**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

Курсовая работа

**по дисциплине «Программирование»**

Тема: Обработка текстовой информации

|  |  |  |
| --- | --- | --- |
| Студент гр. 2308 |  | Попов Н.А. |
| Преподаватель |  | Подклётнов С.Г. |

Санкт-Петербург

2022

**Введение**

Целью работы является поступательное и поэтапное решение поставленной задачи на основе изученного за 1 семестр материала и полное оформление его этапов, зависящих от специфики языка:

* *Правильная постановка задачи*
* *Специфика операционных систем*
* *Специфика структур данных*
* *Правильная постановка алгоритма*
* *Точная программная реализация*
* *Идеальная отладка программы*

***Задание (вариант 87)***

Задан текст, содержащий произвольное количество строк, в которых отдельные слова могут разделяться одним или несколькими пробелами и знаками пунктуации (перенос слов с одной строки на другую не используется). Сформировать новый текст, который является результатом следующего преобразования исходного текста: заменить заданное слово в каждой строке на другое слово, если оно встречается в первой строке текста.

**Постановка задачи и описание решения**

Данная программа имеет следующие основные требования:

* Систематическое использование функций как средства абстракции и структуризации;
* Способ передачи данных между функциями – через параметры;
* Использование рекомендуемых правил оформления кода;
* Наличие комментариев для ключевых фрагментов программы, заголовочных комментариев функций (внешние спецификации);
* Сочетание консоли и файлов для организации ввода исходных данных и вывода результатов;
* Обоснование проектных решений, в т.ч. с точки зрения эффективности разработанных алгоритмов;
* Формирование в ходе работы программы выходных данных, демонстрирующих как результат её работы, так и порождаемый ею вычислительный процесс.

А также следующие дополнительные:

* Дружелюбный интерфейс (диалог с пользователем)
* Меню для выбора вариантов работы
* Очистка экрана после завершения каждого варианта работы
* Учет особенностей различных операционных систем
* Комментарии в коде — на английском
* Все переменные должны быть описаны до первого оператора в функции
* Инициализация при описании допускается только для указателей (начальная установка в NULL).

**Описание решения**

**Теория**

Для написания данной курсовой работу нужно основательно изучить одну из важнейших и сложных тем в Си – строки. Строка – это последовательность ASCII или UNICODE символов. Строки в Си, как и в большинстве языков программирования высокого уровня рассматриваются как отдельный тип. Так как язык Си по своему происхождению является языком системного программирования, то строковый тип данных в Си отсутствует, следовательно в качестве строк в Си используются обычные массивы символов.

Существуют всего два представления формата строк:

* Формат ANSI
* Строки с завершающим нулем (используется в C)
  + *В строках с завершающим нулем, значащие символы строки указываются с первой позиции, а признаком завершения строки является значение ноль*

## Объявление строк в C

Строки реализуются посредством массивов символов, а это значит, что объявление строки в Си имеет тот же синтаксис, что и объявление одномерного символьного массива: **char имя[длина].**

## ****Работа со строками в С****

Так как строки на языке С являются массивами символов, то к любому символу строки можно обратиться по его индексу. Для этого используется синтаксис обращения к элементу массива, поэтому первый символ в строке имеет индекс ноль.

## Массивы строк в С

Объявление массивов строк в языке С также возможно. Для этого используются двумерные массивы символов, что имеет следующий синтаксис:

**char имя[количество][длина];**

## Функции для работы со строками в С

Все функции, предназначенные для работы со строками, можно разделить на три группы:

* Ввод и вывод строк
* Преобразование строк
* Обработка строк

Для ввода и вывода строковой информации можно использовать функции форматированного ввода и вывода **(printf и scanf)**. Для этого в строке формата при вводе или выводе строковой переменной необходимо указать спецификатор типа %s. А также для ввода и вывода строк в библиотеке *<stdio.h>* содержатся специализированные функции **gets** и **puts**.

Функция **gets** предназначена для ввода строк, но использовать эту функцию не рекомендуется - она не контролирует выход за границу строки, а это может произвести к ошибкам. Вместо нее используется функция **fgets** с тремя параметрами: буфер, имя строки, формат ввода. А в свою очередь функция puts предназначена для вывода строк.

## Обработка строк

Также важным разделом является обработка строк. В библиотеке *<string.h>* содержаться функции для различных действий над строками:

* *Функция вычисления длины строки*
* *Функции копирования строк*
* *Функции сравнения строк*
* *Функции объединения (конкатенации) строк*
* *Функции поиска символа в строке*
* *Функция поиска строки в строке*
* *Функция поиска первого символа в строке из заданного набора символов*
* *Функции поиска первого символа в строке не принадлежащему заданному набору символов*
* *Функции поиска первого символа в строке из заданного набора символов*
* *Функция поиска следующего литерала в строке*

***Не менее важной темой будет и работа с файлами***

**Файл** - именованная область внешней памяти, выделенная для хранения массива данных.

Открытие файла в Си осуществляется с помощью функции **fopen()**, которая возвращает указатель на структуру типа FILE.

Функция **fclose()** закрывает поток или потоки, связанные с открытыми при помощи функции **fopen()** файлами.

**Чтение символа из файла**:

**fgetc(поток)** - Аргументом функции является указатель на поток типа **FILE**. Функция возвращает код считанного символа. Если достигнут конец файла или возникла ошибка, возвращается константа EOF.

**Запись символа в файл**:

**fputc(символ, поток)** - Аргументами функции являются символ и указатель на поток типа **FILE**. Функция возвращает код считанного символа.

Функции **fscanf()** и **fprintf()** аналогичны функциям **scanf()** и **printf()**, но работают с файлами данных, и имеют первый аргумент — указатель на файл. Функции **fgets()** и **fputs()** предназначены для ввода-вывода строк, они являются аналогами функций **gets()** и **puts()** для работы с файлами.

Символы в Си читаются из потока до тех пор, пока не будет прочитан символ новой строки **‘\n’**, который включается в строку, или пока не наступит конец потока EOF или не будет прочитано максимальное количество символов. Результат помещается в указатель на строку и заканчивается нуль- символом **‘\0’**. Функция возвращает адрес строки.

***Кроме того, стоит упомянуть про обработку исключений***

***Исключение*** — это событие при выполнении программы, которое приводит к её ненормальному или неправильному поведению.

Существует два вида исключений:

* ***Аппаратные***, которые генерируются процессором. К ним относятся, например
  + деление на 0;
  + выход за границы массива;
  + обращение к невыделенной памяти;
  + переполнение разрядной сетки.
* ***Программные***, генерируемые операционной системой и прикладными программами – возникают тогда, когда программа их явно инициирует. Когда встречается аномальная ситуация, та часть программы, которая ее обнаружила, может сгенерировать, или возбудить, исключение.

Фундаментальная идея обработки исключительных ситуаций состоит в том, что функция, обнаружившая проблему, но не знающая как её решить, генерирует исключение в надежде, что вызвавшая её функция сможет решить возникшую проблему. Функция, которая может решать проблемы данного типа, указывает, что она перехватывает такие исключения.

***В своей работе я использовал самую простую форму исключений - perror()***

***Изучив все темы предыдущих лабораторных работ и основательно подготовившись по вышеперечисленным темам, можно приступать к написанию кода.***

**Разбор кода программы**

***Для начала разберём все темы использованные в программе***

**Ввод и вывод** информации осуществляется через функции стандартной библиотеки. Прототипы рассматриваемых функций находятся в файле stdio.h. Эта библиотека содержит функции

* printf() — для вывода информации
* scanf() — для ввода информации.

## Вывод информации

Функция printf() предназначена для форматированного вывода. Она переводит данные в символьное представление и выводит полученные изображения символов на экран. При этом имеется возможность форматировать данные, то есть влиять на их представление  
на экране.

***Существуют управляющие символы***, которые не выводятся на экран, а управляют расположением выводимых символов. Отличительной чертой управляющего символа является наличие обратного слэша ‘\’ перед ним.

***Существуют и форматы. Они*** нужны для того, чтобы указывать вид, в котором информация будет выведена на экран. Отличительной чертой формата является наличие символа процент ‘%’ перед ним

## Ввод информации

Функция форматированного ввода данных с клавиатуры scanf() выполняет чтение данных, вводимых с клавиатуры, преобразует их во внутренний формат и передает вызывающей функции. При этом задаются правила интерпретации входных данных с помощью спецификаций форматной строки.

## Условный оператор if

Условный оператор if может использоваться в форме ***полной*** или ***неполной*** развилки.

1. if (Условие)  
   {  
     БлокОпераций1;  
   }
2. if (Условие)  
   {  
     БлокОпераций1;  
   }  
   else  
   {  
     БлокОпераций2;  
   }

## *Логические операции в условных операторах*

Условный оператор может проверять

* одновременное выполнение всех условий (операция И - &&)
* выполнение хотя бы одного из условий (операция ИЛИ - ||)
* выполнение только одного из условий (операция исключающее ИЛИ - ^)

**Операторы цикла в языке Си**

***Циклом*** называется блок кода, который для решения задачи требуется повторить несколько раз.  
  
Каждый цикл состоит из

* блока проверки условия повторения цикла
* тела цикла

*Цикл выполняется до тех пор, пока блок проверки условия возвращает истинное значение.*

Тело цикла содержит последовательность операций, которая выполняется в случае истинного условия повторения цикла. После выполнения последней операции тела цикла снова выполняется операция проверки условия повторения цикла. Если это условие не выполняется, то будет выполнена операция, стоящая непосредственно после цикла в коде программы.

**В языке Си следующие виды циклов:**

* while — цикл с предусловием;
* do…while — цикл с постусловием;
* for — параметрический цикл (цикл с заданным числом повторений).

*В своей программе я использовал лишь for()*

## Параметрический цикл for

for — параметрический цикл (цикл с фиксированным числом повторений). Для организации такого цикла необходимо осуществить три операции:

* **Инициализация** - присваивание параметру цикла начального значения;
* **Условие** - проверка условия повторения цикла, чаще всего - сравнение величины параметра с некоторым граничным значением;
* **Модификация** - изменение значения параметра для следующего прохождения тела цикла.
* Эти три операции записываются в скобках и разделяются точкой с запятой
* Как правило, параметром цикла является целочисленная переменная.

**Функции**

***Функция*** — это самостоятельная единица программы, которая спроектирована для реализации конкретной подзадачи.  
Функция является подпрограммой, которая может содержаться в основной программе, а может быть создана отдельно (в библиотеке). Каждая функция выполняет в программе определенные действия.

## Определение функции

Каждая функция в языке Си должна быть определена, то есть должны быть указаны:

* тип возвращаемого значения;
* имя функции;
* информация о формальных аргументах;
* тело функции.
* Различают ***системные*** (в составе систем программирования) и ***собственные*** функции.

## Вызов функции

Общий вид вызова функции:

Переменная = ИмяФункции(СписокФактическихАргументов);

## Возврат в вызывающую функцию

По окончании выполнения вызываемой функции осуществляется возврат значения в точку ее вызова. Это значение присваивается переменной, тип которой должен соответствовать типу возвращаемого значения функции. Функция может передать в вызывающую программу только одно значение. Для передачи возвращаемого значения в вызывающую функцию используется оператор return

**Теперь рассмотрим содержание самой программы**

1. **Подключение библиотек:**

Первым делом нужно подключить стандартную библиотеку ввода/вывода для осуществления взаимодействия программы с пользователем. После этого нужно обратиться к языковой библиотеке, чтобы в последующем подключить русский язык для оформления “интерфейса” программы. В конце подключена библиотека для работы со строками, в частности для вычисления их длин.

1. **Главная функция (интерактивное меню):**

Подключаем русский язык, а далее инициализируем переменную, которая будет переключателем в нашем интерактивном меню. Создаём строку со словом меню, с целью центрирования её относительно следующей строки. Задаём три возможных действия: запуск программы, завершение работы программы, и ознакомление с основной ей функцией.

Сканируем значение переключателя и далее, используя конструкции ветвления, добавляем выбор режима работы (замена конструкции switch). Если значение равно 1 – вызываем функцию mod\_1, которая отвечает за основной блок запуска. Значение 2 завершает программу, а 3 запускает mod\_3 -ознакомительный текстовый блок. Пользователь, выбрав режим, должен будет только ждать следующего появления меню, для выбора дальнейших действий программы.

1. **Функция mod\_1:**

Создаём массив **usd** – массив используемых строк. После выводим первое сообщение, чтобы пользователь понимал для чего создана программа. Искомое слово обозначаем за переменную **first** (массив **search)**, считываем её и сразу же выводим второе сообщение, которое обозначит следующую переменную как ту, на которую нужно заменить – **second** (массив **replace**).

Далее для создания дальнейших циклов введём некоторое количество переменных-счётчиков, а также не забываем про основной массив-строку – **string**. Двумя функциями **snprintf** распределяем введённые переменные по массивам (Функция **snprintf()** идентична **printf()**, за исключением того, что вывод производится в массив, указанный аргументом buf).

Далее открываем два файла. Первый (read.txt) – будет открыт на чтение, т.к. нам нужно будет записывать в него изначальный текст, и второй (final.txt) – открыт на запись, потому что это будет финальная версия введённого текста.

Далее используя цикл, позволяющий начать взаимодействие с нашими файлами, начинаем обрабатывать изначальный массив строк, до момента возникновения NULL (как только наткнёмся на конец файла).

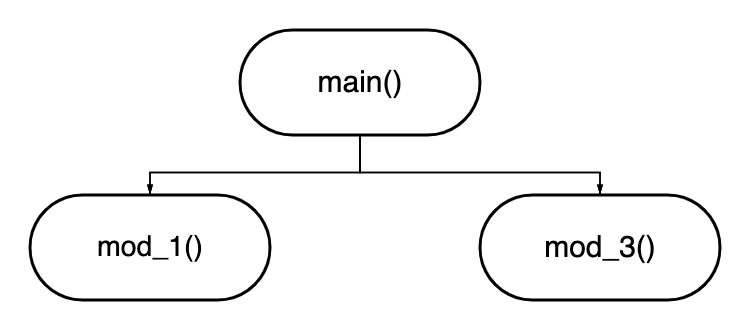
Увеличиваем несколько счётчиков для дальнейших вычислений и циклом **for**, пока не наступит конец строки, считываем пробелы – если его находим меняем на нуль-символ (из-за особенности операционной системы), в противном же случае мы записываем символ в двумерный массив строк. Как только мы закончили со строкой переводим её на следующую и начинаем запись в финальный файл.

Сравниваем длину слова, записанного в строке, и слова, введённого нами ранее, параллельно фиксируя пробелы, для последующего их вывода. Если слова совпадают, снова проверяем счётчик на наличие пробелов и записываем заменённое слово на место заменяемого, с пробелом или без соответственно. В противном случае записываем слова как они были изначально. После всех действий возвращаем меню и ждём дальнейших действий от пользователя.

1. **Функция mod\_3:**

Функция не несёт в себе каких-либо значимых операций и значений. Она служит текстовым блоком и справочником для пользователей, которые не знают или не понимают принцип работы программы. Здесь подробно описано как использовать меню и взаимодействовать с самим кодом.

**Структура вызовов функций**



**Описание переменных**

**Функция main()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | click | int | Переключатель для выбора работы (триггер) |
| 2 | men | char | Меню |

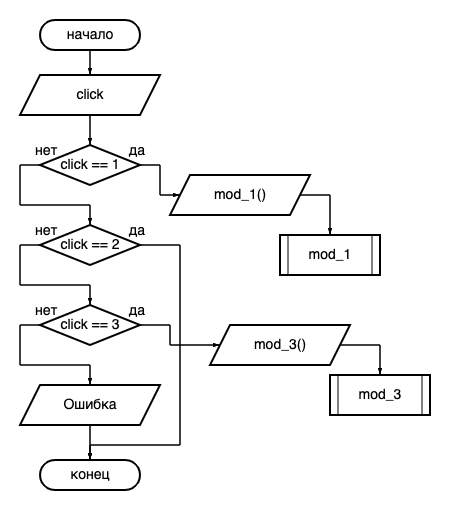
**Функция mod\_1()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | usd | char | Массив используемых строк |
| 2 | first | char | Первое введённое слово / заменяемое слово |
| 3 | search | char | Массив с первым словом |
| 4 | second | char | Второе введённое слово / замена |
| 5 | replace | char | Массив со вторым словом |
| 6 | string | char | Изначальный / Конечный массив строк |
| 7 | n | int | Служебная переменная -счётчик |
| 8 | m | int | Служебная переменная -счётчик |
| 9 | str | int | Служебная переменная -счётчик |
| 10 | i | int | Служебная переменная -счётчик |
| 11 | j | int | Служебная переменная - флаг |

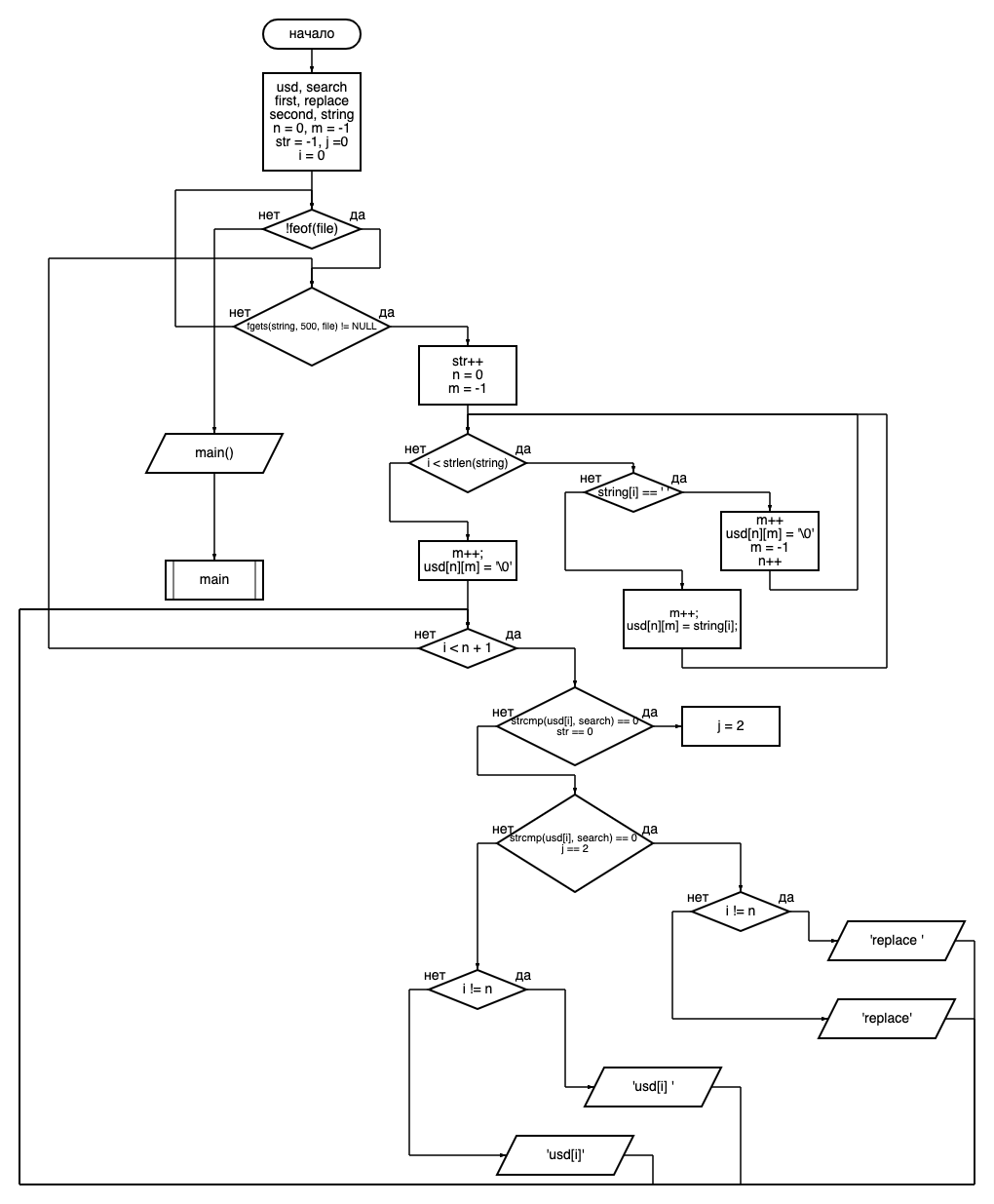
**В функции mod\_3() переменные отсутствуют**

**Схемы алгоритмов**

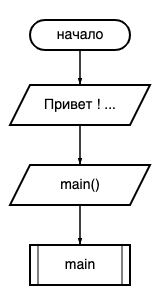
**Функция main()**

****

**Функция mod\_1()**

****

**Функция mod\_3()**

****

**Контрольные примеры**

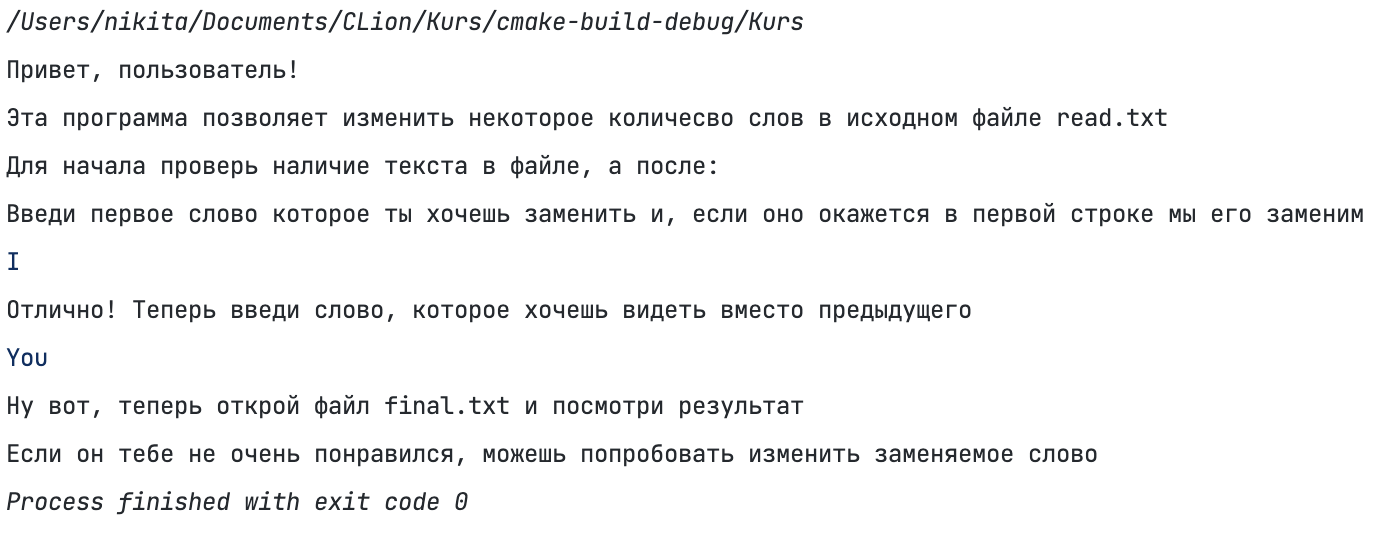
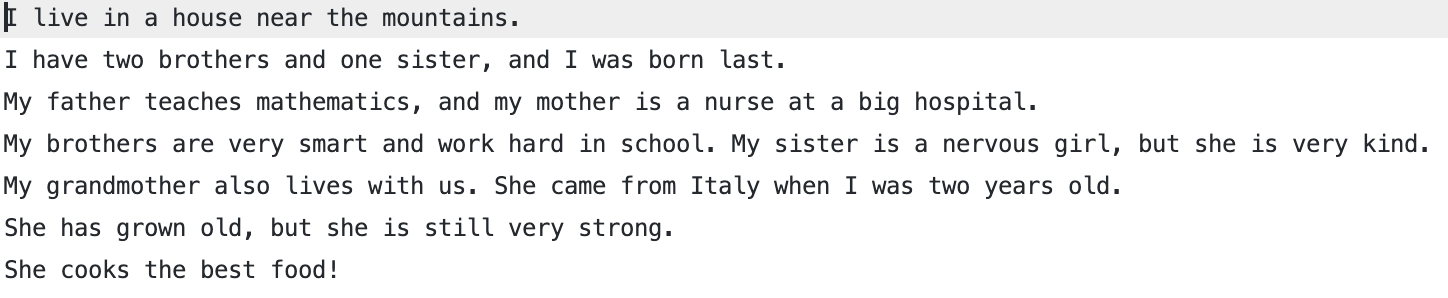
|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| I live in a house near the mountains. I have two brothers and one sister, and I was born last. My father teaches mathematics, and my mother is a nurse at a big hospital. My brothers are very smart and work hard in school. My sister is a nervous girl, but she is very kind. My grandmother also lives with us. She came from Italy when I was two years old. She has grown old, but she is still very strong. She cooks the best food! | You live in a house near the mountains. You have two brothers and one sister, and You was born last. My father teaches mathematics, and my mother is a nurse at a big hospital. My brothers are very smart and work hard in school. My sister is a nervous girl, but she is very kind. My grandmother also lives with us. She came from Italy when You was two years old. She has grown old, but she is still very strong. She cooks the best food! |
| Doctor Klein: Good morning, Cecilia, how are you feeling today? Cecilia: I do not feel very well, Doctor Klein. I hope that you can treat my illness. Doctor Klein: I’m sorry that you feel very sick. Tell me some of your symptoms so that I can give you a proper diagnosis. Cecilia: I have not felt well since yesterday afternoon. The symptoms began with a headache and a gradual sore throat.  I drank tea with lemon and honey, and I went to bed early.  However, I am extremely exhausted, and I don’t feel any better today. Doctor Klein: I have seen these symptoms recently in some of my other patients.  I’ll check your temperature and examine your throat in order to give you a proper diagnosis. Cecilia: Thank you, Doctor. | Doctor Klein: Good morning, Max , how are you feeling today? Max : I do not feel very well, Doctor Klein. I hope that you can treat my illness. Doctor Klein: I’m sorry that you feel very sick. Tell me some of your symptoms so that I can give you a proper diagnosis. Max : I have not felt well since yesterday afternoon. The symptoms began with a headache and a gradual sore throat. I drank tea with lemon and honey, and I went to bed early. However, I am extremely exhausted, and I don’t feel any better today. Doctor Klein: I have seen these symptoms recently in some of my other patients. I’ll check your temperature and examine your throat in order to give you a proper diagnosis. Max : Thank you, Doctor. |
| Margaret was a small girl. She was a little girl. All her friends were taller than her. She was shorter than all her friends. She wanted to be tall. Her mom told her not to worry. One day Margaret would be tall. One day she would be taller than her friends. One day all her friends would be shorter than her. She was happy to hear that. She only had one question for her mom. When would she be taller than her friends? Would it be next year? She hoped it would be next year. She was tired of being the shortest girl. | Margaret was a small girl. She was a little girl. All her friends was taller than her. She was shorter than all her friends. She wanted to be tall. Her mom told her not to worry. One day Margaret would be tall. One day she would be taller than her friends. One day all her friends would be shorter than her. She was happy to hear that. She only had one question for her mom. When would she be taller than her friends? Would it be next year? She hoped it would be next year. She was tired of being the shortest girl. |

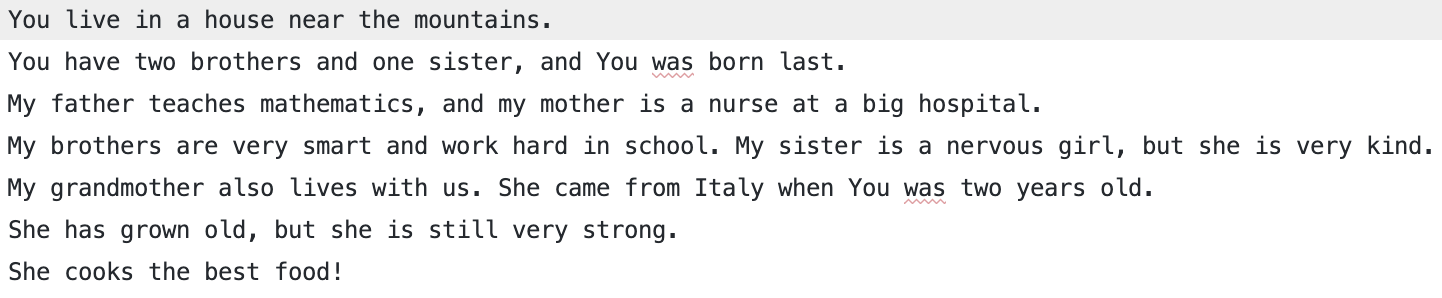
**Текст программы**

#include <stdio.h> // Connecting libraries  
#include <locale.h>  
#include <string.h>  
  
int mod\_1();  
int mod\_3();  
  
int main(){  
 setlocale(**LC\_ALL**, "Rus"); // Сonnecting the Russian language  
 int click;  
 char men[10] = "МЕНЮ";  
 printf("\t%55s\n", men);  
 printf("Нажмите 1 для запуска программы | ");  
 printf("Нажмите 2 для выхода из программы | ");  
 puts("Нажмите 3 для ознакомления с программой");  
 scanf("%d", &click);  
 if (click == 1 ){  
 mod\_1();  
 }  
 if(click == 2){  
 printf("До новых встреч !");  
 return 0;  
 }  
 if (click == 3){  
 mod\_3();  
 }  
 if (click != 1 && click != 2 && click != 3){  
 printf("Ошибка, введи другое число !");  
 }  
}  
  
int mod\_1(){  
 char usd[50][50]; // Massive of used strings  
  
 puts("Привет, пользователь!\n"  
 "Эта программа позволяет изменить некоторое количесво слов в исходном файле read.txt\n"  
 "Для начала проверь наличие текста в файле, а после:\n"  
 "Введи первое слово которое ты хочешь заменить и, если оно окажется в первой строке, мы его заменим");  
 char search[10]; // The search word  
 char first[10];  
 scanf("%s", first);  
  
  
 puts("Отлично! Теперь введи слово, которое хочешь видеть вместо предыдущего");  
 char replace[10]; // Replacement word  
 char second[10];  
 scanf("%s", second);  
  
 printf("Ну вот, теперь открой файл final.txt и посмотри результат\n"  
 "Если он тебе не очень понравился, можешь попробовать изменить заменяемое слово\n");  
  
 char string[500]; // Our first/final string  
  
 int n = 0, m = -1; // Counters  
 int str = -1;  
 int j = 0, i = 0;  
  
 **snprintf**(search, 10, "%s", first); // Scanning first and second words  
 **snprintf**(replace, 10, "%s", second);  
  
 FILE \*file = fopen("/Users/nikita/Documents/CLion/Kurs/read.txt", "r"); // Opening files  
 FILE \*file1 = fopen("/Users/nikita/Documents/CLion/Kurs/final.txt", "w");  
  
 while (!feof(file)) { // Scanning, writing and reading files  
 if (fgets(string, 500, file) != **NULL**) {  
 str++;  
 n = 0;  
 m = -1;  
 for (i = 0; i < strlen(string); i++) {  
 if (string[i] == ' ') {  
 m++;  
 usd[n][m] = '\0';  
 m = -1;  
 n++;  
 } else {  
 m++;  
 usd[n][m] = string[i];  
 }  
 }  
 m++;  
 usd[n][m] = '\0';  
 for (i = 0; i < n + 1; i++) { // Writing final text  
 if (strcmp(usd[i], search) == 0 && str == 0){  
 j = 2;  
 }  
 if (strcmp(usd[i], search) == 0 && j == 2){  
 if (i != n){ // checking spaces  
 fprintf(file1, "%s ", replace);  
 }  
 else {  
 fprintf(file1, "%s", replace);  
 }  
 }  
 else{  
 if (i != n){ // checking spaces  
 fprintf(file1, "%s ", usd[i]);  
 }  
 else {  
 fprintf(file1, "%s", usd[i]);  
 }  
 }  
 }  
 }  
 else {  
 perror("Ошибка ! Текст не найден");  
 }  
 }  
 fclose(file); // Closing files  
 fclose(file1);  
 return main();  
}  
  
int mod\_3(){ // manual  
 puts("Привет !");  
 printf("Эта программа позволяет изменить некоторое количесво слов в исходном файле read.txt\n"  
 "В этом файле содержится небольшое количество текста, который, в свою очередь, будет пересобран в final.txt\n"  
 "Пользователь должен ввести слово, которое он хочет заменить во всём тексте\n"  
 "Если оно содержится в первой строке этого текста, то указанное слово будет заменено на другое,"  
 "введённое следом\n"  
 "Если же такое слово не было найдено, будет возвращён исходный текст\n"  
 "А теперь, когда ты ознакомился с программой, запусти и посмотри на неё в действии !\n");  
 return main();  
}

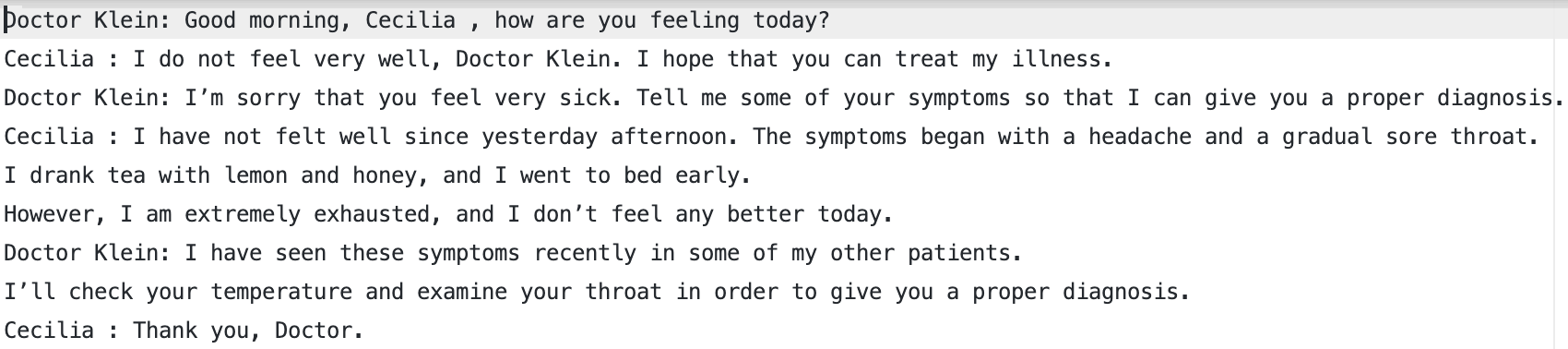
**Примеры выполнения программы**

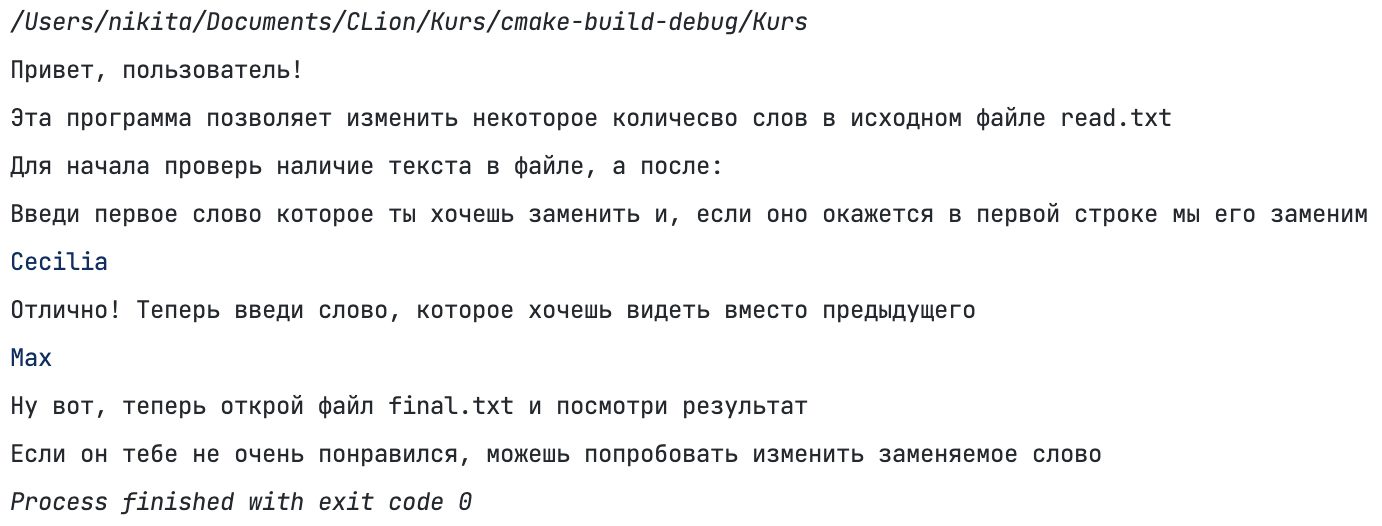
**Пример 1**

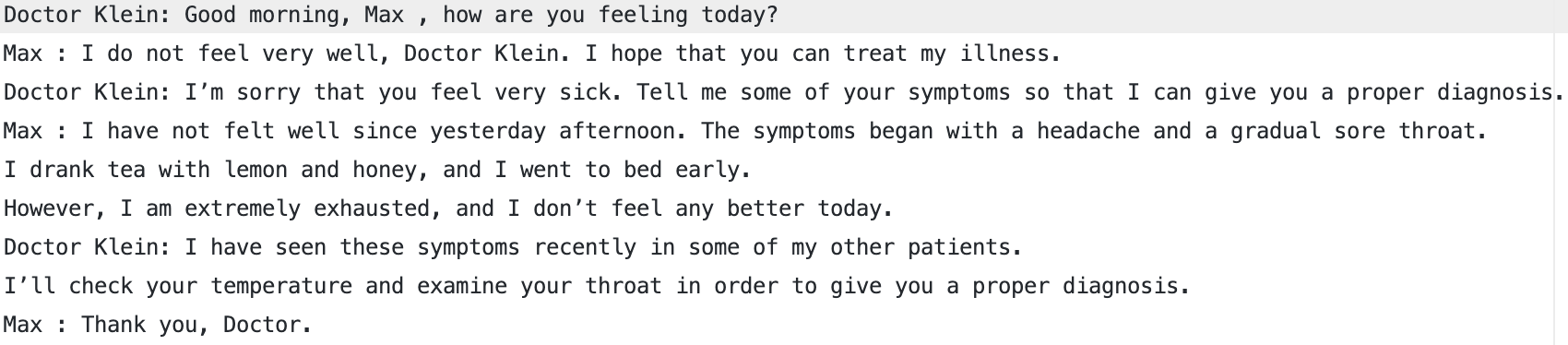
****

****

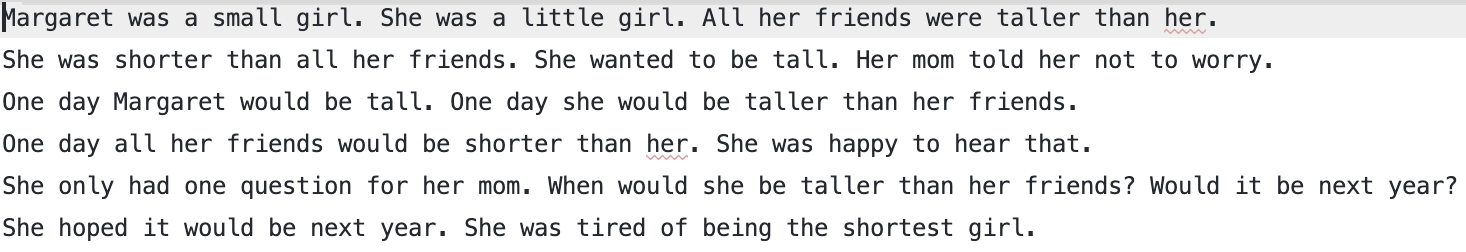
**Пример 2**

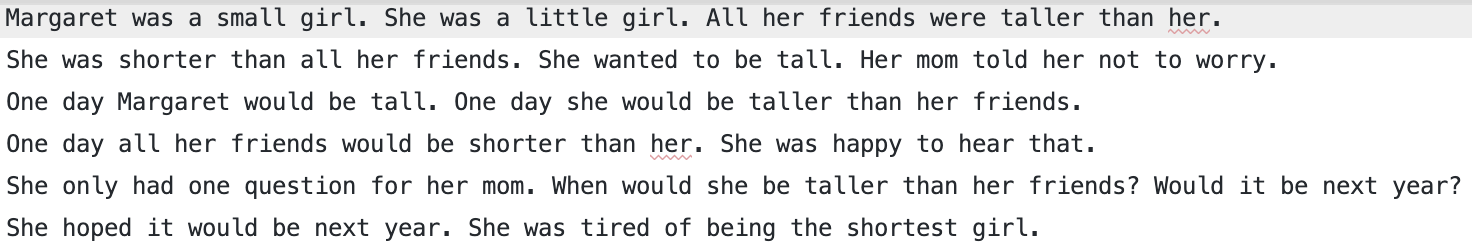


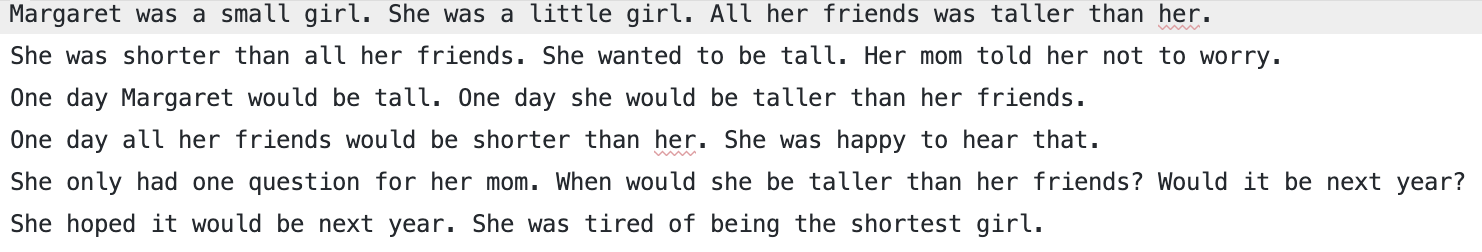




**Пример 3**







**Заключение**

**Заголовочные файлы**

Первым заголовочным файлом стандартной библиотеки является **<stdio.h>**:

Служит для обеспечения выполнения таких операций как ввод/вывод. Из остальных возможных функций данной библиотеки были использованы операции закрытия файлов, их открытия, а также основные функции ввода текста и вывода его в консоль. Кроме того, при помощи этого заголовочного файла стало возможным использовать сканирование строк установленного размера и осуществлять любое взаимодействие с файлами. Нельзя оставить в стороне и последнюю из использованных функций ***fgets()***,которая считывает символы из текущей позиции потока в и включая первый символ новой строки, в конец потока.

Следующим стандартным заголовочным файлом я обозначил **<locale.h>**:

Данный заголовочный файл используется для задач, связанных с локализацией.

В нашей же программе, при помощи этой библиотеки мы подключили русскоязычный интерфейс, с целью осуществления взаимодействия вычислительной машины с пользователем.

Последним заголовочным файлом, которым я воспользовался был **<string.h>**:

Данная библиотека определяет несколько функций для обработки Cи-строк и массивов. В моей программе возникали случаи применения функции ***strlen(),*** вычисляющей длину указанной строки и ***strcmp(),*** которая сравнивает две строки между собой.

**Вывод**

Выполнив данную курсовую работу, я отточил навыки, которые получил за 1 семестр обучения. Работа позволила более тонко изучить язык Си, в частности, особенности его работы на разных операционных системах, особенности возникновения ошибок. Поэтапно выполняя задачи, поставленные данной курсовой работой, я повторил весь пройденный материал, а также изучил некоторые новые функции. Кроме того, я открыл для себя новые возможности Си и основательно закрепил весь пройденный и изученный мной материал.