){
            printf("%s\n", (*sentences)[i]);
        }
    }
}
```

Название файла: sorttextnums.h

```
#ifndef sorttextnums
int cmpsntncs(const void *first, const void *second);
void sorttextnums(char ***sentences, size_t *cnt_sentences);
#endif
```

Название файла: sorttextnums.c

```
#ifndef stdlib
#include <stdlib.h>
#endif
#ifndef ctype
#include <ctype.h>
#endif

#include "sorttextnums.h"


int cmpsntncs(const void *first, const void *second){
```

```
      const char *f = *((const char **)first);
      const char *s = *((const char **)second);
      size_t product_num_f = 1;
      size_t product_num_s = 1;

      // Нахождение произведения цифр первого предложения
      for (size_t i = 0; f[i]; i++){
            if (isdigit(f[i])){
                  product_num_f *= (size_t)f[i];
            }
      }

      // Нахождение произведения цифр второго предложения
      for (size_t i = 0; s[i]; i++){
            if (isdigit(s[i])){
                  product_num_s *= (size_t)s[i];
            }
      }

      return product_num_f - product_num_s;
}

void sorttextnums(char ***sentences, size_t *cnt_sentences){
      qsort(*sentences, *cnt_sentences, sizeof(char *), cmpsntncs);
}
```

Название файла: Makefile

```
all: cw

cw: menu.o getcommand.o readtext.o preprocessing.o printtext.o
freetext.o bananapplesearch.o convertnumsandalpha.o siftedoutput.o
sorttextnums.o
      gcc -std=gnu99 menu.o getcommand.o readtext.o preprocessing.o
printtext.o freetext.o bananapplesearch.o convertnumsandalpha.o
siftedoutput.o sorttextnums.o -o cw

menu.o: menu.c
      gcc -std=gnu99 -c menu.c

getcommand.o: getcommand.c getcommand.h
      gcc -std=gnu99 -c getcommand.c

readtext.o: readtext.c readtext.h
      gcc -std=gnu99 -c readtext.c

preprocessing.o: preprocessing.c preprocessing.h
      gcc -std=gnu99 -c preprocessing.c

printtext.o: printtext.c printtext.h
      gcc -std=gnu99 -c printtext.c

freetext.o: freetext.c freetext.h
      gcc -std=gnu99 -c freetext.c

bananapplesearch.o: bananapplesearch.c bananapplesearch.h
      gcc -std=gnu99 -c bananapplesearch.c
```

```
convertnumsandalpha.o: convertnumsandalpha.c convertnumsandalpha.h
	gcc -std=gnu99 -c convertnumsandalpha.c

siftedoutput.o: siftedoutput.c siftedoutput.h
	gcc -std=gnu99 -c siftedoutput.c

sorttextnums.o: sorttextnums.c sorttextnums.h
	gcc -std=gnu99 -c sorttextnums.c

clear:
	rm *.o
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица 1 – Результаты тестирования функциональности

№ п/п	Входные данные	Выходные данные	Комментарии
	0 kill me. kill me. eae.	kill me. eae.	
	1 apple good. wood not eat. banana yes. bananas few. banana and apple best	apple good. banana yes. banana and apple best.	
	2 9enIs bro.	Denis bro.	
	3 biiiiig, sntnc. small.	small.	
	4 hi 555. but2s 5. no 999	but2s 5. hi 555. no 999.	
	5	Help about the functions that the program implements: …	
	1 banana. banana.	banana.	Проверка препроцессинга с другой функцией.

Таблица 2 – Результаты тестирования ошибок

№ п/п	Входные данные	Выходные данные	Комментарии
1.	command	Error: Unexpected command	
2.	3	Error: Text is missing	
3.	1 -.	Error: Unexpected symbol	