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 Петраков М.В.

Проверил: доцент каф. ВММБ Батин С.Е.

Пермь, 2023

[Задание 1 3](#_Toc151294367)

[1.1. Постановка задания 3](#_Toc151294368)

[1.2. Решение задачи, код программы 3](#_Toc151294369)

[1.3. Тестирование работы программы 3](#_Toc151294370)

[Задание 2 5](#_Toc151294371)

[2.1. Постановка задания 5](#_Toc151294372)

[2.2. Решение задачи, код программы 5](#_Toc151294373)

[2.3. Тестирование работы программы 6](#_Toc151294374)

[Задание 3 8](#_Toc151294375)

[3.1. Постановка задания 8](#_Toc151294376)

[3.2. Решение задачи, код программы 8](#_Toc151294377)

[3.3. Тестирование работы программы 9](#_Toc151294378)

[Задание 4 11](#_Toc151294379)

[4.1. Постановка задания 11](#_Toc151294380)

[4.2. Решение задачи, код программы 11](#_Toc151294381)

[4.3. Тестирование работы программы 13](#_Toc151294382)

[Задание 5 14](#_Toc151294383)

[5.1. Постановка задания 14](#_Toc151294384)

[5.2. Решение задачи, код программы 14](#_Toc151294385)

[5.3. Тестирование работы программы 16](#_Toc151294386)

Задание 1

1.1. Постановка задания

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

Необходимо ввести в командную строку аргумент, равный названию файла. Затем ввести целое число, которое определит размер массива. После этого программа успешно проработает и выдаст ответ (в консоли и файле).

#include <stdio.h>

#include <stdlib.h>

int quantity(int num){//Функция подсчёта количества цифр

    int x = num;

    int k = 0;

    while(x>0){

        x=x/10;

        k++;

    }

    return k;

}

int main(int args, char\* fname[]){

    FILE \* InvNums=fopen(fname[1],"w");//открыть файл, куда будут записываться числа

    printf("Введите размер массива целых чисел\n");//далее в 3 строках происходит создание размера массива

    int r;

    scanf("%d",&r);

    int intmass[r];//создаётся массив размера, заданного ранее

    srand(time(NULL));

    for(int i=0;i<r;i++){

        intmass[i]=rand();//происходит заполнение псевдослучайными числами

        printf("%d\n",intmass[i]);

    }

    for(int i=r-1;i>=0;i--){

        char charmass[quantity(intmass[i])];//создаётся массив символов, записывается число, и вносится в файл

        sprintf(charmass,"%i",intmass[i]);

        fprintf(InvNums,"%s\n",charmass);

    }

    fclose(InvNums);

}

1.3. Тестирование работы программы

Далее представлено использование программы. Алгоритм линейный, состоящий из одного действия.

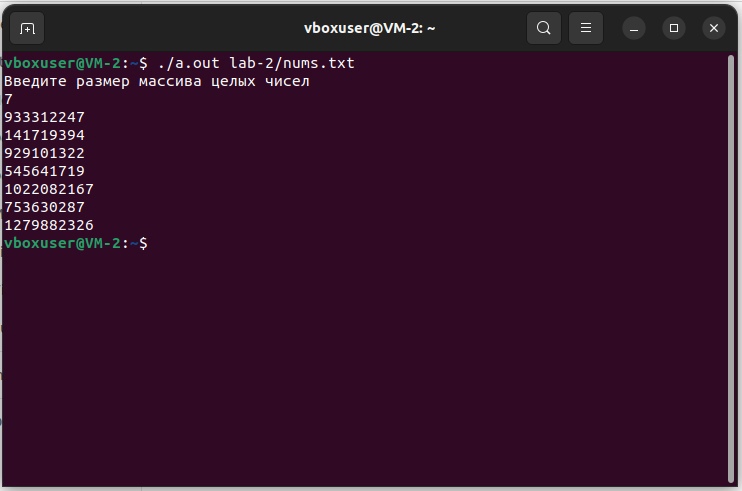


Рис.1 Успешная работа программы ex\_2-1.c (терминал)

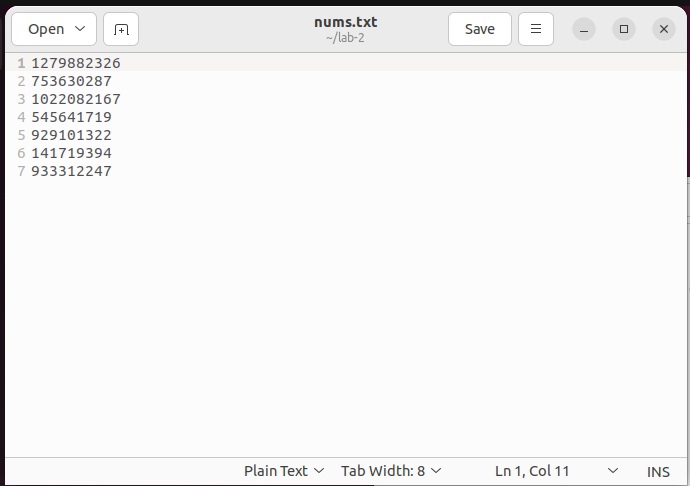


Рис.2 Успешная работа программы ex\_2-1.c (файл)

Задание 2

2.1. Постановка задания

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

Как и в прошлой программе, необходимо ввести в консоль название файла, а также размер массива. Здесь будет создано 2 массива, один будет содержать целые числа, другой – с плавающей точкой.

#include <stdio.h>

#include <stdlib.h>

int quantity(int num){

    int x = num;

    int k = 0;

    while(x>0){

        x=x/10;

        k++;

    }

    return k;

}

void main(int args, char\* fname[]){

    FILE \* InvNums=fopen(fname[1],"w");

    printf("Введите размер массива чисел c плавающей точкой\n");

    int r;

    double fl;

    scanf("%d",&r);//вводится размер двух массивов, один содержит целые числа, другой - с плавающей точкой

    int intmass[r];

    double floatmass[r];//создаётся массив для чисел с плавающей точкой

    srand(time(NULL));

    for(int i=0;i<r;i++){

        intmass[i]=rand();

        floatmass[i]=intmass[i];

        for(int j=0;j<quantity(intmass[i]);j++){//происходит создание числа с плавающей точкой за счёт деления целого числа на 10 в степени количества его цифр

            floatmass[i]=floatmass[i]/10;

        }

    }

    for(int i=0;i<r;i++){//происходит сортировка массива в цикле

        int j=i;

        int k;

        fl=floatmass[i];

        while(j<r){

            if(fl>floatmass[j]){

                fl=floatmass[j];

                k=j;

            }

            j++;

        }

        floatmass[k]=floatmass[i];

        floatmass[i]=fl;

    }

    for(int i=0;i<r;i++){

        printf("%f\n",floatmass[i]);//выводится в консоль массив отсортированных чисел

    }

    for(int i=0;i<r;i++){

        char charmass[quantity(intmass[i])+2];//создаётся массив символов, где элементов на 2 больше,чем в целом числе(точка и ноль)

        sprintf(charmass,"%lf",floatmass[i]);

        fprintf(InvNums,"%s\n",charmass);

    }

    fclose(InvNums);

}

2.3. Тестирование работы программы

Далее представлен пример работы программы при введённом значении 6.

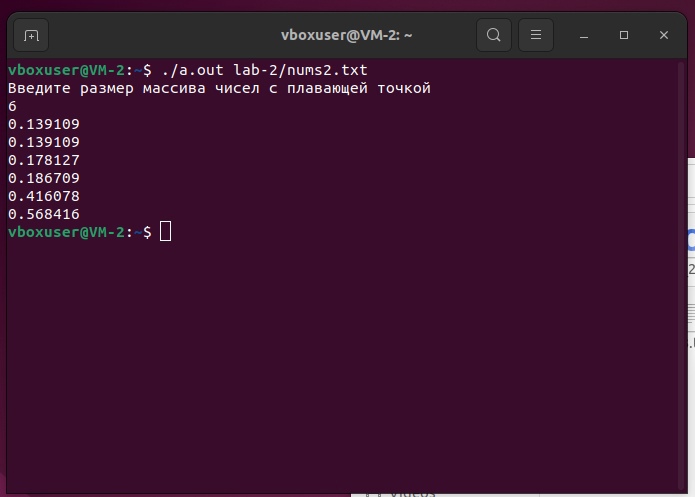


Рис.3 Успешная работа программы ex\_2-2.c (консоль)

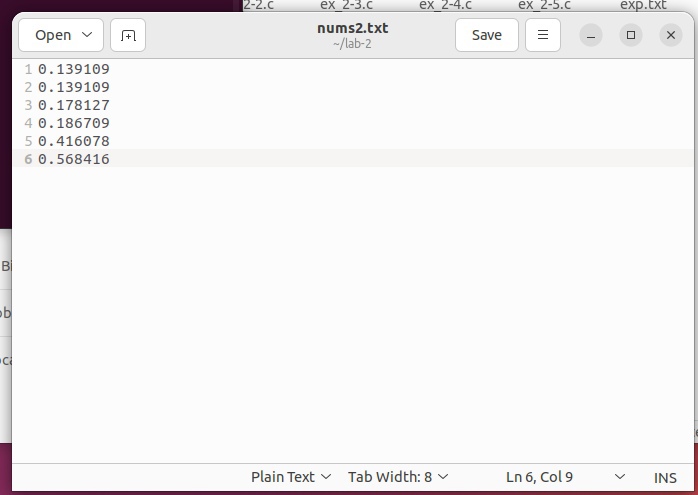


Рис. 4 Успешная работа программы ex\_2-2.c (текстовый файл)

Задание 3

3.1. Постановка задания

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

Здесь используется операция qsort, в которой необходимо указать переменную указатель на начало (в случае массива указателем на начало будет само название массива), размер массива в элементах (в программе он считается ранее), размер элемента в байте (здесь вызывается операция sizeof), и пользовательский метод (в программе он прописан сразу после создания структуры item), а также команда fseek с позицией SEEK\_SET для устранения процедуры повторного открытия файла.

#include <stdio.h>

#include <stdlib.h>

struct item{//создаётся структура, где будут сохранены данные о предмете

    char name[1000];

    double vol;

};

int cmprsn(const void\* a,const void\* b){//процедура сравнения

    double pr\_a, pr\_b;

    pr\_a = ((struct item\*)a)->vol;

    pr\_b = ((struct item\*)b)->vol;

    if(pr\_a==pr\_b){

        return 0;

    }else if(pr\_a>pr\_b){

        return -1;

    }else{

        return 1;

    }

}

int main(int args, char \* fname[]){

    FILE \* IM,\*OM;//создание двух файлов, один для считывания информации, второй для записи

    IM = fopen(fname[1],"r");

    OM = fopen(fname[2],"w");

    char ch;

    if(IM==NULL){

        printf("File not found\n");

        return 0;

    }

    int q=0;

    while(ch!=EOF){

        ch=fgetc(IM);

        if(ch=='\n'){//подсчёт количества товаров

            q++;

        }

    }

    struct item objs[q];//создаётся массив структур(предметов)

    fseek(IM,0,SEEK\_SET);

    int coun=0;

    while(fscanf(IM,"%1000s %lf",objs[coun].name,&objs[coun].vol)==2 && coun<q){

        coun++;

    }

    fclose(IM);

    if(coun>0){//далее происходит сортировк предметов и запись данных о них в файл

        qsort(objs,coun,sizeof(struct item),cmprsn);

        if(OM==NULL){

            printf("ERROR(FILE NOT FOUND)");

            return 0;

        }

        for(int k=0;k<coun;k++){

            fprintf(OM,"%s %.1f\n",objs[k].name,objs[k].vol);

        }

        fclose(OM);

        printf("SUCCESS\n");

    }else{

        printf("ERROR(NON-READABLE STRUCTURES)");

    }

    return 0;

}

3.3. Тестирование работы программы

Далее представлен пример работы программы с файлами T3 и T3\_a.

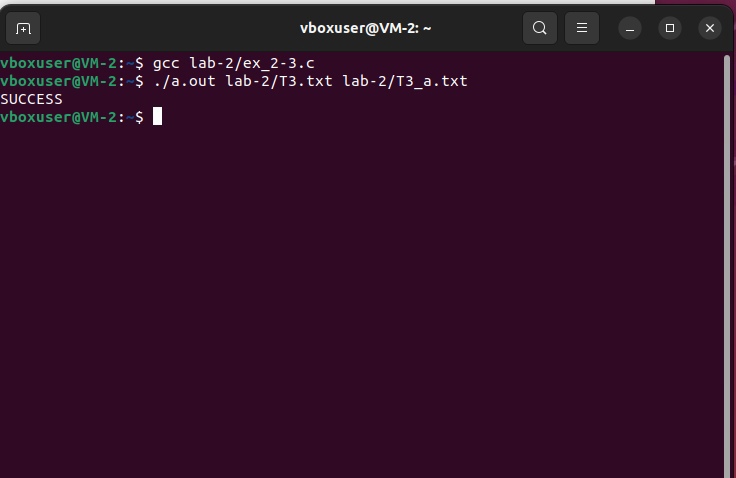


Рис.5 Работа программы ex\_2-3.c (консоль)

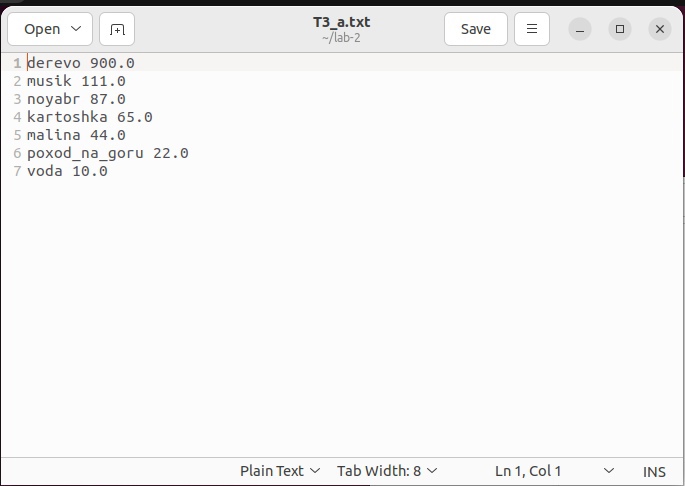
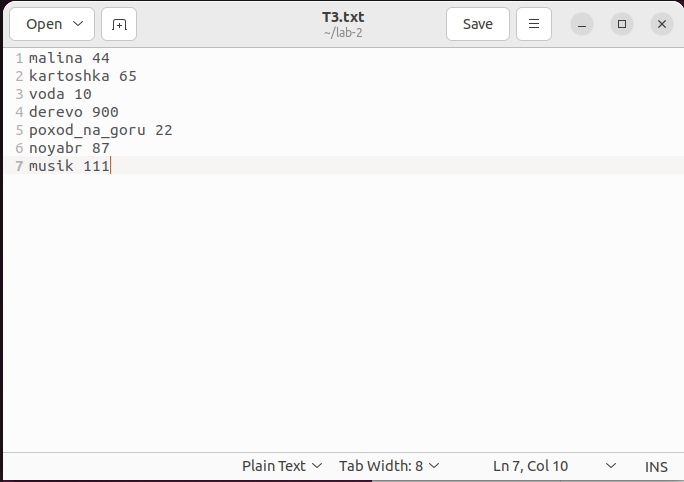


Рис.6-7 Данные в файлах T3.txt и T3\_a.txt (результат работы программы)

Задание 4

4.1. Постановка задания

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

В это программе используется объявление типа структура, содержащая целое число и указатель на структуру, выделение памяти для поданного указателя, освобождение памяти по полученному указателю, ошибка, вызываемая при недостатке памяти или отсутствии элементов, а также заданное число, которое не должен превышать размер стека.

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#define STACK\_OVERFLOW -100

#define STACK\_UNDERFLOW -101

struct st{//объявляется тип Структура, содержащая указатель на следующий элемент(Узел стека)

    double data;

    struct st \*prev;

};

struct Stack{

    double size;

    struct st \* top;

};

void push(struct Stack \* stack, double data, int maxsize){//Добавить элемент

    if(stack->size>=maxsize){

        printf("STACK IS OVERFLOW\n");

        exit(STACK\_OVERFLOW);

    }

    struct st \* adr = malloc(sizeof(struct st));

    adr->data=data;

    adr->prev=stack->top;

    stack->top=adr;

    stack->size++;

}

double pop(struct Stack \* stack){

    if(stack->top==NULL){

        printf("STACK IS UNDERFLOW\n");

        exit(STACK\_UNDERFLOW);

    }

    struct st \* out = stack->top;

    stack->top = stack->top->prev;

    double data = out->data;

    free(out);

    stack->size--;

    return data;

}

double peek(struct Stack \* stack){

    if(stack->top==NULL){

        printf("STACK IS UNDERFLOW\n");

        exit(STACK\_UNDERFLOW);

    }

    struct st \* out = stack->top;

    double data = out->data;

    return data;

}

void main(){

    int maxsize=55;

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

    stack->top=NULL;

    stack->size=0;

    printf("Введите 3 для прекращения работы, 2 для просмотра вершины стека, 1 для добавления элемента в стек и 0 для удаления\n");

    int choice;

    while(choice!=3){

        scanf("%d",&choice);

        switch(choice){

            case 0:

                printf("Удален %lf\n",pop(stack));

                break;

            case 1:

                printf("Введите число для добавления в стек\n");

                double num;

                scanf("%lf\n",&num);

                push(stack,num,maxsize);

                break;

            case 2:

                printf("Вершина равна %lf\n",peek(stack));

                break;

            default:

                printf("Работа прекращена\n");

                break;

        }

    }

    printf("Спасибо за работу\n");

}

4.3. Тестирование работы программы

Далее приводятся примеры работы программы со стеком: переполнение стека и успешное выполнение операций по команде вместе с выходом из действия.

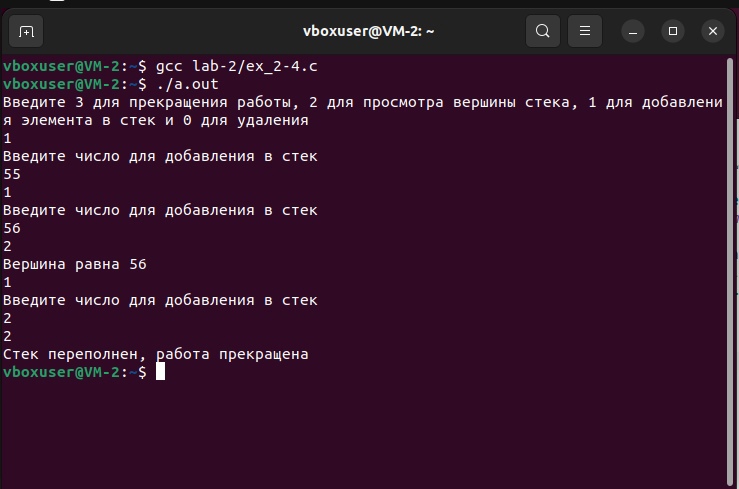


Рис.8 Пример работы программы ex\_2-4.c (переполнение)

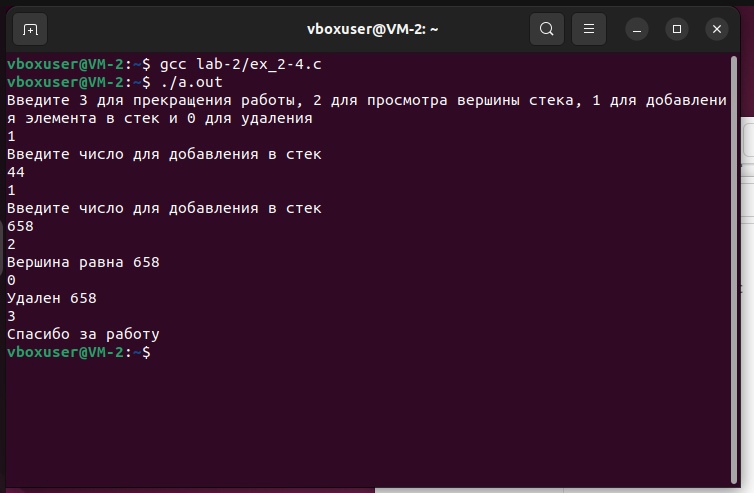


Рис.9 Пример работы программы ex\_2-4.c (успешное выполнение команд)

Задание 5

5.1. Постановка задания

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

В этой программе все арифметические действия выполняются с помощью извлечения ранее положенных элементов стека. Условие существования минимум 2 элементов не проверяется, поэтому ошибка пустоты выводится.

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <ctype.h>

#define STACK\_OVERFLOW -100

#define STACK\_UNDERFLOW -101

#define FILE\_NOT\_FOUND -120

struct st{//объявляется тип Структура, содержащая указатель на следующий элемент(Узел стека)

    double data;

    struct st \*prev;

};

struct Stack{

    double size;

    struct st \* top;

};

void push(struct Stack \* stack, double data, int maxsize){//Добавить элемент

    if(stack->size>=maxsize){

        printf("STACK IS OVERFLOW\n");

        exit(STACK\_OVERFLOW);

    }

    struct st \* adr = malloc(sizeof(struct st));

    adr->data=data;

    adr->prev=stack->top;

    stack->top=adr;

    stack->size++;

}

double pop(struct Stack \* stack){

    if(stack->top==NULL){

        printf("STACK IS UNDERFLOW\n");

        exit(STACK\_UNDERFLOW);

    }

    struct st \* out = stack->top;

    stack->top = stack->top->prev;

    double data = out->data;

    free(out);

    stack->size--;

    return data;

}

double peek(struct Stack \* stack){

    if(stack->top==NULL){

        printf("STACK IS UNDERFLOW\n");

        exit(STACK\_UNDERFLOW);

    }

    struct st \* out = stack->top;

    double data = out->data;

    return data;

}

void main(int cnames, char \* fnames[]){

    int maxsize=55;

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

    stack->top=NULL;

    stack->size=0;

    FILE \*ff = fopen(fnames[1], "r");

    if (ff == NULL){

        printf("FilE NOT FOUND");

        exit(FILE\_NOT\_FOUND);

    }

    char ch;

    double a, b;

    while ((ch = fgetc(ff)) != EOF){

        if (isdigit(ch))

        {

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

        }

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

        {

            b = pop(stack);

            a = pop(stack);

            switch (ch)

            {

            case '+':

                push(stack, a + b,maxsize);

                break;

            case '-':

                push(stack, a - b,maxsize);

                break;

            case '\*':

                push(stack, a \* b,maxsize);

                break;

            case '/':

                push(stack, a / b,maxsize);

                break;

            default:

                break;

            }

        }

    }

    fclose(ff);

    if(stack!=NULL){

        printf("результат равен %lf\n", pop(stack));

    }else{

        printf("стек пуст\n");

    }

}

}

5.3. Тестирование работы программы

Далее представлен пример работы программы. Пример содержится в файле exp.txt

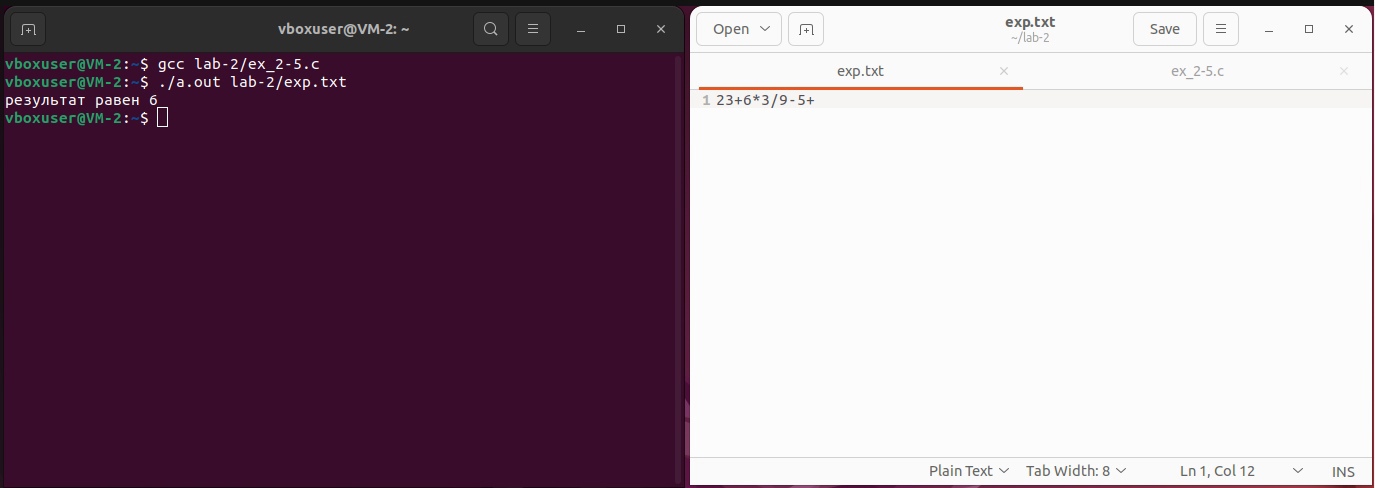


Рис.10 Пример работы программы ex\_2-5.c с текстовым файлом exp.txt