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-21-1б Егоров В.С.

Проверил: Батин С.Е.

Пермь, 2022

Содержание

[Задание 1 3](#_Toc117716082)

[1.1. Постановка задачи 3](#_Toc117716083)

[1.2. Код программы 3](#_Toc117716084)

[1.3. Пример работы программы 4](#_Toc117716085)

[Задание 2 5](#_Toc117716086)

[2.1. Постановка задачи 5](#_Toc117716087)

[2.2. Код программы 5](#_Toc117716088)

[2.3. Пример работы программы 6](#_Toc117716089)

[Задание 3 7](#_Toc117716090)

[3.1. Постановка задачи 7](#_Toc117716091)

[3.2. Код программы 7](#_Toc117716092)

[3.3. Пример работы программы 8](#_Toc117716093)

[Задание 4 10](#_Toc117716094)

[4.1. Постановка задачи 10](#_Toc117716095)

[4.2. Код программы 10](#_Toc117716096)

[4.3. Пример работы программы 11](#_Toc117716097)

[Задание 5 11](#_Toc117716098)

[5.1. Постановка задачи 11](#_Toc117716099)

[5.2. Код программы 11](#_Toc117716100)

[5.3. Пример работы программы 14](#_Toc117716101)

[Задание 6 15](#_Toc117716102)

[6.1. Постановка задачи 15](#_Toc117716103)

[6.2. Код программы 15](#_Toc117716104)

[6.3. Пример работы программы 17](#_Toc117716105)

# 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. Код программы

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

long int random**(**void**){**

long int systemTime **=** time**(NULL);**

long int operator **=** **(**846 **+** 327**)** **%** 1000**;**

long int rand **=** systemTime **\*** operator**;**

**return** rand**;**

**}**

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

int sizeMas**;**

printf**(**"Введите размер массива\n"**);**

scanf**(**"%d"**,** **&**sizeMas**);**

int mas**[**sizeMas**];**

**for(**int i **=** 0**;** i **<** sizeMas**;** i**++){**

mas**[**i**]** **=** random**();**

mas**[**i**]** **=** mas**[**i**]** **%** **(**long int**)** **(**437 **\*** **(**i **+** 59**)** **/** **(**i**+**1**))** **-** i**;**

printf**(**"mas[%d]=%d\n"**,** **(**i **+** 1**),** mas**[**i**]);**

**}**

FILE **\***file**;**

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

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

fprintf**(**file**,** "mas[%d]="**,** j **+** 1**);**

fprintf**(**file**,** "%d"**,** mas**[**j**]);**

fputs**(**"\n"**,** file**);**

**}**

fclose**(**file**);**

**return** 0**;**

**}**

## 1.3. Пример работы программы

На рис.1 представлен пример работы программы.

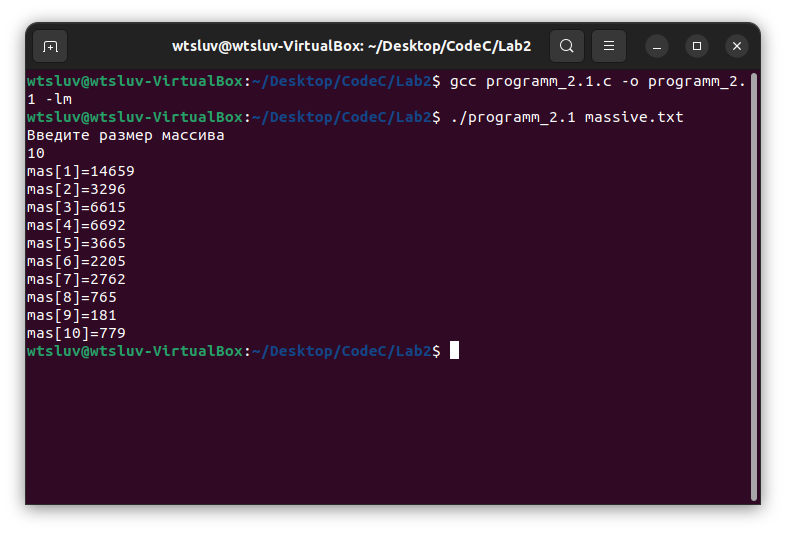


Рисунок 1. Пример работы программы из упражнения 1

На рис.2 представлен получившийся текстовый файл.

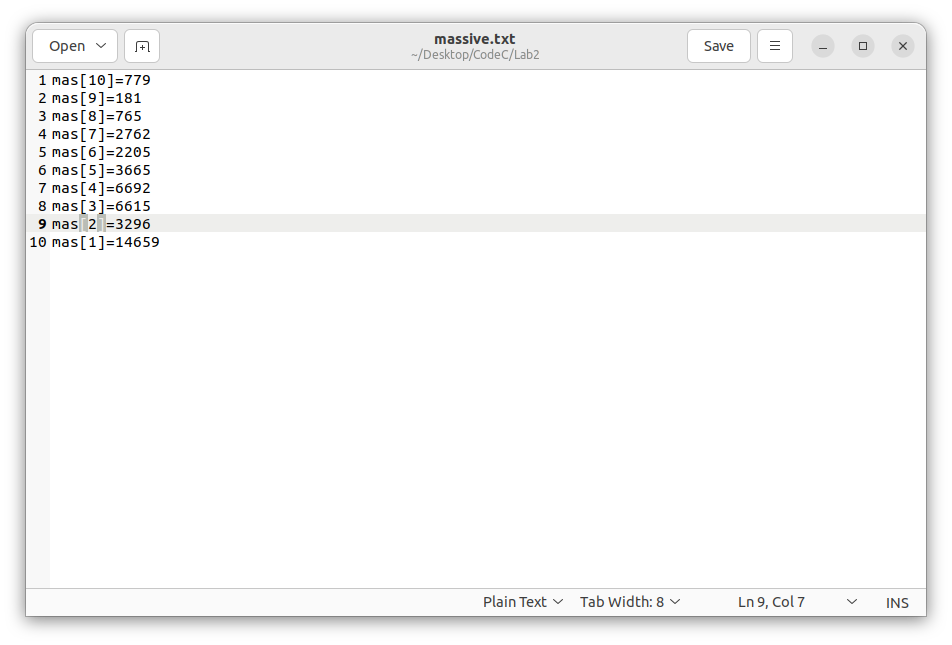


Рисунок 2. Массив в обратном порядке

# 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. Код программы

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int shakerSorting**(**double **\***mas**,** int sizeMas**){**

int left **=** 0**;**

int right **=** sizeMas **-** 1**;**

int k **=** 1**;**

**while((**left **<** right**)** **&** **(**k **>** 0**)){**

k **=** 0**;**

**for** **(**int i **=** left**;** i **<** right**;** i**++){**

**if** **(**mas**[**i**]** **>=** mas**[**i **+** 1**]){**

double auxiliaryMas1 **=** mas**[**i**];**

mas**[**i**]** **=** mas**[**i **+** 1**];**

mas**[**i **+** 1**]** **=** auxiliaryMas1**;**

k **=** 1**;**

**}**

**}**

right**--;**

**for(**int j **=** right**;** j **>** left**;** j**--){**

**if(**mas**[**j **-** 1**]** **>=** mas**[**j**]){**

double auxiliaryMas2 **=** mas**[**j**];**

mas**[**j**]** **=** mas**[**j **-** 1**];**

mas**[**j **-** 1**]** **=** auxiliaryMas2**;**

k **=** 1**;**

**}**

**}**

left**++;**

**}**

**}**

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

printf**(**"Введите размер массива:\n"**);**

int sizeMas**;**

scanf**(**"%d"**,** **&**sizeMas**);**

double mas**[**sizeMas**];**

printf**(**"Массив со случайными числами:\n"**);**

srand**(**time**(NULL));**

**for** **(**int i **=** 0**;** i **<** sizeMas**;** i**++){**

mas**[**i**]** **=** **(**float**)**rand**()** **/** **(**float**)**RAND\_MAX**;**

printf**(**"mas[%d]=%f\n"**,** i**,** mas**[**i**]);**

**}**

shakerSorting**(**mas**,** sizeMas**);**

FILE **\***file**;**

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

**for(**int i **=** 0**;** i **<** sizeMas**;** i**++){**

fprintf**(**file**,** "mas[%d]=%f\n"**,** i**,** mas**[**i**]);**

**}**

fclose**(**file**);**

**return** 0**;**

**}**

## 2.3. Пример работы программы

На рис.3 представлен пример работы программы.

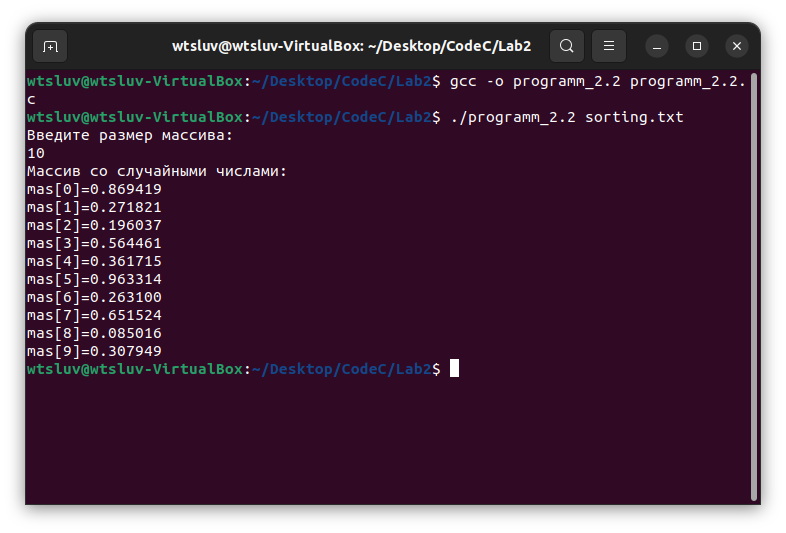


Рисунок 3. Пример работы программы из упражнения 2

На рис.4 представлен получившийся текстовый файл.

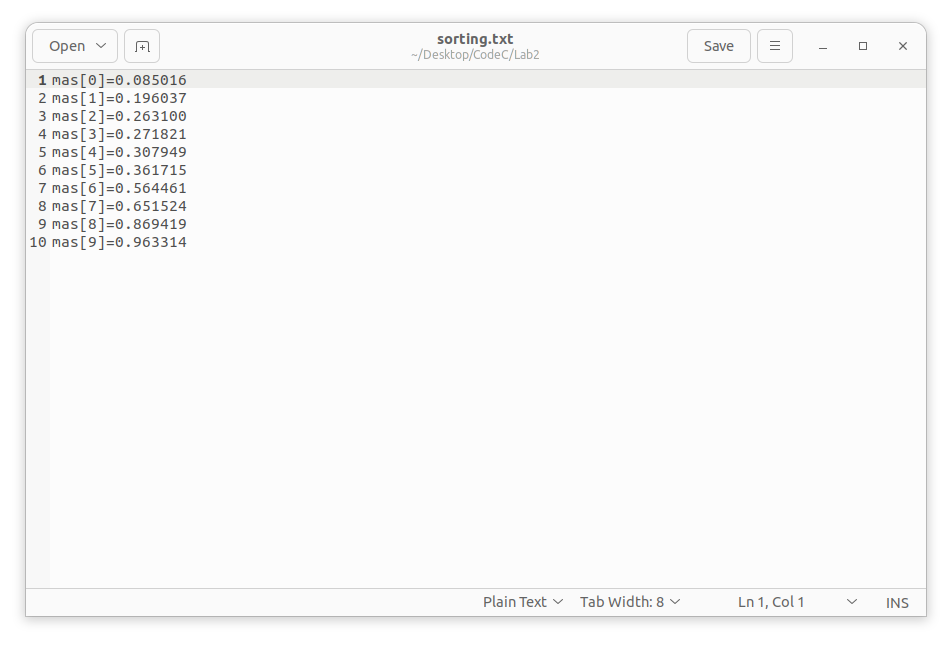


Рисунок 4. Отсортированный массив

# 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. Код программы

#include <stdio.h>

struct goods**{**

char name**[**128**];**

float cost**;**

**};**

int shakerSorting**(**struct goods **\***mas**,** int countString**){**

int left **=** 0**;**

int right **=** countString **-** 1**;**

int k **=** 1**;**

**while((**left **<** right**)** **&** **(**k **>** 0**)){**

k **=** 0**;**

**for** **(**int i **=** left**;** i **<** right**;** i**++){**

**if** **(**mas**[**i**].**cost **>=** mas**[**i **+** 1**].**cost**){**

struct goods tempCost1 **=** mas**[**i**];**

mas**[**i**]** **=** mas**[**i **+** 1**];**

mas**[**i **+** 1**]** **=** tempCost1**;**

k **=** 1**;**

**}**

**}**

right**--;**

**for(**int j **=** right**;** j **>** left**;** j**--){**

**if(**mas**[**j **-** 1**].**cost **>=** mas**[**j**].**cost**){**

struct goods tempCost2 **=** mas**[**j**];**

mas**[**j**]** **=** mas**[**j **-** 1**];**

mas**[**j **-** 1**]** **=** tempCost2**;**

k **=** 1**;**

**}**

**}**

left**++;**

**}**

**}**

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

**{**

FILE **\***inputFile**;**

FILE **\***outputFile**;**

inputFile **=** fopen**(**argv**[**1**],** "r"**);**

int countString**;**

**while(!**feof**(**inputFile**)){**

**if(**fgetc**(**inputFile**)** **==** ‘\n’**){**

countString**++;**

**}**

**}**

struct goods shop**[**countString**];**

int i **=** 0**;**

fseek**(**inputFile**,** 0**,** SEEK\_SET**);**

**while(!**feof**(**inputFile**)){**

fscanf**(**inputFile**,** "%s%f"**,** shop**[**i**].**name**,** **&**shop**[**i**].**cost**);**

i**++;**

**}**

fclose**(**inputFile**);**

shakerSorting**(**shop**,** countString**);**

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

**for(**int c **=** 0**;** c **<** countString**;** c**++){**

fprintf**(**outputFile**,** "%s %.2f\n"**,** shop**[**c**].**name**,** shop**[**c**].**cost**);**

**}**

fclose**(**outputFile**);**

printf**(**"Программа успешно выполнена!\n"**);**

**return** 0**;**

**}**

## 3.3. Пример работы программы

На рис.5 представлен пример работы программы.

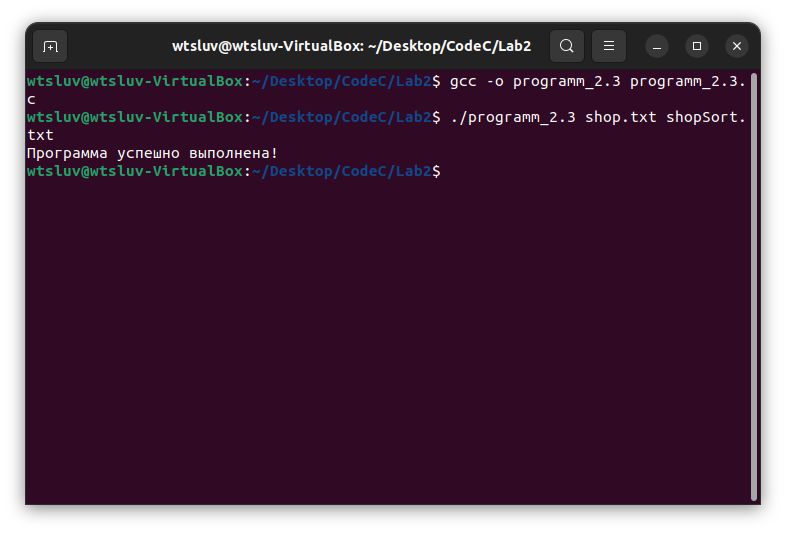


Рисунок 5. Пример работы программы из упражнения 3

На рис.6 представлен входной текстовый файл.



Рисунок 6. Входной текстовый файл из упражнения 3

На рис.7 представлен выходной текстовый файл

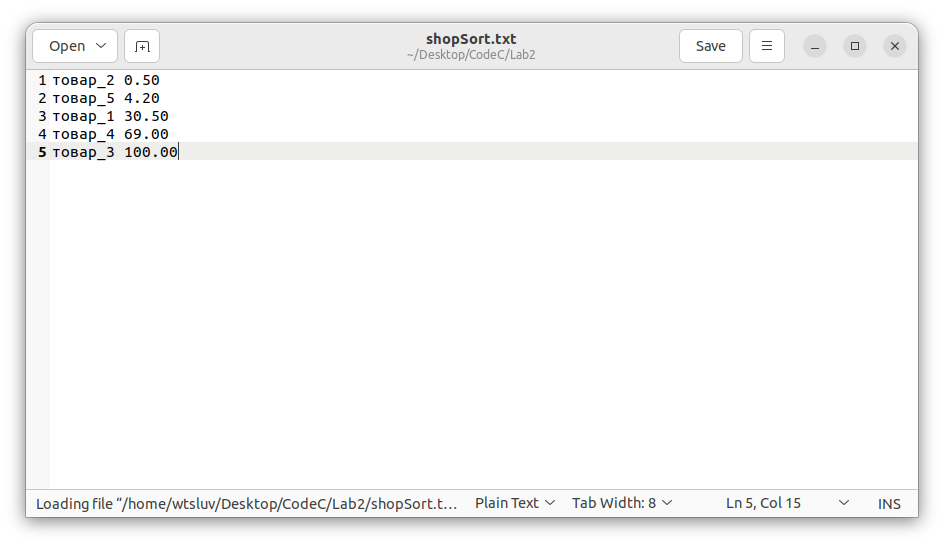


Рисунок 7. Выходной текстовый файл из упражнения 3

# Задание 4

## 4.1. Постановка задачи

Реализовать набор функций по работе со стеком:

* Организовать хранение стека в памяти.
* Реализовать добавление значения в стек.
* Реализовать извлечение значения из стека.
* Реализовать просмотр верхнего значения стека.
* Учесть при работе возможность переполнения стека.

## 4.2. Код программы

#include <stdio.h>

#include <stdlib.h>

#define size 5

struct stack**{**

int elem**[**size**];**

int count**;**

**};**

void push**(**struct stack **\***stk**,** int value**){**

**if** **(**stk**->**count **<** size**){**

stk**->**elem**[**stk**->**count**]** **=** value**;**

stk**->**count**++;**

**}**

**else{**

printf**(**"STACK\_OVERFLOW\n"**);**

**}**

**}**

int pop**(**struct stack **\***stk**){**

int value**;**

**if** **(**stk**->**count **>** 0**){**

stk**->**count**--;**

value **=** stk**->**elem**[**stk**->**count**];**

**return** value**;**

**}**

**else{**

printf**(**"STACK\_UNDERFLOW\_POP\n"**);**

**}**

**}**

int top**(**struct stack **\***stk**){**

**if((**stk**->**count**)** **>** 0**){**

**return** stk**->**elem**[**stk**->**count**-**1**];**

**}**

**else{**

printf**(**"STACK\_UNDERFLOW\_TOP\n"**);**

**}**

**}**

int main**(**void**){**

struct stack **\***stk**;**

stk **=** **(**struct stack**\*)** malloc**(sizeof(**struct stack**));**

printf**(**"Размер стека: %d\n"**,** size**);**

printf**(**"Пытаемся взять значение со стека, когда он еще пуст\n"**);**

pop**(**stk**);** // Для вывода ошибки STACK\_UNDERFLOW\_POP

printf**(**"\nПытаемся взять верхнее значение со стека, когда он еще пуст\n"**);**

top**(**stk**);** // Для вывода ошибки STACK\_UNDERFLOW\_TOP

int arr**[**5**]** **=** **{**1**,** 2**,** 3**,** 4**,** 5**};**

**for** **(**int i **=** 0**;** i **<** 5**;** i**++){**

push**(**stk**,** arr**[**i**]);**

**}**

printf**(**"\nЗапушили массив в стек:\n"**);**

**for** **(**int i **=** 0**;** i **<** 5**;** i**++){**

printf**(**"arr[%d]=%d\n"**,** i**,** arr**[**i**]);**

**}**

printf**(**"\nПытаемся запушить еще один элемент в стек, на которого нет места\n"**);**

push**(**stk**,** 6**);** //для вывода ошибки STACK\_OVERFLOW

printf**(**"\nПроверим верхнее значение стека: %d\n"**,** top**(**stk**));**

printf**(**"\nВозьмем это значение: %d\n"**,** pop**(**stk**));**

printf**(**"\nПроверим количество элементов в стекек: %d\n"**,** stk**->**count**);**

**return** 0**;**

**}**

## 4.3. Пример работы программы

На рис.8 представлен пример работы программы.

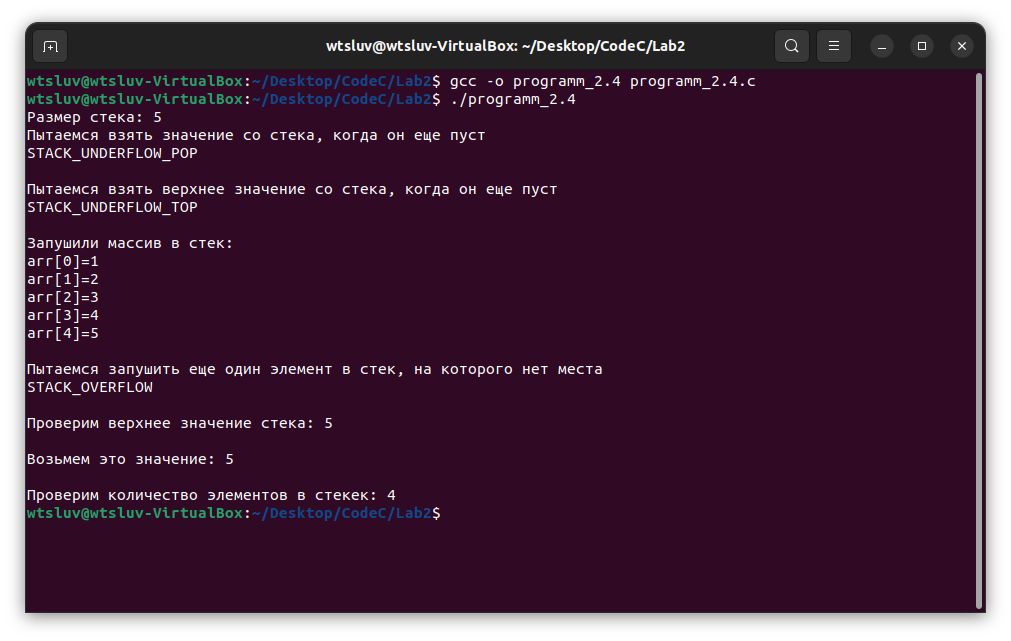


Рисунок 8. Пример работы программы

# 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 «/».

## 5.2. Код программы

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <ctype.h>

#define size 55

struct stack**{**

int elem**[**size**];**

int count**;**

**};**

void push**(**struct stack **\***stk**,** int value**){**

**if** **(**stk**->**count **<** size**){**

stk**->**elem**[**stk**->**count**]** **=** value**;**

stk**->**count**++;**

**}**

**else{**

printf**(**"STACK\_OVERFLOW\_PUSH\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**}**

int pop**(**struct stack **\***stk**){**

int value**;**

**if** **(**stk**->**count **>** 0**){**

stk**->**count**--;**

value **=** stk**->**elem**[**stk**->**count**];**

**return** value**;**

**}**

**else{**

printf**(**"STACK\_UNDERFLOW\_POP\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**}**

int top**(**struct stack **\***stk**){**

**if((**stk**->**count**)** **>** 0**){**

**return** stk**->**elem**[**stk**->**count**-**1**];**

**}**

**else{**

**return** 0**;**

printf**(**"STACK\_UNDERFLOW\_TOP\n"**);**

**}**

**}**

bool isempty**(**struct stack **\***stk**){**

**if** **(**stk**->**count **==** 0**){**

**return** **true;**

**}**

**else** **{**

**return** **false;**

**}**

**}**

//вычисление выражения в стеке

int operation**(**struct stack **\***stk**,** char operator**,** int i**){**

int secondValue**;**

int firstValue**;**

secondValue **=** pop**(**stk**);**

firstValue **=** pop**(**stk**);**

**switch** **(**operator**){**

**case** '+'**:**

push**(**stk**,** firstValue **+** secondValue**);**

**break;**

**case** '-'**:**

push**(**stk**,** firstValue **-** secondValue**);**

**break;**

**case** '\*'**:**

push**(**stk**,** firstValue **\*** secondValue**);**

**break;**

**case** '/'**:**

push**(**stk**,** firstValue **/** secondValue**);**

**break;**

**}**

**}**

bool isOperator**(**char operator**){**

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

**return** **true;**

**}**

**else{**

**return** **false;**

**}**

**}**

//вычисление выражения в постфиксной записи

int calculating**(**char **\***array**,** int row**,** struct stack **\***stk**){**

**if** **(!**isempty**(**stk**)){**

**while(!**isempty**(**stk**)){**

pop**(**stk**);**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** row**;** i**++){**

**if** **(**array**[**i**]** **==** '\n' **||** array**[**i**]** **==** '\0'**){**

printf**(**" = %d\n"**,** pop**(**stk**));** //для вывода

**continue;**

**}**

**else{**

**if** **(**isdigit**(**array**[**i**])){**

push**(**stk**,** array**[**i**]** **-** '0'**);** // array[i] - число, пушим в стек

**}**

**if** **(**isOperator**(**array**[**i**])){**

operation**(**stk**,** array**[**i**],** i**);** // array[i] - операнд, выполняем операцию

**}**

**}**

printf**(**"%c"**,** array**[**i**]);** //для вывода

**}**

**}**

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

FILE **\***file**;**

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

//определение количества символов в тексте = размер массива (последний элемент в массиве: '\0')

int col **=** 0**;**

**for** **(**char c **=** fgetc**(**file**);** c **!=** EOF**;** c **=** fgetc**(**file**)){**

col**++;**

**}**

char array**[**col**];**

//чтение и запись

fseek**(**file**,** 0**,** SEEK\_SET**);**

int i **=** 0**;**

**while** **(!**feof**(**file**)){**

array**[**i**]** **=** fgetc**(**file**);**

i**++;**

**}**

fclose**(**file**);**

struct stack **\***stk**;**

stk **=** **(**struct stack**\*)** malloc**(sizeof(**struct stack**));** //выделяем память под стек

char **\***p\_a **=** **&**array**[**0**];**

calculating**(**p\_a**,** col**,** stk**);**

free**(**stk**);**

**}**

## *5.3. Пример работы программы*

На рис.9 изображен исходный файл:

98+

84/

347-+

22\*

24+5\*

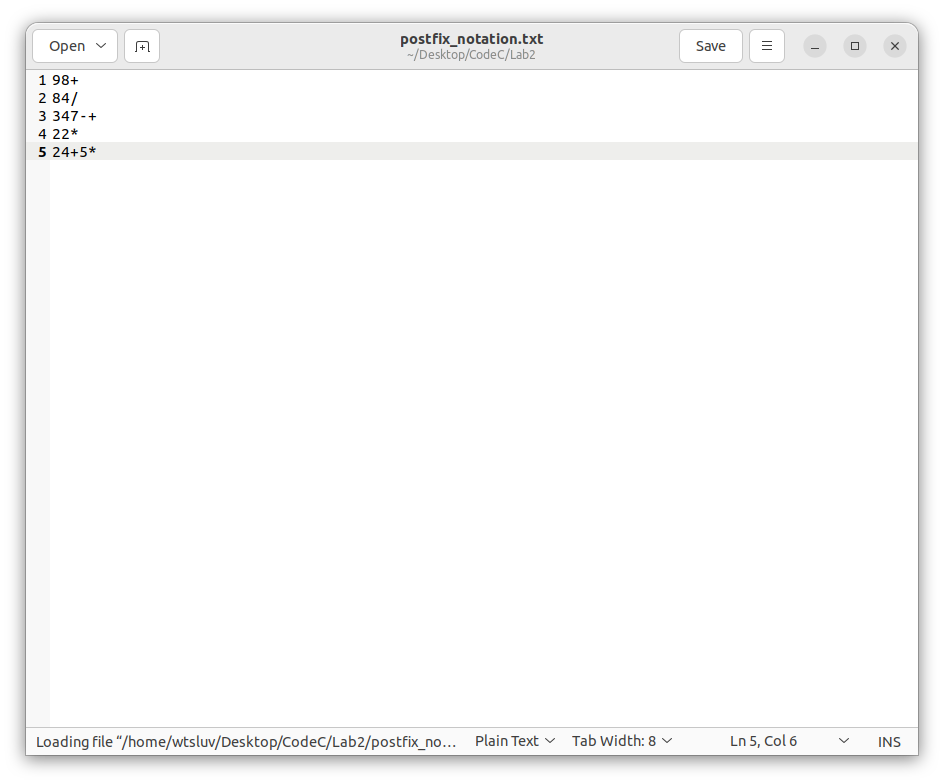


Рисунок 9. Входной файл

На рис.10 изображен пример работы программы:

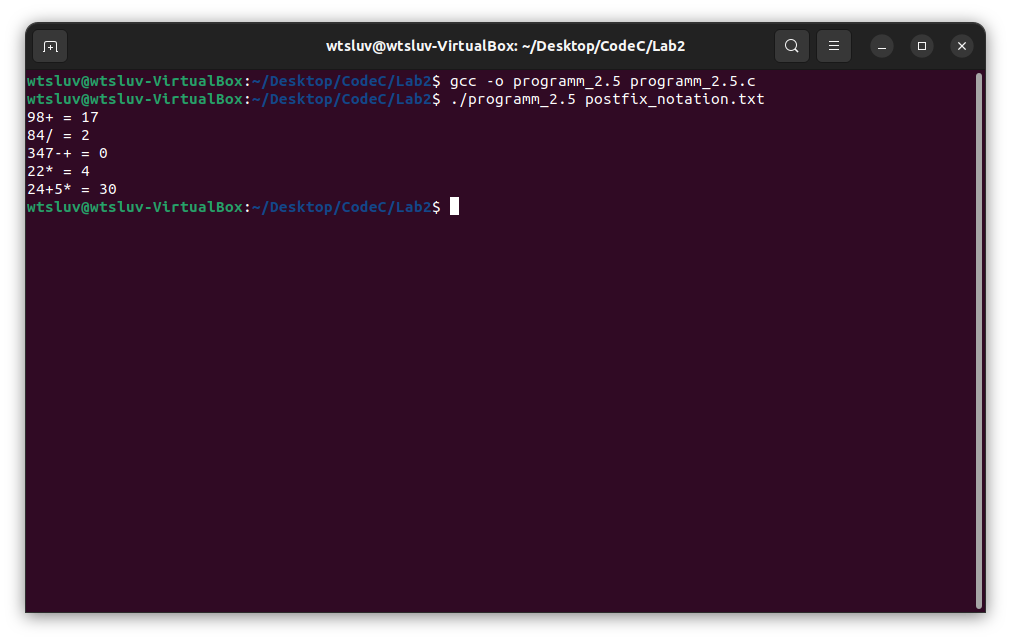


Рисунок 10. Пример работы программы

# 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.

## 6.2. Код программы

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <ctype.h>

#define size 55

struct stack**{**

char elem**[**size**];**

int count**;**

**};**

void push**(**struct stack **\***stk**,** char value**){**

**if** **(**stk**->**count **<** size**){**

stk**->**elem**[**stk**->**count**]** **=** value**;**

stk**->**count**++;**

**}**

**else{**

printf**(**"STACK\_OVERFLOW\_PUSH\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**}**

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

char value**;**

**if** **(**stk**->**count **>** 0**){**

stk**->**count**--;**

value **=** stk**->**elem**[**stk**->**count**];**

**return** value**;**

**}**

**else{**

printf**(**"STACK\_UNDERFLOW\_POP\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**}**

char top**(**struct stack **\***stk**){**

**if((**stk**->**count**)** **>** 0**){**

**return** stk**->**elem**[**stk**->**count**-**1**];**

**}**

**}**

bool isempty**(**struct stack **\***stk**){**

**if** **(**stk**->**count **==** 0**){**

**return** **true;**

**}**

**else** **{**

**return** **false;**

**}**

**}**

int isOperator\_Priority**(**char operator**){**

**if(**isdigit**(**operator**)){**

**return** 4**;**

**}**

**switch(**operator**){**

**case** '\*'**:**

**case** '/'**:**

**return** 3**;**

**break;**

**case** '+'**:**

**case** '-'**:**

**return** 2**;**

**break;**

**case** '('**:**

**return** 0**;**

**break;**

**case** ')'**:**

**return** 1**;**

**break;**

**default:**

**return** **-**1**;**

**break;**

**}**

**}**

int main**(**void**){**

printf**(**"Введите выражение в инфиксном виде:\n"**);**

struct stack **\***stk **=** **(**struct stack**\*)** **(**malloc**(sizeof(**struct stack**)));**

char buffer**[**64**];**// массив для считывания ввода

char outputString**[**64**];**// массив для записи постфиксного выражения

scanf**(**"%s"**,** buffer**);**

int oBrace **=** 0**;**

int eBrace **=** 0**;**

**for(**int i **=** 0**;** i **<** 64**;** i**++){**

**if(**buffer**[**i**]** **==** '('**){**

oBrace**++;**// считаем количество открывающихся скобок

**if(**isOperator\_Priority**(**buffer**[**i **+** 1**])** **==** 2 **||** isOperator\_Priority**(**buffer**[**i **+** 1**])** **==** 3**){**

printf**(**"Syntax error: there should be no operators after the opening brace\n"**);**

exit**(**EXIT\_FAILURE**);**//не должно быть операторов после открывающейся скобки

**}**

**}**

**if(**buffer**[**i**]** **==** ')'**){**

eBrace**++;**// считаем количество закрывающихся скобок

**if(**isOperator\_Priority**(**buffer**[**i **-** 1**])** **==** 2 **||** isOperator\_Priority**(**buffer**[**i **-** 1**])** **==** 3**){**

printf**(**"Syntax error: there should be no operators before the closing brace\n"**);**

exit**(**EXIT\_FAILURE**);**//не должно быть операторов перед закрывающейся скобкой

**}**

**}**

**if(**isOperator\_Priority**(**buffer**[**0**])** **==** 2 **||** isOperator\_Priority**(**buffer**[**0**])** **==** 3**){**

printf**(**"Syntax error: the first character should not be the operator\n"**);**

exit**(**EXIT\_FAILURE**);**//первый символ не должен быть оператором

**}**

**if(**isdigit**(**buffer**[**i**])){**

**if(**isOperator\_Priority**(**buffer**[**i**+**1**])** **==** 4**){**

printf**(**"Syntax error: the digit must not be followed by a digit\n"**);**

exit**(**EXIT\_FAILURE**);**//число не должно стоять после числа

**}**

**}**

**if(**isOperator\_Priority**(**buffer**[**i**])** **==** 2 **||** isOperator\_Priority**(**buffer**[**i**])** **==** 3**){**

**if(**isOperator\_Priority**(**buffer**[**i **+** 1**])** **==** 2 **||** isOperator\_Priority**(**buffer**[**i **+** 1**])** **==** 3**){**

printf**(**"Syntax error: there should be no operator after the operator\n"**);**

exit**(**EXIT\_FAILURE**);**// не должно быть оператора после оператора

**}**

**}**

**if(**buffer**[**i**]** **==** '\0'**){**// последний символ не должен быть оператором

**if(**isOperator\_Priority**(**buffer**[**i **-** 1**])** **==** 3 **||** isOperator\_Priority**(**buffer**[**i **+** 1**])** **==** 2**){**

printf**(**"Syntax error: the last character must not be an operator\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**}**

**}**

**if(**oBrace **!=** eBrace**){** // проверяем, чтобы количество скобок было одинаковым

printf**(**"Syntax error: the number of opening and closing brace is not equal.\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

int j **=** 0**;**

int i **=** 0**;**

int k**;**

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

//printf("in1\n");

int priority **=** isOperator\_Priority**(**buffer**[**i**]);**

//printf("priority:%d\n", priority);

//printf("in2\n");

**switch(**priority**){**

**case** 2**:**

**case** 3**:**

//printf("case23\n");

//printf("%c\n", buffer[i]);

//printf("priority\_top:%d\n",isOperator\_Priority(top(stk), i));

//printf("top:%c\n", top(stk));

**if(**isempty**(**stk**)){**

push**(**stk**,** buffer**[**i**]);**

**}**

**else{**

//printf("priority\_while:%d\n", priority);

**while(**isOperator\_Priority**(**top**(**stk**))** **>=** priority**){**

outputString**[**i **+** j**]** **=** pop**(**stk**);**

j**++;**

**}**

push**(**stk**,** buffer**[**i**]);**

**}**

**break;**

**case** 0**:**

//printf("case0\n");

push**(**stk**,** buffer**[**i**]);**

**break;**

**case** 1**:**

//printf("case1\n");

**while(**top**(**stk**)** **!=** '('**){**

outputString**[**i **+** j**]** **=** pop**(**stk**);**

j**++;**

**}**

pop**(**stk**);**

**break;**

**case** 4**:**

//printf("case4\n");

outputString**[**i **+** j**]** **=** buffer**[**i**];**

**break;**

**case** **-**1**:**

printf**(**"Unknown syntax\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

//printf("%c\n", outputString[i + j]);

i**++;**

k **=** i **+** j**;**

**}**

**while(!**isempty**(**stk**)){**

outputString**[**k**]** **=** pop**(**stk**);**

k**++;**

**}**

**for** **(**int i **=** 0**;** i **<=** k**;** i**++){**

printf**(**"%c"**,** outputString**[**i**]);**

**}**

printf**(**"\n"**);**

**}**

## 6.3. Пример работы программы

На рис.11 изображен пример работы программы.

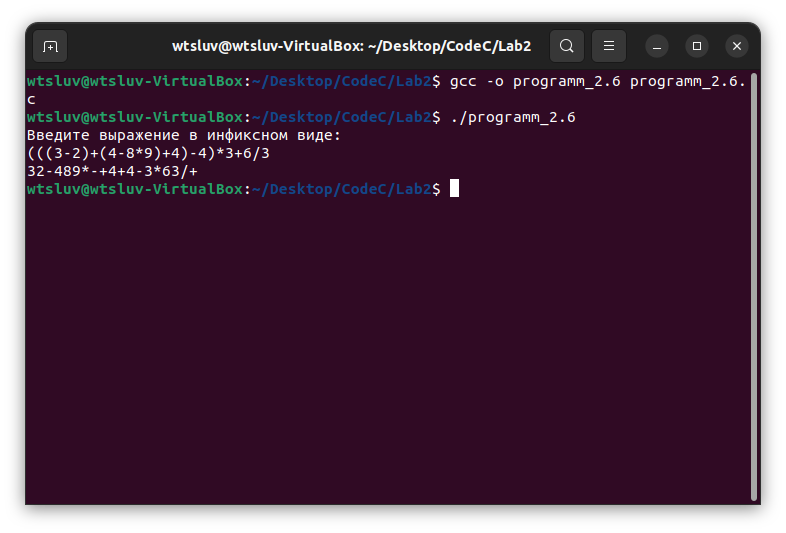


Рисунок 11. Пример работы программы