Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 2

«Текстовые файлы»

Проверил: Выполнил:

асс. каф. ЭВМ ст. гр. 250505

И.Г. Скиба М.С. Патюпин

МИНСК 2023

# 1 Цель и задачи лабораторной работы

**1.1** Цель работы – освоить принципы работы с текстовыми файлами в языке С.

**1.2** Задача:написать программу сжатия текстового файла по алгоритму:

- Подсчет частоты встречи каждого слова в файле.

- Поиск самого популярного среди длинных слов (А).

- Поиск самого непопулярного среди коротких слов (В).

- Замена всех слов А и В друг на друга.

- Повтор пунктов 2-4 до тех пор, пока это имеет смысл.

# 2 Результаты выполнения лабораторной работы

**2.1.1** Блок-схемы алгоритма выполнения задания

На рисунке 1.1 приведена блок-схема алгоритма для выполнения задания.

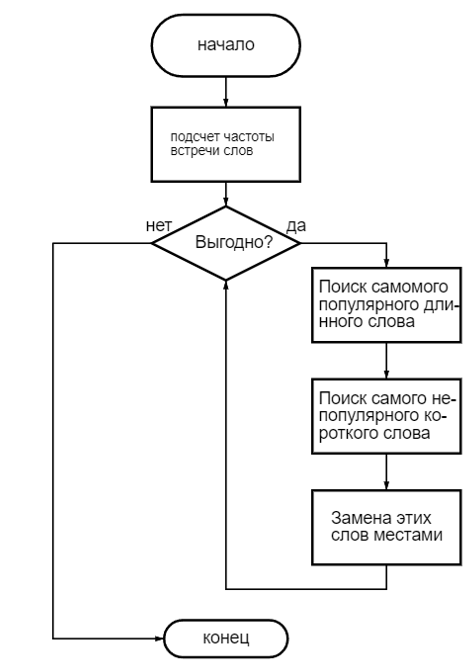


Рисунок 1.1

**2.1.2** Листинг компьютерной программы:

**2.1.2.1** Файл main.c

#define f\_name "C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt"

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MAX\_WORD\_LENGTH 100

#include "func.h"

int main() {

long int size;

size = get\_file\_size\_by\_path(f\_name);

printf\_s("Initial file size: %ld\n", size);

int num\_words = 0;

Word\* words = read\_words\_from\_file(f\_name, &num\_words);

Word\* most\_frequent\_long\_word;

Word\* least\_frequent\_short\_word;

int max\_pairs = num\_words / 2;

char\*\* pairs = malloc(max\_pairs \* sizeof(char\*));

for (int i = 0; i < max\_pairs; i++) {

pairs[i] = malloc(MAX\_WORD\_LENGTH \* 2 + 1);

pairs[i][0] = '\0';

}

int pair\_count = 0;

while (1) {

find\_and\_save\_words(words, num\_words, &most\_frequent\_long\_word, &least\_frequent\_short\_word);

int old\_size = find\_old\_size(most\_frequent\_long\_word, least\_frequent\_short\_word);

//

int new\_size = find\_new\_size(most\_frequent\_long\_word, least\_frequent\_short\_word);

//

if (new\_size + strlen(most\_frequent\_long\_word) + 1 + strlen(least\_frequent\_short\_word) >= old\_size) {//+ добавить место под свои слова для припащения

break;

}

replace\_words\_in\_file(f\_name, most\_frequent\_long\_word, least\_frequent\_short\_word);

sprintf(pairs[pair\_count], "%s %s", most\_frequent\_long\_word->word, least\_frequent\_short\_word->word);

pair\_count++;

}

FILE\* file = fopen(f\_name, "a");

fprintf(file, " (/\*/\*/ ");

for (int i = 0; i < pair\_count; i++) {

fprintf(file, "%s\n", pairs[i]);

}

fclose(file);

for (int i = 0; i < max\_pairs; i++) {

free(pairs[i]);

}

free(pairs);

free(words);

size = get\_file\_size\_by\_path(f\_name);

printf\_s("New file size: %ld\n", size);

printf\_s("Okk");

getchar();

return 0;

}

**2.1.2.2** Файл func.h

#ifndef INC\_5\_1\_FUNCTIONS\_H

#define INC\_5\_1\_FUNCTIONS\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <sys/stat.h>

typedef struct {

char word[MAX\_WORD\_LENGTH];

int count;

int was\_used;

} Word;

Word\* read\_words\_from\_file(char\* filename, int\* num\_words);

void find\_and\_save\_words(Word words[], int num\_words, Word\*\* most\_frequent\_long\_word\_ptr, Word\*\* least\_frequent\_short\_word\_ptr);

char\* read\_line(FILE\* file);

void replace\_words\_in\_file(char\* filename, Word\* word\_a, Word\* word\_b);

int find\_old\_size(Word\* word1, Word\* word2);

int find\_new\_size(Word\* word1, Word\* word2);

void printWords(Word words[], int size);

#endif //INC\_5\_1\_FUNCTIONS\_H

**2.1.2.3** Файл func.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MAX\_WORD\_LENGTH 100

#include "func.h"

Word\* read\_words\_from\_file(char\* filename, int\* num\_words) {

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Error\n");

exit(1);

}

Word\* words = malloc(sizeof(Word));

char buffer[MAX\_WORD\_LENGTH];

while (fscanf(file, "%s", buffer) != EOF) {

int found = 0;

for (int i = 0; i < \*num\_words; i++) {

if (strcmp(buffer, words[i].word) == 0) {

words[i].count++;

found = 1;

break;

}

}

if (!found) {

words = realloc(words, (\*num\_words + 1) \* sizeof(Word));

strcpy(words[\*num\_words].word, buffer);

words[\*num\_words].count = 1;

words[\*num\_words].was\_used = 0;

(\*num\_words)++;

}

}

fclose(file);

return words;

}

void find\_and\_save\_words(Word words[], int num\_words, Word\*\* most\_frequent\_long\_word\_ptr, Word\*\* least\_frequent\_short\_word\_ptr) {

Word\* most\_frequent\_long\_word = NULL;

Word\* least\_frequent\_short\_word = NULL;

for (int i = 0; i < num\_words; i++) {

if (!words[i].was\_used) {

if (most\_frequent\_long\_word == NULL || (strlen(words[i].word) > strlen(most\_frequent\_long\_word->word) && words[i].count > most\_frequent\_long\_word->count)) {

most\_frequent\_long\_word = &words[i];

}

if (least\_frequent\_short\_word == NULL || (strlen(words[i].word) < strlen(least\_frequent\_short\_word->word) && words[i].count < least\_frequent\_short\_word->count)) {

least\_frequent\_short\_word = &words[i];

}

}

}

if (!most\_frequent\_long\_word->was\_used && !least\_frequent\_short\_word->was\_used) {

most\_frequent\_long\_word->was\_used = 1;

least\_frequent\_short\_word->was\_used = 1;

\*most\_frequent\_long\_word\_ptr = most\_frequent\_long\_word;

\*least\_frequent\_short\_word\_ptr = least\_frequent\_short\_word;

}

}

char\* read\_line(FILE\* file) {

size\_t buffer\_size = 1024;

char\* buffer = malloc(buffer\_size);

size\_t len = 0;

while (fgets(buffer + len, buffer\_size - len, file) != NULL) {

len += strlen(buffer + len);

if (buffer[len - 1] == '\n') {

break;

}

buffer\_size \*= 2;

buffer = realloc(buffer, buffer\_size);

}

if (len == 0) {

free(buffer);

return NULL;

}

return buffer;

}

void replace\_words\_in\_file(char\* filename, Word\* word\_a, Word\* word\_b) {

FILE\* file = fopen(filename, "r");

FILE\* new\_file = tmpfile();

char buffer[MAX\_WORD\_LENGTH];

char\* line = NULL;

char\* next\_line = read\_line(file);

while (next\_line != NULL) {

line = next\_line;

char\* word = strtok(line, " \n");

while (word != NULL) {

if (strcmp(word, word\_a->word) == 0) {

fprintf(new\_file, "%s", word\_b->word);

}

else if (strcmp(word, word\_b->word) == 0) {

fprintf(new\_file, "%s", word\_a->word);

}

else {

fprintf(new\_file, "%s", word);

}

word = strtok(NULL, " \n");

if (word != NULL) {

fprintf(new\_file, " ");

}

}

next\_line = read\_line(file);

if (next\_line != NULL) {

fprintf(new\_file, "\n");

}

free(line);

}

fclose(file);

file = fopen(filename, "w");

rewind(new\_file);

while (fgets(buffer, MAX\_WORD\_LENGTH, new\_file) != NULL) {

fputs(buffer, file);

}

fclose(file);

fclose(new\_file);

}

int find\_old\_size(Word\* word1, Word\* word2) {

return strlen(word1->word) \* word1->count + strlen(word2->word) \* word2->count;

}

int find\_new\_size(Word\* word1, Word\* word2) {

return strlen(word1->word) \* word2->count + strlen(word2->word) \* word1->count + strlen(word1->word) + strlen(word2->word)+2;

}

long int get\_file\_size\_by\_path(const char\* path) {

struct stat st;

if (stat(path, &st) == 0) {

return st.st\_size;

}

else {

return -1;

}

}

**2.1.3** Результат выполнения компьютерной программы, представлен на рисунке 1.2

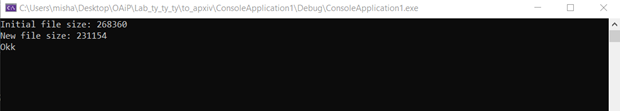


Рисунок 1.2

**2.2 Результат выполнения программы 2**

Разархивировать прежде заархивированный файл.

**2.2.1** Блок-схема алгоритма смешанного типа предоставлена на рисунке 2.1.

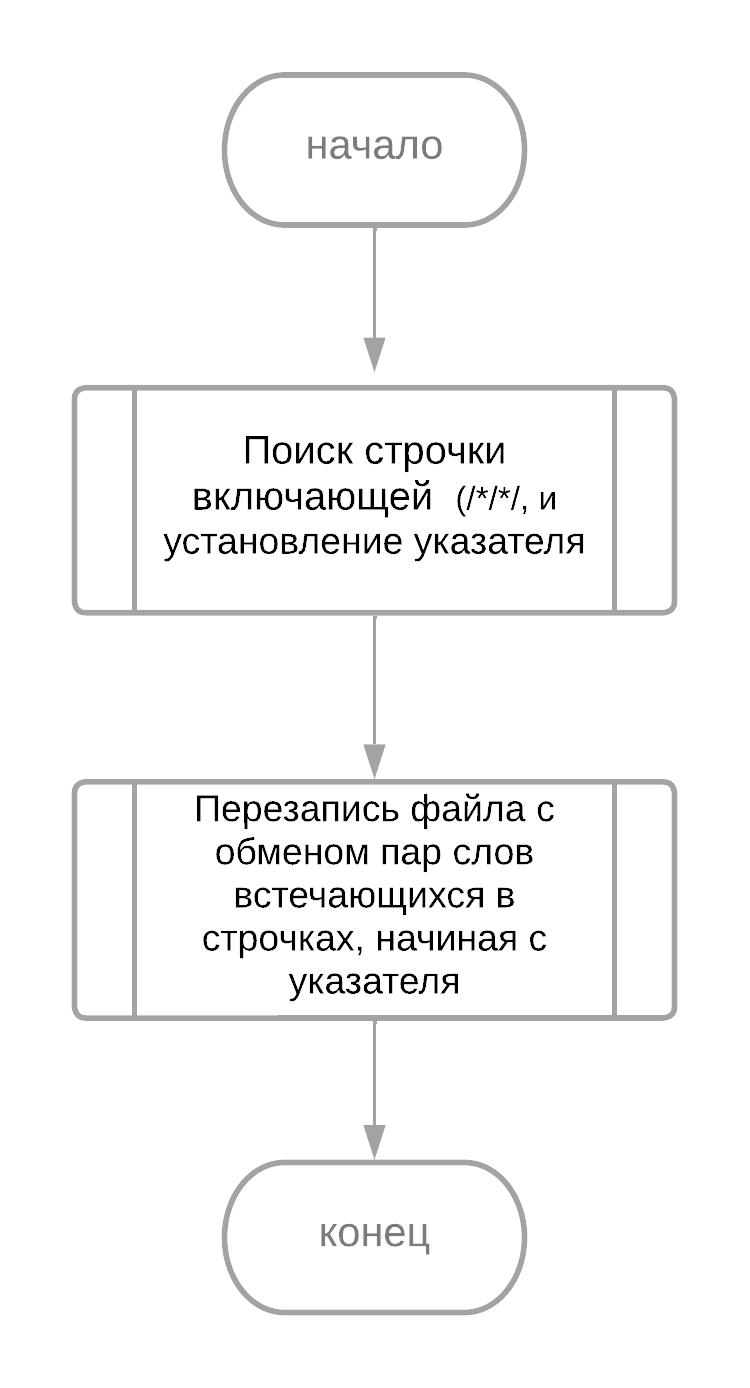


Рисунок 2.1

**2.2.2** Листинг компьютерной программы:

**2.2.2.1** Файл func.h

#ifndef INC\_5\_1\_FUNCTIONS\_H

#define INC\_5\_1\_FUNCTIONS\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <io.h>

#include <locale.h>

#include <sys/stat.h>

void replace\_words\_in\_file(char\* filename, char\* word\_a, char\* word\_b);

void remove\_stupid\_space(char\* filename);

#endif //INC\_5\_1\_FUNCTIONS\_H

**2.2.2.2** Файл func.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MAX\_WORD\_LENGTH 100

#include "func.h"

long int get\_file\_size\_by\_path(const char\* path) {

struct stat st;

if (stat(path, &st) == 0) {

return st.st\_size;

}

else {

return -1;

}

}

void replace\_words\_in\_file(char\* filename, char\* word\_a, char\* word\_b) {

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Error8\n");

exit(1);

}

FILE\* new\_file = fopen("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/compr.txt", "w");

if (new\_file == NULL) {

printf("Error14\n");

exit(1);

}

size\_t buffer\_size = 1024;

char\* buffer = malloc(buffer\_size);

while (fgets(buffer, buffer\_size, file) != NULL) {

size\_t len = strlen(buffer);

while (len > 0 && buffer[len - 1] != '\n') {

buffer\_size \*= 2;

buffer = realloc(buffer, buffer\_size);

if (fgets(buffer + len, buffer\_size - len, file) == NULL) break;

len += strlen(buffer + len);

}

char\* word = strtok(buffer, " \n");

while (word != NULL) {

if (strcmp(word, word\_a) == 0) {

fprintf(new\_file, "%s", word\_b);

}

else if (strcmp(word, word\_b) == 0) {

fprintf(new\_file, "%s", word\_a);

}

else {

fprintf(new\_file, "%s", word);

}

word = strtok(NULL, " \n");

if (word != NULL)

fprintf(new\_file, " ");

}

fprintf(new\_file, "\n");

}

fclose(file);

fclose(new\_file);

remove(filename);

rename("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/compr.txt", filename);

free(buffer);

}

void remove\_stupid\_space(char\* filename) {

FILE\* file = fopen(filename, "r+");

if (file == NULL) {

printf("Error57\n", filename);

exit(1);

}

fseek(file, -1L, SEEK\_END);

char last\_char;

fread(&last\_char, 1, 1, file);

if (last\_char == ' ') {

long pos = ftell(file);

int fd = \_fileno(file);

\_chsize\_s(fd, pos - 1);

}

fclose(file);

}

**2.2.2.3** Файл main.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MAX\_WORD\_LENGTH 100

#include "func.h"

int main() {

long int size = get\_file\_size\_by\_path("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt");

printf\_s("Initial file size: %ld\n", size);

FILE\* text = fopen("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt", "r");

if (text == NULL) {

printf("Error10\n");

exit(1);

}

int separator\_pos = -1;

char buffer[MAX\_WORD\_LENGTH];

while (fscanf(text, "%s", buffer) != EOF) {

if (strcmp(buffer, "(/\*/\*/") == 0) {

separator\_pos = ftell(text);

break;

}

}

int num\_words = 0;

while (fscanf(text, "%s", buffer) != EOF) {

num\_words++;

}

char\*\* words = malloc(num\_words \* sizeof(char\*));

for (int i = 0; i < num\_words; i++) {

words[i] = malloc(MAX\_WORD\_LENGTH \* sizeof(char));

}

fseek(text, separator\_pos, SEEK\_SET);

int index = 0;

while (fscanf(text, "%s", buffer) != EOF) {

strcpy(words[index], buffer);

index++;

}

fclose(text);

for (int i = num\_words - 2; i >= 0; i -= 2) {

replace\_words\_in\_file("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt", words[i], words[i + 1]);

}

FILE\* file = fopen("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt", "r");

FILE\* new\_file = fopen("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/compr.txt", "w");

if (new\_file == NULL) {

printf("Error43\n");

exit(1);

}

size\_t buffer\_size = 1024;

char\* buffer2 = malloc(buffer\_size);

int line\_count = 0;

while (fgets(buffer2, buffer\_size, file) != NULL) {

size\_t len = strlen(buffer2);

while (len > 0 && buffer2[len - 1] != '\n') {

buffer\_size \*= 2;

buffer2 = realloc(buffer2, buffer\_size);

if (fgets(buffer2 + len, buffer\_size - len, file) == NULL) break;

len += strlen(buffer2 + len);

}

char\* word = strtok(buffer2, " \n");

while (word != NULL) {

if (strcmp(word, "(/\*/\*/") == 0) {

fseek(new\_file, -1L, SEEK\_CUR);

break;

}

else {

fprintf(new\_file, "%s", word);

word = strtok(NULL, " \n");

if (word != NULL)

fprintf(new\_file, " ");

}

}

if (word != NULL)

break;

line\_count++;

if (line\_count >= 1)

fprintf(new\_file, "\n");

}

fclose(file);

fclose(new\_file);

remove\_stupid\_space("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/compr.txt");

remove("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt");

rename("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/compr.txt", "C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt");

free(buffer2);

for (int i = 0; i < num\_words; i++) {

free(words[i]);

}

free(words);

size = get\_file\_size\_by\_path("C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt");

printf\_s("New file size: %ld\n", size);

printf\_s("kkO");

getchar();

return 0;

}

**2.2.3** Результат выполнения компьютерной программы, представлен на рисунке 2.2

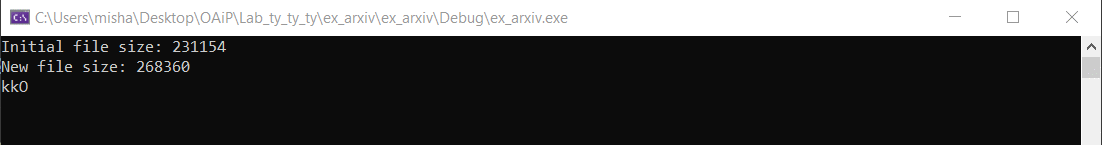


Рисунок 2.2

**2.3 Выводы по результатам выполнения ЛР**

В результате выполнения ЛР я научился составлять линейные и разветвляющиеся алгоритмы, а так же писать код на языке Си по составленному алгоритму, а также работать с файлами.

**3 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Лекционный материал по теме «Введение в Си» .

[2] «Как программировать на С» Х.М. Дейтел, П. Дж. Дейтел

[3] Основы алгоритмизации и программирования : язык Си : учебно-метод. пособие / Ю. А. Луцик, А. М. Ковальчук, Е. А. Сасин. – Минск