МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

(ФГБОУ ВО «ВГТУ», ВГТУ)

**Факультет** информационных технологий и компьютерной безопасности

**Кафедра** систем управления и информационных технологий в строительстве

Отчет по лабораторной работе №3

по дисциплине: «Информатика»

Выполнил студент: Зайцев Илья

Группа: бИСТ-221

Руководитель: доцент, к.т.н. Ефимова О.Е.

Работа защищена « » 2023г.

С оценкой

(подпись)

**Лабораторная работа №3. Решение нелинейных алгебраических уравнений.**

Цель работы: реализовать программные методы решения нелинейных алгебраических уравнений: метод деления отрезка пополам, метод хорд, метод секущих, метод касательных, метод простых итераций, метод секущих, комбинированный метод касательных и секущих.

**Задание на лабораторную работу**

1. Реализовать метод деления отрезка пополам и метод хорд.

2. Реализовать метод касательных и метод простых итераций.

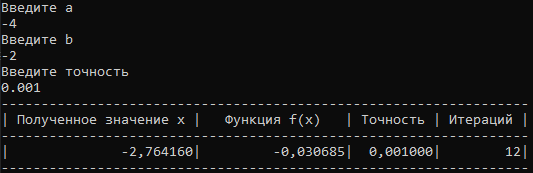
3. Реализовать метод секущих, построить его блок-схему, реализовать комбинированный метод хорд и касательных.

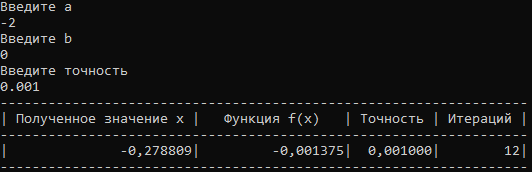


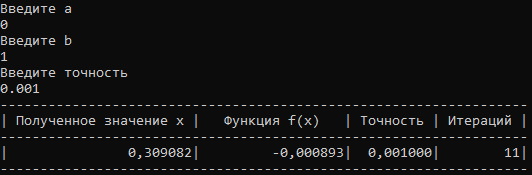
**Ход выполнения**

**Задание 1**

**Метод деления пополам**







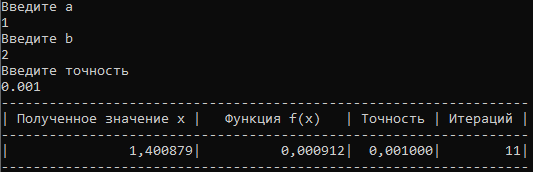
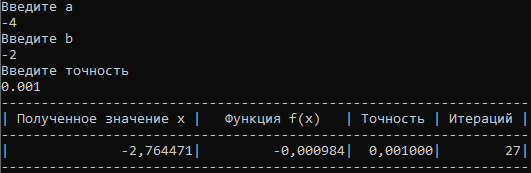
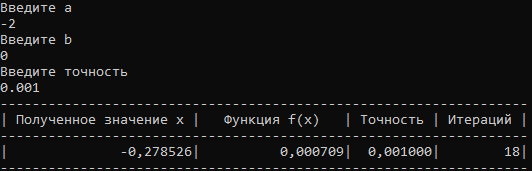
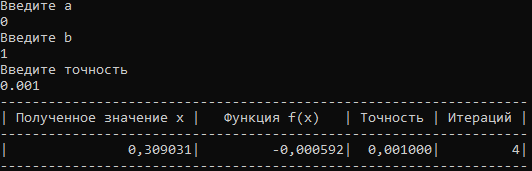


Рисунок 1.1, 1.2, 1.3, 1.4 – Результат работы метода деления пополам

**Метод хорд**







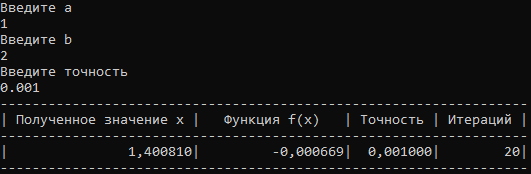
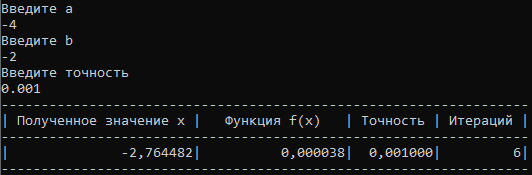
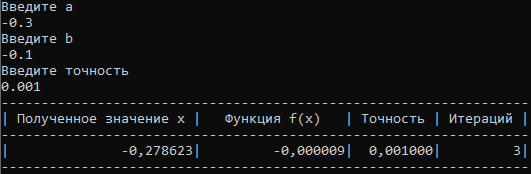


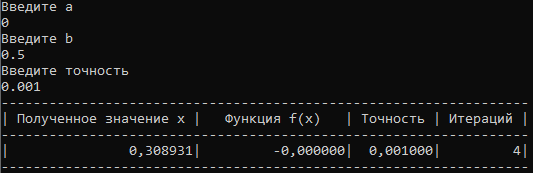
Рисунок 2.1, 2.2, 2.3, 2.4 – Результат работы метода хорд

**Задание 2**

**Метод касательных**







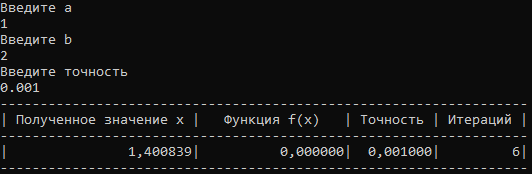
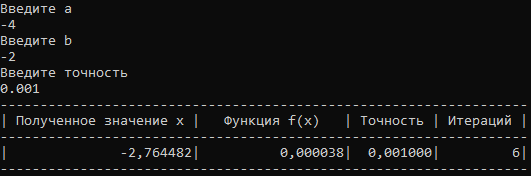
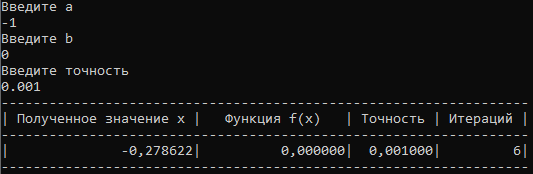
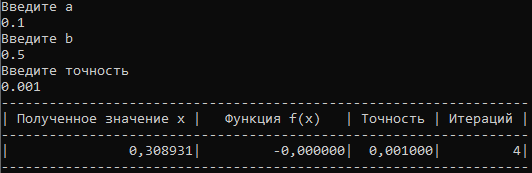


Рисунок 3.1, 3.2, 3.3, 3.4 – Результат работы метода касательных

**Метод простых итераций**







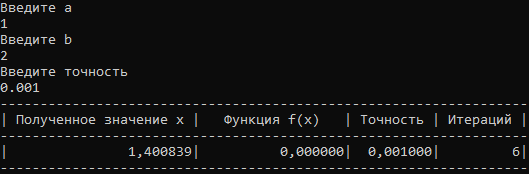


Рисунок 4.1, 4.2, 4.3, 4.4 – Результат работы метода простых итераций

**Задание 3**

**Метод секущих**

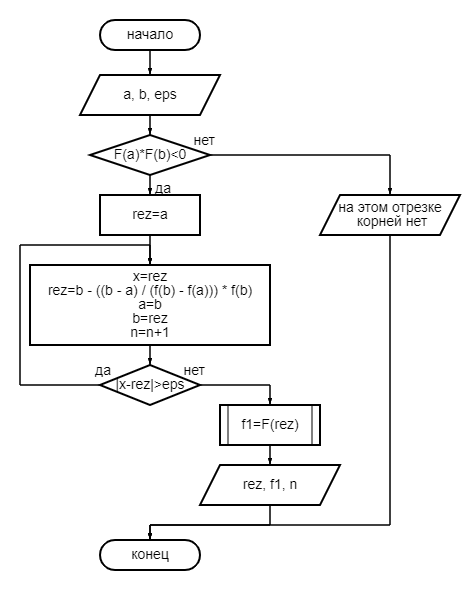
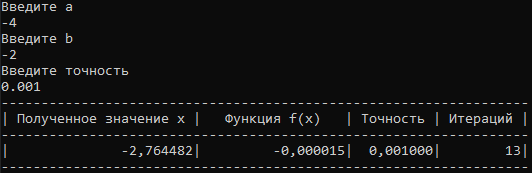
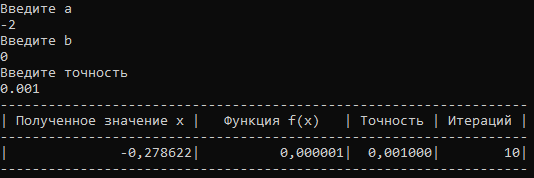
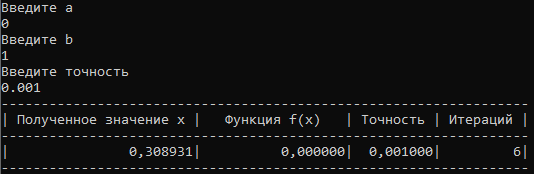
****

Рисунок 5-Блок-схема метода секущих

В методе секущих на каждом шаге вычисляется только одно значение функции, в то время как в методе касательных вычисляются значения функции и ее производной. В методе Ньютона из-за погрешности в определении производной для нахождения корня требуется выполнить большее количество шагов.







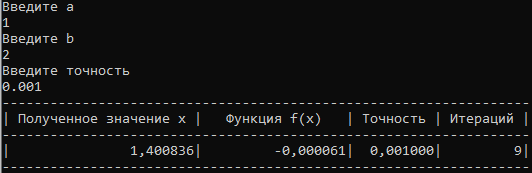
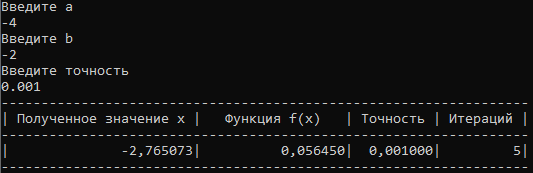


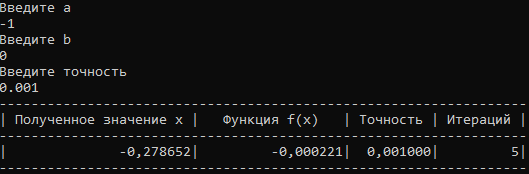
Рисунок 6.1, 6.2, 6.3, 6.4 – Результат работы метода секущих

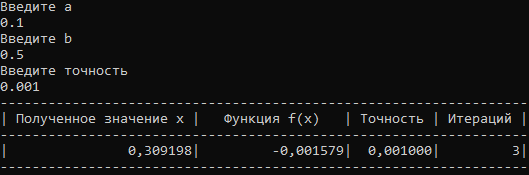
**Комбинированный метод хорд и касательных**

Методы хорд и касательных дают приближения корня с разных сторон. Поэтому их часто применяют в сочетании друг с другом, и уточнение корня происходит быстрее.

Преимущества комбинированного метода хорд и касательных заключаются в том, что он обычно сходится быстрее, чем методы хорд и касательных, применяемые по отдельности. Также он может сходиться к корню, который находится далеко от начального приближения. Однако он может быть более сложным в реализации и требовать больше вычислительных ресурсов.







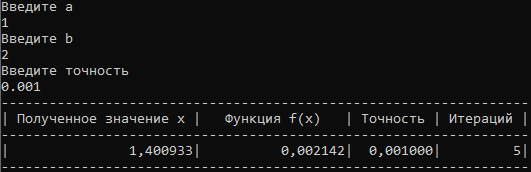


Рисунок 7.1, 7.2, 7.3, 7.4 – Результат работы комбинированного метода хорд и касательных

**Вывод**: в ходе лабораторной работы были применены методы для поиска корней нелинейного выражения и выявлены следующие результаты:

* Самым быстрым методом в решении нашего выражения является комбинированный метод хорд и касательных;
* Самым точным методом в решении нашего выражения является метод касательных;
* Самым медленным методом в решении нашего выражения является метод хорд;

**Листинг программы**

#include <iostream>

using namespace std;

void ex\_1\_1(float a, float b, float eps); //Метод деления пополам

void ex\_1\_2(float a, float b, float eps); //Метод хорд

void ex\_2\_1(float a, float b, float eps); //Метод касательных

void ex\_2\_2(float a, float b, float eps); //Метод простых итераций

void ex\_3\_1(float a, float b, float eps); //Метод секущих

void ex\_3\_2(float a, float b, float eps); //Комбинированный метод хорд и касательных

void tabl(float a, float b, float c, int d);

float f(float x);

float df\_first(float x);

float df\_second(float x);

float g(float x);

int main()

{

setlocale(LC\_ALL, "RUS");

float a, b, eps;

int v, iter=0;

do

{

cout << "Выберите действие:\n"

"1-Решение уравнения методом деления отрезка пополам\n"

"2-Решение уравнения методом хорд\n"

"3-Решение уравнения методом Ньютона (метод касательных)\n"

"4-Решение уравнения методом последовательных приближений (метод простых итераций)\n"

"5-Решение уравнения методом секущих\n"

"6-Решение уравнения комбинированным методом хорд и касательных\n"

"0-Выход" << endl;

cin >> v;

if (v > 0 && v < 7)

{

cout << "Введите a" << endl;

cin >> a;

cout << "Введите b" << endl;

cin >> b;

cout << "Введите точность" << endl;

cin >> eps;

if (f(a) \* f(b) >= 0)

{

cout << "На этом отрезке корней нет" << endl;

continue;

}

}

else continue;

switch (v)

{

case 1:

{

ex\_1\_1(a, b, eps);

break;

}

case 2:

{

ex\_1\_2(a, b, eps);

break;

}

case 3:

{

ex\_2\_1(a, b, eps);

break;

}

case 4:

{

ex\_2\_2(a, b, eps);

break;

}

case 5:

{

ex\_3\_1(a, b, eps);

break;

}

case 6:

{

ex\_3\_2(a, b, eps);

break;

}

}

} while (v != 0);

}

void tabl(float a, float b, float c, int d)

{

printf("------------------------------------------------------------------\n");

printf("| Полученное значение x | Функция f(x) | Точность | Итераций |\n");

printf("------------------------------------------------------------------\n");

printf("|%23lf|%18lf|%10lf|%10d|\n", a, b, c, d);

printf("------------------------------------------------------------------\n");

}

float f(float x)

{

return 3 \* powf(x, 4) + 4 \* powf(x, 3) - 12 \* powf(x, 2) + 1;

}

float df\_first(float x)

{

return 12 \* powf(x, 3) + 12 \* powf(x, 2) - 24 \* x;

}

float df\_second(float x)

{

return 36 \* x \* x + 24 \* x - 24;

}

float g(float x) {

if (x == 0) x = 0.01f;

return -1.f/(3\*powf(x, 3)+4\*powf(x, 2)-12\*x);

}

void ex\_1\_1(float a, float b, float eps)

{

int n=1;

float f\_1 = f(a), f\_2, t;

do {

t = (a + b) / 2;

f\_2 = f(t);

if (f\_1 \* f\_2 < 0) b = t;

else

{

a = t;

f\_1 = f\_2;

}

n++;

} while (abs(b - a) > eps);

tabl((a + b) / 2, f((a + b) / 2), eps, n);

}

void ex\_1\_2(float a, float b, float eps)

{

float c;

int n=1;

do {

c = (f(b) \* a - f(a) \* b) / (f(b) - f(a));

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

else b = c;

n++;

} while (abs(f(c))>eps);

tabl(c, f(c), eps, n);

}

void ex\_2\_1(float a, float b, float eps)

{

if (a == 0)a = 0.001f;

if (b == 0)b = 0.001f;

int n = 1;

float fu = f(b), f\_2 = df\_second(b), f\_1;

float x, h;

if (fu \* f\_2 > 0) x = b;

else x = a;

do {

fu = f(x);

f\_1 = df\_first(x);

h = fu / f\_1;

x = x - h;

n++;

} while (fabs(h)>eps);

tabl(x, f(x), eps, n);

}

void ex\_2\_2(float a, float b, float eps)

{

float x0, x=a;

int n = 1;

do {

x0 = x;

x = g(x0);

n++;

} while (fabs(x0-x) > eps);

tabl(x, f(x), eps, n);

}

void ex\_3\_1(float a, float b, float eps)

{

float x, rez=a;

int n=1;

do

{

x = rez;

rez = b - ((b - a) / (f(b) - f(a))) \* f(b);

a = b;

b = rez;

n++;

} while (fabs(x - rez) >= eps);

tabl(rez, f(rez), eps, n);

}

void ex\_3\_2(float a, float b, float eps)

{

int n = 1;

float xh, xk;

if (f(a) \* df\_second(a) > 0) {

xk = a;

xh = b;

}

else {

xk = b;

xh = a;

}

do

{

xk = xk - (f(xk)) / (df\_first(xk));

xh= (xh - (f(xh)) / (f(xk) - f(xh)) \* (xk - xh));

n++;

} while (fabs(xk - xh) > eps);

tabl(xk, f(xk), eps, n);

}