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

**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 C, 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**

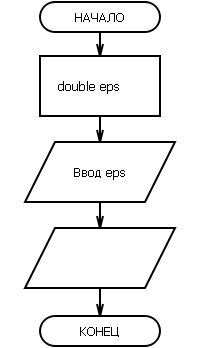
Целью данного задания является вычисление заданного выражения в варианте №19 и подсчитать, сколько членов ряда и цепной дроби понадобится для нахождения суммы.

**Дано**: Соотношение С.Рамануджанова, имеющего вид:

Формула 1.1.1 - Соотношение С.Рамануджанова;

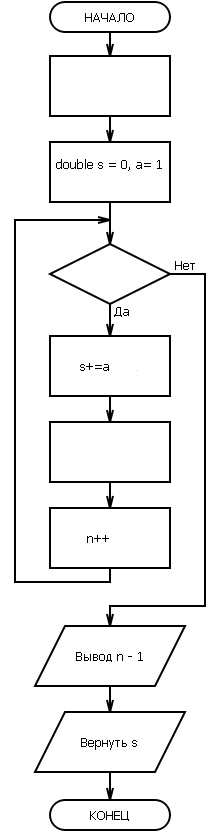
**Найти**: Вычислить, сколько членов ряда и цепной дроби нужно взять, чтобы достичь заданной точности.

**Решение**: Приведем для начала блок-схему:



Вывод FindRoot() +

FindDrob()

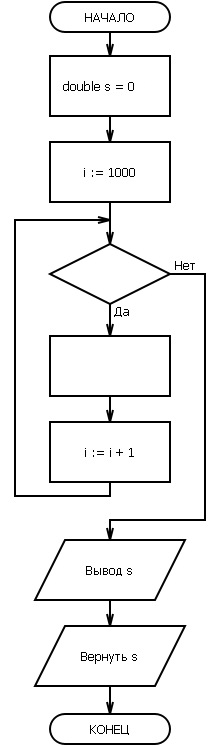


s+=a

a \*= 1.0/(2.0 \* n + 1.0)

fabs(a) > eps

n = 0



s := 1/(1 + i \* s)

i := i - 1

i > 0

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 для

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

|  |  |
| --- | --- |
| Переменная | Значение в программе, заданное пользователем |
| e | 0.00001 |

Листинг 1.1.9 –Листинг программного кода;

|  |  |
| --- | --- |
| Переменная | Вывод суммы числового ряда |
| s | 1.410686 |

Листинг 1.1.9 –Листинг программного кода;

|  |  |
| --- | --- |
| Переменная | Кол-во членов ряда |
| n | 6 |

Листинг 1.1.9 –Листинг программного кода;

|  |  |
| --- | --- |
| Переменная | Кол-во членов дроби |
| i | 1000 |

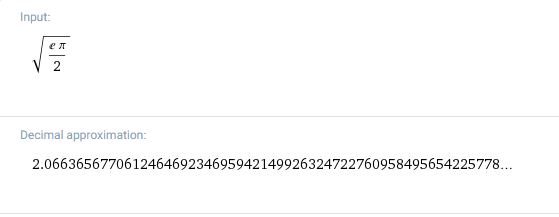
Листинг 1.1.9 –Листинг программного кода;

|  |  |
| --- | --- |
| Переменная | Вывод суммы цепной дроби |
| s | 0.655680 |

Листинг 1.1.9 –Листинг программного кода;

|  |  |
| --- | --- |
| Переменная | Вывод суммы членов ряда и цепной дроби |
| s | 2.066365 |

Листинг 1.1.9 –Листинг программного кода;



Листинг 1.1.9 –Листинг программного кода;

**В заключении** можно сказать, что на примере С.Рамануджанова мы рассмотрели одни 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**

**Дано:**

Уравнение: (уравнение №1).

Уравнение: (уравнение №5).

Уравнение: (уравнение №6).

**Найти:**

Значение **x** методом деления отрезка пополам.

**Решение:**

Значение переменных a, b и e (точность) вводятся с клавиатуры пользователем.

Для начала рассмотрим блок-схему будущей программы.

На следующей странице представлен листинг программного кода, где рассмотрен метод половинного деления на уравнении №6.#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define eps 0.00001

double f(double,double,double);

double FindRoot(double,double,double,double,double);

double f(double x,double c,double d)

{

return sin(c\*x) - d;

}

double FindRoot(double a,double b,double x,double c, double d)

{

double m; int i = 0;

double lenght = b-a;

if(fabs(f(a,c,d))<eps)

return a;

else if(fabs(f(b,c,d))<eps)

return b;

while(lenght>=eps){

m = (a+b)/2;// seredina otrezka

if(fabs(f(m,c,d))<eps)

return m;

if(f(a,c,d) \* f(m,c,d) < 0)

b = m;

else

a = m;

lenght/=2; i++;

}

return m;

}

int main()

{

double a,b,c,d,x;

printf("Введите длину a и b: "); scanf("%lf%lf",&a,&b);

printf("Введите c и d: "); scanf("%lf%lf",&c,&d);

if(f(a,c,d)\*f(b,c,d)>0)

printf("Ф-я не имеет корней на данном отрезке\n");

x = FindRoot(a,b,x,c,d);

printf("x = %lf, f(x) = %lf\n",x,f(x,c,d));

return 0;

}

Листинг 1.1.9 –Листинг программного кода дл

Рассмотрим уравнение №1:

**=0**

Листинг 1.1.9 –Листинг программного кода дл

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | d |
| 1 | 2 | -1.5 | -0.5 |
| 1 | 2 | -1.6 | -0.4 |
| -2 | -1 | -1.9 | -0.5 |
| -1.5 | -0.5 | -2.9 | -0.5 |

Листинг 1.1.9 –Листинг программного кода дл

|  |  |
| --- | --- |
| i | Значение х |
| 14 | 1.745331 |
| 11 | 1.706299 |
| 14 | -1.929047 |
| 15 | -0.902756 |

Листинг 1.1.9 –Листинг программного кода дл

Рассмотрим уравнение №5:

Листинг 1.1.9 –Листинг программного кода дл

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

|  |  |
| --- | --- |
| i | х |
| 17 | 1.378565 |
| 18 | 1.558512 |
| 18 | 1.228661 |
| 20 | 1.848537 |

Листинг 1.1.9 –Листинг программного кода дл

Листинг 1.1.9 –Листинг программного кода дл

Уравнение №6:

Листинг 1.1.9 –Листинг программного кода дл

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

|  |  |
| --- | --- |
| i | х |
| 17 | 1.378565 |
| 18 | 1.558512 |
| 18 | 1.228661 |
| 20 | 1.848537 |

Листинг 1.1.9 –Листинг программного кода дл

Листинг 1.1.9 –Листинг программного кода дл

Листинг 1.1.9 –Листинг программного кода дл

**1.2.2 Метод секущих**

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

**Дано:**

Уравнение: (уравнение №1).

Уравнение: (уравнение №5).

Уравнение: (уравнение №6).

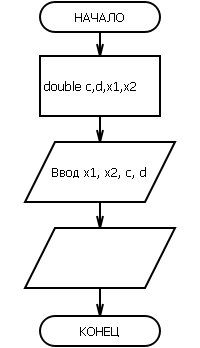
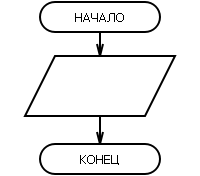
**Найти:**

Значение **x** методом секущих.

**Решение:**

Значение переменных **a**, **b**, **c**, **d** вводятся с клавиатуры пользователем.

Для начала рассмотрим блок-схему будущей программы:

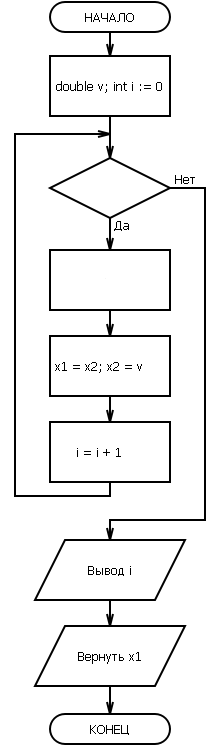


Вывод

decision

Вернуть

pow(x,4)+c\*pow(x,3)-d\*x



v = (x1 \* fx(x2, c, d) - x2 \* fx(x1, c, d)) / (fx(x2, c, d) - fx(x1, c, d))

fabs(x1 - x2) > eps

Листинг 1.1.9 –Листинг программного кода дл

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; //х1, х2 - 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; //

}

Таблица 1.1.9 – Листинг программного кода для

Рассмотрим уравнение №5:

Таблица 1.1.9 – Листинг программного кода для

|  |  |  |  |
| --- | --- | --- | --- |
| x1 | x2 | c | d |
| 1 | 2 | 2 | 5 |

Таблица 1.1.9 – Листинг программного кода для

|  |  |
| --- | --- |
| i | F(x) |
| 9 | 1.241897 |

Таблица 1.1.9 – Листинг программного кода для



Таблица 1.1.9 – Листинг программного кода для

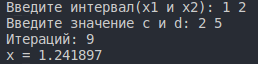


Таблица 1.1.9 – Листинг программного кода для

Рассмотрим уравнение №6:

Таблица 1.1.9 – Листинг программного кода для

|  |  |  |  |
| --- | --- | --- | --- |
| x1 | x2 | c | d |
| 1 | 2 | 2 | 5 |

Таблица 1.1.9 – Листинг программного кода для

|  |  |
| --- | --- |
| i | F(x) |
| 9 | 1.174933 |

Таблица 1.1.9 – Листинг программного кода для

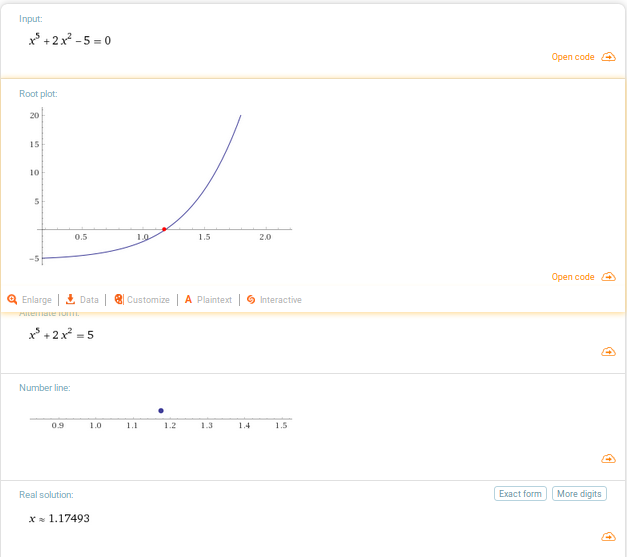


Таблица 1.1.9 – Листинг программного кода для

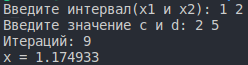


Таблица 1.1.9 – Листинг программного кода для

Рассмотрим уравнение №1:

Таблица 1.1.9 – Листинг программного кода для

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | d |
| 0 | 1 | 1 | 2 |

Таблица 1.1.9 – Листинг программного кода для

|  |  |
| --- | --- |
| i | F(x) |
| 39 | 0.785398 |

Таблица 1.1.9 – Листинг программного кода для

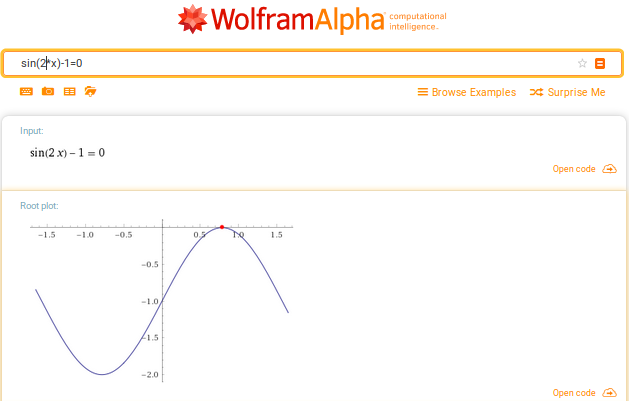
****

Таблица 1.1.9 – Листинг программного кода для

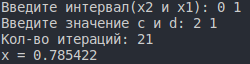


Таблица 1.1.9 – Листинг программного кода для

**Используя данные программы и сервиса WolframAlpha проверим, насколько верен результат между двумя методами, опираясь на количество итераций.**

**Уравнение №5:**

**Метод половинного деления:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | i | F(x) |
|  |  |  |  |  |  |

Таблица 1.1.9 – Листинг программного кода для

**Метод секущих:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x1 | x2 | c | d | i | F(x) |
| 1 | 2 | 2 | 5 | 9 | 1.241897 |

Таблица 1.1.9 – Листинг программного кода для

**Уравнение №6:**

**Метод половинного деления:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | i | F(x) |
|  |  |  |  |  |  |

Таблица 1.1.9 – Листинг программного кода для

**Метод секущих:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | i | F(x) |
| 1 | 2 | 2 | 5 | 9 | 1.174933 |

Таблица 1.1.9 – Листинг программного кода для

**Уравнение №1:**

**Метод половинного деления:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | i | F(x) |
|  |  |  |  |  |  |

Таблица 1.1.9 – Листинг программного кода для

**Метод секущих:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | i | F(x) |
| 0 | 1 | 1 | 2 | 39 | 0.785398 |

Таблица 1.1.9 – Листинг программного кода для

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.

Конец

return 1

return 0

HALT

ZERO\_GOTO

NEG\_GOTO

STORE

OUTPUT

INPUT

DIV

SUB

LOAD

GOTO

ADD

fseek

(\*prog,argument,SEEK\_SET)

fseek

(\*prog,argument,SEEK\_SET)

нет

нет

да

да

\*acc == 0

\*acc < 0

Вывод \*acc

scanf("%d", &ram[argument])

\*acc /= ram[argument]

\*acc -= ram[argument]

\*acc = ram[argument]

fseek

(\*prog,argument,SEEK\_SET)

ram[argument] = \*acc

acc +=ram[argument]

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;

}

Начало

i = 0; line[i] != ' ' && line[i] != '\n';i++

comm[i] = '\0'

comm[i] = line[i]

да

!strcmp(comm,"ADD")

нет

2

1

\*command = ADD

да

\*command = NEG\_GOTO;

\*command = ZERO\_GOTO;

\*command = STORE;

да

\*command = OUTPUT;

да

\*command = INPUT;

!strcmp(comm,"INPUT")

да

!strcmp(comm,"SUB")

\*command = SUB;

нет

нет

!strcmp(comm,"OUTPUT")

да

\*command = GOTO;

!strcmp(comm,"GOTO")

нет

нет

\*command = LOAD;

!strcmp

(comm,"STORE")

да

да

!strcmp(comm,"LOAD")

нет

!strcmp(comm,"DIV")

\*command = DIV;

нет

!strcmp

(comm,"NEG\_GOTO")

да

нет

нет

!strcmp

(comm,"ZERO\_GOTO")

да

1

нет

2

4

3

3

4

нет

да

\*command = HALT;

!strcmp(comm,"HALT")

нет

\*command = atoi(comm);

; i != EOF && i != '\n';i++

\*argument = atoi(arg);

arg[i] = line[i];

Конец

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);

}

Начало

ram=

(int\*)calloc(100,sizeof(int))

acc = 0

prog = fopen

(filename,"r")) == NULL

да

Нет

fgets

(line,COMMAND\_SIZE,prog)

!= NULL

да

Intrpr

(line,&command,&argument)

result = alu(command,

argument,&acc,ram,&prog)

Нет

result

да

Конец

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;

}

}

**Список литературы**