Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Факультет ИВТ

Кафедра вычислительных систем

**Курсовая работа**

на тему "ЧАСТОТНЫЙ АНАЛИЗ ТЕКСТА"

Выполнил:

студент гр. ИС-742

Добаев А.В.

Проверили:

Старший преподаватель Кафедры ВС

Перышкова Е.Н.

**Новосибирск, 2018**

**-Тема курсовой работы**

Поиск палиндромов в тексте

**-Задание на курсовую работу**

Разработать программу palindrom, выполняющую поиск всех палиндромов в заданном тексте. Команда palindrom принимает в качестве аргумента командной строки имя файла, содержащего текст на русском языке. Все найденные палиндромы распечатываются на экране.

**-Критерии оценки**

Оценка «*удовлетворительно»*: реализована проверка того, что весь текст входного файла целиком является палиндромом. Не предусмотрено динамическое выделение памяти под входные данные.

Оценка *«хорошо»:* реализована предварительная обработка текста: из каждого предложения удаляются все знаки препинания. После этого осуществляется проверка каждого предложения на выполнение свойства палиндрома. Обязательно динамическое выделение памяти под входные данные.

Оценка *«отлично»:* реализована предварительная обработка текста: из текста удаляются все пробелы и знаки препинания так, чтобы получилось одно большое слово. Для поиска подпалиндромов используется алгоритм, основанный на применении динамического программирования (http://comp-science.narod.ru/WebPage/p6.html). Обязательно динамическое выделение памяти под входные данные.

**-Указания к выполнению задания**

Палиндромом называются слово (потоп, шалаш) или текст (а роза упала на лапу Азора), читающиеся одинаково в обоих направлениях. Запуск программы должен производиться со следующими аргументами командной строки: $ palindrom text.txt Программа выполняет поиск всех палиндромов в файле text.txt и распечатывает результат работы на экране

**Работа алгоритма поиска палиндромов(псевдокод):**

***Проверка всего текста на палиндорм***

Цикл (start = 0; start < end; start++, end--)

Цикл (str[start] == ' ' || str[start] == '\n')

start++;

Конец цикла

Цикл (str[end] == ' ' || str[end] == '\n')

end--;

Конец цикла

Если (str[start] != str[end])

status = 0;

continue;

Конец если

Если (status)

printf("It is palindrom\n");

fputws(str, stdout);

Иначе

printf("It is not palindrom\n");

fputws(str, stdout);

Конец если

Конец цикла

***Удаление знаков препинания***

Цикл (int i = 0; i < wcslen(str); i++)

Если (str[index] == COMMA) {

Цикл (int i = index; str[i] != '\0'; i++)

str[i] = str[i + 1];

Конец цикла

Если (str[index] != '\0')

symbol\_check(str, index);

Иначе Если (str[index] == '.' || str[index] == EXC || str[index] == QUE)

str[index] = '\n';

Конец Если

Конец Цикла

***Проверка каждого предложения на палиндром.***

void palindrom\_task2(wchar\_t \*str)

unsigned int start = 0;

symbol\_check\_all(str);

int eof\_status = 1;

Цикл (eof\_status)

int status = 1;

unsigned int end = search\_new\_line(str, start) - 1;

int pst = start, pend = end;

Если (str[search\_new\_line(str, start)] == '\0')

eof\_status = 0;

continue;

Конец Если

Цикл (start; start < end; start++, end--)

Цикл (str[start] == ' ')

start++;

Конец цикла

Цикл (str[end] == ' ')

end--;

Конец цикла

Если (str[start] != str[end])

status = 0;

continue;

Конец если

Конец цикла

Если (status)

print\_line(pst, pend, str);

printf(" - It is palindrom\n");

Иначе

print\_line(pst, pend, str);

printf(" - It is not palindrom\n");

Конец если

start = search\_new\_line(str, start) + 1;

Цикл (str[start] == '\n') {

start++;

Конец цикла

Цикл (str[start] == ' ')

start++;

Конец цикла

Конец цикла

***Удаление стороннего символа***

void delete\_other\_symbol(wchar\_t \*str, int index)

Если (str[index] >= RU\_U\_S && str[index] <= RU\_L\_E)

symbol\_correct(str, index);

Иначе Если ((str[index] >= EN\_U\_S && str[index] <= EN\_U\_E)

|| (str[index] >= EN\_L\_S && str[index] <= EN\_L\_E))

symbol\_correct(str, index);

Иначе

Цикл (int i = index; str[i] != '\0'; i++)

str[i] = str[i + 1];

Конец цикла

Если (str[index] != '\0')

delete\_other\_symbol(str, index);

Конец Если

Конец Если

Определение типа палиндрома

int check\_type\_palindrom(wchar\_t \*str, int mp)

int lp = mp - 1, rp = mp + 1;

Если (lp >= 0 && rp != '\0')

Если (str[lp] == str[rp])

return ODD;

Иначе Если (str[mp] == str[rp])

return EVEN;

Иначе return NOPE;

Конец если

Иначе

return NOPE;

Конец если

Поиск палиндромов

void palindrom\_task3(wchar\_t \*str)

delete\_all\_other\_symbols(str);

Цикл (int i = 0; i < wcslen(str); i++)

int type = check\_type\_palindrom(str, i);

int lp, rp;

switch (type)

case ODD:

lp = i - 1;

rp = i + 1;

search\_subpalindrome(str, lp, rp);

break;

case EVEN:

lp = i;

rp = i + 1;

search\_subpalindrome(str, lp, rp);;

default:

break;

Конец цикла

***Поиск подпалиндромов***

void search\_subpalindrome(wchar\_t \*str, int lp, int rp)

int n = 0;

Цикл (str[rp] == str[lp]) {

Если (lp >= 0 && str[rp] != '\0') {

lp--;

rp++;

n++;

Конец Если

Конец Цикла

lp++;

rp--;

Если (n > 2) {

print\_in\_file(lp, rp, str, stdout);

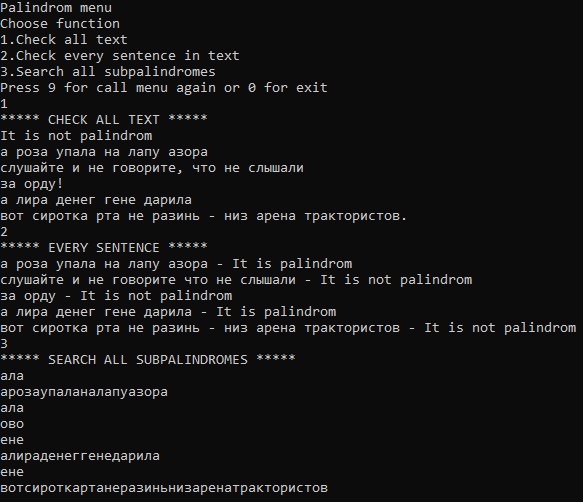
Конец Если

**Входные данные:** берутся из произвольного файла, который заполнен тексом.

**Запуск:** ./palindrom “name\_your\_file”

|  |  |
| --- | --- |
| ***Test.txt*** | ***Result*** |
| А роза упала на лапу азора  Слушайте и не говорите, что не слышали  За ОРДУ!  А лира денег Гене дарила  Вот сиротка рта не разинь - низ арена трактористов. | **1.Check all text:**  It is not palindrom  а роза упала на лапу азора  слушайте и не говорите, что не слышали  за орду!  а лира денег гене дарила  вот сиротка рта не разинь - низ арена трактористов.  **2.Check every sentence in text:**  а роза упала на лапу азора - It is palindrom  слушайте и не говорите что не слышали - It is not palindrom  за орду - It is not palindrom  а лира денег гене дарила - It is palindrom  вот сиротка рта не разинь - низ арена трактористов - It is not palindrom  **3.Search all subpalindromes:**  ала  арозаупаланалапуазора  ала  ово  ене  алираденеггенедарила  ене  вотсироткартанеразиньнизаренатрактористов |

**Скриншот**



**Исходный код:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include "palindrom.h"

main.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main(int argc, char \*argv[]) {

const char \* in\_file\_name = NULL;

if (argc != 2) {

printf("Incorrect input, try again. (Example: ./prog \*.txt\n");

return 0;

}

in\_file\_name = argv[1];

setlocale(LC\_ALL, "RUS");

f = fopen(in\_file\_name, "r");

if (f == NULL) {

printf("Error open file");

return 0;

}

fseek(f, 0, SEEK\_END);

unsigned int text\_size = ftell(f);

rewind(f);

unsigned int i = 0;

wchar\_t \*str = (wchar\_t\*)malloc(sizeof(wchar\_t)\* text\_size);

while (!feof(f)) {

str[i] = fgetwc(f);

i++;

}

str[i] = '\0';

menu();

int m = 1;

while (m != 0) {

scanf("%d", &m);

switch (m) {

case 1:

printf("\*\*\*\*\* CHECK ALL TEXT \*\*\*\*\*\n");

palindrom\_task1(str);

break;

case 2:

printf("\*\*\*\*\* EVERY SENTENCE \*\*\*\*\*\n");

palindrom\_task2(str);

break;

case 3:

printf("\*\*\*\*\* SEARCH ALL SUBPALINDROMES \*\*\*\*\*\n");

palindrom\_task3(str);

break;

case 9:

menu();

break;

case 0:

return 0;

default:

printf("Incorrect input, try again\n");

break;

}

}

fclose(f);

return 0;

}

functions.c\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "palindrom.h"

#define \_CRT\_SECURE\_NO\_WARNINGS

//\*\*\*\*\*\*\*\* 1 part \*\*\*\*\*\*\*\*//

void print\_line(int start, int end, wchar\_t \*str) {

for (; start <= end; start++) {

fputwc(str[start], stdout);

}

}

void text\_correct\_all(wchar\_t \*str) {

for (int i = 0; i < wcslen(str); i++) {

symbol\_correct(str, i);

}

}

void symbol\_correct(wchar\_t \*str, size\_t index) {

if (iswupper(str[index]))

str[index] = str[index] + 0x020;

}

void palindrom\_task1(wchar\_t \*str) {

unsigned int start, end, status = 1;

fseek(f, 0, SEEK\_END);

end = ftell(f) - 1;

rewind(f);

text\_correct\_all(str);

for (start = 0; start < end; start++, end--) {

while (str[start] == ' ')

start++;

while (str[end] == ' ')

end--;

if (str[start] != str[end]) {

status = 0;

continue;

}

}

if (status) {

printf("It is palindrom\n");

fputws(str, stdout);

}

else {

printf("It is not palindrom\n");

fputws(str, stdout);

}

}

void menu() {

printf("Palindrom menu\n");

printf("Choose function\n");

printf("1.Check all text\n");

printf("2.Check every sentence in text\n");

printf("3.Search all subpalindromes\n");

printf("Press 9 for call menu again or 0 for exit\n");

}

// \*\*\*\*\*\*\*\*\*\* 2part \*\*\*\*\*\*\*\*\*\*\*//

void symbol\_check(wchar\_t \*str, size\_t index) {

symbol\_correct(str, index);

if (str[index] == COMMA) {

for (int i = index; str[i] != '\0'; i++) {

str[i] = str[i + 1];

}

if (str[index] != '\0')

symbol\_check(str, index);

}

else if (str[index] == '.' || str[index] == EXC || str[index] == QUE) {

str[index] = '\n';

}

}

void symbol\_check\_all(wchar\_t \*str) {

for (int i = 0; i < wcslen(str); i++) {

symbol\_check(str, i);

}

}

void palindrom\_task2(wchar\_t \*str) {

unsigned int start = 0;

symbol\_check\_all(str);

int eof\_status = 1;

while (eof\_status) {

int status = 1;

unsigned int end = search\_new\_line(str, start) - 1;

int pst = start, pend = end;

if (str[search\_new\_line(str, start)] == '\0') {

eof\_status = 0;

continue;

}

for (start; start < end; start++, end--) {

while (str[start] == ' ')

start++;

while (str[end] == ' ')

end--;

if (str[start] != str[end]) {

status = 0;

continue;

}

}

if (status) {

print\_line(pst, pend, str);

printf(" - It is palindrom\n");

}

else {

print\_line(pst, pend, str);

printf(" - It is not palindrom\n");

}

start = search\_new\_line(str, start) + 1;

while (str[start] == '\n') {

start++;

}

while (str[start] == ' ')

start++;

}

}

int search\_new\_line(wchar\_t \*str, unsigned int start\_line) {

unsigned int new\_line = start\_line;

while (str[new\_line] != '\n' && str[new\_line] != '\0') {

new\_line++;

}

return new\_line;

}

// \*\*\*\*\*\*\*\*\*\*\* part 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void delete\_all\_other\_symbols(wchar\_t \*str) {

for (int i = 0; i < wcslen(str); i++) {

delete\_other\_symbol(str, i);

}

}

void delete\_other\_symbol(wchar\_t \*str, int index) {

if (str[index] >= RU\_U\_S && str[index] <= RU\_L\_E) {

symbol\_correct(str, index);

}

else if ((str[index] >= EN\_U\_S && str[index] <= EN\_U\_E)

|| (str[index] >= EN\_L\_S && str[index] <= EN\_L\_E)) {

symbol\_correct(str, index);

}

else {

for (int i = index; str[i] != '\0'; i++) {

str[i] = str[i + 1];

}

if (str[index] != '\0')

delete\_other\_symbol(str, index);

}

}

void palindrom\_task3(wchar\_t \*str) {

delete\_all\_other\_symbols(str);

for (int i = 0; i < wcslen(str); i++) {

int type = check\_type\_palindrom(str, i);

int lp, rp;

switch (type)

{

case ODD:

lp = i - 1;

rp = i + 1;

search\_subpalindrome(str, lp, rp);

break;

case EVEN:

lp = i;

rp = i + 1;

search\_subpalindrome(str, lp, rp);;

default:

break;

}

}

}

void print\_in\_file(int start, int end, wchar\_t\* str, FILE \*o) {

for (; start <= end; start++) {

fseek(o, 0, SEEK\_END);

fputwc(str[start], o);

}

fputwc('\n', o);

rewind(o);

}

void search\_subpalindrome(wchar\_t \*str, int lp, int rp) {

int n = 0;

while (str[rp] == str[lp]) {

if (lp >= 0 && str[rp] != '\0') {

lp--;

rp++;

n++;

}

}

lp++;

rp--;

if (n > 0) {

print\_in\_file(lp, rp, str, stdout);

}

}

int check\_type\_palindrom(wchar\_t \*str, int mp) {

int lp = mp - 1, rp = mp + 1;

if (lp >= 0 && rp != '\0') {

if (str[lp] == str[rp])

return ODD;

else if (str[mp] == str[rp])

return EVEN;

else return NOPE;

}

else

return NOPE;

}

palindrom.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <wchar.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#define RU\_U\_S 0x0410

#define RU\_U\_E 0x042f

#define RU\_L\_S 0x0430

#define RU\_L\_E 0x044f

#define EN\_U\_S 0x0041

#define EN\_U\_E 0x005A

#define EN\_L\_S 0x0061

#define EN\_L\_E 0x007A

#define COMMA 0x002C

#define POINT 0x002F

#define EXC 0x0021 // восклицательный знак

#define QUE 0x003F // вопросительный

#define ODD 1

#define EVEN 2

#define NOPE 0

FILE \*f;

void symbol\_correct(wchar\_t \*str, size\_t index);

int search\_new\_line(wchar\_t \*str, unsigned int start\_line);

int check\_type\_palindrom(wchar\_t \*str, int mp);

void delete\_other\_symbol(wchar\_t \*str, int index);

void delete\_all\_other\_symbols(wchar\_t \*str);

void symbol\_check\_all(wchar\_t \*str);

void symbol\_check(wchar\_t \*str, size\_t index);

void search\_subpalindrome(wchar\_t \*str, int lp, int rp);

void palindrom\_task1(wchar\_t \*str);

void palindrom\_task2(wchar\_t \*str);

void palindrom\_task3(wchar\_t \*str);

void text\_correct\_all(wchar\_t \*str);

void print\_line(int start, int end, wchar\_t \*str);

void menu(void);