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**

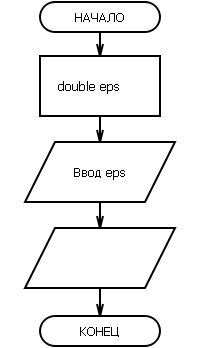
Целью данного задания является вычисление заданного выражения в варианте №19 и подсчитать, сколько членов ряда и цепной дроби понадобится для нахождения суммы.

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

Формула 1.1.1 - Соотношение С.Рамануджанова;

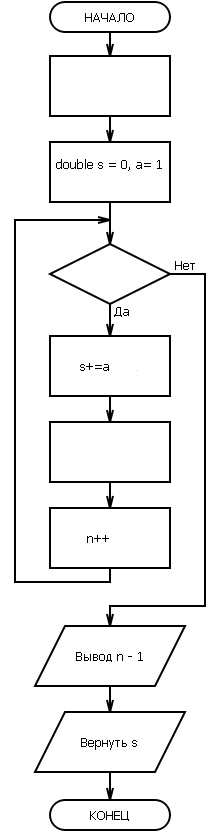
**Найти**: Вычислить, сколько членов ряда и цепной дроби нужно взять, чтобы достичь заданной точности.

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



Вывод FindRoot() +

FindDrob()

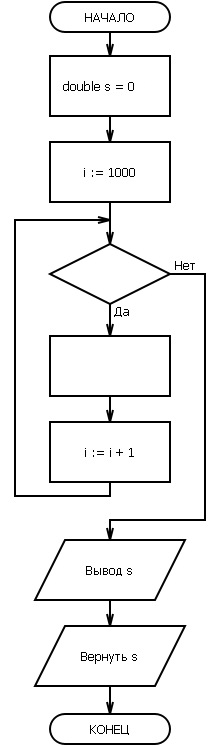


s+=a

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 |

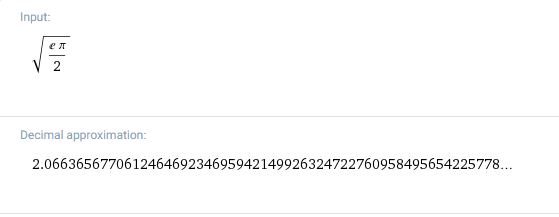
Листинг 1.1.9 –Листинг программного кода;

|  |  |
| --- | --- |
| 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.// подключаем библиотеки

#include<stdio.h>

#include<stdlib.h>

#include<math.h> // библиотека для работы с математическими операциями

double F(double, double, double); // объявлем прототипы функций

double FindRoot(double (\*f)(double, double, double), double, double, double, double,double);

double F(double x, double c, double d)// функция для полинома

{

return pow(x,5)+c\*pow(x,2)-d; // возвращаем уравнение

}// функция нахождения корня

double FindRoot(double (\*f)(double, double, double), double a, double b, double c, double d,

double eps){

double cde;

while ((b - a) > eps)

{

cde = (a + b) / 2;

if ((f(a, c, d) \* f(cde, c, d)) > 0) a = cde;

else b = cde; printf("%lf\n", cde);

}

return cde;

}

int main()

{

double a, b, eps, x;

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

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

printf("Введите точность(eps(например: 0.0001)): "); scanf("%lf", &eps);

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

printf("Значение x = %lf\n", x);

return 0;

}

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).

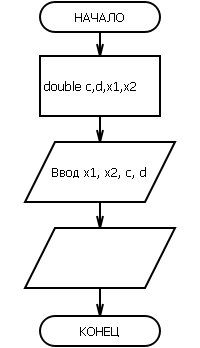
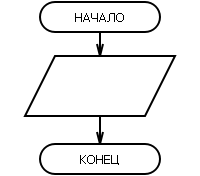
**Найти:**

Значение **x** методом секущих.

**Решение:**

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

Для начала рассмотрим блок-схему будущей программы:

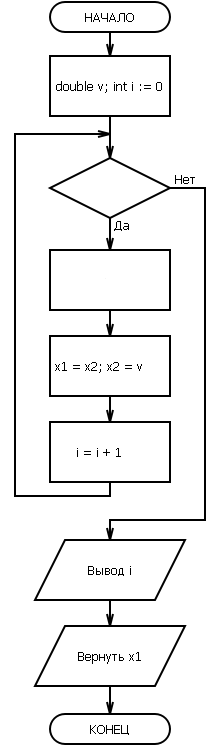


Вывод

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>

#include <math.h>

#define eps 0.00001

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 c, double d) // вычисляемая функция

{

return sin(c \* x) - d;

}

double decision(func fx, double x1, double x2, double c, double d)

{

double v, y; int i= 0;

do // пока не достигнута точность eps(0.0000001)

{

y = v;

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

x1 = x2; x2 = v;

} while (fabs(y - v) >= eps);

return x1;

}

int main()

{

double c, d;

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);// Вывод в консоль интервала

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

printf("x = %lf\n", decision(fx, x1, x2, c, d)); // Вывод в консоль ответа

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

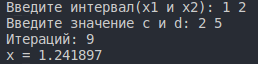


Таблица 1.1.9 – Листинг программного кода для

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

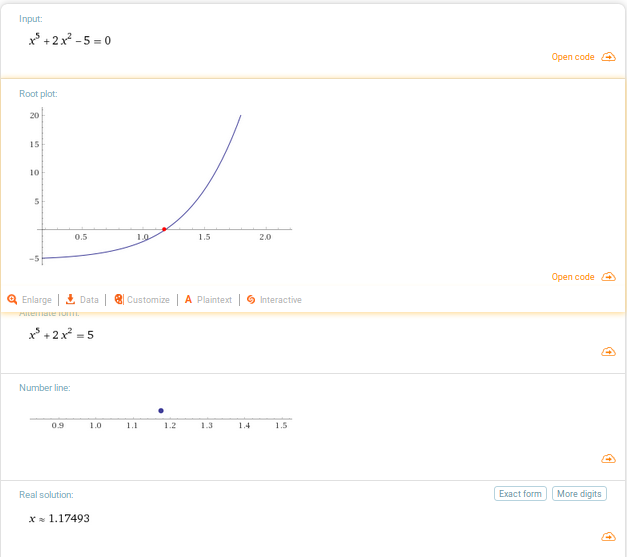


Таблица 1.1.9 – Листинг программного кода для

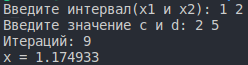


Таблица 1.1.9 – Листинг программного кода для

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

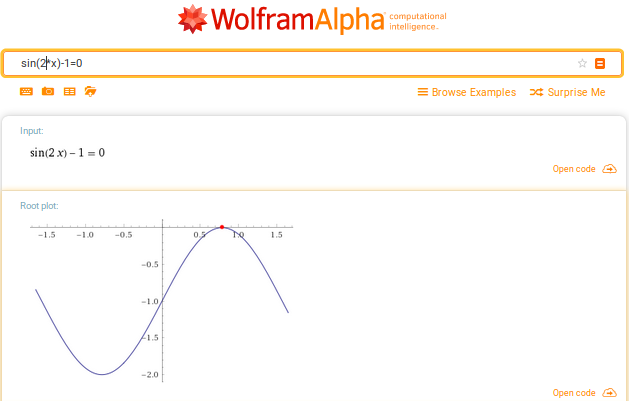
****

Таблица 1.1.9 – Листинг программного кода для

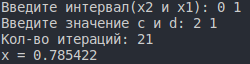


Таблица 1.1.9 – Листинг программного кода для

**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.



Таблица 1.1.9 – Листинг программного кода для нахождения соотношения с помощью числового ряда;

В данной части программы, пользователь может выбрать с помощью команд управления интересующий его пункт меню. В данном случае, управления реализуется с помощью библиотеки conio.h.

На следующей странице представлен листинг работы раздела меню программы.

void menu(int \*ram, FILE \*battary, FILE \*command, int \*code\_list)

{

char select;

int flag = 1, i = 0;

while (flag == 1)

{

logo();

printf("\t\t\t\tWelcome!\n");

printf("\t\tMenu: \n\n");

output\_menu(i);

select = getch();

switch(select)

{

case 13:

{

if (i == 1) {

list\_command(command, code\_list);

}

if (i == 4) {

flag = 0;

break;

}

if (i == 3) {

run\_shell(command, code\_list, battary);

}

if (i == 2) {

run\_command(command, code\_list);

}

if (i == 0) {

input\_command();

}

}

case 119:

{

if (i == 0) {

i = 4;

output\_menu(i);

system("cls");

}

if (i > 0) {

i--;

output\_menu(i);

system("cls");

}

}

case 115:

{

if (i == 4) {

i = 0;

output\_menu(i);

system("cls");

} else {

i++;

output\_menu(i);

system("cls");

}

}

}

}

}

**Список литературы**