Звіт з лабораторної роботи

за дисципліною "програмування"

студента групи ПА-17-1

Панасенка Егора Сергійовича

Кафедра комп’ютерних технологій, фпм, дну

2017/2018 навч.р.

1. Постановка задачі:
   1. За допомогою текстового редактора створити файл, що містить текст. Довжина рядка тексту не повинна перевищувати 80 символів. Це вихідний файл.
   2. Програма повинна:
      1. інформацію вихідного файлу записати в вихідний файл;
      2. реалізувати дії, зазначені в індивідуальному відповідну інформацію в вихідний файл.
   3. Ім'я вихідного файлу задає користувач під час виконання або в командному рядку.
   4. Ім'я вихідного файлу: перші символи - не більше трьох символів з імені вихідного файлу, а решта символи \_out.
   5. Вихідний файл повинен мати розширення .dat.
   6. Програма повинна забезпечувати за запитом користувача:
      1. висновок тексту вихідного файлу на екран дисплея;
      2. висновок тексту вихідного файлу на екран дисплея.
   7. Закінчені послідовності дій оформити у вигляді функцій. Всі необхідні дані для функцій повинні передаватися їм в якості параметрів. Використання глобальних змінних у функціях не допускається.
   8. Визначення функцій і головну функцію розмістити в двох окремих файлах.
   9. Індивідуальне завдання: Варіант 9. Визначити спочатку рядки, що починаються з однобуквених слів, а потім всі інші.
2. Опис ходу розв’язку:
   1. У файлі Makefile задаються функції для компіляції та очищення від бінарних файлів
   2. У файлі main.c знаходиться головна функція у якій виконуються такі дії
      1. Якщо з командного рядка ввели існуючий файл, то відкрити його, а якщо ні то запитувати файл доки не отримаємо якийсь.
      2. Узяти усі рядки з файлу
      3. Відібрати спочатку файли з одним символом, а потім з декількома.
      4. Вивести данні за вибором.
      5. Закрити файл, звільнити пам’ять і вийти з програми.
   3. У файлі functions.h задаються усі потрібні функції.
   4. У файлі functions.c задаються такі алгоритми функцій:
      1. say: Виводить текст у файл або у консоль
      2. get\_string: Забирає посимвольно рядок за допомогою динамічного масиву та виводить адресу масива.
      3. get\_inputfile: Запитує назву файлу, доки не може відкрити його і виводить назву файлу, а через аргументи задає поток файлу.
      4. create\_outputfile: Генерує назву вихідного файлу, створює і відкриває його.
      5. get\_text: забирає текст з потоку у двовимірний динамічний массив складений із рядків, а рядки з символів і виводить адресу отриманого массиву.
      6. sort\_text: спочатку складає рядки з однією латинською буквою у новий масив, а потім усі інші рядки і виводить адресу нового масиву.
      7. check\_line: перевіряє чи починається рядок з однієї букви.
      8. ign\_other: у якій будуть ігноруватися усі данні до кінця рядка для того щоб непотрібні дані не заважали в майбутньому.
      9. menu: у якій запитується число від одного до значення аргументу count для вибору меню та виводиться відповідь у числі.
3. Вихідний текст програми розв’язку задачі

## Makefile

OBJ = main.o functions.o

DEPS = main.h functions.h

LIBS =

CFLAGS =

CC = gcc

EXTENSION = .c

%.o: %$(EXTENSION) $(DEPS)

$(CC) -c -o $@ $< $(CFLAGS)

main: $(OBJ)

$(CC) -o $@ $^ $(CFLAGS) $(LIBS)

.PHONY: clean

clean:

rm -f \*.o \*~ core \*~ main

## main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "functions.h"

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

char \* input\_filename, output\_filename[12];

FILE \* input\_file = NULL, \* output\_file = NULL;

char \*\* text, \*\* outtext, ans;

int i,lines;

if (argc > 1) {

input\_filename = argv[1];

input\_file = fopen(input\_filename, "r");

if (input\_file == NULL) {

if (menu(stdin,stdout,"Вы задали неверное имя файла. Хотите ввести данные вручную?\n1. Да\n2.Нет",2) == 2)

return 0;

return 1;

}

}

if (input\_file == NULL)

input\_filename = get\_inputfile(stdin,stdout,&input\_file);

text = get\_text(input\_file,&lines);

outtext = sort\_text(text,lines);

ans = menu(stdin,stdout,"Вывести:\n1. На экран\n2. В файл\n3. Оба варианта",3);

if (ans == 1 || ans == 3)

for (i=0;i<lines;i++)

printf("%s\n",outtext[i]);

if (ans == 2 || ans == 3) {

create\_outputfile(stdin,stdout,input\_filename,&output\_file,output\_filename);

if (output\_file == NULL) {

printf("Не получилось создать файл %s\n",output\_filename);

return 1;

}

for (i=0;i<lines;i++) {

fprintf(output\_file,"%s\n",outtext[i]);

}

fclose(output\_file);

printf("Выходной файл: %s\n",output\_filename);

}

// Exit

fclose(input\_file);

free(input\_filename);

for (i=0; i<lines; i++) {

free(text[i]);

}

free(text);

return 0;

}

## Functions.h

// This is start of the header guard

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

void say(FILE \* output, char \* text);

char \* get\_string (FILE \* input);

char \* get\_inputfile (FILE \* input, FILE \* output, FILE \*\* input\_file);

void create\_outputfile (FILE \* input, FILE \* output, char \* input\_filename, FILE \*\* output\_file, char \* output\_filename);

char \*\* get\_text (FILE \* input, int \* lines);

char \*\* sort\_text(char \*\* text, int lines);

char check\_line(char \* text);

void ign\_other(FILE \* input);

char menu (FILE \* input, FILE \* output, char \* message, char count);

// This is the end of the header guard

#endif

## Functions.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "functions.h"

#include <ctype.h>

void say (FILE \* pFile, char \* text) {

fprintf(pFile,"%s",text);

}

char \* get\_string (FILE \* input) {

char \*a = (char \*) calloc(1,sizeof(char)),c = 0;

int length = 0;

// While not end of line ask new character

while ((c = fgetc(input)) != EOF && c != '\n') {

// Increase length

length++;

// Increase length of input string

a = (char \*) realloc (a, (length + 1) \* sizeof(char));

// Copy character to string

a[length-1] = c;

// Get new character

}

a[length] = 0;

return a;

}

char \* get\_inputfile (FILE \* input, FILE \* output, FILE \*\* input\_file) {

char \* input\_filename;

while (\*input\_file == NULL) {

// Ask string

say(output,"Введите имя файла: ");

input\_filename = get\_string(input);

\*input\_file = fopen(input\_filename, "r");

if (\*input\_file == NULL) {

system("clear");

say(output,"Такого файла не существует. Попробуйте ещё\n");

}

}

return input\_filename;

}

void create\_outputfile (FILE \* input, FILE \* output, char \* input\_filename, FILE \*\* output\_file, char \* output\_filename) {

char i;

for (i=0; i<3 && input\_filename[i] != '.'; i++) {

output\_filename[i] = input\_filename[i];

}

output\_filename[i]=0;

strcat(output\_filename,"\_out.dat");

\*output\_file = fopen(output\_filename, "w");

}

char \*\* get\_text (FILE \* input, int \* lines) {

char \*\* text = (char \*\*) calloc(0,sizeof(char\*));

int length = 0;

while (!feof(input)) {

// Increase length

length++;

// Increase length of input string

text = (char \*\*) realloc (text, (length + 1) \* sizeof(char\*));

// Copy character to string

text[length-1] = get\_string(input);

}

\*lines = length;

return text;

}

char \*\* sort\_text(char \*\* text, int lines) {

char \*\* outtext = (char \*\*) calloc(lines,sizeof(char\*));

int x=0,i;

for (i=0; i<lines; i++)

if (check\_line(text[i])) {

outtext[x]=text[i];

x++;

}

for (i=0; i<lines; i++)

if (!check\_line(text[i])) {

outtext[x]=text[i];

x++;

}

return outtext;

}

char check\_line(char \* text) {

if (isalpha(text[0]) && !isalpha(text[1]) && text[1]!='-' && text[1]!='\'')

return 1;

else

for (int i = 1; text[i] && !isalpha(text[i-1]); i++)

if (isalpha(text[i]) && !isalpha(text[i+1]) && text[i+1]!='-' && text[i+1]!='\'')

return 1;

return 0;

}

/\* Ignore all data until end of line \*/

void ign\_other(FILE \* input) {

char c = 0;

// While current character is not new line

while (c!='\n')

// Get new character

c=fgetc(input);

}

char menu (FILE \* input, FILE \* output, char \* message, char count) {

// Main menu

char answer = 0;

// While answer is incorrect make menu

while (answer < 1 || answer > count) {

// Ask menu

printf("%s\n",message);

// Get answer

answer = getchar()-'0';

// Ignore all after answer to avoid incorrect data in future

ign\_other(input);

// ask to try again

if (answer < 1 || answer > count) {

system("clear");

printf("Вводить нужно только цифры от 1 до %hhi. Попробуйте ещё\n",count);

}

}

return answer;

}

1. Опис інтерфейсу програми:
   1. Запит файлу
   2. Вибір виходу за допомогою меню
   3. Вихід даних за вибором
2. Опис тестових прикладів:

## input.txt

abc

A B C

## Console

Введите имя файла: input.txt

Вывести:

1. На экран

2. В файл

3. Оба варианта

1

A B C

abc