**Техническое задание (ТЗ)**

**Проект: Калькулятор для решения уравнений методом секущих**

**1. Общая информация**

1.1. Название:  
Калькулятор для решения алгебраических уравнений методом секущих.

1.2. Описание:  
Программа позволяет решать уравнения методом секущих с заданной точностью. Пользователь вводит уравнение, начальные приближения и точность, после чего программа вычисляет корень уравнения и выводит результат. Программа поддерживает повторные вычисления без перезапуска.

1.3. Цель разработки:  
Создать программный продукт для нахождения корней алгебраических уравнений методом секущих с возможностью ввода произвольных уравнений и параметров вычисления.

**2. Функциональные требования**

2.1. Программа должна предоставлять возможность ввода следующих данных:

* Алгебраическое уравнение (в формате С++), где переменной является x.
* Первое и второе начальные приближения x0 и x1​ для метода секущих.
* Шаг итераций.
* Точность вычисления ε погрешность (например, 0.01).

2.2. Программа должна решать уравнение с использованием метода секущих:

* Выполнять итерации по формуле метода секущих до тех пор, пока разница между двумя последовательными приближениями не станет меньше заданной точности ε погрешности.
* На каждой итерации выводить текущее значение x и значение функции f(x).

2.3. Программа должна:

* Выводить найденный корень уравнения.
* Предлагать пользователю возможность повторного ввода уравнения и параметров после завершения текущего вычисления.

**3. Нефункциональные требования**

3.1. **Интерфейс**:  
Программа должна работать в консольном режиме. Ввод данных производится с клавиатуры, результат выводится в консоль.

3.2. **Платформенная совместимость**:  
Программа должна быть совместима с операционными системами Windows, macOS и Linux, где установлен С++

3.3. **Надежность**:  
Программа должна обрабатывать следующие исключения:

* Некорректный формат уравнения (например, неправильное использование операторов).
* Деление на ноль в ходе выполнения итераций.
* Недопустимые типы данных (например, ввод текста вместо чисел для начальных приближений и точности).

3.4. **Производительность**:  
Программа должна завершать выполнение расчета за разумное время (в течение нескольких секунд для уравнений с обычной сложностью).

**4. Структура программы**

4.1. Модули:

* Основной модуль программы, который выполняет следующие функции:
  + Ввод данных от пользователя.
  + Обработка уравнения и выполнение метода секущих.
  + Вывод результата.



* Рис.1. График функции *y*= *x2-cosx*
* **Подготовка тестовых вариантов**

В формируемые тестовые варианты в качестве входных данных целесообразно добавить значение *nmax* – максимально допустимое количество итераций (*n<=nmax*). Это позволить сделать программу более гибкой, хорошо управляемой пользователем. Тогда каждый набор входных данных будет состоять из следующего списка: *nmax,* *eps, x0, x1.*

*Первый набор* тестовых вариантов для проверки работы разрабатываемой программы представлен таблицей на рисунке 2. Входные данные: *nmax=10, eps=0.01, x0 =-1, x1 =-0.5.*

*Второй набор тестовых вариантов* представлен таблицей на рисунке 3. Исходные *данные*: *nmax=10, eps=0.01, x0 =-0.5, x1 =-1*. Решением является x5=-0,824577.



Рис.2. Результаты для тестового варианта №2

*Третий тестовый набор* формируем с исходными данными: *nmax=10, eps=0.001, x0 =-1, x1 =-0.5.* Очевидно, что на экран пользователя в качестве искомого результата будет выводиться значение x5=-0,824107 (см. рис.4).



Рис.3. Результаты для тестового варианта №3

*Четвертый тестовый набор* формируем с исходными данными: *eps=0.01, x0 =-05, x1 =-0.5. (x0 =x1).* Пользователю должно быть выведено сообщение о недопустимости такого набора входных данных.

Кроме того, возможна ситуация, когда мы хотим ограничить количество итераций еще и их количеством. То есть задать необходимо *nmax –* максимально допустимое количество итераций *(n<=nmax).*

Для проверки такой ситуации вводим еще один тестовый набор (№5): *nmax*=10, *eps = 0.1E-10,* *x0 =-1, x1 =-0.5.* На рисунке 5 видно, что точность не достигнута за 10 итераций. В качестве результата будет выведено значение *x10=-0,8241323.*



Рис.4. Результаты для тестового варианта №5

* **Разработка алгоритма**

На рисунке 4 в виде UML-диаграммы деятельности с дорожками представлен алгоритм, реализующий метод секущих. Рисунок выполнен в среде MS VISIO.



Рис.5. Алгоритм метода секущих

Код программы

#include <iostream>

#include <cmath>

using namespace std;

// Функция, для которой ищем корень

double func(double x)

{

// Пример уравнения

return (x \* x) - cos(x);

}

// Функция для метода секущих

double MetodSec(double x0, double x1, int nmax, double e)

{

double metodsec; // Здесь будет храниться значение для метода секущих

// Печать заголовка таблицы

cout << "Итерация\tx0\t\t\tx1\t\t\tx\_next\t\t\tТочность" << endl;

for (int i = 0; i < nmax; i++)

{

double f0 = func(x0);

double f1 = func(x1);

// Проверка на малую разность значений функции

if (fabs(f1 - f0) < 1e-10)

{

cout << "Малое значение функции. Результат может быть не точным." << endl;

return x1;

}

// Формула метода секущих

metodsec = x1 - f1 \* (x1 - x0) / (f1 - f0);

// Вывод текущей итерации

cout << i + 1 << "\t\t" << x0 << "\t\t" << x1 << "\t\t" << metodsec << "\t\t" << fabs(metodsec - x1) << endl;

// Если достигнута точность, завершаем

if (fabs(metodsec - x1) < e)

{

cout << "\nРешение найдено за " << i + 1 << " шагов" << endl;

return metodsec;

}

// Обновляем значения для следующей итерации

x1 = metodsec;

}

cout << "Метод не сошелся после " << nmax << " шагов" << endl;

return metodsec;

}

int main()

{

setlocale(LC\_ALL, "rus");

while (true) // Бесконечный цикл для повторного использования программы

{

// Переменные для ввода данных

double x0, x1; // Начальные приближения

int nmax; // Количество итераций

double e; // Точность

// Ввод данных от пользователя

cout << "Введите х0: ";

cin >> x0;

cout << "Введите х1: ";

cin >> x1;

cout << "Введите количество шагов: ";

cin >> nmax;

cout << "Введите точность е: ";

cin >> e;

cout << "Все данные введены, начинается расчет." << endl;

// Вызываем функцию для нахождения корня

double root = MetodSec(x0, x1, nmax, e);

cout << "\nПриближенное значение корня: " << root << endl;

// Спрашиваем пользователя, хочет ли он продолжить

char choice;

cout << "Хотите выполнить еще один расчет? (д/н): ";

cin >> choice;

// Если пользователь ввел 'н' (нет), выходим из цикла и завершаем программу

if (choice == 'н' || choice == 'Н')

{

cout << "Программа завершена." << endl;

break;

}

}

return 0;

}

#include <iostream>

#include <cmath>

using namespace std;

// Функции, для которой ищем корень

double func(double x)

{

// Здесь можно изменить уравнение

return (1/x) - (3.14\*cos(3.14\*x));

// Пример уравнения

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

}

// Функция для метода секущих

double MetodSec(double x0, double x1, int nmax, double e)

{

double metodsec; // Здесь будет храниться значение для метода секущих

// Печать заголовка таблицы

cout << "Итерация\tx0\t\t\tx1\t\t\tx\_next\t\t\tТочность" << endl;

for (int i = 0; i < nmax; i++)

{

double f0 = func(x0);

double f1 = func(x1);

// Проверка на малую разность значений функции

if (fabs(f1 - f0) < 1e-10)

{

cout << "Малое значение функции. Результат может быть не точным." << endl;

return x1;

}

// Формула метода секущих

metodsec = x1 - f1 \* (x1 - x0) / (f1 - f0);

// Вывод текущей итерации

cout << i + 1 << "\t\t" << x0 << "\t\t" << x1 << "\t\t" << metodsec << "\t\t" << fabs(metodsec - x1) << endl;

// Если достигнута точность, завершаем

if (fabs(metodsec - x1) < e)

{

cout << "\nРешение найдено за " << i+1 << " шагов" << endl;

return metodsec;

}

// Обновляем значения для следующей итерации

x1 = metodsec;

}

cout << "Метод не сошелся после " << nmax << " шагов" << endl;

return metodsec;

}

int main()

{

setlocale(LC\_ALL, "rus");

// Переменные для ввода данных

double x0, x1; // Начальные приближения

int nmax; // Количество итераций

double e; // Точность

cout << "Введите х0: ";

cin >> x0;

cout << "Введите х1: ";

cin >> x1;

cout << "Введите количество шагов: ";

cin >> nmax;

cout << "Введите точность е: ";

cin >> e;

cout << "Все данные введены, начинается расчет." << endl;

// Вызываем функцию

double root = MetodSec(x0, x1, nmax, e);

cout << "\nПриближенное значение корня: " << root << endl;

return 0;

}