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 Носов Д.О.

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 5](#__RefHeading___Toc5712_3569584462)

[2.1. Постановка задачи 5](#__RefHeading___Toc5714_3569584462)

[2.2. Решение задачи, код программы 5](#__RefHeading___Toc5716_3569584462)

[2.3. Тестирование работы программы 7](#__RefHeading___Toc5718_3569584462)

[Задание 3 7](#__RefHeading___Toc5720_3569584462)

[3.1. Постановка задачи 7](#__RefHeading___Toc1057_2420135925)

[3.2. Решение задачи, код программы 8](#__RefHeading___Toc5724_3569584462)

[3.3. Тестирование работы программы 9](#__RefHeading___Toc5726_3569584462)

[Задание 4 10](#__RefHeading___Toc5728_3569584462)

[4.1. Постановка задачи 10](#__RefHeading___Toc5730_3569584462)

[4.2. Решение задачи, код программы 10](#__RefHeading___Toc5732_3569584462)

[4.3. Тестирование работы программы 13](#__RefHeading___Toc5734_3569584462)

[Задание 5 13](#__RefHeading___Toc5736_3569584462)

[5.1. Постановка задачи 13](#__RefHeading___Toc5738_3569584462)

[5.2. Решение задачи, код программы 13](#__RefHeading___Toc5740_3569584462)

[5.3. Тестирование работы программы 16](#__RefHeading___Toc5742_3569584462)

[Задание 6 16](#__RefHeading___Toc1059_2420135925)

[6.1. Постановка задачи 16](#__RefHeading___Toc5738_35695844621)

[6.2. Решение задачи, схема трансляции 17](#__RefHeading___Toc5740_35695844621)

[6.3. Решение задачи, код программы 17](#__RefHeading___Toc5740_356958446212)

[6.4. Тестирование работы программы 19](#__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>

int\* randomIntArr(int len){

//Резервируем память для массива чисел

int\* mas = (int\*)malloc(len\*sizeof(int));

//Генератор массива псевдослучайных чисел

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

int num = rand();

//Вывод чисел в терминал

printf("%d\n", num);

mas[i] = num;

}

return mas;

}

int sizeofNum(int num){ //Счетчик цифр в числе

int numb = num;

int count = 0;

while(numb != 0){

numb /= 10;

count++;

}

return count;

}

int main () {

//Динамическое изменение последовательности

//псевдослучайных чисел зависящих от кол-ва секунд

//прошедших с 01.01.1970

srand(time(NULL));

int len = rand()%10;

int\* mas = randomIntArr(len);

//Проверка на открытие файла для записи

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

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

//Считывание данных с массива в обратном порядке

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

//Счетчик цифр в числе

int count = sizeofNum(mas[i]);

//Перевод числа mas[i] в массив символов num\_char

char num\_char[count];

sprintf(num\_char, "%d", mas[i]);

//Посимвольная запись чисел в файл

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

fputc(num\_char[j],file);

}

fprintf(file, "\n");

}

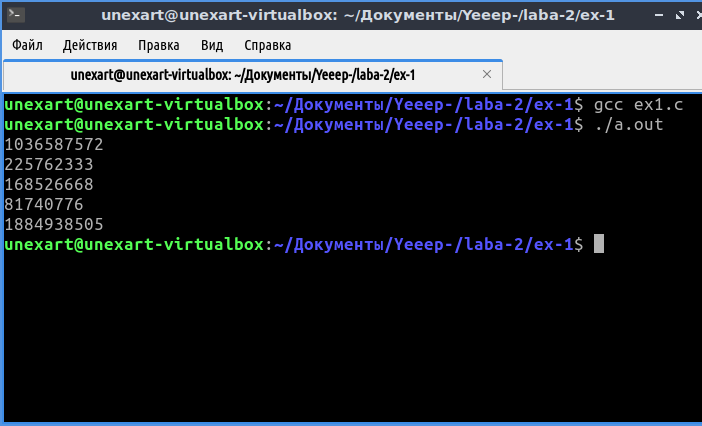
}

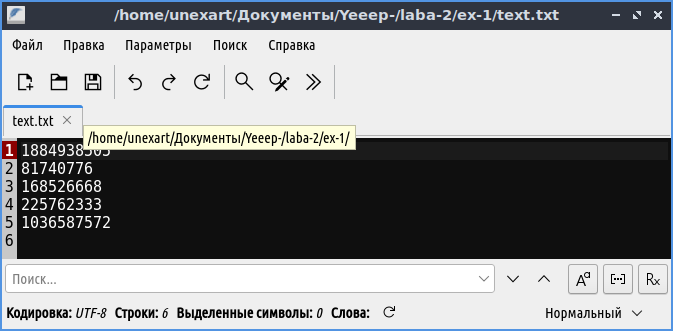
fclose(file);

free(mas);

}

## 1.3. Тестирование работы программы

Рис.1.1 Тестирование программы ex1.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>

//сортировка пузырьком

void mas\_sort(float\* mas, int len){

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

for(int j = i+1; j < len; j++){

if(mas[j] < mas[i]){

float min = mas[j];

mas[j] = mas[i];

mas[i] = min;

}

}

}

}

//метод возвращающий массив случацных чисел в окрестности (0;1)

float\* randomFloatArr(int len){

//Динамическое изменение последовательности

//псевдослучайных чисел зависящих от кол-ва секунд

//прошедших с 01.01.1970

srand(time(NULL));

float \*mas = (float\*)malloc(len \* sizeof(float)); //Резервирование памяти для массива

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

//Запись в массив случайных чисел от 0 до 1

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

}

return mas;

}

void main(){

printf("Введите размерность массива: ");

int len;

scanf("%d", &len);

float\* mas = randomFloatArr(len);

mas\_sort(mas,len);

//Проверка открытия файла

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

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

//Перебор значений массива mas

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

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

float num = mas[i];

int count1 = 0;//Счетчик символов после запятой

int count2 = 0;//Счетчик цифр после запятой

while(num != (int)num){

num \*= 10;

count1++;

}

int num2 = (int)num;

while(num2 != 0){

num2 /= 10;

count2++;

}

//Перевод числа num в массив символов num\_char

char num\_char[count2];

sprintf(num\_char, "%d", (int)num);

//Запись символов начала float

fputc('0',file);

fputc('.',file);

//Запись нулей поле символа "."

for (int j = 0; j < count1-count2; j++){

fputc('0',file);

}

//Запись чисел в массив

for (int j = 0; j < count2; j++){

fputc(num\_char[j],file);

}

fprintf(file, "\n");

}

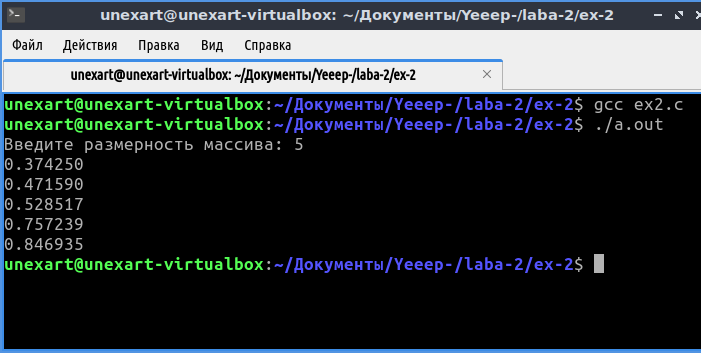
}

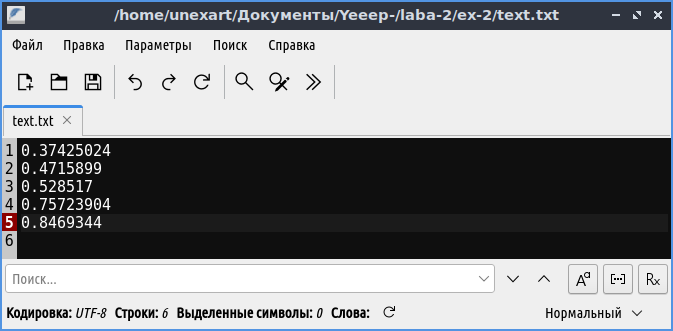
fclose(file);

free(mas);

}

## 2.3. Тестирование работы программы

Рис.2.1 Тестирование программы ex2.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>

#include <ctype.h>

#define n 100

//структура наименования и цены товара

struct product{

double price;

char name[n];

};

//метод посчета строчек в файле

int numb\_of\_strings(char \* filename){

FILE \*file = fopen(filename, "r");

int count = 0;

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

char ch;

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

if(ch == '\n'){

count++;

}

}

}

fclose(file);

return count;

}

//сортировка пузырьком

void sort(struct product items[]){

int len = numb\_of\_strings("text.txt");

struct product tmp;

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

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

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

tmp = items[j];

items[j] = items[j+1];

items[j+1] = tmp;

}

}

}

}

void main(){

int num\_strings = numb\_of\_strings("text.txt");

//создание массива структур

struct product items[num\_strings];

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

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

//чтение файла и запись информации в структуру

for(int i = 0; i < num\_strings; i++){

fscanf(file, "%100s %lf", items[i].name, & items[i].price);

}

fclose(file);

}

file = fopen("text1.txt", "w");

//сортировка массива структур

sort(items);

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

//запись отсортированных данных в файл

for (int i = 0; i < num\_strings; i++) {

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

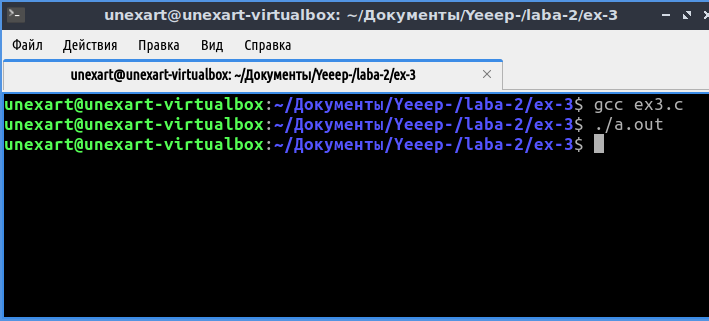
}

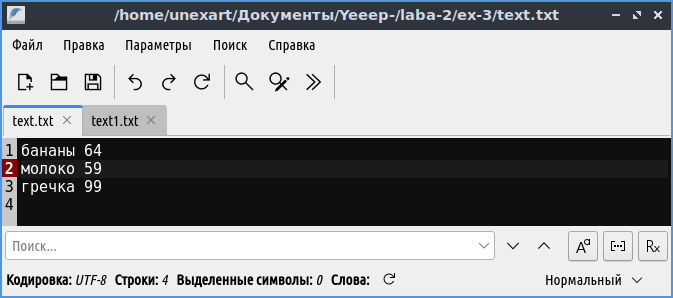
}

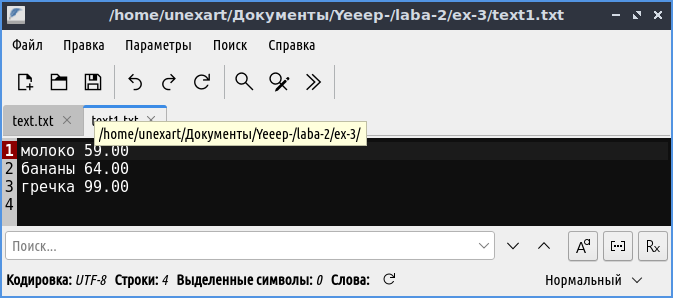
fclose(file);

}

## 3.3. Тестирование работы программы

Рис.3.1 Тестирование программы ex3.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 <stdlib.h>

#include <stdio.h>

//Структура содержащая значение узла стека и ссылку

struct Node\_tag{

int value;

struct Node\_tag \*next;

};

struct Stack{

struct Node\_tag \*head;

int size;

};

//Метод добавления нового узла с перезаписью ссылки вершины

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

if (stack->size >= maxsize) {

perror("Невозможно добавить узел(память переполнена)");//Проверка переполнения памяти

} else {

struct Node\_tag \*in = malloc(sizeof(struct Node\_tag));

in->next = stack->head; //Присвоение ссылки старой вершины к новой

in->value = value;//Значение новой вершины

stack->head = in;

stack->size++;

}

}

//Метод создания стека

struct Stack\* createStack() {

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

if (newStack == NULL) {

perror("Недостаточно оперативной памяти для создания нового стека");

} else {

newStack->head = NULL;

newStack->size = 0;

return newStack;

}

}

//Метод извлечения узла из стека с перезаписью ссылки вершины

int pop(struct Stack \*stack){

if (stack->head == NULL) {

perror("Невозможно извлеч узел(стек пуст)");//Проверка пустого стека

} else {

struct Node\_tag \*out = stack->head;

stack->head = stack->head->next; //Присвоение вершины следующему узлу

int value = out->value;//Значение выходящего узла

free(out);

return value;

}

}

//Метод просмотра верхнего значения

int peek(const struct Stack \*stack){

if (stack->head == NULL) {

perror("Невозможно вывести узел(стек пуст)");//Проверка пустого стека

}else{

return stack->head->value;

}

}

void main() {

int maxsize;

printf("Введите максимальный размер стека:");

scanf("%d", &maxsize);

struct Stack \*stack = createStack();//Задаем пустой стек

for (int i = 0; i < 10; i++){//Генератор значений стека

push(stack, i, maxsize);

}

while (stack->head) {

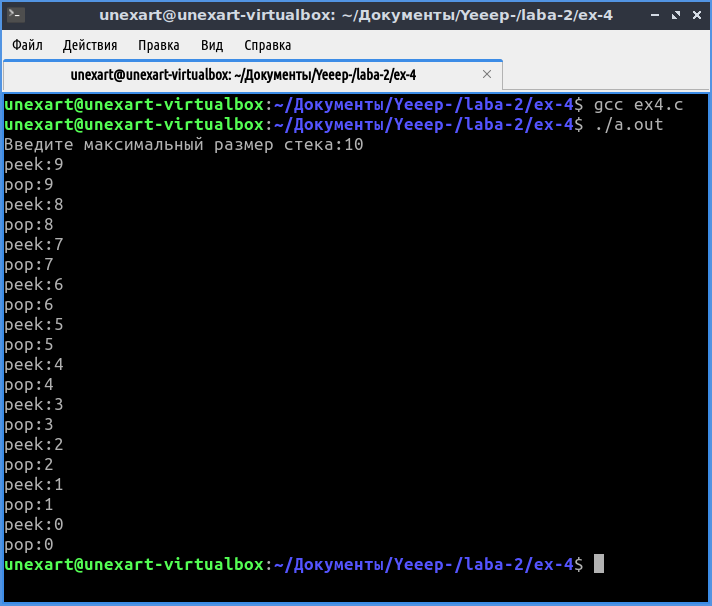
printf("peek:%d\n", peek(stack));//Проверка вершины

printf("pop:%d\n", pop(stack)); //Извлекание вершины

}

}

## 4.3. Тестирование работы программы

Рис. 4 Тестирование программы ex4.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

//\*\*\*\*\*\*\*\*\*\*\*\* ex4\_copy.c \*\*\*\*\*\*\*\*\*\*\*\*\*

#include <stdlib.h>

#include <stdio.h>

struct Node\_tag{

int value;

struct Node\_tag \*next;

};

struct Stack{

struct Node\_tag \*head;

int size;

};

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

if (stack->size >= maxsize) {

perror("Невозможно добавить узел(память переполнена)");

} else {

struct Node\_tag \*in = malloc(sizeof(struct Node\_tag));

in->next = stack->head;

in->value = value;

stack->head = in;

stack->size++;

}

}

struct Stack\* createStack() {

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

if (newStack == NULL) {

perror("Недостаточно оперативной памяти для создания нового стека");

} else {

newStack->head = NULL;

newStack->size = 0;

return newStack;

}

}

int pop(struct Stack \*stack){

if (stack->head == NULL) {

perror("Невозможно извлеч узел(стек пуст)");

} else {

struct Node\_tag \*out = stack->head;

stack->head = stack->head->next;

int value = out->value;

free(out);

return value;

}

}

int peek(const struct Stack \*stack){

if (stack->head == NULL) {

perror("Невозможно вывести узел(стек пуст)");

}else{

return stack->head->value;

}

}

//\*\*\*\*\*\*\*\*\*\*\* ex5.c \*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <stdio.h>

#include <ctype.h>

#include "ex4\_copy.c"

int calc(int var1, int var2, char operation){ //Калькулятор

switch (operation) {

case '+':

return var1 + var2;

break;

case '-':

return var1 - var2;

break;

case '\*':

return var1 \* var2;

break;

case '/':

return var1 / var2;

break;

}

}

int fileSize(char \*filename){

FILE \*file = fopen(filename, "r");

int count = 0;

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

char ch;

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

count++;

}

}

fclose(file);

return count;

}

int main(){

struct Stack \*stack = createStack();

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

if (file == NULL) {

perror("Ошибка открытия файла!");

} else {

int maxsize = fileSize("text.txt");

char ch;

//Считывание постфиксной записи

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

if (isdigit(ch)) {

push(stack, ch - '0', maxsize);

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

//Запись в стек результата операции

push(stack, calc(pop(stack), pop(stack), ch), maxsize);

}

}

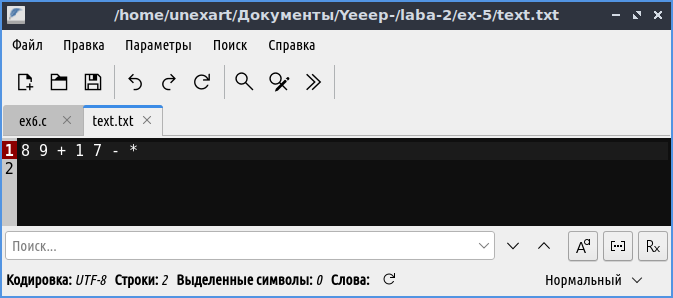
}

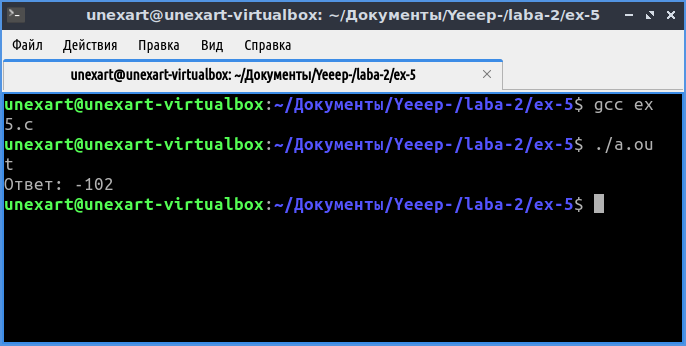
fclose(file);

printf("Ответ: %d\n", peek(stack));

}

## 5.3. Тестирование работы программы

Рис. 5.1 Содержимое тестового файла text.txt

Рис. 5.2 Тестирование работы программы ex5.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, схема трансляции

*factor → (expr)*

*|* *digit* {*post[index] := digit || index++* }

*term →* *factor more**Term*

*moreTerm→ \* factor* { *post[index] := ‘\*’ || index++* } *moreTerm*

*| / factor* { *post[index] := ‘/’* *|| index++* } *moreTerm*

*| ε*

*expr → term more**Expr*

*moreExpr→ + term* { *post[index] := ‘+’ || index++*} *moreExpr*

*| - term* { *post[index] := ‘-’ || index++* } *moreExpr*

*| ε*

*parse → expr* { *print(expr)* }

## 6.3. Решение задачи, код программы

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define n 100

char infExpr[n]; //инфиксное выражение

int infPos = 0; //номер символа инфиксного выражения

char postExpr[n];//постфиксное выражение

int postPos = 0; //номер символа постфиксного выражения

void factor() { //нетерминал продукции скобок и цифр

if (isdigit(infExpr[infPos])) {

postExpr[postPos++] = infExpr[infPos++];

} else if (infExpr[infPos] == '(') {

infPos++;

expr();

if (infExpr[infPos] == ')') {

infPos++;

} else {

perror("Ошибка расположения скобок");

exit(0);

}

} else {

perror("Встреча неожиданного символа");

exit(0);

}

}

void moreTerm() { //нетерминал продукций с операциями '\*' и '/'

if (infExpr[infPos] == '\*' || infExpr[infPos] == '/') {

char op = infExpr[infPos++];

factor();

moreTerm();

postExpr[postPos++] = op;

}

}

void term() { //нетерминал наивысшего приоритета продукций

factor(); // скобки - наивысший приоритет

moreTerm();// '\*' и '/' - повышенный приоритет

}

void moreExpr() { //нетерминал продукций с операциями '+' и '-'

if (infExpr[infPos] == '+' || infExpr[infPos] == '-') {

char op = infExpr[infPos++];

term();

moreExpr();

postExpr[postPos++] = op;

}

}

void expr() { //нетерминал формирования выражаения

term(); // наивысший приоритет

moreExpr();// нижайший приоритет

}

void parse() { //нетерминал перевода в постфиксную запись и вывода

expr();

printf("Постфиксное выражение: %s\n", postExpr);

}

int main() {

printf("Введите инфиксное выражение: ");

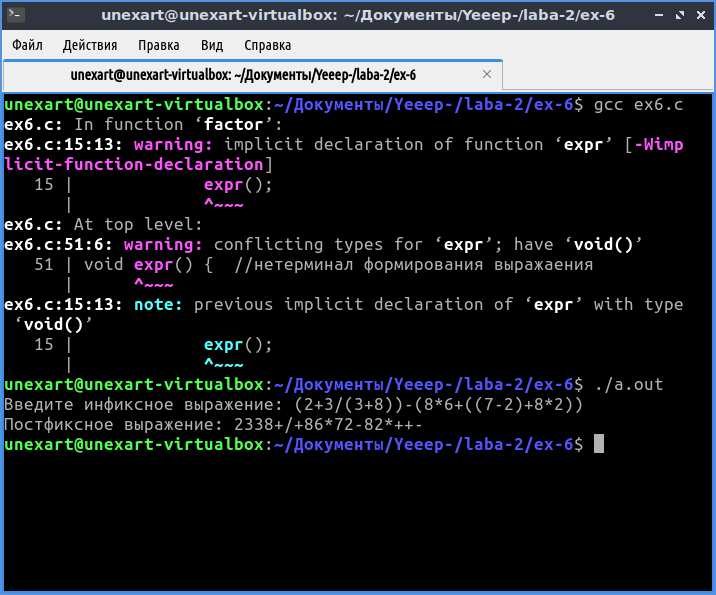
scanf("%s", infExpr);

parse();

return 0;

}

## 6.4. Тестирование работы программы

Рис. 6.1 Тестирование работы программы ex6.c