Министерство образования и науки РФ
Государственное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

РЕКУРСИВНЫЕ ФУНКЦИИ И ПРОЦЕДУРЫ

Лабораторная работа № 3

по курсу «Программирование на ЯВУ»

Вариант № 4

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)

Проверил: _____ Сулимова В.В.
(подпись)

Тула 2011

Цель работы

Цель работы заключается в том, чтобы научиться описывать и использовать рекурсивные функции. Также необходимо написать программу, обрабатывающую данные рекурсивным способом.

Задание

Описать рекурсивную функцию ROOT(F,A,B,EPS), которая методом деления отрезка пополам находит с точностью EPS>0 корень уравнения F(X)=0 на отрезке [A,B] (считать, что A<B, F(A)<F(B) и F(X) - непрерывная и монотонная функция на отрезке [A,B].

Теоретическая справка

Схема алгоритма

На рисунке 1 представлена схема алгоритма ввода данных, расчета корня функции и его вывода.

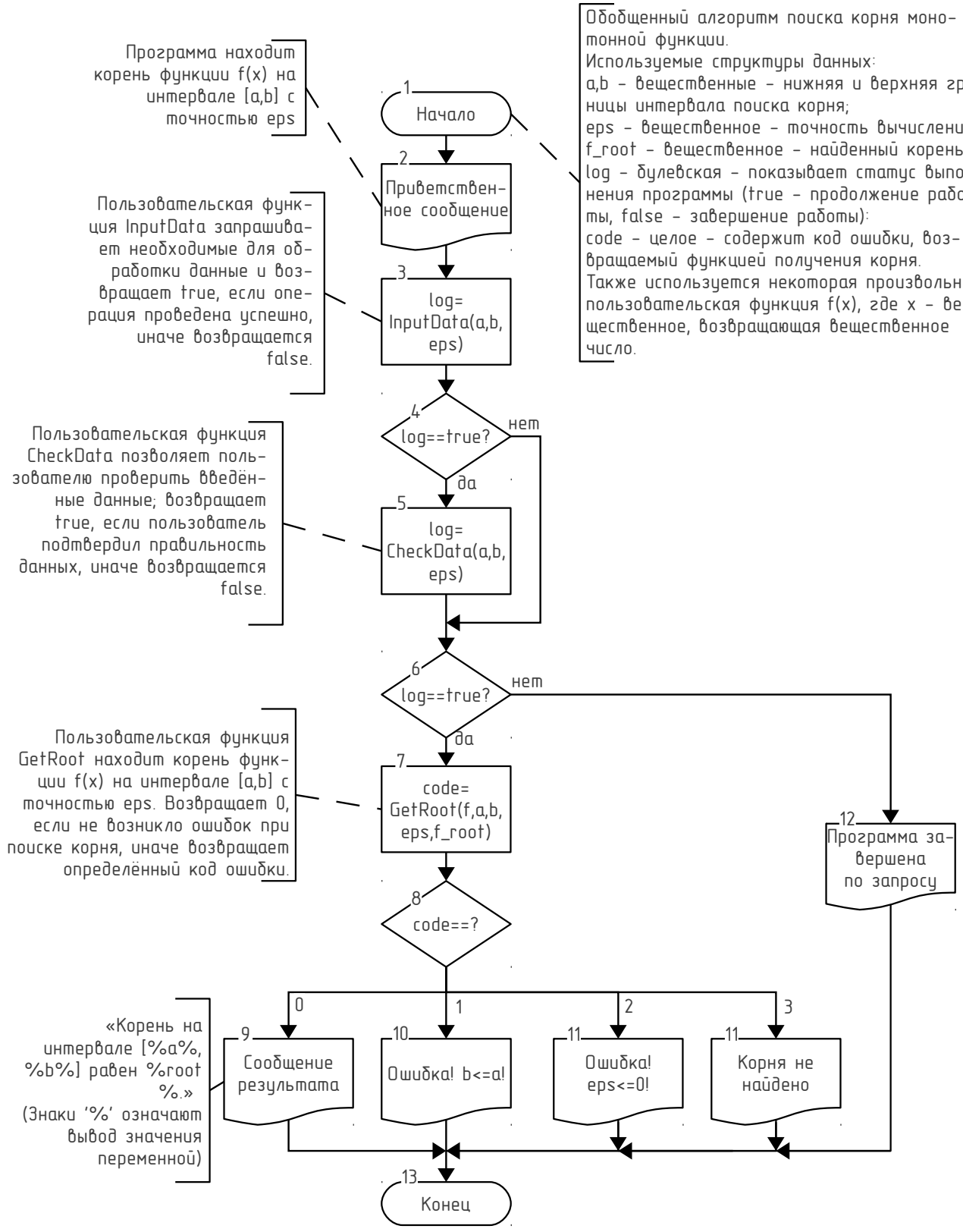

Рисунок 1 — Блок-схема обобщенного алгоритма поиска корня функции

На рисунках 2 и 3 представлена схема алгоритма ввода параметров расчета корня функции.

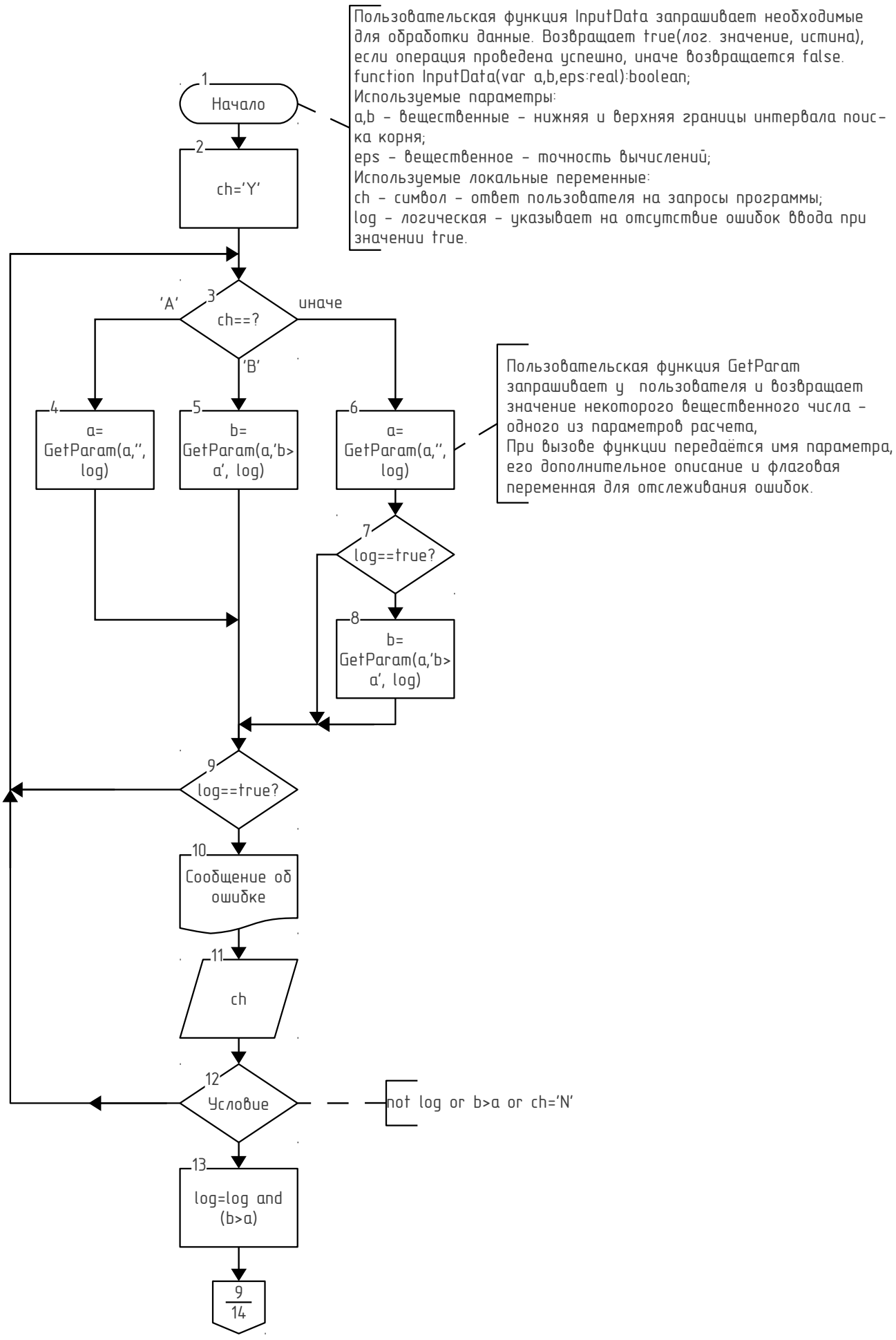

Рисунок 2 — Блок-схема алгоритма ввода параметров расчета корня функции(начало)

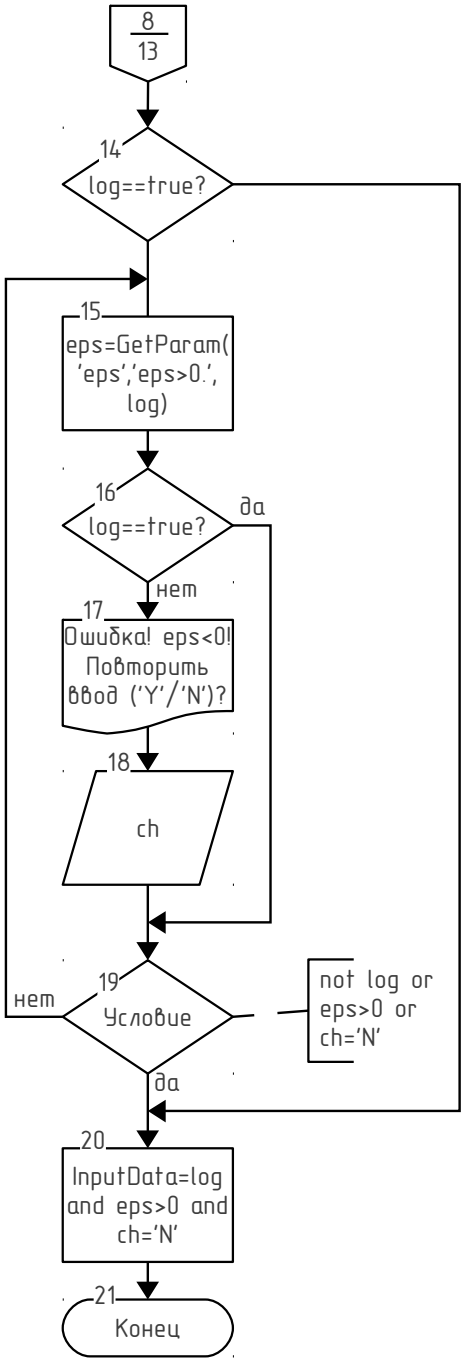

Рисунок 3 — Блок-схема алгоритма ввода параметров расчета корня функции(продолжение)

На рисунке 4 представлена блок-схема алгоритма запроса одного из параметров расчета.

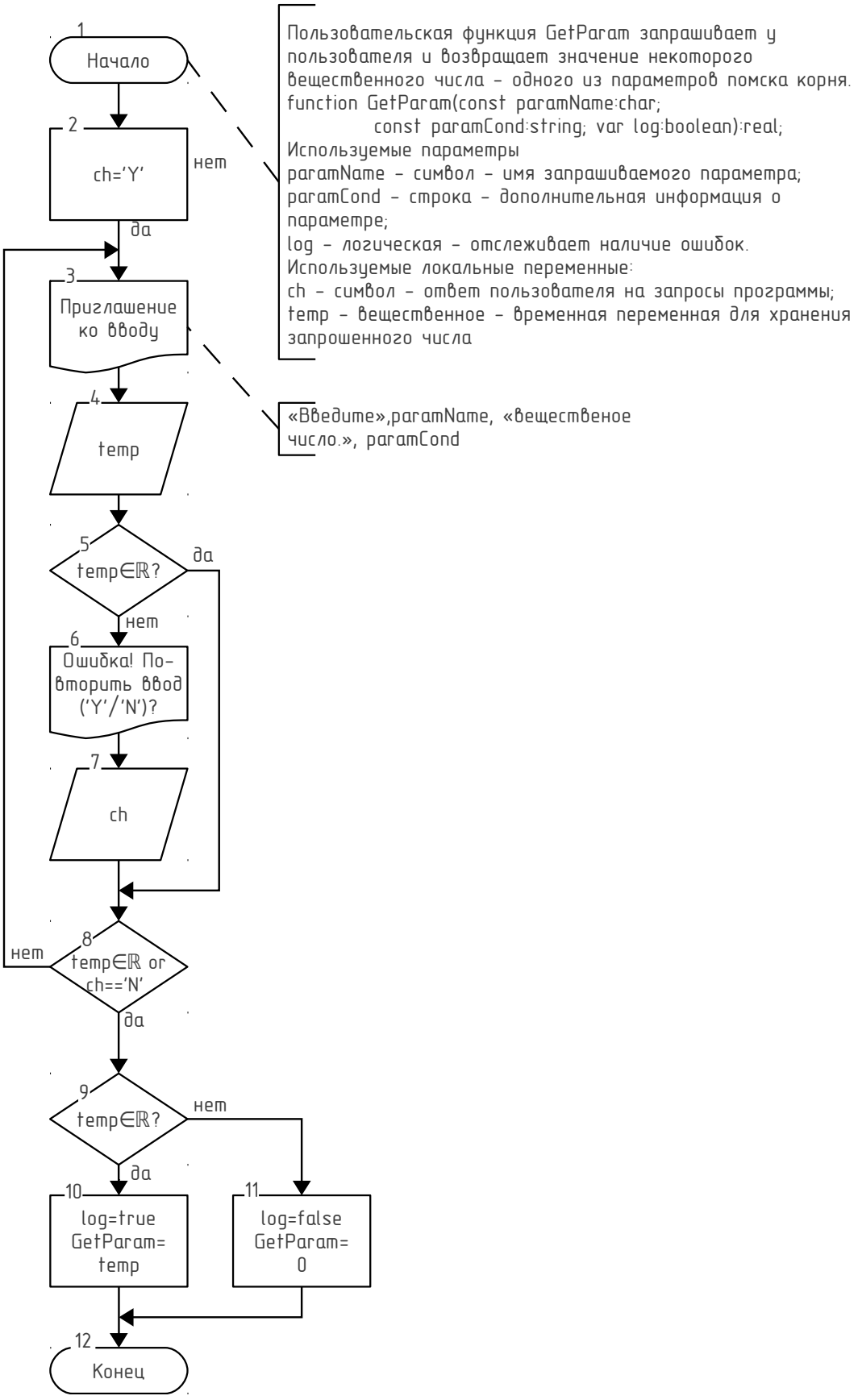

Рисунок 4 — Блок-схема алгоритма ввода произвольного параметра

На рисунке 5 представлена схема алгоритма организации проверки пользователем введённых данных.

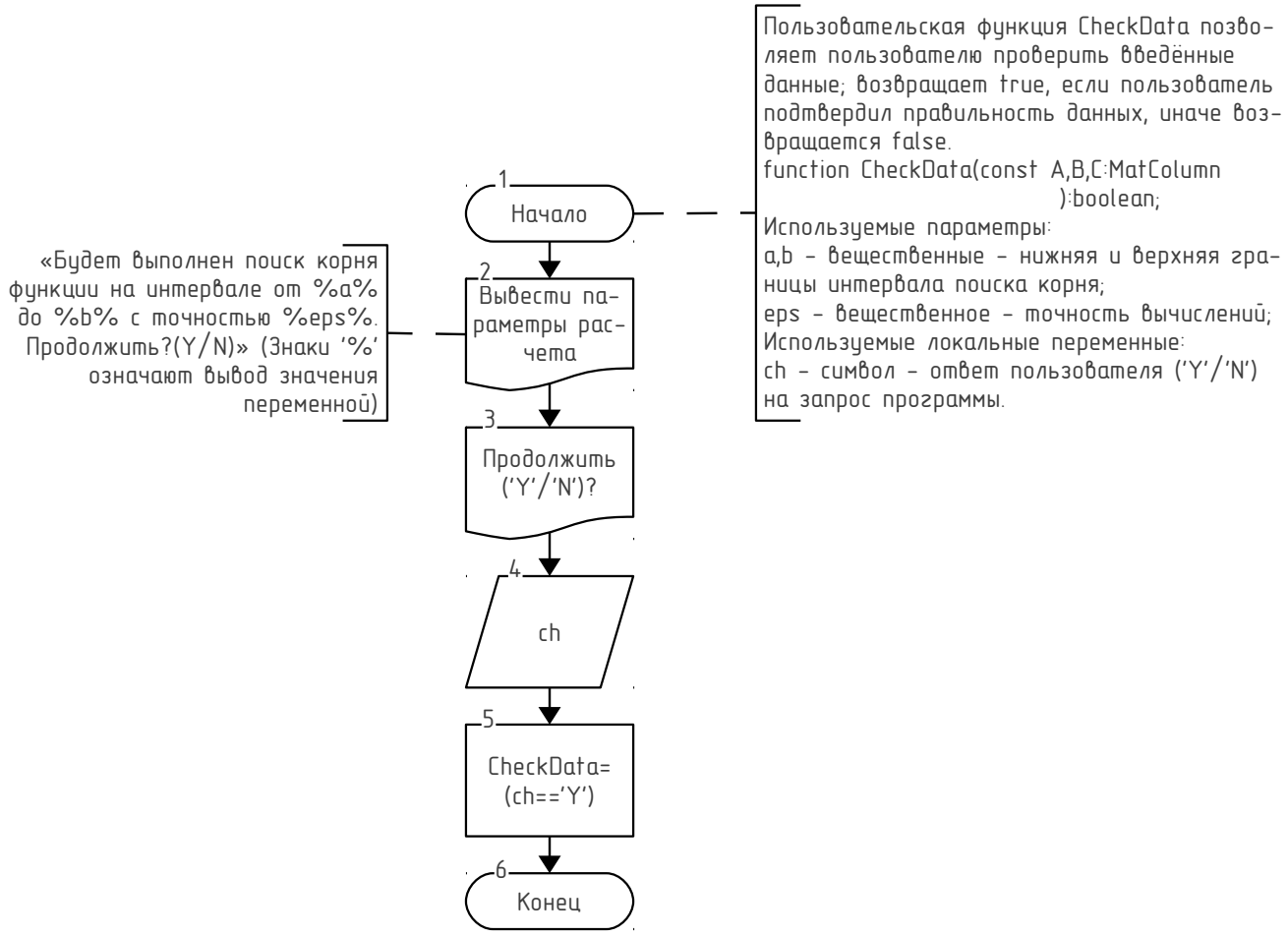

Рисунок 5 — Блок-схема алгоритма организации проверки пользователем введённых данных

На рисунке 6 представлена схема алгоритма проверки параметров для расчета корня функции и получения корня.

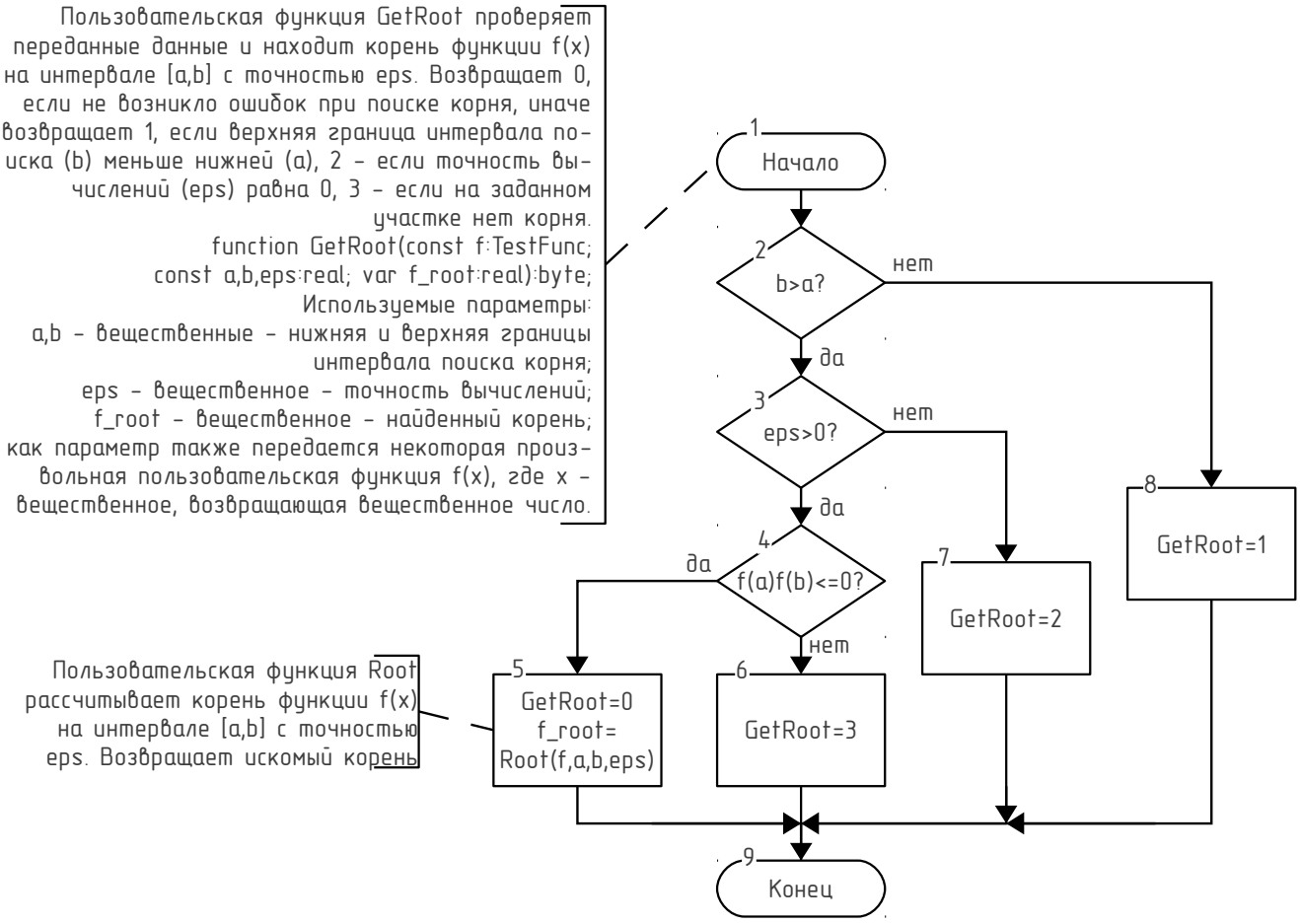

Рисунок 6 — Блок-схема алгоритма оценки данных и получения корня функции

На рисунке 7 представлена схема алгоритма расчета корня функции методом половинного деления.

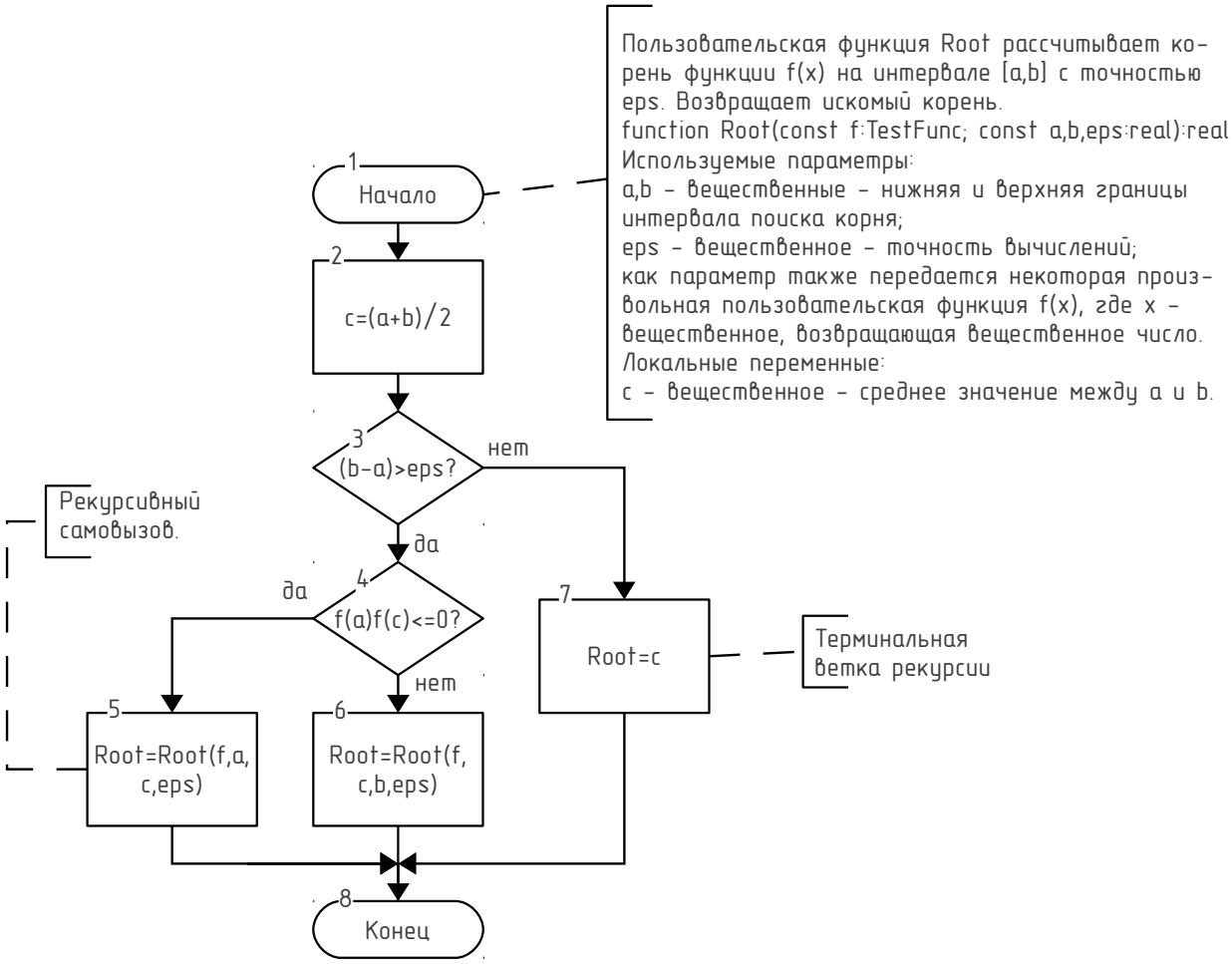

Рисунок 7 — Схема алгоритма расчета корня функции методом половинного деления

Инструкция пользователю

Данная программа позволяет найти корень монотонной функции на заданном промежутке.

Для работы программы необходимо ввести с клавиатуры сначала нижнюю границу отрезка поиска корня, затем верхнюю. Эти праметры - вещественные числа, причём верхняя граница обязательно строго больше нижней, в противном случае имеется возможность ввести заново верхнюю границу, или нижнюю границу,или обе сразу заново. Также есть возможность отказаться от повторного воода и завершить программу. Затем вводится требемая точность вычислений - вещественное число, большее нуля. После этого предоставляется возможность проверить введённые данные; если они корректны, для продолжения работы нужно ввести 'y', ввод любого другого значения приведёт к завершению работы программы с соответствующим сообщением.

В качестве результатов работы программа выведет искомый корень функции; иначе если функция не имеет корня на данном участке или она на нем не монотонна, будет выведено сообщение об отсутствии корня функции на данном промежутке.

Инструкция программисту

При создании программы вычисления корня монотонной функции на заданном отрезке были предприняты следующие действия.

В основной части программы определён процедурный тип TestFunc как function(const x:real):real, для передачи через этот тип произвольной функции как параметра подпрограмм.

Были введены структуры данных, описание которых представлено в таблице 1.

Таблица 1 - Структуры данных, используемые в в основной части программы оценки элементов массива

имя	тип	предназначение
a,b	real	Границы отрезка поиска корня функции
eps	real	Точность вычислений.
log	boolean	Указывает на отсутствие ошибок (значение true) ввода данных или желания пользователя прервать программу.
code	byte	Код ошибки вычисления корня функции; код 0 - нет ошибки

Кроме того, в процессе создания вышеуказанной программы были определены следующие подпрограммы:

1. Анализируемая функция f - любая функция, по типу подходящяя к TestFunc.

function f(const x:real):real;

В данной программе функция возвращает синус от x. В таблице 2 приведено описание её параметра.

Таблица 2 - Параметры-константы функции, для которой ищется корень

имя	тип	предназначение
x	real	Аргумент функции sin(x).

2. Функция GetParam запрашивает у пользователя произвольный параметр расчета - вещественное число. Возвращает запрошенный параметр; кроме того, в случае ошибки пользователя уведомляет об этом основную программу с помощью переменной log.

function GetParam(const paramName, paramCond:string;var log:boolean):real;

В теле функции в цикле с постусловием происходит запрос у пользователя вещественного числа, также выводится дополнительная информация о параметре. Запрошенное значение сначала считывается в буфер, потом производится попытка его извлечения (функцией val). Если попытка была успешной, функция завершает работу и передает указанное значение и значение log, равное true . Если нет, то производится запрос пользователя о продолжении работы. Если пользователь ответил 'N', функция завершает работу и возвращает значение log, равное false. Иначе операция повторяется до достижения двух вышеуказанных условий.

Используемые функцией параметры-константы приведены в таблице 3; параметры-переменные - в таблице 4, локальные переменные - в таблице 5.

Таблица 3 - Параметры-константы функции ввода произвольного параметра расчета корня

имя	**тип**	**предназначение**
paramName	string	Имя запрашиваемого параметра.
paramCond	string	Дополнительная информация о запрашиваемом параметре.

Таблица 4 - Параметры-переменные функции ввода произвольного параметра расчета корня

имя	**тип**	**предназначение**
log	boolean	Указывает на желание пользователя продолжить работу программы(при значении true, false - завершение программы)

Таблица 5 - Локальные переменные функции ввода произвольного параметра расчета корня

имя	**тип**	**предназначение**
ch	char	Содержит ответ пользователя на запросы программы о повторении ввода данных.
buf	string	Буфер, который содержит значения в строковом формате на случай ошибочного ввода параметра.
code	integer	Код ошибки извлечения значения из буфера;код 0 - нет ошибки.
temp	real	Временная переменная для хранения запрошенного параметра.

3. Функция InputData вводит параметры расчёта - границы отрезка поиска корня и точность вычислений; возвращает true, если не было ошибки ввода, иначе

возвращается false. Функция InputData использует функцию GetParam, поэтому она должна быть глобальной по отношению к InputData.

function InputData(var a,b,eps:real):boolean;

В цикле с постусловием происходит запрос параметров следующим образом:

Перед входом в цикл значение переменной ch инициализируется как 'Y'. Уже в цикле происходит ветвление с помощью оператора case по переменной ch: если ch равно 'A', то происходит запрос параметра a, если 'B', то запрос параметра b, иначе запрашиваются оба значения. Если b оказывается больше a, пользователю предлагается закончить работу (ответом 'N') или ввести заново значения указанным выше образом. Затем, если не возникло ошибок при вводе, запрашивается значение eps до тех пор, пока оно не будет больше 0 или пользователь не откажется от ввода. Вся процедура повторяется, пока все параметры не будут введены правильно, и тогда будет возвращено значение true, или пользователь не откажется от продолжения работы (и тогда возвращается false).

Используемые функцией параметры-переменные приведены в таблице 6; локальные переменные - в таблице 7.

Таблица 6 - Параметры-переменные функции ввода данных для расчета корня

имя	**тип**	**предназначение**
a,b	real	Границы отрезка поиска корня функции
eps	real	Точность вычислений.

Таблица 7 - Локальные переменные функции ввода данных для расчета корня

имя	**тип**	**предназначение**
ch	char	Содержит ответ пользователя на запрос программы о правильности данных.
log	boolean	Указывает на желание пользователя продолжить работу программы(при значении true, false - завершение программы)

4. Функция CheckData позволяет организовать проверку пользователем введенных им ранее значений. Возвращает true, если пользователь подтвердил правильность данных.

function CheckData(const a,b,eps:real):boolean;

Функция выводит значения параметров и спрашивает пользователя о продолжении работы. Если он отвечает 'Y', то функция возвращает true, иначе возвращается false.

Используемые функцией параметры-константы приведены в таблице 8; локальные переменные - в таблице 9.

Таблица 8 - Параметры-константы функции проверки пользователем введенных значений

имя	**тип**	**предназначение**
a,b	real	Границы отрезка поиска корня функции
eps	real	Точность вычислений.

Таблица 9 - Локальные переменные функции проверки пользователем введенных значений

имя	**тип**	**предназначение**
ch	char	Содержит ответ пользователя на запрос программы о правильности данных.

5. Функция GetRoot проверяет переданные данные и получает корень функции f с помощью функции Root (определена локально по отношению к GetRoot). Возвращает отпределённый код ошибки:

0 - ошибки нет;

1 - верхняя граница отрезка поиска корня меньше нижней;

2 - точность вычислений меньше или равна 0;

3 - корня на заданном участке не найдено.

function GetRoot(const f:TestFunc; const a,b,eps:real;var f_root:real):byte;

Функция проверяет вышеуказанные условия, и если они выполняются, возвращает соответственный код ошибки. Если возвращается код 0, то через переменную f_root возвращается найденный корень.

Используемые функцией параметры-константы приведены в таблице 10, параметры-переменные - в таблице 11.

Таблица 10 - Параметры-константы функции проверки данных и получения корня

имя	**тип**	**предназначение**
a,b	real	Границы отрезка поиска корня функции
eps	real	Точность вычислений.

f	TestFunc	Анализируемая функция

Таблица 11 - Параметры-переменные функции проверки данных и получения корня

имя	тип	предназначение
f_root	real	Корень анализируемой функции

5.1 Функция Root расчитывает корень функции f. Возвращает искомый корень. Является рекурсивной; поэтому проверки передаваемых значений в целях повышения производительности в ней не производятся, и поэтому Root определена локальной функцией по отношению к GetRoot.

function Root(const f:TestFunc; const a,b,eps:real):real;

Сначала находится среднее значение между a и b и записывает его в c. Если разница между a и b меньше eps, то возвращается c; иначе если f(a)*f(c)<=0, то возвращается значение от рекурсивного вызова Root(f,a,c,eps), иначе возвращается значение от вызова Root(f,c,b,eps).

Используемые функцией параметры-константы приведены в таблице 12; локальные переменные - в таблице 13.

Таблица 12 - Параметры-константы функции расчета корня

имя	тип	предназначение
a,b	real	Границы отрезка поиска корня функции
eps	real	Точность вычислений.
f	TestFunc	Анализируемая функция

Таблица 13 - Локальные переменные функции расчета корня

имя	тип	предназначение
c	real	Среднее значение между a и b.

Текст программы

Ниже представлен текст программы, написанной на языке Turbo Pascal 7, которая находит корень функции sin(x) на заданном промежутке и с заданной точностью.

```
{-------ПРОГРАММА ПОИСКА КОРНЯ ФУНКЦИИ------}
program FunctionRoot;
type TestFunc=function(const x:real):real;
{$F+}
{АНАЛИЗИРУЕМАЯ ФУНКЦИЯ}
function f(const x:real):real;
begin
    f:=sin(x);
end;
{$F-}
{------ФУНКЦИЯ ВВОДА ВЕЩЕСТВЕННОГО ПАРАМЕТРА-----}
{Параметры:
paramName - имя запрашиваемого параметра;
paramCond - дополнительная информация о параметре;
log - переменная состояния}
function GetParam(const paramName,paramCond:string;var log:boolean):real;
var temp:real; buf:string; code:integer; ch:char;
begin
    ch:='Y'; temp:=0; {инициализация переменных}
    repeat
        WriteLn('Введите ',paramName,' - вещественное число. ',paramCond);
        Write(paramName,':');ReadLn(buf); Val(buf,temp,code);
        if code<>0 then begin {введено не вещественное число?}
            WriteLn('Недопустимое значение ',paramName,'.');
            WriteLn('Повторить ввод?(y/n)');
            ReadLn(ch);
        end;
    until (Upcase(ch)='N') or (code=0);
    {пока пользователь не отказался или число некорректное}
    log:=code=0;{число введено корректно?}
    GetParam:=temp;{вернём значение}
end;

{--------ФУНКЦИЯ ВВОДА ДАННЫХ-------}
{Параметры:
a,b - границы отрезка поиска корня;
eps - точность вычислений;
log - переменная состояния}
function InputData(var a,b,eps:real):boolean;
```

```
var log:boolean; ch:char;
begin
    ch:='Y'; {инициализация так и только так! см. ниже, почему}
    repeat
        case UpCase(ch) of
            'A':a:=GetParam('a','',log);
            'B':b:=GetParam('b','b>a.',log);
            else begin
                a:=GetParam('a','',log);
                if log then
                    b:=GetParam('b','b>a.',log);
            end;
        end;
        if log and not (b>a) then begin
            WriteLn('Ошибка! b<=a!');
            WriteLn('Повторить ввод?');
            WriteLn('Y - ввести a и b;');
            WriteLn('A - ввести только a;');
            WriteLn('B - ввести только b;');
            WriteLn('N - отмена.');
            ReadLn(ch);
        end;
    until not(log) or (b>a) or (UpCase(ch)='N');
    log:=log and (b>a);
    if log then
        repeat
            eps:=GetParam('eps','eps>0.',log);
            if log and not (eps>0) then begin
                WriteLn('Ошибка! eps<=0!');
                WriteLn('Повторить ввод?(Y/N)');
                ReadLn(ch);
            end;
        until not(log) or (eps>0) or (UpCase(ch)='N');
    log:=log and (eps>0);
    InputData:=log and (ch<>'N');
end;

{-------ФУНКЦИЯ ПРОВЕРКИ ДАННЫХ------}
{Параметры:
a,b - границы отрезка поиска корня;
eps - точность вычислений}
function CheckData(const a,b,eps:real):boolean;
var ch:char;
begin
```

```
    WriteLn('Будет выполнен поиск корня');
    WriteLn(' на интервале от ',a:1:4,' до ',b:1:4,' с точностью ',eps:1:8);
    WriteLn('Продолжить?(Y/N)');
    ReadLn(ch);
    CheckData:=UpCase(ch)='Y';
end;

{------ФУНКЦИЯ ПОЛУЧЕНИЯ КОРНЯ------}
{Параметры:
f - анализируемая функция;
a,b - границы отрезка поиска корня;
eps - точность вычислений;
f_root - найденный корень}
function GetRoot(const f:TestFunc; const a,b,eps:real;var f_root:real):byte;

{------ФУНКЦИЯ РАСЧЕТА КОРНЯ------}
{Параметры:
f - анализируемая функция;
a,b - границы отрезка поиска корня;
eps - точность вычислений;}
function Root(const f:TestFunc; const a,b,eps:real):real;
var c:real;
begin
    c:=(a+b)/2;
    if b-a>eps then
        if f(c)*f(a)<=0 then
            Root:=Root(f,a,c,eps)
        else
            Root:=Root(f,c,b,eps)
    else
        Root:=c;
end;
begin
    if not(b>a) then
        GetRoot:=1
    else
        if not(eps>0) then
            GetRoot:=2
        else
            if f(a)*f(b)>0 then
                GetRoot:=3
            else begin
                GetRoot:=0;
                f_root:=Root(f,a,b,eps);
```

```
            end;
end;

{------ОСНОВНАЯ ПРОГРАММА-------}
{a,b - границы отрезка поиска корня;
eps - точность вычислений;
log - переменная состояния;
code - код ошибки поиска корня}
const func_name='sin(x)';
var a,b,eps,f_root:real;log:boolean; code:byte;
BEGIN
    WriteLn('Программа находит корень функции ',func_name,' на отрезке [a,b] с
                точностью eps.');
    log:=InputData(a,b,eps);
    if log then
        log:=CheckData(a,b,eps);
    if log then begin
        code:=GetRoot(f,a,b,eps,f_root);
        case code of
            0:WriteLn('Корень на промежутке [',a:1:4,',',b:1:4,'] равен '
                            ,f_root:1:8);
            1:WriteLn('Верхняя граница интервала поиска корня меньше нижней...');
            2:WriteLn('Точность вычислений меньше/равна 0!');
            3:WriteLn('Корня на промежутке [',a:1:4,',',b:1:4,'] не найдено.');
        end
    end
    else
        WriteLn('Программа завершена пользователем...');
END.
```

Тестовые примеры

На рисунке 8 представлен пример работы программы нахождения корня функции для функции y=sin(x) на отрезке [3,4] с точностью 0.00001.

```
Программа находит корень функции sin(x) на отрезке [a,b] с точностью eps.
Введите a - вещественное число.
a:3
Введите b - вещественное число. b>a.
b:4
Введите eps - вещественное число. eps>0.
eps:0.00001
Будет выполнен поиск корня
 на интервале от 3.0000 до 4.0000 с точностью 0.00001000
Продолжить?(Y/N)
y
Корень на промежутке [3.0000,4.0000] равен 3.14159012
```

Рисунок 8— Результат работы программы поиска корня функции

На рисунке 9 приведен результат работы программы при ошибках пользователя для функции sin(x).

```
Программа находит корень функции sin(x) на отрезке [a,b] с точностью eps.
Введите a - вещественное число.
a:3
Введите b - вещественное число. b>a.
b:3
Ошибка! b<=a!
Повторить ввод?
Y - ввести a и b;
A - ввести только a;
B - ввести только b;
N - отмена.
b
Введите b - вещественное число. b>a.
b:3
Ошибка! b<=a!
Повторить ввод?
Y - ввести a и b;
A - ввести только a;
B - ввести только b;
N - отмена.
a
Введите a - вещественное число.
a:4
Ошибка! b<=a!
Повторить ввод?
Y - ввести a и b;
A - ввести только a;
B - ввести только b;
N - отмена.
a
Введите a - вещественное число.
a:2
Введите eps - вещественное число. eps>0.
eps:0.0001
Будет выполнен поиск корня
 на интервале от 2.0000 до 3.0000 с точностью 0.00010000
Продолжить?(Y/N)
y
Корня на промежутке [2.0000,3.0000] не найдено.
```

Рисунок 9— Результат работы программы при ошибочных значениях

Вывод

В ходе выполнения данной лабораторной работы я научился использовать рекурсию при написании программ. Рекурсия представляет собой мощный инструмент, который позволяет изящно и компактно описать очень сложные для итеративной реализации алгоритмы. Однако минусами рекурсии является повышенный расход памяти и (иногда) меньшая понятность и наглядность. Поэтому если имеется очевидная реализация с помощью циклов, то лучше предпочесть последнее.