

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 №10**

Выполнил: \_\_\_\_\_  
студент группы ИВТ-11 Куртюшкин Д.С.  
Проверил: \_\_\_\_\_  
доцент, к.т.н. Артамонов Ю.Н.

Котельники – 2019



## Введение

В связи с развитием информационных и телекоммуникационных технологий, начиная с XX века, мы смогли застать различные изменения в сферах жизни людей, а также в отрасли хозяйствования. В результате глобальных изменений в экономике и взятие курса на инновационное развитие, предприятия и организации во всем мире стали использовать различные средства ЭВМ(электронно-вычислительных машин), а позже и программного обеспечения. Но не смотря на эффективность электронных средств, они требовали определенного опыта и знаний, требуемых от человека. Возникла необходимость учета аппаратной организации каждой конкретной машины, то есть необходимость перекодировки программ при переносе с одной машины на другую – зачастую наблюдалась непереносимость алгоритмов, разработанных для одних машин при переносе на другие. Практически не представлялось возможным понять принципы построения чужой программы. Написанные на машинных кодах программы содержали минимум избыточной информации, которая бы позволяла обнаружить формальные ошибки кодирования. В результате, технические ошибки при набивке программы могли приводить к обескураживающим последствиям, а обнаружить такие ошибки было очень сложно. Так, например, для управления процессами ЭВМ в 1949 году был создан язык Ассемблер, который был более понятным, чем машинный язык. Спустя несколько десятилетий, люди открыли для себя множество других языков программирования «высокого уровня», таких как: ФОРТРАН (FORmula TRANslator – переводчик формул), АЛГОЛ (ALGOritmic Language – алгоритмический язык), БЕЙСИК (BASIC – Beginner's Allpurpose Symbolic Instruciones Code – дословно: «многоцелевой код символических инструкций для начинающих»). В 1970 году Никлаус Вирт создал язык PASCAL (Паскаль). Этот язык обладает весьма развитыми средствами, особенно те его версии, которые используются в настоящее время. В настоящее время используется еще несколько мощных языков программирования, одним из которых является язык C/C++, созданный Бьёрном Страуструпом. Он предоставил возможность людям создавать программное обеспечение для решения не только организационных и экономических задач, но и повседневных. Основным отличием языка C(Си) от C++ в том, что у второго присутствует поддержка ООП(Объектно-ориентированного программирования).

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.

При выполнении работы следует понять, что представляет из себя вычисление с помощью членов ряда. **Числовой ряд** – это сумма членов числовой последовательности вида:

$$\sum_{k=1}^{\infty} a_k = a_1 + a_2 + \dots + a_n + \dots$$

Формула 1.1.2 – Сумма членов числовой последовательности;

где  $a_k$  называют **общим членом числового ряда** или  $k$ -ым членом ряда.

**Частичная сумма** числового ряда – это сумма вида формула, где  $n$  – некоторое натуральное число.

**Числовой ряд**  $\sum_{k=1}^{\infty} a_k$  (формула 1.1.2) называется **сходящимся**, если существует конечный предел последовательности частичных сумм:

$$S = \lim_{n \rightarrow +\infty} S_n$$

Формула 1.1.3 – Предел последовательности частичных сумм;

Если предел последовательности частичных сумм числового ряда не существует или бесконечен, то ряд  $\sum_{k=1}^{\infty} a_k$  (формула 1.1.2) называется **расходящимся**.

В качестве примера **расходящегося** ряда можно привести сумму геометрической прогрессии со знаменателем больше, чем единица:

$$1 + 2 + 4 + 8 + \dots + 2^{n-1} + \dots = \sum_{k=1}^{\infty} 2^{k-1}$$

Формула 1.1.4 – Примера расходящегося ряда;

$n$ -ая частичная сумма определяется выражением:

$$S_n = \frac{a_1 \cdot (1 - q^n)}{1 - q} = \frac{1 \cdot (1 - 2^n)}{1 - 2} = 2^n - 1$$

Формула 1.1.5 –  $N$ -ая частичная сумма;

а предел частичных сумм бесконечен:

$$\lim_{n \rightarrow +\infty} S_n = \lim_{n \rightarrow +\infty} (2^n - 1) = +\infty$$

Формула 1.1.6 – Предел частичных сумм;

После того, как мы разобрали вычисление с помощью числового ряда, можно сказать, что для данного представления будет истинна следующая последовательность:

$$\frac{1}{4}\left(x^2 - \frac{\pi^2}{3}\right) = \sum_{n=1}^{\infty} (-1)^n \frac{\cos(nx)}{n^2} = -\cos(x) + \frac{\cos(2x)}{2^2} - \dots$$

Формула 1.1.7 – Нахождение представления в виде числового ряда;

**Дано:** Представление, имеющее вид:

$$\frac{1}{4}\left(x^2 - \frac{\pi^2}{3}\right)$$

Формула 1.1.8 - Представление;

**Найти:** Вычислить, сколько слагаемых нужно взять, чтобы достичь заданной точности.

**Решение:**

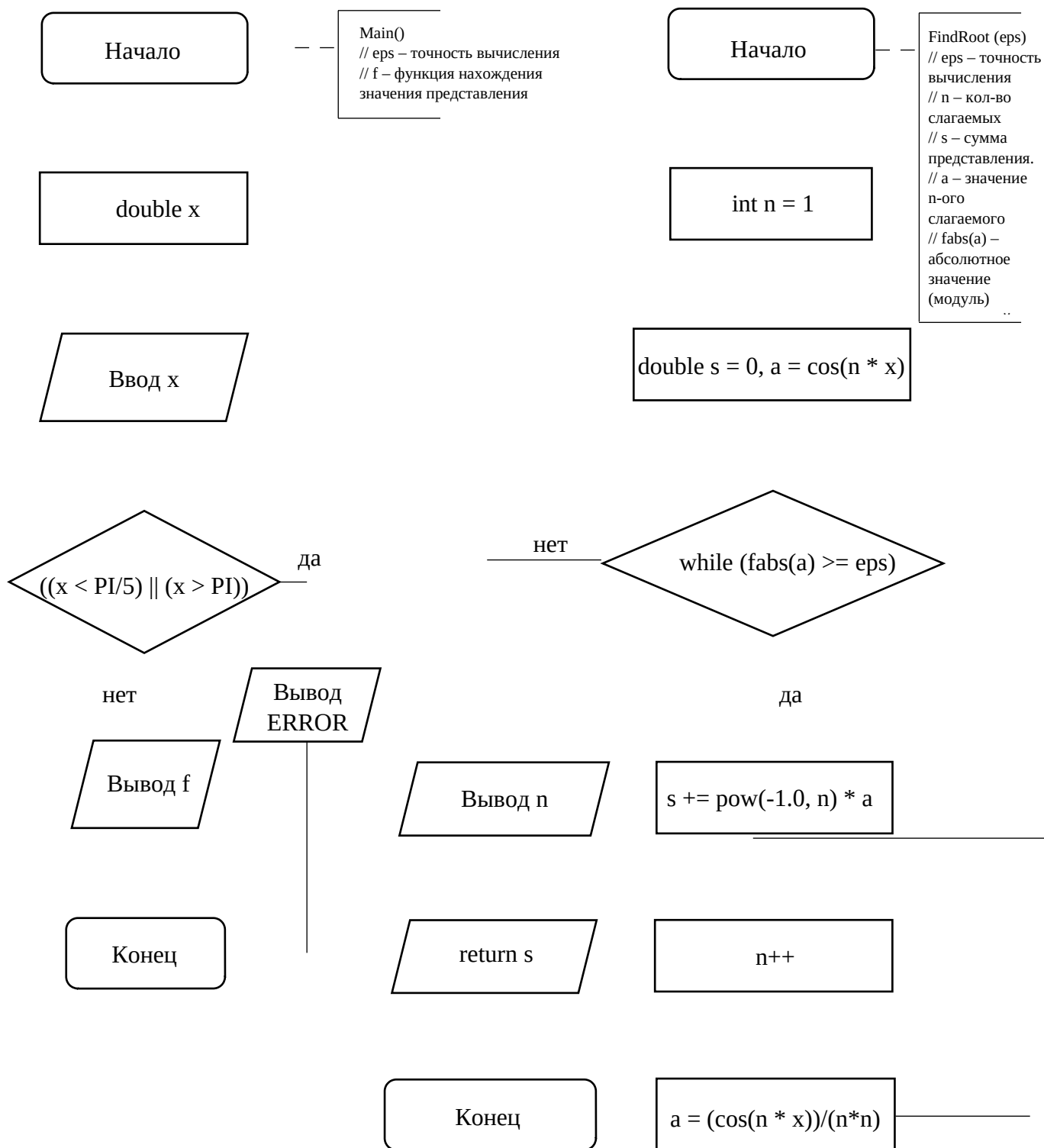
Для начала определимся с тем, какие на вход будут подаваться значения. Под «значениями» подразумевается тип данных, с которым будет работать программа. В данном случае, что и на вход, что и на выводе, мы будем получать значения переменных типа **double**, так как данный тип имеет более высокую точность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

Также нам дано ограничение по **x**, а именно:

$$\frac{\pi}{5} \leq x \leq \pi$$

Формула 1.1.9 – Ограничение по **x**;

После разбора, можно сказать, что мы можем приступить к разработке программы. На следующей странице представлена блок-схема программы для нахождения данного нам представления:



Блок-схема 1.1.10 –Блок-схема программы по нахождению представления с помощью числового ряда;

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

```

#include<stdio.h> // стандартный заголовочный файл ввода-вывода
#include<stdlib.h>
#include<math.h> // математическая библиотека
#define PI 3.141592 // константа со значением числа Пи
#define eps 1e-9 // константа со значением точности вычисления
double f(double); // прототип функции f
double f(double x) // функция нахождения числового ряда
{
    int n = 1; // n – кол-во слагаемых
    double s = 0, a = cos(n * x); // s – сумма числового ряда, a – значение слагаемого
    /* пока переменная a по модулю больше заданной точности, то выполняем цикл */
    while (fabs(a) >= eps)
    {
        s += pow(-1.0, n) * a; // суммируем новое слагаемое.
        n++; // увеличиваем кол-во слагаемых в числовом ряде
        a = (cos(n * x))/(n*n); // находим n-ое слагаемое
    }
    printf("Кол-во слагаемых: %d\n", n); // выводим кол-во слагаемых
    return s; // возвращаем значение суммы в числовом ряде
}

int main()
{
    double x; // объявляем переменную x типа double
    printf("Введите значение x: "); // выводим сообщение о том, что нужно ввести x
    scanf("%lf", &x); // вводим значение x
    /* устанавливаем ограничения для значения x. Если условие истинно, мы выводим
    ** сообщение об ошибке и заканчиваем работу программы. Иначе, продолжаем */
    if ((x < PI/5) || (x > PI)) printf("ERRORn");
    else printf("%lf\n", f(x)); // выводим значение, которое вернет функция f
    return 0;
}

```

Листинг 1.1.11 –Листинг программного кода по нахождению представления в виде числового ряда

Приведем таблицы входных и выходных данных, а также результат:

Переменная	Значение в программе, заданное по умолчанию
e	1e-9

Таблица 1.1.12 – Входные данные программы;

Переменная x	Переменная n	Переменная s
1.13	1486	-0.503242
1.02	3485	-0.562367
0.65	2970	-0.716842
0.97	468	-0.587243
2.56	948	0.815931
2.18	1890	0.365633

Таблица 1.1.13 –Выходные данные программы при разных значениях **x**(где **n** – кол-во слагаемых в числовом ряде, **s** – сумма числового ряда);

Теперь сверим хотя бы одно значение **x** через онлайн-сервис **WolframAlpha**:

WolframAlpha computational intelligence.

Input:  $(1/4) * (\text{pow}(\pi/5, 2) - (\text{pow}(\pi, 2) / 3))$

Result:  $-\frac{11\pi^2}{150}$

Decimal approximation: -0.72377098941321963204786267332425108325633795653099131260...

Рисунок 1.1.14 – Верификация тестового **x** с помощью WolframAlpha;



**В заключении,** можно сказать, что в блоке «суммирование рядов и вычисление элементарных функции» мы реализовали вычисление выражения по определенному представлению и рассмотрели подробно решение задачи.

## 1.1 Приближенные методы нахождения корней уравнения

Классическим средством изучения математических моделей и исследований на их основе свойств реальных объектов являются аналитические методы. Эти методы дают наиболее полную информацию о решении задачи, и они до настоящего времени не утратили своего значения. Однако, к сожалению, класс задач, для которого они могут использоваться, весьма ограничен. Поэтому решение, как правило, осуществляется численными методами.

Численные методы представляют собой отдельную область математики и применяются в различных прикладных направлениях. Для решения задач конкретного типа разрабатывается специальное программное обеспечение, основой алгоритмов которого и служат численные методы.

Актуальность темы заключается в том, что использование численных методов упрощает алгоритм решения задачи полагает возможным нахождение решения абсолютно всех классов экстремальных задач.

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[6,19].

### 1.2.1 Метод касательных

Существует один из методов алгебраических уравнений, называемый в мире как метод Ньютона(или метод касательных). Представим себе, что нам нужно найти корень уравнения в окрестности точки  $x=x_0$ , тогда нам нужно последовательно выполнять поиск приближений для корня уравнения, которая находится по следующей схеме:

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)}$$

Формула 1.2.1.1- Нахождение корня уравнения с помощью метода касательных

Где  $f'(x)$  – производная от функции  $f(x)$ ,  $x_n$  – приближение для корня уравнения на  $n$ -ом шаге.

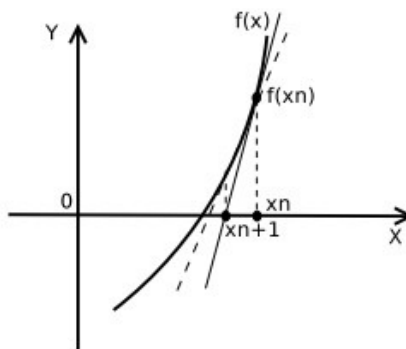


Рисунок 1.2.1.2 – Иллюстрация нахождения корня уравнения методом касательных;

**Дано:**

Уравнение:  $x^3 + c x^2 + d = 0$  (уравнение №4).

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

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

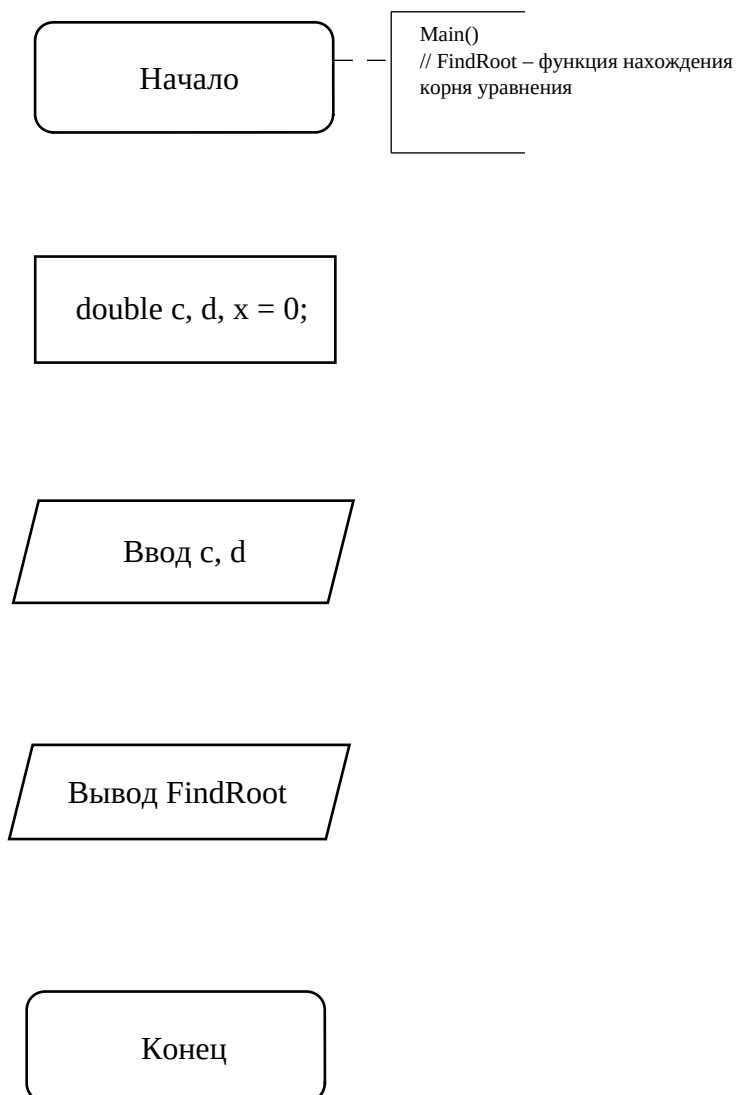
**Найти:**

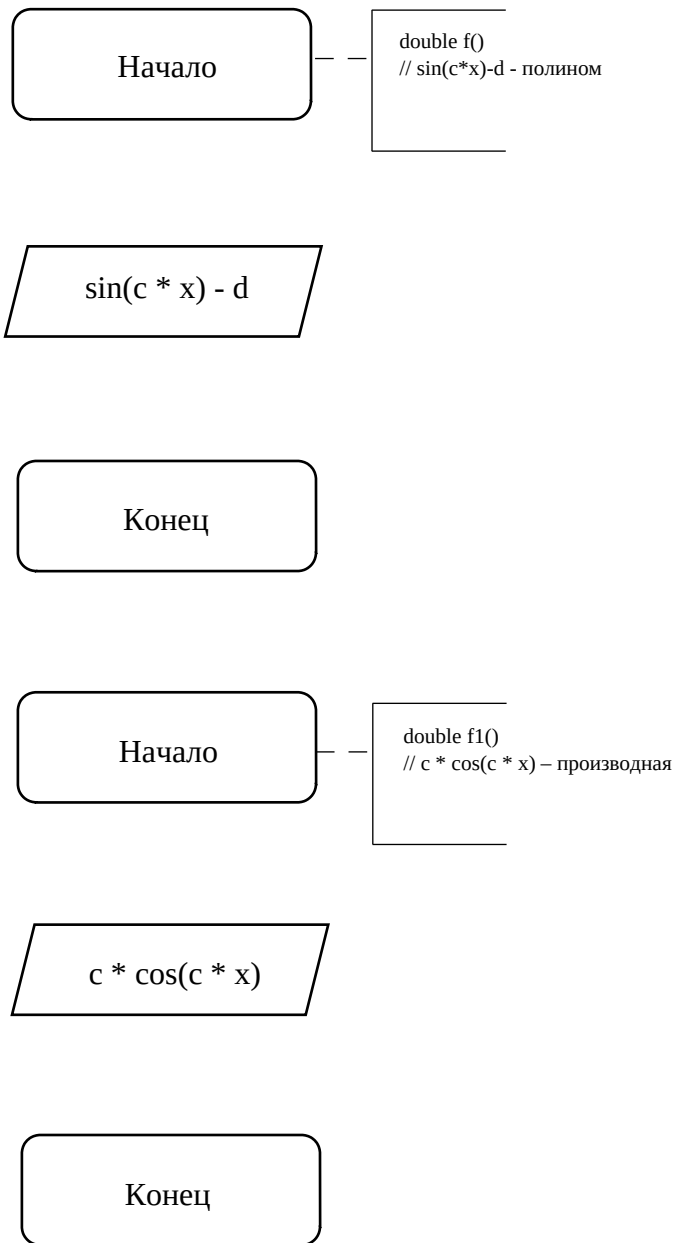
Значение **x** методом касательных.

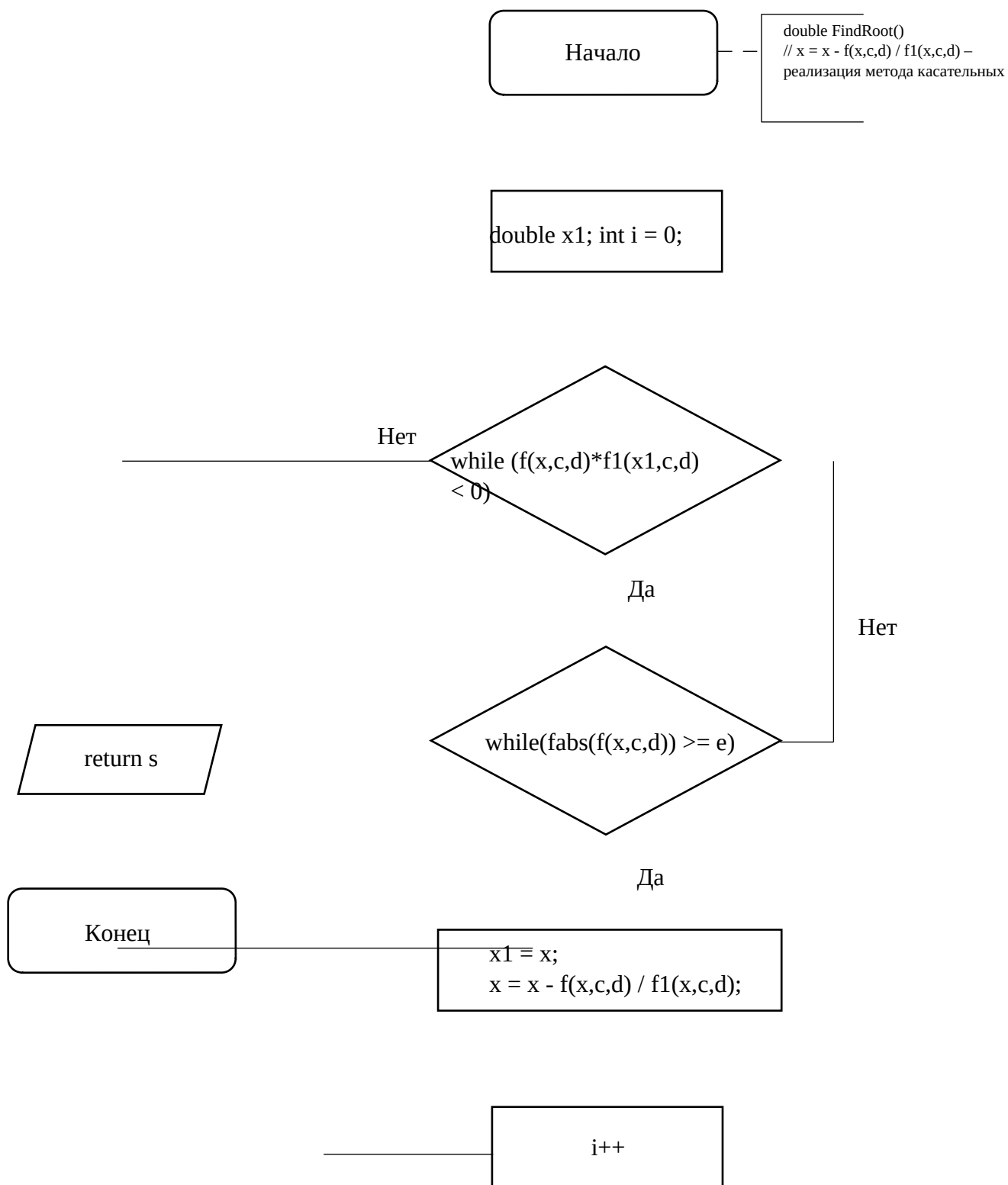
**Решение:**

Значение переменных **d** и **c** вводятся с клавиатуры пользователем. Значение точности(переменная **e**):  $10^{-13}$

Для начала рассмотрим блок-схему будущей программы.







Блок-схема 1.2.1.3 –Блок-схема программы нахождения корня методом касательных.

На следующей странице представлен листинг программного кода.

```

#include<stdio.h> // стандартный заголовочный файл ввода-вывода
#include<math.h> // математическая библиотека
#define e 1e-13 // константа точности вычисления
double f1(double, double, double); // прототип функции f1
double f(double, double, double); // прототип функции f
double FindRoot(double,double,double); // прототип функции FindRoot
double f1(double x, double c, double d) // объявляем функцию f и передаем c, d и x
{
    return 3 * pow(x, 2)+c*2*x+d; // возвращаем значение производной
}
double f(double x, double c, double d) // объявляем функцию f и передаем c, d и x
{
    return pow(x,3)+c*pow(x, 2)+d; // возвращаем значение функции
}
double FindRoot(double x, double c, double d) // объявляем функцию f передаем c, d и x
{
    double x1; int i = 0; // объявляем переменные
    do
    {
        do
        {
            x1 = x; // присваиваем переменной x1 значение x
            x = x - f(x,c,d) / f1(x,c,d); // реализуем метод касательных

            i++; // увеличиваем кол-во итераций
        } while(fabs(f(x,c,d)) >= e); // пока значение функция по модулю больше точности,
        выполняем
    } while (f(x,c,d)*f1(x1,c,d) < 0); // если при значение функции меньше 0, то выполняем
    printf("Kol-vo iteracii: %d\n", i); // выводим кол-во итераций
    return x; // возвращаем значение x
}
int main() // объявляем функцию main

```

```

{
double c, d, x = 1; // объявляем переменные c, d и x
printf("Введите значение c и d: "); // просим ввести значение c и d
scanf("%lf %lf", &c, &d); // вводим значение c и d
printf("f(x) = %lf\n", FindRoot(x, c, d)); // выводим корень уравнения
return 0;
}

```

Листинг 1.2.1.4 – Листинг программного кода нахождения корня методом касательных

Решение уравнения № 4 без параметров **c** и **d** в WolframAlpha:

WolframAlpha computational intelligence.

pow(x, 3)+c \* pow(x, 2)+d=0

Input:  
 $x^3 + c x^2 + d = 0$

Alternate forms:  
 $x^2 (c + x) + d = 0$   
 $d = -c x^2 - x^3$

Real solution:  
 $c = 0, \quad x = -(-1)^{2/3} \sqrt[3]{d}$

Solutions:  

$$x = \frac{1}{3} \left( \frac{\sqrt[3]{3 \sqrt{3} \sqrt{4 c^3 d + 27 d^2} - 2 c^3 - 27 d}}{\sqrt[3]{2}} + \frac{\sqrt[3]{2} c^2}{\sqrt[3]{3 \sqrt{3} \sqrt{4 c^3 d + 27 d^2} - 2 c^3 - 27 d}} - c \right)$$

$$x = -\frac{(1 - i \sqrt{3}) \sqrt[3]{3 \sqrt{3} \sqrt{4 c^3 d + 27 d^2} - 2 c^3 - 27 d}}{6 \sqrt[3]{2}} - \frac{(1 + i \sqrt{3}) c^2}{3 \cdot 2^{2/3} \sqrt[3]{3 \sqrt{3} \sqrt{4 c^3 d + 27 d^2} - 2 c^3 - 27 d}} - \frac{c}{3}$$

Рисунок 1.2.1.5 – Решение уравнения № 4 без параметров **c** и **d** в WolframAlpha

Теперь рассмотрим уравнение №4 с введенными пользователем значениями **c** и **d**.

$$x^3 + c x^2 + d$$

Формула 1.2.1.6 – Уравнение №4

Введем значения **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.1.7 – Входные данные для программы

Посмотрим на результат, который выдает нам WolframAlpha:

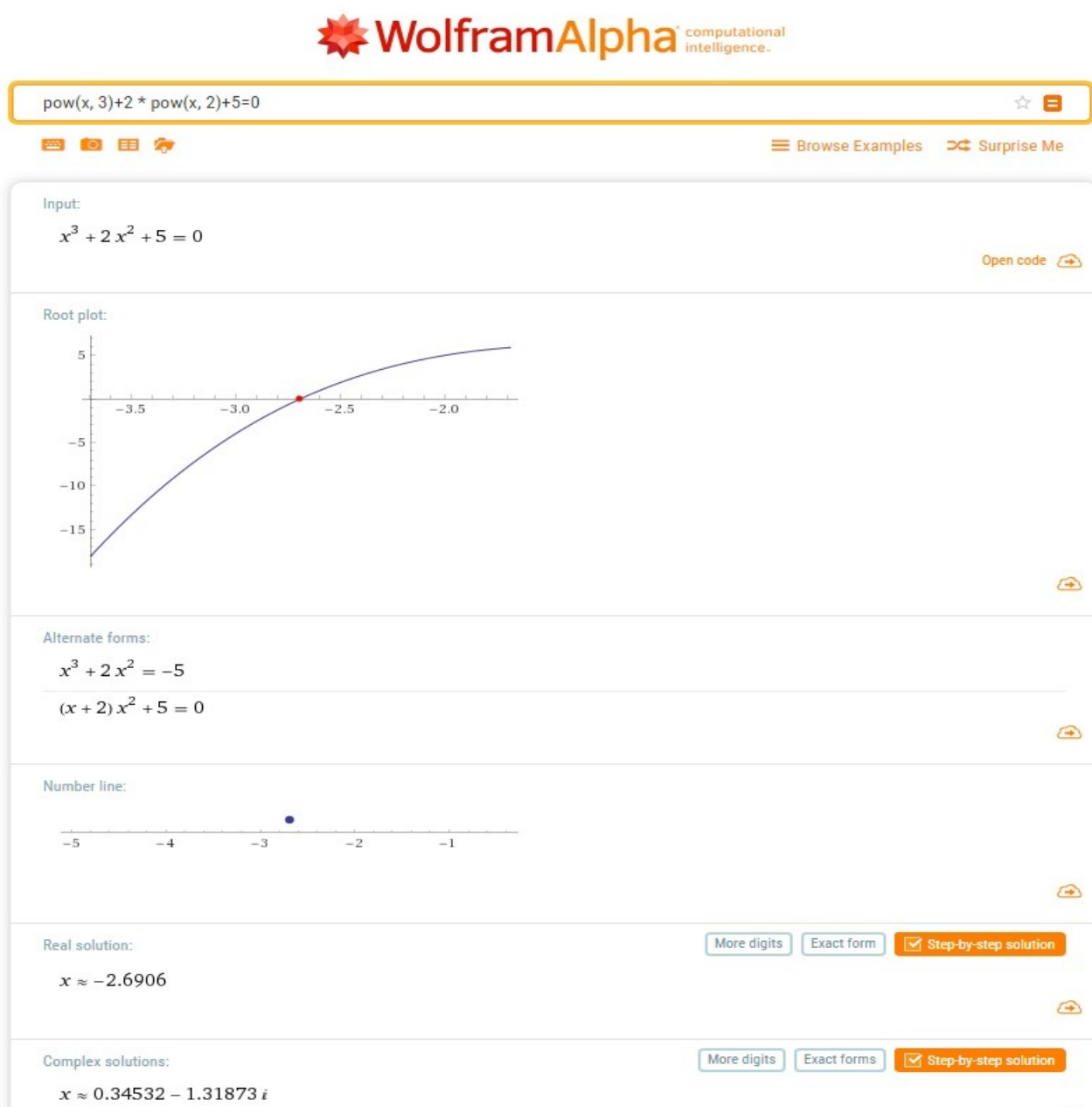


Рисунок 1.2.1.8 – Верификация значения  $x$  с помощью WolframAlpha

Теперь посмотрим, что выдает наша программа:

```
goru00@notebook:~/Documents/prog_homework/group_coursework/zadanie_10$ gcc 2_1.c -lm
goru00@notebook:~/Documents/prog_homework/group_coursework/zadanie_10$ ./a.out
Введите значение с и d: 2 5
Kol-vo iteracii: 29
f(x) = -2.690647
```

Рисунок 1.2.1.9 – Иллюстрация работы программы

Переменная <b>i</b>	Ответ(f(x))
29	-2.690647

Таблица 1.2.1.10 – Выходные данные

Рассмотрим результаты, которые выдает программа при различных значениях **c** и **d**.

Переменная <b>c</b>	Переменная <b>d</b>
1	3
5	8
7	2
1	4

Таблица 1.2.1.11 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
28	-1.863707
28	-5.286279
64	-7.040350
30	-2.000000

Таблица 1.2.1.12 – Выходные данные

Как мы видим, программа работает правильно. Теперь перейдем к следующему уравнению и выполним то же самое, что и в первом.

Рассмотрим уравнение №5:

$$x^4 + c x^3 - dx$$

Формула 1.2.1.13 – Уравнение №5

Рассмотрим уравнение без параметров **c** и **d**:

Рисунок 1.2.1.14 – Верификация значения **x** с помощью WolframAlpha

Введем значения **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.1.15 – Входные данные

WolframAlpha говорит нам, что есть корень  $x=1.2419$ :



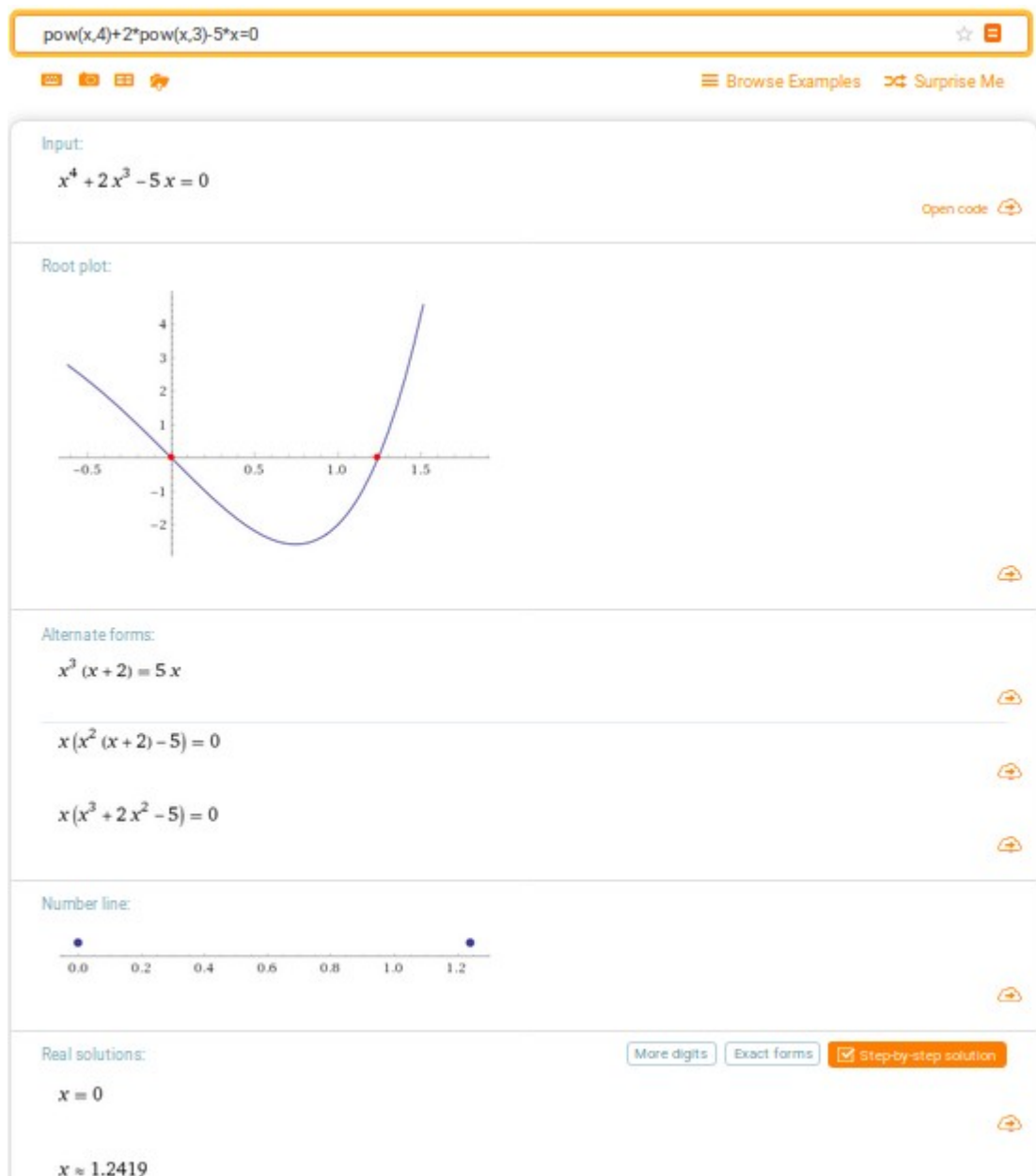


Рисунок 1.2.1.16 – Верификация данных через WolframAlpha

Проверим результаты WolframAlpha на нашей программе.

```
goru00@notebook:~/Documents/prog_homework/group_coursework/zadanie_10$ gcc 2 1.c -lm
goru00@notebook:~/Documents/prog_homework/group_coursework/zadanie_10$ ./a.out
Введите значение с и d: 2 5
Kol-vo iteracii: 16
f(x) = 1.241897
goru00@notebook:~/Documents/prog_homework/group_coursework/zadanie_10$
```

Рисунок 1.2.1.17 – Иллюстрация работы программы

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.1.18 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
16	1.241897

Таблица 1.2.1.19 – Выходные данные

Попробуем провести еще какие-либо проверки при различных значениях **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
1	3
5	8
7	2
1	4

Таблица 1.2.1.20 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
12	1.174559
13	1.141336
20	0.515853
18	1.314596

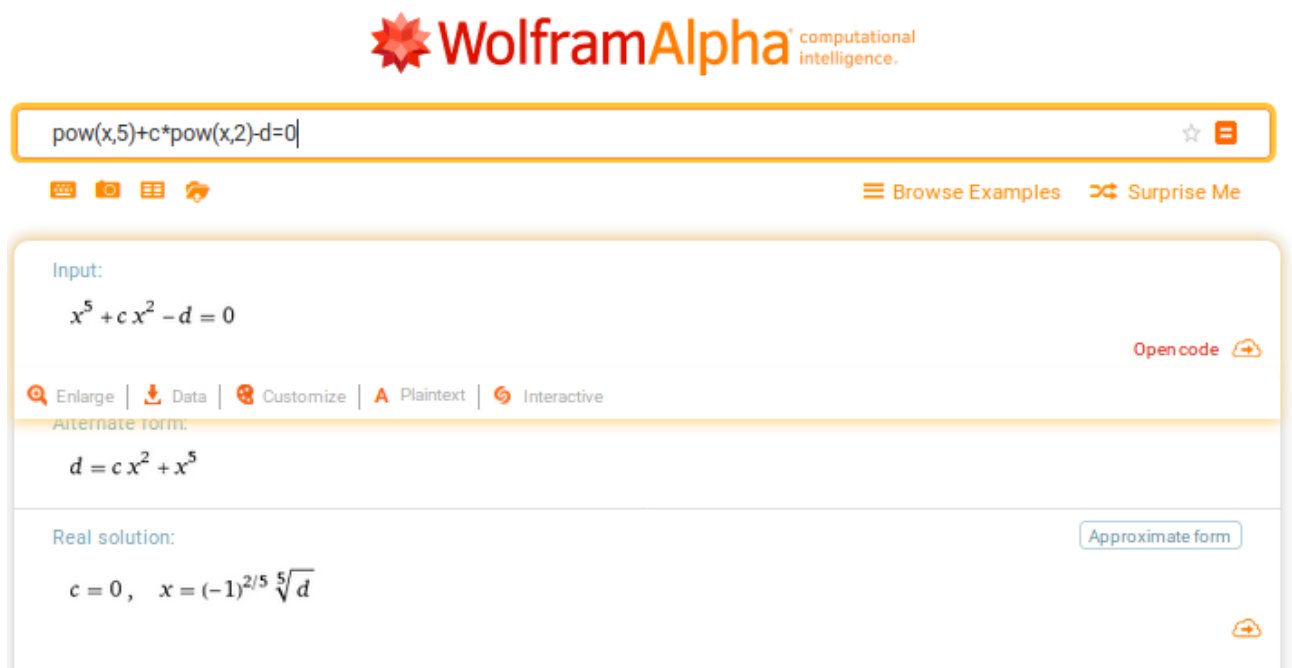
Таблица 1.2.1.21 – Выходные данные

Осталось рассмотреть последнее уравнение №6:

$$x^5 + c x^2 - d$$

Формула 1.2.1.22 – Уравнение №6

Рассмотрим уравнение без параметров **c** и **d**:



WolframAlpha<sup>®</sup> computational intelligence.

pow(x,5)+c\*pow(x,2)-d=0

Input:

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

Open code

Enlarge | Data | Customize | Plaintext | Interactive

Alternate form:

$$d = c x^2 + x^5$$

Real solution:

$$c = 0, \quad x = (-1)^{2/5} \sqrt[5]{d}$$

Approximate form

Рисунок 1.2.1.23 – Верификация данных через WolframAlpha

Введем значения **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.1.24 – Входные данные

Проверим сначала результаты через WolframAlpha:

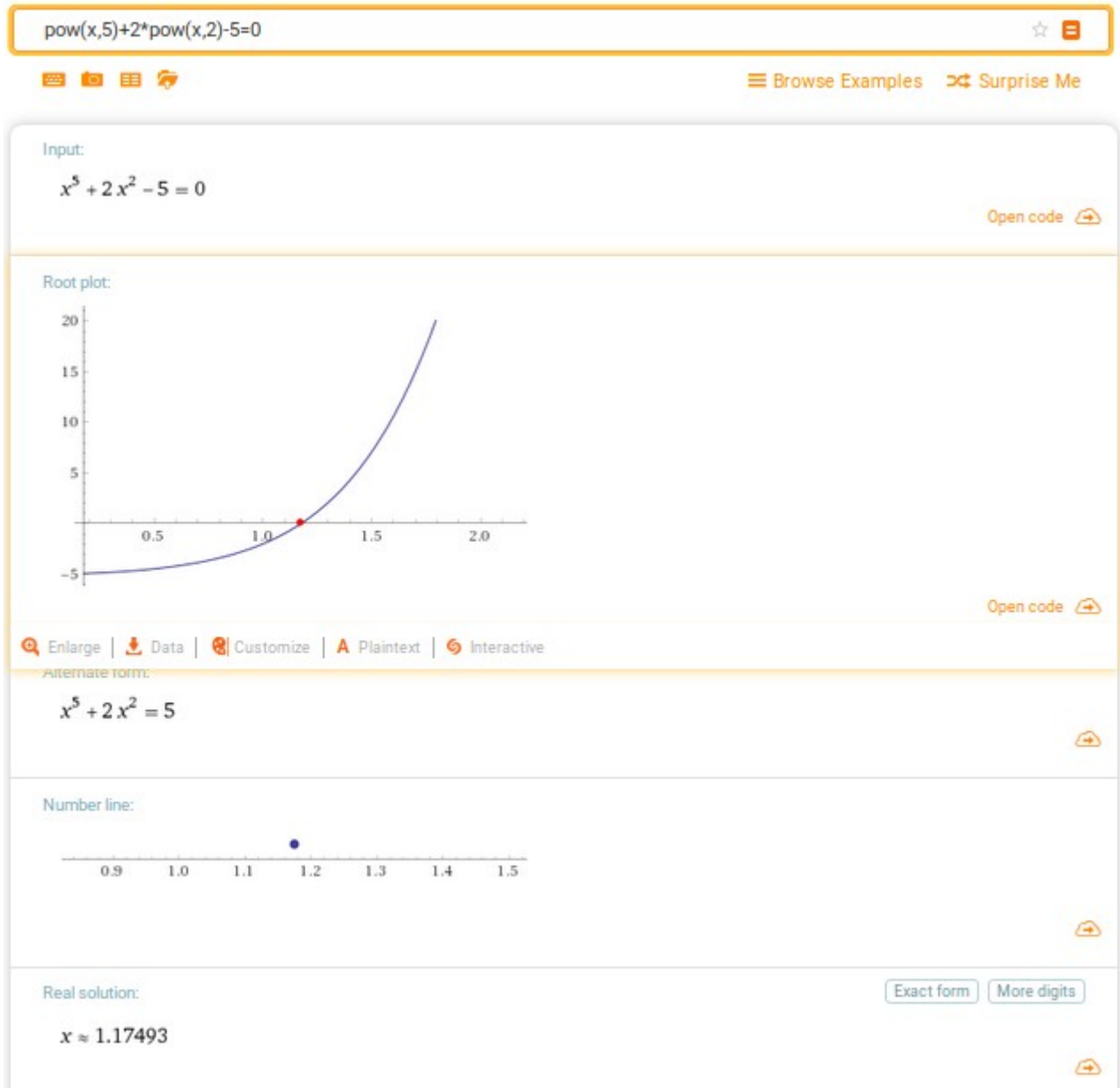


Рисунок 1.2.1.25 – Верификация данных через WolframAlpha

Теперь посмотрим на результат работы программы:

```
goru00@notebook:~/Documents/prog_homework/group_coursework/zadanie_10$ ./a.out
Введите значение с и d: 2 5
Kol-vo iteracii: 50
f(x) = 1.174933
```

Рисунок 1.2.1.26 – Иллюстрация работы программы

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.1.27 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
50	1.174933

Таблица 1.2.1.28 – Выходные данные

Как мы видим, программа работает правильно. Проведем еще несколько тестов для заключения.

Переменная <b>c</b>	Переменная <b>d</b>
1	3
5	8
7	2
1	4

Таблица 1.2.1.29 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
35	1.118340
97	1.118196
29	0.528960
38	1.205569

Таблица 1.2.1.30 – Выходные данные

**В заключение** к разделу можно сказать, что на протяжении решения уравнений(уравнение №4, №5, №6) мы смогли определить корни уравнения с помощью метода касательных, попутно сверяя данные с WolframAlpha и нашей программой.

## 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  $x \in (a, b)$  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

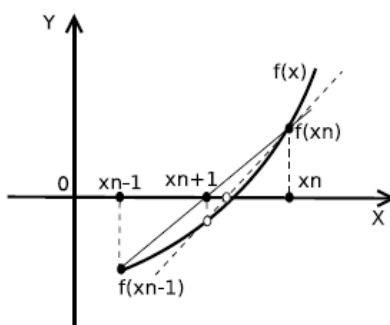


Рисунок 1.2.2.2 – Иллюстрация нахождения корня методом секущих

**Дано:**

Уравнение:  $x^3 + c x^2 + d = 0$  (уравнение №4).

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

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

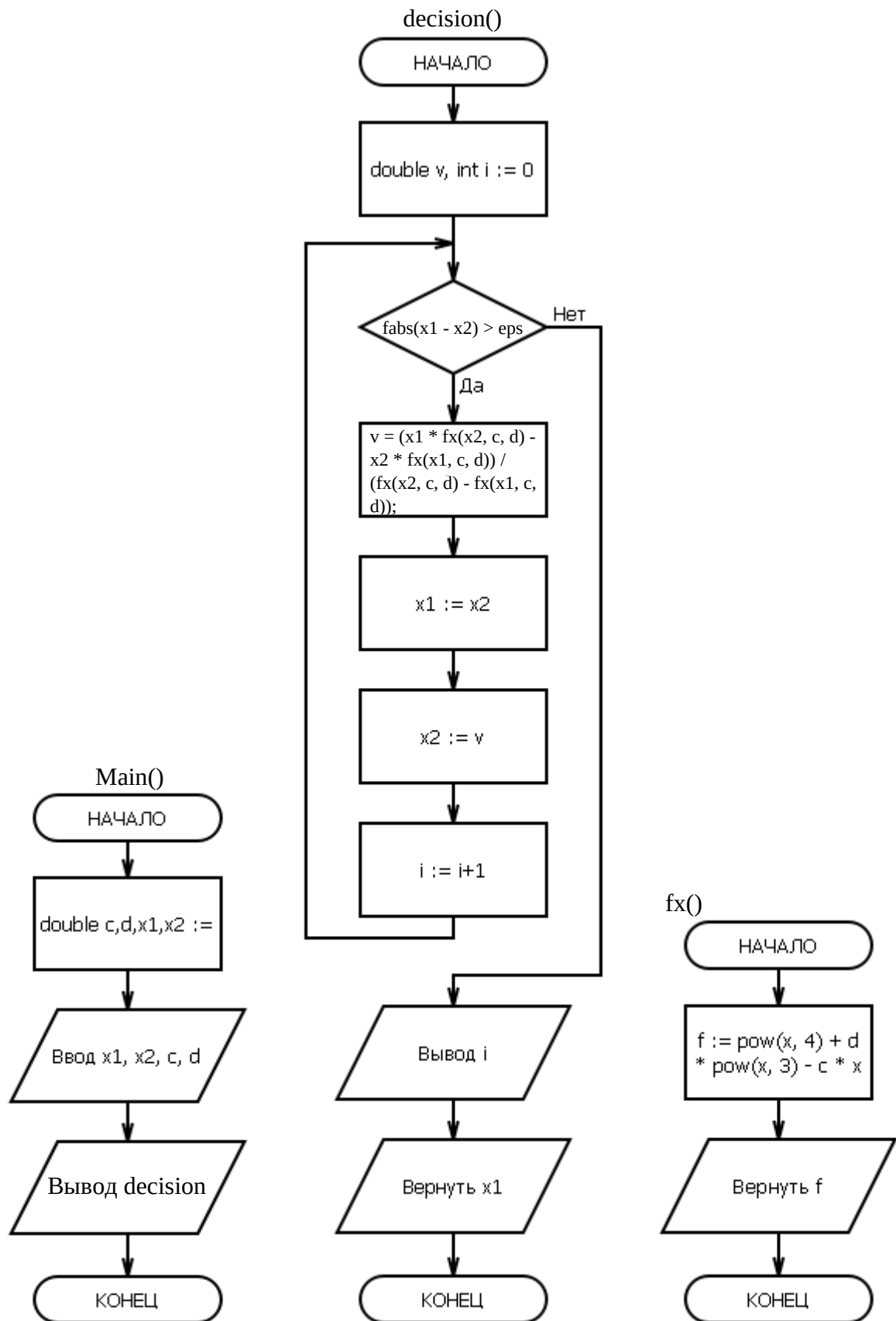
**Найти:**

Значение  $x$  методом секущих.

**Решение:**

Значение переменных  $c$ ,  $d$ ,  $x1$  и  $x2$  вводятся с клавиатуры пользователем. Значение точности(переменная  $\epsilon$ ):  $10^{-13}$ .

Для начала рассмотрим блок-схему будущей программы. На следующей странице представлена блок-схема программного кода.



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

```

#include <stdio.h>
#include <math.h>
#define eps 1e-13
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;
    return f;
}
double decision(func fx, double x1, double x2, double c, double d)
{
    double v; int i = 0;
    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));
        x1 = x2; x2 = v;
        i++;
    }
    printf("Итераций: %d\n", i);
    return x1;
}
int main()
{
    double c, d;
    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); // Вывод в консоль
интервала
    printf("Введите значение c и d: "); scanf("%lf %lf", &c, &d);
    printf("x = %f\n", decision(fx, x1, x2, c, d)); // Вывод в консоль ответа
    return 0;
}

```

Листинг 1.2.2.3 - Листинг программного кода нахождения корня уравнения методом секущих



Рассмотрим уравнение №4:

$$x^3 + c x^2 + d$$

Формула 1.2.2.4 – Уравнение №4

Проверим сначала результаты через WolframAlpha:

Введем значения **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.2.5 – Входные данные

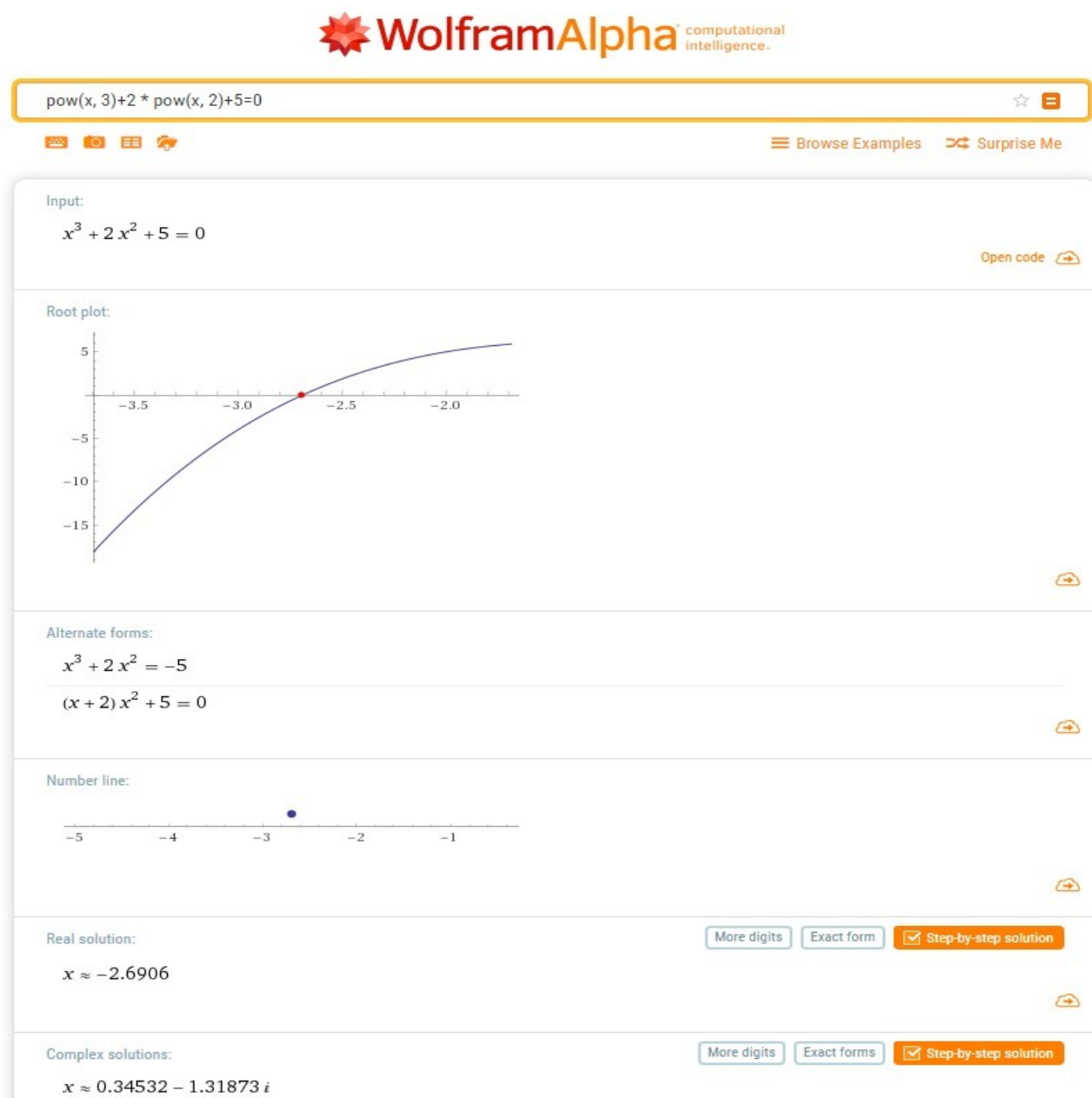


Рисунок 1.2.2.6 – Верификация данных через WolframAlpha

Теперь посмотрим, что выдает наша программа:

```
goru00@notebook:~/Documents/prog_homework/coursework$ ./a.out
Введите интервал(x2 и x1): 0 1
Введите значение c и d: 2 5
Итераций: 15
x = -2.690647
goru00@notebook:~/Documents/prog_homework/coursework$
```

Рисунок 1.2.2.7 – Иллюстрация работы программы

Переменная <b>x1</b>	Переменная <b>x2</b>	Переменная <b>c</b>	Переменная <b>d</b>
0	1	2	5

Таблица 1.2.2.8 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
15	-2.690647

Таблица 1.2.2.9 – Выходные данные

Попробуем провести еще какие-либо проверки при различных значениях **c, d, x1, x2**:

Переменная <b>x1</b>	Переменная <b>x2</b>	Переменная <b>c</b>	Переменная <b>d</b>
0	1	1	3
0	1	5	8
0	1	7	2
0	1	1	4

Таблица 1.2.2.10 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
10	-1.863707
26	-5.286279
41	-7.040350
2	-2.000000

Таблица 1.2.2.11 – Выходные данные

Как мы видим, программа работает правильно. Теперь перейдем к следующему уравнению и выполним то же самое, что и в первом.

Рассмотрим уравнение №5:

$$x^4 + c x^3 - dx$$

Формула 1.2.2.12 – Уравнение №5

Введем значения **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.2.13 – Входные данные

Так же посмотрим для начала на результаты WolframAlpha:

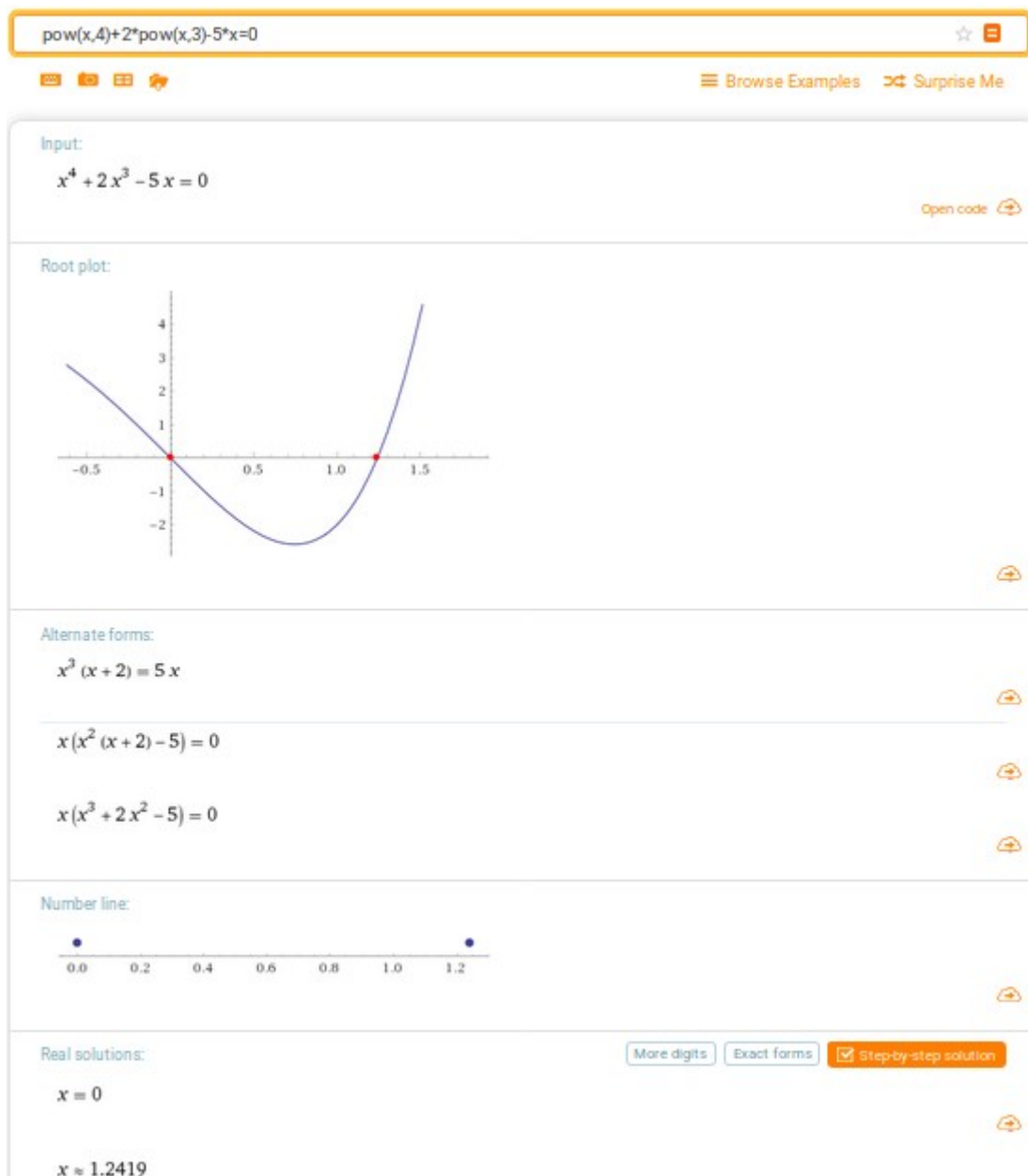


Рисунок 1.2.2.14 – Верификация данных через WolframAlpha

Теперь посмотрим на результат работы программы:

```
goru00@notebook:~/Documents/prog_homework/coursework$ ./a.out
Введите интервал(x2 и x1): 1 2
Введите значение с и d: 2 5
Итераций: 9
x = 1.241897
goru00@notebook:~/Documents/prog_homework/coursework$
```

Рисунок 1.2.2.15 – Иллюстрация работы программы

Переменная <b>x1</b>	Переменная <b>x2</b>	Переменная <b>c</b>	Переменная <b>d</b>
1	2	2	5

Таблица 1.2.2.16 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
---------------------	-------------

9	1.241897
---	----------

Таблица 1.2.2.17 – Выходные данные

Попробуем провести еще какие-либо проверки при различных значениях **c**, **d**, **x1**, **x2**:

Переменная <b>x1</b>	Переменная <b>x2</b>	Переменная <b>c</b>	Переменная <b>d</b>
1	2	1	3
1	2	5	8
0	1	7	2
0	1	1	4

Таблица 1.2.2.18 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
9	1.174559
9	1.141336
11	0.515853
10	1.314596

Таблица 1.2.2.19 – Выходные данные

Осталось рассмотреть последнее уравнение №6:

$$x^5 + c x^2 - d$$

Формула 1.2.2.20 – Уравнение №6

Введем значения **c** и **d**:

Переменная <b>c</b>	Переменная <b>d</b>
2	5

Таблица 1.2.2.21 – Входные данные

Проверим сначала результаты через WolframAlpha:

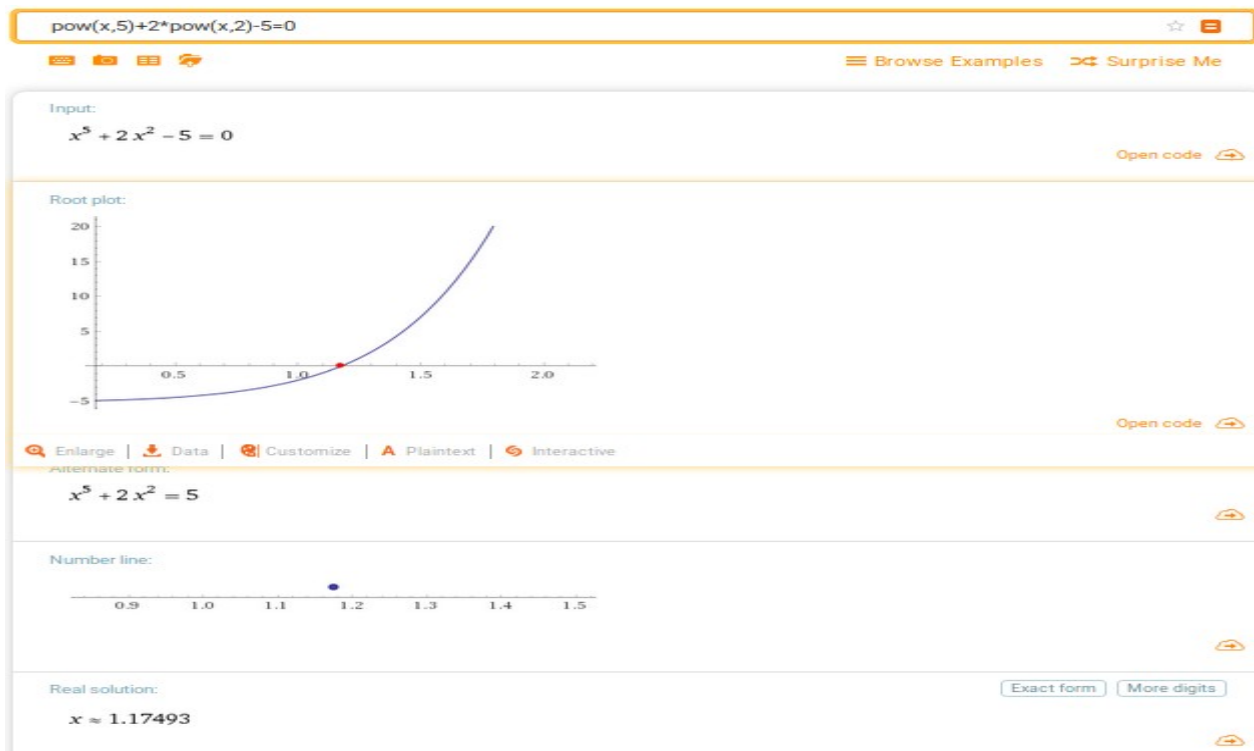


Рисунок 1.2.2.22 – Иллюстрация работы программы

Теперь посмотрим на результат работы программы:

```
goru00@notebook:~/Documents/prog_homework/coursework$ ./a.out
Введите интервал(x2 и x1): 1 2
Введите значение c и d: 2 5
Итераций: 9
x = 1.174933
goru00@notebook:~/Documents/prog_homework/coursework$
```

Рисунок 1.2.2.23 – Иллюстрация работы программы

Переменная <b>x1</b>	Переменная <b>x2</b>	Переменная <b>c</b>	Переменная <b>d</b>
1	2	2	5

Таблица 1.2.2.24 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
9	1.174933

Таблица 1.2.2.25 – Выходные данные

Попробуем провести еще какие-либо проверки при различных значениях **c**, **d**, **x1**, **x2**:

Переменная <b>x1</b>	Переменная <b>x2</b>	Переменная <b>c</b>	Переменная <b>d</b>
1	2	1	3
1	2	5	8
0	1	7	2
0	1	1	4

Таблица 1.2.2.26 – Входные данные

Переменная <b>i</b>	Ответ(f(x))
9	1.118340
8	1.118196
13	0.528960
10	1.205569

Таблица 1.2.2.27 – Выходные данные

В заключении первой главы мы проверим скорость вычисления первого и второго метода, а также попробуем придумать и решить несколько своих уравнений. Сравнивать мы будем по количеству итераций.

### Метод касательных

1)  $x^3 + cx^2 + d$

<b>c</b>	<b>d</b>	<b>i</b>	<b>x</b>
2	5	29	-2.690647
1	3	28	-1.863707
5	8	28	-5.286279
7	2	64	-7.040350
1	4	30	-2.000000

### Метод секущих

1)  $x^3 + cx^2 + d$

<b>x1</b>	<b>x2</b>	<b>c</b>	<b>d</b>	<b>i</b>	<b>x</b>
0	1	2	5	15	-2.690647
0	1	1	3	10	-1.863707
0	1	5	8	26	-5.286279
0	1	7	2	41	-7.040350
0	1	1	4	2	-2.000000

$$2) \ x^4 + c x^3 - d x$$

c	d	i	x
2	5	16	1.241897
1	3	12	1.174559
5	8	13	1.141336
7	2	20	0.515853
1	4	18	1.314596

$$2) \ x^3 + c x^2 + d$$

x1	x2	c	d	i	x
1	2	2	5	9	1.241897
1	2	1	3	9	1.174559
1	2	5	8	9	1.141336
0	1	7	2	11	0.515853
0	1	1	4	10	1.314596

$$3) \ x^5 + c x^2 - d$$

x1c	x2d	c i	d	x	x
1 2	2 5	250	5 1.174933	1.174933	1.174933
1 1	2 3	135	3 1.118340	1.118340	1.118340
1 5	2 8	597	8 1.118196	1.118196	1.118196
0 7	1 2	729	2 0.528960	0.528960	0.528960
0 1	1 4	138	4 1.205569	1.205569	1.205569

$$3) \ x^5 + c x^2 - d$$

Как мы можем увидеть, решения методов сходятся, но мы имеем разное число итераций. Можно сделать вывод, что метод секущих намного эффективнее метода касательных.

## Глава II. РАЗРАБОТКА ИГРОВОЙ ПРОГРАММЫ

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

В варианте №10 предлагается написать игровую программу «Крестики-Нолики». Эта игра была придумана еще римским изобретателем, но не успевшим доделать её, но несколько столетий назад французский математик случайно, решая трёхуровневую систему уравнений, открывает миру данную игру. Смысл игры заключается в том, чтобы расположить крестики или нолики по горизонтали, вертикали или по диагонали. Каждый игрок(обычно участвует в игре только двое людей) делает по очереди свой ход. Побеждает тот, кто первый соберет данную последовательность. Если ни один из игроков не собрал данную последовательность, объявляется ничья или игра начинается заново.

Для начала рассмотрения материала по главе, стоит обратить внимание на то, что данная программа компилировалась на наборе инструментов разработки программного обеспечения для создания приложений под ОС Windows – MinGW, который находится в открытом доступе в сети Интернет. Помимо этого, мы затронем работу заголовочного файла **conio.h**, работающий в операционных системах(ОС) MS-DOS, для создания текстового интерфейса пользователя, а также для «живого» взаимодействия с игровой программой. Кроме того, игра является кроссплатформенной(то есть файл с программой можно скомпилировать как и на Windows, так и на Linux).

В игре также присутствуют модификации. Это результаты, которые можно посмотреть в самой игре или в самом файле, откуда берутся результаты. Кроме этого, чтобы попасть в раздел «Результаты», пользователю предлагается интерактивное меню, где помимо этого раздела он может выбрать другие.

Перед вступлением в разбор основной части игровой программы, мы обратим своё внимание на стартовое меню нашей программы.

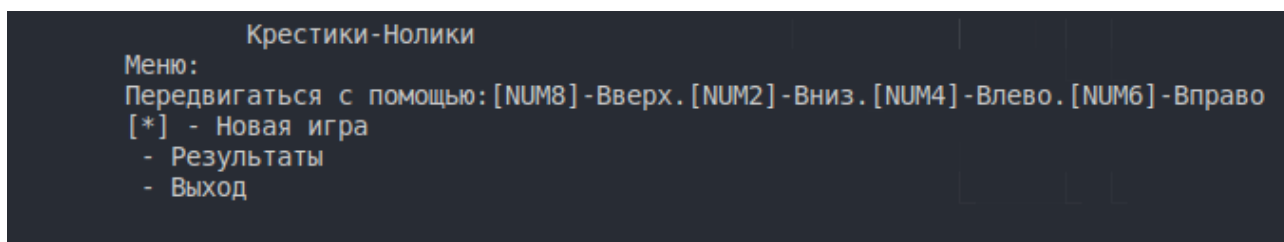


Рисунок 2.1 – Иллюстрация работы программы в пункте «Меню»

В данной части программы, пользователь может выбрать с помощью команд управления интересующий его пункт меню. В данном случае, управление реализуется с помощью библиотеки **conio.h** под Windows или bash-команды **stty**(устанавливает определенные характеристики терминального ввода/вывода для устройства).

Рассмотрим программный код реализации раздела «Меню». На следующей странице представлен листинг программного кода.

```

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;
                display();
                return i_menu;
            }
            if (i_menu > i_start) {
                i_menu--;
                display();
                return i_menu;
            }
        }
        case 50: // вниз
        {
            if (i_menu == i_end) {
                i_menu = i_start;
                display();
                return i_menu;
            }
        }
    }
}

```



```

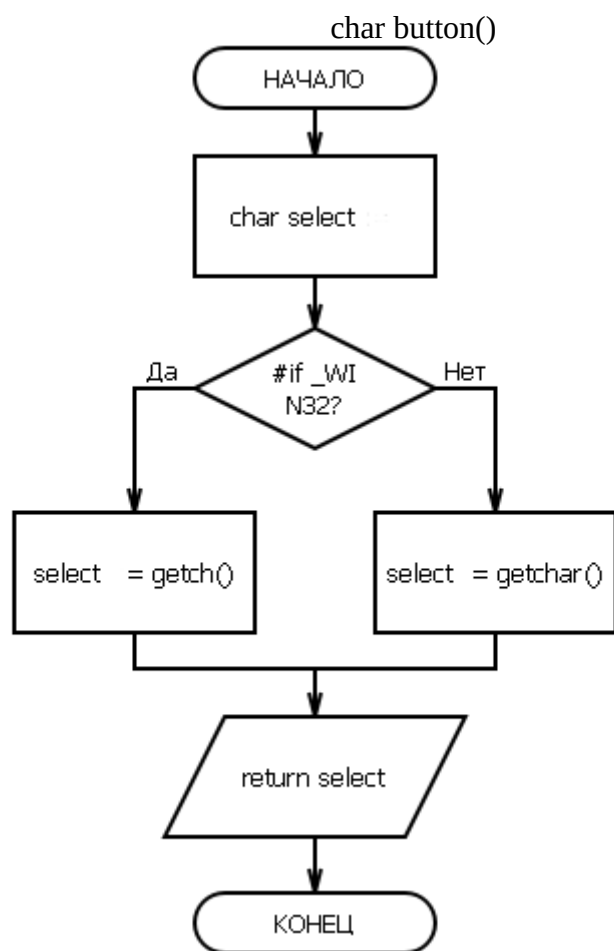
        } else {
            i_menu++;
            display();
            return i_menu;
        }
    }
    default:
    {
        display();
    }
}

char button()
{
    char select;
    #if _WIN32
        select = getch();
        printf("WIN\n");
    #else
        system("stty raw");
        select = getchar();
        system("stty cooked");
        printf("UNIX\n");
    #endif
    return select;
}

```

Листинг 2.2 –Программный код работы программы в пункте «Меню»

Теперь рассмотрим блок-схемы нашего программного кода:









## **Список литературы**