УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №1.3

по предмету

Основы алгоритмизации и программирования

Вариант 3

Выполнил:

Бетеня К.С.

Проверила:

Данилова Г.В.

Группа 351005

Минск 2023

Задание:

Вычислите значение кубического корня  с точностью EPS с использованием итерационной формулы Ньютона:

Y0=1

Yi= 1/3 (2Yi-1+ X/Y2i-1) (i=1,2,3,...)

Вычисления производить пока |Yi - Yi-1| не станет меньше EPS. Подсчитайте количество итераций, за которое достигается эта точность.

Код программы на **Delphi**:

Program Lab\_3;

Uses

System.SysUtils;

Const

MinEPS = 0; // minimum EPS

MaxEPS = 0.1; // maximum EPS

Var

X: Double = 0;

EPS: Double = MaxEPS;

Y: Double = 0; // current element

Y0: Double = 1; // previous element

Number: Integer = 0; // operation counter

Coefficient: Integer = 1; // stores the sign of the entered X

IsCorrect: Boolean = True;

Begin

// condition

Writeln(' The program calculates the value of the cube root ', #13#10,

'from the number X entered by a person.', #13#10,

'With accuracy up to the number EPS entered by the user.', #13#10,

#13#10,

'Restrictions X: No restrictions.', #13#10,

'Restrictions EPS: (0; 0.1).', #13#10, #13#10);

Repeat

Try

Begin

IsCorrect := True;

// input X

Writeln('Write the cube root of which number you want to find?');

Read(X);

// EPS input

Writeln('With what EPS must the calculations be made?');

Read(EPS);

// EPS check

// for EPS more than 0.1

If (EPS > MaxEPS) Then

Begin

// passing the error

Raise Exception.Create('EPS is too high. EPS must be less than

0.1.');

End

Else

// for EPS less than 0

If (EPS < MinEPS) Then

Begin

// passing the error

Raise Exception.Create('EPS cannot be less than 0');

End

Else

// for EPS = 0

If (EPS = MinEPS) Then

Begin

// passing the error

Raise Exception.Create('EPS cannot be 0');

End;

// display current information about EPS and X

Writeln(#13#10, 'Your X: ', FormatFloat('0.###',X),

#13#10, 'Your EPS: ', FormatFloat('0.###',EPS),

#13#10);

// checking the sign

If (X < 0) Then

Begin

// change of sign

Coefficient := -1;

X := -X;

End;

// conclusion

If (X = 0) Then

Begin

// at X = 0

Y := X;

Number := 1;

End

Else

// for numbers from -1 to 1, except zero

If (X < 1) Then

Begin

// first operation

Y := ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number := Number + 1;

// checking for the possibility of the next operation

If (Y0 - Y > EPS) Then

Begin

Y0 := Y;

Y := ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number := Number + 1;

// since the difference between A and A0 is small,

// then you can do a complete search of all the remaining

numbers

While (Y0 - Y > EPS) Do

Begin

Y0 := Y;

Y := ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number := Number + 1;

End;

End;

End

// for all numbers except the range from -1 to 1

Else

Begin

// first operation

Y := ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number := Number + 1;

// checking for the possibility of the next operation

If (Y - Y0 > EPS) Then

Begin

Y0 := Y;

Y := ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number := Number + 1;

// since the difference between A and A0 is small,

// then you can do a complete search of all the remaining

numbers

While (Y0 - Y > EPS) Do

Begin

Y0 := Y;

Y := ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number := Number + 1;

End;

End;

End;

// output of final information

// cube root output

Writeln('Cube root of ', FormatFloat('0.###',X), ' = ',

FormatFloat('0.###', Coefficient \* Y));

// number of operations to achieve accuracy

Writeln('Number of operations for which EPS was achieved - ',

Number);

End;

Except

// catching a mistake

On Error: Exception Do

Begin

Writeln(Error.Message);

End;

End;

Until (IsCorrect);

Readln;

Readln;

End.

Код программы на **C++**:

#include <iostream>

#include <iomanip>

int main()

{

const double MinEPS = 0; // minimum EPS

const double MaxEPS = 0.1; // maximum EPS

double EPS = MaxEPS, X = 0;

double Y = 0; // current element

double Y0 = 1; // previous element

int Number = 0; // operation counter

int Сoefficient = 1; // stores the sign of the entered X

bool IsCorrectX = false, IsCorrectEPS = false;

// formatted output

std::cout << std::setprecision(3) << std::fixed;

// condition

std::cout << " The program calculates the value of the cube root \n"

<< "from the number X entered by a person.\n"

<< "With accuracy up to the number EPS entered by the user.\n\n"

<< "Restrictions X: No restrictions.\n"

<< "Restrictions EPS: (0; 0.1).\n\n\n";

do

{

IsCorrectX = false;

// input X

std::cout << "Write the cube root of which number you want to

find?\n";

std::cin >> X;

// X check

if (std::cin.get() != '\n')

{

std::cout << "X entered incorrectly.\n";

IsCorrectX = true;

std::cin.clear();

while (std::cin.get() != '\n') {}

}

} while (IsCorrectX);

do {

IsCorrectEPS = false;

// EPS input

std::cout << "With what EPS must the calculations be made?\n";

std::cin >> EPS;

// EPS check

// for EPS more than 0.1

if (EPS > MaxEPS)

{

// error

std::cout << "EPS is too high. EPS must be less than 0.1.\n";

IsCorrectEPS = true;

std::cin.clear();

while (std::cin.get() != '\n') {}

}

// EPS = 0.1

if (EPS == MaxEPS)

{

// error

std::cout << "EPS must be less than 0.1.\n";

IsCorrectEPS = true;

std::cin.clear();

while (std::cin.get() != '\n') {}

}

// for EPS less than 0

else if (EPS < MinEPS)

{

// error

std::cout << "EPS cannot be less than 0.\n";

IsCorrectEPS = true;

std::cin.clear();

while (std::cin.get() != '\n') {}

}

// for EPS = 0

else if (EPS == MinEPS)

{

// error

std::cout << "EPS cannot be 0.\n";

IsCorrectEPS = true;

std::cin.clear();

while (std::cin.get() != '\n') {}

}

// special case

else if (std::cin.get() != '\n')

{

// error

std::cout << "EPS entered incorrectly.\n";

IsCorrectEPS = true;

std::cin.clear();

while (std::cin.get() != '\n') {}

}

} while (IsCorrectEPS);

// display current information about EPS and X

std::cout << "\n\nYour X: " << X << "\nYour EPS: " << EPS;

// checking the sign

if (X < 0)

{

// change of sign

Сoefficient = -1;

X = -X;

}

// conclusion

if (X == 0)

{

// at X = 0

Y = X;

Number = 1;

}

// for numbers from -1 to 1, except zero

else if (X < 1)

{

// first operation

Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number++;

// checking for the possibility of the next operation

if (Y0 - Y > EPS)

{

Y0 = Y;

Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number++;

// since the difference between A and A0 is small,

// then you can do a complete search of all the remaining

numbers

while (Y0 - Y > EPS)

{

Y0 = Y;

Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number++;

}

}

}

// for all numbers except the range from -1 to 1

else

{

// first operation

Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number++;

// checking for the possibility of the next operation

if (Y - Y0 > EPS)

{

Y0 = Y;

Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number++;

// since the difference between A and A0 is small,

// then you can do a complete search of all the remaining

numbers

while (Y0 - Y > EPS)

{

Y0 = Y;

Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;

Number++;

}

}

}

// output of final information

// cube root output

std::cout << "\nCube root of " << X << " = " << Сoefficient \* Y;

// number of operations to achieve accuracy

std::cout << "\nNumber of operations for which accuracy was achieved - " <<

Number;

std::cout << "\n\n";

return 0;

}

Код программы на **Java**:

import java.text.DecimalFormat;  
import java.util.Scanner;  
  
public class lab\_3  
{  
 public static void main(String[] args)  
 {  
 final double MinEPS = 0; // minimum EPS  
 final double MaxEPS = 0.1; // maximum EPS  
 double EPS = MaxEPS, X = 0;  
 double Y = 0; // current element  
 double Y0 = 1; // previous element  
 int Number = 0; // operation counter  
 int Coefficient = 1; // stores the sign of the entered X  
 boolean IsCorrect = true;  
  
 // condition  
 System.*out*.println(" The program calculates the value of the cube root

\n"  
 + "from the number X entered by a person.\n"  
 + "With accuracy up to the number EPS entered by the user.\n\n"  
 + "Restrictions X: No restrictions.\n"  
 + "Restrictions EPS: (0; 0,1).\n\n");  
 do  
 {  
 try {  
 DecimalFormat dF = new DecimalFormat( "0.###" );  
 Scanner in = new Scanner(System.*in*);  
 // input X  
 System.*out*.println("Write the cube root of which number you want

to find?");  
 X = in.nextDouble();  
  
 // EPS input  
 System.*out*.println("With what EPS must the calculations be

made?");  
 EPS = in.nextDouble();  
  
 // EPS check  
 // for EPS more than 0,1  
 if (EPS > MaxEPS)  
 {  
 //passing the error  
 throw new Exception("EPS is too high. EPS must be less than

0.1.");  
 }

// for EPS = 0,1  
 if (EPS == MaxEPS)  
 {  
 throw new Exception("EPS must be less than 0.1.");  
 }

// for EPS less than 0  
 else if (EPS < MinEPS)  
 {  
 //passing the error  
 throw new Exception("EPS cannot be less than 0");  
 }  
 // for EPS = 0  
 else if (EPS == MinEPS)  
 {  
 //passing the error  
 throw new Exception("EPS cannot be 0");  
 }  
  
 //display current information about EPS and X  
 System.*out*.println("\nYour X: " + dF.format(X) + "\nYour EPS: " +

dF.format(EPS));  
  
 // check the sign  
 if (X < 0)  
 {  
 // change of sign  
 Coefficient = -1;  
 X = -X;  
 }  
  
 // conclusion  
 if (X == 0)  
 {  
 // at X = 0  
 Y = X;  
 Number = 1;  
 }  
 // for number from -1 to 1, except zero  
 else if (X < 1)  
 {  
 // first operation  
 Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;  
 Number++;  
 // checking for the possibility of the next operation  
 if (Y0 - Y > EPS)  
 {  
 Y0 = Y;  
 Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;  
 Number++;  
 // since the difference between A and A0 is small,  
 // then you can do a complete search of all the remaining

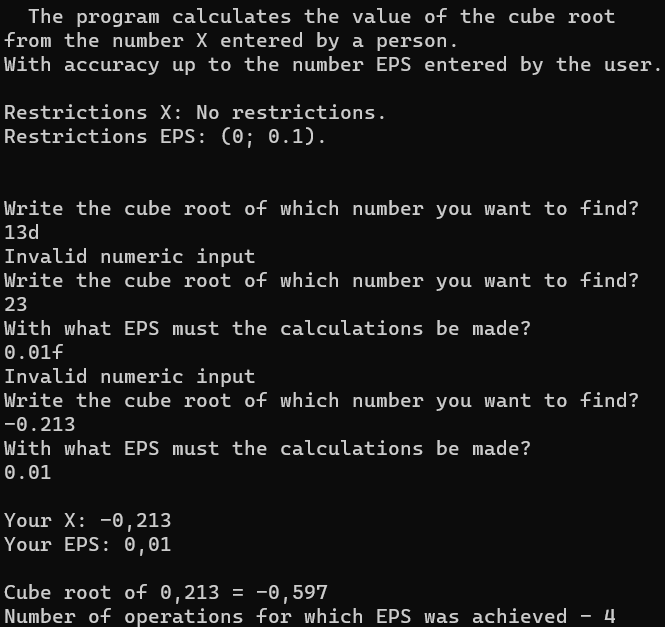
numbers  
 while (Y0 - Y > EPS)  
 {  
 Y0 = Y;  
 Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;  
 Number++;  
 }  
 }  
 }  
 // for all numbers except the range from -1 to 1  
 else  
 {  
 // first operation  
 Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;  
 Number++;  
 // checking for the possibility of the next operation  
 if (Y - Y0 > EPS)  
 {  
 Y0 = Y;  
 Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;  
 Number++;  
 // since the difference between A and A0 is small,  
 // then you can do a complete search of all the remaining

numbers  
 while (Y0 - Y > EPS)  
 {  
 Y0 = Y;  
 Y = ((Y0 \* 2) + (X / (Y0 \* Y0))) / 3;  
 Number++;  
 }  
 }  
 }  
  
 // output of final information  
 // cube root output  
 System.*out*.println("\nCube root of " + dF.format(X) + " = " +

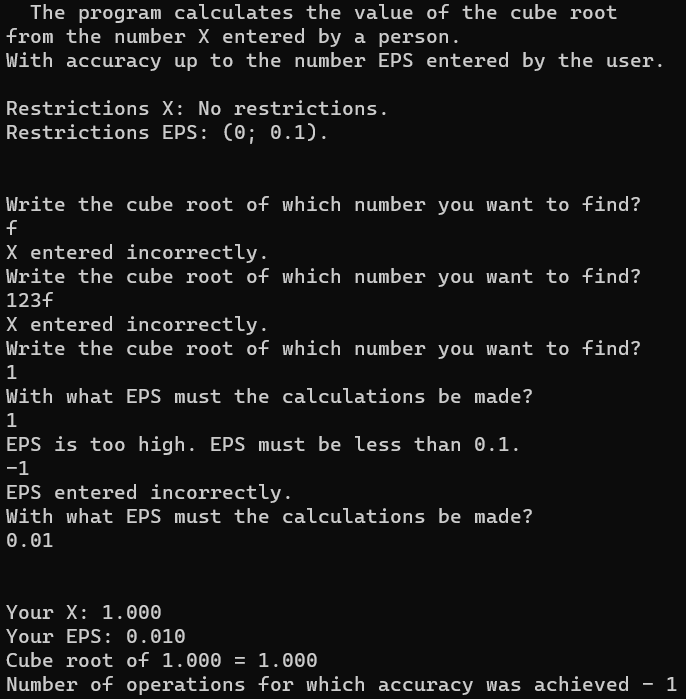
dF.format(Coefficient \* Y));  
 // number of operations to achieve accuracy  
 System.*out*.println("Number of operations for which accuracy was

achieved - " + Number);  
 IsCorrect = false;  
 in.close();  
 } catch (Exception error) {  
 // catching a mistake  
 if (error.getMessage() == null)  
 {  
 System.*out*.println("Invalid type!!!");  
 }  
 else  
 {  
 System.*out*.println(error.getMessage());  
 }  
 }  
 } while (IsCorrect);  
 }  
}

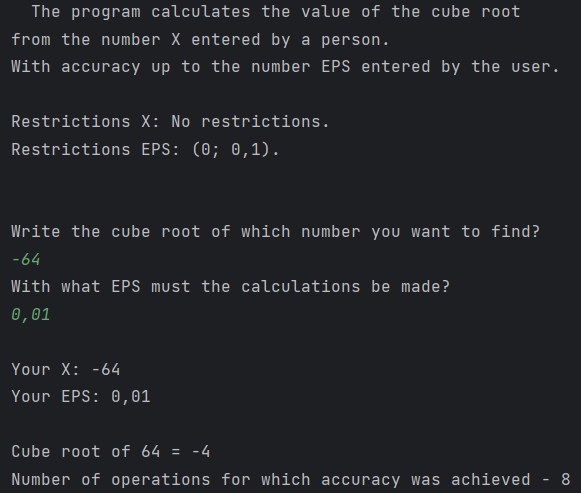
Результат на **Delphi**:



Результат на **C++**:



Результат на **Java**:



Блок-схема:

