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

Курсовая работа
по дисциплине «Программирование»
Тема: Обработка строк.

Студент гр. 0381 _____ Самойлов З.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Самойлов З.А.

Группа 0381

Тема работы: обработка строк.

Исходные данные: программа должна быть модульной, обработка данных должна производится при возможности с использованием функций стандартных библиотек языка. Сохранение, обработка и вывод данных должны осуществляться с помощью работы со структурами и динамической памятью.

Содержание пояснительной записки:
«Содержание», «Введение», «Задание работы», «Ход выполнения работы» «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 23.11.2018
Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 0381 _____ Самойлов З.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В курсовой работе была реализована обработка текста произвольной длины, для этого были использованы структуры и динамические массивы. Также в работе активно использовались стандартная библиотека языка Си, содержащиеся в них функции и типы данных. В программе реализован элементарный интерфейс общения с пользователем и выполнение запрашиваемых им действий.
Пример работы программы приведён в приложении А.

SUMMARY

In the course work was implemented text processing of arbitrary length, for this were used structures and dynamic arrays. In addition, standard C libraries were actively used in the work, the functions and data types contained in them. The program implements an elementary interface for communicating with the user and performing the actions requested by him. An example of the program is given in Appendix A.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: создать стабильную программу, производящую выбранную пользователем обработку данных. Реализация программы должна содержать работу со структурами (для хранения текста, отдельных предложений и слов, а также дополнительных данных), работу с динамически выделенной памятью и использование стандартных библиотек, в том числе для работы с национальным алфавитом (wchar.h).

Разработка велась на базе операционной системы Linux Ubuntu 20.04 в текстовом редакторе Atom. Компиляция и линковка производилась с помощью пакета утилит GNU Compiler Collection.

В результате была создана программа, считывающая вводимый пользователем с консоли текст, выводящая меню со списком доступных функций и выполняющая выбранные. По запросу пользователя программа выводит текст. При завершении работы производятся действия по очистке динамически выделенной памяти.

1 ЗАДАНИЕ

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. Распечатать каждое слово которое встречается не более одного раза в тексте.
2. Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04/1889”.
3. Отсортировать предложения по произведению длин слов в предложении.
4. Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной цифры.

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.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Создание структур и общего заголовочный файл.

Для работы с текстом были созданы три структуры Text, Sentense и Word, при объявлении при помощи оператора typedef они были определены как новые типы данных Text, Sentence и Word соответственно. Структуры были объявлены в заголовочном файле, и была подключена препроцессорная директива pragma once для избежание повторного объявления. (Далее будем опускать, что каждый заголовочный файл проходил аналогичное подключение).

Содержание структур: в Text содержится переменная – счётчик предложений sentences_count и динамический массив указателей на структуру Sentense. В структуре Sentence содержатся переменные, характеризующие предложение: счетчик слов words_count, счетчик произведения длин всех слов предложения words_length_multiplication, два флага symbol и number, отвечающие за наличие в предложении специальных символов «#»/«№» и чисел соответственно, а также динамический массив указателей на структуру Word. В структуре Word содержатся счетчик appearence, отвечающий за кол-во повторений слова в тексте (нам важно только то, является ли значение единичным или нет), symbols_count содержит длину слова, флаг sign, помогающий в расстановке пробела и знаков препинания при выводе текста, указатель entity на динамическую строку, в которой находится само слово.

В заголовочном файле structs.h содержится описание всех структур. Его содержание можно посмотреть в приложении Б.

2.2. Чтение и вывод текста.

В файле input.c содержаться три функции, созданных для считывания текста в структуры, первоначальной обработки (удаление повторяющихся предложений) и вывода текста.

Функция void input(Text * text) производит считывание всего входного текста в структуру Text, разделяя ввод по словам в указатели на структуру Sentence и указатели на структуру Word, сохраняя слова в динамическую строку entity. Для получения данных со стандартного потока ввода используется безопасное посимвольное считывание до символа переноса строки (‘\n’). Перевыделение памяти для слова происходит посимвольно. При попадании пробела или запятой в ввод перевыделяется память под новую структуру Word для следующего слова. При попадании точки в ввод перевыделяется память для нового предложения (кроме того случая, когда точка является заключительной в тексте). Стоит заметить, что, по условию задачи, разделителями считаются пробел, запятая и точка, их комбинации недопустимы.

Функция void filter(Text * text) последовательно проверяет на повторение предложения с использованием ключа copy = 1 и функции int wcscasecmp(const wchar_t *_s1_, const wchar_t *_s2_) заголовочного файла wchar.h для сравнения слов в предложениях. Если слова не совпадают, то ключ переводится в 0 до проверки следующих предложений. Если же после проверки ключ равен единице, то все entity во втором предложении очищаются, а words_count принимает нулевое значение, что позволяет избежать неверный вывод пустого предложения.

Функция void output(Text * text) выводит текст, восстанавливая пробелы и знаки препинания по флагу sign.

2.3. Функции для работы с пользователем.

В файле main.c создан простой алгоритм для взаимодействия с пользователем. После ввода текста ему предлагается ввести один из ключей для выполнения соответствующего действия, согласно аннотации. При вводе

неправильного ключа пользователь получает сообщение о неверно введенном ключе и выполняется повторный вывод аннотации.

2.4. Функции для выполнения указанных операций.

В данном разделе описываются функции, реализованные в handling.c.

2.5.1 Печать уникальных слов.

Функция void print(Text * text) присваивает appearence каждой структуры слов значение 1. Затем выполняется последовательное сравнение слов без учета регистра. При совпадении appearence двух слов увеличивается на 1. В целях экономии времени алгоритм выполняет проверку только тех слов, чей appearence равен единице. После окончания проверки, слова с единичным appearence построчно выводятся пользователю.

2.5.2 Замена формата дат

Функция void date(Text * text) выполняет поиск подстрок вида “<день> <месяц> <год> г.” в концах предложений, заменяет первое слово в подстроке на «слово» вида ДД/ММ/ГГГГ (год>999), остальные слова очищаются.

2.5.3 Функция сортировки текста по произведению длин слов в предложении

Функция void sort(Text * text) считает произведение длин слов words_length_multiplication для каждого предложения. Затем, с помощью стандартного алгоритма сортировки указатели на структуры предложений меняют свой порядок. Результирующий порядок предложений работает по правилу неубывания words_length_multiplication следующего предложения.

2.5.4 Функция удаления некорректных предложений

Функция void delete(Text * text) проверяет предложения на наличие цифр и/или символов «#»/«№», выставляя соответствующие флаги number и symbol для каждого предложения. Затем, ориентируясь по флагам, функция удаляет некорректные предложения.

2.5. Очистка памяти и завершение работы.

После вызова пользователем ключа завершения работы программы происходит последовательное освобождение памяти и завершение выполнения программы.

2.6. Создание Make файлы.

После создания всех файлов с исходным кодом необходимых функций был создан файл makefile для консольной утилиты make, в которой были установлены корректные зависимости компиляции и цели для каждого бинарного файла отдельных исходников и дополнительная функция clean, удаляющая объектные файлы после компиляции исполняемого файла.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы были на практике применены приёмы работы со структурами, динамической памятью, стандартными функциями библиотек языка Си. Была создана программа с обработкой ввода и выполняющая функции, указанные в задании и выбранные пользователем. Программа осуществляет взаимодействие с пользователем через консольный интерфейс и корректно обрабатывает различные виды корректных входных данных (в соответствии с условием).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.Керниган Б. и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978 288 с.

2. Онлайн справочник по C и C++. c-cpp.ru

3. Стив Оолайн C Elements of Style M.: M&T books, 1992 456 с.

4. Стандартная библиотека языка программирования C и страницы заголовочных файлов на Вики. https://ru.wikipedia.org/wiki/Стандартная_библиотека_языка_Си

ПРИЛОЖЕНИЕ А
ПРИМЕР РАБОТЫ ПРОГРАММЫ

Пример вывода консоли после вызова make (An example of console output after a call to make):

```
richsweetwafer@ASUS-TUF:~/Education/Programming/cw$ make
gcc -c main.c
gcc -c input.c
gcc -c handling.c
gcc -o main main.o input.o handling.o
```

Пример ввода и редактирования предложений. (An example of the input and editing of sentences):

```
Введите текст:
Два столетия тому назад шла война с французами.Бой за Малоярославец состоялся 12 октября 1812 г.Наш герой,Степанов Виктор,шел 3 номером.Он был № в отряде.вИктор хотел убежать в тот день,дезертировать.Кто мог знать,что герой получит смертельное ранение ещё прежде,чем заметит врага.
Конец ввода.
Введенный текст:
Два столетия тому назад шла война с французами.Бой за Малоярославец состоялся 12 октября 1812 г.Наш герой,Степанов Виктор,шел 3 номером.Он был № в отряде.вИктор хотел убежать в тот день,дезертировать.Кто мог знать,что герой получит смертельное ранение ещё прежде,чем заметит врага.

Выберите действие над текстом:
Напишите "1", чтобы вывести текст.
Напишите "2", чтобы распечатать все слова, встречающиеся в тексте не более одного раза.
Напишите "3", чтобы заменить строку вида "<день> <месяц> <год> г." на "ДД/ММ/ГГГГ".
Напишите "4", чтобы отсортировать предложения в порядке неубывания по произведению длин содержащихся в них слов.
Напишите "5", чтобы удалить все предложения, содержащие '#' или '№', но не числа.
Напишите "0", чтобы прекратить выполнение программы.
3
1
Два столетия тому назад шла война с французами.Бой за Малоярославец состоялся 12/10/1812.Наш герой,Степанов Виктор,шел 3 номером.Он был № в отряде.вИктор хотел убежать в тот день,дезертировать.Кто мог знать,что герой получит смертельное ранение ещё прежде,чем заметит врага.
2
Два
```

```
столетия
тому
назад
шла
война
с
французами
Бой
за
Малоярославец
состоялся
12/10/1812
Наш
Степанов
шел
3
номером
Он
был
№
отряде
хотел
убежать
тот
день
дезертировать
Кто
мог
знать
что
получит
смертельное
ранение
ещё
прежде
чем
заметит
врага
1
Два столетия тому назад шла война с французами.Бой за Малоярославец состоялся 12/10/1812.Наш герой,Степанов Виктор,шел 3 номером.Он был № в отряде.вИктор хотел убежать в тот день,дезертировать.Кто мог знать,что герой получит смертельное ранение ещё прежде,чем заметит врага.
7
Введено неверное значение.
Напишите "1", чтобы вывести текст.
Напишите "2", чтобы распечатать все слова, встречающиеся в тексте не более одного раза.
Напишите "3", чтобы заменить строку вида "<день> <месяц> <год> г." на "ДД/ММ/ГГГГ".
Напишите "4", чтобы отсортировать предложения в порядке неубывания
```

```
по произведению длин содержащихся в них слов.
Напишите "5", чтобы удалить все предложения, содержащие '#' или
'№', но не числа.
Напишите "0", чтобы прекратить выполнение программы.
5
1
Два столетия тому назад шла война с французами.Бой за
Малоярославец состоялся 12/10/1812.Наш герой,Степанов Виктор,шел 3
номером.вИктор хотел убежать в тот день,дезертировать.Кто мог
знать,что герой получит смертельное ранение ещё прежде,чем заметит
врага.
3
1
Два столетия тому назад шла война с французами.Бой за
Малоярославец состоялся 12/10/1812.Наш герой,Степанов Виктор,шел 3
номером.вИктор хотел убежать в тот день,дезертировать.Кто мог
знать,что герой получит смертельное ранение ещё прежде,чем заметит
врага.
4
1
Бой за Малоярославец состоялся 12/10/1812.Наш герой,Степанов
Виктор,шел 3 номером.вИктор хотел убежать в тот
день,дезертировать.Два столетия тому назад шла война с
французами.Кто мог знать,что герой получит смертельное ранение ещё
прежде,чем заметит врага.
0
Exiting the program...
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

файл: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>
#include "structs.h"
#include "input.h"
#include "handling.h"

int main(){
    setlocale(LC_ALL, "ru_RU.UTF-8");
    int par;
    Text text;

    wprintf(L"Введите текст:\n");

    input(&text);
    filter(&text);

    wprintf(L"Конец ввода.\nВведенный текст:\n");

    output(&text);

    wprintf(L"\nВыберите действие над текстом: ");
    annotation();

    while (1){
      wscanf(L"%d", &par);

      switch (par) {
```

```
        case 1:
          output(&text);
          break;

        case 2:
          print(&text);
          break;

        case 3:
          date(&text);
          break;

        case 4:
          sort(&text);
          break;

        case 5:Ы
          delete(&text);
          break;

        case 0:
          wprintf(L"Exiting the program...\n");
          //ОСВОБОЖДЕНИЕ ПАМЯТИ
          for (int t = 0; t < text.sentences_count; t++){
            for (int s = 0; s < text.sentence_p[t].words_count;
s++){
              free(text.sentence_p[t].word_p[s].entity);
            }
            if (text.sentence_p[t].words_count == 0)
              continue;
            free(text.sentence_p[t].word_p);
          }
          free(text.sentence_p);
          // ОСВОБОЖДЕНИЕ ПЯМЯТИ
          return 0;
```

```
        default:
          wprintf(L"Введено неверное значение.");
          annotation();
          break;
      }
    }
}
```

файл structs.h:

```
#pragma once
#include <wchar.h>

typedef struct Word{
  wchar_t *entity;
  int appearence;
  int symbols_count;
  int sign;
} Word;
typedef struct Sentence{
  Word *word_p;
  int words_count;
  long int words_length_multiplication;
  int symbol;
  int number;
} Sentence;
typedef struct Text{
  Sentence *sentence_p;
  int sentences_count;
} Text;
```

файл input.h:

```
#pragma once
#include "structs.h"
```

```
void input(Text * text);
void filter(Text * text);
void output(Text * text);
```

файл input.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>
#include "structs.h"
#include "input.h"

void input(Text * text){
  wchar_t ch;
  wchar_t prev_ch;
  text->sentences_count = 1;

  text->sentence_p = calloc(1, sizeof(Sentence));
  text->sentence_p->words_count = 1;

  text->sentence_p->word_p = calloc(1, sizeof(Word));
  text->sentence_p->word_p[0].symbols_count = 1;
  text->sentence_p->word_p[0].entity = calloc(2,
sizeof(wchar_t));

  while(ch = getwchar()){
    if (prev_ch == L'.'){
      if (ch != L'\n'){
        text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count - 1].sign = 0;
        text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count -
1].entity[text->sentence_p[text->sentences_count -
```

```
1].word_p[text->sentence_p[text->sentences_count - 1].words_count
- 1].symbols_count-1] = L'\0';
        text->sentences_count += 1;
        text->sentence_p = realloc(text->sentence_p, text-
>sentences_count * sizeof(Sentence));
        text->sentence_p[text->sentences_count - 1].words_count =
1;
        text->sentence_p[text->sentences_count - 1].word_p =
calloc(1, sizeof(Word));
        text->sentence_p[text->sentences_count -
1].word_p[0].symbols_count = 1;
        text->sentence_p[text->sentences_count -
1].word_p[0].entity = calloc(2, sizeof(wchar_t));
      }
      else break;
    }
    else if (ch == L' ' || ch == L','){
      if (ch == L' '){
        text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count - 1].sign = 1;
      }
      else if (ch == L','){
        text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count - 1].sign = 2;
      }
      text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count -
1].entity[text->sentence_p[text->sentences_count -
1].word_p[text->sentence_p[text->sentences_count - 1].words_count
- 1].symbols_count-1] = L'\0';
      text->sentence_p[text->sentences_count - 1].words_count +=
1;
      text->sentence_p[text->sentences_count - 1].word_p =
realloc(text->sentence_p[text->sentences_count - 1].word_p, text-
```

```
>sentence_p[text->sentences_count - 1].words_count *
sizeof(Word));
      text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count -
1].symbols_count = 1;
      text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count - 1].entity =
calloc(2, sizeof(wchar_t));
      continue;
    }
    else if (ch == L'.'){
      prev_ch = L'.';
      continue;
    }
    text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count -
1].entity[text->sentence_p[text->sentences_count -
1].word_p[text->sentence_p[text->sentences_count - 1].words_count
- 1].symbols_count - 1] = ch;
    text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count - 1].entity =
realloc(text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count - 1].entity,
(++text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count -
1].symbols_count + 1)*sizeof(wchar_t));
    prev_ch = ch;
  };
  text->sentence_p[text->sentences_count - 1].word_p[text-
>sentence_p[text->sentences_count - 1].words_count -
1].entity[text->sentence_p[text->sentences_count -
1].word_p[text->sentence_p[text->sentences_count - 1].words_count
- 1].symbols_count-1] = L'\0';
}
```

```
void filter(Text * text){
  int copy = 0;
  for (int t = 0; t < text->sentences_count; t++){
    for (int s = t+1; s < text->sentences_count; s++){
      if (text->sentence_p[t].words_count == text-
>sentence_p[s].words_count){
        copy = 1;
        for (int i = 0; i < text->sentence_p[t].words_count; i++)
{
          if (wcscasecmp(text->sentence_p[t].word_p[i].entity,
text->sentence_p[s].word_p[i].entity) != 0){
            copy = 0;
            break;
          }
        }
        if (copy == 1){
          for (int w = 0; w < text->sentence_p[s].words_count; w+
+){
            free(text->sentence_p[s].word_p[w].entity);
          }
          free(text->sentence_p[s].word_p);
          text->sentence_p[s].words_count = 0;
        }
      }
    }
  }
}

void output(Text * text){
  for (int t = 0; t < text->sentences_count; t++){
    for (int s = 0; s < text->sentence_p[t].words_count; s++){
      wprintf(text->sentence_p[t].word_p[s].entity);
      if (s !=  text->sentence_p[t].words_count - 1)
        if (text->sentence_p[t].word_p[s].sign == 1) wprintf(L"
");
```

```
        else if (text->sentence_p[t].word_p[s].sign == 2)
wprintf(L",");
    }
    if (text->sentence_p[t].words_count == 0)
      continue;
    wprintf(L".");
  }
  wprintf(L"\n");
}
```

файл handling.h:

```
#pragma once
#include "structs.h"

void print(Text * text);
void date(Text * text);
void sort(Text * text);
void delete(Text * text);
void annotation();
```

файл handling.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>
#include "structs.h"
#include "handling.h"

void print(Text * text){
  for (int t = 0; t < text->sentences_count; t++){
    for (int s = 0; s < text->sentence_p[t].words_count; s++){
      text->sentence_p[t].word_p[s].appearence = 1;
    }
  }
```

```
  for (int t = 0; t < text->sentences_count; t++){
    for (int s = 0; s < text->sentence_p[t].words_count; s++){
      if (text->sentence_p[t].word_p[s].appearence == 1) {
        for (int i = s+1; i < text->sentence_p[t].words_count; i+
+) {
          if (wcscasecmp(text->sentence_p[t].word_p[s].entity,
text->sentence_p[t].word_p[i].entity) == 0) {
            text->sentence_p[t].word_p[s].appearence += 1;
            text->sentence_p[t].word_p[i].appearence += 1;
          }
          for (int t1 = t+1; t1 < text->sentences_count; t1++) {
            for (int s1 = 0; s1 < text-
>sentence_p[t1].words_count; s1++) {
              if (wcscasecmp(text-
>sentence_p[t].word_p[s].entity, text-
>sentence_p[t1].word_p[s1].entity) == 0) {
                text->sentence_p[t].word_p[s].appearence += 1;
                text->sentence_p[t1].word_p[s1].appearence += 1;
              }
            }
          }
        }
      }
    }
  }
  for (int t = 0; t < text->sentences_count; t++){
    for (int s = 0; s < text->sentence_p[t].words_count; s++){
      if (text->sentence_p[t].word_p[s].appearence == 1){
        wprintf(text->sentence_p[t].word_p[s].entity);
        wprintf(L"\n");
      }
    }
  }
}
```

```
void date(Text * text){
  int year;
  wchar_t month[3];
  wchar_t days[31][3] = {L"1", L"2", L"3", L"4", L"5", L"6",
L"7", L"8", L"9", L"10", L"11", L"12", L"13", L"14", L"15",
L"16", L"17", L"18", L"19", L"20",
                          L"21", L"22", L"23", L"24", L"25",
L"26", L"27", L"28", L"29", L"30", L"31",};
  wchar_t months[12][9] = {L"января", L"февраля", L"марта",
L"апреля", L"мая", L"июня", L"июля", L"августа", L"сентября",
L"октября", L"ноября", L"декабря"};
  for (int t = 0; t < text->sentences_count; t++){
    for (int s = 0; s < text->sentence_p[t].words_count-3; s++){
      if(wcscasecmp(text->sentence_p[t].word_p[s+3].entity, L"г")
== 0 && text->sentence_p[t].word_p[s+2].symbols_count == 5){
        for (int i = 0; i < 31; i++) {
            if (wcscmp(text->sentence_p[t].word_p[s].entity,
days[i]) == 0){
              for (int j = 0; j < 12; j++) {
                if(wcscasecmp(text-
>sentence_p[t].word_p[s+1].entity, months[j]) == 0){
                  swprintf(month, 3, L"%d", j+1);
                  wchar_t *p = &text-
>sentence_p[t].word_p[s+2].entity[text-
>sentence_p[t].word_p[s+2].symbols_count];
                  year = wcstol(text-
>sentence_p[t].word_p[s+2].entity, &p, 10);
                  if (year > 999){
                    text->sentence_p[t].word_p[s].entity =
realloc(text->sentence_p[t].word_p[s].entity, 30 *
sizeof(wchar_t));
                    wcscat(text->sentence_p[t].word_p[s].entity,
L"/");
```

```
                    wcscat(text->sentence_p[t].word_p[s].entity,
month);
                    wcscat(text->sentence_p[t].word_p[s].entity,
L"/");
                    wcscat(text->sentence_p[t].word_p[s].entity,
text->sentence_p[t].word_p[s+2].entity);
                    text->sentence_p[t].word_p[s].sign = 0;
                    text->sentence_p[t].word_p[s].symbols_count
+= text->sentence_p[t].word_p[s+1].symbols_count + text-
>sentence_p[t].word_p[s+2].symbols_count + text-
>sentence_p[t].word_p[s+3].symbols_count - 3;
                    text->sentence_p[t].words_count -= 3;
                    for (size_t z = 1; z < 4; z++) {
                      free(text-
>sentence_p[t].word_p[s+z].entity);
                      text->sentence_p[t].word_p[s+z].sign = 0;
                      text-
>sentence_p[t].word_p[s+z].symbols_count = 0;
                    }
                    break;
                  }
                }
              }
            }
          }
        }
      }
    }
  }


  void sort(Text * text){
    Sentence p;
    for (int t = 0; t < text->sentences_count; t++) {
      text->sentence_p[t].words_length_multiplication = 1;
```

```
      for (int s = 0; s < text->sentence_p[t].words_count; s++) {
        text->sentence_p[t].words_length_multiplication *= text-
>sentence_p[t].word_p[s].symbols_count;
      }
    }
    for(int i = 0; i < text->sentences_count; i++){
      for (int j = i+1; j < text->sentences_count; j++) {
        if (text->sentence_p[i].words_length_multiplication >
text->sentence_p[j].words_length_multiplication){
          p = text->sentence_p[i];
          text->sentence_p[i] = text->sentence_p[j];
          text->sentence_p[j] = p;
        }
      }
    }
  }


  void delete(Text * text){
    for (int t = 0; t < text->sentences_count; t++) {
      text->sentence_p[t].symbol = 0;
      text->sentence_p[t].number = 0;
    }
    for (int t = 0; t < text->sentences_count; t++){
      for (int s = 0; s < text->sentence_p[t].words_count; s++){
        if (wcsstr(text->sentence_p[t].word_p[s].entity, L"#") !=
NULL)
          text->sentence_p[t].symbol = 1;
        else if (wcsstr(text->sentence_p[t].word_p[s].entity,
L"№") != NULL)
          text->sentence_p[t].symbol = 1;
        for (int w = 0; w < text-
>sentence_p[t].word_p[s].symbols_count; w++)
          if (iswdigit(text->sentence_p[t].word_p[s].entity[w]))
            text->sentence_p[t].number = 1;
```

```
      }
      if (text->sentence_p[t].number != 1 && text-
>sentence_p[t].symbol == 1){
        for (int s = 0; s < text->sentence_p[t].words_count; s++)
{
          free(text->sentence_p[t].word_p[s].entity);
        }
        free(text->sentence_p[t].word_p);
        text->sentence_p[t].words_count = 0;
      }
    }
  }


  void annotation(){
    wprintf(L"\nНапишите \"1\", чтобы вывести текст.\n");
    wprintf(L"Напишите \"2\", чтобы распечатать все слова,
встречающиеся в тексте не более одного раза.\n");
    wprintf(L"Напишите \"3\", чтобы заменить строку вида \"<день>
<месяц> <год> г.\" на \"ДД/ММ/ГГГГ\".\n");
    wprintf(L"Напишите \"4\", чтобы отсортировать предложения в
порядке неубывания по произведению длин содержащихся в них слов.\
n");
    wprintf(L"Напишите \"5\", чтобы удалить все предложения,
содержащие \'#\' или \'№\', но не числа.\n");
    wprintf(L"Напишите \"0\", чтобы прекратить выполнение
программы.\n");
  }
```

Makefile:

```
main: main.o input.o handling.o
     gcc -o main main.o input.o handling.o
main.o: main.c input.h handling.h structs.h
     gcc -c main.c
handling.o: handling.h handling.c structs.h
```

```
	gcc -c handling.c
input.o: input.h input.c structs.h
	gcc -c input.c
clean:
	rm -rf *.o
```