Цели работы:

а) освоение методов решения нелинейных уравнений;
б) совершенствование навыков по алгоритмизации и программированию вычислительных задач.

Вариант задания:

14.	$ln(x+a)+(x+b)^5=0$	Хорд $\varepsilon = 3\cdot10^{-5}$	$a = 2.11;\ b = -4.03$

Математическая часть:

Уравнением называется равенство

$$f(x) = 0, \tag{2.1}$$

справедливое при некоторых значениях $x=x*$, называемыми корнями этого уравнения или нулями функции $f(x)$. Решение уравнения заключается в определении его корней. Среди корней $x*$ могут быть и комплексные, однако в данной работе вычисляются только действительные корни.

Вычисление каждого из действительных корней складывается из двух этапов:

1) отделение корня, т.е. нахождение возможно малого интервала $[a, b]$, в пределах которого находится один и только один корень $x*$ уравнения;

2) уточнение значения корня, т.е. вычисление с заданной степенью точности.

При использовании рассматриваемых ниже методов решения уравнения (2.1) к функции $f(x)$ на интервале $[a,b]$ предъявляются следующие требования:

a) функция $f(x)$ непрерывна и дважды дифференцируема (т.е. существует первая и вторая производные);

b) первая производная $f'(x)$ непрерывна, сохраняет знак и не обращается в нуль;

c) вторая производная $f''(x)$ непрерывна и сохраняет знак.

Отделение корней может производиться аналитическим или графическим способами. Аналитический способ основывается на теореме Коши, утверждающей, что для непрерывной функции $f(x)$ (первое требование “a”), принимающей на концах интервала $[a, b]$ разные знаки, т.е. $f(a)\cdot f(b)<0$, уравнение (2.1) имеет внутри этого интервала хотя бы один корень (рис. 1). Если к этому добавить второе требование “b”, означающее монотонность функции $f(x)$, то этот корень оказывается единственным.

В этих условиях отделение корня сводится к вычислению значений функции $f(x)$ для последовательности точек α_1, α_2, …, α_n и сопоставлению знаков $f(\alpha_k)$, $f(\alpha_{k+1})$ в соседних точках α_k и α_{k+1}. Каждый интервал $[\alpha_k$, $\alpha_{k+1}]$, для которого $f(\alpha_k) \cdot f(\alpha_{k+1})<0$, содержит, по крайней мере, один корень уравнения. Этот корень является единственным, если на этом интервале выполняется второе требование “b”. В противном случае следует интервал $[\alpha_k, \alpha_{k+1}]$ разделить на меньшие интервалы, повторяя для каждого из них указанные действия.

При использовании графического способа уравнение (2.1) можно также представить в виде

$$f_1(x) = f_2(x) \tag{2.2}$$

и построить графики функций $y=f_1(x)$ и $y=f_2(x)$. Абсцисса точки пересечения этих графиков дает приближенное значение x^0 корня x^* уравнения

$$f(x) = f_1(x) - f_2(x) = 0.$$

Представление уравнения (2.1) в форме (2.2) не является, естественно, однозначным и его следует подбирать так, чтобы построение графиков было возможно простым.

Из того же чертежа следует определить и тот интервал $[a, b]$, в пределах которого данный корень является единственным (если это необходимо для выбранного метода последующего уточнения значения корня x^0);

Метод хорд:

Пусть определен интервал $[a, b]$, в котором лежит один корень x^* уравнения (2.1) $f(x)=0$.

Учитывая, что $f(a) \cdot f(b)<0$, определяем первое приближение как точку пересечения с осью абсцисс хорды A_0B_0, соединяющей точки $A_0[a, f(a)]$ и $B_0[b, f(b)]$ (рис. 2.6,а).

Для нахождения последующего приближения вычислим значение $f(x_1)$ и сопоставим со значениями $f(a)$ и $f(b)$. Выберем тот из интервалов $[a,x_1]$ или $[x_1,b]$, на концах которого функция $f(x)$ имеет разные знаки (именно внутри этого интервала лежит искомый корень x^*). Применим предыдущий прием к этому интервалу, получая последующее приближение – точку x_2.

Заметим, что для случая, изображенного на рис. 9а, производные $f'(x)$ и $f''(x)$ сохраняют положительный знак ($f'(x)>0$, $f''(x)>0$; $f'(x) \cdot f''(x)>0$) и все приближения $x_1, x_2, \ldots$ образуют возрастающую последовательность, ограниченную значением $x=x*$. Следовательно, $\lim_{n\to\infty} x_n = x^*$, и при этом в любом из приближений соответствующая хорда проходит через начальную точку $B_0[b,f(b)]$.

Для получения формулы, определяющей последующие приближения, рассмотрим переход от x_n и x_{n+1}. В этом случае уравнение хорды B_nB_0 как прямой, проходящей через точки B_n,B_0, имеет вид

$$\frac{y-f(x_n)}{f(b)-f(x_n)} = \frac{x-x_n}{b-x_n}.$$

Если для определения x_{n+1} положить $y(x_{n+1})=0$, то получим

$$x_{n+1} = x_n - f(x_n)\frac{b-x_n}{f(b)-f(x_n)} \qquad (n=0,1,2,...) \qquad (2.15)$$

Для оценки погрешностей вычислений используется неравенство

$$|x_{n+1} - x^*| \le \frac{M-m}{m}|x_{n+1} - x_n|, \qquad \text{где} \quad 0< m \le |f'(x)| \le M < 1.$$

Если при этом $M \le 2m$, то $| x_{n+1}-x*| \le | x_{n+1}-x_n|$, и для заданной погрешности ε вычисления прекращаются при $| x_{n+1}-x_n| \le \varepsilon$ (как это имело место и для методов последовательных приближений и метода касательных).

При выполнении упомянутых требований *(“a”,”b”,”c”)* возможны и иные картины построений для метода хорд, определяемые сочетаниями знаков производных $f'(x)$ и $f''(x)$.

Рис. 2.6,а соответствует рассмотренному уже случаю $f'(x)>0, f''(x)>0$ (функция $f(x)$ монотонно возрастает и выпукла вниз). Случай $f'(x),f''(x)<0$ (рис. 2.6,б) приводит к аналогичным построениям, и последовательность $x_1,x_2,\dots$ оказывается так же возрастающей.

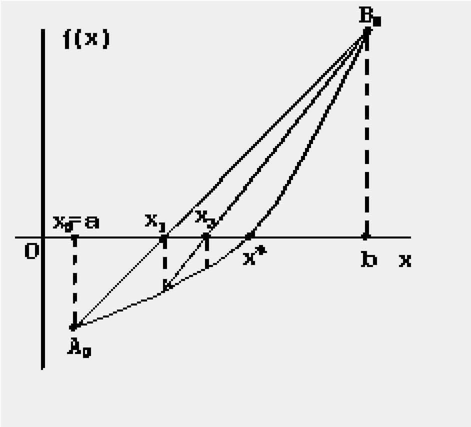
Однако в случаях $f'(x)>0, f''(x)<0$ (рис. 9,в) и $f'(x) <0, f''(x)>0$ (рис. 2.6,г) после определения каждого x_n различными оказываются знаки значений функций $f(a)$ и $f(x_n)$ (а не $f(x_n)$ и $f(b)$, как ранее). Поэтому “неподвижной” для всех хорд оказывается точка $A_0[a,f(a)]$ (а не $B_0[b,f(b)]$). В результате расчетными являются формулы

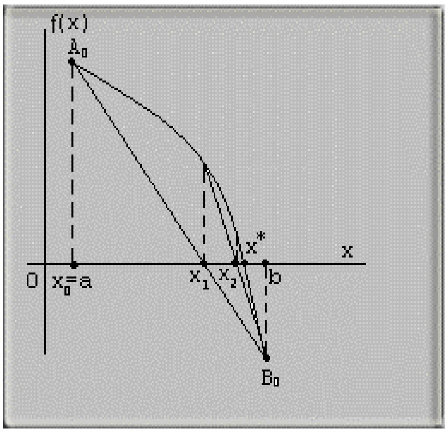
$$x_{n+1} = x_n - f(x_n)\frac{x_n-a}{f(x_n)-f(a)} \qquad (n=0,1,2,\dots), \qquad (2.16)$$

а последовательность $x_1,x_2,\dots$ оказывается убывающей.

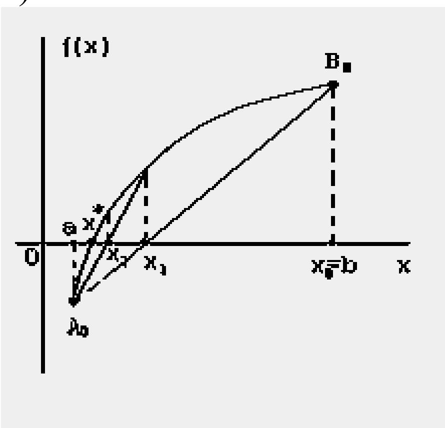
Таким образом, если $f'(x)\cdot f''(x)>0$, то следует использовать формулы (2.15), выбирая за начальное значение $x_0=a$, если же $f'(x)\cdot f''(x)<0$, то используются формулы (2.16) и начальным является $x_0=b$.

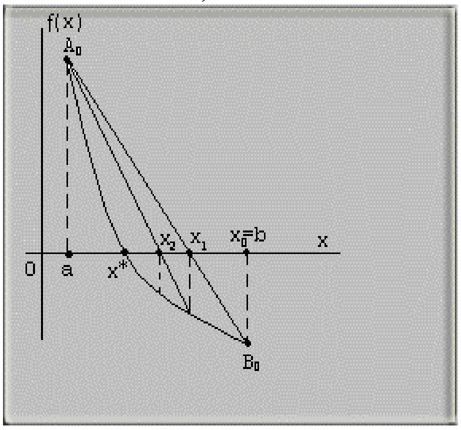
а) б)





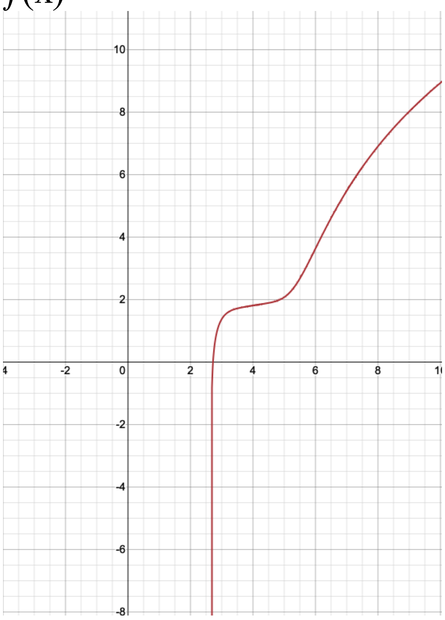
в) г)





Расчеты:

f(x)



f(x) = ln(x + 2.11) + (x - 4.03)^5 = 0

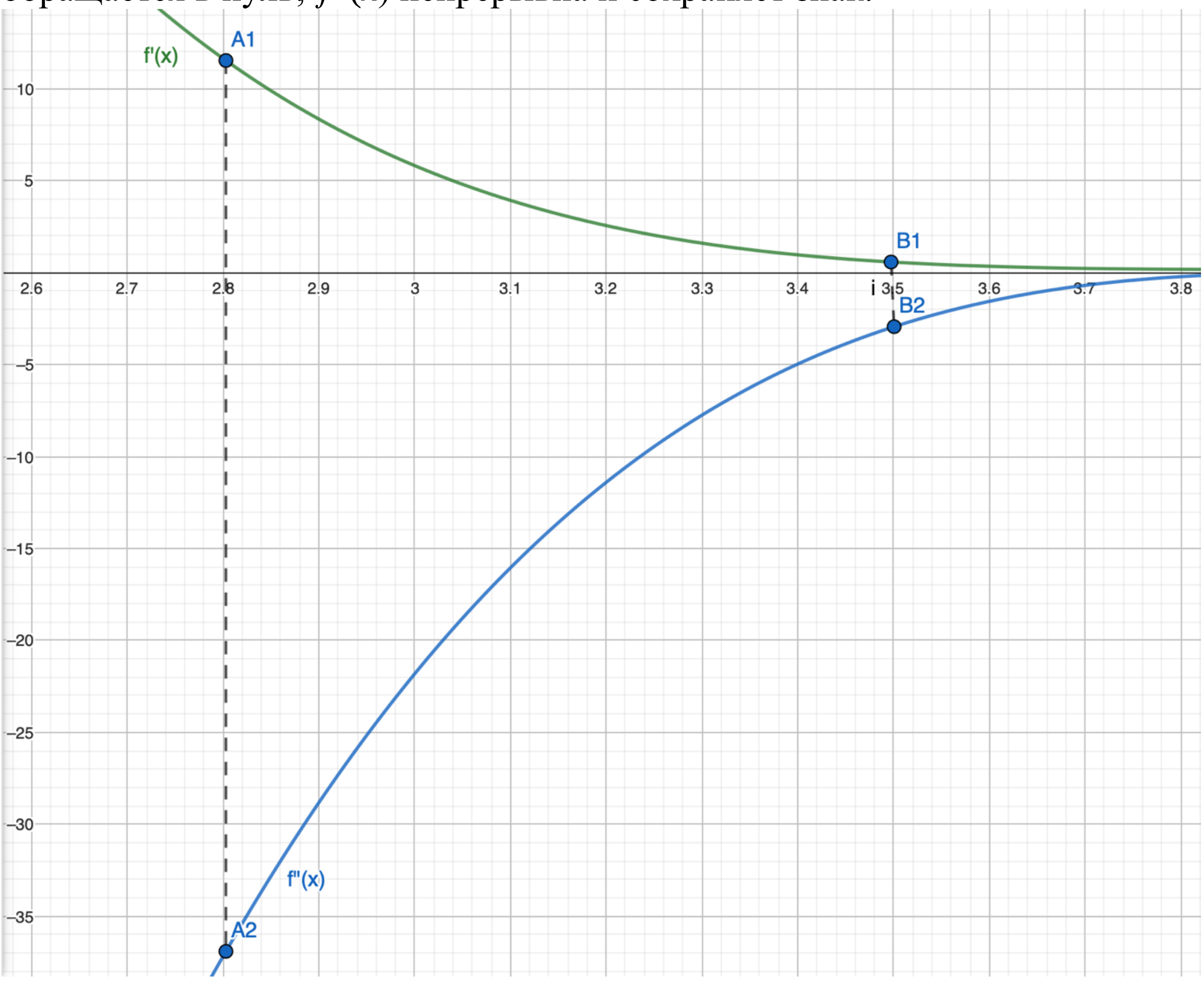
Пусть начальное приближение [2,8; 3,5].

$$f\,'(x) = 5\,(x - 4.03)^4 + \frac{1}{x+2.11};\quad f''(x) = 20(x-4.03)^3 + \frac{1}{(x+2.11)^2};$$

$f(2{,}8) = -1{,}2 < 0$; $f(3{,}5) = 1{,}7 > 0$,

Поскольку f (2.8)* f(3.4) <0 (т.е. значения функции на его концах имеют противоположные знаки), то корень лежит в пределах [2.8; 3.5].

На следующем графике видно, что f'(x) непрерывна, сохраняет знак и не обращается в нуль; $f''(x)$ непрерывна и сохраняет знак.



Поскольку f(a)f ''(a) < 0, то $x_0 = 3.5$

Шаг	x	F(x)
x2	3.09477	0.934098
x3	2.96718	0.268664
x4	2.93709	0.059566
x5	2.93073	0.012391
x6	2.92942	0.002543
x7	2.92915	0.000520
x8	2.92910	0.000106
x9	2.92909	0.000022

Т.к. |x9 – x8| = 0.00003, то за приближенное значение корня следует принять $x \approx {}^{*}x_9 = 2.92909$, f(x) = 0.000022

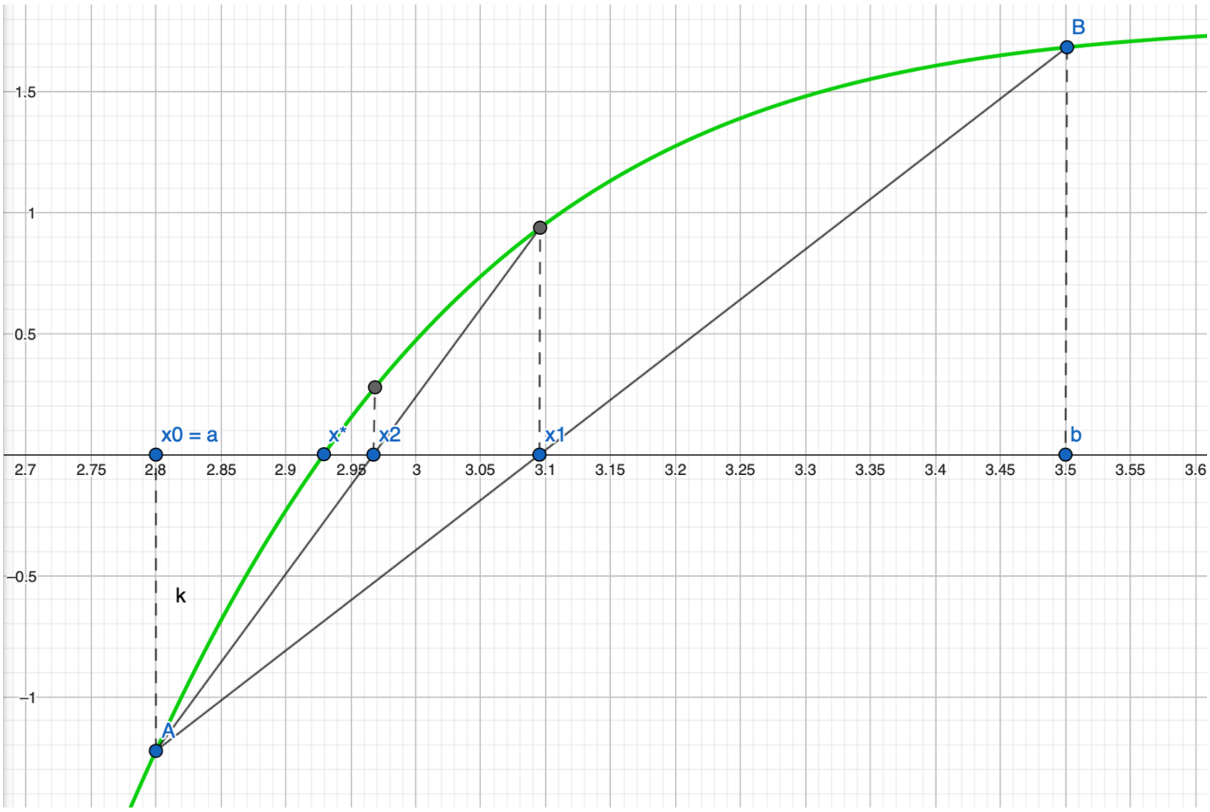
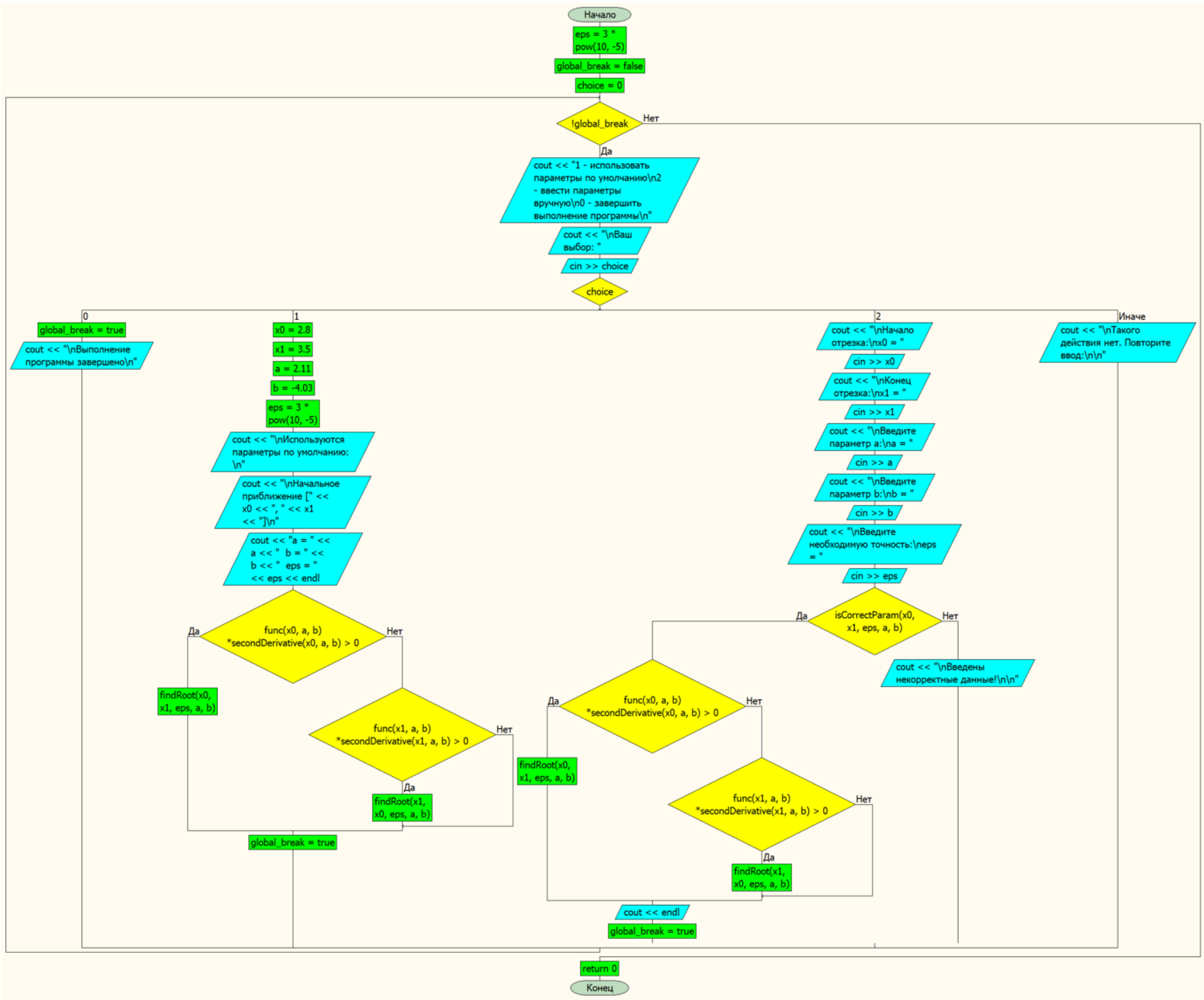
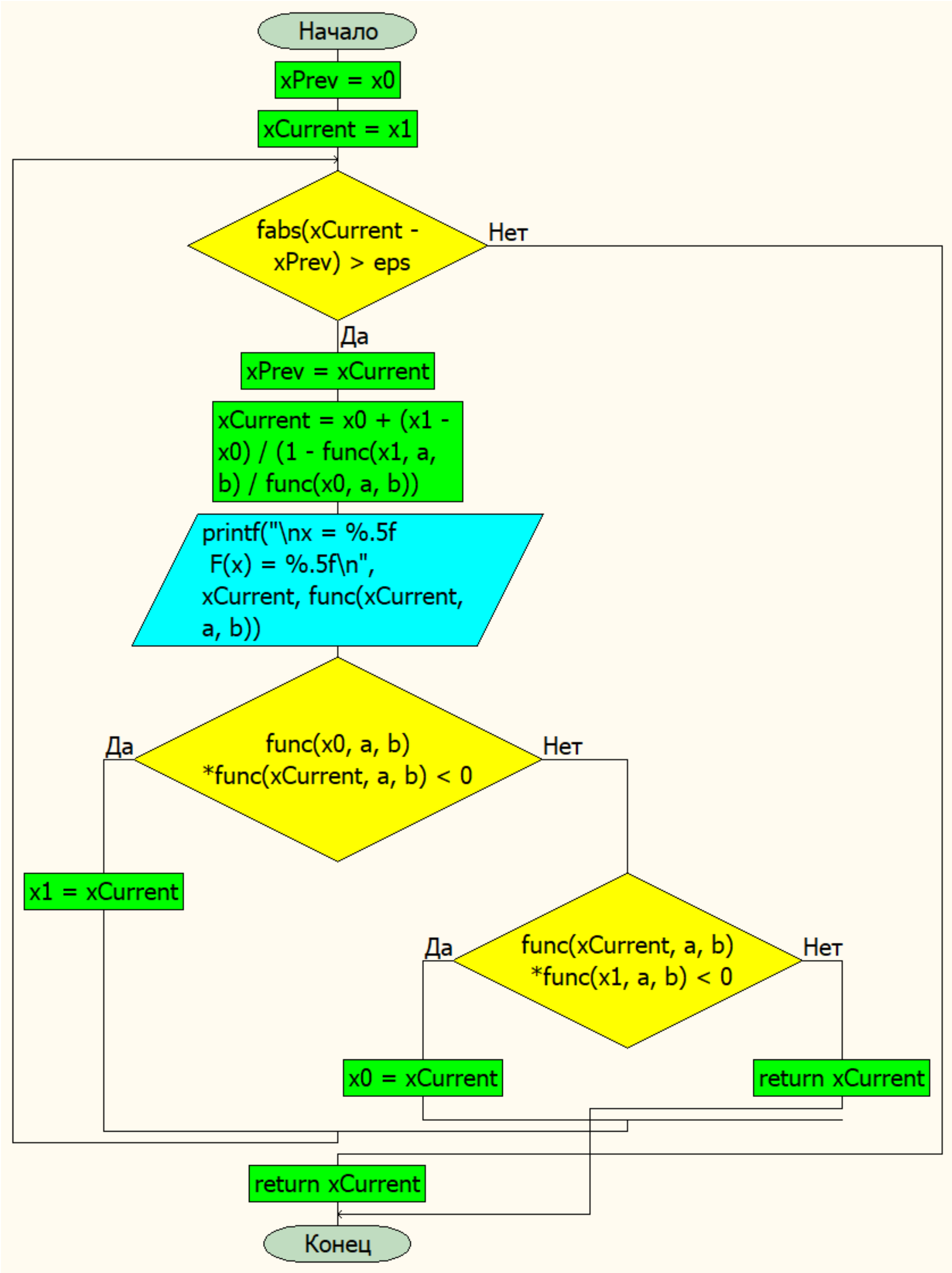


Схема алгоритма:

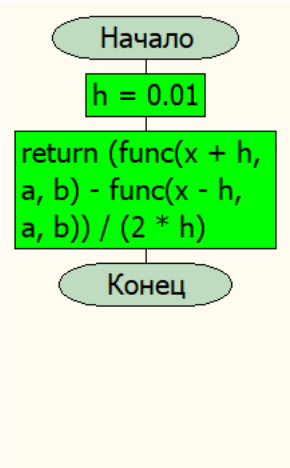
main():



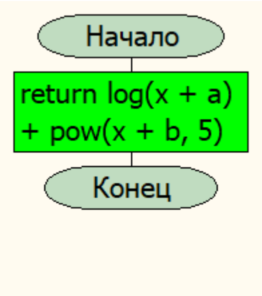
findRoot():



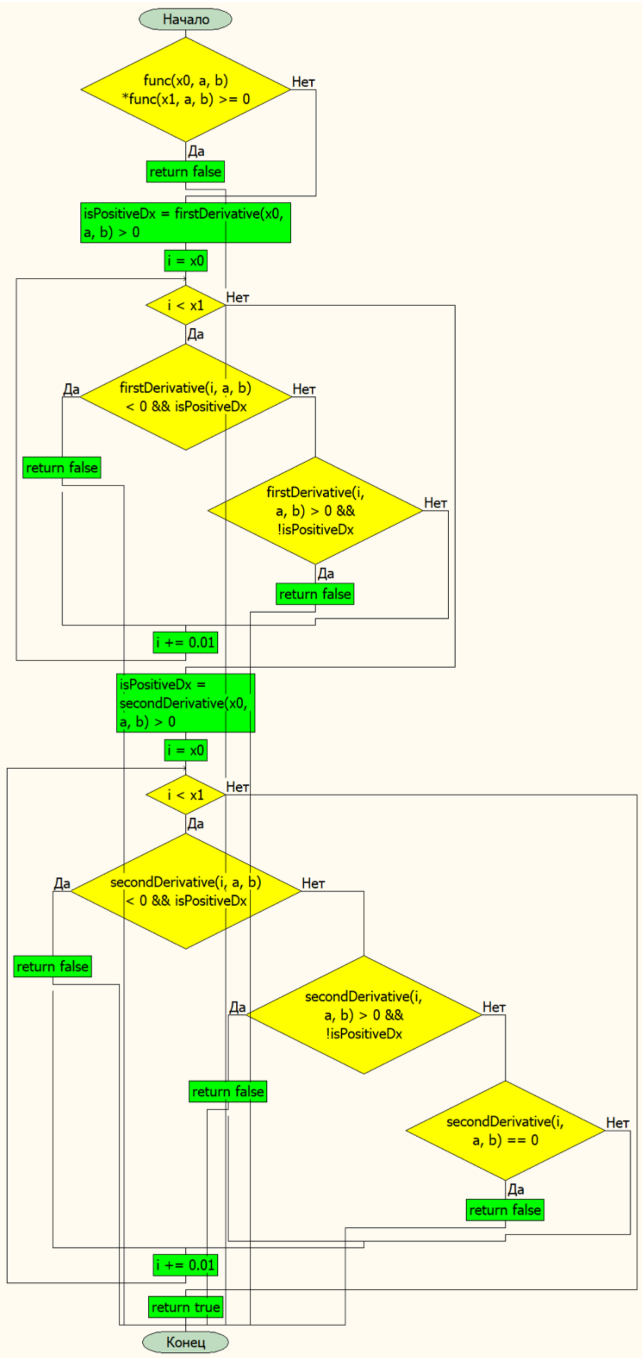
firstDerivative():



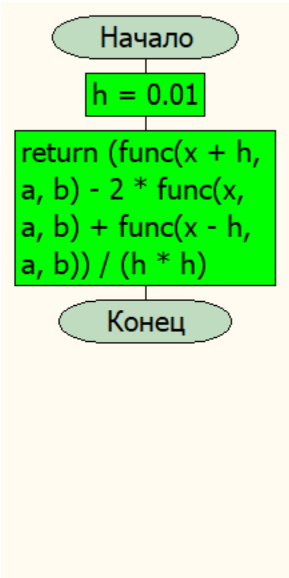
Func():



Iscorrectparam():



secondDerivative():



Результат работы программы:

1 – используются данные из аналитических расчетов:

```
1 - использовать параметры по умолчанию
2 - ввести параметры вручную
0 - завершить выполнение программы

Ваш выбор: 1

Используются параметры по умолчанию:

Начальное приближение [2.8, 3.5]
a = 2.11  b = -4.03  eps = 3e-05

x = 3.09477  F(x) = 0.93410

x = 2.96718  F(x) = 0.26866

x = 2.93709  F(x) = 0.05957

x = 2.93073  F(x) = 0.01239

x = 2.92942  F(x) = 0.00254

x = 2.92915  F(x) = 0.00052

x = 2.92910  F(x) = 0.00011

x = 2.92909  F(x) = 0.00002

Результат:
x = 2.92909  F(x) = 0.00002

Program ended with exit code: 0
```

Рис1.Данные из аналит.расчетов

2 – ручной ввод

```
1 - использовать параметры по умолчанию
2 - ввести параметры вручную
0 - завершить выполнение программы

Ваш выбор: 2

Начало отрезка:
x0 = 2.8

Конец отрезка:
x1 = 3.5

Введите параметр a:
a = 2.11

Введите параметр b:
b = -4.03

Введите необходимую точность:
eps = 0.0003

x = 3.09477  F(x) = 0.93410

x = 2.96718  F(x) = 0.26866

x = 2.93709  F(x) = 0.05957

x = 2.93073  F(x) = 0.01239

x = 2.92942  F(x) = 0.00254

x = 2.92915  F(x) = 0.00052

Program ended with exit code: 0
```

Рис2.Ручной ввод; корректный

```
1 – использовать параметры по умолчанию
2 – ввести параметры вручную
0 – завершить выполнение программы

Ваш выбор: 2

Начало отрезка:
x0 = -3

Конец отрезка:
x1 = 3

Введите параметр a:
a = 2.11

Введите параметр b:
b = -4.03

Введите необходимую точность:
eps = 0.00003

Введены некорректные данные!

1 – использовать параметры по умолчанию
2 – ввести параметры вручную
0 – завершить выполнение программы

Ваш выбор:
```

Рис2.Ручной ввод; не выполняется т.Коши

```
1 – использовать параметры по умолчанию
2 – ввести параметры вручную
0 – завершить выполнение программы

Ваш выбор: 2

Начало отрезка:
x0 = 3

Конец отрезка:
x1 = 4

Введите параметр a:
a = 2.11

Введите параметр b:
b = -4.03

Введите необходимую точность:
eps = 0.00003

Введены некорректные данные!

1 – использовать параметры по умолчанию
2 – ввести параметры вручную
0 – завершить выполнение программы

Ваш выбор:
```

Рис3.Ручной ввод; внутри интервала нет корня

Листинг:

```
#include <iostream>
#include <cmath>

using namespace std;

double func(double x, double a, double b) {
   return log(x + a) + pow(x + b, 5);
}

double firstDerivative(double x, double a, double b) {
   double h = 0.01; // шаг производной
   return (func(x + h, a, b) - func(x - h, a, b)) / (2 * h);
}

double secondDerivative(double x, double a, double b) {
   double h = 0.01; // шаг производной
   return (func(x + h, a, b) - 2 * func(x, a, b) + func(x - h, a, b)) / (h * h);
}

double findRoot(double x0, double x1, double eps, double a, double b) {
   double xPrev = x0;
   double xCurrent = x1;

   while (fabs(xCurrent - xPrev) > eps) {
      xPrev = xCurrent;
      xCurrent = x0 + (x1 - x0) / (1 - func(x1, a, b) / func(x0, a, b));

      printf("\nx = %.5f  F(x) = %.5f\n" , xCurrent, func(xCurrent, a, b));

      if (func(x0, a, b) * func(xCurrent, a, b) < 0) {
         x1 = xCurrent;
      }
      else if (func(xCurrent, a, b) * func(x1, a, b) < 0) {
         x0 = xCurrent;
      }
      else
         return xCurrent;
   }

   return xCurrent;
}

bool isCorrectParam(double x0, double x1, double eps, double a, double b) {
   if ( func(x0, a, b) * func(x1, a, b) >= 0 )
      return false;

   bool isPositiveDx = firstDerivative(x0, a, b) > 0;
   double i = x0;

   while (i < x1) {
      if ( firstDerivative(i, a, b) < 0 && isPositiveDx )
         return false;
      else if ( firstDerivative(i, a, b) > 0 && !isPositiveDx)
         return false;
      i += 0.01;
   }

   isPositiveDx = secondDerivative(x0, a, b) > 0;
   i = x0;
```

```
    while (i < x1) {
        if ( secondDerivative(i, a, b) < 0 && isPositiveDx )
            return false;
        else if ( secondDerivative(i, a, b) > 0 && !isPositiveDx )
            return false;
        else if ( secondDerivative(i, a, b) == 0 )
            return false;
        i += 0.01;
    }
    return true;
}

int main() {
    double x0, x1;
    double a, b;
    double eps = 3 * pow(10, -5);
    bool global_break = false;
    int choice = 0;

    while (!global_break) {
        cout << "1 - использовать параметры по умолчанию\n2 - ввести параметры вручную\n0 -
завершить выполнение программы\n";
        cout << "\nВаш выбор: ";
        cin >> choice;

        switch (choice){
            case 0:
                global_break = true;
                cout << "\nВыполнение программы завершено\n";
                break;

            case 1:
                x0 = 2.8; x1 = 3.5;
                a = 2.11; b = -4.03;
                eps = 3 * pow(10, -5);

                cout << "\nИспользуются параметры по умолчанию: \n";
                cout << "\nНачальное приближение [" << x0 << ", " << x1 << "]\n";
                cout << "a = " << a << "  b = " << b << "  eps = " << eps << endl;

                if (func(x0, a, b) * secondDerivative(x0, a, b) > 0) {
                    findRoot(x0, x1, eps, a, b);
                } else if (func(x1, a, b) * secondDerivative(x1, a, b) > 0) {
                    findRoot(x1, x0, eps, a, b);
                }

                global_break = true;
                break;

            case 2:
                cout << "\nНачало отрезка:\nx0 = ";
                cin >> x0;

                cout << "\nКонец отрезка:\nx1 = ";
                cin >> x1;

                cout << "\nВведите параметр a:\na = ";
                cin >> a;

                cout << "\nВведите параметр b:\nb = ";
```

```
            cin >> b;

            cout << "\nВведите необходимую точность:\neps = ";
            cin >> eps;

            if (isCorrectParam(x0, x1, eps, a, b)) {
                if (func(x0, a, b) * secondDerivative(x0, a, b) > 0) {
                    findRoot(x0, x1, eps, a, b);
                } else if (func(x1, a, b) * secondDerivative(x1, a, b) > 0) {
                    findRoot(x1, x0, eps, a, b);
                }
                cout << endl;
                global_break = true;

            } else {
                cout << "\nВведены некорректные данные!\n\n";
            }
            break;

        default:
            cout << "\nТакого действия нет. Повторите ввод:\n\n";
            break;
        }
    }

    return 0;
}
```

Сравнение результатов программных и аналитических расчетов:

Результат аналитических расчетов: x = 2.92909; f(x) = 0.000022

Результат программных расчетов: x = 2.92909; f(x) = 0.00002

Результаты совпадают.

Вывод:

Я освоил решение нелинейных уравнений методом хорд. Усовершенствовал навыки по алгоритмизации и программированию вычислительных задач.