**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информатикa»

**Отчет по лабораторной работе №3**

по дисциплине «Информатика»

Выполнил: студент группы БВТ1905

Семко Елена Константиновна

Руководитель:

Волков Андрей Иванович

Москва 2019

## Содержание

1 Постановка задачи……………………………………………………………………3  
2 Схемы алгоритмов………………………………………………..……………….4-13  
3 Текст программы…………………………………………………………………14-16  
4 Результаты тестирования……………………………………………………………17  
5 Приложение………………………………………………………………………….18  
6 Вывод…………………………………………………………………………………19

## 1 Постановка задачи

Реализовать на Visual C++ приложение для решения квадратного уравнения

## 2 Схемы алгоритмов

## Схема алгоритма функции, выполняющей «Защиту от дурака», представлена на рисунках 1-5

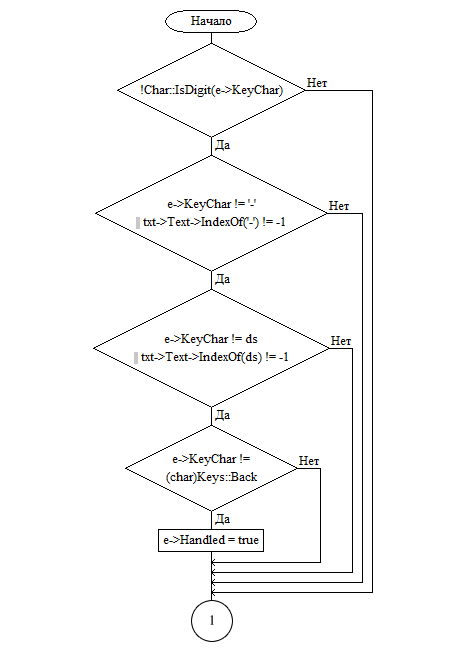


Рисунок 1 – Функция, выполняющая «Защиту от дурака»

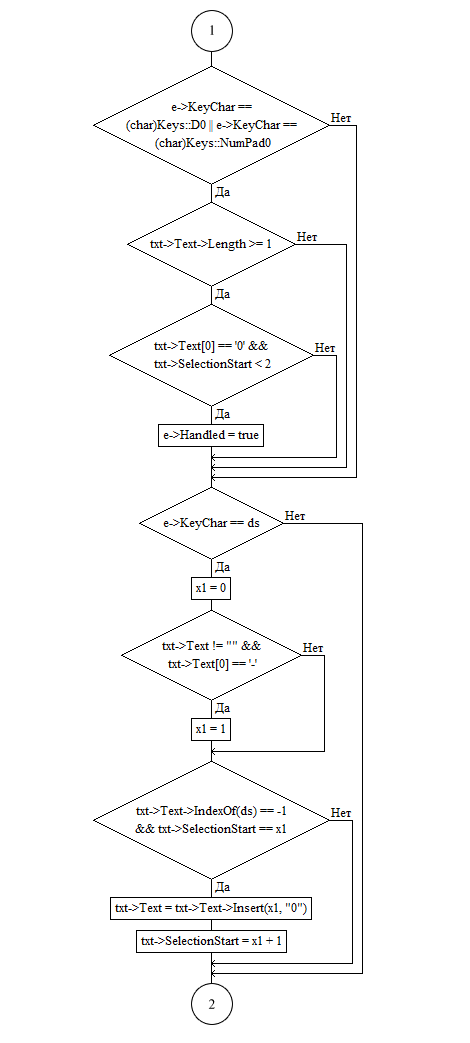


Рисунок 2 - Функция, выполняющая «Защиту от дурака»

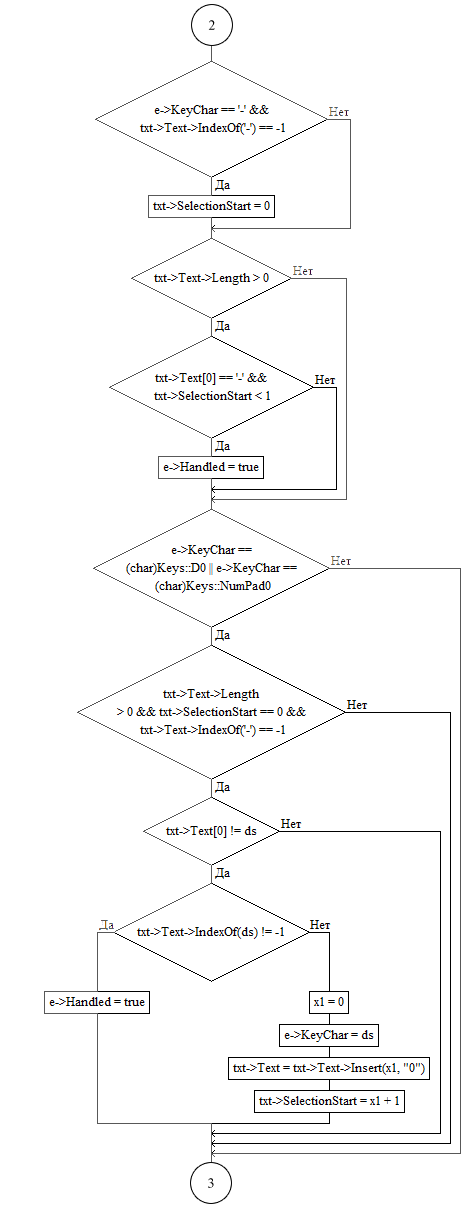


Рисунок 3 - Функция, выполняющая «Защиту от дурака»

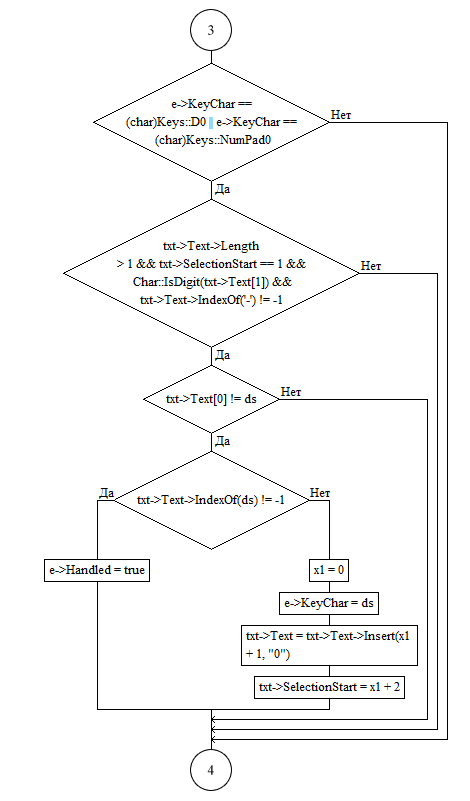


Рисунок 4 - Функция, выполняющая «Защиту от дурака»

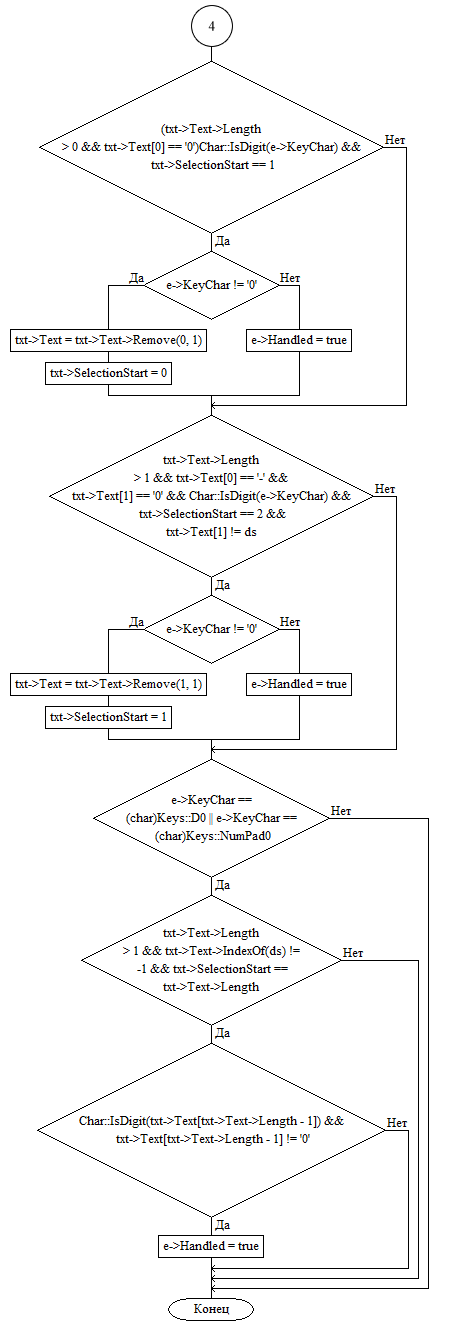


Рисунок 5 - Функция, выполняющая «Защиту от дурака»

Схема алгоритма функции, проверяющей вводимые значения, представлена на рисунке 6

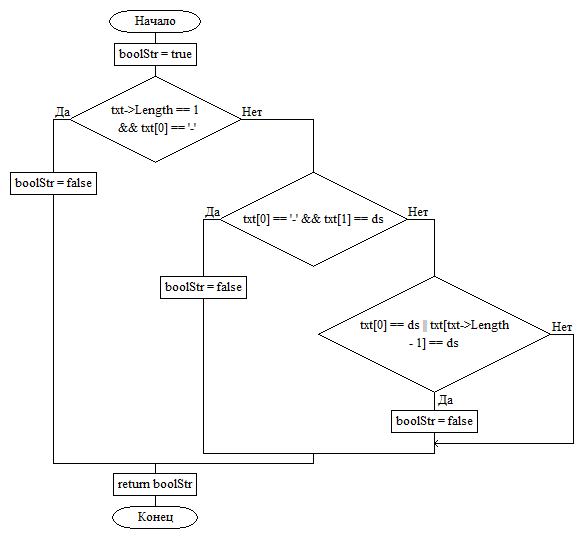


Рисунок 6 – Функция, проверяющая входящие значения

На рисунке 7 изображена блок-схема алгоритма события Form1\_Load (String^ txt)

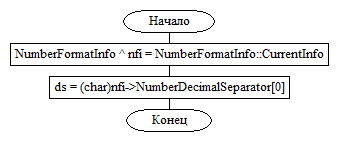


Рисунок 7

Схема алгоритма события «Ввод первого коэффициента» представлена на рисунке 8

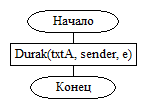


Рисунок 8 – Функция обработки события «Ввод первого коэффициента»

Схема алгоритма события «Ввод второго коэффициента» представлена на рисунке 9

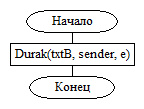


Рисунок 9 - Функция обработки события «Ввод второго коэффициента»

Схема алгоритма события «Ввод третьего коэффициента» представлена на рисунке 10

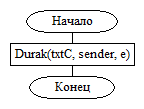


Рисунок 10 – Функция обработки события «Ввод третьего коэффициента»

Схема алгоритма функции обработки события «Нажатие кнопки «Решить»

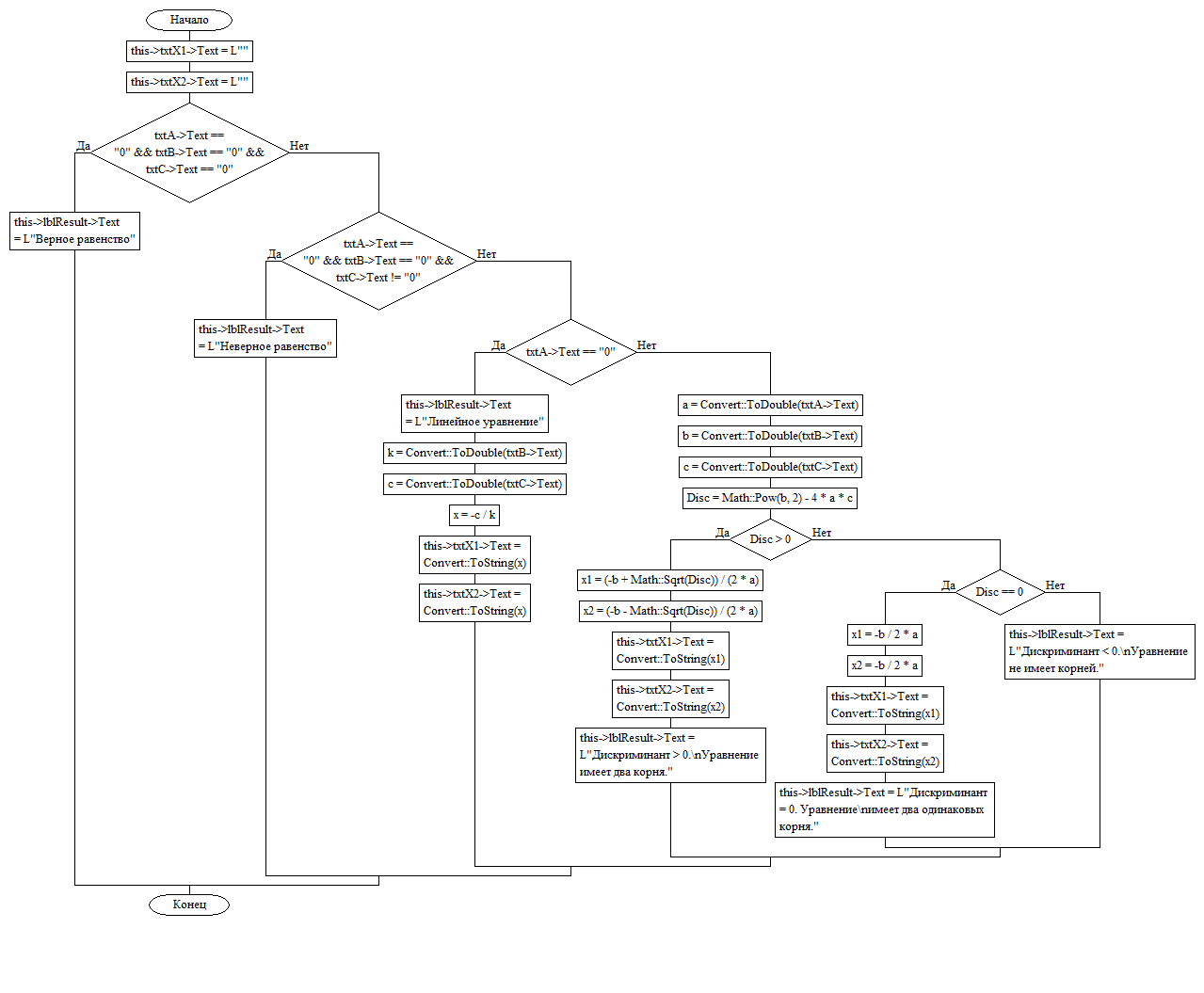


Рисунок 11 – Функция обработки события «Нажатие кнопки «Решить»

Схема алгоритма события «Закрытие программы»

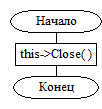


Рисунок 12 – Функция обработки события «Закрытие программы»

Схема алгоритма события «Сброс»

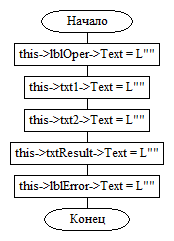


Рисунок 13 – Функция обработки события «Сброс»

## 3 Текст программы

Ниже представлен текст программы

#pragma endregion

private: System::Void Form1\_Load(System::Object^ sender, System::EventArgs^ e) {

// Определение десятичного разделителя (точки или запятой)

// из региональных настроек.

NumberFormatInfo^ nfi = NumberFormatInfo::CurrentInfo;

ds = (char)nfi->NumberDecimalSeparator[0];

}

private: System::Void lblKorni\_Click(System::Object^ sender, System::EventArgs^ e) {

}

// Организация ввода в txtA только действительных чисел.

// Фильтрация вводимых с клавиатуры действительных чисел.

private: System::Void txtA\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

Durak(txtA, sender, e); // Вызов функции для защиты от дурака

}

// Организация ввода в txtB только действительных чисел.

// Фильтрация вводимых с клавиатуры действительных чисел.

private: System::Void txtB\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

Durak(txtB, sender, e); // Вызов функции для защиты от дурака

}

// Организация ввода в txtC только действительных чисел.

// Фильтрация вводимых с клавиатуры действительных чисел.

private: System::Void txtC\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

Durak(txtC, sender, e); // Вызов функции для защиты от дурака

}

// Обработка кнопки "Решить"

private: System::Void btnAction\_Click(System::Object^ sender, System::EventArgs^ e) {

// Блок обработки ошибки

try

{

if (txtA->Text->Length < 1 || txtB->Text->Length < 1 // Проверяем не пустые ли входные данные

|| txtC->Text->Length < 1)

{

throw 0; // возвращаем код ошибки

}

if (IsTrueNumber(txtA->Text) == false ||

IsTrueNumber(txtB->Text) == false || IsTrueNumber(txtC->Text) == false)

{

throw 'e';

}

this->lblResult->ForeColor = System::Drawing::Color::Green; // Делаем LabelResult зелёным

this->txtX1->Text = L"";

this->txtX2->Text = L"";

if (txtA->Text == "0" && txtB->Text == "0" && txtC->Text == "0")

{

this->lblResult->Text = L"Верное равенство";

}

else if (txtA->Text == "0" && txtB->Text == "0" && txtC->Text != "0")

{

this->lblResult->Text = L"Неверное равенство";

}

else if (txtA->Text == "0")

{

this->lblResult->Text = L"Линейное уравнение";

double k, c, x;

k = Convert::ToDouble(txtB->Text); // конвертируем коэфициент k в Double

c = Convert::ToDouble(txtC->Text); // конвертируем свободный член с в Double

x = -c/k; // ищем x

this->txtX1->Text = Convert::ToString(x); // Конвертиаця x в String

this->txtX2->Text = Convert::ToString(x); // Конвертиаця x в String

}

else

{

double a, b, c, Disc, x1, x2; // Объявляем переменные для входных данных, дискриминанта и корней

// инициализация входных данных посредоством конвертации их в Double

a = Convert::ToDouble(txtA->Text); // конвертируем txtA

b = Convert::ToDouble(txtB->Text); // конвертируем txtB

c = Convert::ToDouble(txtC->Text); // конвертируем txtC

Disc = Math::Pow(b,2) - 4\*a\*c; // Задаём формулу дискриминанта и присваиваем значение в переменную Disc

if (Disc > 0) // Если дискриминант положительный

{

x1 = (-b + Math::Sqrt(Disc))/(2\*a); // По формуле вычисляем x1

x2 = (-b - Math::Sqrt(Disc))/(2\*a); // По формуле вычисляем x2

this->txtX1->Text = Convert::ToString(x1); // Конвертиаця x1 в String

this->txtX2->Text = Convert::ToString(x2); // Конвертиаця x2 в String

this->lblResult->Text = L"Дискриминант > 0.\nУравнение имеет два корня."; // Вывод сообщения о дискриминанте

}

else if (Disc == 0) // Если дискриминант нулевой

{

x1 = -b/2\*a; // По формуле вычисляем x1

x2 = -b/2\*a; // По формуле вычисляем x2

this->txtX1->Text = Convert::ToString(x1); // Конвертиация x1 в String

this->txtX2->Text = Convert::ToString(x2); // Конвертиация x2 в String

this->lblResult->Text = L"Дискриминант = 0. Уравнение\nимеет два одинаковых корня."; // Вывод сообщения о дискриминанте

}

else // Если дискриминант отрицательный

{

this->lblResult->Text = L"Дискриминант < 0.\nУравнение не имеет корней."; // Вывод сообщения о дискриминанте

}

}

}

catch(int a) // обработка ошибки

{

this->lblResult->Text = L"Одно или несколько\nтекстовых полей пустые!"; // Вывод сообщения об ошибке

this->lblResult->ForeColor = System::Drawing::Color::Red; // Делаем LabelResult красным

}

catch(char b)

{

this->lblResult->Text = "Введены некорректные данные"; // Вывод сообщения об ошибке

this->lblResult->ForeColor = System::Drawing::Color::Red; // Делаем LabelResult красны

}

}

// Обработка кнопки "Закрыть"

private: System::Void btnClose\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Close(); // Закрываем окно

}

// Обработка кнопки "Сброс"

private: System::Void btnReset\_Click(System::Object^ sender, System::EventArgs^ e) {

this->txtA->Text = L""; // Обнуляем txtA

this->txtB->Text = L""; // Обнуляем txtB

this->txtC->Text = L""; // Обнуляем txtC

this->txtX1->Text = L""; // Обнуляем txtX1

this->txtX2->Text = L""; // Обнуляем txtX2

this->lblResult->Text = L""; // Обнуляем txtResult

}

private: System::Void txtA\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void txtB\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void txtC\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

};

**4 Результаты тестирования**

В таблице 1 представлены результаты тестирования программы.

Таблица 1 - Тесты

|  |  |  |
| --- | --- | --- |
|  | Таблица 1 |  |
| **Введенные значения** | **Выбранная операция** | **Результат** |
| 1)А = 2 В = 4 С = -6 | Решить | x1 = 7; x2 = -3 Дискриминант > 0. Уравнение имеет два корня. |
| 2)А = 1 В = 2 С = 1 | Решить | x1 = -25; x2 = -25 Дискриминант = 0. Уравнение имеет два одинаковых корня. |
| 3)А = 3 В = 1 С = 2 | Решить | Дискриминант < 0. Уравнение не имеет корней. |
| 4)А = 0 В = 3 С = -6 | Решить | x1 = 2; x2 = 2 Линейное уравнение |
| 5)А = 0 В = 0 С = 7 | Решить | Неверное равенство |
| 6)А = 0 В = 0 С = 0 | Решить | Верное равенство |

## 5 Приложения

На рисунке 14 представлена корректная работа программы при D > 0

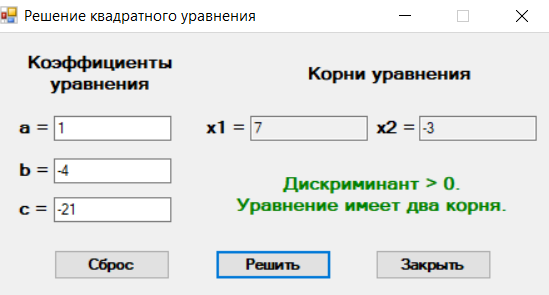


Рисунок 14 – Решение уравнения при дискриминанте больше нуля

На рисунке 15 представлена корректная работа программы при D = 0

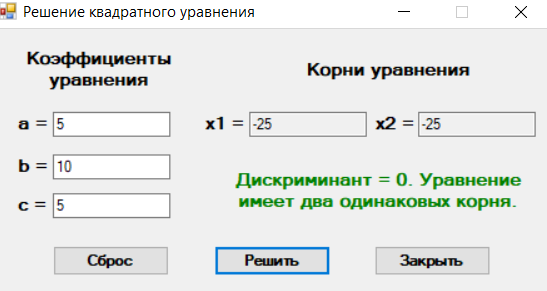


Рисунок 15 - Решение уравнения при дискриминанте равном нулю

На рисунке 16 представлена корректная работа программы при D < 0

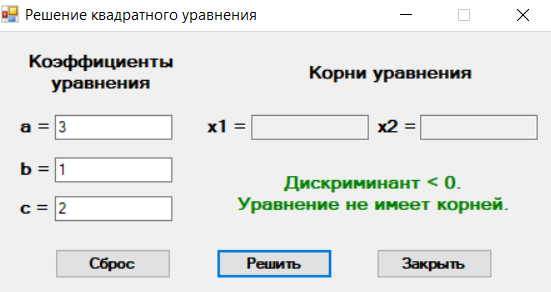


Рисунок 16 - Решение уравнения при дискриминанте меньше нуля

## Вывод

В ходе данной лабораторной работы были получены знания для создания простейшей программы по вычислению корней квадратного и линейного уравнений.