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 №3**

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

студент группы ИВТ-11 Гранчак Е.В.

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

доцент, к. т. н. Артамонов Ю. Н.

Котельники – 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc41059087)

[1. Разработка численных алгоритмов. 4](#_Toc41059088)

[1.1 Суммирование рядов и вычисление элементарных функций. 4](#_Toc41059089)

[1.2 Приближённые методы нахождения корней уравнения 10](#_Toc41059090)

[2. Разработка игровой программы 26](#_Toc41059091)

[2.1. Игровая программа «Календарь-ежедневник» 26](#_Toc41059092)

[ЗАКЛЮЧЕНИЕ 32](#_Toc41059093)

[Список использованных источников 33](#_Toc41059094)

# **ВВЕДЕНИЕ**

В XX веке стремительно начали развиваться информационные технологий, наш мир стал более мобильным и быстрым за счёт новых изобретений. Сейчас, невозможно представить, как современные люди обходились бы без технологий, сильно облегчающих жизнь. Например, бухгалтерам было бы сложно вести учёт и составлять отсчёты, постоянно всё пересчитывая, но с появлением компьютеров – их работа стала немного проще. Тоже самое можно сказать о различных предприятиях, где существуют запрограммированные станки, выполняющие отведённую им работу. Существует множество языков программирования, такие как: С++, Си, С#, Python, PHP, LISP и т.д. В данной курсовой работе рассмотрим программы, написанные на языке Си, ведь он довольно известный и популярный язык программирования. Поговорим о нём немного подробнее.

Язык Си – это универсальный язык программирования, который первоначально появился в системе UNIX, и развивался как основной язык систем, хоть его и называют языком системного программирования, потому что он удобен для написания любых вычислительных программ. Этот язык в основном работает с символами, числами и адресами. Данный язык программирования был разработан американцем по имени Деннис Ритчи в 1972 году.

Перейдём непосредственно к целям и задачам курсовой работы по дисциплине «Программирование на языке высокого уровня». Её целью является разработка численных алгоритмов, включающая в себя знания, полученные из таких дисциплин как «Линейная алгебра», «Математический анализ», и игровой программы, где необходимо проявить творческий подход к реализации программы.

**1. Разработка численных алгоритмов.**

## 1.1 Суммирование рядов и вычисление элементарных функций.

В варианте №3 необходимо проверить ряд на сходимость.

Дано представление в виде цепной дроби:

Формула 1.1.1 – Цепная дробь tan (x)

Вычислить tan(x) и проверить сколько членов дроби надо взять для получения результата с заданной точностью при различных значениях х.

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 для вычисления цепной дроби

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);

printf("\n\n");

// вызов функции seriesTg

seriesTg (x,epsilon,&result);

// вывод на экран значения epsilon, tan(x), result для сравнения при заданной точности

printf ("\n" //перенос каретки на следующую строку

" Точность вычислений: %15.10lf\n"

" Левая часть: %15.10lf\n"

" Правая часть: %15.10lf\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 - счётчик итераций

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 больше или равна заданной точности

while (fabs (left-right) >= epsilon )

{

// если условие выполняется, то правой части присваивается значение функции compute (n, x)

right=compute(n, x);

// постинкремент счётчика итераций

n++;

/\* вывод на экран значений для проверки количества итераций,

необходимых для заданной точности и вывод ряда в зависимости от х\*/

printf (" %d) х=%10.6lf\n", n-1, right);

}

//перенос каретки на следующую строку

printf("\n\n");

// 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);

// преинкремент

--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 итерации.

Приведём блок-схему, соответствующую данной программе.

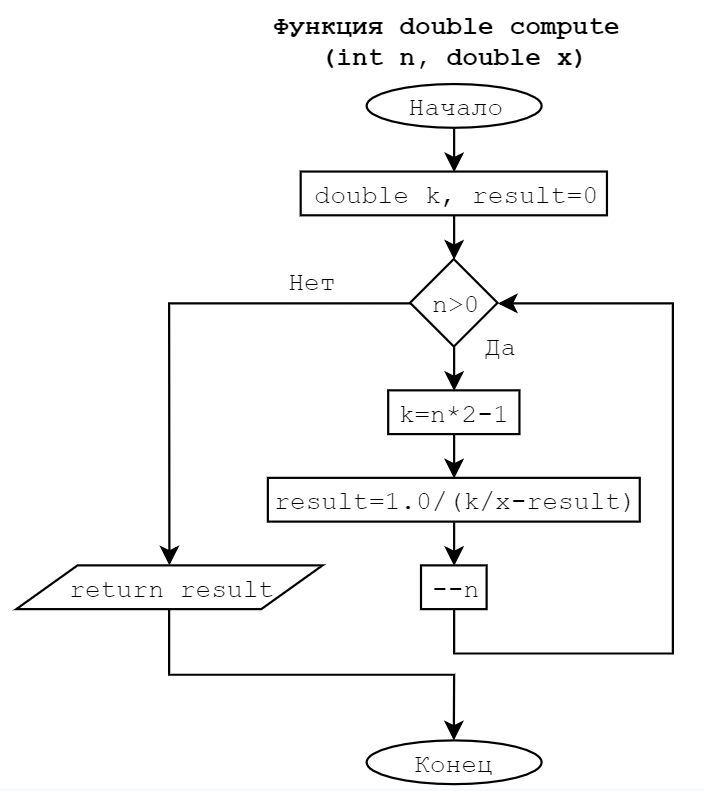
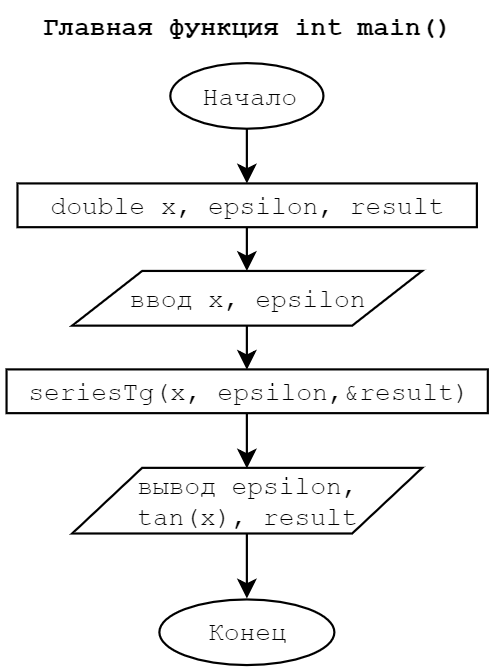


Рис 1.1.8 – Функция int main() Рис 1.1.9 – Функция double compute()

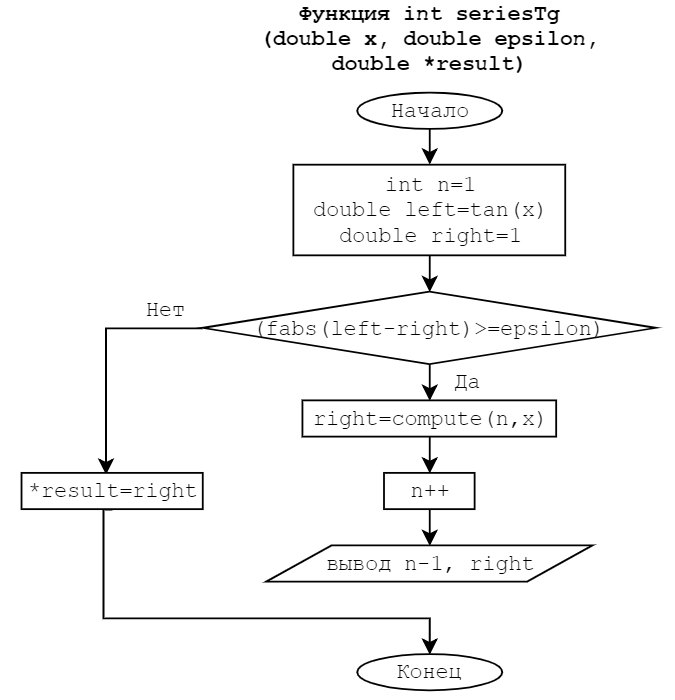


Рис 1.1.10 – Функция int seriesTg()

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 семи итерациях.

Таким образом, можно сделать вывод о том, что при заданных значениях x и epsilon, цепная дробь действительно сходится, и это доказывают результаты испытаний, приведённые в таблицах 1.1.3 – 1.1.7 и 1.1.11 – 1.1.15.

**1.2 Приближённые методы нахождения корней уравнения**

В варианте №3 необходимо проанализировать такие методы, как: метод касательных и метод деления отрезка пополам. А также сравнить число итераций при одном и том же значении точности вычисления. Для сравнения необходимо использовать следующие уравнения:

Так как данные уравнения мы будем решать двумя методами, приведём сначала их решение без параметров.

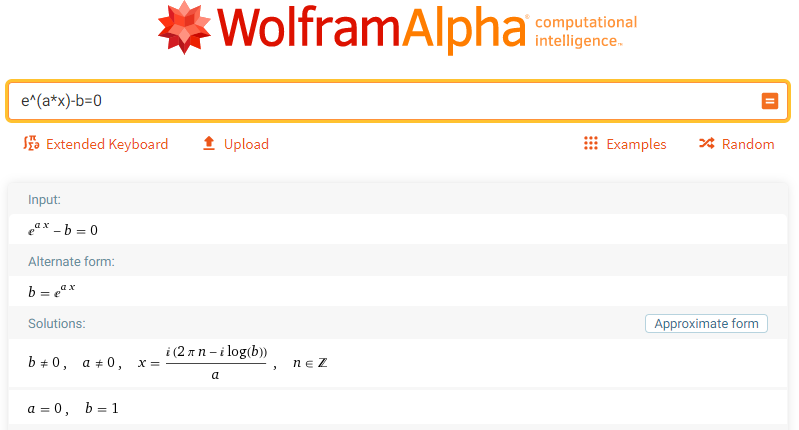


Рис. 1.2.1 – Решение уравнения вида без параметров на WolframAlpha

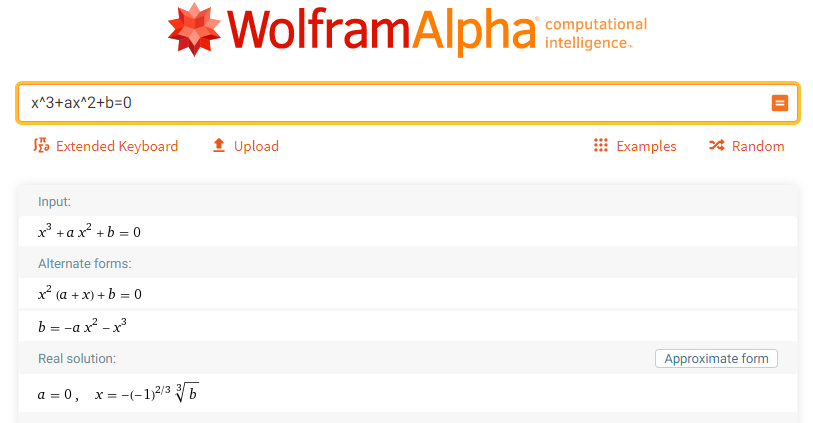


Рис. 1.2.2 – Решение уравнения вида без параметров на WolframAlpha

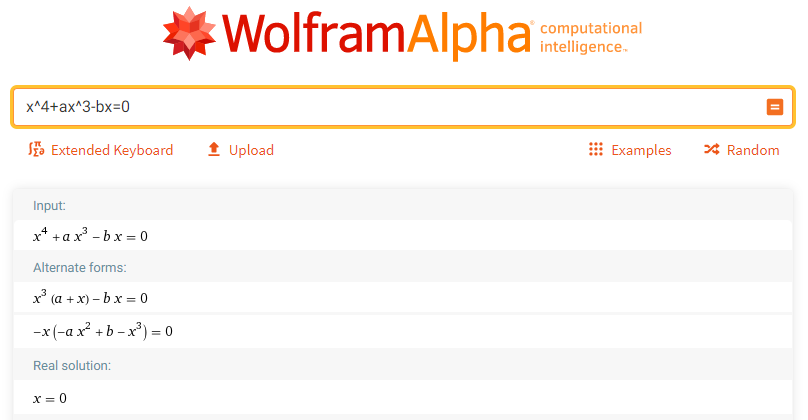


Рис. 1.2.3 – Решение уравнения вида без параметров на WolframAlpha

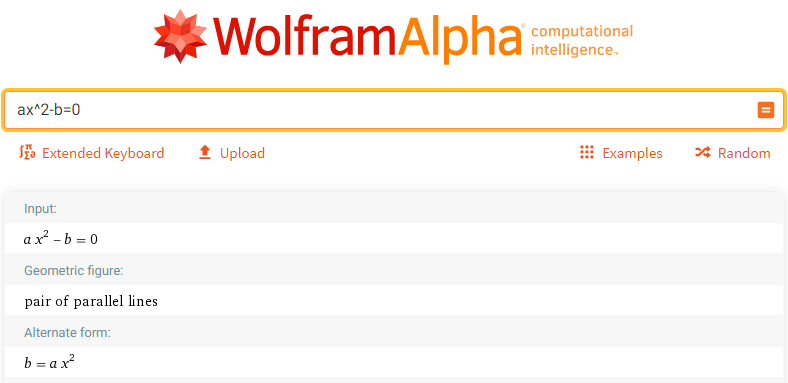


Рис. 1.2.4 – Решение уравнения вида без параметров на WolframAlpha

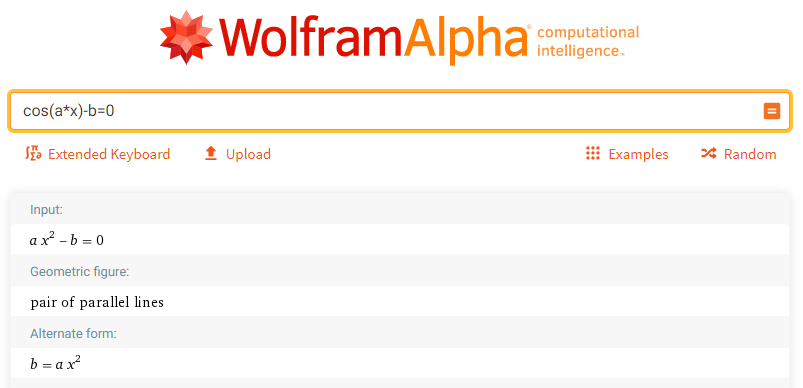


Рис. 1.2.5 – Решение уравнения вида без параметров на WolframAlpha

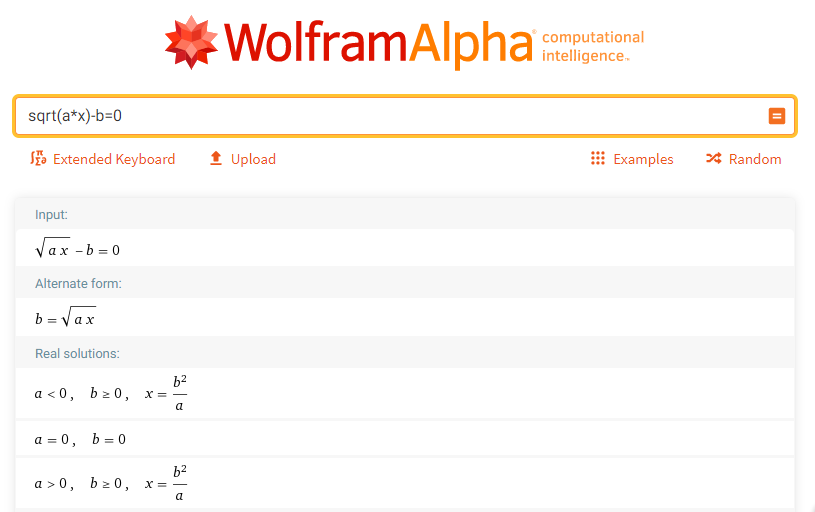


Рис 1.2.6 – Решение уравнения вида без параметров на WolframAlpha

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 абсцисс в точке хn+1, тогда мы уже отталкиваемся от точки xn+1 и проводим касательную в той точке к заданной функции.

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

#define e 2.71828

// 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=1, epsilon, c, d;

printf ("введите точность: ");

// ввод значения epsilon

scanf ("%lf", &epsilon);

// ввод значения c

printf ("Введите с: ");

scanf("%lf",&c);

// ввод значения d

printf ("Введите d: ");

scanf("%lf",&d);

printf ("\n\n");

// вывод значения, которое возвращает функция root()

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 pow(x,4)+c\*pow(x,3)-d\*x; //5

//return pow(x,3)+c\*pow(x,2)+d; //4

//return pow(e, c \* x) - d; //2

// дополнительные уравнения

//return c\*pow(x,2)-d;

//return cos(c\*x)-d;

//return sqrt(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\n",k,x,f(x,c,d));

}

// пока абсолютное значение функции от аргумента х >= epsilon

while(fabs(f(x,c,d))>=epsilon);

}while(f(x,c,d)\*f(x1,c,d)<0);

printf ("\n\n");

printf("\n Ответ х = %lf\n"

"При итерации = %d\n",x,k );

// 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;

}

Листинг 1.2.7 – Листинг программного кода метода касательных

Теперь представим программу в виде блок-схемы, разделённой по функциям.

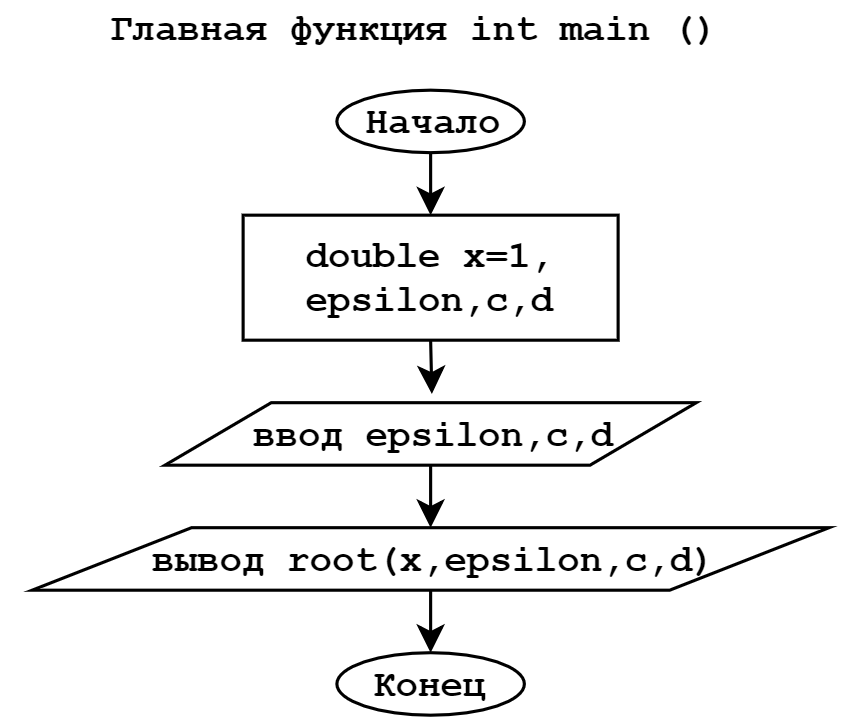
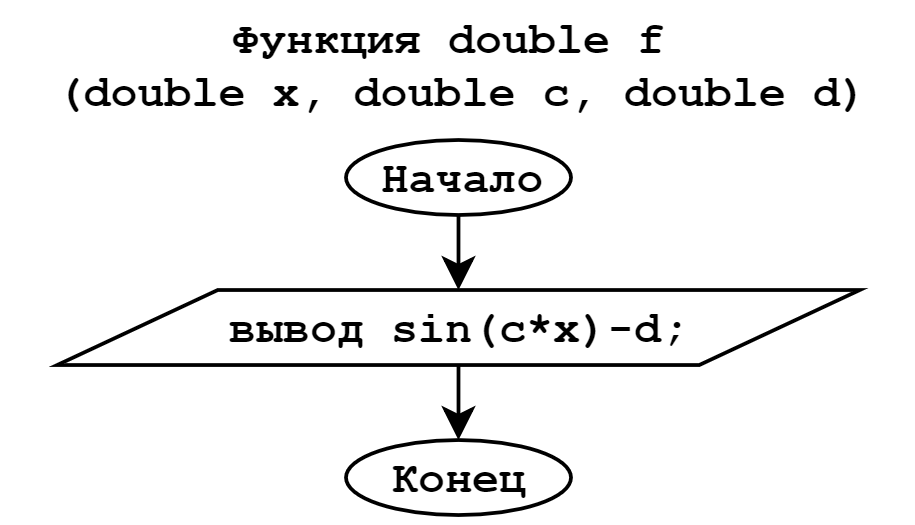
 

Рис. 1.2.8 – Функция int main() Рис. 1.2.9 – Функция double f()

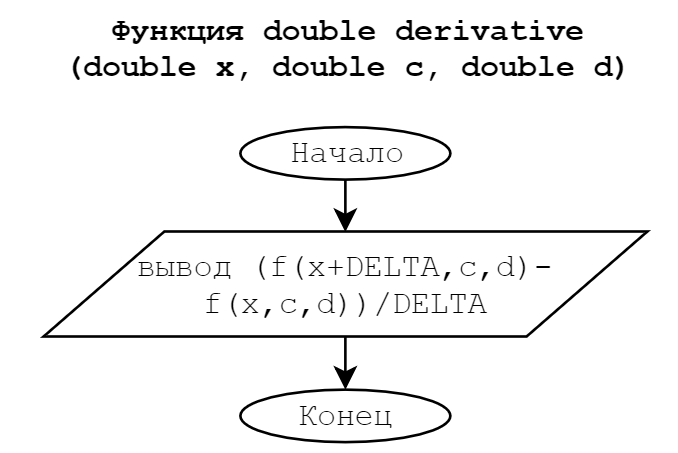


Рис. 1.2.10 – Функция double derivative()

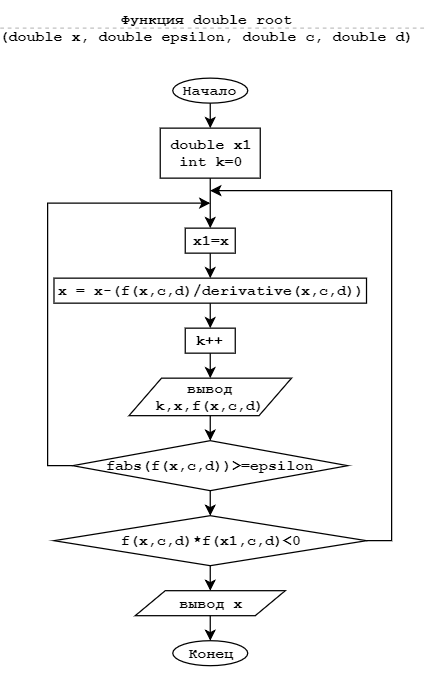


Рис. 1.2.11 – Функция double root()

Чтобы убедиться на практике, что программа работает корректно, целесообразно привести таблицу значений входных и выходных данных.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **с** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 2 | 0.5 | 0.00001 | 7 | -0,346574 |
| 1 | 0.2 | 6 | -1,609433 |

Таблица 1.2.12 – Уравнение № 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **с** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 2 | 3 | 0.00001 | 12 | -2,485584 |
| 4 | 2 | 21 | -4,117942 |

Таблица 1.2.13 – Уравнение № 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **с** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 1 | 1 | 0.00001 | 5 | 0,754878 |
| 2 | 9 | 10 | 1,584544 |

Таблица 1.2.14 – Уравнение № 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **с** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 2 | 3 | 0.00001 | 3 | 1,224746 |
| 0,5 | 6 | 5 | 3,464104 |

Таблица 1.2.15 – Уравнение № 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **с** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 2 | 0.4 | 0.00001 | 3 | 0,579641 |
| 1 | 0.2 | 3 | 1,369439 |

Таблица 1.2.16 – Уравнение № 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **с** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 2 | 3 | 0.00001 | 7 | 4.500000 |
| 8 | 5 | 7 | 3,125000 |

Таблица 1.2.17 – Уравнение №6

Метод деления отрезка пополам: . Для решения этой задачи, необходимо получить концы отрезка [a, b] от пользователя. В этом отрезке будет происходить поиск корня функции. Предположим, что на этом отрезке в точке x0 график функции f(x) пересекает ось абсцисс. Тогда нужно сдвигать левую и правую границу отрезка [a, b] в точку в зависимости от знаков функции f(x).

Для сравнения необходимо использовать данные уравнения:

Ниже представлен листинг данного метода с пояснением в виде комментариев.

Значения переменных a, b и e вводятся с клавиатуры пользователем. Установим, что точность вычислений будет равняться 0,00001.

//Подключение нужных библиотек

//стандартный заголовочный файл

#include <stdio.h>

//математическая библиотека

#include <math.h>

//точность вычислений

#define epsilon 0.00001

//значение е

#define e 2.71828

//прототип функции f

double f(double, double, double);

//прототип функции нахождения корня

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

//объявление функции f

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

{

return pow(e, c \* x) - d; ///1///

///2///return pow(x, 3) + c \* pow(x, 2) + d;

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

///4///return c \* pow(x, 2) - d;

///5///return cos(c \* x) - d;

///6///return sqrt(c \* x) - d;

}

// объявление функции нахождения корня

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

// объявляем переменные m, i, lenght

double m;

int i = 0;

double lenght = b - a;

//если значение функции меньше точности, возвращаем а

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

return a;

//если значение функции меньше точности, возвращаем b

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

return b;

//пока длина больше или равна точности, выполняем:

while (lenght >= epsilon)

{

//находим середину отрезка

m = (a + b) / 2;

//если значение функции меньше точности, возвращаем m

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

return m;

//если произведение функции меньше 0, то присваиваем b значение m

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

b = m;

//иначе присваиваем a значение m

else

a = m;

//уменьшаем длину

lenght /= 2;

//увеличиваем число итераций

i++;

}

//вывод на печать количества итераций

printf("Число итераций: %d\n", i);

//возвращаем m

return m;

}

//главная функция

int main()

{ //объявляем переменные и вводим значения a,b,c,d

double a, b, c, d, x;

printf("Введите а и b: ");

//ввод интервала

scanf("%lf%lf", &a, &b);

printf("Введите c и d: ");

//ввод параметров

scanf("%lf%lf", &c, &d);

//если точки нет на интервале, сообщаем об ошибке

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

printf("Функция не имеет коней на данном отрезке\n");

//находим корень пр ипомощи функции FinfRoot

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

//выводим полученное значение х на экран

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

//завершение работы

return 0;

}

Листинг 1.2.18 – Листинг программного кода метода деления отрезка пополам

Теперь представим программу в виде блок-схемы, разделённой по функциям.

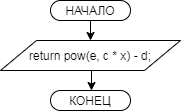


Рис. 1.2.19 ­– Функция double f()

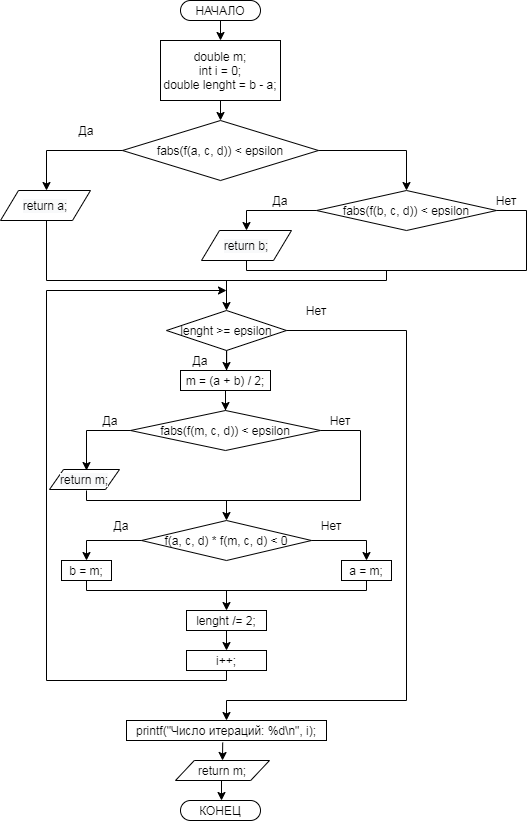


Рис. 1.2.20 – Функция double FindRoot()

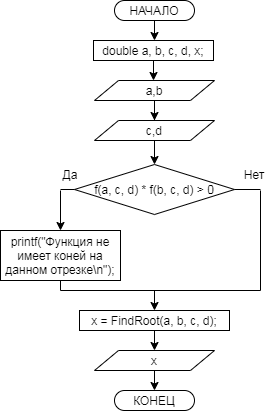


Рис. 1.2.21 – Главная функция int main()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| -.05 | 0 | 2 | 0.5 | 0.00001 | 14 | -0,346573 |
| -2 | -1 | 1 | 0.2 | 13 | -1,609436 |

Таблица 1.2.22 – Уравнение №1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| -2 | -1 | 0.2 | 3 | 0.00001 | 16 | -1.512093 |
| -2 | -1.75 | 0.1 | 7 | 15 | -1.946846 |

Таблица 1.2.23 – Уравнение №2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| -3 | -2 | 3 | 1 | 0.00001 | 17 | -2,879387 |
| 1,5 | 2 | 2 | 9 | 16 | 1,584541 |

Таблица 1.2.24 – Уравнение №3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 1 | 2 | 2 | 3 | 0.00001 | 15 | 1,224747 |
| -4 | -2 | 0,5 | 6 | 17 | -3,464104 |

Таблица 1.2.25 – Уравнение №4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| -1 | -0,5 | 2 | 0,4 | 0.00001 | 14 | -0,579636 |
| -2 | -1 | 1 | 0,2 | 13 | -1,369446 |

Таблица 1.2.26 – Уравнение №5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **epsilon** | **Кол-во итераций** | **Корень х** |
| 0 | 7 | 2 | 3 | 0.00001 | 16 | 4,500023 |
| 0 | 10 | 8 | 5 | 13 | 3,125000 |

Таблица 1.2.27 – Уравнение №6

В данном разделе произвели сравнение результатов двух методов, а именно метод касательных и метод деления отрезка пополам. Также верифицировали данные с WolframAlpha, для подтверждения результатов работы программ. 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. При сравнении методов, убедились в том, что метод касательных более точно находит корень уравнения, чем метод деления отрезка пополам.

# **2. Разработка игровой программы**

## 2.1. Игровая программа «Календарь-ежедневник»

Чтобы написать игровую программу, нужно в первую очередь проанализировать условия. Ведь именно они дают возможность модернизировать программу и делают её уникальной. На заданную дату в заданное время реализовать возможность запланировать событие, за которое в программе будет отвечать запись в файл. Реализовать функции создания, редактирования и удаления событий.

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<time.h>

#include<conio.h>

#include<windows.h>

int MenuNote = 0, i = 0, j = 0, StartNote = 0, EndMenuNote = 4;

void menu()

{

int i = 0, flag = 1, StartNote = 0, MenuNote = 0;

while (flag == 1)

{

logo();

OutputMenu(MenuNote);

JoyMenu(&flag);

}

if (MenuNote == 0) new\_note();

if (MenuNote == 4) exit(1);

if (MenuNote == 3)

{

printf("\n");

printf("Нажмите <backspace>, чтобы вернуться обратно в меню\n");

printf("\n");

show\_note();

}

}

void new\_note()

{

FILE\* date = fopen("datebase.txt", "a+");

char note[100], datenote[100] = "| Note: ";

printf("Note: ");

scanf("%s", note);

MenuNote = 2;

int flag = 1;

int flag\_num;

int day = 0;

int month = 0;

int year = 1970;

system("cls");

printf("| Note: %s |\n| Date: [%d:00:00]|", note, day);

while (flag == 1)

{

flag\_num = MenuNote;

JoyMenu(&flag);

if (flag\_num < MenuNote)

{

day--;

if (day < 0) day = 30;

}

else if (flag\_num > MenuNote)

{

day++;

if (day > 30) day = 0;

}

MenuNote = 2;

printf("| Note: %s |\n| Date: [%d:00:00] |", note, day);

}

flag = 1;

system("cls");

printf("| Note: %s |\n| Date: [%d:00:00] |", note, day);

MenuNote = 2;

while (flag == 1)

{

flag\_num = MenuNote;

JoyMenu(&flag);

if (flag\_num < MenuNote)

{

month--;

if (month < 0) month = 12;

}

else if (flag\_num > MenuNote)

{

month++;

if (month > 12) month = 0;

}

MenuNote = 2;

printf("| Note: %s |\n| Date: [%d:00:00] |", note, day, month);

}

flag = 1;

system("cls");

printf("| Note: %s|\n| Date: [%d:%d:00] |", note, day, month);

MenuNote = 2;

while (flag == 1)

{

flag\_num = MenuNote;

JoyMenu(&flag);

if (flag\_num < MenuNote)

{

year--;

if (year < 1970) year = 1970;

}

else if (flag\_num > MenuNote) year++;

MenuNote = 2;

printf("| Note: %s |\n| date: [%d:%d:%d] |", note, day, month, year);

}

char\* s\_day, \* s\_month, \* s\_year;

sprintf(s\_day, "%d", day);

int k = 0;

for (; datenote[k] != '\0'; k++)

for (int i = 0; note[i] != '\0'; i++, k++) datenote[k] = note[i];

printf("%s\n", datenote);

if (day > 10) datenote[k] = \*s\_day;

else if (day < 10) datenote[k] != \*s\_day;

}

void show\_note()

{

FILE\* date = fopen("datebase.txt", "r+");

int flag = 1;

char mass[100];

while (fgets(mass, 100, date))

{

if (mass[strlen(mass) - 1] == '\n') mass[strlen(mass) - 1] = '\0';

printf("%s\n", mass);

}

if (getch() == 8) { fclose(date); menu(); }

else if (getch() != 0) show\_note();

}

void logo()

{

system("cls");

printf("\t\t\t Ежедневник \n");

printf("\t\tМеню: \n");

}

void OutputMenu(int kursor)

{ if (kursor == 0)

{ printf("\t\t[\*] - New Note\n\t\t - Edit Note\n\t\t - Delete Note\n\t\t - Show Note's\n\t\t - Shutdown\n"); }

if (kursor == 1)

{ printf("\t\t - New Note\n\t\t[\*] - Edit Note\n\t\t - Delete Note\n\t\t - Show Note's\n\t\t - Shutdown\n"); }

if (kursor == 2)

{ printf("\t\t[\*] - New Note\n\t\t - Edit Note\n\t\t[\*] - Delete Note\n\t\t - Show Note's\n\t\t - Shutdown\n"); }

if (kursor == 3)

{ printf("\t\t[\*] - New Note\n\t\t - Edit Note\n\t\t - Delete Note\n\t\t[\*] - Show Note's\n\t\t - Shutdown\n"); }

if (kursor == 4)

{ printf("\t\t[\*] - New Note\n\t\t - Edit Note\n\t\t - Delete Note\n\t\t - Show Note's\n\t\t[\*] - Shutdown\n"); }

}

int JoyMenu(int\* flag)

{

char select = getch();

switch (select)

{

case 13:

{

system("cls");

\*flag = 0;

return MenuNote;

}

case 56:

{

if (MenuNote == StartNote) {

MenuNote = EndMenuNote;

//OutputMenu{i);

system("cls");

return MenuNote;

}

if (MenuNote > StartNote) {

MenuNote--;

//OutputMenu(i);

system("cls");

return MenuNote;

}

}

case 50: {

if (MenuNote == EndMenuNote)

{

MenuNote = StartNote;

//OutputMenu(i);

system("cls");

return MenuNote;

}

else {

MenuNote++;

//OutputMenu(i);

system("cls");

return MenuNote;

}

}

default: system("cls");

}

}

int main() { menu(); return 0;}

Листинг 2.1.1 – Листинг программного кода игры «Календарь-ежедневник»

# **ЗАКЛЮЧЕНИЕ**

Результатом данной курсовой работы является:

* Разработка численных программ, а именно вычисление цепной дроби и приближённые методы нахождения корней уравнения;
* Построение блок-схем важных функций программ;
* Реализация расчётных таблиц;
* Разработка игровой программы.

Общий вывод заключается в том, что программирование каких-либо математических функций и формул – значительно упрощает процесс решения поставленных задач.

# **Список использованных источников**

[1]. «Основы программирования на языках Си и C++ для начинающих». 2017г.

[2]. Керниган, Ритчи «Язык программирования С» второе издание 2016г.

[3]. Подбельский, Фомин: Курс программирования на языке Си. 2018г.

[4]. Артамонов. Ю. Н. Методические указания по выполнению курсовой работы по дисциплине «Программирование на языке высокого уровня» 2018г.

[5]. Брайн У. Язык программирования С. 2016г

[6]. Волков Е. А. Численные методы. — М. : Физматлит, 2018г.

[7]. Электронный ресурс WolframAlpha URL: [https://www.wolframalpha.com](https://vk.com/away.php?to=https%3A%2F%2Fwww.wolframalpha.com&cc_key=)

[8]. Электронный ресурс, ссылка на web-страницу Метод Ньютона: URL: [https://ru.wikipedia.org/wiki/Метод\_Ньютона](https://vk.com/away.php?to=https%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%25D0%259C%25D0%25B5%25D1%2582%25D0%25BE%25D0%25B4_%25D0%259D%25D1%258C%25D1%258E%25D1%2582%25D0%25BE%25D0%25BD%25D0%25B0&cc_key=)

[9]. Юркин А. Задачник по программированию на языке Си. 2016г.

[10]. Электронный ресурс, ссылка на web-страницу: https://www.desmos.com/calculator