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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка строк на языке С

Студент гр. 1382 Шушков Е. В.

Преподаватель Жангиров Т. Р.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Шушков Е. В.

Группа 1382

Тема работы: Обработка строк на языке С

Исходные данные:

Вариант 13

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

Для хранения предложения и для хранения текста требуется реализовать структуры Sentence и Text

Программа должна сохранить (считать) текст в виде динамического массива предложений и оперировать далее только с ним. Функции обработки также должны принимать на вход либо текст (Text), либо предложение (Sentence).

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) Сделать сдвиг слов в предложении на положительное целое число N. Например, предложение “abc b#c ИЙ два” при N = 2 должно принять вид “ИЙ два abc b#c”.
2) Вывести все уникальные кириллические и латинские символы в тексте.
3) Подсчитать и вывести количество слов (плюс слова в скобках) длина которых равна 1, 2, 3, и.т.д..
4) Удалить все слова которые заканчиваются на заглавный символ.

Все сортировки и операции со строками должны осуществляться с использованием функций стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.
Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Также, должен быть написан Makefile.

Предполагаемый объем пояснительной записки:

Не менее 32 страницы.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 22.12.2021

Дата защиты реферата: 24.12.2021

Студент _____ Шушков Е. В.

Преподаватель _____ Жангиров Т. Р.

АННОТАЦИЯ

Курсовая работа включает в себя реализацию программы для обработки текста на языке Си. Для хранения и работы с текстом были использованы структуры и функции стандартных библиотек языка Си.

Программа удаляет из введённого текста повторяющиеся предложения и предлагает пользователю выбрать одну из доступных команд для обработки текста. Для удобного взаимодействия с программой на экран выводятся подсказки по пользованию. Пользователь может обрабатывать текст, комбинируя команды, до тех, пока не введёт команду для выхода из программы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы - написать программу, которая обрабатывает строки в зависимости от введённой пользователем команды. Для того, чтобы создать такое приложение необходимо решить следующие задачи:

- Настроить работу с текстом: ввод, хранение и вывод.
- Реализовать работу с кириллицей, т. е. широкими символами.
- Написать Makefile для сборки программы.
- Реализовать интерфейс для взаимодействия с пользователем.
- Реализовать правильную работу с динамической памятью.

Для работы с текстом программа использует динамическую память, которая выделяется и\или освобождается при поступлении новых данных. Библиотека stdlib.h содержит в себе инструменты для работы с динамической памятью. Эти инструменты используют указатели разных типов для обращения к памяти.

Для комфортной работы с кодом программа разбита на несколько файлов с разными функциями. Файлы собираются с помощью Makefile.

Для обработки кириллических символов используются библиотеки wchar.h и wctype.h, которые позволяют правильно работать с широкими символами.

1. Ход выполнения работы

1.1. Ввод текста.

Для хранения текста и работы с ним были реализованы структуры Sentence и Text. Большинство из функций в коде работают именно с указателями на тип этих структур.

Чтения текста происходит через функции SentRead() и TextRead(). Первая — считывает предложения и возвращает указатель на структуру Sentence. Вторая — использует функцию SentRead(), чтобы по предложениям считать текст. Возвращает структуру Text. Также в TextRead() происходит сравнение предложений по ходу чтения с помощью функции str_is_equal(). Если есть повторы, то предложение игнорируется. Таким образом, мы не выделяем лишней памяти.

Если же компьютер не может выделить больше памяти для текста, программа выдаёт пользователю ошибку, освобождает всю память и завершает программу.

1.2. Выбор операции.

Для успешного взаимодействия пользователя встречает интуитивно понятный текстовой интерфейс. Далее после ввода своего текста пользователь получает инструкцию с номерами команд и их описаниями функций. В зависимости от введённого значения программа выполняет определённые инструкции.

1.3. Первая подзадача.

Чтобы сместить все слова в выбранном пользователем предложении, для начала в новый массив широких символов выписывается последнее слово перед точкой. После этого все символы текста смещаются вправо на длину последнего слова. Затем в начало предложения записывается последнее слово из массива. Этот цикл повторяется необходимое количество раз, заданное пользователем.

1.4. Вторая подзадача.

Функция создаёт с помощью calloc (т.е. во всех ячейках будет ноль) 4 массива указателей на тип данных int для латинских и кириллических больших-маленьких символов. Пробегаясь по тексту, от каждого символа, в зависимости от его характеристик (языка и регистра), отнимается “первый” символ “его” алфавита. Таким образом, мы получаем его “номер” по алфавиту и прибавляем единицу к значению из массива с соответствующим индексом. Когда программа дочитывает текст, она идёт по каждому из 4 массивов (в двух циклах: для кириллических и латинских символов) и выводит на экран расшифрованные уникальные (где значения в массиве равны одному) символы с помощью прибавления “первого” символа алфавита к индексу.

1.5. Третья подзадача.

Функция пробегается по тексту два раза. В первый — она ищет максимальную длину слова, чтобы создать массив для количества слов разных длин. Во второй — она считывает длины слов и прибавляет единицу к массиву с индексом этой длины. После второго “пробега” выводит на экран длину слова и количество слов с такой длиной.

1.6. Четвёртая подзадача.

Функция считывает предложение с конца, и когда находит символ на конце слова равный этому символу в верхнем регистре (реализовано с помощью функции towupper()), начинает считывать его длину. После этого правая от слова часть предложения смещается влево на эту длину. Если слово было больше правой части, то лишняя часть предложения за точкой стирается. Длина предложения также уменьшается на длину слова. После удаления всех нужных слов проверяется, имеет ли предложение смысл (остались ли в нём слова). Если нет, то это предложение помечается через специальное поле zero_sent структуры Sentence, таким образом, оно будет игнорироваться при выводе текста. На экран выводится все остальные предложения.

Заключение

В результате выполнения курсовой работы было создано консольное приложение для обработки текста согласно запросам пользователя. Все задачи также были успешно выполнены: программа может сделать сдвиг слов в предложении, вывести все уникальные кириллические и латинские символы в тексте, подсчитать и вывести количество слов длин, которые есть в тексте, а также удалить все слова которые заканчиваются на заглавный символ. Можно сделать вывод о соответствии полученного результата поставленной цели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- Сайт (онлайн-справочник) www.c-cpp.ru
- Сайт (онлайн курс) программирование Си, 1 семестр

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
Здравствуйте, вас приветствует программа для обработки текста!
Сейчас вам предложат ввести текст с клавиатуры.
Предложения нужно разделять одним пробелом. Ввод текста завершится после переноса строки.
<-------------------------------------------------------------------------------------------------------------------->
Введите свой текст:
Текст для примера. Слово. СлОво. Текст.
<-------------------------------------------------------------------------------------------------------------------->
Ваш текст без повторных предложений:
Текст для примера. Слово. Текст.
<-------------------------------------------------------------------------------------------------------------------->
Выберите команду, которую хотите выполнить:
1. Сделать сдвиг слов в предложении на положительное целое число N.
2. Вывести все уникальные кириллические и латинские символы в тексте.
3. Удалить все слова которые заканчиваются на заглавный символ.
4. Подсчитать и вывести количество слов (плюс слова в скобках), длина которых равна 1, 2, 3 и т.д.
Другой символ. Выход из программы.
```

Пример 1. Интерфейс программы.

```
<-------------------------------------------------------------------------------------------------------------------->
Введите свой текст:
Раз море океан переговоры.
<-------------------------------------------------------------------------------------------------------------------->
```

```
<-------------------------------------------------------------------------------------------------------------------->
Введите номер предложения:
1
Введите, на сколько позиций хотите сдвинуть предложение:
3
Предложение со сдвигом:
море океан переговоры Раз.
<-------------------------------------------------------------------------------------------------------------------->
```

Пример 2. Первая подзадача.

```
<------------------------------------------------------------------------------------------------------------------------>
Введите свой текст:
AAAAAAAAAA6AAAAAAA. AAAAAAAAAhAAAAAAA.
<------------------------------------------------------------------------------------------------------------------------>
```

```
<------------------------------------------------------------------------------------------------------------------------>
Уникальные латинские символы:
h
Уникальные кириллические символы:
б
<------------------------------------------------------------------------------------------------------------------------>
```

Пример 3. Вторая подзадача

```
<------------------------------------------------------------------------------------------------------------------------>
Ваш текст без повторных предложений:
пРОВЕРКА текстаА. ЧтО))) КаК))) почему!. ВопросЫ. от5веты. РАЗА. РАЗ. Ра. отóветы.
<------------------------------------------------------------------------------------------------------------------------>
Выберите команду, которую хотите выполнить:
1. Сделать сдвиг слов в предложении на положительное целое число N.
2. Вывести все уникальные кириллические и латинские символы в тексте.
3. Удалить все слова которые заканчиваются на заглавный символ.
4. Подсчитать и вывести количество слов (плюс слова в скобках), длина которых равна 1, 2, 3 и т.д.
Другой символ. Выход из программы.

3
<------------------------------------------------------------------------------------------------------------------------>
Обработанный текст:
 ЧтО))) КаК))) почему!. от5веты. Ра. отóветы.
<------------------------------------------------------------------------------------------------------------------------>
```

Пример 4. Третья подзадача

```
<------------------------------------------------------------------------------------------------------------------------>
Введите свой текст:
Раз два три со2рок. Я 223. Почта (а).
<------------------------------------------------------------------------------------------------------------------------>
```

```
<------------------------------------------------------------------------------------------------------------------------>
Количество слов с определённой длинной:
Слов размером 1: 2
Слов размером 3: 4
Слов размером 5: 1
Слов размером 6: 1
<------------------------------------------------------------------------------------------------------------------------>
```

Пример 5. Четвёртая подзадача

Пример 6. Выход из программы.

ПРИЛОЖЕНИЕ В

КОД ПРОГРАММЫ

Название файла: main.c

```
#include "main.h"
#include "read_text.h"
#include "mutual_act.h"
#include "tasks.h"

int main() {
  //установка языка системы
  setlocale(LC_ALL, "");
  int x, break_point = 1;
  struct Text txt0;
  wprintf(L"Здравствуйте, вас приветствует программа для обработки текста!\n");
  wprintf(L"Сейчас вам предложат ввести текст с клавиатуры.\n");
  wprintf(L"Предложения нужно разделять запятой и одним пробелом. Ввод текста завершится после переноса строки.\n");

wprintf(L"<----------------------------------------------------------------------------------------------------------->\n");
  wprintf(L"Введите свой текст:\n");
  txt0 = TextRead();
  if (txt0.sent_count == -1) {
     return 0;
  }
```

```
wprintf(L"<----------------------------------------------------------------------------
---------------------------->\n");
  wprintf(L"Ваш текст без повторных предложений:\n");
  print_text(txt0);
  wprintf(L"\n");
  while (break_point != 100) {

wprintf(L"<----------------------------------------------------------------------------
---------------------------->\n");
      Comand();
      int sym = 'x';
      wscanf(L"%d", &sym);


wprintf(L"<----------------------------------------------------------------------------
---------------------------->\n");
      int num, N;
      switch (sym) {
         case 1:
            wprintf(L"Введите номер предложения:\n");
            wscanf(L"%ld", &num);
            if(num<txt0.sent_count){
               if (num == 1) {
                  num = 0;
               }
               wprintf(L"Введите, на сколько позиций хотите сдвинуть
предложение:\n");
```

```
            wscanf(L"%d", &N);
            wprintf(L"Предложение со сдвигом:\n");
            WordShift(txt0.text[num]->Sent, N);
            wprintf(L"\n");
        }else{
            wprintf(L"Предложения с таким номером не существует!\n");
        }
        break;
    case 2:
        UniqueSymbols(txt0);
        break;
    case 3:
        wprintf(L"Обработанный текст:\n");
        RemovingLastUpper(txt0);
        break;
    case 4:
        wprintf(L"Количество слов с определённой длинной:\n");
        WordLensCount(txt0);
        break;
    default:
        Free_Mem(txt0.text, txt0.sent_count);
        wprintf(L"Приятного дня!\n");
        break_point = 100;
    }
  }
  return 0;
}
```

Название файла main.h:

```
#include <stdlib.h>
#include <strings.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>

#define MEM_STEP 5

struct Sentence {
  wchar_t *Sent;
  int size;
  int zero_sent;
};

struct Text {
  struct Sentence **text;
  int sent_count;
  int size;
};
```

Название файла: mutual_act.c

```
#include "main.h"

void Comand(){
  wprintf(L"Выберите команду, которую хотите выполнить:\n");
  wprintf(L"1. Сделать сдвиг слов в предложении на положительное целое число N.\n");
```

```
  wprintf(L"2. Вывести все уникальные кириллические и латинские символы в тексте.\n");
  wprintf(L"3. Удалить все слова которые заканчиваются на заглавный символ.\n");
  wprintf(L"4. Подсчитать и вывести количество слов (плюс слова в скобках), длина которых равна 1, 2, 3 и т.д.\n");
  wprintf(L"Другой символ. Выход из программы.\n\n");
}

void Free_Mem(struct Sentence** txt, int num){
  for(int i=0;i<=num;i++){
     free(txt[i]);
  }
  free(txt);
}

void print_text(struct Text txt){
  for(int i=0; i<txt.sent_count; i++){
     if(txt.text[i]->zero_sent==1){
        for(int j=0; j<wcslen(txt.text[i]->Sent); j++){
           wprintf(L"%lc", txt.text[i]->Sent[j]);
        }
     }
  }
}

wchar_t* DoublePointDelete(wchar_t* sent){
  for(int k=0;k< wcslen(sent);k++){
```

```
      if(sent[k]==' ' && sent[k+1]==' '){
         for(int q=k+1;q<wcslen(sent);q++){
            sent[q]=sent[q+1];
         }
         k--;
      }
   }
}
```

Название файла: mutual_act.h

```
void Comand();
void Free_Mem(struct Sentence** txt, int num);
void print_text(struct Text txt);
wchar_t* DoublePointDelete(wchar_t* sent);
```

Название файла: read_text.c

```
#include "main.h"
#include "mutual_act.h"

//Проверка на равенство предложений(возвращает 1, если одинаковые. Разные - 0)
int str_is_equal(wchar_t* str_1, wchar_t* str_2)
{
   if(wcslen(str_1)==wcslen(str_2)){
      for(int i=0; i<wcslen(str_1);i++){
         if(towlower(str_1[i])!=towlower(str_2[i])) return 0;
      }
   } else return 0;
```

```
    return 1;

}

//чтение предложения
struct Sentence* SentRead(){

    int size = MEM_STEP;
    int num = 0;
    wchar_t sym;
    wchar_t* sent = malloc(size*sizeof(wchar_t));
    if(sent==NULL){
        wprintf(L"Компьютер не смог выделить память!");
        return 0;
    }

    while(sym!=L'.' && sym!=L'\n'){
        sym=getwchar();
        sent[num++]=sym;

        if(num==size-1){
            size+=MEM_STEP;
            sent=realloc(sent,size*sizeof(wchar_t));
            if(sent==NULL){
                free(sent);
                wprintf(L"Компьютер не смог выделить память!");
                return 0;
            }
```

```
        }
    }
    sent[num]=L'\0';
    struct Sentence* sentence = malloc(sizeof(struct Sentence));
    if(sentence==NULL){
        wprintf(L"Компьютер не смог выделить память!");
        free(sent);
        return 0;
    }
    sentence->Sent = sent;
    sentence->size = num;
    sentence->zero_sent=1;
    return sentence;

}

//чтение текста
struct Text TextRead(){
    int size = MEM_STEP;
    struct Sentence** Text=malloc(size*sizeof(struct Sentece*));

    if(Text==NULL){
        wprintf(L"Компьютер не смог выделить память!");
        struct Text txtx;
        txtx.sent_count=-1;
        return txtx;
    }
```

```
int num=0, replay_signal=0;
struct Sentence* sent;
while(1){
   sent = SentRead();
   if(sent==0){
      wprintf(L"Компьютер не смог выделить память!");
      Free_Mem(Text,num);
      struct Text txtx;
      txtx.sent_count=-1;
      return txtx;
   }

   if(sent->Sent[0]==L'\n'){
      break;
   }

   for(int i=0;i<num;i++){
      if(str_is_equal(Text[i]->Sent, sent->Sent)==1){
         replay_signal++;
      }
   }

   if(replay_signal==0){
      Text[num]=sent;
      num++;
   }else{
      replay_signal=0;
   }
```

```
        if(num==size-1){
            size+=MEM_STEP;
            Text=realloc(Text,size*sizeof(struct Sentence*));
            if(Text==NULL){
                wprintf(L"Компьютер не смог выделить память!");
                Free_Mem(Text,num);
                struct Text txtx;
                txtx.sent_count=-1;
                return txtx;
            }
        }
    }
    struct Text txt;
    txt.text = Text;
    txt.sent_count=num;
    txt.size = size;
    return txt;
}
```

Название файла: read_text.h

```
int str_is_equal(wchar_t* str_1, wchar_t* str_2);
struct Sentence* SentRead();
struct Text TextRead();
```

Название файла: tasks.c

```
#include "main.h"
#include "mutual_act.h"
```

```
//смещение слов в предложениии
void WordShift(wchar_t *sent, int shift_count) {

  int i = 0;
  int sent_s = wcslen(sent) - 2;
  while (i < shift_count) {
    int size_last_word = 0;
    wchar_t sym = 'a';

    //поиск кол-ва индексов для смещения первого слова, т.е. длина последнего+пробел
    for (int k = sent_s; sym != L' ' && k>-2; k--) {
      sym = sent[k];
      size_last_word++;
    }

    wchar_t *last_word = malloc((size_last_word - 1) * sizeof(wchar_t));

    //Запись последнего слова в отдельный массив
    int idx_last_word = 0;
    for (int k = sent_s - size_last_word + 2; k < sent_s + 1; k++) {
      last_word[idx_last_word++] = sent[k];

    }

    //смещение слов прошлого массива
    for (int k = sent_s - size_last_word; k > -1; k--) {
```

```
            sent[k + size_last_word] = sent[k];
        }

        for (int k = 0; k < size_last_word; k++) {
            sent[k] = ' ';
        }

        // подстановка последнего слова в начало
        for (int k = 0; k < size_last_word - 1; k++) {
            sent[k] = last_word[k];
        }

        free(last_word);
        i++;
    }

    DoublePointDelete(sent);
    wprintf(L"%ls", sent);
}

//поиск уникальных символов
void UniqueSymbols(struct Text pr_text) {
    //Создаём массивы для каждого алфавита
    int *latin_large = calloc(26, sizeof(int));
    int *cyrillic_large = calloc(33, sizeof(int));
    int *latin_low = calloc(26, sizeof(int));
    int *cyrillic_low = calloc(33, sizeof(int));
```

```
//проверяем каждый символ. В зависимости от алфавита прибавляем к "его" ячейке единицу
for (int i = 0; i < pr_text.sent_count; i++) {
    wchar_t *sent = pr_text.text[i]->Sent;
    for (int j = 0; j < wcslen(sent); j++) {
        if ('A' <= sent[j] && sent[j] <= 'Z') {
            latin_large[sent[j] - 'A']++;
        } else if ('a' <= sent[j] && sent[j] <= 'z') {
            latin_low[sent[j] - 'a']++;
        } else if (L'A' <= sent[j] && sent[j] <= L'Я') {
            cyrillic_large[sent[j] - L'A']++;
        } else if (L'a' <= sent[j] && sent[j] <= L'я') {
            cyrillic_low[sent[j] - L'a']++;
        }
    }
}

wprintf(L"Уникальные латинские символы:\n");

int br_p = 0, br_p2 = 0;

//выводим уникальные латинские символы
for (int i = 0; i < 26; i++) {
    if (latin_large[i] == 1) {
        wprintf(L"%c ", i + 'A');
        br_p++;
    }
    if (latin_low[i] == 1) {
```

```
            wprintf(L"%c ", i + 'A');
            br_p++;
        }
    }
    if (br_p == 0) {
        wprintf(L"Нет");
    }

    wprintf(L"\nУникальные кириллические символы:\n");
    //выводим уникальные кириллические символы
    for (int i = 0; i < 33; i++) {
        if (cyrillic_large[i] == 1) {
            br_p2++;
            wprintf(L"%lc ", i + L'A');
        }
        if (cyrillic_low[i] == 1) {
            br_p2++;
            wprintf(L"%lc ", i + L'а');
        }
    }

    if (br_p2 == 0) {
        wprintf(L"Нет");
    }
    wprintf(L"\n");

    free(latin_large);
    free(latin_low);
```

```
    free(cyrillic_large);
    free(cyrillic_low);

}

//Удаление слов с Заглавной на конце
void RemovingLastUpper(struct Text pr_text) {
    int len_word = 0;
    wchar_t sym = ' ';
    for (int i = 0; i < pr_text.sent_count; i++) {
        int real_size = 0;
        wchar_t *sent = pr_text.text[i]->Sent;
        for (int j = wcslen(sent) - 1; j > -1; j--) {
            sym = towupper(sent[j]);
            len_word = 0;

            if (sent[j] == sym && iswalpha(sent[j]) &&
                (sent[j + 1] == L' ' || sent[j + 1] == L'.' || sent[j + 1] == L',')) {
                int first_sym;
                for (int k = j; k > -1 && sent[k] != L' ' && sent[k] != L'.' && sent[k]
!= L','; k--) {
                    len_word++;
                    first_sym = k;
                }
                real_size += len_word;
                for (int k = first_sym; k <= wcslen(sent); k++) {
                    if ((k + len_word) <= wcslen(sent)) {
```

```
                sent[k] = sent[k + len_word];
            } else {
                sent[k] = ' ';
            }
        }
    }
    DoublePointDelete(sent);
    if (sent[wcslen(sent) - 2] == ' ') {
        sent[wcslen(sent) - 2] = sent[wcslen(sent) - 1];
        sent[wcslen(sent) - 1] = ' ';
    }

    if (wcscasecmp(sent, L".") == 0 || wcscasecmp(sent, L". ") == 0
        ) {
        pr_text.text[i]->zero_sent = 0;
    }
  }

  print_text(pr_text);
  wprintf(L"\n");
}

//кол-во слов опред. длины
void WordLensCount(struct Text pr_text) {
  int read_size = 0, max_size = 0;

  //Поиск наибольшей длины слова
```

```
  for (int i = 0; i < pr_text.sent_count; i++) {
     wchar_t *sent = pr_text.text[i]->Sent;
     for (int j = 0; j < wcslen(sent); j++) {
        if (sent[j] != L' ' && sent[j] != L'.' && sent[j] != L'(' && sent[j] != L')')
{
           read_size++;
        } else {
           if (read_size > max_size) {
              max_size = read_size;
           }
           read_size = 0;
        }

     }
  }

  int *counts = calloc(max_size + 1, sizeof(int));

  //запись длин слов
  for (int i = 0; i < pr_text.sent_count; i++) {
     wchar_t *sent = pr_text.text[i]->Sent;
     for (int j = 0; j < wcslen(sent); j++) {
        if (sent[j] != L' ' && sent[j] != L'.' && sent[j] != L'(' && sent[j] != L')')
{
           read_size++;
        } else {
           counts[read_size] += 1;
           read_size = 0;
```

```
        }


    }
  }

  for (int i = 1; i < max_size + 1; i++) {
    if (counts[i] != 0) {
      wprintf(L"Слов размером %d: %d\n", i, counts[i]);
    }
  }
  free(counts);
}
```

Название файла: task.h

```
void WordShift(wchar_t *sent, int shift_count);
void UniqueSymbols(struct Text pr_text);
void RemovingLastUpper(struct Text pr_text);
void WordLensCount(struct Text pr_text);
```

Название файла: Makefile

```
all: main.o read_text.o tasks.o mutual_act.o
  gcc main.o read_text.o tasks.o mutual_act.o -o main
main.o: main.c main.h read_text.h tasks.h mutual_act.h
  gcc -c main.c
read_text.o: read_text.c main.h read_text.h
  gcc -c read_text.c
tasks.o: tasks.c main.h tasks.h
```

```
  GCC -C TASKS.C

MUTUAL_ACT.O: MUTUAL_ACT.C MAIN.H MUTUAL_ACT.H

  GCC -C MUTUAL_ACT.C

CLEAN:

  RM *.O
```