УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.2

по предмету «Основы алгоритмизации и программирования»

Вариант 15

Выполнила:

Городко К. Е.

Гр. 351005

Проверила:

Данилова Г. В.

Минск 2024

**ЗАДАНИЕ:**

С помощью рекурсивной функции найти с заданной точностью квадратный корень Y = √X, воспользовавшись итерационной формулой Ньютона:

Y0=1

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

Вычисления производить, пока |Yi - Yi-1| не станет меньше EPS.

**КОД ПРОГРАММЫ DELPHI:**

Unit Unit1;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,

Vcl.Menus, Vcl.ExtDlgs;

Type

TMainForm = Class(TForm)

LabelCondition: TLabel;

LabelX: TLabel;

EditX: TEdit;

LabelEps: TLabel;

EditEps: TEdit;

ButtonResult: TButton;

EditY: TEdit;

LabelY: TLabel;

MainMenu: TMainMenu;

MMFile: TMenuItem;

MMOpenFile: TMenuItem;

MMSaveFile: TMenuItem;

MMExit: TMenuItem;

MMSeparator: TMenuItem;

MMInstruction: TMenuItem;

MMDevInfo: TMenuItem;

OpenTextFileDialog: TOpenTextFileDialog;

SaveTextFileDialog: TSaveTextFileDialog;

PopupMenu: TPopupMenu;

Procedure ButtonResultClick(Sender: TObject);

Procedure EditXKeyPress(Sender: TObject; Var Key: Char);

Procedure EditEpsKeyPress(Sender: TObject; Var Key: Char);

Procedure EditXKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure EditEpsKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Procedure EditChange(Sender: TObject);

Procedure EditXExit(Sender: TObject);

Procedure EditEpsExit(Sender: TObject);

Procedure MMOpenFileClick(Sender: TObject);

Procedure MMSaveFileClick(Sender: TObject);

Procedure MMExitClick(Sender: TObject);

Procedure MMDevInfoClick(Sender: TObject);

Procedure MMInstructionClick(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Procedure FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

End;

Var

MainForm: TMainForm;

IsSaved: Boolean;

Implementation

{$R \*.dfm}

Const

NULL = #0;

BACKSPACE = #8;

COMMA = ',';

MIN = 0;

MAX\_EPS = 1;

MAX\_X = 100000;

Function CalcSquareRoot(X, Eps, Prev: Real): Real;

Var

Y, Delta: Real;

Begin

Y := 1 / 2 \* (Prev + X / Prev);

Delta := Abs(Y - Prev);

If Delta > Eps Then

CalcSquareRoot := CalcSquareRoot(X, Eps, Y)

Else

CalcSquareRoot := Y;

End;

Function IsNumCorrect(NumStr: String; Const MAXNUM: Integer): Boolean;

Var

Num: Extended;

IsCorrect: Boolean;

Begin

IsCorrect := True;

If Not TryStrToFloat(NumStr, Num) Then

Begin

IsCorrect := False;

Application.MessageBox('Значения должны быть вещественными числами!',

'Ошибка', MB\_OK + MB\_ICONERROR);

End;

If IsCorrect And ((Num > MAXNUM) Or (Num < MIN)) Then

Begin

IsCorrect := False;

Application.MessageBox('Значение не попадает в диапазон!', 'Ошибка',

MB\_OK + MB\_ICONERROR);

End;

IsNumCorrect := IsCorrect;

End;

Function IsNumEqualZero(NumStr: String): Boolean;

Var

IsEqualZero: Boolean;

Begin

IsEqualZero := False;

If StrToFloat(NumStr) = MIN Then

Begin

IsEqualZero := True;

Application.MessageBox('Значение не должно быть равно 0!', 'Ошибка',

MB\_OK + MB\_ICONERROR);

End;

IsNumEqualZero := IsEqualZero;

End;

Function IsCommaInEdit(Txt: String): Boolean;

Var

I, Len: Integer;

IsCommaFound: Boolean;

Begin

Len := Length(Txt);

IsCommaFound := False;

I := 1;

While (I < Len + 1) And Not IsCommaFound Do

Begin

If Txt[I] = COMMA Then

IsCommaFound := True;

Inc(I);

End;

IsCommaInEdit := IsCommaFound;

End;

Function LengthBeforeComma(Txt: String): Integer;

Var

I: Integer;

Begin

I := 1;

While Txt[I] <> COMMA Do

Inc(I);

LengthBeforeComma := I;

End;

Procedure EditKeyPress(SelStart, Len: Integer; EditText: String; Var Key: Char);

Begin

If Not CharInSet(Key, ['0' .. '9', COMMA, BACKSPACE]) Then

Key := NULL

Else If IsCommaInEdit(EditText) And (Len - LengthBeforeComma(EditText) = 10)

And (Key <> BACKSPACE) Then

Key := NULL

Else If (Len > 0) Then

If (SelStart = 0) And (Key = '0') Then

Key := NULL

Else If (EditText = '0') And (SelStart = Len) And Not CharInSet(Key,

[BACKSPACE, COMMA]) Then

Key := NULL

Else If (Key = COMMA) And (IsCommaInEdit(EditText) Or (SelStart = 0)) Then

Key := NULL;

End;

Procedure TMainForm.ButtonResultClick(Sender: TObject);

Var

X, Y, Eps: Real;

Begin

If IsNumEqualZero(EditX.Text) Then

EditX.Text := ''

Else If IsNumEqualZero(EditEps.Text) Then

EditEps.Text := ''

Else

Begin

X := StrToFloat(EditX.Text);

Eps := StrToFloat(EditEps.Text);

Y := 1;

Y := CalcSquareRoot(X, Eps, Y);

EditY.Text := FormatFloat('0.##########', Y);

EditY.Enabled := True;

MMSaveFile.Enabled := True;

End;

End;

Procedure TMainForm.EditEpsExit(Sender: TObject);

Begin

If (EditEps.Text <> '') And (Not IsNumCorrect(EditEps.Text, MAX\_X) Or

IsNumEqualZero(EditEps.Text)) Then

EditEps.Text := '';

End;

Procedure TMainForm.EditEpsKeyDown(Sender: TObject; Var Key: Word; Shift:

TShiftState);

Begin

If Key = VK\_UP Then

EditX.SetFocus

Else If Key = VK\_DOWN Then

If ButtonResult.Enabled Then

ButtonResult.SetFocus

Else

EditX.SetFocus;

End;

Procedure TMainForm.EditEpsKeyPress(Sender: TObject; Var Key: Char);

Var

SelStart, Len: Integer;

EditText: String;

Begin

SelStart := EditEps.SelStart;

EditText := EditEps.Text;

Len := Length(EditText);

EditKeyPress(SelStart, Len, EditText, Key);

If (SelStart = 0) And Not(CharInSet(Key, ['0', BACKSPACE]) Or ((Key = '1') And

(Len = 0))) Then

Key := NULL

Else If (EditText = '1') And (Key <> BACKSPACE) Then

Key := NULL;

End;

Procedure TMainForm.EditChange(Sender: TObject);

Begin

If (EditX.Text <> '') And Not IsNumCorrect(EditX.Text, MAX\_X) Then

EditX.Text := '';

If (EditEps.Text <> '') And Not IsNumCorrect(EditEps.Text, MAX\_EPS) Then

EditEps.Text := '';

ButtonResult.Enabled := (EditX.Text <> '') And (EditEps.Text <> '');

EditY.Enabled := False;

EditY.Text := '';

IsSaved := False;

MMSaveFile.Enabled := False;

End;

Procedure TMainForm.EditXExit(Sender: TObject);

Begin

If (EditX.Text <> '') And (Not IsNumCorrect(EditX.Text, MAX\_X) Or

IsNumEqualZero(EditX.Text)) Then

EditX.Text := '';

End;

Procedure TMainForm.EditXKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Begin

If Key = VK\_DOWN Then

EditEps.SetFocus

Else If Key = VK\_UP Then

If ButtonResult.Enabled Then

ButtonResult.SetFocus

Else

EditEps.SetFocus;

End;

Procedure TMainForm.EditXKeyPress(Sender: TObject; Var Key: Char);

Var

SelStart, Len: Integer;

EditText: String;

Begin

SelStart := EditX.SelStart;

EditText := EditX.Text;

Len := Length(EditText);

EditKeyPress(SelStart, Len, EditText, Key);

If (Len > 4) And (Not IsCommaInEdit(EditText) Or (SelStart <

LengthBeforeComma(EditText))) And ((StrToFloat(EditText) > MAX\_X Div 10) And

Not CharInSet(Key, [BACKSPACE, COMMA]) Or ((StrToFloat(EditText) = MAX\_X Div

10) And Not CharInSet(Key, [BACKSPACE, COMMA, '0']))) Then

Key := NULL;

If (Len > 0) And (StrToFloat(EditText) = MAX\_X) And (Key <> BACKSPACE) Then

Key := NULL;

End;

Function IsFilePathCorrect(Path: String): Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

If ExtractFileExt(Path) <> '.txt' Then

Begin

Application.MessageBox('Файл не является текстовым!', 'Ошибка', MB\_OK);

IsCorrect := False;

End;

IsFilePathCorrect := IsCorrect;

End;

Procedure TMainForm.MMOpenFileClick(Sender: TObject);

Var

XStr, EpsStr: String;

IsCorrect: Boolean;

FIn: TextFile;

Begin

If OpenTextFileDialog.Execute And

IsFilePathCorrect(OpenTextFileDialog.FileName) Then

Begin

Try

Try

IsCorrect := True;

AssignFile(FIn, OpenTextFileDialog.FileName);

Reset(FIn);

Readln(FIn, XStr);

IsCorrect := IsNumCorrect(XStr, MAX\_X) And Not IsNumEqualZero(XStr);

If IsCorrect Then

Begin

Readln(FIn, EpsStr);

IsCorrect := IsNumCorrect(EpsStr, MAX\_EPS) And Not

IsNumEqualZero(EpsStr);

If Not Eof(FIn) Then

Begin

IsCorrect := False;

Application.MessageBox('В файле некорректное количество

строк!', 'Ошибка', MB\_ICONERROR);

End;

End;

Finally

CloseFile(FIn);

End;

Except

IsCorrect := False;

Application.MessageBox('Файл недоступен для чтения!', 'Ошибка',

MB\_ICONERROR);

End;

If IsCorrect Then

Begin

EditX.Text := XStr;

EditEps.Text := EpsStr;

End;

End;

End;

Procedure TMainForm.MMSaveFileClick(Sender: TObject);

Var

IsCorrect: Boolean;

FOut: TextFile;

Begin

IsCorrect := SaveTextFileDialog.Execute And

IsFilePathCorrect(SaveTextFileDialog.FileName);

If IsCorrect Then

Try

Try

IsCorrect := True;

AssignFile(FOut, SaveTextFileDialog.FileName);

Rewrite(FOut);

Writeln(FOut, 'Квадратный корень из ', EditX.Text, ' с точностью ',

EditEps.Text, ' равен ', EditY.Text);

Finally

CloseFile(FOut);

End;

Except

IsCorrect := False;

Application.MessageBox('Произошла ошибка при записи в файл!', 'Ошибка',

MB\_ICONERROR);

End;

IsSaved := IsCorrect;

End;

Procedure TMainForm.FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Var

ButtonSelected: Integer;

Begin

If MMSaveFile.Enabled And Not IsSaved Then

Begin

ButtonSelected := Application.MessageBox('Желаете сохранить результат в

файл?', 'Выход', MB\_YESNOCANCEL +

MB\_ICONQUESTION);

If ButtonSelected = MrYes Then

Begin

MMSaveFileClick(Sender);

If Not IsSaved Then

Close;

End

Else If ButtonSelected = MrNo Then

CanClose := True

Else

CanClose := False;

End

Else

Begin

ButtonSelected := Application.MessageBox('Вы уверены, что хотите выйти?',

'Выход', MB\_YESNO + MB\_ICONQUESTION);

If ButtonSelected = MrYes Then

CanClose := True

Else

CanClose := False;

End;

End;

Procedure TMainForm.FormKeyDown(Sender: TObject; Var Key: Word; Shift: TShiftState);

Begin

If (Key = VK\_RETURN) And ButtonResult.Enabled Then

ButtonResult.Click

Else If Key = VK\_F1 Then

MMInstruction.Click;

End;

Procedure TMainForm.MMDevInfoClick(Sender: TObject);

Begin

Application.MessageBox('Городко Ксения Евгеньевна'#13#10'351005 группа',

'О разработчике', MB\_OK);

End;

Procedure TMainForm.MMExitClick(Sender: TObject);

Begin

MainForm.Close;

End;

Procedure TMainForm.MMInstructionClick(Sender: TObject);

Begin

Application.MessageBox('1. Х и Eps должны быть вещественными числами.'#13#10'2.

Диапазон Х - (0, 100000); диапазон Eps - (0, 1].'#13#10

'3. В файле должно быть 2 строки: первая со значением X,

вторая - с Eps.','Инструкция', MB\_OK);

End;

End.

**КОД ПРОГРАММЫ С++:**

#include <iostream>

#include <fstream>

using namespace std;

const double MIN = 0;

const double MAX\_X = 100000;

const double MAX\_EPS = 1;

const string CONSOLE\_CHOICE = "0";

const string FILE\_CHOICE = "1";

bool isRangeIncorrect(double num, const double MAX) {

bool isIncorrect;

isIncorrect = false;

if ((num > MAX) || (num < MIN) || (num == MIN)) {

isIncorrect = true;

cout << "Значение не попадает в диапазон! Повторите попытку.\n";

}

return isIncorrect;

}

double getNumFromConsole(string outputMessage, const double MAX) {

double num;

bool isIncorrect;

num = 0;

do {

isIncorrect = false;

cout << outputMessage;

cin >> num;

if (cin.get() != '\n') {

cin.clear();

while (cin.get() != '\n');

isIncorrect = true;

cout << "Значения должны быть вещественными числами! Повторите

попытку.\n";

}

if (!isIncorrect)

isIncorrect = isRangeIncorrect(num, MAX);

} while (isIncorrect);

return num;

}

bool isFileExtIncorrect(string path, bool isIncorrect) {

if (!isIncorrect && (path.substr(path.size() - 4) != ".txt")) {

cout << "Введенный Вами файл не является текстовым! Повторите попытку.\n";

isIncorrect = true;

}

return isIncorrect;

}

string getFileInputPath() {

string path;

bool isIncorrect;

do {

isIncorrect = false;

cout << "В файле должно быть 2 строки, первая - со значением X, вторая - со

значением точности.\n";

cout << "Введите путь к файлу.\n";

cin >> path;

ifstream fin(path);

fin.open(path);

if (!fin.is\_open()) {

cout << "Файл недоступен для чтения! Повторите попытку.\n";

isIncorrect = true;

}

isIncorrect = isFileExtIncorrect(path, isIncorrect);

fin.close();

} while (isIncorrect);

return path;

}

double getNumFromFile(bool& isIncorrect, const double MAX, ifstream& fin) {

double num;

num = 0;

fin >> num;

if (!isIncorrect && fin.fail()) {

fin.clear();

isIncorrect = true;

cout << "Значения должны быть вещественными числами! Повторите попытку.\n";

}

if (!isIncorrect)

isIncorrect = isRangeIncorrect(num, MAX);

return num;

}

void getDataFromFile(double& x, double& eps, string path) {

bool isIncorrect;

do {

isIncorrect = false;

ifstream fin(path);

try {

x = getNumFromFile(isIncorrect, MAX\_X, fin);

if (!isIncorrect) {

eps = getNumFromFile(isIncorrect, MAX\_EPS, fin);

if (!fin.eof()) {

cout << "В файле некорректное количество строк! Повторите

попытку.\n";

isIncorrect = true;

}

}

}

catch (string errorMessage) {

cout << "Произошла ошибка при чтении из файла! Повторите попытку.\n";

isIncorrect = true;

}

fin.close();

if (isIncorrect)

path = getFileInputPath();

} while (isIncorrect);

cout << "Данные из файла успешно считаны.\n";

}

string getChoice(string outputMessage) {

string choice;

bool isIncorrect;

do {

isIncorrect = false;

cout << outputMessage;

cin >> choice;

if ((choice != CONSOLE\_CHOICE) && (choice != FILE\_CHOICE)) {

isIncorrect = true;

cout << "Введено некорректное значение! Повторите попытку.\n";

}

} while (isIncorrect);

return choice;

}

void inputData(double& x, double& eps) {

string choice;

string finPath;

x = 0;

choice = getChoice("Введите 0, если хотите вводить данные через консоль, или 1,

если использовать файл: ");

if (choice == CONSOLE\_CHOICE) {

x = getNumFromConsole("Введите X: ", MAX\_X);

eps = getNumFromConsole("Введите точность: ", MAX\_EPS);

}

else {

finPath = getFileInputPath();

getDataFromFile(x, eps, finPath);

}

}

double calcSquareRoot(double x, double eps, double prev) {

double y;

double delta;

y = 1.0 / 2.0 \* (prev + x / prev);

delta = abs(y - prev);

if (delta > eps)

return calcSquareRoot(x, eps, y);

else

return y;

}

string getFileOutputPath() {

string path;

bool isIncorrect;

do {

isIncorrect = false;

cout << "Введите путь к файлу, в который нужно записать результат.\n";

cin >> path;

ofstream fout(path);

fout.open(path);

if (!fout.is\_open()) {

cout << "Файл недоступен для записи! Повторите попытку.\n";

isIncorrect = true;

}

isIncorrect = isFileExtIncorrect(path, isIncorrect);

fout.close();

} while (isIncorrect);

return path;

}

void writeResultIntoFile(string path, double y, double x, double eps) {

bool isIncorrect;

do {

isIncorrect = false;

ofstream fout(path);

try {

fout << "Квадратный корень из " << x << " с точностью " << eps <<

" равен " << y;

}

catch (string errorMessage) {

cout << "Произошла ошибка при записи в файл! Повторите попытку.\n";

isIncorrect = true;

}

fout.close();

if (isIncorrect)

path = getFileOutputPath();

} while (isIncorrect);

cout << "Результат записан.\n";

}

void outputResult(double y, double x, double eps) {

string choice;

string foutPath;

choice = getChoice("Введите 0, если нужно вывести результат в консоль, или 1,

если записать в файл: ");

if (choice == CONSOLE\_CHOICE)

cout << "Квадратный корень из " << x << " с точностью " << eps << " равен "

<< y;

else {

foutPath = getFileOutputPath();

writeResultIntoFile(foutPath, y, x, eps);

}

}

void writeCondition() {

cout << "Данная программа, используя итерационную формулу Ньютона, находит с

заданной точностью квадратный корень из X.\n";

cout << "X и Eps (точность) - вещественные числа. X должно находиться в диапазоне

(0, 100000], а Eps - (0, 1].\n";

}

int main() {

setlocale(LC\_ALL, "Russian");

double x;

double eps;

double y;

y = 1;

writeCondition();

inputData(x, eps);

y = calcSquareRoot(x, eps, y);

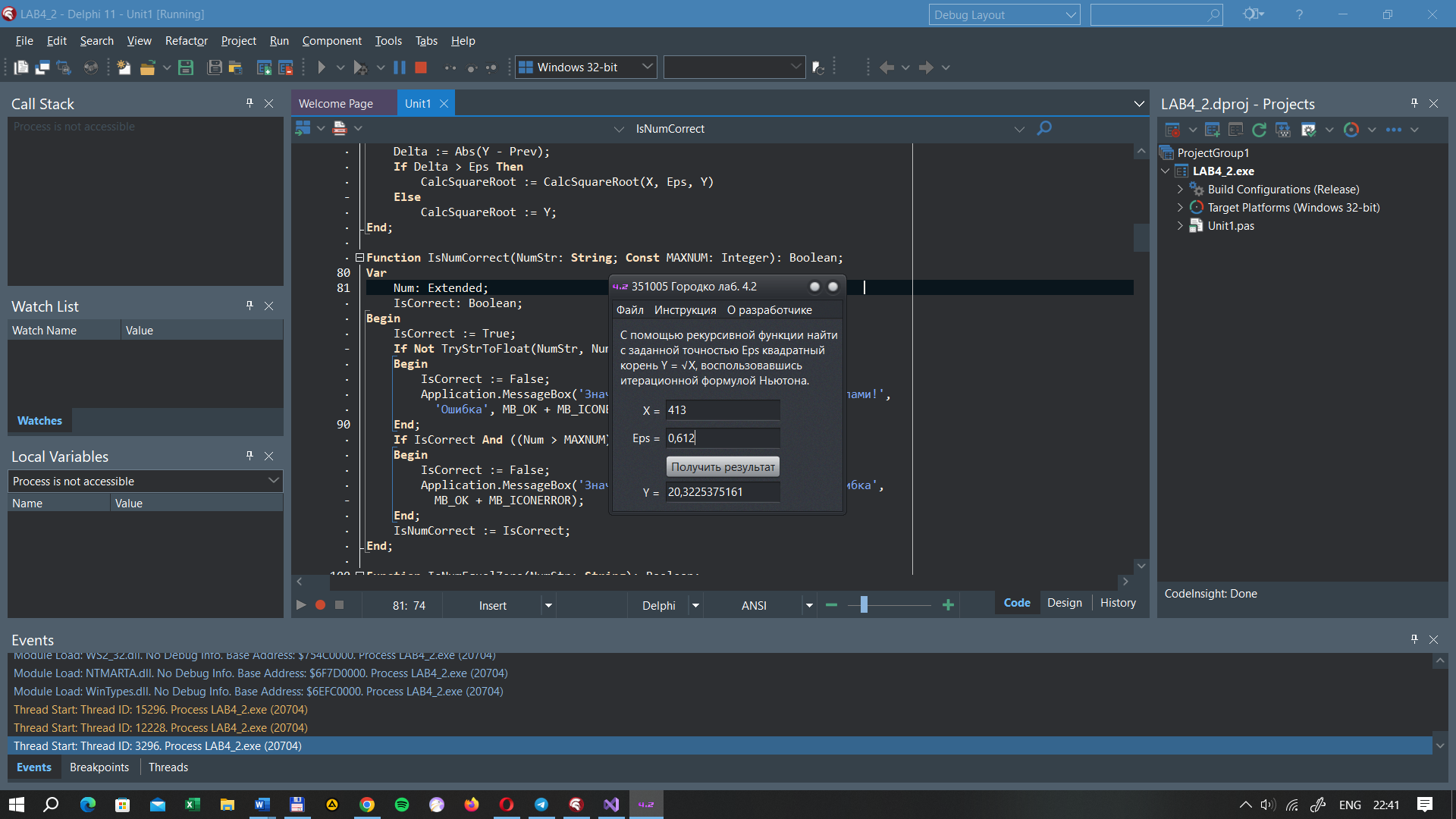
outputResult(y, x, eps);

return 0;

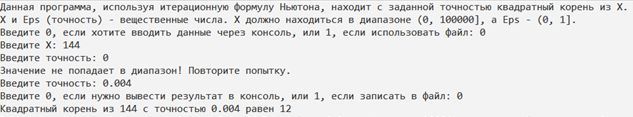
}

**СКРИНШОТЫ:**

**Delphi:**



**C++****:**



**БЛОК-СХЕМА:**

