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;
}
```

4. Опис інтерфейсу програми:
    1. Запит файлу
    2. Вибір виходу за допомогою меню
    3. Вихід даних за вибором
5. 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
```