Содержание

[Введение 4](#_Toc22546896)

[1 Описание метода решения задачи 5](#_Toc22546897)

[1.1 Корни уравнения 5](#_Toc22546898)

[1.2 Отделение корней 6](#_Toc22546899)

[1.3 Численное решение уравнения методом половинного деления 6](#_Toc22546900)

[2 Разработка структур данных 9](#_Toc22546901)

[3 Разработка алгоритмов 10](#_Toc22546902)

[4 Тестирование программы 11](#_Toc22546903)

[5 Разработка документации 12](#_Toc22546904)

[Заключение 13](#_Toc22546905)

[Библиографический список 14](#_Toc22546906)

[Приложение А. Листинг программы 15](#_Toc22546907)

Введение

Целью курсового проекта является разработке программной системы для решения математической задачи уточнения корней уравнения численными методами (методом деления пополам).

Задачами курсового проектирования является:

* изучение метода решения задачи;
* разработка алгоритма и структур данных;
* разработка программы, позволяющей решить задачу уточнения корней уравнения;
* выполнить контрольный расчет в ручном режиме;
* провести тестирование работы программы на контрольном примере;
* выполнить тестирование работы программы на нескольких тестовых примерах;
* разработать документацию к программе.

Пояснительная записка к курсовому проекту состоит из следующих разделов:

* введение;
* описание метода решения задачи;
* разработка структур данных;
* разработка алгоритмов;
* тестирование программы;
* разработка документации;
* заключение;
* библиографический список;
* приложение (листинг программных модулей).

# Описание метода решения задачи

## Корни уравнения

Функция называется алгебраической, если для получения ее числового значения по данному значению аргумента х требуется выполнить арифметические операции и возведение в степень с рациональным показателем.

Если в запись уравнения входят только алгебраические функции, то уравнение называется алгебраическим.

Алгебраическое уравнение всегда может быть приведено к виду:

где .

Все неалгебраические функции: показательная *ах*, логарифмическая *logax*, тригонометрические sin *x*, cos *x*, tg *x*, ctg *x* и обратные тригонометрические arcsin *x*, arcos *x*, arctg *x*, arcctg *x* называются трансцендентными.

Если в запись уравнения входят трансцендентные функции, то уравнение называется трансцендентным, например, tg *x* = *ax*.

Решение уравнения с одним неизвестным *х* заключается в отыскании корней, то есть тех значений *х*, которые обращают уравнение в тождество.

В общем случае для уравнения отсутствуют аналитические формулы, определяющие его корни.

Задача отыскания корней сводится к нахождению всех точек *xi* пересечения графика функции *f(x)* с осью *x* (см. рисунок 1). Из рисунка видно, что число точек пересечения графика функции с осью *x* может быть несколько. Поэтому в качестве первого шага при решении любого уравнения проводят отделение его корней.

Это означает, что ось *x* разбивают на такие отрезки, что в каждом из них содержится только один корень уравнения. После этого следует уточнить положение каждого корня в пределах допустимой погрешности.

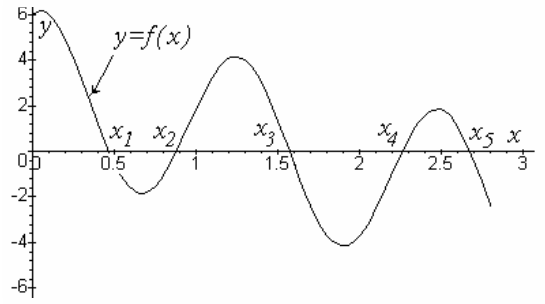


Рисунок 1 – Геометрическая интерпретация корней уравнения f(x)=0

## Отделение корней

Для отделения корней полезна следующая теорема: если непрерывная функция f(x) принимает значения разных знаков на концах отрезка [a, b], т.е. , то внутри этого отрезка содержится, по меньшей мере, один корень уравнения . На основе этой же теоремы реализуется самый простой и надежный метод численного определения корней уравнений: метод половинного деления.

Процесс отделения корней начинается с установления знаков *f(x)* в граничных точках интересующего нас отрезка определения переменной *x*: *x = a* и *x = b*.

Затем определяются знаки *f(x)* в ряде промежуточных точек x = α1, α2..., выбор которых должен учитывать особенности функции *f(x)*.

Если окажется, что *f(αi)f(αi+1) < 0*, то в интервале (αi, αi+1) есть корень уравнения . Необходимо убедиться, является ли этот корень единственным на данном интервале. Для отделения корней практически достаточно провести процесс половинного деления, последовательно деля исходный отрезок [a, b] на 2, 4, 8 и т. д. равных частей и определяя знаки *f(x)* в точках деления.

## Численное решение уравнения методом половинного деления

Предположим, что процесс отделения корней проведен и на отрезке [a, b] находится ровно один корень ξ уравнения . Необходимо определить его положение с погрешностью ε.

Метод половинного деления заключается в следующем (рисунок 2).

Сначала определяем середину с отрезка [a, b] c = (a+b)/2 и вычисляем значение функции f(c). Далее делаем выбор, какую из двух частей взять для уточнения корня. Очевидно, что корень будет находиться в той половине исходного отрезка, на концах которой функция имеет разные знаки.

На рисунке 2 таким будет правый отрезок – отрезок [a, c]. Для очередного шага уточнения положения корня отрезок [c, b] из рассмотрения исключаем, а с отрезком [a, c] продолжаем процесс деления, как и с первоначальным отрезком [a, b], формально переприсваивая новому значению *b* значение *c*. Если же реализуется ситуация, когда функция имеет разные знаки на концах отрезка [c, b], то из рассмотрения следует исключить отрезок [a, c], формально переприсваивая новому значению *а* значение *c*.

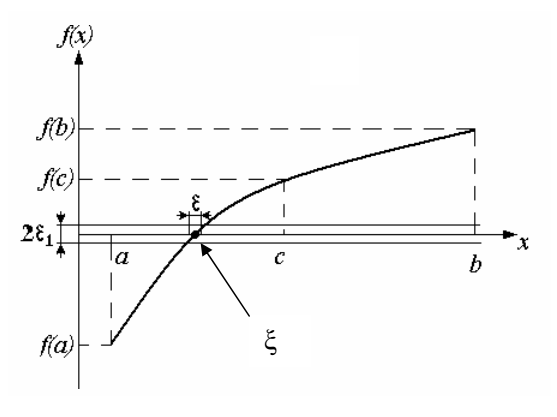


Рисунок 2 – Графическая интерпретация метода половинного деления

В результате мы получим последовательность вложенных друг в друга отрезков все уменьшающейся длины: [a1, b1], [a2, b2], ... [an, bn]. Этот повторяющийся (итерационный) процесс будем продолжать до тех пор, пока длина отрезка [an, bn] не станет меньше заданной погрешности ε вычислений.

Тогда искомый корень

ξ ≈ an ≈ bn ≈ (an + bn)/2.

Следует учитывать, что функция *f(x)* вычисляется с некоторой абсолютной погрешностью ε1. Вблизи корня значения функции *f(x)* малы по абсолютной величине и могут оказаться сравнимыми с погрешностью ее вычисления. Другими словами, при подходе к корню мы можем попасть в “полосу шумов” 2ε1 (рисунок 2) и дальнейшее уточнение корня становится бессмысленным. Поэтому надо задать ширину “полосы шумов” и прекратить итерационный процесс при попадании в нее. Также необходимо иметь в виду, что при уменьшении длины интервала [an, bn] увеличивается погрешность вычисления его длины an−bn за счет вычитания двух близких чисел.

Метод половинного деления обладает довольно большой скоростью сходимости. Так как за каждую итерацию интервал, где расположен корень, уменьшается в два раза, то через *n* итераций длина интервала будет равна (b−a)/2n. За 10 итераций интервал уменьшится в 210 ≈1024 ≈103 раз, а за 20 итераций – в 220≈106 раз.

# Разработка структур данных

Для решения задачи необходимо определить типы данных, которые будут использоваться для хранения исходных и промежуточных данных, а также результатов вычислений.

Для реализации подсчета числа итераций необходимы данные целого типа. Язык *Free Pascal* предоставляет возможность использования следующих целочисленных типов (Таблица 1):

Таблица 1 – Целые типы данных

|  |  |  |
| --- | --- | --- |
| **Тип** | **Диапазон значений** | **Размер, байт** |
| Byte | 0 .. 255 | 1 |
| Shortint | -128 .. 127 | 1 |
| Smallint | -32768 .. 32767 | 2 |
| Word | 0 .. 65535 | 2 |
| Integer | smallint или longint | 2 или 4 |
| Cardinal | longword | 4 |
| Longint | -2147483648 .. 2147483647 | 4 |
| Longword | 0..4294967295 | 4 |
| Int64 | -9223372036854775808 .. 9223372036854775807 | 8 |
| QWord | 0 .. 18446744073709551615 | 8 |

Диапазона значений типов *Byte*, *ShortInt* может не хватить для подсчета количества итераций. Диапазон значений типов *Cardinal*, *Longint*, *Longword*, *Int64*, *QWord* явно избыточен для решения поставленной задачи. Количество итераций не может быть отрицательным, поэтому тип *Word* предпочтительнее, чем *Smallint*.

Для реализации математических вычислений необходимы данные вещественного типа. Язык *Free Pascal* предоставляет возможность использования вещественных типов данных:

Таблица 2 – Вещественные типы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Диапазон значений** | **Значимых разрядов** | **Размер, байт** |
| Real | Зависит от платформы | Зависит от платформы | 4 или 8 |
| Single | 1.5E-45 .. 3.4E38 | 7-8 | 4 |
| Double | 5.0E-324 .. 1.7E308 | 15-16 | 8 |
| Extended | 1.9E-4932 .. 1.1E4932 | 19-20 | 10 |
| Comp | -2E64+1 .. 2E63-1 | 19-20 | 8 |
| Currency | -922337203685477.5808 | 922337203685477.5807 | 8 |

Тип данных *Real* зависит от платформы, и использовать его нежелательно. Тип *Currency* предназначен для операций с числами с фиксированной точностью (операции с денежными величинами). Тип *Comp* трактуется как вещественное число без дробной части.

Точность данных типа *Single* недостаточна для решения задач вычислительной математики, поэтому предпочтение следует отдать типам *Double* или *Extended*.

# Разработка алгоритмов

Процесс уточнения корня уравнения является итерационным процессом, поэтому за основу в качестве обобщенной схемы возьмем общую схему алгоритма итерационного процесса, представленную на рисунке (Рисунок 3.1).



Рисунок 3.1 – Общая схема итерационного процесса

Для метода половинного деления входными данными алгоритма будут:

* максимальное количество итераций *MaxIterCount*;
* границы отрезка [*Left*, *Right*], на котором был локализован единственный корень уравнения;
* точность локализации корня *EpsX*.
* погрешность вычисления корня *EpsY*.

Блок подготовки итерационного процесса заключается в том, что обнуляется счетчик итераций.

Тело цикла будет включать в себя следующие действия:

* определение середины отрезка: *Middle* = (*Left*+*Right*)/2;
* вычисление значения функции *f*(*Middle*), и выбор, какую из двух частей взять для уточнения корня.

Условием завершение цикла является:

* либо достижение максимального количества итераций (*IterCount* = *MaxIterCount*);
* либо достижение заданной ширины отрезка: ;
* либо достижение того, что значение функции в центре не превышает заданную точность |*f(Middle)*|<EpsY.

На рисунке (Рисунок 3.2) представлен алгоритм уточнения корня методом половинного деления.

Общий алгоритм (Рисунок 3.3) будет заключаться в следующем:

* ввод исходных данных (*Left*, *Right*, *MaxIterCount*, *EpsX*, *EpsY*);
* проверка исходных данных;
* выполнение уточнения корня методом половинного деления;
* вывод результатов.

Исходные данные корректные, если:

* число итераций положительно: *MaxIterCount* > 0;
* погрешности находятся в интервале допустимых значений 0<*EpsX*<0.1; 0<*EpsY*<0.1
* левая граница отрезка не превышает правую *Left* ≤ *Right*;
* на границах отрезка функция *f(x)* имеет значения разного знака: .



Рисунок 3.2 – Алгоритм уточнения корня методом половинного деления



Рисунок 3.3 – Общая схема алгоритма решения задачи

# Тестирование программы

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом

В [6] дается следующее определение тестирования: тестирование – это процесс выполнения программы с целью нахождения ошибок.

Для тестирования программы уточнения коря квадратного уравнения необходимо:

* выбрать уравнение, корень которого легко найти подбором и провести тестирования работы программы;
* выбрать уравнение, найти его корни при помощи другого программного средства и проверить найденное решение с тем, что будет выдано программой.

Проведем тестирование программы для уравнения . Корнями уравнения, очевидно, являются значения .

Зададим интервал локализации корня и выполним тестовый запуск программы:

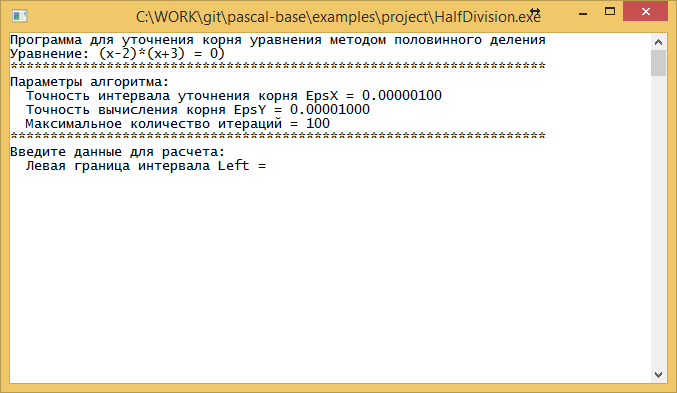


Рисунок 4.1 – Запуск программы

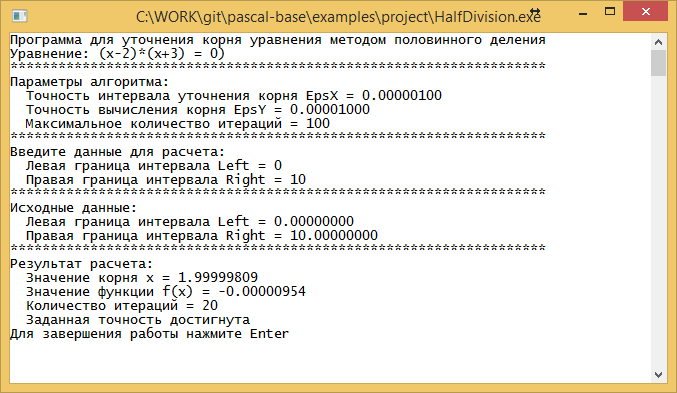


Рисунок 4.2 – Результаты выполнения программы

В результате запуска требуемый корень был найден с заданной точность. Повторим тестирование для второго корня.

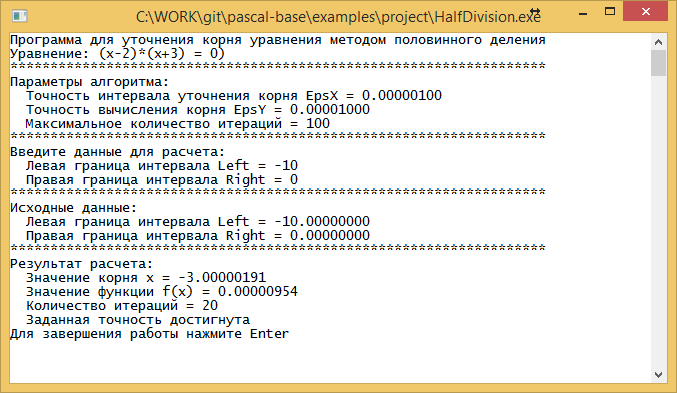


Рисунок 4.2 – Результаты выполнения программы для второго корня уравнения

В результате запуска второй корень также был найден с заданной точность.

Проведем тестирование для функции, корни которой сложно определить подбором. Для локализации интервалов корней необходимо построить график функции и таблицу ее значений.

Рисунок 4.4 – График функции



Рисунок 4.5 – Таблица значений функции

В таблице значений выделены интервалы локализации корня. Выберем интервал локализации корня и выполним тестовый запуск программы:

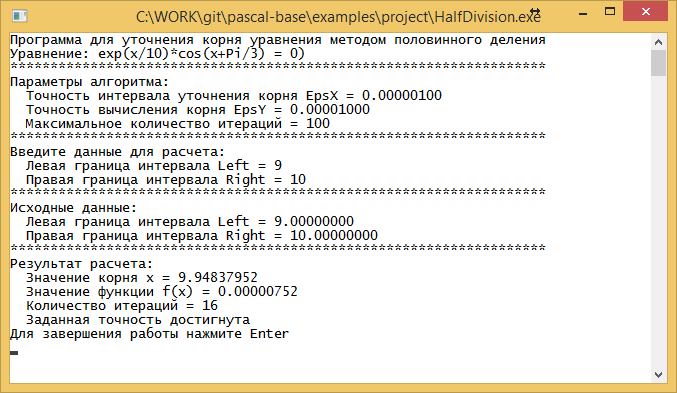


Рисунок 4.2 – Результаты выполнения программы для корня уравнения и =0

Результаты запуска показывают, что корень найден с заданной точностью.

По результатам испытаний считаем, что программа работает корректно.

# Разработка документации

В данном разделе приведено руководство программиста по использованию программы уточнения корня методом половинного деления *HalfDivision*.

## 5.1. Назначение и условия применения программы

Программа *HalfDivision* предназначена для уточнения корня уравнения методом половинного деления. Программа позволяет задавать интервал уточнения корня и может быть применена для решения любого уравнения с одной неизвестной.

Для выполнения программы необходимо наличие операционной системы *Windows*. Особые требования к составу периферийных устройств не предъявляются.

Персонал, использующий программу (программист) должен обладать практическими навыками работы с текстовым пользовательским интерфейсом.

## 5.2 Характеристика программы

Программа *HalfDivision* является независимым программным обеспечением и обеспечивает решение задач вычислительной математики – уточнение корня уравнения методом половинного деления.

Работа программы осуществляется в консольном режиме.

Размер исполняемого файла *HalfDivision*.*exe* – 95 129 байт.

## 5.3 Обращение к программе

Запуск программы осуществляется стандартным для операционной системы способом – двойным щелчком левой кнопки мыши на исполняемом файле *HalfDivision*.*exe* или выполнением команды в *HalfDivision* в командной строке в каталоге программы.

После запуска в окне консоли (Рисунок 5.1) появляется следующая информация:

* назначение программы;
* параметры алгоритма.

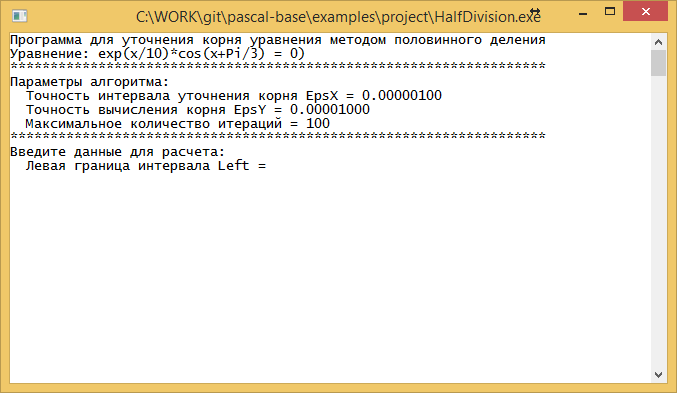


Рисунок 5.1 – Результат запуска программы HalfDivision

Пользователь должен ввести сначала левую границу интервала уточнения корня, затем – правую.

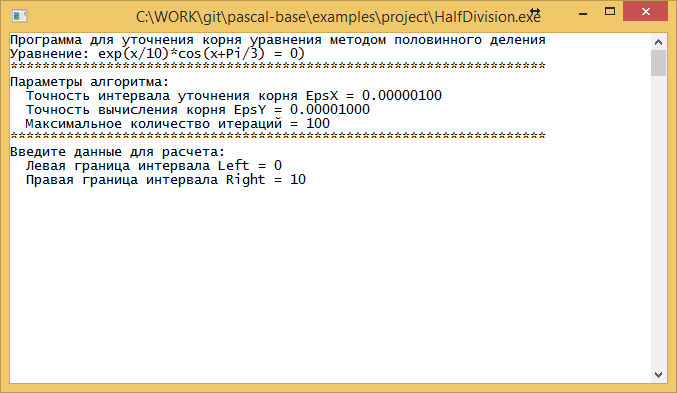


Рисунок 5.2 – Результат запуска программы HalfDivision

После ввода данных проверяется их корректность:

* функция должна иметь разный знак на границах интервала;
* левая граница должна быть меньше правой.

В случае ввода некорректных данных пользователю выводится сообщение об ошибке и запрашивается повторный ввод данных (Рисунок 5.3).

Если введены корректные данные, то осуществляется расчет – уточнение корня и осуществляется вывод исходных данных и результатов вычислений (Рисунок 5.4).

## 5.4 Входные и выходные данные

Входными данными программы являются:

* исходное уравнение;
* точность интервала уточнения корня;
* точность вычисления корня;
* максимальное количество итераций;
* левая и правая границы интервала.

Исходное уравнение задается в виде функции. Точность интервала уточнения корня, точность вычисления корня и максимальное количество итераций задаются при помощи констант. Левая и правая границы интервала уточнения корня вводятся пользователем в процессе работы программы.

Выходными данными программы являются:

* значение корня *x*;
* значение функции в точке *x;*
* количество итераций, за которое произошло уточнение корня.

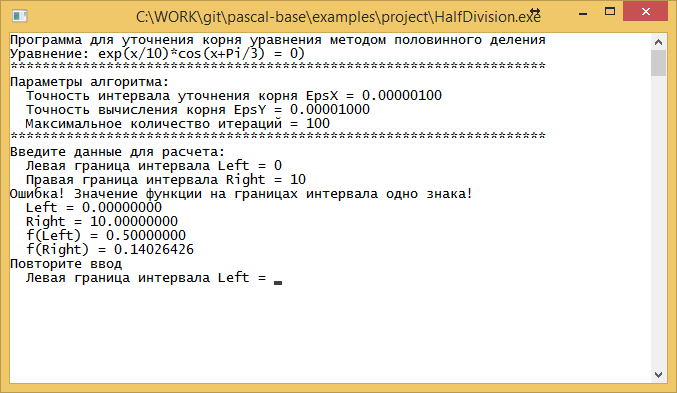


Рисунок 5.3 – Пример обработки некорректных данных

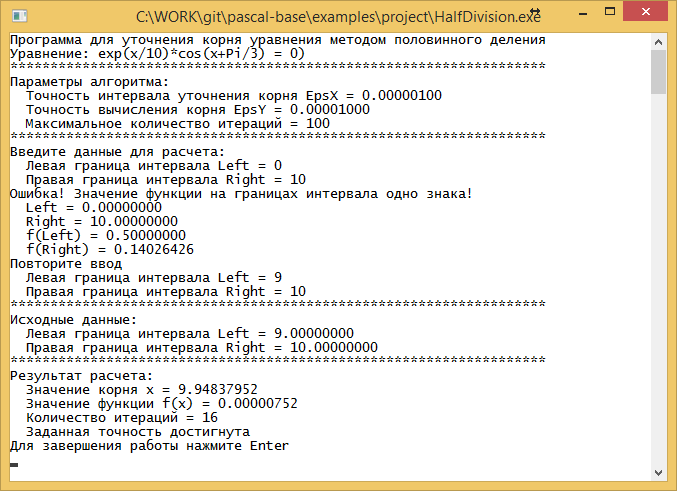


Рисунок 5.4 – Пример обработки некорректных данных

Заключение

Библиографический список

1. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
2. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
3. ГОСТ 19.104-78\* ЕСПД. Основные надписи.
4. ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам.
5. ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом.
6. ГОСТ 19.504-79\* ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
7. ГОСТ 19.604-78\* ЕСПД. Правила внесения изменения в программные документы, выполненные печатным способом.
8. Самарский А.А., Гулин А.В. Численные методы. - М.: 1989.
9. Калиткин Н.Н. Численные методы. - М.: Наука, 1978.
10. Демидович Б.П., Марон И.А. Основы вычислительной математики. - М.: 1970.
11. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах . - М.: Наука, 1972.
12. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран, Паскаль. – Томск: МП «Раско», 1991.
13. Майерс Г.[en] Надежность программного обеспечения. М: Мир, 1980

Приложение А. Листинг программы

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

{ }

{ Проект HalfDivision }

{ Copyright (c) 2019 ФГБОУ РГРТУ им. В.Ф. Уткина }

{ ФВТ/кафедр КТ }

{ }

{ Разработчик: ст. гр. 848 Иванов И.И. }

{ Модифицирован: 25 ноября 2019 }

{ }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

program HalfDivision;

//установка кодовой страницы для отображения русских символов

{$codepage utf8}

type

//тип данных для вычислений

TValue = Double;

const

EPS\_X = 1e-6; //точность интервала уточнения корня

EPS\_Y = 1e-5; //точность вычисления корня

MAX\_ITER\_COUNT = 100; //максимальное количество итераций

//функция для уточнения корня

// x - аргумент фукнкции

// результат - значение exp(x/10)\*cos(x+Pi/3)

function f(x: TValue): TValue;

begin

f := exp(x/10)\*cos(x+Pi/3);

end;

var

Left, Right: TValue; //границы интервала

Middle : TValue; //центр интервала

IterCount: word; // счетчик итераций

IsCorrect: boolean; // признак корректности ввода

begin

writeln('Программа для уточнения корня уравнения методом половинного деления');

writeln('Уравнение: exp(x/10)\*cos(x+Pi/3) = 0)');

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln('Параметры алгоритма:');

writeln(' Точность интервала уточнения корня EpsX = ', EPS\_X:0:8);

writeln(' Точность вычисления корня EpsY = ', EPS\_Y:0:8);

writeln(' Максимальное количество итераций = ', MAX\_ITER\_COUNT);

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln('Введите данные для расчета:');

IsCorrect := true;

repeat

write(' Левая граница интервала Left = ');

readln(Left);

write(' Правая граница интервала Right = ');

readln(Right);

if Left > Right then

begin

IsCorrect := false;

writeln('Ошибка! Правая граница меньше левой!');

writeln(' Left = ', Left:0:8);

writeln(' Right = ', Right:0:8);

writeln('Повторите ввод');

end;

if f(Left)\*f(Right) > 0 then

begin

IsCorrect := false;

writeln('Ошибка! Значение функции на границах интервала одного знака!');

writeln(' Left = ', Left:0:8);

writeln(' Right = ', Right:0:8);

writeln(' f(Left) = ', f(Left):0:8);

writeln(' f(Right) = ', f(Right):0:8);

writeln('Повторите ввод');

end;

until IsCorrect;

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln('Исходные данные:');

writeln(' Левая граница интервала Left = ', Left:0:8);

writeln(' Правая граница интервала Right = ', Right:0:8);

IterCount := 0;

repeat

//вычисляем середину интервала

Middle := Right/2 + Left/2;

//определяем, в какой части лежит корень

if f(Left)\*f(Middle)<=0 then

Right := Middle

else

Left := Middle;

Inc(IterCount);

until (IterCount = MAX\_ITER\_COUNT) //достигнуто максимальное количество итераций

or (abs(Right-Left)<EPS\_X) //или достугнута точность по X

or (abs(f(Middle))<EPS\_Y); //или достигнута точность по Y

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'); writeln('Результат расчета:');

writeln(' Значение корня x = ', Middle:0:8);

writeln(' Значение функции f(x) = ', f(Middle):0:8);

writeln(' Количество итераций = ', IterCount);

write(' Заданная точность ');

if IterCount = MAX\_ITER\_COUNT then

writeln('не достигнута')

else

write('достигнута');

writeln;

writeln('Для завершения работы нажмите Enter');

readln;

end.