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 №11**

Выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студентка группы ИВТ-11 Лачугина Н.В.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доцент, к. т. н. Артамонов Ю. Н.

Котельники – 2019

**Оглавление**

[Раздел 1 – Разработка численных алгоритмов. 3](#_Toc9451248)

[Часть 1 – Суммирование рядов и вычисление элементарных функций. 3](#_Toc9451249)

[Часть – 2 Приближённые методы нахождения корней уравнения 10](#_Toc9451250)

**Раздел 1 – Разработка численных алгоритмов.**

## **Часть 1 – Суммирование рядов и вычисление элементарных функций.**

В варианте 11 представлено задание на сходимость.

**Дано:** представление в виде цепной дроби:

Формула 1.1.1 Цепная дробь tan (x)

**Найти:** вычислить **tan (x)** и проверить, сколько членов дроби надо взять для получения результата с заданной точностью при различных значениях **х**.

**Решение:** Ниже представлен листинг данной программы с пояснением в виде комментариев.

|  |
| --- |
| // подключение библиотек  //стандартный заголовочный файл ввода-вывода  **#include<stdio.h>**  // математическая библиотека  **#include<math.h>**  // объявление константы числа Пи  **#define PI 3.1415926535897**  //прототип функции seriesTg  **int seriesTg(double, double, double\*);**  //прототип функции compute  **double compute(int, double);**  // главная функция main ()  **int main()**  **{**  // объявление переменных типа double  **double x, epsilon, result;**  **printf ("Введите x: ");**  // ввод значения x  **scanf ("%lf", &x);**  **printf ("Введите epsilon: ");**  // ввод значения epsilon  **scanf ("%lf", &epsilon);**  // вызов функции seriesTg  **seriesTg (x,epsilon,&result);**  // вывод на экран значения epsilon, tan(x), result для сравнения при заданной точности  **printf ("Epsilon: %25.20lf\nTg: %30.20lf\nResult: %26.20lf\n", epsilon, tan(x), result);**  // успешное выполнение программы  **return 0;**  **}**  // объявление функции seriesTg  **int seriesTg (double x, double epsilon, double \*result)**  **{**  // объявление переменной типа int  **int n = 1;**  // объявление переменных left и right типа double и присваивание им значений  **double left = tan(x);**  **double right = 1;**  // цикл пока модуль левой части минус правая часть уравнения меньше заданной точности  **while (fabs (left - right) >= epsilon ){**  // если условие выполняется, то правой части присваивается значение функции compute (n, x)  **right = compute(n, x);**  // постинкремент  **n++;**  // вывод на экран значений для проверки количества интераций, необходимых для заданной точности  **printf (“Член ряда: %d\nВывод ряда в зависимости от х: %lf\n”, n-1, right);**  **}**  // присваивание переменной right к result  **\*result = right;**  // возврат значения n  **return n;**  **}**  // объявление функции compute  **double compute (int n, double x)**  **{**  // объявление переменных k и result типа double  **double k, result = 0;**  // цикл пока член ряда n больше нуля выполняем  **while (n > 0) {**  // присваиваем переменной k выражение  **k = n\*2-1;**  // присваиваем переменной result выражение подсчёта цепной дроби  **result = 1.0/(k/x-result);**  // прединкремент  **--n;**  **}**  // возврат значения result  **return result;**  **}** |

Листинг 1.1.2 – листинг программного кода для нахождения соотношения с помощью цепной дроби.

Приведём таблицы входных и выходных данных переменных x, epsilon, и результат.

|  |  |
| --- | --- |
| Переменная | Значение в программе, заданное пользователем |
| epsilon | 0.001 |

Таблица 1.1.3 – входные данные для задания точности

|  |  |
| --- | --- |
| Переменная | Значение в программе, заданное пользователем |
| х | 1 |

Таблица 1.1.4 –входные данные для значения х

|  |  |
| --- | --- |
| Член ряда n | Вывод ряда в зависимости от х |
| 1 | 1.000000 |
| 2 | 1.500000 |
| 3 | 1.555556 |
| 4 | 1.557377 |

Таблица 1.1.5 – выходные данные членов цепной дроби

|  |  |
| --- | --- |
| Переменная | Вывод ряда цепной дроби |
| left | 1.557407 |

Таблица 1.1.6 – выходные данные цепной дроби

|  |  |
| --- | --- |
| Переменная | Вывод ряда цепной дроби |
| result | 1.557377 |

Таблица 1.1.7 – выходные данные для проверки на сходимость

При epsilon = 0.001 ряд сходится, используя 4 интерации.

Приведём блок-схему, соответствующую данной программе.

Начало Начало

double x, epsilon, result double k, result = 0

ввод х, epsilon

Нет

n > 0

seriesTg (x, epsilon, &result)

Да

k = n\*2-1

вывод epsilon, tan(x), result

return right

result = 1.0/(k/x-result)

Конец

Конец

--n

Рис. 1.1.8 int main ()

Рис. 1.1.9 double compute (int n, double x)

Начало

int x

double left = tan(x), right = 1

(fabs(left – right) >= epsilon)

\*result = right

Да

right = compute(n,x)

return n

n++

Конец

Вывод n-1, right

Рис. 1.1.10 – int seriesTg (double x, double epsilon, double \*result)

Для проверки попробуем ввести другое значение х и epsilon.

|  |  |
| --- | --- |
| Переменная | Значение в программе, заданное пользователем |
| epsilon | 0.000001 |

Таблица 1.1.11 – входные данные для задания точности

|  |  |
| --- | --- |
| Переменная | Значение в программе, заданное пользователем |
| х | 1.5 |

Таблица 1.1.12 –входные данные для значения х

|  |  |
| --- | --- |
| Член ряда n | Вывод ряда в зависимости от х |
| 1 | 1.500000 |
| 2 | 6.000000 |
| 3 | 12.750000 |
| 4 | 14.042553 |
| 5 | 14.100000 |
| 6 | 14.101397 |
| 7 | 14.101420 |

Таблица 1.1.13 – выходные данные членов цепной дроби

|  |  |
| --- | --- |
| Переменная | Вывод ряда цепной дроби |
| left | 14.101419 |

Таблица 1.1.14 – выходные данные цепной дроби

|  |  |
| --- | --- |
| Переменная | Вывод ряда цепной дроби |
| result | 14.101419 |

Таблица 1.1.15 – выходные данные для проверки на сходимость

При новых значениях x и epsilon ряд сходится, но уже при 7 интераций.

1**.2 Вывод по части 1**

Таким образом, при заданных значениях x и epsilon цепная дробь действительно сходится, что доказывают результаты таблицы 1.1.3 – 1.1.7 и 1.1.11 – 1.1.15.

**Часть – 2 Приближённые методы нахождения корней уравнения**

В варианте 11 необходимо проанализировать такие методы, как:

* Метод касательных;
* Метод секущих.

А также сравнить число интераций при одном и том же значении точности вычисления.

Рассмотрим для начала первый метод – метод касательных.

Предположим, что на отрезке [a, b] есть корень уравнения y=f(x), а также функция непрерывна на этом отрезке, и на интервале от (-∞;a] ᵕ [b, +∞) существуют различные от нуля пределы. Для нахождения корня по методу касательных необходимо взять произвольную точку **хn** и провести касательную в этой точке к заданной функции. Допустим, что касательная пересечёт ось абцисс в точке **хn+1**, тогда мы уже отталкиваемся от точки **xn+1**и проводим касательную в той точке к заданной функции.

Для сравнения необходимо использовать данные уравнения:

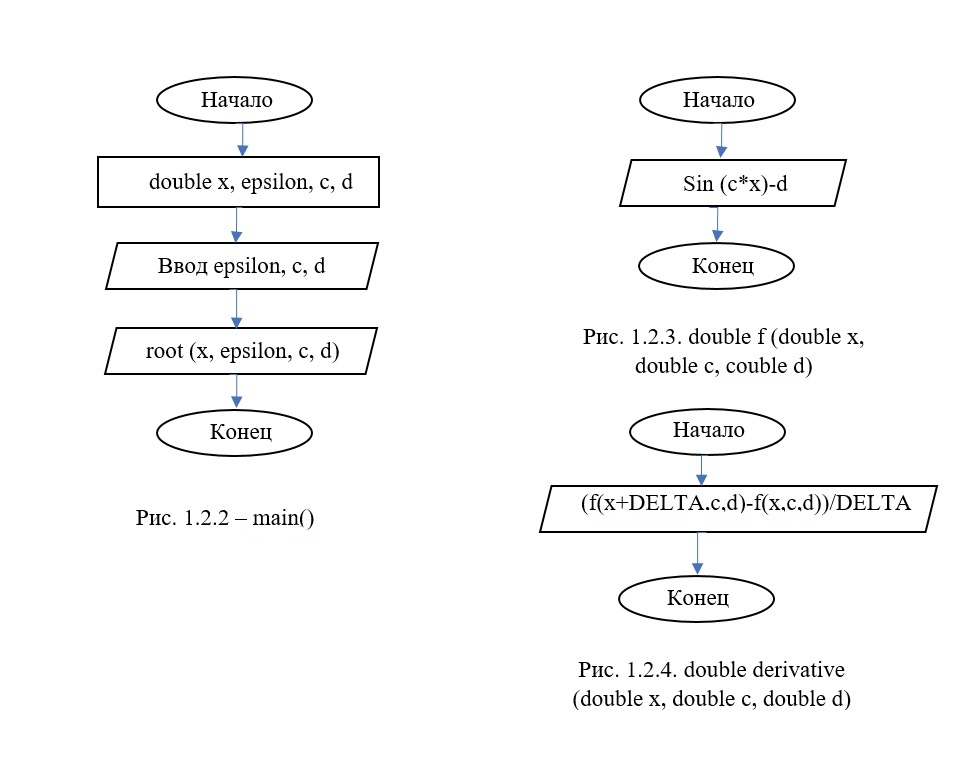
* Уравнение 1.
* Уравнение 5.
* Уравнение 6.

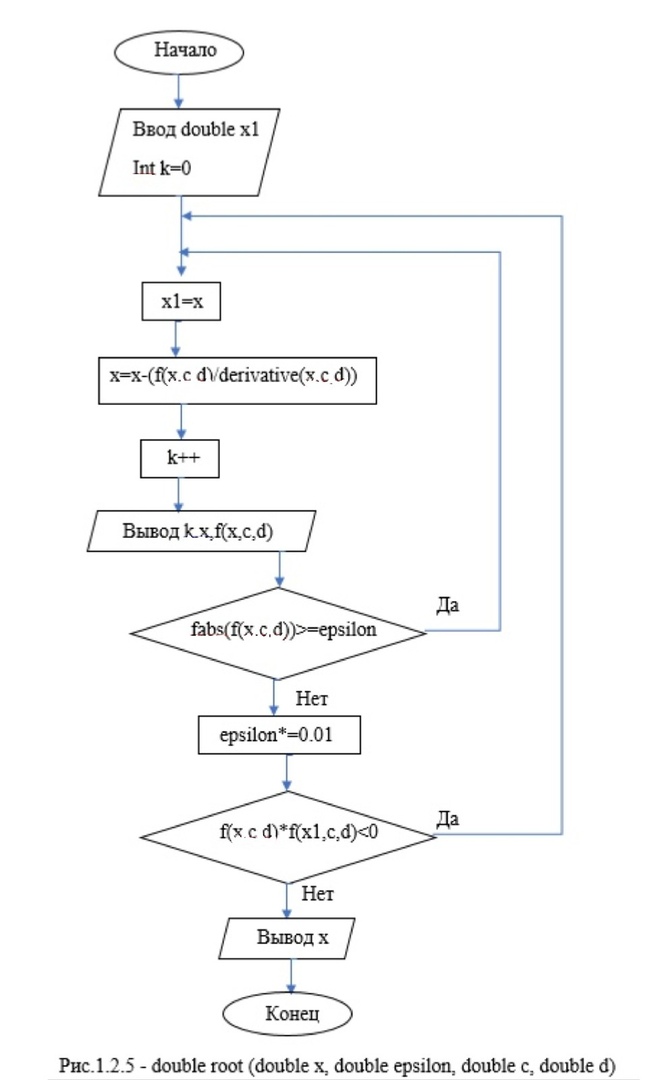
Ниже представлен листинг данного метода с пояснением в виде комментариев.

|  |
| --- |
| // подключение библиотек  // стандартный заголовочный файл ввода-вывода  **#include <stdio.h>**  // математическая библиотека  **#include <math.h>**  // объявление константы  **#define DELTA 1e-2**  // прототип функции f - функция  **double f(double, double, double );**  // прототип функции root - поиск корня  **double root (double, double, double, double );**  // прототип функции derivative - производная  **double derivative (double, double, double );**  // главная функция main ()  **int main (){**  // объявление переменных типа double  **double x=0, epsilon, xn, c, d;**  **printf ("введите точность: ");**  // ввод значения epsilon  **scanf ("%lf", &epsilon);**  // ввод значения c  **printf ("Введите с-угол наклона: ");**  **scanf("%lf",&c);**  // ввод значения d  **printf ("Введите d: ");**  **scanf("%lf",&d);**  // вывод значения, которое возвращает функция root()  **printf ("%lf\n", root (x, epsilon,c,d));**  // успешное выполнение программы  **return 0;**  **}**  // объявление функции f - запись уравнения  **double f (double x, double c, double d) {**  //возврат значения функции  **return sin(c\*x)-d;**  **}**  // объявление функции root - считает корень  **double root (double x, double epsilon, double c, double d) {**  // объявляем переменную типа double  **double x1;**  // объявляем переменную k для подсчёта интераций  **int k=0;**  // цикл с предусловием  **do{**  // цикл с предусловием  **do{**  // присваивание переменной х1 значение х  **x1 = x;**  // присваиваем переменной х выражение последовательное приближение х корня  **x = x-(f(x,c,d)/derivative(x,c,d));**  // постинкремент  **k++;**  // вывод на экран значений k, x, f(x, c, d)  **printf("%d)x=%lf f(x)=%.10lf\n",k,x,f(x,c,d));**  }  // пока абсолютное значение функции от аргумента х >= epsilon  **while(fabs(f(x,c,d))>=epsilon);**  // увеличиваем значение epsilon  **epsilon \*= 0.01;**  **}while(f(x,c,d)\*f(x1,c,d)<0);**  // возврат значения x  **return x;**  **}**  // объявление функции derivative - производная  **double derivative (double x, double c, double d){**  // возврат значения derivitive  **return (f(x+DELTA,c,d)-f(x,c,d))/DELTA;**  }  // объявление функции derivative - производная  **double derivative (double x){**  // возврат значения derivitive  **return (f(x+DELTA)-f(x))/DELTA;**  **}** |

Листинг 1.2.1 – листинг программного кода метода касательных.

Теперь представим программу в виде блок-схемы на следующей странице.





Рассмотрим теперь метод секущих. Этот метод состоит в нахождении корня, то есть точки пересечения функции с осью абцисс. Для начала нужно определить точки **хn** и **хn-1**, их должен ввести пользователь, при условии, что **хn** > **хn-1**. Далее находим значения в этих точках и проводим прямую с координатами: (**хn** ,f(**хn)**) и ( **хn-1,** f(**хn-1)),**  если есть корень, то будет точка пересечения с осью «х». Цикл нахождения будет продолжаться до тех пор, пока не будет достигнута точность вычислений (epsilon).

Этот корень можно найти по формуле:

[#include](https://vk.com/im?sel=187443967&st=%23include) <stdio.h>   
[#include](https://vk.com/im?sel=187443967&st=%23include) <math.h>   
  
double f(double, double , double );   
  
int main ( )   
{   
double x, x1, x2, epsilon, otv=0, y, c, d;   
  
printf ("введите интервал от xn-1 до xn\n");   
printf ("введите xn: ");   
scanf ("%lf", &x1);   
printf ("введите xn-1: ");   
scanf ("%lf", &x2);   
printf ("введите точность вычислений: ");   
scanf ("%lf", &epsilon);   
printf ("введите c: ");   
scanf ("%lf", &c);   
printf ("введите d: ");   
scanf ("%lf", &d);   
  
do {   
 y=otv;   
 otv=x2-((x2-x1)/(f(x2,c,d)-f(x1,c,d)))\*f(x2,c,d);   
 x1=x2;   
 x2=otv;   
}   
while (fabs(y-otv)>=epsilon);   
  
printf ("Ответ: %lf\n", otv);   
return 0;   
} 

double f(double x, double c, double d)   
{   
return sin(c\*x)-d;   
}