Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 2

тема «Функции, массивы, структуры и рекурсия в языке программирования C»

по дисциплине «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-22-1б Богданов Д.О.

Проверил: Батин Сергей Евгеньевич

Пермь, 2023

**Содержание**

[Задание 1 3](#_Toc157534821)

[1.1. Постановка задачи 3](#_Toc157534822)

[1.2. Решение задачи, код программы 3](#_Toc157534823)

[1.3. Описание действий 4](#_Toc157534824)

[1.4. Тестирование работы программы 4](#_Toc157534825)

[Задание 2 5](#_Toc157534826)

[2.1. Постановка задачи 5](#_Toc157534827)

[2.2. Решение задачи, код программы 5](#_Toc157534828)

[2.3. Описание действий 6](#_Toc157534829)

[2.4. Тестирование работы программы 6](#_Toc157534830)

[Задание 3 7](#_Toc157534831)

[3.1. Постановка задачи 7](#_Toc157534832)

[3.2. Решение задачи, код программы 7](#_Toc157534833)

[3.3. Описание действий 8](#_Toc157534834)

[3.4. Тестирование работы программы 9](#_Toc157534835)

[Задание 4 11](#_Toc157534836)

[4.1. Постановка задачи 11](#_Toc157534837)

[4.2. Решение задачи, код программы 11](#_Toc157534838)

[4.3. Описание действий 13](#_Toc157534839)

[4.4. Тестирование работы программы 13](#_Toc157534840)

[Задание 5 14](#_Toc157534841)

[5.1. Постановка задачи 14](#_Toc157534842)

[5.2. Решение задачи, код программы 14](#_Toc157534843)

[5.3. Описание действий 16](#_Toc157534844)

[5.4. Тестирование работы программы 16](#_Toc157534845)

# Задание 1

## 1.1. Постановка задачи

Написать программу, которая генерирует массив из псевдослучайных целых чисел и выводит его в консоль в прямом порядке, и записывает в файл в обратном порядке.

## 1.2. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void write\_file(int n, int arr[], const char \*path)

{

FILE\* file;

file = fopen(path,"w");

if (file == NULL){

printf("Ошибка чтения файла\n");

return;

}

for(int i = n-1; i >= 0; i--)

fprintf(file, "%d ", arr[i]);

fclose(file);

}

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

{

if (argc <= 1) {

printf("Путь к файлу не указан, программа завершила действие");

return 1;

}

int n, \*arr;

srand(time(NULL));

printf("Введите размерность желаемого массива\n");

scanf("%d", &n);

arr = malloc(n \* sizeof(n));

if (arr == NULL) {

printf("Ошибка при выделении памяти\n");

return 1;

}

printf("Ваш массив:\n");

for (int i = 0; i < n; i++) {

arr[i] = rand() %100;

printf("%d ", arr[i]);

}

write\_file(n, arr, argv[1]);

free (arr);

return 0;

}

## 1.3. Описание действий

В программе используются 2 функции main и write\_file. В main проверка на наличие пути к файлу, задание сида для псевдослучайных чисел, создание динамического массива размером int от пользователя, при помощи malloc, вывод массива на экран. Write\_file открывает файл для записи, проверяет на успешное открытие, записывает обратным циклом эл. Массива.

## 1.4. Тестирование работы программы

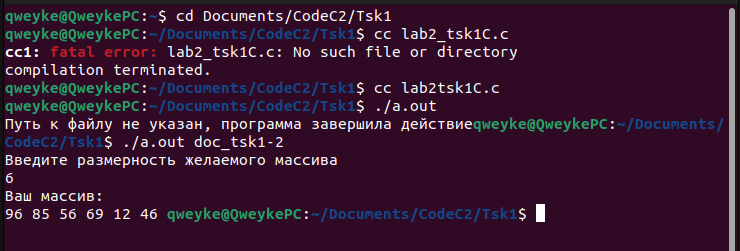


Рис.1 Тестирование программы lab2tsk1C.c

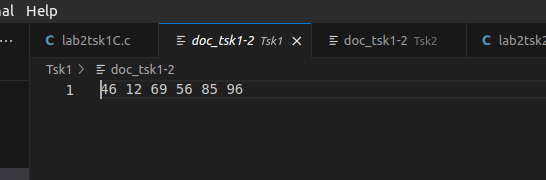


Рис.2 Тестирование программы lab2tsk1C.c

# Задание 2

## 2.1. Постановка задачи

Написать программу, которая принимает размер произвольного массива, создает его, заполняет случайными числами с плавающей точкой в интервале (0, 1). Выводит этот массив в консоль и записывает в файл в отсортированном виде.

## 2.2. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int compare(const void \*a, const void \*b)

{

if (\*(double\*)a == \*(double\*)b) return 0;

return (\*(double\*)a > \*(double\*)b) ? 1 : -1;

}

void write\_file(int n, double arr[], const char \*path)

{

FILE \*file;

file = fopen(path, "w");

if (file == NULL) {

printf("Ошибка чтения файла");

return;

}

qsort(arr, n, sizeof(double), compare);

for (int i = 0; i < n; i++)

fprintf(file, "%.3lf ", arr[i]);

fclose(file);

}

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

{

if (argc <= 1) {

printf("Не указан путь к файлу");

return 1;

}

int i, n;

double \*arr;

srand(time(NULL));

printf("Введите размерность массива\n");

scanf("%d", &n);

arr = malloc(n \* sizeof(double));

if (arr == NULL) {

printf("Ошибка выделения памяти массива");

return 1;

}

printf("Ваш массив:");

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

arr[i] = (double)rand()/ (double)RAND\_MAX;

printf("%.3lf ", arr[i]);

}

printf("\n");

write\_file(n, arr, argv[1]);

free(arr);

return 0;

}

## 2.3. Описание действий

В программе используются 3 функции main, write\_file, compare для квик сортировки. В main проверка на наличие пути к файлу, задание сида для псевдослучайных чисел, создание динамического массива размером int из элментов типа double (0;1) от пользователя, при помощи malloc, вывод массива на экран. Write\_file открывает файл для записи, проверяет на успешное открытие, записывает сортированные qsort эл. Массива.

## 2.4. Тестирование работы программы

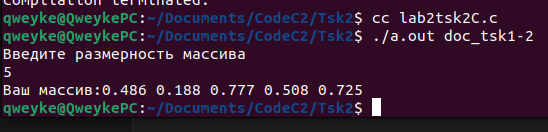


Рис.3 Тестирование программы lab2tsk2C.c

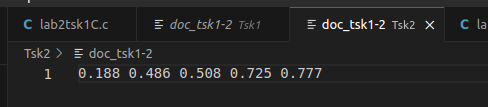


Рис.4 Тестирование программы lab2tsk2C.c

# Задание 3

## 3.1. Постановка задачи

Создать структуру для хранения данных о названии и цене товара. Считать из входного файла данные о товарах. Написать функцию для сортировки товаров по цене. Вывести отсортированные данные в выходной файл.

## 3.2. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

#include <float.h>

#define NAME\_LENGHT 30

struct goods {

char name[NAME\_LENGHT];

double price;

};

int compare(const void \*a, const void \*b)

{

double diff = ((struct goods \*)a)->price - ((struct goods \*)b)->price;

if (diff > DBL\_EPSILON) {

return 1;

} else if (diff < -DBL\_EPSILON) {

return -1;

} else {

return 0;

}

}

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

{

if (argc < 2) {

if (argc = 1){

printf("Путь к файлам не указан");

return 1;

}

printf("Путь к файлу 2 не указан");

return 1;

}

FILE \*file\_in, \*file\_out;

char \*path1 = argv[1];

char \*path2 = argv[2];

file\_in = fopen(path1, "r");

if (file\_in == NULL){

printf("Ошибка чтения файла\n");

return 1;

}

int n, cnt, previous = '\n';

while ((n = fgetc(file\_in)) != EOF) {

if (previous == '\n') cnt++;

previous = n;

}

rewind(file\_in);

struct goods \*goods\_l = ((struct goods\*) malloc(cnt \*sizeof(struct goods)));

int i = 0;

while (fscanf(file\_in, "%s %lf", goods\_l[i].name, &(goods\_l[i].price)) != EOF) {

printf("%s %.2lf\n", goods\_l[i].name,goods\_l[i].price);

i++;

}

fclose(file\_in);

file\_out = fopen(path2,"w");

if (file\_out == NULL) {

printf("Ошибка чтения файла 2");

return 0;

}

printf("\n");

qsort(goods\_l, cnt, sizeof(struct goods), compare);

for (int j = 0; j < cnt; j++) {

fprintf(file\_out, "%s %.2lf\n", goods\_l[j].name,goods\_l[j].price);

}

free(goods\_l);

fclose(file\_out);

printf("Файл успешно отсортирован\n");

return 0;

}

## 3.3. Описание действий

Первой составляющей программы является структура данных, описывающая две характеристики товара: строку наименования длиной из константы и цену товара с типом дабл. Основная функция main проверяет правильность указанного к файлу пути, открывает входной файл и считывает кол-во строк-позиций товара внутри него, выделяется память под структуры goods количеством, идентичным наименованиям товара, происходит заполнение и вывод содержимого файла. С помощью метода compare проводится qsort по цене товара от наименьшей к наибольшей, выходной файл заполнен, конец. Реализованы вспомогательные действия по закрытию, открытию, очистке памяти.

## 3.4. Тестирование работы программы

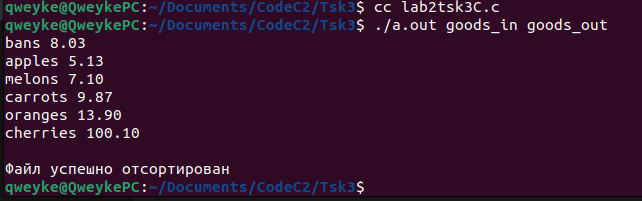


Рис.5 Тестирование программы lab2tsk3C.c

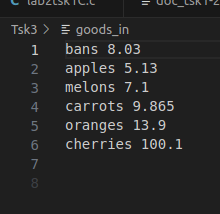


Рис.6 Тестирование программы lab2tsk3C.c, входной файл

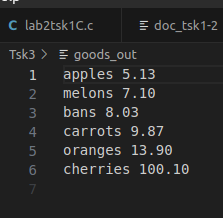


Рис.7 Тестирование программы lab2tsk3C.c, выходной файл

# Задание 4

## 4.1. Постановка задачи

Реализовать набор функций по работе со стеком. Для реализации стека воспользоваться структурой: следующий элемент стека хранит ссылку на предыдущий и значение.

· Реализовать добавление значения в стек.

· Реализовать извлечение значения из стека.

· Реализовать просмотр верхнего значения стека.

· Учесть при работе возможность переполнения стека.

## 4.2. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

typedef struct stack {

int data;

struct stack\* previous;

} STK;

STK\* push(STK\* top, int data)

{

STK\* elem = malloc(sizeof(STK));

elem->data = data;

elem->previous = top;

printf("Добавлен элемент %d\n \n", elem->data);

return elem;

}

STK\* pop(STK\* top)

{

if (top == NULL) {

printf("Нет элементов в стеке\n \n");

return top;

}

printf("Извлечен объект %d\n \n", top->data);

STK\* elem\_ntop = top->previous;

free(top);

return elem\_ntop;

}

int main(void)

{

int n, i, c, choice;

STK\* top = NULL;

do {

printf("Выберите вариант действия:\n \n");

printf("1 - добавить элемент(ы);\n");

printf("2 - снять элемент(ы);\n");

printf("3 - посмотреть верхушку;\n");

printf("4 - выход\n \n");

scanf(" %d", &choice);

switch(choice) {

case 1:

printf("Сколько элементов необходимо добавить в стэк:\n");

scanf("%d", &n);

printf("\n");

for (i = 1; i <= n; i++) {

top = push(top,i);

}

break;

case 2:

printf("Сколько элементов хотите извлечь? \n");

scanf("%d", &n);

printf("\n");

for (i = 1; i <= n; i++) {

top = pop(top);

}

break;

case 3:

printf("\n");

if (top == NULL){

printf("Объектов нет\n");

return 0;

}

printf("Объект %d \n", top->data);

break;

case 4:

printf("\n");

printf("Завершение программы\n");

break;

default:

printf("\n");

printf("Некорректиный выбор, попробуйте еще раз");

}

} while(choice != 4);

return 0;

}

## 4.3. Описание действий

Определяем структуру stack, которая будет хранить в себе число(данные) и ссылку на предыдущий элемент. Создает 2 функции – одна добавляет, другая снимает элемент из стэка. Первая – push выделяет память для очередной структуры, заполняет данные текущим числом, сдвигает предыдущий верхний эл., делая его предыдущим, возвращает новую структуру, определяя ее как верхний нынешний элемент (изначально top, верхний элемент, пуст = NULL). Pop, делает все то же самое, но меняет нынешний верхний эл. предыдущим. В функции main организован бесконечный цикл с выбором на свитче, есть действие по отображению верхнего текущего элемента.

## 4.4. Тестирование работы программы

|  |  |
| --- | --- |
|  |  |

Рис.8 Результат работы программы lab2tsk4C.c

# Задание 5

## 5.1. Постановка задачи

Написать программу, вычисляющую значение выражения, записанного в постфиксной ( , считываемого из входного файла. Считать, что выражение может содержать только цифры и знаки «+», «-», «\*» и «/». Примечание: для реализации воспользоваться функциями работы со стеком из предыдущего упражнения.

## 5.2. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

typedef struct stack {

int digit;

struct stack \*previous;

} STK;

STK\* push(STK\* top, int digit)

{

STK\* elem = malloc(sizeof(STK));

elem->previous = top;

elem->digit = digit;

return elem;

}

STK\* pop(STK\* top)

{

if (top == NULL) return top;

STK\* elem\_now = top->previous;

free(top);

return elem\_now;

}

STK\* show(const STK\* top)

{

const STK\* current = top;

while (current != NULL) {

printf("%d ", current->digit);

current = current->previous;

}

}

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

{

STK\* top = NULL;

FILE \*file;

char \*path = argv[1];

file = fopen(path, "r");

int chr;

while ((chr = fgetc(file)) != EOF) {

if (isdigit(chr)) {

top = push(top, chr - '0');

}

else if (chr == '+' || chr == '-' || chr == '\*' || chr == '/') {

if (top == NULL) {

printf("Операторы не введены\n");

return 1;

}

show(top);

printf("\n");

int oper1 = top->digit;

int oper2 = 0;

top = pop(top);

if (top != NULL) oper2 = top->digit;

top = pop(top);

int res;

switch (chr) {

case '+':

res = oper1 + oper2;

break;

case '-':

res = oper2 - oper1;

break;

case '\*':

res = oper1 \* oper2;

break;

case '/':

res = oper2 / oper1;

break;

}

top = push(top, res);

}

else if (chr != ' ' && chr != '\n') {

printf("Файл содержит некорректный тип символа: %c \n", chr);

continue;

}

}

fclose(file);

if (top != NULL) printf("Ответ (убил): %d ", top->digit);

else printf("Значений не было найдено");

return 0;

}

## 5.3. Описание действий

Создается стэк, как и в 4 задании. В функции main считывается посимвольно содержание входного файла, если программа считывает число, то записывает его, как элемент стэка, как только встречает знак – popает 2 верхних элемента и проводит арифметическую операцию с полученным символом. Чтобы получить из стэка 2 числа, сначала алгоритм берет текущее значение top эл. и записывает в oper1, при этом oper2 = 0 по умолчанию, снимает верхний эл. Далее, если верхний текущий элемент не пуст, то переписывает oper2 его значением, снимает элемент. Результат записывается как новый верхний элемент. И так далее до конца файла, после выводит итоговое значение. Реализована вспомогательная функция show, отображающая весь текущий состав стэка.

## 5.4. Тестирование работы программы

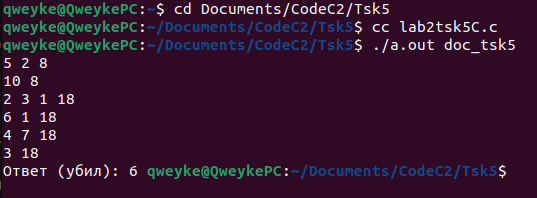


Рис.9 Результат работы программы lab2tsk5C.c

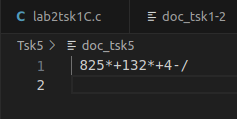


Рис.10 Результат работы программы lab2tsk5C.c