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 Си

Студентка гр. 0382 _____ Ситченко К.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Ситченко К.С.

Группа 0382

Тема работы: Обработка строк на языке Си

Исходные данные:
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. Заменить в тексте все подстроки “high noon” на “полдень” и “полночь” на “midnight”.
2. Найти и вывести все даты в тексте заданные в виде “ DD/MM/YYYY ” или “ YYYY-MM-DD ” в порядке возрастания этих дат.
3. Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.

4. Для всех вхождений дат в тексте вида “<day> <month> <year>, вывести эти даты в виде “DD.MM.YYYY” и строку “Happened” если эта дата была до текущей и “Not Happened” в противном случае. Например, для даты “03 Jan 1666” вывести “03.01.1666 Happened”.

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.

Содержание пояснительной записки:

Перечисляются требуемые разделы пояснительной записки (обязательны разделы «Содержание», «Введение», «Заключение», «Список использованных источников»)

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 02.11.2020

Дата сдачи реферата: 21.21.2020

Дата защиты реферата: 22.12.2020

Студентка _____ Ситченко К.С.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

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

Сначала программа печатает подсказку о том, что можно вводить текст, считывает его и выводит следующую подсказку о дальнейших доступных действиях с текстом. Предусматривается опция выхода из программы, а также возможность ввода пользователем неверной команды (программа сообщает об этом пользователю посредством вывода на экран сообщения). Затем выполняется обработка текста, соответствующая введенной команде и вывод данных.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В ходе данной курсовой работы производится создание программы по обработке пользовательского текста в зависимости от опции, выбранной пользователем.

Для реализации программы использовались функции стандартных библиотек языка Си, которые подключались с помощью заголовочных файлов stdio.h, stdlib.h, wchar.h, ctype.h, time.h, locale.h. Для работы с текстом, под который выделалась динамическая память, использовались структуры. Сборка программы происходит с помощью Makefile.

Разработка и тестирование данной программы производились в редакторе Vim и IDE Repl.it.

1. ЦЕЛЬ РАБОТЫ

Цель данной курсовой работы заключается в создании программы, принимающей на вход текст и обрабатывающий его в соответствии с желанием пользователя.

Для достижения данной цели было необходимо реализовать следующие задачи:

– считывание файла;

– функции обработки данного текста;

– создание Makefile для сборки программы.

2. ВЫПОЛНЕНИЕ РАБОТЫ

2.1. Структуры

Для хранения и работы с текстом в данной лабораторной работе использовались такие структуры как Text, Sentence, My_date. Sentence – структура для хранения предложений. Имеет 2 поля: wchar_t* sent – динамический массив для текста предложения и int size – целое число, обозначающее количество символов предложения. Text – структура для хранения текста. Имеет 3 поля: struct Sentence** lines – динамический массив для всех предложений текста, int max_amount – целое число, максимальное количество предложений, int real_amount – целое число, отображающее, сколько предложений содержится в тексте. My_date – структура для работы с датами. Имеет 3 поля: int year – цело число, год, int month – целое число, месяц, int day – целое число, день, wchar_t src[11] – массив, хранящий всю дату.

2.2. Функции ввода и вывода текста

Для реализации считывания текста используется две функции: *read_line* и *read_text*. *read_line* посимвольно считывает текст с помощью *getwchar*, удаляет из начала предложений незначащие пробелы и табуляцию. Формирует переменную *res (struct Sentence*)*, записывает в ее поля предложение и его размер и возвращает ее. Когда текст заканчивается, функция возвращает *NULL*. Функция *read_text* используется для формирования текста из предложений, полученных с помощью *read_line*. Помимо формирования текста в цикле ведется подсчет предложений, которые в него входят, и в случае если их больше указанного максимального количества, это количество увеличивается вдвое и происходит изменение ранее выделенной под текст памяти. Так же в данной функции происходит удаление повторно встречающихся предложений, которые сравниваются посимвольно, но без учета регистра).

Для вывода текста на экран используется функция *show_text*.

2.3. Функции изменения текста

first_word – функция для поиска первого слова в предложении.

last_word – функция для поиска последнего слова в предложении.

is_alright – функция, принимающая на вход предложение и сравнивающая его первое и последнее слово.

delete_lines – функция, в которой происходит удаление предложений, начинающихся и заканчивающихся на одно и то же слово.

find_and_change – функция, которая находит и заменяет в предложении все подстроки “high noon” на “полдень” и “полночь” на “midnight”.

change_substrings – функция, которая отправляет каждое предложение текста в функцию *find_and_change*.

2.4. Функции для работы с датами

is_date – функция, обеспечивающая проверку дат.

get_date – функция, формирующая дату.

cmp – функция-компаратор.

sort_date – функция, которая с помощью *is_date* и *get_date* находит даты, записывает их в массив, сортирует с помощью *qsort* и *cmp*, а затем выводит на экран результат.

display_date – функция, которая отправляет каждое предложение текста в *sort_date*.

settime – функция для форматирования даты.

h_nh – функция, которая находит в тексте все вхождения дат формата <day> <month> <year>, сравнивает с текущей датой и выводит сообщение, истекла эта дата или нет.

2.5. Основная функция

Для реализации работы с широкими символами помимо библиотеки *wchar.h* используется функция *setlocale* из библиотеки *locale.h.* На экран выводится сообщение “Please enter your text:”, а затем идет считывание текста посредством

функции *read_text*. После удаления всех повторяющихся предложений исправленный текст выводится на экран, после чего печатается подсказка с просьбой выбора команды. Введенная опция обрабатывается с помощью switch:

– если пользователь ввел “1”, то вызывается функция *change_substrings*, а ее результат выводится на экран;

– если пользователь ввел “2”, то вызывается функция *display_date;*

– если пользователь ввел “3”, то вызывается функция *delete_lines*, а ее результат выводится на экран;

– если пользователь ввел “4”, то вызывается функция *h_nh*;

– если пользователь ввел “5”, то работа с программой завершается;

– если пользователь ввел опцию, не представленную выше, то выводится сообщение *“Invalid action”*, а затем появляется возможность ввести другое значение.

После завершения работы с пользователем происходит очистка выделенной ранее памяти.

3. ТЕСТИРОВАНИЕ

На вход программа получила следующий текст:

Тени исчезают в high noon. At полночь the universe smells like stars.

Never say never. Always say yes. Always say yes.

Contcert will be on 10 Apr 2021. Feynman was born on 11 May 1918 in Queens.

We were there on 2018-09-28. Его день рождения 04/04/1986.

Сначала программа удалила повторяющиеся предложения и вывела результат:

Тени исчезают в high noon. At полночь the universe smells like stars. Never say never. Always say yes. Contcert will be on 10 Apr 2021. Feynman was born on 11 May 1918 in Queens. We were there on 2018-09-28. Его день рождения 04/04/1986.

Затем была выбрана опция замены подстрок. Программа сработала верно и вывела:

Тени исчезают в полдень. At midnigh the universe smells like stars. Never say never. Always say yes. Contcert will be on 10 Apr 2021. Feynman was born on 11 May 1918 in Queens. We were there on 2018-09-28. Его день рождения 04/04/1986.

Опция вывода дат на экран также вывела верный результат:

04/04/1986

2018-09-28

Далее была выбрана опция удаления предложений с одинаковыми первым и последним словом. Программа сработала верно:

Тени исчезают в полдень. At midnigh the universe smells like stars. Always say yes. Contcert will be on 10 Apr 2021. Feynman was born on 11 May 1918 in Queens. We were there on 2018-09-28. Его день рождения 04/04/1986.

Вывод дат с сообщением о том, произошло ли это событие или нет правильный:

При вводе опции “7” команда напечатала сообщение о неправильной команде и снова предоставила выбор опций.

Ввод опции “5” завершил работу программы.

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были изучены основные способы обработки текста на языке Си. Разработана программа для обработки текста в соответствии с выбранной пользователем опцией. Для реализации программы использовались функции стандартных библиотек языка Си и структуры. Код программы разделен на отдельные файлы, для сборки которых используется Makefile.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Программирование на C и C++ URL: http://www.c-cpp.ru/ (дата обращения: 18.12.2020).

2. Cplusplus URL: http://www.cplusplus.com/ (дата обращения: 20.12.2020).

3. Все о Hi-Tech URL: http://www.cplusplus.com/ (дата обращения: 18.12.2020).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

2.1. Название файла: struct.c

```
struct Sentence{
  wchar_t* sent;
  int size;
};

struct Text{
  struct Sentence** lines;
  int max_amount;
  int real_amount;
};

struct My_date{
  int year;
  int month;
  int day;
  wchar_t src[11];
};
```

2.1. Название файла: in_n_out.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include <ctype.h>
#include "struct.h"
#define MAX_SENT 20
#define MAX_SYM 100


struct Sentence* read_line(void){
  wchar_t buf[MAX_SYM];
  int sym, k=0, fl = 1, flag = 0;
  while ((sym = getwchar()) != EOF){
    if (sym == '\n')
      if (flag)
        return NULL;
      else
        flag = 1;
    else
      flag = 0;
    if (isspace(sym) && fl)
      continue;
    fl = 0;
    buf[k++] = sym;
    if (sym == '.'){
      struct Sentence* res = malloc(sizeof(struct Sentence*));
      buf[k] = '\0';
      res->size = k+1;
      res->sent = malloc(res->size*sizeof(wchar_t));
      wcsncpy(res->sent, buf, res->size);
      return res;
    }
```

```
    if (k == MAX_SYM)
      return NULL;
  }
  return NULL;
}

struct Text* read_text(void){
  struct Text* txt;
  struct Sentence* line;
  int i = 0, j = 0, k;
  txt = malloc(sizeof(struct Text));
  txt->max_amount = MAX_SENT;
  txt->real_amount = 0;
  txt->lines = malloc(MAX_SENT*sizeof(struct Sentence*));
  while ((line = read_line()) != NULL) {
    txt->lines[txt->real_amount++] = line;
    if (txt->real_amount == txt->max_amount){
      txt->max_amount *= 2;
      txt->lines = realloc(txt->lines, txt->max_amount);
    }
  }
  int fl = 0;
  for (i = 0; i < txt->real_amount; i++){
    for (j = i + 1; j < txt->real_amount; j++){
      if  ((wcscasecmp(txt->lines[i]->sent,  txt->lines[j]->sent))
== 0) {
        for (k = j; k < txt->real_amount-1; k++)
          txt->lines[k] = txt->lines[k+1];
        txt->real_amount--;
        j--;
        fl = 1;
      }
    }
  }
  return txt;
}

void show_text(struct Text* txt){
  for (int i = 0; i < txt->real_amount; i++){
    printf("%ls ",txt->lines[i]->sent);
  }
  printf("\n");
}
```

2.2. Название файла: in_n_out.h

```
struct Sentence* read_line(void);

struct Text* read_text(void);

void show_text(struct Text* txt);
```

3.1. Название файла: changes.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include "struct.h"
```

```
wchar_t* first_word(struct Sentence *line){
  int size = 25;
  wchar_t* word = malloc(size*sizeof(wchar_t*));
  int sym, i = 0;
  while(1){
    sym = line->sent[i];
    if (sym == ' ' || sym == '.' || sym == ','){
      word[i] = '\0';
      break;
    }
    else word[i++] = sym;
    if(i == size) {
      size *= 2;
      word = realloc(word, size);
    }
  }
  return word;
}

wchar_t* last_word(struct Sentence *line){
  short size = 25;
  wchar_t* word = malloc(size*sizeof(wchar_t*));
  int sym, j, k = 0, i = 2;
  while(1){
    sym = line->sent[line->size-1-i];
    if (sym == ' ' || sym == ','){
      word[i] = '\0';
      break;
    }
    else word[k++] = sym;
    if(line->size - i == -1){
      word[k] = '\0';
      break;
    }
    if(i == size){
      size *= 2;
      word = realloc(word, size);
    }
    i++;
  }
  for (i = 0; i < wcslen(word)/2; i++){
    j = word[i];
    word[i] = word[wcslen(word)-1-i];
    word[wcslen(word)-1-i] = j;
  }
  return word;
}

int is_alright(struct Sentence* line){
  wchar_t *f_word, *l_word;
  f_word = first_word(line);
  l_word = last_word(line);
  if(wcscasecmp(f_word, l_word) == 0)
    return 1;
  else
    return 0;
```

```
}

struct Text* delete_lines(struct Text* text){
  int i, k;
  for (i = 0; i < text->real_amount; i++){
    if (is_alright(text->lines[i]) != 0){
      for (k = i; k < text->real_amount-1; k++)
        text->lines[k] = text->lines[k+1];
        text->real_amount--;
        i--;
    }
  }
  return text;
}

struct Sentence* find_and_change(struct Sentence* line){
  wchar_t* ptr = wcsstr(line->sent, L"полночь");
  if (ptr)
    wmemcpy(ptr, L"midnight", wcslen(L"midnight")-1);

  int fl = 0;
  wchar_t* res = malloc(256*sizeof(wchar_t));
  wchar_t* ptr2, *ptr_res = res, *ptr_line = line->sent;
  wchar_t* repl = L"полдень";
  wchar_t* subs = L"high noon";
  for (; (*ptr_res = *ptr_line); ++ptr_line, ++ptr_res){
    if (!wcsncmp(ptr_line, subs, wcslen(subs))){
      fl = 1;
      wcscpy(ptr_res, repl);
      ptr_line += wcslen(subs)-1;
      ptr_res += wcslen(repl)-1;
    }
  }
  if (fl == 1){
    wcscpy(line->sent, L"");
    line->size = wcslen(res);
    wcscpy(line->sent, res);
  }

  free(res);
  return line;

}


struct Text* change_substrings(struct Text* text){
  int i, k;
  for (i = 0; i < text->real_amount; i++)
    find_and_change(text->lines[i]);
  return text;
}
```

3.2. Название файла: changes.h

```
wchar_t* first_word(struct Sentence *line);

wchar_t* last_word(struct Sentence *line);

int is_alright(struct Sentence* line);
```

```
struct Text* delete_lines(struct Text* text);
struct Sentence* find_and_change(struct Sentence* line);
struct Text* change_substrings(struct Text* text);
```

4.1. Название файла: date.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include <ctype.h>
#include <time.h>
#include "struct.h"

int is_date(wchar_t* ptr){
  if (wcslen(ptr)<10)
    return 0;
  int k = 0, i;
  for (i = 0; i <10; i++){
    if (isdigit(ptr[i]))
      k++;
  }
  if ((k==8) && ((ptr[2]==L'/' && ptr[5]==L'/') || (ptr[4]==L'-'
&& ptr[7]==L'-')))
    return 1;
  return 0;
}

void get_date(wchar_t* ptr, int* year, int* month, int* day){
  if (ptr[2] == L'/'){
    *day = (ptr[0] - L'0') * 10 + ptr[1] - L'0';
    *month = (ptr[3] - L'0') * 10 + ptr[4] - L'0';
    *year = (ptr[6] - L'0') * 1000 + (ptr[7] - L'0') * 100 +
(ptr[8] - L'0') * 10 + ptr[9] - L'0';
  }
  else {
    *day = (ptr[8] - L'0') * 10 + ptr[9] - L'0';
    *month = (ptr[5] - L'0') * 10 + ptr[6] - L'0';
    *year = (ptr[0] - L'0') * 1000 + (ptr[1] - L'0') * 100 +
(ptr[2] - L'0') * 10 + ptr[3] - L'0';
  }
}

int cmp(const void * a, const void * b){
  struct My_date *first = (struct My_date*) a;
  struct My_date *second = (struct My_date*) b;
  if (first->year == second->year && first->month == second->month
&& first->day == second->day)
    return 0;
  return (first->year*10000 + first->month*100 + first->day -
(second->year*10000 + second->month*100 + second->day));
}

void sort_date(wchar_t* ptr){
  int max_amount = 20;
  int real_amount = 0;
  struct My_date* ptr_date;
```

```
  ptr_date = malloc(max_amount*sizeof(struct My_date));

  int i, j, d, m, y;

  if (wcslen(ptr) < 10)
    return;

  for (i = 0; i < wcslen(ptr)-10; i++){
    if (is_date(&ptr[i])){
      get_date(&ptr[i], &y, &m, &d);
      if (real_amount == max_amount){
        max_amount *= 2;
        ptr_date   =   realloc(ptr_date,   max_amount*sizeof(struct
My_date));
      }
      ptr_date[real_amount].year = y;
      ptr_date[real_amount].month = m;
      ptr_date[real_amount].day = d;
      wcsncpy(ptr_date[real_amount].src, &ptr[i], 10);
      ptr_date[real_amount].src[10] = L'\0';
      real_amount++;
    }
  }
  qsort(ptr_date, real_amount, sizeof(struct My_date), cmp);
  for (i = 0; i < real_amount; i++)
    printf("%ls\n", ptr_date[i].src);
  free(ptr_date);
}

void display_date(struct Text* text){
  int i, k = 0;
  for (i =0; i < text->real_amount; i++){
    k += text->lines[i]->size;
  }
  wchar_t* buf = malloc(k*sizeof(wchar_t));
  buf[0] = L'\0';
  for (i =0; i < text->real_amount; i++)
    wcscat(buf, text->lines[i]->sent);
  sort_date(buf);
  free(buf);
}

wchar_t* settime(struct tm *a){
  wchar_t s[11];
  wchar_t* tmp;
  s[0] = 0;
  int length = wcsftime(s, 11, L"%d.%m.%Y", a);
  tmp = (wchar_t*) malloc(sizeof(s));
  wcscpy(tmp, s);
  return(tmp);
}

void h_nh(struct Text* text){
  struct tm *today;
  const time_t timer = time(NULL);
  today = localtime(&timer);
```

```
  int sym;
  int year, day;
  wchar_t month[4];
  wchar_t* date, *date3;
  for(int i = 0; i < text->real_amount; i++) {
    for (int j = 0; j < text->lines[i]->size; j++){
      sym = text->lines[i]->sent[j];
      if (isdigit(sym)){
        if (isalpha(text->lines[i]->sent[j+4])){
          date = malloc(11*sizeof(wchar_t));
          wcsncpy(date, &text->lines[i]->sent[j], 11);
          if((swscanf(date, L"%2d %3ls %4d", &day, month, &year))
== 3){
            month[3] = L'\0';
            int  month_name  =    !wcscmp(month,  L"Jan")  ?  0  :
!wcscmp(month,  L"Feb")  ?  1  :  !wcscmp(month,  L"Mar")  ?  2  :
!wcscmp(month,  L"Apr")  ?  3  :  !wcscmp(month,  L"May")  ?  4  :
!wcscmp(month,  L"Jun")  ?  5  :  !wcscmp(month,  L"Jul")  ?  6  :
!wcscmp(month,  L"Aug")  ?  7  :  !wcscmp(month,  L"Sep")  ?  8  :
!wcscmp(month, L"Oct") ? 9 : !wcscmp(month, L"Nov") ? 10 : 11;
            struct tm date2 = {0};
            date2.tm_year = year-1900;
            date2.tm_mon = month_name;
            date2.tm_mday = day;
            time_t t_date = mktime(&date2);
            date3 = settime(&date2);

            if (t_date < timer)
              printf("%ls Happened\n", date3);
            if (t_date > timer)
              printf("%ls Not happened\n", date3);
          }
        }
        j += 10;
        continue;
        free(date);
      }
    }
  }
}
```

4.2. Название файла: date.h

```
int is_date(wchar_t*ptr);

void get_date(wchar_t* ptr, int* year, int* month, int* day);

int cmp(const void * a, const void * b);

void sort_date(wchar_t* ptr);

void display_date(struct Text* text);

wchar_t* settime(struct tm *a);

void h_nh(struct Text* text);
```

4.1. Название файла: sum.c

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

```
int sum(int arr[], int n){
  int sum=0; int i, index=0;
  for (i=1; i<n; i++)
    if (abs(arr[i]) > abs(arr[index]))
      index = i;
  for (i=index; i < n; i++)
    sum += arr[i];
return sum;
}
```

4.2. Название файла: sum.h

```
int sum(int arr[], int n);
```

5. Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include "struct.h"
#include "in_n_out.h"
#include "changes.h"
#include "date.h"

int main(){
  setlocale(LC_ALL, "");
  struct Text* text;
  printf("Please enter your text:\n");
  text = read_text();
  show_text(text);

  printf("\nPlease choose one of next actions:\n1. Change
substrings; \n2. Display dates; \n3. Delete sentences; \n4.
Happened / Not Happened; \n5. Quit program.\nAction: ");

  int action;
  wscanf(L"%d", &action);
  while (action != 5){
    switch (action){
        case 1:
          change_substrings(text);
          show_text(text);
          break;
        case 2:
          display_date(text);
          break;
        case 3:
          delete_lines(text);
          show_text(text);
          break;
        case 4:
          h_nh(text);
          break;
        default:
          printf("Invalid action\n");
```

```
            }
            printf("Action: ");
            wscanf(L"%d", &action);
          }
          for(int i = 0; i < text->real_amount; i++)
            free(text->lines[i]);
          free(text->lines);
          free(text);
        }
```

6. Название файла: Makefile

```
main: main.o .o in_n_out.o changes.o date.o
	gcc main.o .o in_n_out.o changes.o date.o -o main
main.o: main.c
	gcc -c main.c
in_n_out.o: in_n_out.c
	gcc -c in_n_out.c
abs_min.o: abs_min.c
	gcc -c abs_min.c
changes.o: changes.c
	gcc -c changes.c
date.o: date.c
	gcc -c date.c
clean:
	rm *.o main
```

ПРИЛОЖЕНИЕ В

ПРИМЕР РАБОТЫ ПРОГРАММЫ