Филиал “Котельники” государственного бюджетного
образовательного учреждения высшего образования
Московской области «Университет «Дубна»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
по курсовой работе по дисциплине “Программирование на языке высокого уровня”

Вариант №19

Выполнил:_____
студент группы ИВТ-11 Третьяков Д.А.
Проверил:_____
доцент, к.т.н. Артамонов Ю.Н.

Котельники – 2019

Оглавление

Введение

Целью курсовой работы является изучение языка высокого уровня Си(C) для решения определенных задач: разработку игровых программ, а также решение численных алгоритмов и приближенных методов нахождения корней уравнений. Охарактеризовать процесс решения задач (разработка численных алгоритмов, приближенные методы нахождения корней уравнений, разработка игровых программ), предоставив методы решения и пояснения к каждой задаче (1. Дано; 2.Найти; Решение), описание входных данных(тип входных данных, ограничения, обработка ошибочного ввода, тестовые наборы входных данных), описание выходных данных(тип выходных данных, верификация выходных данных с использованием Wolfram (http://www.wolframalpha.com/), блок-схема реализуемого алгоритма, листинг программы на языке С, расчетные таблицы соответствия входных и выходных данных, выводы по результатам тестирования программного приложения на расчетных примерах.

Глава I . РАЗРАБОТКА ЧИСЛЕННЫХ АЛГОРИТМОВ

1.1 Суммирование рядов и вычисление элементарных функций

Целью данного задания является вычисление заданного выражения в варианте №19 и подсчитать, сколько членов ряда и цепной дроби понадобится для нахождения суммы.

Дано: Соотношение С.Рамануджанова, имеющего вид:

$$\sqrt{\frac{e \cdot \pi}{2}}$$

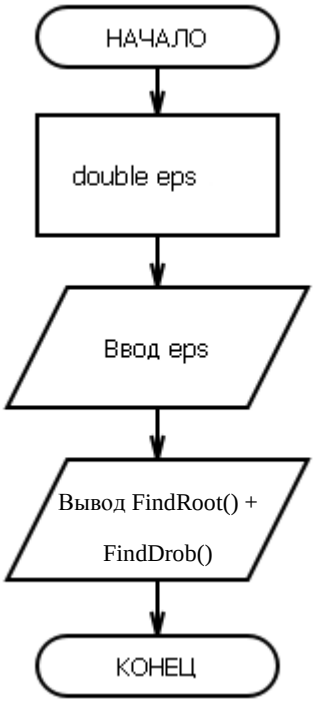
Формула 1.1.1 - Соотношение С.Рамануджанова;

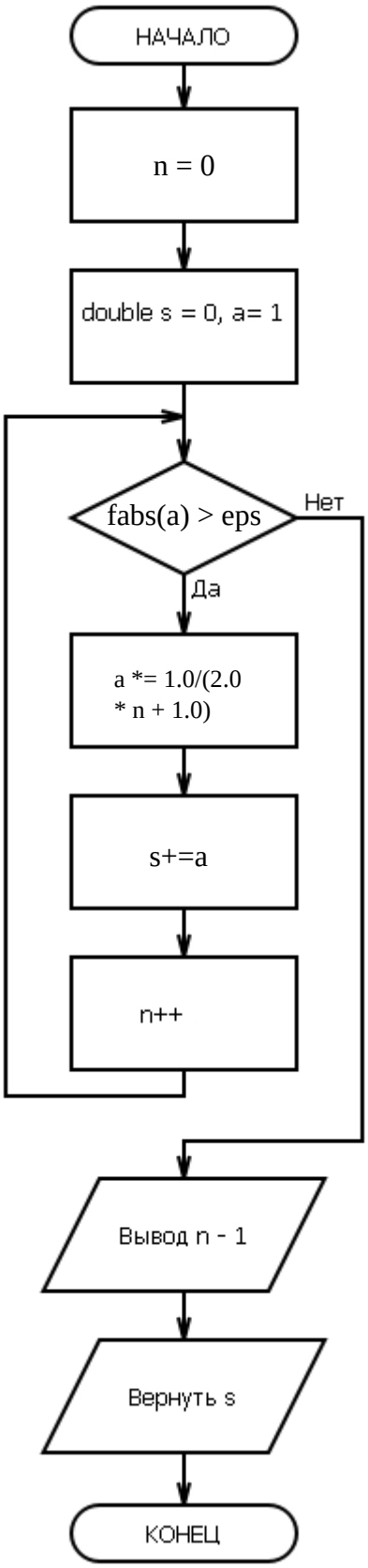
Найти: Вычислить, сколько членов ряда и цепной дроби нужно взять, чтобы достичь заданной точности.

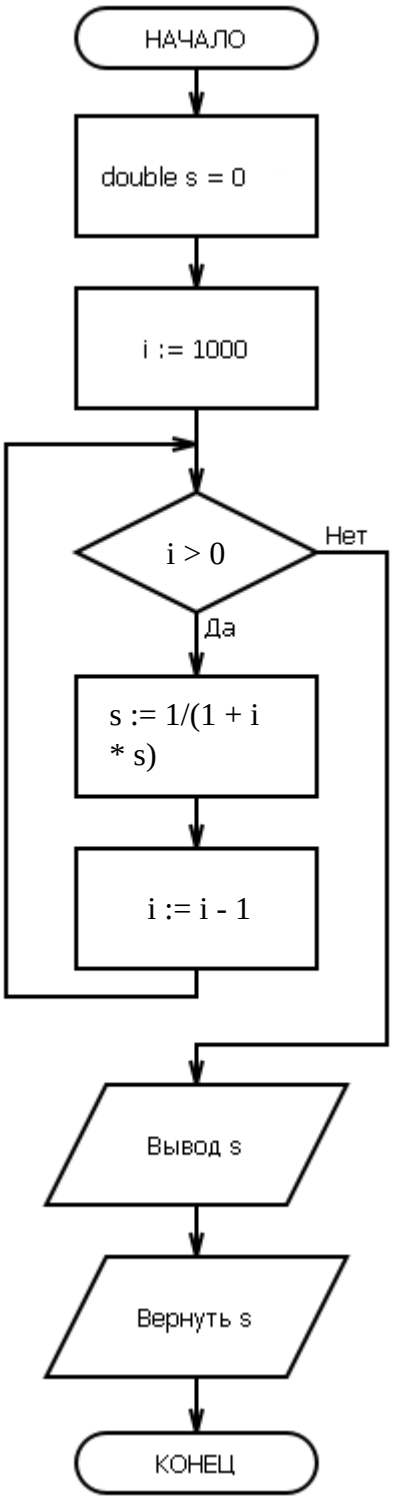
Решение:

Для начала определимся с тем, какие на вход будут подаваться значения. Под «значениями» подразумевается тип данных, с которым будет работать программа. В данном случае, что и на вход, что и на выводе, мы будем получать значения переменных типа double, так как данный тип имеет более высокую точность вычисления, чем тип данных float.

Приведем для начала блок-схему:




НАЧАЛО
n = 0
double s = 0, a= 1
fabs(a) > eps
Нет
Да
a *= 1.0/(2.0 * n + 1.0)
s+=a
n++
Вывод n - 1
Вернуть s
КОНЕЦ



Блок-схема 1.1.8 –Блок-схема программы по нахождению соотношения с помощью числового ряда;

На следующей странице представлен листинг программного кода по нахождению соотношения с помощью числового ряда.

```
#include<stdio.h>
#include<math.h>
double FindRoot(double);
double FindRoot(double eps)
{
	int n = 0; double s = 0, a = 1;
	while (fabs(a) > eps)
	{
		a *= 1.0/(2.0 * n + 1.0); s += a;
		n++;
	}
	printf("%lf\t%d\n", s, n); return s;
}
double FindDrob()
{
	double s = 0;
	for (int i = 1000; i > 0; i--) s = 1/(1 + i * s);
	printf("%lf\n", s);
	return s;
}
int main()
{
	double eps;
	printf("Введите точность вычисления: "); scanf("%lf", &eps);
	printf("%lf\n",FindRoot(eps) + FindDrob());
	return 0;
}
```

Листинг 1.1.9 –Листинг программного кода нахождения соотношения Рамануджанова рядом и цепной дробью

Приведем таблицы входных и выходных данных переменной **e**, а также результат:

Переменная	Значение в программе, заданное пользователем
e	0.00001

Таблица 1.1.10 – Входные данные для программы

Переменная	Вывод суммы числового ряда
S	1.410686

Таблица 1.1.11 – Сумма числового ряда

Переменная	Кол-во членов ряда
n	6

Таблица 1.1.12 – Выходные данные

Переменная	Кол-во членов дроби
i	1000

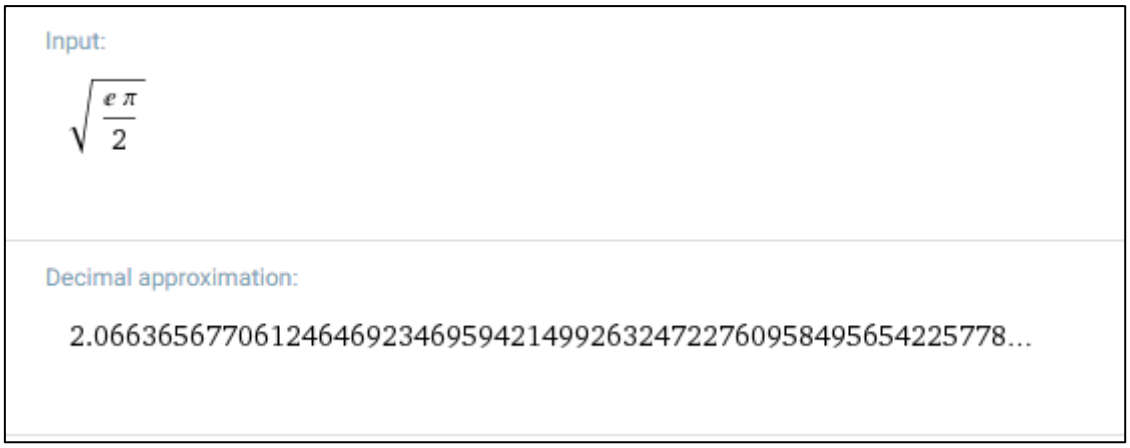
Таблица 1.1.13 – Кол-во членов ряда;

Переменная	Вывод суммы цепной дроби
s	0.655680

Таблица 1.1.14 – Сумма цепной дроби

Переменная	Вывод суммы членов ряда и цепной дроби
S	2.066365

Таблица 1.1.15 – Сумма соотношения



Листинг 1.1.16 – Верификация данных через WolframAlpha

В заключении можно сказать, что на примере С.Рамануджанова мы рассмотрели один из принципов записи чисел в виде числового ряда и цепной дроби.

1.2 Приближённые методы нахождения корней уравнения

В варианте №19 необходимо провести анализ метода деления отрезка пополам и метода секущих и сравнить число итераций при одном и том же значении точности вычисления.

1.2.1 Метод деления отрезка пополам

Предположим, что на отрезке [a,b] в точке X0 график функции f(x) пересекает ось абсцисс. Тогда нужно сдвигать левую и правую границу отрезка [a,b] в точку $C = \frac{a+b}{2}$

Дано:

Уравнение: $\sin(ax) - b = 0$ (уравнение №1).

Уравнение: $x^4 + ax^3 - bx = 0$ (уравнение №5).

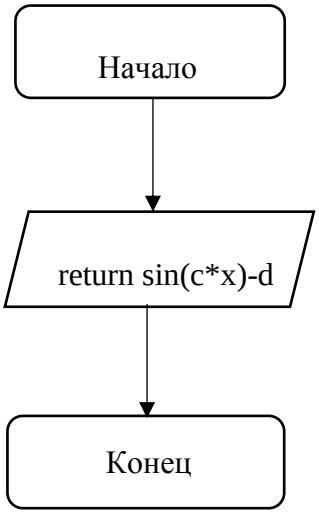
Уравнение: $x^5 + ax^2 - b = 0$ (уравнение №6).

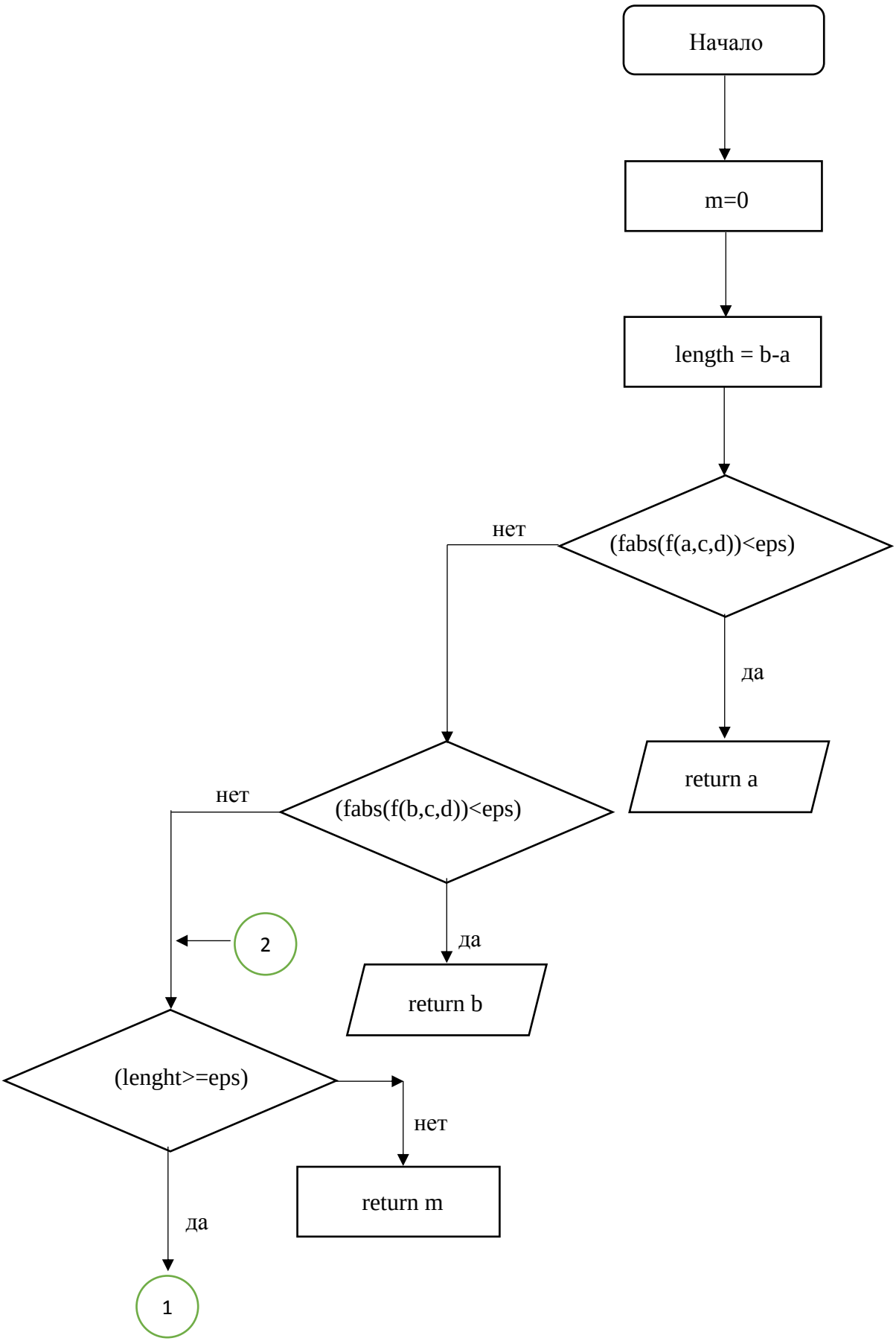
Найти:

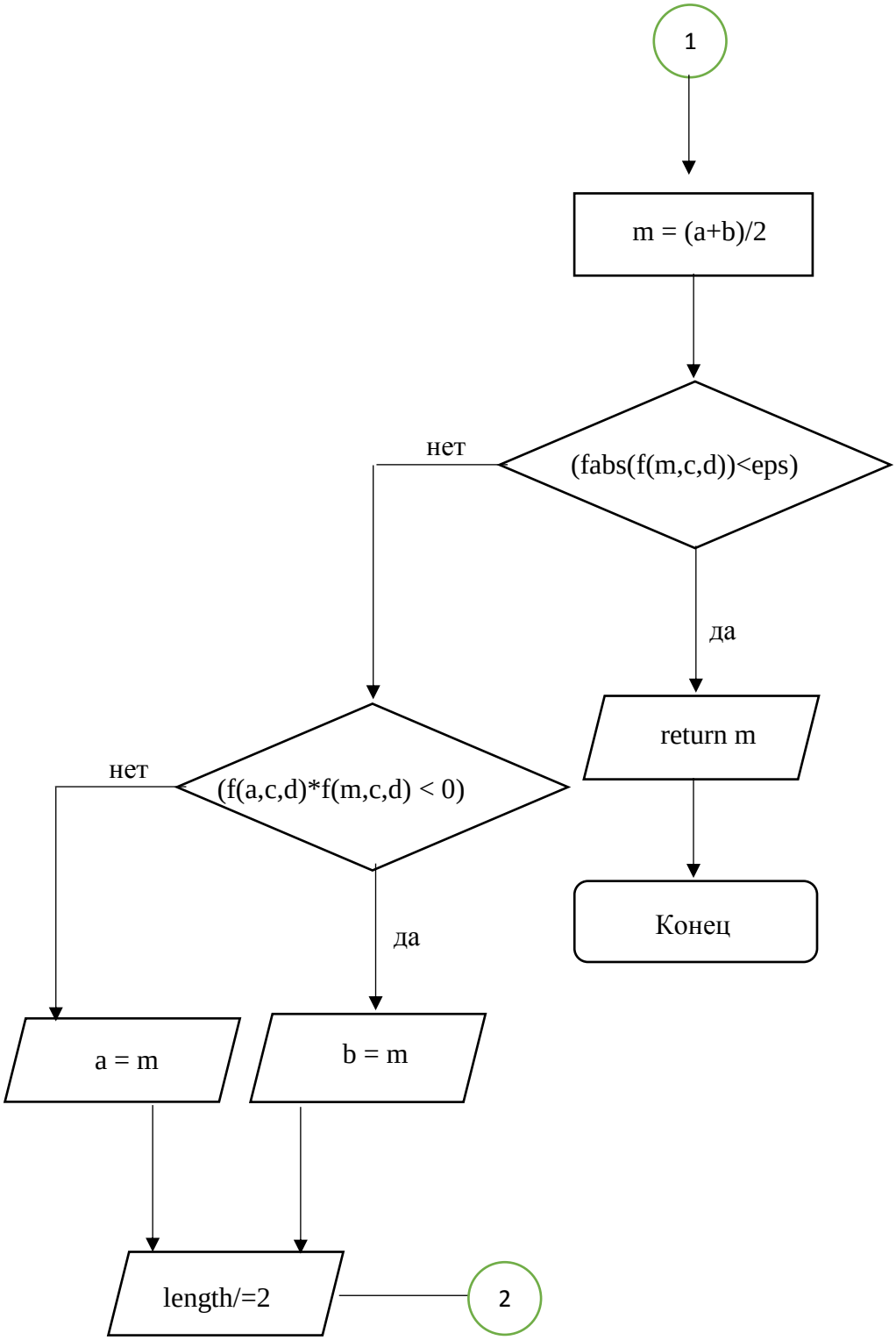
Значение **x** методом деления отрезка пополам.

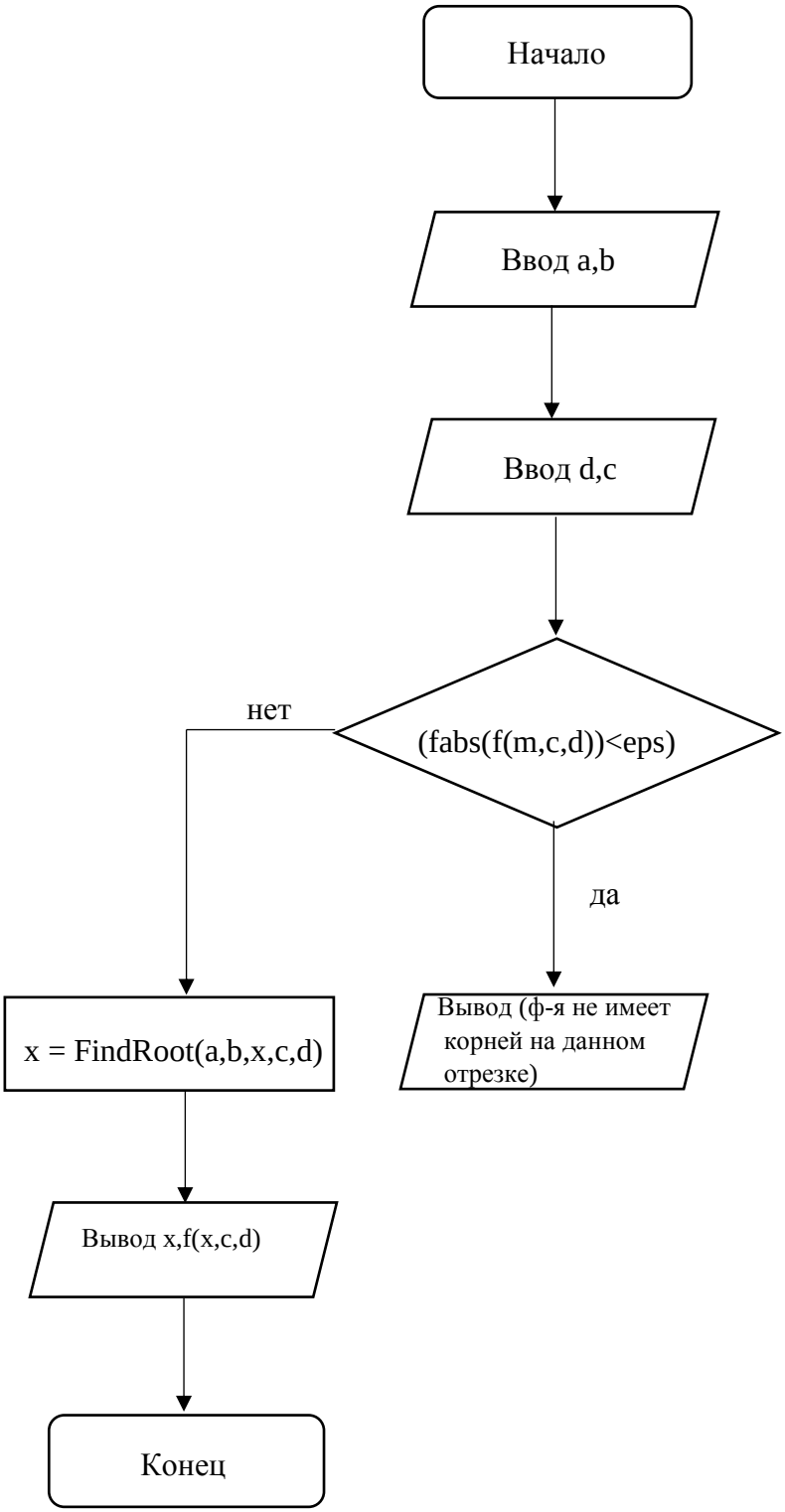
Решение:

Значение переменных a, b и e (точность) вводятся с клавиатуры пользователем. Значение e(точность вычисления)=**1e-6**. Рассмотрим блок схемы реализуемого алгоритма.




Начало
m=0
length = b-a
нет
(fabs(f(a,c,d))<eps)
да
return a
нет
(fabs(f(b,c,d))<eps)
да
return b
2
(lenght>=eps)
нет
return m
да
1


1
m = (a+b)/2
нет
(fabs(f(m,c,d))<eps)
да
return m
Конец
нет
(f(a,c,d)*f(m,c,d) < 0)
да
a = m
b = m
length/=2
2



Блок-схема 1.2.1.1 – Блок-схема метода половинного деления

На следующей странице представлен листинг программного кода, где рассмотрен метод половинного деления на уравнении №6.

```
#include<stdio.h> // подключаем закголовочный файл ввода-вывода
#include<math.h> // подключаем математическую библиотеку
#define eps 0.00001 // точность вычисления
double f(double,double,double); // прототип функции f
double FindRoot(double,double,double,double,double); прототип функции FindRoot
double f(double x,double c,double d) // объявляем функцию f
{
	return sin(c*x) - d; // полином
}
double FindRoot(double a,double b,double x,double c, double d) // объявляем функцию FindRoot
{
	double m; int i = 0; // объявляем переменные m, i
	double lenght = b-a; // объявляем длину
	if(fabs(f(a,c,d))<eps) // если значение функции меньше точности, возвращаем a
	return a; // возвращаем a
	else if(fabs(f(b,c,d))<eps) // если значение функции меньше точности, возвращаем b
	return b; // возвращаем b
	while(lenght>=eps){ // пока длина больше или равна точности, выполняем
	m = (a+b)/2;// находим середину отрезка
	if(fabs(f(m,c,d))<eps) // если значение функции меньше точности, возвращаем m
	  return m; // возвращаем m
	if(f(a,c,d) * f(m,c,d) < 0) // если произведение функций меньше 0, то
		b = m; // присваиваем b значение m
	else // иначе
		a = m; // присваиваем a значение m
	lenght/=2; i++; // уменьшаем длину и увеличиваем число итераций
   }
   printf(“Число итераций: %d\n”, i);
   return m; // возвращаем m
}
int main()
{
	double a,b,c,d,x; // объявляем переменные
	printf("Введите длину a и b: "); scanf("%lf%lf",&a,&b); // вводим интервал
	printf("Введите c и d: "); scanf("%lf%lf",&c,&d); // вводим параметры
	if(f(a,c,d)*f(b,c,d)>0) // если точки нет на интервале, сообщаем ошибку
		printf("Ф-я не имеет корней на данном отрезке\n");
	x = FindRoot(a,b,x,c,d); // находим точку с помощью функции
	printf("x = %lf, f(x) = %lf\n",x,f(x,c,d)); // выводим значение x
	return 0; // завершаем работу
}
```

Листинг 1.2.1.2 –Листинг программного кода метода половинного деления

Рассмотрим уравнение №1:

$$\sin(c * x) - d = 0$$

Формула 1.2.1.3 – Уравнение №1

a	b	c	d
1	2	-1.5	-0.5
1	2	-1.6	-0.4
-2	-1	-1.9	-0.5

Таблица 1.2.1.4 – Входные данные

I	Значение x
14	1.745331
11	1.706299
14	-1.929047
15	-0.902756

Таблица 1.2.1.5 – Выходные данные

Проверка:

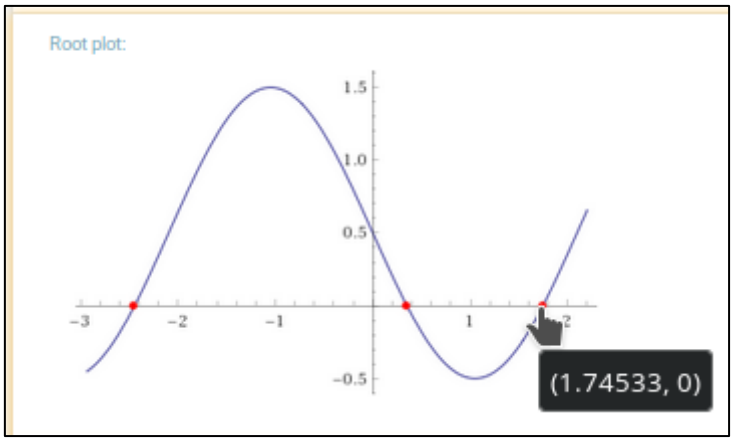


Рисунок 1.2.1.6 – Верификация с помощью WolframAlpha

Рассмотрим уравнение №5:

$$x^4 + cx^3 - dx = 0$$

Формула 1.2.1.7 –Уравнение №5

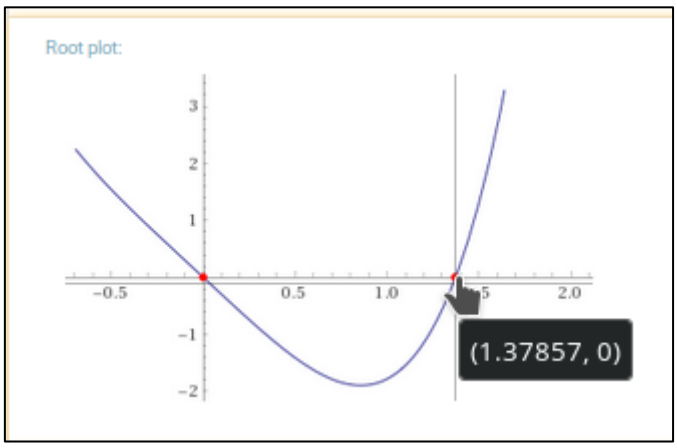
a	b	c	d
1.1	2	0.2	3
1	2.5	0.5	5
1	3	-0.5	1.1
1.5	9	0.2	7

Листинг 1.2.1.8 – Входные данные

I	x
17	1.378565
18	1.558512
18	1.228661
20	1.848537

Листинг 1.2.1.9 – Выходные данные

Проверка:



Листинг 1.2.1.10 – Верификация с помощью WolframAlpha

Уравнение №6:

$$x^5 + cx^2 - d = 0$$

Листинг 1.2.1.11 – Уравнение №6

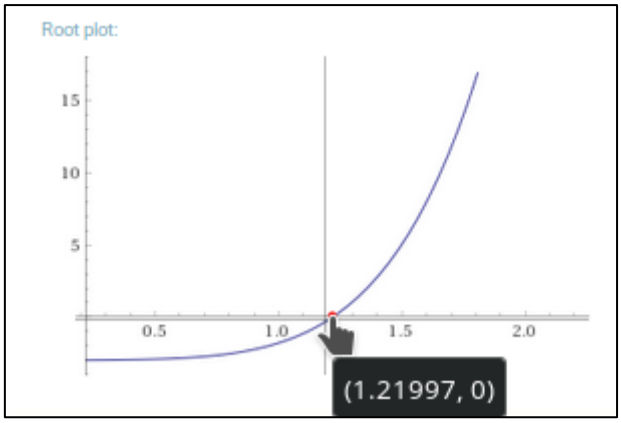
a	b	c	d
1.1	2	0.2	3
1	1.5	0.5	5
1	1.5	-0.5	1.1
1	1.5	0.2	7

Листинг 1.2.1.12 – Входные данные

i	x
17	1.219964
1	1.327278
16	1.114738
16	1.457405

Листинг 1.2.1.13 – Выходные данные

Проверка:



Листинг 1.2.1.14 – Верификация с помощью WolframAlpha

Теперь перейдем к методу секущих.

1.2.2 Метод секущих

От рассмотренного выше случая в методе хорд(секущих) по-иному выбирается точка, в которой проверяется значение функции. В частности, через граничные точки графика функции $f(x) = 0$ проводится хорда. В качестве контрольной точки выбирается точка пересечения этой хорды с координатной осью. Во всем остальном алгоритм такой же, как и в методе половинного деления. При поиске корня на интервале $x \epsilon (a, b)$ контрольная точка внутри этого интервала выбирается как:

$$c = \frac{f(b)a - f(a)b}{f(b) - f(a)}$$

Формула 1.2.2.1 – Нахождение корня методом секущих

Дано:

Уравнение: $\sin(cx) - d = 0$ (уравнение №1).

Уравнение: $x^4 + cx^3 - dx = 0$ (уравнение №5).

Уравнение: $x^5 + cx^2 - d = 0$ (уравнение №6).

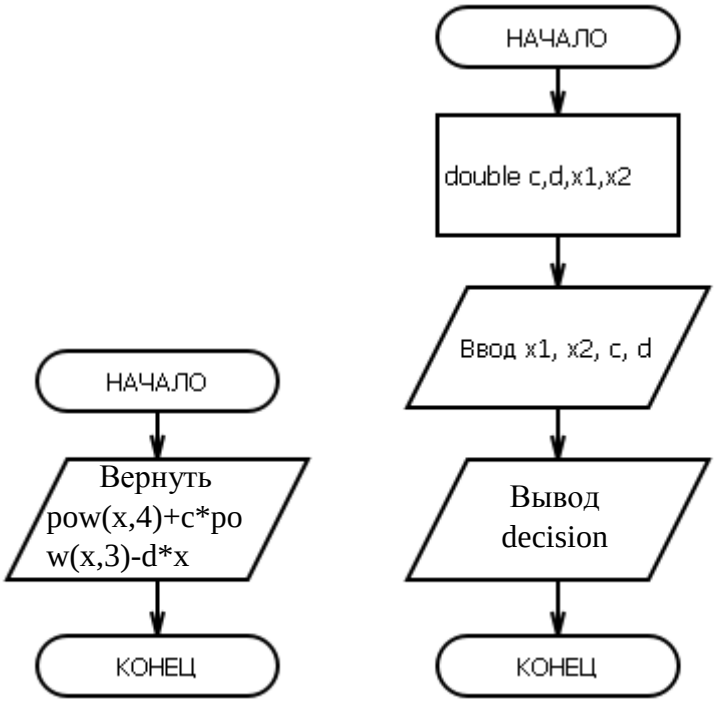
Найти:

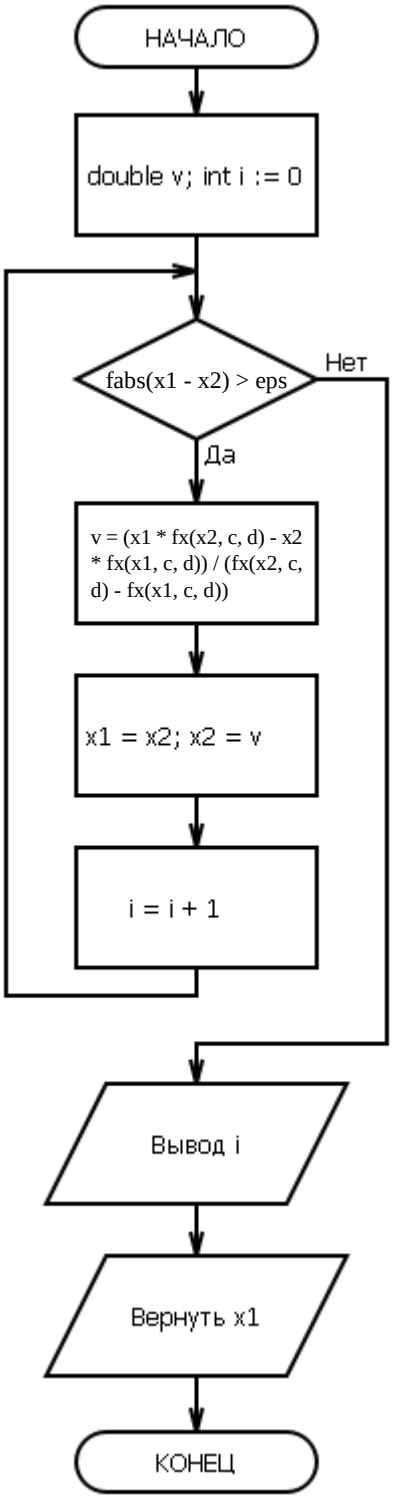
Значение **x** методом секущих.

Решение:

Значение переменных **a**, **b**, **c**, **d** вводятся с клавиатуры пользователем. Значение e(точность вычисления)=**1e-6**

Для начала рассмотрим блок-схему будущей программы:





Блок-схема 1.2.2.2 – Блок-схема метода секущих

На следующей странице представлен листинг программного кода, где рассмотрен метод секущих уравнения №6.

```
#include <stdio.h> // стандартный заголовочный файл
#include <math.h> // математическая библиотека
#define eps 1e-13 // точность вычисления
typedef double (*func)(double x, double c, double d); // задаем тип func
double fx(double, double, double);      // прототип вычисляемой функции
double fx(double x, double d, double c) // вычисляемая функция
 {
  double f = pow(x, 4) + d * pow(x, 3) - c * x; // полином
  return f; // возвращаем полином
 }
double decision(func fx, double x1, double x2, double c, double d) // объявляем функцию
 {
  double v; int i = 0; объявляем переменную v- корень, i – кол-во итераций
  while (fabs(x1 - x2) > eps) // пока не достигнута точность eps(0.0000001)
 {
    v = (x1 * fx(x2, c, d) - x2 * fx(x1, c, d)) / (fx(x2, c, d) - fx(x1, c, d)); // находим корень
    x1 = x2; x2 = v; // уменьшаем интервал
    i++; // увеличиваем число итераций
 }
  printf(“Итераций: %d\n”, i); // выводим кол-во итерацй
  return x1; // возвращаем значение корня
 }
int main()
{
  double c, d; // объявляем переменные c – угол, d – значение по y
  double x1, x2; //x1, x2 - начало и конец отрезка, для которого применяем метод секущих
  printf("Введите интервал(x1 и x2): "); scanf("%lf %lf", &x1, &x2);// Ввод интервала
  printf("Введите значение c и d: "); scanf("%lf %lf", &c, &d); Ввод параметров
  printf("x = %f\n", decision(fx, x1, x2, c, d)); // Вывод находимой точки
  return 0; //
}
```

Листинг 1.2.2.3 – Листинг программного кода метода секущих

Рассмотрим уравнение №5:

$$x^4 + cx^3 - dx = 0$$

Формула 1.2.2.4 – Уравнение №5

a	b	c	d
1.1	2	0.2	3
1	2.5	0.5	5
1	3	-0.5	1.1
1.5	9	0.2	7

Таблица 1.2.2.5 – Входные данные

i	x
8	1.378570
10	1.558507
9	1.228663
9	1.848534

Таблица 1.2.2.6 – Выходные данные

Как видим, данные сходятся.

Рассмотрим уравнение №6:

$$x^5 + cx^2 - d = 0$$

Формула 1.2.2.7 – Уравнение №6

a	b	c	d
1.1	2	0.2	3
1	1.5	0.5	5
1	1.5	-0.5	1.1
1	1.5	0.2	7

Таблица 1.2.2.8 – Входные данные

i	x
7	1.219966
6	1.327278
7	1.114737
5	1.457410

Таблица 1.2.2.9 – Выходные данные

Рассмотрим уравнение №1:

$$sin(cx) - d = 0$$

Формула 1.2.2.10 – Уравнение №1

a	b	c	d
1	2	-1.5	-0.5
1	2	-1.6	-0.4
-2	-1	-1.9	-0.5

Таблица 1.2.2.11 – Входные данные

I	Значение x
6	1.745329
6	1.706297
4	-1.929048

Таблица 1.2.2.12 – Выходные данные

Используя данные программы и сервиса WolframAlpha проверим, насколько верен результат между двумя методами, опираясь на количество итераций хотя бы одного уравнения.

Уравнение №5:

Метод половинного деления:

a	b	c	d	i	F(x)
1.1	2	0.2	3	17	1.378565

Таблица 1.2.2.13 – Входные и выходные данные метода половинного деления

Метод секущих:

x1	x2	c	d	i	F(x)
1.1	2	0.2	3	8	1.378570

Таблица 1.2.2.14 – Входные и выходные данные метода секущих

Уравнение №6:

Метод половинного деления:

a	b	c	d	i	F(x)
1.1	2	0.2	3	17	1.219964

Таблица 1.2.2.15 – Входные и выходные данные метода половинного деления

Метод секущих:

a	b	c	d	i	F(x)
1.1	2	0.2	3	7	1.219966

Таблица 1.2.2.16 – Входные и выходные данные метода секущих

Уравнение №1:

Метод половинного деления:

a	b	c	d	i	F(x)
1	2	-1.5	-0.5	14	1.745331

Таблица 1.2.2.17 – Входные и выходные данные метода половинного деления

Метод секущих:

a	b	c	d	i	F(x)
1	2	-1.5	-0.5	6	1.175329

Таблица 1.2.2.18 – Входные и выходные данные метода секущих

Можно сделать вывод, что при работе с методами нахождения корня уравнения, полученные выходные данные могут стремиться к ответу быстрее или медленнее друг друга в зависимости от начального приближения, который может задать пользователь. Используя онлайн калькулятор Wolfram, который доступен по ссылке www.wolframalpha.com/ , можно наглядно увидит результаты, что были продемонстрированы.

Глава II. РАЗРАБОТКА ИГРОВОЙ ПРОГРАММЫ

В последней части курсовой работы мы разберем написание игровой программы на языке Си.

В варианте №19 предлагается написать программу, моделирующую компьютер. В данное устройство входит специальный регистр - аккумулятор, в который помещается результат арифметических операций, различных операций сравнения. Так же присутствует оперативная память из 100 ячеек, куда можно записать любое целое число. Для работы с устройством присутствуют команды, каждая из которых выполняет то или иное действие. Например, 10 - выводится слово на терминал из указанного адреса памяти, 20 - в аккумулятор помещается слово из указанного адреса памяти.

Программа имеет такие модификации, как: Меню, визуализация(перемещение по меню), Shell(командная строка) и именуемые команды для работы с операциями компьютера.

Для начала рассмотрим иллюстрации работы программы:

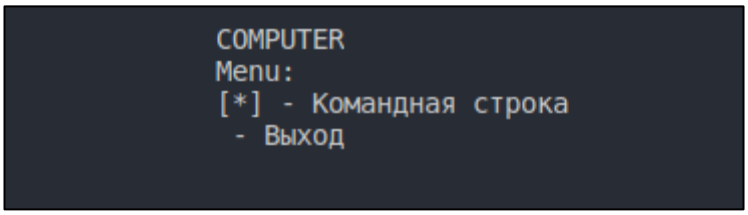


Рисунок 2.1 – Меню программы

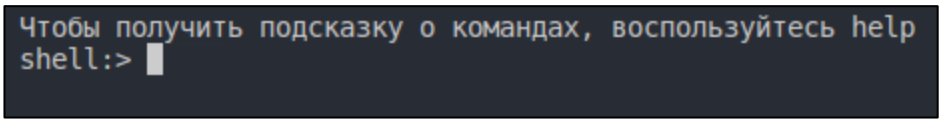


Рисунок 2.2 – Работа shell

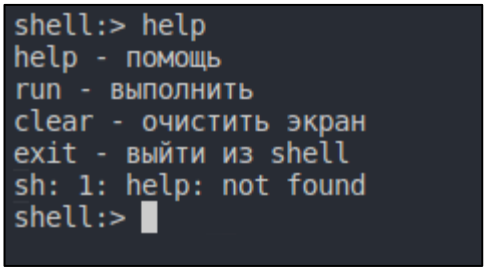


Рисунок 2.3 – Вспомогающие функции

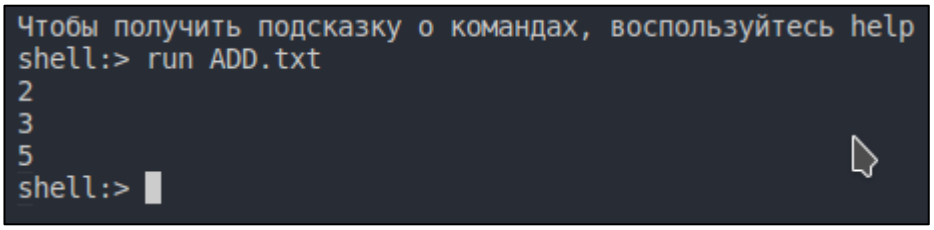
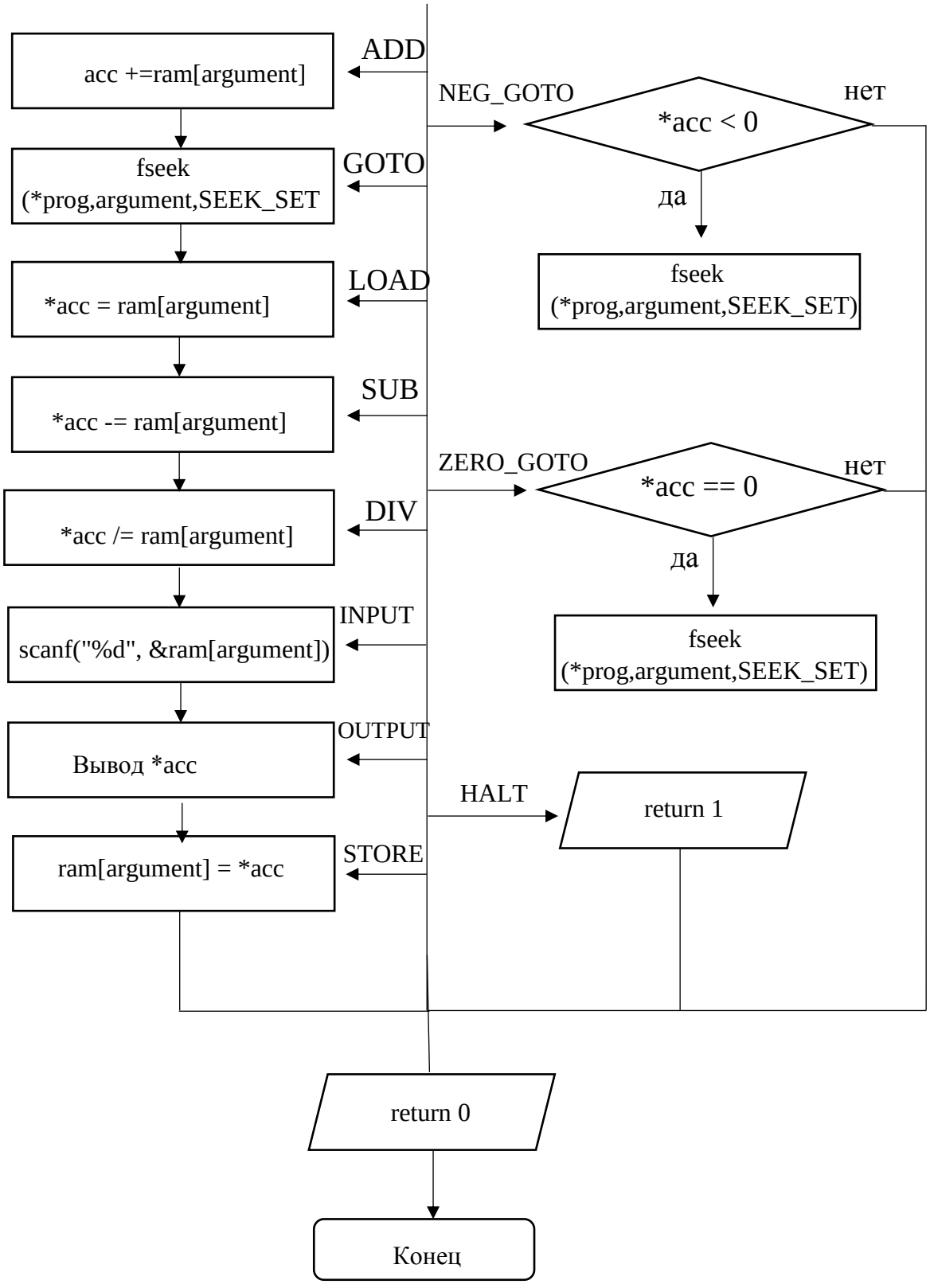


Рисунок 2.4 – Пример работы

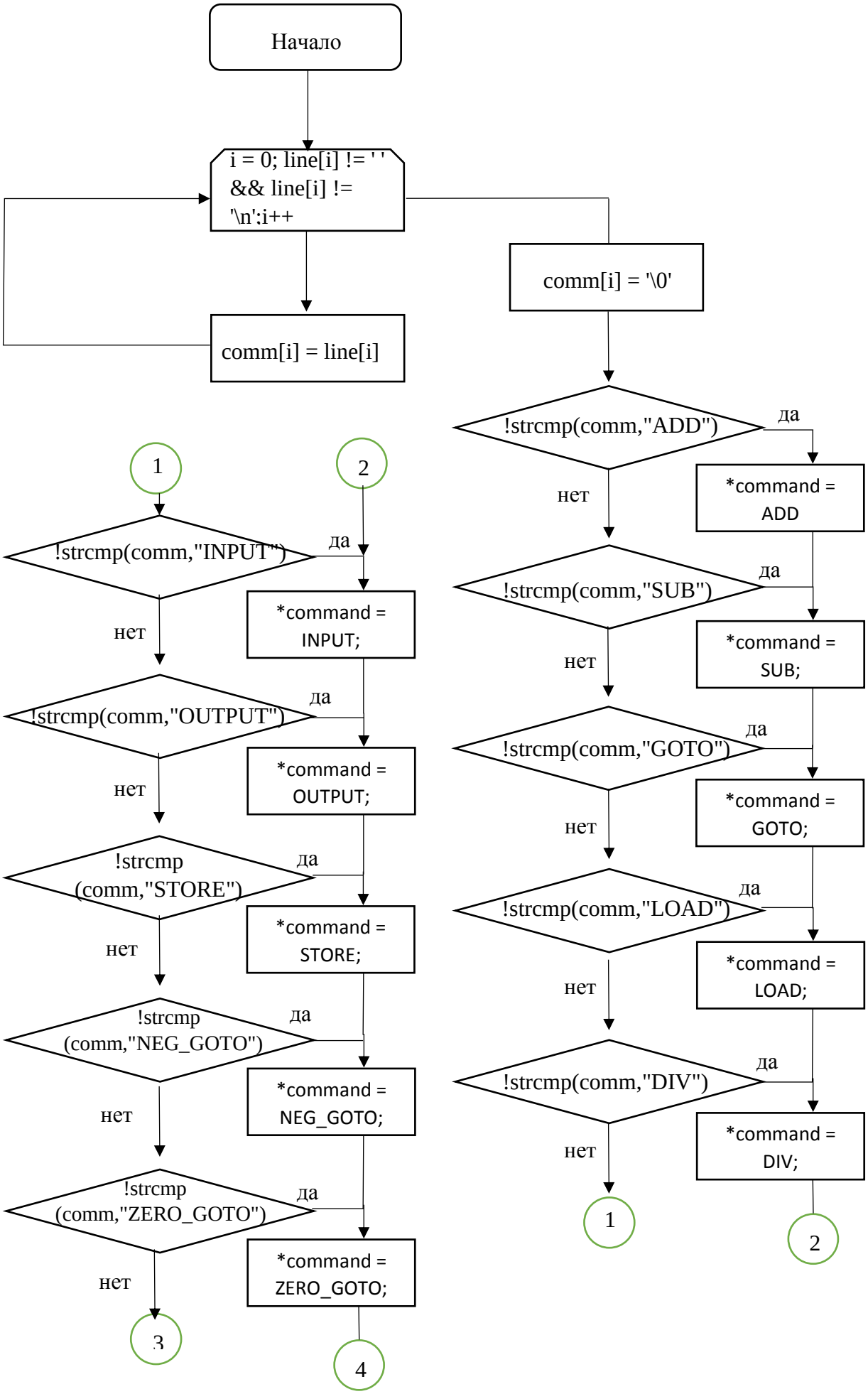
На следующей странице представлены листинги и блок-схемы игровой программы.

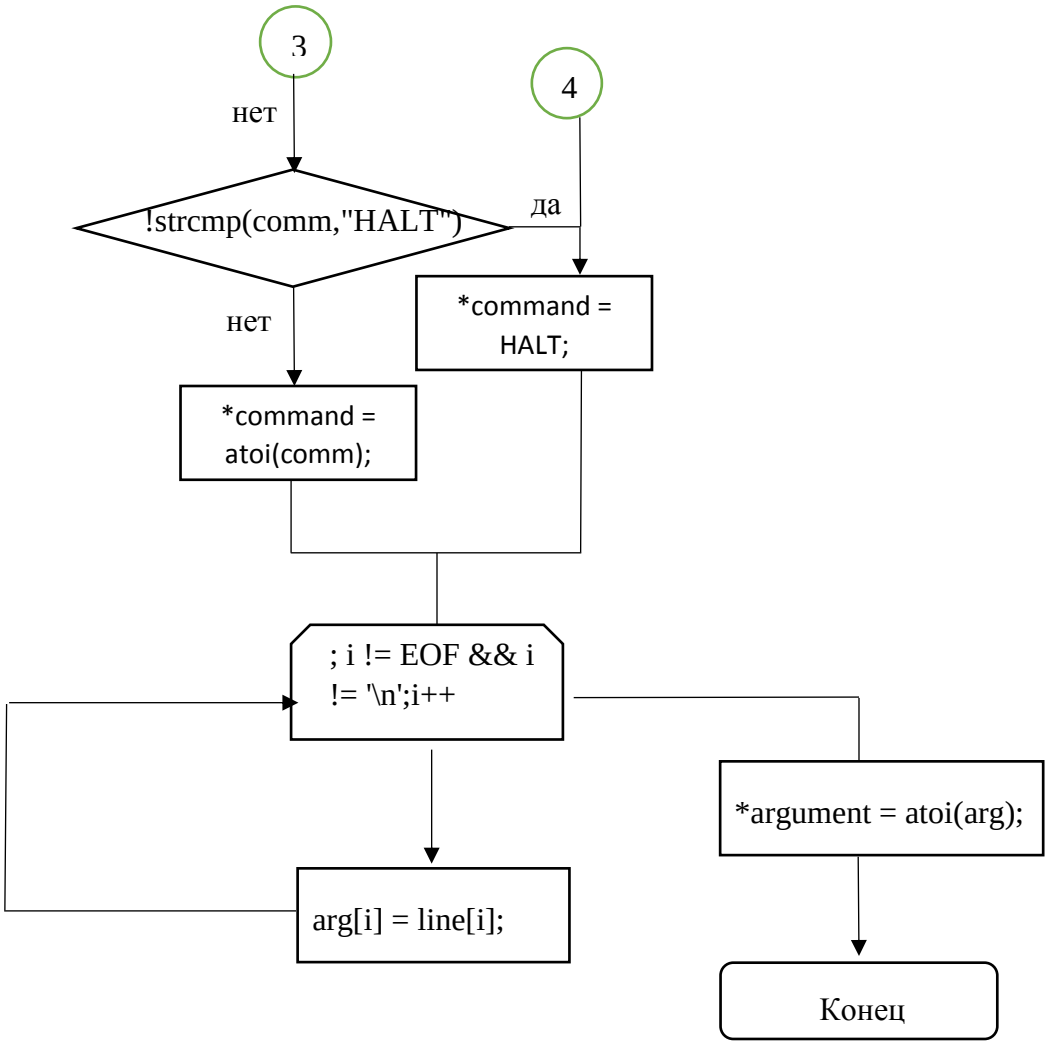
Блок-схема 2.5 – Реализация команд компьютера



```
int alu(int command,int argument,int *acc, int *ram,FILE **prog) // получает команду и
аргумент
{
        switch(command){
          case ADD:
            *acc += ram[argument];
                 break;
                case GOTO:
                  fseek(*prog,argument,SEEK_SET);
                  break;
                case LOAD:
                  *acc = ram[argument];
                  break;
            case SUB:
             *acc -= ram[argument];
                 break;
                case DIV:
             *acc /= ram[argument];
                 break;
                case INPUT:
                  scanf("%d", &ram[argument]);
                  break;
            case OUTPUT:
                  printf("%d\n",*acc);
                  break;
                case STORE:
                  ram[argument] = *acc;
                  break;
                case NEG_GOTO:
                  if(*acc < 0)
                        fseek(*prog,argument,SEEK_SET);
                  break;
                case ZERO_GOTO:
                  if(*acc == 0)
                        fseek(*prog,argument,SEEK_SET);
                  break;
                case HALT:
                  return 1;
                  break;
    }
        return 0;
}
```

Листинг 2.6 – Реализация команд компьютера

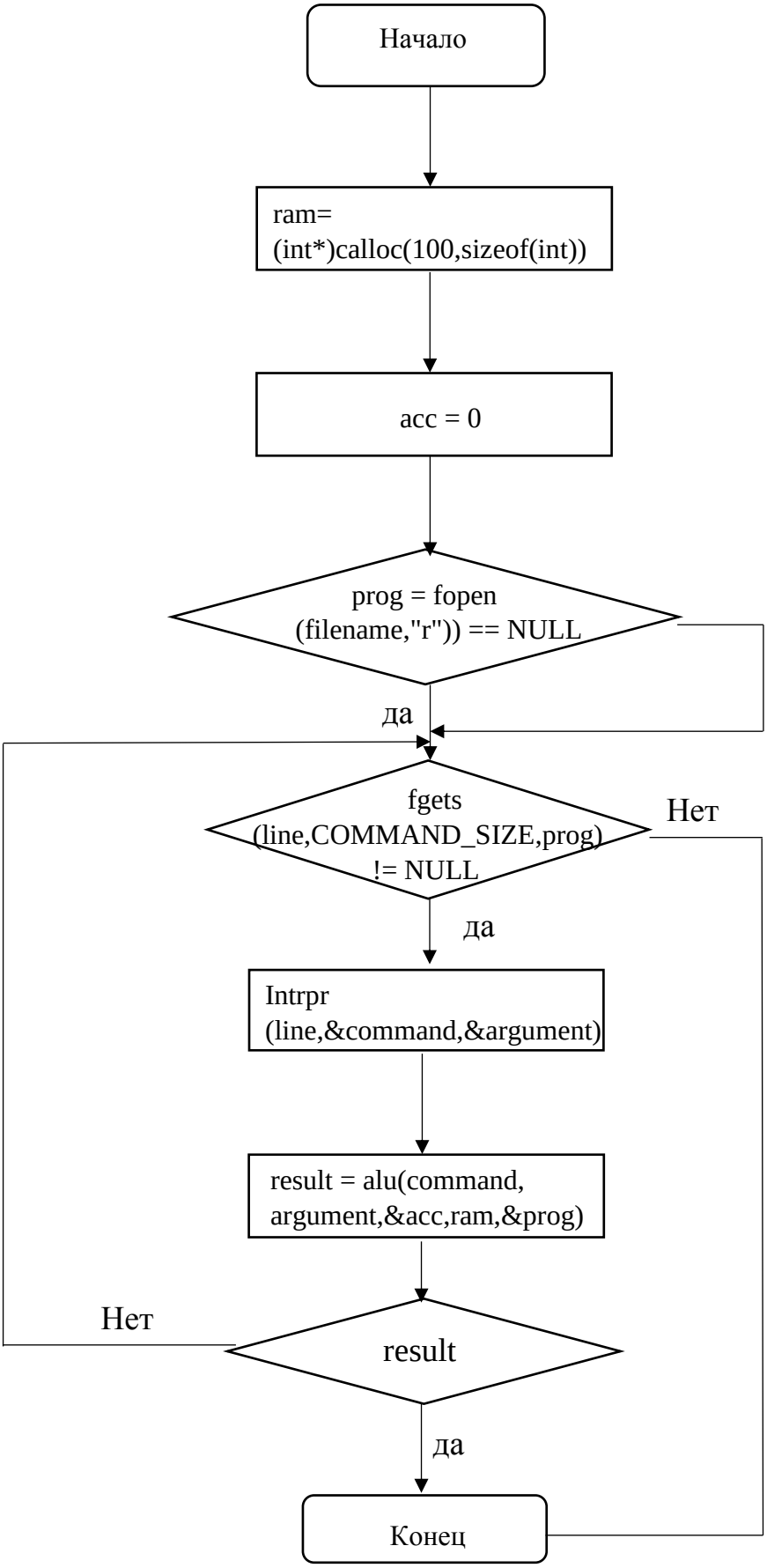

Начало
i = 0; line[i] != ' ' && line[i] != '\n';i++
comm[i] = line[i]
comm[i] = '\0'
!strcmp(comm,"ADD")
да
нет
*command = ADD
!strcmp(comm,"SUB")
да
нет
*command = SUB;
!strcmp(comm,"GOTO")
да
нет
*command = GOTO;
!strcmp(comm,"LOAD")
да
нет
*command = LOAD;
!strcmp(comm,"DIV")
да
нет
*command = DIV;
1
2
!strcmp(comm,"INPUT")
да
нет
*command = INPUT;
!strcmp(comm,"OUTPUT")
да
нет
*command = OUTPUT;
!strcmp (comm,"STORE")
да
нет
*command = STORE;
!strcmp (comm,"NEG_GOTO")
да
нет
*command = NEG_GOTO;
!strcmp (comm,"ZERO_GOTO")
да
нет
*command = ZERO_GOTO;
3
4



Блок схема 2.7 – Реализация ввода и обработки команд

```
void intrpr(char *line,int *command, int *argument) //получает команду и аргумент из потока ввода и преобразовывает буквенные комманды в числовые
{
        char comm[COMMAND_SIZE];
        char arg[COMMAND_SIZE];
        int i;
        for(i = 0; line[i] != ' ' && line[i] != '\n';i++)
    comm[i] = line[i];
  comm[i] = '\0';
  if(!strcmp(comm,"ADD"))
    *command = ADD;
  else if(!strcmp(comm,"SUB"))
                *command = SUB;
        else if(!strcmp(comm,"GOTO"))
                *command = GOTO;
        else if(!strcmp(comm,"LOAD"))
                *command = LOAD;
        else if(!strcmp(comm,"DIV"))
                *command = DIV;
        else if(!strcmp(comm,"INPUT"))
                *command = INPUT;
        else if(!strcmp(comm,"OUTPUT"))
                *command = OUTPUT;
        else if(!strcmp(comm,"STORE"))
                *command = STORE;
        else if(!strcmp(comm,"NEG_GOTO"))
                *command = NEG_GOTO;
        else if(!strcmp(comm,"ZERO_GOTO"))
                *command = ZERO_GOTO;
        else if(!strcmp(comm,"HALT"))
                *command = HALT;
        else
                *command = atoi(comm);
        for(; i != EOF && i != '\n';i++)
                arg[i] = line[i];
        *argument = atoi(arg);
}
```

Листинг 2.8 – Реализация ввода и обработки команд



Блок схема 2.9 – Реализация обработки команд и работа с памятью

```
void computer(char *filename) // считывает значение из файла с операциями и передает
аргументы в след ф-ю, где на основе выбора вып. операции
{
	int *ram;
	char line[COMMAND_SIZE];
	int command;
	int argument;
	int result;
	ram=(int*)calloc(100,sizeof(int));
	int acc = 0;
	FILE *prog;
	 if((prog = fopen(filename,"r")) == NULL){
	printf("do not open this file");
	}
	printf("Вывод: \n");
	while(fgets(line,COMMAND_SIZE,prog) != NULL){

		intrpr(line,&command,&argument);
		result = alu(command,argument,&acc,ram,&prog);
		if(result)
			break;
	}

}
```

Листинг 2.10 – Реализация обработки команд и работа с памятью

Приложение А

```
#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define COMMAND_SIZE 30

#define ADD 30

#define SUB 31

#define GOTO 40

#define LOAD 20

#define DIV 32

#define INPUT 10

#define OUTPUT 11

#define STORE 21

#define NEG_GOTO 41

#define ZERO_GOTO 42

#define HALT 43

#define help printf("help - помощь\nrun - выполнить\nclear - очистить экран\nexit - выйти из shell\n")

#define COMMAND_LENGTH 100


int *ram;

char line[COMMAND_SIZE];

int command;

int argument;

int result;

int acc = 0;

int i = 0, j = 0, i_start = 0, i_end = 1, i_menu = 0;


char button();

void intrpr(char*,int*, int*);

int alu(int,int,int*,int*,FILE **);
```

```
void computer();
void run();
void intrpr();
void logo();
void menu();
void output_menu();
int joy_menu(int *);
void display();
char button();
int alu(int command,int argument,int *acc, int *ram,FILE **prog)
{
                                        switch(command){
                                        case ADD:
                                                *acc += ram[argument];
                                        break;
                                        case GOTO:
                                                fseek(*prog,argument,SEEK_SET);
                                                break;
                                        case LOAD:
                                                *acc = ram[argument];
                                                break;
                                        case SUB:
                                                *acc -= ram[argument];
                                                break;
                                        case DIV:
                                                *acc /= ram[argument];
                                                break;
                                        case INPUT:
                                                scanf("%d", &ram[argument]);
                                                break;
                                        case OUTPUT:
```

```
                printf("%d\n",*acc);
                break;
            case STORE:
                ram[argument] = *acc;
                break;
            case NEG_GOTO:
                if(*acc < 0)
                fseek(*prog,argument,SEEK_SET);
                break;
            case ZERO_GOTO:
                if(*acc == 0)
                fseek(*prog,argument,SEEK_SET);
                break;
            case HALT:
                return 1;
                break;
            }
        return 0;
}
void computer(char *filename)
{
    int *ram;
    char line[COMMAND_SIZE];
    int command;
    int argument;
    int result;
    ram=(int*)calloc(100,sizeof(int));
    int acc = 0;
    FILE *prog;
     if((prog = fopen(filename,"r")) == NULL){
    printf("ERROR!\n");
```

```
        }
        while(fgets(line,COMMAND_SIZE,prog) != NULL){

                intrpr(line,&command,&argument);
                result = alu(command,argument,&acc,ram,&prog);
                if(result)
                        break;
        }

}
void run()
{
        char ex[2][30] = {"exit", "help"};
        printf("shell:> ");
        char command[COMMAND_LENGTH];
   while(fgets(command,COMMAND_LENGTH,stdin) != NULL){
        char *h=strchr(command,'\n');
        *h = '\0';
                command[COMMAND_LENGTH-1] = '\0';
                if(!strncmp(command,"run ",4)) {
                        computer(strchr(command,' ')+1);
                        continue;
                } else if(strcmp(ex[0],command) == 0) {
                        exit(1);
                } else if (strcmp(ex[1],command) == 0) {
                        help;
                }
                printf("shell:> ");
        system(command);
   }
}
```

```
void intrpr(char *line,int *command, int *argument)
{
                        char comm[COMMAND_SIZE];
                        char arg[COMMAND_SIZE];
                        int i;
                        for(i = 0; line[i] != ' ' && line[i] != '\n';i++)
                                comm[i] = line[i];
                        comm[i] = '\0';
                        if(!strcmp(comm,"ADD"))
                                *command = ADD;
                        else if(!strcmp(comm,"SUB"))
                                *command = SUB;
                        else if(!strcmp(comm,"GOTO"))
                                *command = GOTO;
                        else if(!strcmp(comm,"LOAD"))
                                *command = LOAD;
                        else if(!strcmp(comm,"DIV"))
                                *command = DIV;
                        else if(!strcmp(comm,"INPUT"))
                                *command = INPUT;
                        else if(!strcmp(comm,"OUTPUT"))
                                *command = OUTPUT;
                        else if(!strcmp(comm,"STORE"))
                                *command = STORE;
                        else if(!strcmp(comm,"NEG_GOTO"))
                                *command = NEG_GOTO;
                        else if(!strcmp(comm,"ZERO_GOTO"))
                                *command = ZERO_GOTO;
                        else if(!strcmp(comm,"HALT"))
                                *command = HALT;
                        else
```

```
                        *command = atoi(comm);
                for(; i != EOF && i != '\n';i++)
                        arg[i] = line[i];
                *argument = atoi(arg);
}
void logo()
{
        display();
        printf("\t\tCOMPUTER\n\t\tMenu:\n");
}
void menu()
{
                int flag = 1;
                while (flag == 1)
                {
                        logo();
                        output_menu(i_menu);
                        joy_menu(&flag);
                }
                if (i_menu == 0) {
                        display();
                        printf("Чтобы получить подсказку о командах, воспользуйтесь
help\n");
                        run();
                }
                if (i_menu == 2)
                        exit(1);
}
void output_menu(int kursor)
{
        if (kursor == 0)
                printf("\t\t[*] - Командная строка\n\t\t - Выход\n");
```

```
	if (kursor == 1)
		printf("\t\t - Командная строка\n\t\t[*] - Выход\n");
}
int joy_menu(int *flag)
{
	char select;
	select = button();
	switch(select)
	{
		case 13:
		{
			display();
			*flag = 0;
			return i_menu;
		}
		case 56: // вверх
		{
			if (i_menu == i_start) {
				i_menu = i_end;
				//output_menu(i);
				display();
				return i_menu;
			}
			if (i_menu > i_start) {
				i_menu--;
				//output_menu(i);
				display();
				return i_menu;
			}
		}
		case 50: // вниз
```

```
                {
                        if (i_menu == i_end) {
                                i_menu = i_start;
                                //output_menu(i);
                                display();
                                return i_menu;
                        } else {
                                i_menu++;
                                //output_menu(i);
                                display();
                                return i_menu;
                        }
                }
                default:
                {
                        display();
                }
        }
}
void display()
{
        #ifdef _WIN32
                system("cls");
        #else
                system("clear");
        #endif
}
char button()
{
   char select;
   #if _WIN32
```

```
        select = getch();
    #else
        system("stty raw");
        select = getchar();
        system("stty cooked");
    #endif
    return select;
}
int main()
{
                        ram=(int*)calloc(100,sizeof(int));
                        menu();
                        FILE *prog;
                        if((prog = fopen("program.txt","r")) == NULL)
                                return -1;
                        while(fgets(line,COMMAND_SIZE,prog) != NULL){
                                intrpr(line,&command,&argument);
                                result = alu(command,argument,&acc,ram,&prog);
                                if(result)
                                        break;
                        }
                        return 0;
}
```

Приложение Б

Список литературы

[1].Васильев А.Н. Программирование на С в примерах и задачах. — Москва: Издательство «Э», 2017 —560 с.

[2].Ссылка на web-страницу

Язык Си (язык программирования): [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Си_(язык_программирования) (Дата обращения: 02.06.2019).

[3]. Ссылка на на web-страницу

GitHub:[Электронный ресурс] URL: (Дата обращения:20.05.2019)

[4].Ссылка на web-страницу

Числовые ряды, их суммы, сходимость, примеры: [Электронный ресурс] URL:

https://function-x.ru/rows1.html (Дата обращения: 30.04.2019)

[5]. Б.У. Керниган, Д.М. Ритчи.Язык программирования С.—Санкт-Петербург: Издательство

«Невский диалект»,2001 — 352 с.

[6]. Ссылка на сайт в целом

WolframAlpha :[Электронный ресурс] URL: https://www.wolframalpha.com (Дата обращения:10.04.2019)

[7]. Ссылка на web-страницу

Метод секущих: [Электронный ресурс] URL:http://www.machinelearning.ru/wiki/index.php?title=Ме.. (Дата обращения: 17.03.2019)

[8]. Ссылка на web-страницу

Метод Ньютона:[Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Метод_Ньютона(Дата обращения: 23.04.2019)

[9]. Как найти сумму ряда: примеры решений, определение: [Электронный ресурс] URL: https://математика24.рф/kak-najti-summu-ryada.html(Дата обращения: 26.04.2019)

[10]. Скляров В.А. Программирование на языках С и С++ - М.: Высшая школа, 1996.