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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа с текстом

Студент гр. 9383 _____ Ноздрин В.Я.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Ноздрин В.Я.

Группа 9383

Тема работы : Работа с текстом

Исходные данные:

Язык программированя Си, сборка программ под Linux

Содержание пояснительной записки:

«Содержание», «Введение», «Задание», «Файлы программы», «Функции программы», «Примеры работы программы», «Заключение».

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 16.10.2019

Дата сдачи реферата: 14.12.2019

Дата защиты реферата: 14.12.2019

Студент _____ Ноздрин В.Я.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В данной курсовой работе была написана программа, которая получает на вход текст из файла. После считывания программа удаляет повторно встречающиеся предложения. Дальше программа принимает пользовательский ввод и выполняет одно из возможных действий с текстом:

- Вывод всех предложений-анаграм в тексте.
- Сортировка предложений текста по количеству загланых букв в них.
- Замена всех гласных букв в тексте на две следующие буквы по алфавиту.
- Заменить слово, введенное пользователем, на другое слово, введенное пользователем, во всем тексте.
- Печать текста
- Выход из программы
- Печать подсказки

SUMMARY

This project is a program that reads text from file and make some changes that depends on user decision. Program deletes every sentence that repeats any other sentence of the text. There is the list of features:

- Print all anagrams in text.
- Sort sentences in text by number of capital letters in them.
- Replace every vowel with two following letters of alphabet.
- Standard find-and-replace function.
- Show text on screen.
- Exit program.
- Show tip.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы:

Реализовать программу на языке программирования Си. Программа должна выполнять преобразования с введенным текстом в зависимости от пользовательского ввода.

Для написания программы необходимо:

1. Изучить стандартную библиотеку языка Си.
2. Изучить инструменты работы с динамической памятью.
3. Изучить библиотеку wchar.h.
4. Изучить утилиту Make.
5. Изучить структуры в Си.
6. Созданать функций для ввода и вывода текста.
7. Разделенить программу на файлы и создать заголовочные файлы.
8. Провести тестирование программы.

1. ЗАДАНИЕ

Вариант 1

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. Вывести все предложения, которые являются анаграммами друг для друга. Учитывать надо только буквы и цифры.
2. Отсортировать предложения (фактически, массив структур) по количеству заглавных букв в предложении.
3. Заменить каждую гласную буквы двумя другими буквами идущими следующими по алфавиту. Например, “ясЕнь” должно быть преобразовано в “абсЁЖнь”.
4. Заменить все вхождения одного слова (заданного пользователем) на другое слово (заданного пользователем).

Все сортировки и операции со строками должны осуществляться с использованием функций стандартной библиотеки.

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. ИСХОДНЫЙ КОД

2.1. Файл main.c

Файл main.c является основным файлом программы. В main.c вводятся данные, и выполняется обработка текста.

2.2. Файлы text.h и text.c

В файлах text.h и text.c реализованы ввод и вывод данных.

В этих файлах определены структуры Sentence_t и Text_t, реализованы функции чтения read_sentence () и read_text (), функции печати print_sentence () и print_text (), функция сравнения предложений.

2.3. Файлы text_func.h и text_func.c

В файлах text_func.h и text_func.c реализованы функции обработки теста .

В этих файлах реализованы следующие функции: delete_duplicates () - функция, удаляющая повторяющиеся предложения из текста; print_anagrams () - функция, выводящая анаграммы в тексте; cap_sort () - функция, соритрующая текст по количеству загавынх слов в предолжениях; vowels_shift () - функция, заменяющая гласные буквы соответственно заданию; snt_vowels_shift () подфункция предыдущей функциии для одного предложения; snt_find_and_replace () - find-and-replace функция; check_anagram () - функция, проверяющая, являются ли предложения анаграмами; cap_cmp () - функция, сравнивающая предложения по количеству загавных букв.

2.4. Файл Makefile

Makefile используется утилитой make для компиляции всей программы.

3. ФУНКЦИИ

3.1. Функции read_sentence () и read_text ()

Функции принимают поток и считывают из него предложение или текст соответственно. Считывание предложения заканчивается символом перевода строки, точкой, или концом файла. Считывание текста заканчивается при считывании пустого предложения. Считывание предложений производится посимвольно, а считывание текст – есть считвывание предложений в цикле. Параллельно со считыванием заполняются поля создаваемых структур “текст” и “предложение”.

3.2. Функции print_sentence () и print_text ()

Выводят на экран предложение и текст соответственно.

3.3. Функции обработки

Функции реализованы с использованием стандартных библиотек и алгоритмы их действия не нуждаются в подробном описании.

3.4. Функция main()

Функция main, реализует пользовательский интерфейс и соответственно пользовательскому вводу вызывает соответствующие функции обработки теста.

4. ТЕСТИРОВАНИЕ ПРОГРАММЫ

Файл input_text.txt:

Она раздавлена — все Больн1о.1Она разДАвлена — все больно. 2Она РАздаВЛена — все больно. Она раз1давлена — все Больно.

Входные данные:

input_text.txt

1

2

5

3

5

Выходные данные:

Печать текста:

Она раздавлена — все Больн1о.1Она разДАвлена — все больно. 2Она РАздаВЛена — все больно. Она раз1давлена — все Больно.

Анаграммы:

[0|0]:Она раздавлена — все Больн1о.

[0|3]: Она раз1давлена — все Больно.

Сортировка:

2Она РАздаВЛена — все больно.1Она разДАвлена — все больно.Она раздавлена — все Больн1о. Она раз1давлена — все Больно.

Замена гласных:

2ПРнбв РБВздбвВЛжзнбв — всжз бпрльнпр.1ПРнбв рбвзДБВвлжзнбв — всжз бпрльнпр.ПРнбв рбвздбввлжзнбв — всжз Бпрльн1пр. ПРнбв рбвз1дбввлжзнбв — всжз Бпрльнпр.

Обработка выполнена корректна.

ЗАКЛЮЧЕНИЕ

Были изучены основы работы со стандартной библиотекой языка программирования Си. Релизованы функции согласно условию задания. Функции были разделены на несколько файлов для удобной работы с ними. Был написан Makefile для быстрой компиляции проекта.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

1. Файл main.c:

```
##include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include <wctype.h>
#include <locale.h>
#include "text.h"
#include "text_func.h"

int main ()
{
   FILE *pFile;
   Text_t *text;
   setlocale (LC_ALL, "");

   /* READ TEXT BEGIN */
      fputws(L"Hi there!\nEnter filename with your text (not longer than 100):\n",
stdout);
   wchar_t wfile_path[100];
   char file_path[100];

   while(1) {
      fgetws(wfile_path, 100, stdin);

      if (!wcstombs(file_path, wfile_path, sizeof(file_path)))
      {
         fputws(L"Something went wrong... Try again!\n", stdout);
         continue;
      }

      for (int i = 0; file_path[i]; i++)
      {   if (file_path[i] == '\n'){
            file_path[i] = '\0';
            break;
      }}
      pFile = fopen(file_path, "r");
      if (pFile) {
         text = read_text(pFile);
         fclose(pFile);
```

```
            break;
        }
        fputws(L"Something went wrong... Try again!\n", stdout);
    }
    /* READ TEXT END */

    delete_duplicates (text);

    wchar_t *tip = L"Type number 0-5 to choose an option!\n0 : to finish the program\
n1 : show all anagrams in the text\n2 : to sort sentences of the text by the number of
capital letters on each one\n3 : to replace every vowels with two following letters of
alphabet\n4 : \"Find & Replace All\" option\n5 : to show the text\n6 : to show this tip
again\n";
    fputws(tip, stdout);

    char command;
    do
    {
        wchar_t tmp[100];
        fgetws(tmp, 100, stdin);
        command = tmp[0];
        switch (command)
        {
            case '0':
                fputws(L"Bye!\n", stdout);
                break;
            case '1':
                print_anagrams(text);
                  fputws(L"Done! Type next command to continue. (6 to show the tip)\n",
stdout);
                break;
            case '2':
                cap_sort(text);
                  fputws(L"Done! Type next command to continue. (6 to show the tip)\n",
stdout);
                break;
            case '3':
                vowels_shift(text);
                  fputws(L"Done! Type next command to continue. (6 to show the tip)\n",
stdout);
                break;
            case '4':
                fputws(L"Enter word to replace:\n", stdout);
                wchar_t old[101];
```

```
            fgetws(old, 100, stdin);
            for (int i = 0; old[i]; i++)
            {   if (old[i] == '\n'){
                    old[i] = '\0';
                    break;
            }}
            fputws(L"Enter word to replace with:\n", stdout);
            wchar_t new[101];
            fgetws(new, 100, stdin);
            for (int i = 0; new[i]; i++)
            {   if (new[i] == '\n'){
                    new[i] = '\0';
                    break;
            }}
            for (int i = 0; i < text->len; i++)
            {
               snt_find_and_replace(text->sentences[i], old, new);
            }
              fputws(L"Done! Type next command to continue. (6 to show the tip)\n",
stdout);
            break;
        case '5':
            print_text(*text);
              fputws(L"Done! Type next command to continue. (6 to show the tip)\n",
stdout);
            break;
        case '6':
            fputws(tip, stdout);
            break;
        default:
             fputws(L"Oops, you look like you misspelled something! Try again! Type
6 to show the tip\n", stdout);
            break;
      }
   } while (command != '0');
   for (int i = 0; i < text->len; i++)
   {
      free ((text->sentences[i])->str);
      free (text->sentences[i]);
   }
   free(text->sentences);
   free(text);
   return 0;
}
```

2. Файл text.h:

```
#ifndef TEXT_H
#define TEXT_H


/* includes */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include <wctype.h>
#include <locale.h>


/* define structures */
typedef struct
{
   wchar_t *str;
   int len, size;
   int cap_c;
} Sentence_t;

typedef struct
{
   Sentence_t **sentences;
   int len, size;
} Text_t;


/* read functions */
Sentence_t *read_sentence (FILE * stream);
Text_t *read_text (FILE * stream);

/* print functions */
void print_sentence (Sentence_t snt);
void print_text (Text_t text);

/* comparator functions */
int snt_str_cmp (void* snt1, void* snt2);

#endif

}
```

3. Файл text.c:

```
#include "text.h"


Sentence_t *read_sentence (FILE * stream)
{
   Sentence_t *sent = realloc (NULL, sizeof(Sentence_t));
   sent->str = NULL; // wchar_t
   sent->len = 0; // int
   sent->size = 0; // int
   sent->cap_c = 0; // int
   wchar_t wc;
   do
   {
      wc = getwc (stream);
      if (sent-> len == 0 && wc == WEOF)
         return NULL;
      if (sent->len + 4 >= sent->size)
      {
         sent->size += 64;
         wchar_t *tmp = (wchar_t*)realloc (sent->str, sent->size * sizeof(wchar_t));
         if (!tmp)
         {
            if (sent->len == 0)
               return NULL;
            else
            {
               sent->str[sent->len] = L'\0';
               return sent;
            }
         }
         sent->str = tmp;
      }
      if (wc)
      {
         if (iswupper (wc))
            sent->cap_c++;
         sent->str[sent->len++] = wc;
      }
   } while (wc != L'.' && wc != L'\n' && wc != WEOF);
   sent->str[sent->len] = L'\0';
   return sent;
}
```

```
Text_t *read_text (FILE * stream)
{
   Text_t *text = realloc(NULL, sizeof(Text_t));
   text->sentences = NULL; // Sentence**
   text->len = 0; // int
   text-> size = 0; // int
   Sentence_t *snt;
   while ((snt = read_sentence (stream)))
   {
      if (text->len + 4 >= text->size)
      {
         text->size += 256;
              Sentence_t **tmp = (Sentence_t**)realloc (text->sentences, text->size *
sizeof(Sentence_t*));
         if (!tmp)
            return text;
         text->sentences = tmp;
      }
      text->sentences[text->len++] = snt;
   }
   return text;
}

void print_sentence (Sentence_t snt)
{
   wprintf (L"%ls", snt.str);
}

void print_text (Text_t text)
{
   wprintf(L"~~~~~~~Text~~~~~~~\n");
   for(int i = 0; i < text.len; i++)
      wprintf(L"%ls", text.sentences[i]->str);
   wprintf(L"\n~~~~~~~~~~~~~~~~~~\n");
}

int snt_str_cmp (void* snt1, void* snt2)
{
   wchar_t *p1 = (wchar_t *)snt1;
   wchar_t *p2 = (wchar_t *)snt2;

   while ( *p1 && *p1 == *p2 ) {p1++; p2++;}
   return (*p1 > *p2) - (*p2 > *p1);
}
```

4. Файл text_func.h

```
#ifndef TEXT_FUNC_H
#define TEXT_FUNC_H

#include "text.h"

void delete_duplicates (Text_t *text);
void print_anagrams (const Text_t *text);
void cap_sort (Text_t *text);
void vowels_shift (Text_t *text);
void snt_vowels_shift (Sentence_t *snt);
void snt_find_and_replace (Sentence_t *snt, wchar_t *old, wchar_t *new);
int check_anagram (wchar_t *a, wchar_t *b);
int cap_cmp (const void* s1, const void* s2);

#endif
```

5. Файл text_func.c

```
#include "text_func.h"

void delete_duplicates (Text_t *text)
{
   int new_len = 0;
   for (int i = 0; i < text->len-1; i++)
   { if (text->sentences[i])
      for (int j = i+1; j< text->len; j++)
      { if (text->sentences[j])
         if (text->sentences[i]->len == text->sentences[j]->len)
         {
            /* copy snt[i] to s1 */
            wchar_t s1[text->sentences[i]->len+1];
            wcscpy(s1, text->sentences[i]->str);
            /* copy snt[j] to s2 */
            wchar_t s2[text->sentences[j]->len+1];
            wcscpy(s2, text->sentences[j]->str);

            /* lower both snt */
            for(int k = 0; s1[k] ; k++){
               s1[k] = towlower(s1[k]);
               s2[k] = towlower(s2[k]);
            }
```

```
            /* delete snt[j] if needed */
            if(!snt_str_cmp((void*)(&s1), (void*)(&s2)))
            {
               free(text->sentences[j]->str);
               free(text->sentences[j]);
               text->sentences[j] = NULL;
            }
         }
      }
   }
   /* shift text to remove NULL sentences */
   for (int i = 0; i < text->len; i++)
      if (text->sentences[i])
      {
         if (new_len != i) {
            text->sentences[new_len] = text->sentences[i];
            text->sentences[i] = NULL;
         }
         new_len++;
      }
   text->len = new_len;
}

void print_anagrams (const Text_t *text)
{
   int mask[text->len];
   memset(mask, 0, text->len);
   for (int i = 0; i < text->len-1; i++)
      if (mask[i] == 0)
      {
         for (int j = i + 1; j < text->len; j++)
            if (check_anagram(text->sentences[i]->str, text->sentences[j]->str))
            {
               mask[i] = 1;
               mask[j] = 1;
            }
         for (int j = 0; j < text->len; j++)
            if (mask[j] == 1) {
               wprintf(L"[%d|%d]:", i, j);
               print_sentence(*(text->sentences[j]));
               wprintf(L"\n");
               mask[j] = -1;
            }
      }
```

```
}

void cap_sort (Text_t *text)
{
   qsort(text->sentences, text->len, sizeof(Sentence_t*), cap_cmp);
}

void vowels_shift (Text_t *text)
{
   for (int i = 0; i < text->len; i++)
   {
      snt_vowels_shift(text->sentences[i]);
   }
}

void snt_vowels_shift (Sentence_t *snt)
{
   wchar_t *new_str = (wchar_t*) malloc((snt->len*2)*sizeof(wchar_t));
   int new_len = 0;
   for (int i = 0; snt->str[i]; i++)
   {
      if (wcschr(L"Yy", snt->str[i]))
      {
         new_str[new_len++] = (snt->str[i]+1);
         new_str[new_len++] = (snt->str[i]-24);
         continue;
      }
      if(wcschr(L"AEIOUaeiouАЕЁИОУЫЭаеёиоуыэ", snt->str[i]))
      {
         new_str[new_len++] = (snt->str[i]+1);
         new_str[new_len++] = (snt->str[i]+2);
         continue;
      }
      if (wcschr(L"Юю", snt->str[i]))
      {
         new_str[new_len++] = (snt->str[i]+1);
         new_str[new_len++] = (snt->str[i]-30);
         continue;
      }
      if (wcschr(L"Яя", snt->str[i]))
      {
         new_str[new_len++] = (snt->str[i]-31);
         new_str[new_len++] = (snt->str[i]-30);
         continue;
```

```
        }
        new_str[new_len++] = snt->str[i];
    }
    new_str[new_len] = L'\0';
    free(snt->str);
    snt->str = new_str;
    snt->len = new_len;
}

void snt_find_and_replace (Sentence_t *snt, wchar_t *old, wchar_t *new)
{
    wchar_t *new_str;
    int i, counter = 0;
    int new_len = wcslen(new);
    int old_len = wcslen(old);
    wchar_t *s = snt->str;
    for (i = 0; s[i]; i++)
        if (wcsstr(s + i, old) == (s + i))
        {
            counter++;
            i += old_len - 1;
        }
    if (old_len < new_len)
            new_str = (wchar_t *) malloc (sizeof(wchar_t) * (i + counter * (new_len -
old_len) + 2));
    else
        new_str = (wchar_t *) malloc (sizeof(wchar_t) * (i + counter * old_len + 2));
    i = 0;
    while (*s)
    {
        if (wcsstr (s, old) == s)
        {
            wcscpy (new_str + i, new);
            i += new_len;
            s += old_len;
        }
        else
            new_str[i++] = *(s++);
    }
    new_str[i] = '\0';
    free (snt->str);
    snt->str = new_str;
}
```

```
int check_anagram (wchar_t *a, wchar_t *b)
{
   int fst[128] = {0}, snd[128] = {0};
   // Calculating frequency of characters of first string
   for (int i = 0; a[i]; i++)
   {
      if (47 < (int)a[i] && (int)a[i] < 58)
         fst[a[i]-L'0'+0]++;
      if (64 < (int)a[i] && (int)a[i] < 91)
         fst[a[i]-L'A'+0+10]++;
      if (96 < (int)a[i] && (int)a[i] < 123)
         fst[a[i]-L'a'+0+10+26]++;
      if (1039 < (int)a[i] && (int)a[i] < 1104)
         fst[a[i]-L'A'+0+10+26+26]++;
   }
   for (int i = 0; b[i]; i++)
   {
      if (47 < (int)b[i] && (int)b[i] < 58)
         snd[b[i]-L'0'+0]++;
      if (64 < (int)b[i] && (int)b[i] < 91)
         snd[b[i]-L'A'+0+10]++;
      if (96 < (int)b[i] && (int)b[i] < 123)
         snd[b[i]-L'a'+0+10+26]++;
      if (1039 < (int)b[i] && (int)b[i] < 1104)
         snd[b[i]-L'A'+0+10+26+26]++;
   }
   // Comparing frequency of characters
   for (int i = 0; i < 128; i++)
   {
      if (fst[i] != snd[i])
      return 0;
   }
   return 1;
}

int cap_cmp (const void* s1, const void* s2)
{
   Sentence_t** snt1 = (Sentence_t**) s1;
   Sentence_t** snt2 = (Sentence_t**) s2;
   return ((*snt1)->cap_c < (*snt2)->cap_c);
}
```

6. **Файл Makefile:**

```
CC=gcc
CFLAGS=-c -Wall

all: main

main: main.o text.o text_func.o
	$(CC) main.o text.o text_func.o -o main.out

main.o: main.c text.o text_func.o
	$(CC) $(CFLAGS) main.c

text.o: text.c
	$(CC) $(CFLAGS) text.c

text_func.o: text_func.c text.o
	$(CC) $(CFLAGS) text_func.c

clean:
	rm -rf *.o
```