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 Васин М.А.

Проверил: доц. Батин С.Е.

Пермь, 2023

**Содержание**

[Задание 1 3](#_Toc146190206)

[1.1. Постановка задачи 3](#_Toc146190207)

[1.2. Решение задачи, код программы 3](#_Toc146190208)

[1.3. Тестирование работы программы 3](#_Toc146190209)

[Задание 2 3](#_Toc146190210)

[2.1. Постановка задачи 3](#_Toc146190211)

[2.2. Решение задачи, код программы 3](#_Toc146190212)

[2.3. Тестирование работы программы 3](#_Toc146190213)

[Задание 3 3](#_Toc146190214)

[3.1. Постановка задачи 3](#_Toc146190215)

[3.2 Решение задачи, код программы 3](#_Toc146190216)

[3.3. Тестирование работы программы 3](#_Toc146190217)

[Задание 4 3](#_Toc146190218)

[4.1. Постановка задачи 3](#_Toc146190219)

[4.2. Решение задачи, код программы 3](#_Toc146190220)

[4.3. Тестирование работы программы 3](#_Toc146190221)

[Задание 5 4](#_Toc146190222)

[5.1. Постановка задачи 4](#_Toc146190223)

[5.2. Решение задачи, код программы 4](#_Toc146190224)

[5.3. Тестирование работы программы 4](#_Toc146190225)

# 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>

int main()

{

int n, m;

printf("enter n: ");

scanf("%d", &n);

printf("enter m: ");

scanf("%d", &m);

int array[n][m];

int i, j;

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

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

array[i][j] = rand()%100;

}

}

printf("array\n");

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

for(j=0; j<m; j++)

{

printf("%d ", array[i][j]);

}

printf("\n");

}

FILE \*file = fopen("z1.txt", "w");

if (file == NULL) {

perror("Error");

return 1;

}

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

for (int j = m - 1; j >= 0; j--) {

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

}

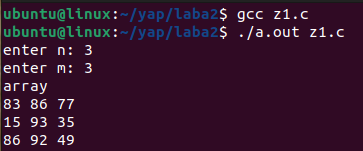
fprintf(file, "\n");

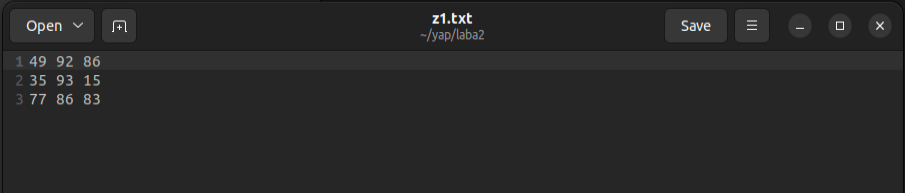
}

fclose(file);

}

## 1.3. Тестирование работы 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>

int main()

{

int n, m;

printf("enter n: ");

scanf("%d", &n);

printf("enter m: ");

scanf("%d", &m);

float array[n][m];

int i, j;

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

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

array[i][j] = (float)rand() / RAND\_MAX;

}

}

printf("array\n");

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

for(j=0; j<m; j++)

{

printf("%.6f ", array[i][j]);

}

printf("\n");

}

float flatArray[n \* m];

int index = 0;

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

for (j = 0; j < m; j++) {

flatArray[index++] = array[i][j];

}

}

// bubble sort

for (int i = 0; i < n \* m - 1; i++) {

for (int j = 0; j < n \* m - i - 1; j++) {

if (flatArray[j] > flatArray[j + 1]) {

float temp = flatArray[j];

flatArray[j] = flatArray[j + 1];

flatArray[j + 1] = temp;

}

}

}

printf("sorted array\n");

index = 0;

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

for (j = 0; j < m; j++) {

array[i][j] = flatArray[index++];

printf("%.6f ", array[i][j]);

}

printf("\n");

}

FILE \*file = fopen("z2.txt", "w");

if (file == NULL) {

fprintf(stderr, "error\n");

return 1;

}

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

for (j = 0; j < m; j++) {

fprintf(file, "%.6f ", array[i][j]);

}

fprintf(file, "\n");

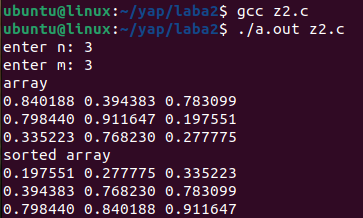
}

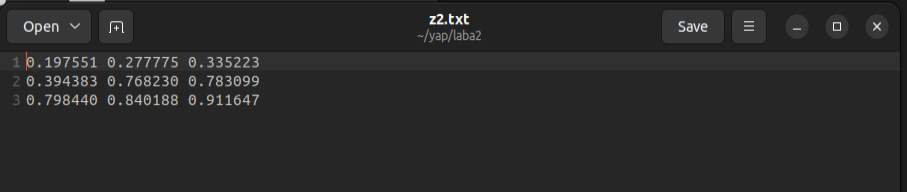
fclose(file);

return 0;

}

## 2.3. Тестирование работы 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>

#include <stdlib.h>

struct Product {

char name[50];

float price;

};

void sortProducts(struct Product \*products, int numProducts) {

for (int i = 0; i < numProducts - 1; i++) {

for (int j = 0; j < numProducts - i - 1; j++) {

if (products[j].price > products[j + 1].price) {

struct Product temp = products[j];

products[j] = products[j + 1];

products[j + 1] = temp;

}

}

}

}

int main() {

FILE \*inputFile = fopen("z3.txt", "r");

if (inputFile == NULL) {

fprintf(stderr, "error\n");

return 1;

}

int numProducts = 0;

char name[50];

float price;

while (fscanf(inputFile, "%s%f", name, &price) == 2) {

numProducts++;

}

struct Product \*products = (struct Product \*)malloc(numProducts \* sizeof(struct Product));

fseek(inputFile, 0, SEEK\_SET);

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

fscanf(inputFile, "%s%f", products[i].name, &products[i].price);

}

fclose(inputFile);

sortProducts(products, numProducts);

FILE \*outputFile = fopen("z3\_output.txt", "w");

if (outputFile == NULL) {

fprintf(stderr, "error\n");

free(products);

return 1;

}

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

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

}

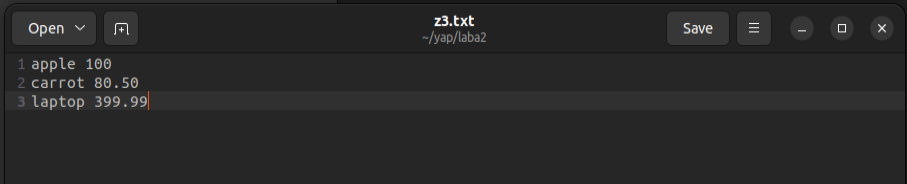
fclose(outputFile);

free(products);

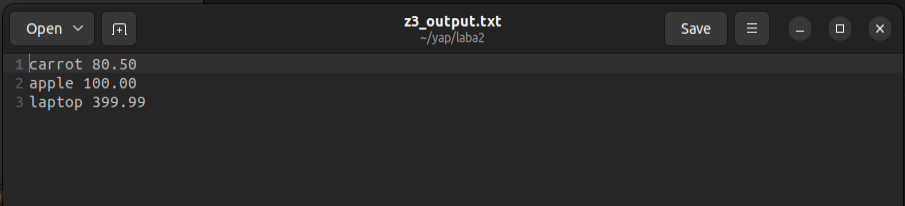
return 0;

}

## 3.3. Тестирование работы программы







# 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 Node {

int data;

struct Node\* next;

} Node;

typedef struct {

Node\* top;

int size;

int capacity;

} Stack;

void initializeStack(Stack\* stack, int capacity) {

stack->top = NULL;

stack->size = 0;

stack->capacity = capacity;

}

int isEmpty(Stack\* stack) {

return stack->size == 0;

}

int isFull(Stack\* stack) {

return stack->size == stack->capacity;

}

void push(Stack\* stack, int value) {

if (isFull(stack)) {

printf("stack is full\n");

return;

}

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

if (newNode == NULL) {

printf("error with new node\n");

return;

}

newNode->data = value;

newNode->next = stack->top;

stack->top = newNode;

stack->size++;

}

int pop(Stack\* stack) {

if (isEmpty(stack)) {

printf("stack is empty\n");

return -1;

}

Node\* temp = stack->top;

int value = temp->data;

stack->top = temp->next;

free(temp);

stack->size--;

return value;

}

int peek(Stack\* stack) {

if (isEmpty(stack)) {

printf("stack is empty\n");

return -1;

}

return stack->top->data;

}

void freeStack(Stack\* stack) {

while (!isEmpty(stack)) {

pop(stack);

}

}

int main() {

Stack myStack;

initializeStack(&myStack, 5);

push(&myStack, 10);

push(&myStack, 20);

push(&myStack, 30);

printf("upper value: %d\n", peek(&myStack));

printf("Extracted value: %d\n", pop(&myStack));

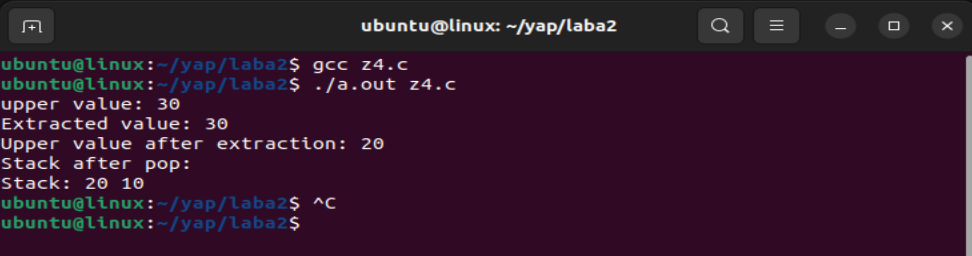
printf("Upper value after extraction: %d\n", peek(&myStack));

freeStack(&myStack);

return 0;

}

## 4.3. Тестирование работы программы



# 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>

#include <ctype.h>

typedef struct Node {

double data;

struct Node\* next;

} Node;

typedef struct {

Node\* top;

int size;

int capacity;

} Stack;

void initializeStack(Stack\* stack, int capacity) {

stack->top = NULL;

stack->size = 0;

stack->capacity = capacity;

}

int isEmpty(Stack\* stack) {

return stack->size == 0;

}

int isFull(Stack\* stack) {

return stack->size == stack->capacity;

}

void push(Stack\* stack, double value) {

if (isFull(stack)) {

printf("stack is full\n");

return;

}

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

if (newNode == NULL) {

printf("error with new node\n");

return;

}

newNode->data = value;

newNode->next = stack->top;

stack->top = newNode;

stack->size++;

}

double pop(Stack\* stack) {

if (isEmpty(stack)) {

printf("stack is empty\n");

return -1.0;

}

Node\* temp = stack->top;

double value = temp->data;

stack->top = temp->next;

free(temp);

stack->size--;

return value;

}

double evaluatePostfix(char\* expression) {

Stack stack;

initializeStack(&stack, 100);

while (\*expression) {

if (isdigit(\*expression) || (\*expression == '.')) {

push(&stack, strtod(expression, &expression));

} else {

double operand2 = pop(&stack);

double operand1 = pop(&stack);

switch (\*expression) {

case '+':

push(&stack, operand1 + operand2);

break;

case '-':

push(&stack, operand1 - operand2);

break;

case '\*':

push(&stack, operand1 \* operand2);

break;

case '/':

push(&stack, operand1 / operand2);

break;

default:

printf("Invalid character: %c\n", \*expression);

return -1.0;

}

}

expression++;

}

double result = pop(&stack);

if (!isEmpty(&stack)) {

printf("not enough operators\n");

return -1.0;

}

return result;

}

int main() {

FILE \*file = fopen("input.txt", "r");

if (file == NULL) {

perror("error");

return 1;

}

fseek(file, 0, SEEK\_END);

long fileSize = ftell(file);

rewind(file);

char \*expression = (char\*)malloc(fileSize + 1);

if (expression == NULL) {

perror("error");

fclose(file);

return 1;

}

fread(expression, 1, fileSize, file);

expression[fileSize] = '\0';

fclose(file);

double result = evaluatePostfix(expression);

if (result != -1.0) {

printf("result: %lf\n", result);

}

free(expression);

return 0;

}

## 5.3. Тестирование работы программы

