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 С»

по дисциплине «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-22-1б Зверев А.А.

Проверил: Батин Сергей Евгеньевич

Пермь, 2023

**Содержание**

[Задание 1 3](#__RefHeading___Toc5704_3569584462)

[1.1. Постановка задачи 3](#__RefHeading___Toc5706_3569584462)

[1.2. Решение задачи, код программы 3](#__RefHeading___Toc5708_3569584462)

[1.3. Тестирование работы программы 4](#__RefHeading___Toc5710_3569584462)

[Задание 2 4](#__RefHeading___Toc5712_3569584462)

[2.1. Постановка задачи 4](#__RefHeading___Toc5714_3569584462)

[2.2. Решение задачи, код программы 4](#__RefHeading___Toc5716_3569584462)

[2.3. Тестирование работы программы 6](#__RefHeading___Toc5718_3569584462)

[Задание 3 6](#__RefHeading___Toc5720_3569584462)

[3.1. Постановка задачи 6](#__RefHeading___Toc1057_2420135925)

[3.2. Решение задачи, код программы 6](#__RefHeading___Toc5724_3569584462)

[3.3. Тестирование работы программы 8](#__RefHeading___Toc5726_3569584462)

[Задание 4 9](#__RefHeading___Toc5728_3569584462)

[4.1. Постановка задачи 9](#__RefHeading___Toc5730_3569584462)

[4.2. Решение задачи, код программы 9](#__RefHeading___Toc5732_3569584462)

[4.3. Тестирование работы программы 10](#__RefHeading___Toc5734_3569584462)

[Задание 5 11](#__RefHeading___Toc5736_3569584462)

[5.1. Постановка задачи 11](#__RefHeading___Toc5738_3569584462)

[5.2. Решение задачи, код программы 11](#__RefHeading___Toc5740_3569584462)

[5.3. Тестирование работы программы 13](#__RefHeading___Toc5742_3569584462)

[Задание 6 13](#__RefHeading___Toc1059_2420135925)

[6.1. Постановка задачи 13](#__RefHeading___Toc5738_35695844621)

[6.2. Решение задачи, код программы 14](#__RefHeading___Toc5740_35695844621)

[6.3. Тестирование работы программы 16](#__RefHeading___Toc5742_35695844621)

# 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 randomArr(int arr[], int len){

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

arr[i] = rand() % 100 + 1;

}

}

void outputArr(int arr[], int len){

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

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

printf("\n");

}

printf("\n");

}

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

int arrlen;

printf("Input length of array: ");

scanf("%d", &arrlen);

if (arrlen <=0)

{

printf("Error of length of array");

return 1;

}

int \*newrandomArray = malloc(arrlen \* sizeof(int));

srand(time(NULL));

randomArr(newrandomArray, arrlen);

printf("Made array\n");

outputArr(newrandomArray, arrlen);

FILE \*file;

file = fopen (argv[1], "w");

if(file == NULL){

printf("ERROR of inputing file");

exit(EXIT\_FAILURE);

}

for(int i = arrlen - 1;i >= 0; i--){

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

fputs("\n", file);

}

fclose(file);

free(newrandomArray);

return 0;

}

## 1.3. Тестирование работы программы

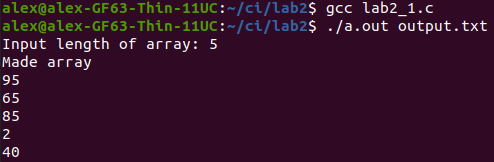


Рис.1.1 Тестирование программы lab2\_1.c

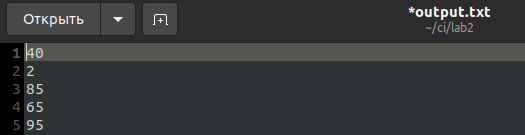


Рис. 1.2 Файл после выполнения программы

# 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>

double\* randomArr(int len){

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

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

// arr[i] = (double)rand();

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

}

return arr;

}

void outputArr(double arr[], int len){

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

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

printf("\n");

}

printf("\n");

}

int compar(const void \*a, const void \*b){

double double\_a = \*((double\*)a);

double double\_b = \*((double\*)b);

if (double\_a == double\_b) return 0;

else if (double\_a < double\_b) return -1;

else return 1;

}

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

{

int arrlen;

printf("Input length of array: ");

scanf("%d", &arrlen);

if (arrlen <=0)

{

printf("Error of length of array");

return 1;

}

double\* arr = randomArr(arrlen);

printf("Made array\n");

outputArr(arr, arrlen);

qsort(arr, arrlen, sizeof(double), compar);

FILE \*file;

file = fopen (argv[1], "w");

if(file == NULL){

printf("ERROR of inputing file");

exit(EXIT\_FAILURE);

}

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

fprintf(file, "%f\n", arr[i]);

}

fclose(file);

free(arr);

return 0;

}

## 2.3. Тестирование работы программы

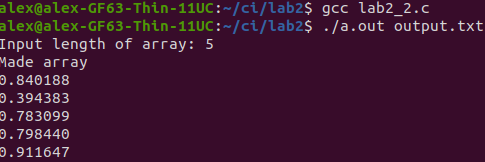


Рис.2.1 Тестирование программы lab2\_2.c

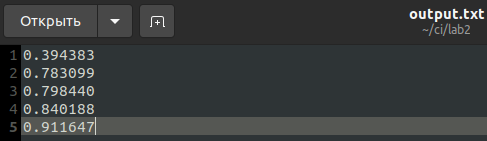


Рис. 2.2 Файл после выполнения программы

# 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>

struct item

{

char name[100];

double price;

};

int compar(const void \*a, const void \*b){

double price\_a = ((struct item\*)a)->price;

double price\_b = ((struct item\*)b)->price;

if (price\_a == price\_b) return 0;

else if (price\_a < price\_b) return -1;

else return 1;

}

int main(int argc, char const \*argv[2])

{

struct item items[100]; // less than 100 items

FILE \*file = fopen(argv[1], "r");

if (file == NULL) {

printf("ERROR of inputing file 1");

return -1;

//exit(EXIT\_FAILURE);

}

int counter = 0;

while (fscanf(file, "%1000s %lf", items[counter].name, &

items[counter].price) == 2 && counter < 100) {

counter++;

}

fclose(file);

if (counter > 0){

qsort(items, counter, sizeof(struct item), compar);

file = fopen(argv[2], "w");

if (file == NULL) {

printf("ERROR of inputing file 2");

return -1;

//exit(EXIT\_FAILURE);

}

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

fprintf(file, "%s %.2f\n", items[i].name, items[i].price);

}

fclose(file);

printf("Success\n");

}

else{

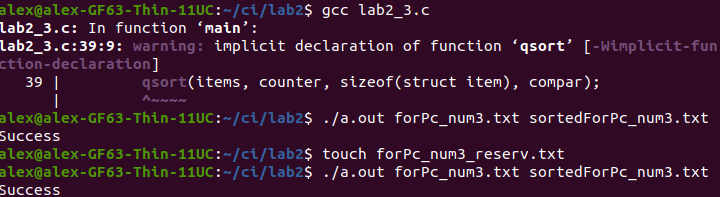
printf("Error of reading data from file 1");

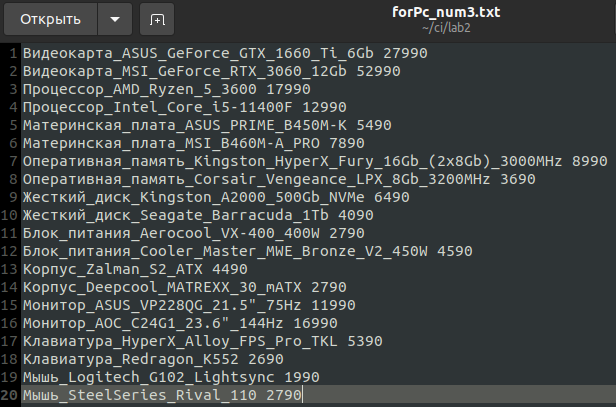
}

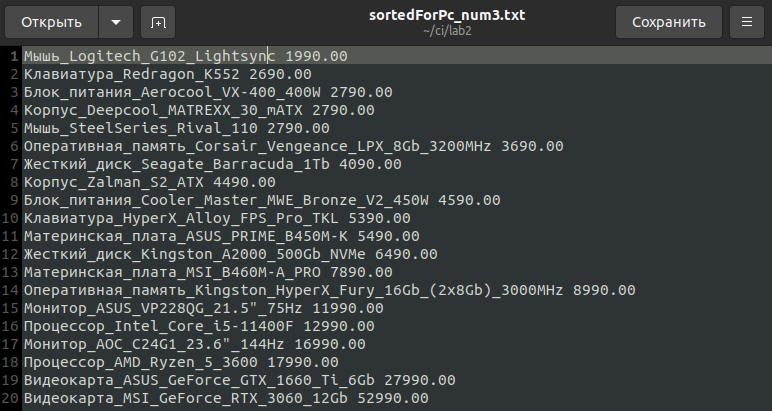
return 0;

}

## 3.3. Тестирование работы программы

Рис.3.1 Тестирование программы lab2\_3.c

Рис.3.2 Файл с данными

Рис.3.3 Файл результат работы программы (отсортированный)

# 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>

struct Node {

int data;

struct Node\* next;

};

struct Stack {

struct Node\* top;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Error of getting memory for node\n");

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

struct Stack\* createStack() {

struct Stack\* newStack = (struct Stack\*)malloc(sizeof(struct Stack));

if (newStack == NULL) {

printf("Error of getting memory for \n");

exit(EXIT\_FAILURE);

}

newStack->top = NULL;

return newStack;

}

// Функция для добавления элемента в стек

void push(struct Stack\* stack, int data) {

struct Node\* newNode = createNode(data);

newNode->next = stack->top;

stack->top = newNode;

}

// Функция для извлечения элемента из стека

int pop(struct Stack\* stack) {

if (stack->top == NULL) {

printf("Stack is free, can't use pop\n");

exit(EXIT\_FAILURE);

}

struct Node\* temp = stack->top;

int data = temp->data;

stack->top = temp->next;

free(temp);

return data;

}

// Функция для просмотра верхнего элемента стека

int peek(struct Stack\* stack) {

if (stack->top == NULL) {

printf("Stack is free, can't use peek\n");

exit(EXIT\_FAILURE);

}

return stack->top->data;

}

int main() {

struct Stack\* stack = createStack();

push(stack, 10);

push(stack, 20);

push(stack, 30);

push(stack, 40);

push(stack, 50);

printf("Upper element of stack: %d\n", peek(stack));

while (stack->top != NULL) {

printf("%d \n", pop(stack));

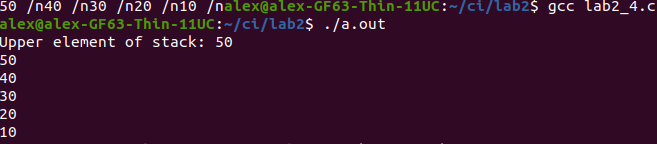
}

free(stack);

return 0;

}

## 4.3. Тестирование работы программы

Рис. 4 Тестирование программы lab2\_4.c

# Задание 5

## 5.1. Постановка задачи

Написать программу, вычисляющую значение выражения, записанного в постфиксной (обратной польской) записи, 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>

struct Node {

int data;

struct Node\* next;

};

struct Stack {

struct Node\* top;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Error of getting memory for node\n");

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

struct Stack\* createStack() {

struct Stack\* newStack = (struct Stack\*)malloc(sizeof(struct Stack));

if (newStack == NULL) {

printf("Error of getting memory for \n");

exit(EXIT\_FAILURE);

}

newStack->top = NULL;

return newStack;

}

// Функция для добавления элемента в стек

void push(struct Stack\* stack, int data) {

struct Node\* newNode = createNode(data);

newNode->next = stack->top;

stack->top = newNode;

}

// Функция для извлечения элемента из стека

int pop(struct Stack\* stack) {

if (stack->top == NULL) {

printf("Stack is free, can't use pop\n");

exit(EXIT\_FAILURE);

}

struct Node\* temp = stack->top;

int data = temp->data;

stack->top = temp->next;

free(temp);

return data;

}

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

FILE\* file = fopen(argv[1], "r");

if (file == NULL) {

printf("Error of inputing file\n");

return -1;

}

struct Stack\* stack = createStack();

char c;

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

if (isdigit(c)) {

push(stack, c - '0');

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

int operand2 = pop(stack);

int operand1 = pop(stack);

switch (c) {

case '+':

push(stack, operand1 + operand2);

break;

case '-':

push(stack, operand1 - operand2);

break;

case '\*':

push(stack, operand1 \* operand2);

break;

case '/':

push(stack, operand1 / operand2);

break;

}

}

}

fclose(file);

if (stack->top != NULL) {

int result = pop(stack);

printf("Result: %d\n", result);

} else {

printf("Error stack is free\n");

}

free(stack);

return 0;

}

## 5.3. Тестирование работы программы

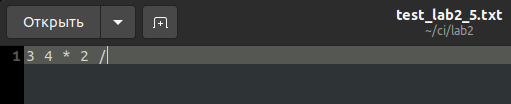
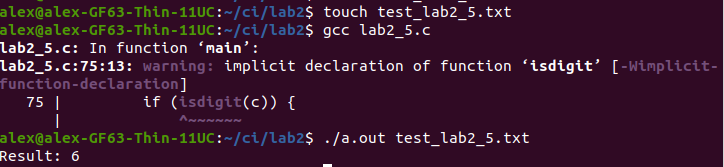


Рис. 5.1 Содержимое тестового файла test\_lab2\_5.txt

Рис. 5.2 Тестирование работы программы lab2\_5.c

# Задание 6

## 6.1. Постановка задачи

Написать программу, реализующую перевод инфиксной записи арифметического выражения в постфиксную. Исходное арифметическое выражение состоит из цифр, знаков «+», «-», «\*», «/» и скобок. При реализации воспользоваться предиктивным анализатором и схемой трансляции из лекции о простом однопроходном компиляторе. Леворекурсивная грамматика для арифметических выражений представлена на слайде 9 лекции. Для реализации программы следует на ее основе

· составить схему трансляции

· заменить леворекурсивную грамматику равнозначной праворекурсивной

· реализовать предиктивный анализатор по составленной грамматике

## 6.2. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

struct Stack {

char data;

struct Stack\* next;

};

struct Stack\* createNode(char data) {

struct Stack\* newNode = (struct Stack\*)malloc(sizeof(struct Stack));

if (newNode == NULL) {

printf("Error of getting memory for node\n");

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Функция для добавления элемента в стек

void push(struct Stack\*\* stack, char data) {

struct Stack\* newNode = createNode(data);

newNode->next = \*stack;

\*stack = newNode;

}

// Функция для извлечения элемента из стека

char pop(struct Stack\*\* stack) {

if (\*stack == NULL) {

printf("Stack is free, can't use pop\n");

exit(EXIT\_FAILURE);

}

struct Stack\* temp = \*stack;

char data = temp->data;

\*stack = temp->next;

free(temp);

return data;

}

// Функция для просмотра верхнего элемента стека

char peek(struct Stack\* stack) {

if (stack == NULL) {

printf("Stack is free, can't use peek\n");

exit(EXIT\_FAILURE);

}

return stack->data;

}

int main() {

char infixExpr[100];

printf("Input infix expression: ");

scanf("%s", infixExpr);

struct Stack\* oper = NULL;

int i = 0;

printf("Postfix expression: ");

while (infixExpr[i] != '\0') {

char c = infixExpr[i];

if (isdigit(c)) {

printf("%c ", c);

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

while (oper != NULL && (peek(oper) == '\*' || peek(oper) == '/') && (c == '+' || c == '-')) {

printf("%c ", pop(&oper));

}

push(&oper, c);

} else if (c == '(') {

push(&oper, c);

} else if (c == ')') {

while (oper != NULL && peek(oper) != '(') {

printf("%c ", pop(&oper));

}

pop(&oper);

}

i++;

}

while (oper != NULL) {

printf("%c ", pop(&oper));

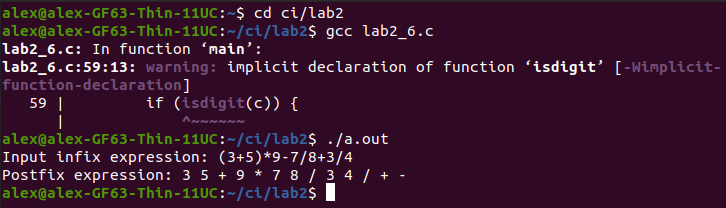
}

printf("\n");

return 0;

}

## 6.3. Тестирование работы программы

Рис. 6.1 Тестирование работы программы lab2\_6.c