**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 №2**

по дисциплине «Информатика»

Выполнил: студент группы БВТ1905

Семко Елена Константиновна

Руководитель:

Волков Андрей Иванович

Москва 2019

## Содержание

1 Постановка задачи…………………………………………………………………….3  
2 Схемы алгоритмов…………………………………………………………………4-32  
3 Текст программы…………………………………………………………………33-45  
4 Результаты тестирования………………………………………………………...46-48  
5 Вывод…………………………………………………………………………………49

## 1 Постановка задачи

В среде разработки Visual Studio требуется создать калькулятор. Возможности программы должны включать в себя операции сложения, вычитания, деления и умножения чисел. Кроме того, должен быть реализован функционал сброса всех введенных данных (используя кнопку сброса). Программа должна правильно работать с действительными числами, реагировать на возможные ошибки ввода некорректных данных (выдавая соответствующие сообщения) и на попытку деления на ноль.

## 2 Схемы алгоритмов

## Схема алгоритма функции выбора простейших математических операций представлена на рисунке 1

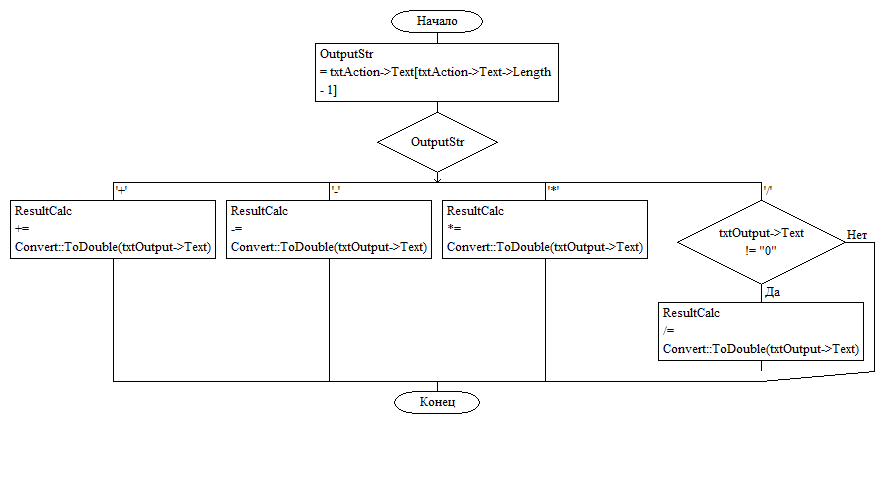


Рисунок 1 –Функция выбора простейших математических операций

Схема алгоритма функции блокировки кнопок представлена на рисунке 2

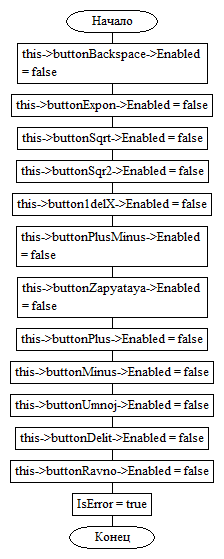


Рисунок 2 –Функция блокировки кнопок

Схема алгоритма функции разблокирования кнопок представлена на рисунке 3

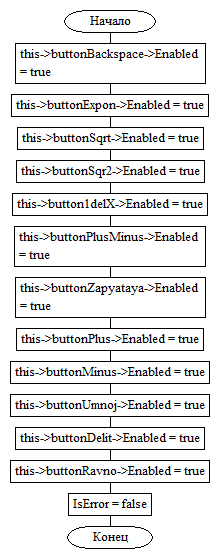


Рисунок 3 – Функция разблокирования кнопок

Схема алгоритма функции проверки результата на переполнение представлена на рисунке 4

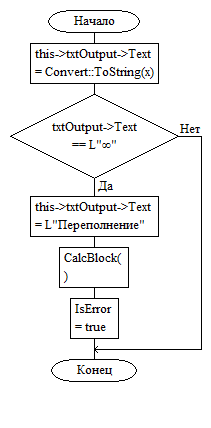


Рисунок 4 – Функция проверки результата на переполнение

Схема алгоритма для кнопки «Backspace» представлена на рисунке 5

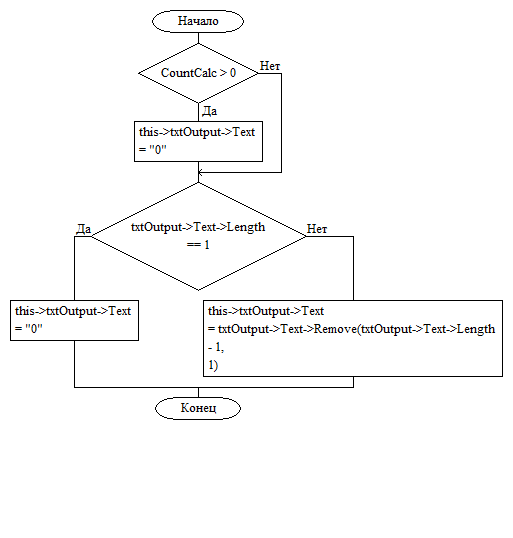


Рисунок 5 – Функция обработки события нажатия на кнопку «Backspace»

Схема алгоритма для кнопки «1» представлена на рисунке 6

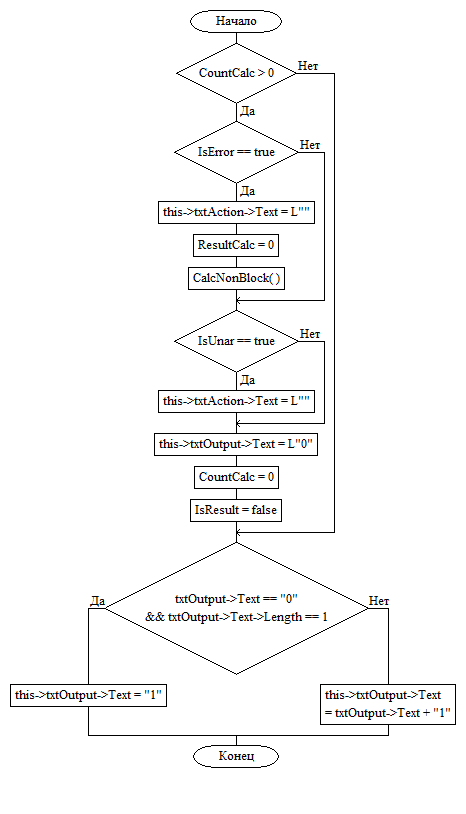


Рисунок 6 – Функция обработки события нажатия на кнопку 1

Схема алгоритма для кнопки «2» представлена на рисунке 7

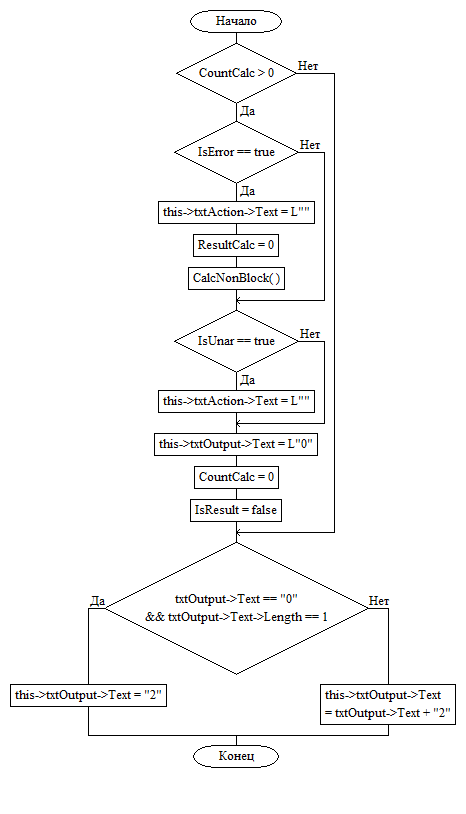


Рисунок 7 – Функция обработки события нажатия на кнопку «2»

Схема алгоритма для кнопки «3» представлена на рисунке 8

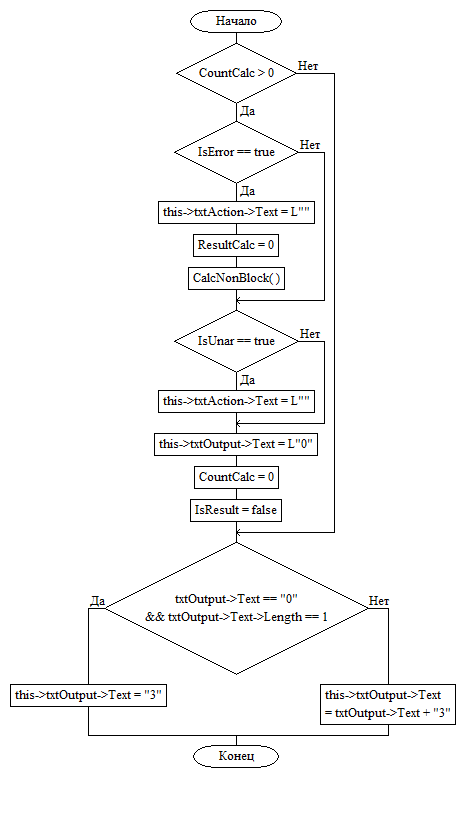


Рисунок 8 - Функция обработки события нажатия на кнопку «3»

Схема алгоритма для кнопки «4» представлена на рисунке 9

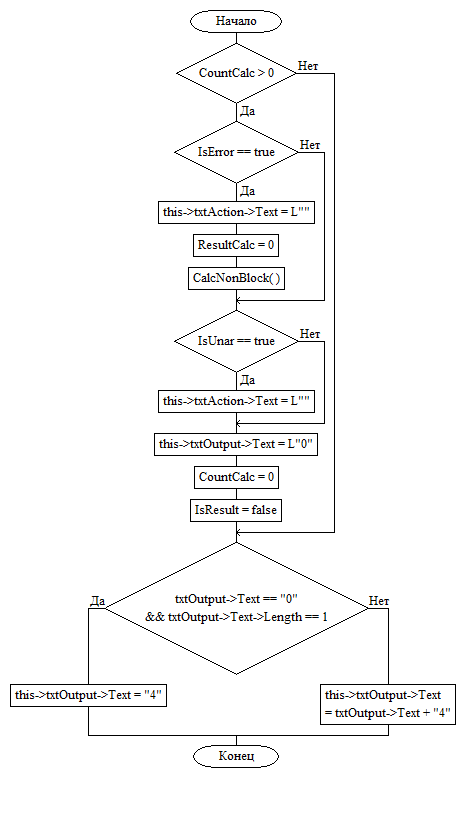


Рисунок 9 – Функция обработки события нажатия на кнопку «4»

Схема алгоритма для кнопки «5» представлена на рисунке 10

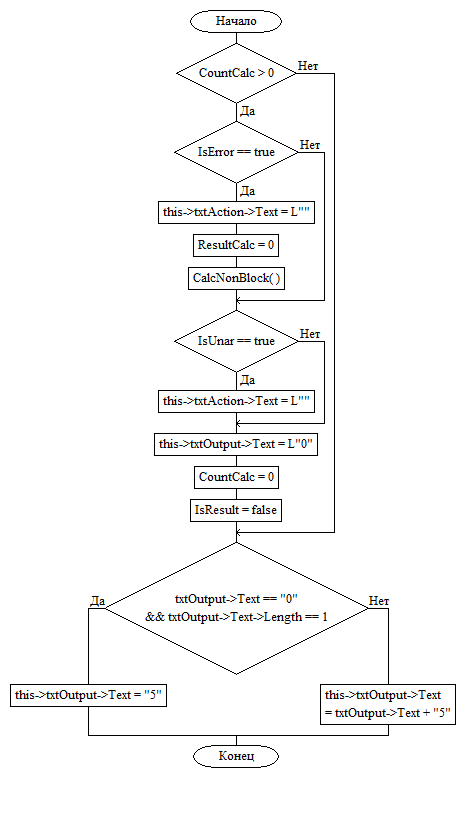


Рисунок 10 – Функция обработки события нажатия на кнопку «5»

Схема алгоритма для кнопки «6» представлена на рисунке 11

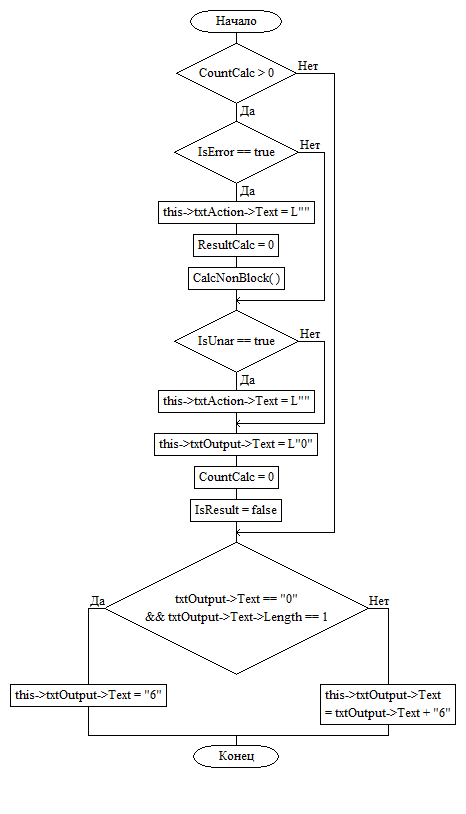


Рисунок 11 – Функция обработки события нажатия на кнопку «6»

Схема алгоритма для кнопки «7» представлена на рисунке 12

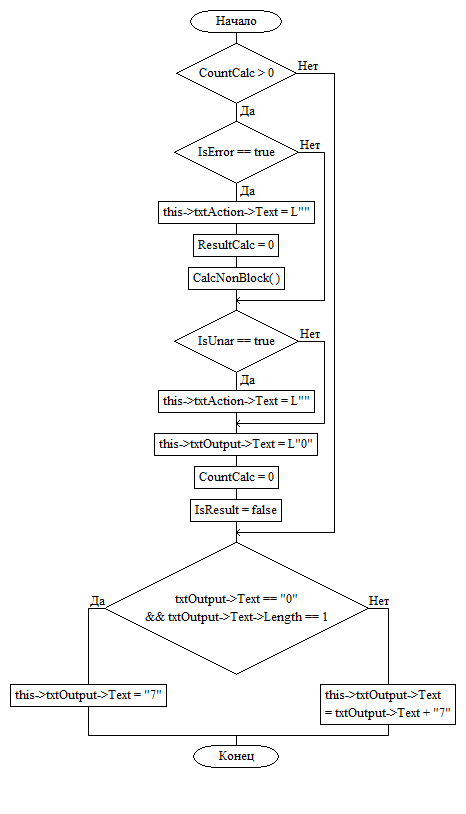


Рисунок 12 – Функция обработки события нажатия на кнопку «7»

Схема алгоритма для кнопки «8» представлена на рисунке 13

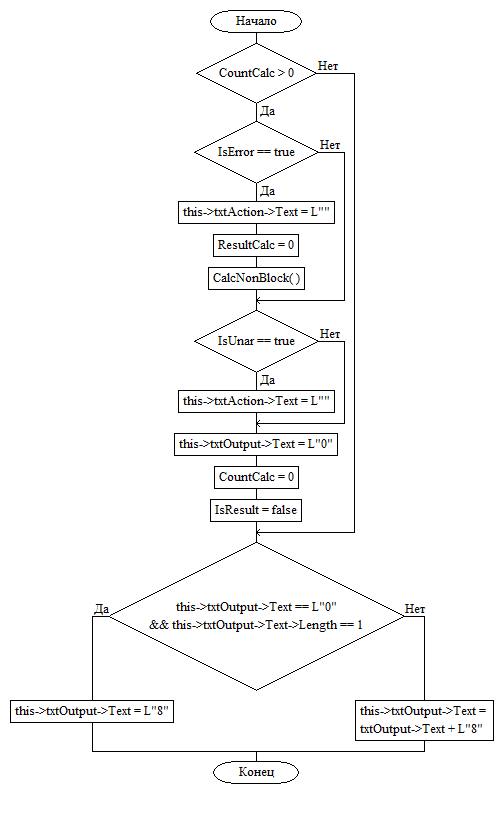


Рисунок 13 – Функция обработки события нажатия на кнопку «8»

Схема алгоритма для кнопки «9» представлена на рисунке 14

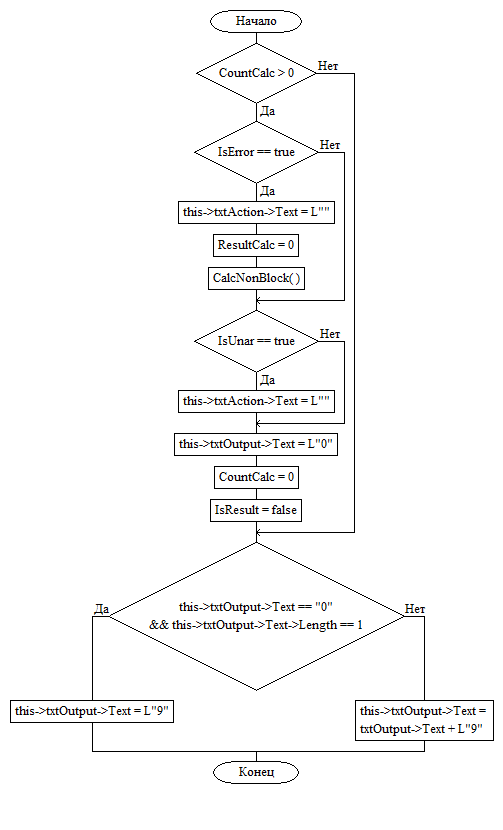


Рисунок 14 - Функция обработки события нажатия на кнопку «9»

Схема алгоритма для кнопки «0» представлена на рисунке 15

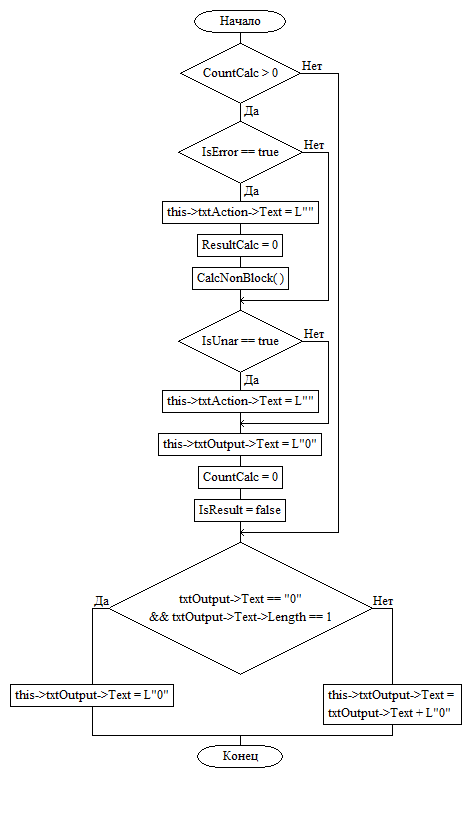


Рисунок 15 – Функция обработки события нажатия на кнопку «0»

Схема алгоритма для кнопки «CE» представлена на рисунке 16

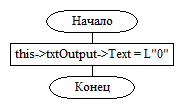


Рисунок 16 – Функция обработки события нажатия на кнопку «СЕ»

Схема алгоритма для кнопки «С» представлена на рисунке 17

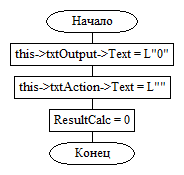


Рисунок 17 – Функция обработки события нажатия на кнопку «С»

Схема алгоритма для кнопки «Запятая» представлена на рисунке 18

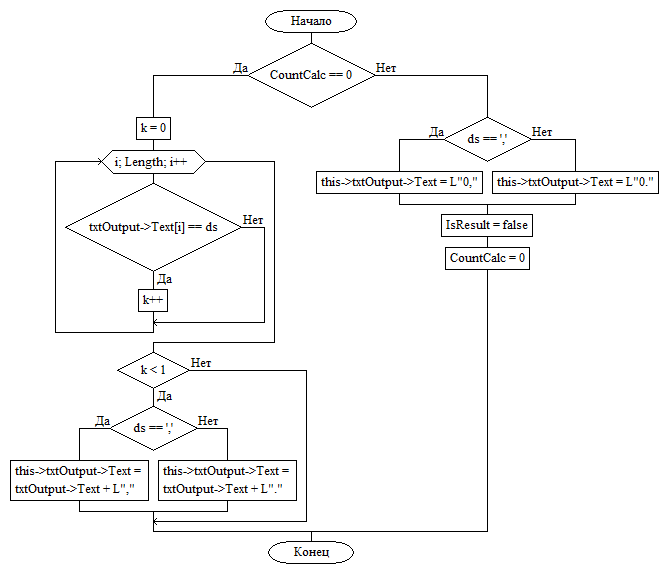


Рисунок 18 – Функция обработки события нажатия на кнопку «Запятая»

Схема алгоритма для кнопки «Корень» представлена на рисунке 19

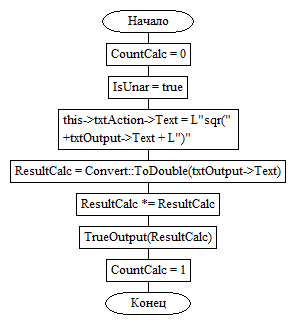


Рисунок 19 – Функция обработки события нажатия на кнопку «Корень»

Схема алгоритма для кнопки «Возведение в квадрат» представлена на рисунке 20

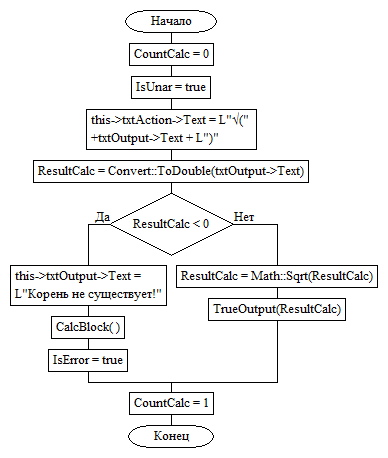


Рисунок 20 – Функция обработки события нажатия на кнопку «Возведение в квадрат»

Схема алгоритма для кнопки «1/х» представлена на рисунке 21

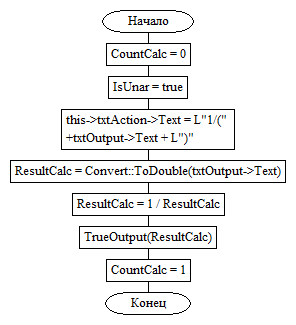


Рисунок 21 – Функция обработки события нажатия на кнопку «1/х»

Схема алгоритма для кнопки «Плюс-Минус» представлена на рисунке 22

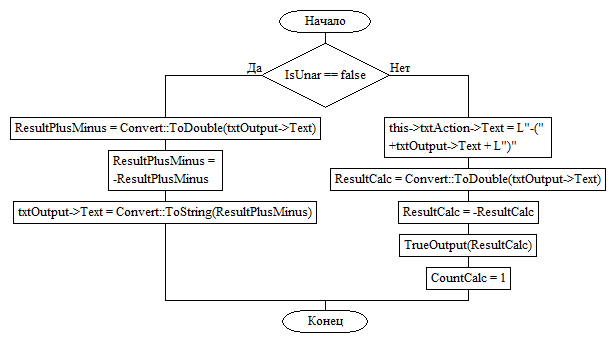


Рисунок 22 – Функция обработки события нажатия на кнопку «Плюс-Минус»

Схема алгоритма для кнопки «Экспонента» представлена на рисунке 23

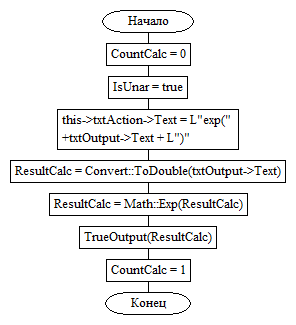


Рисунок 23 – Функция обработки события нажатия на кнопку «Экспонента»

Схема алгоритма для кнопки «Сложение» представлена на рисунке 24

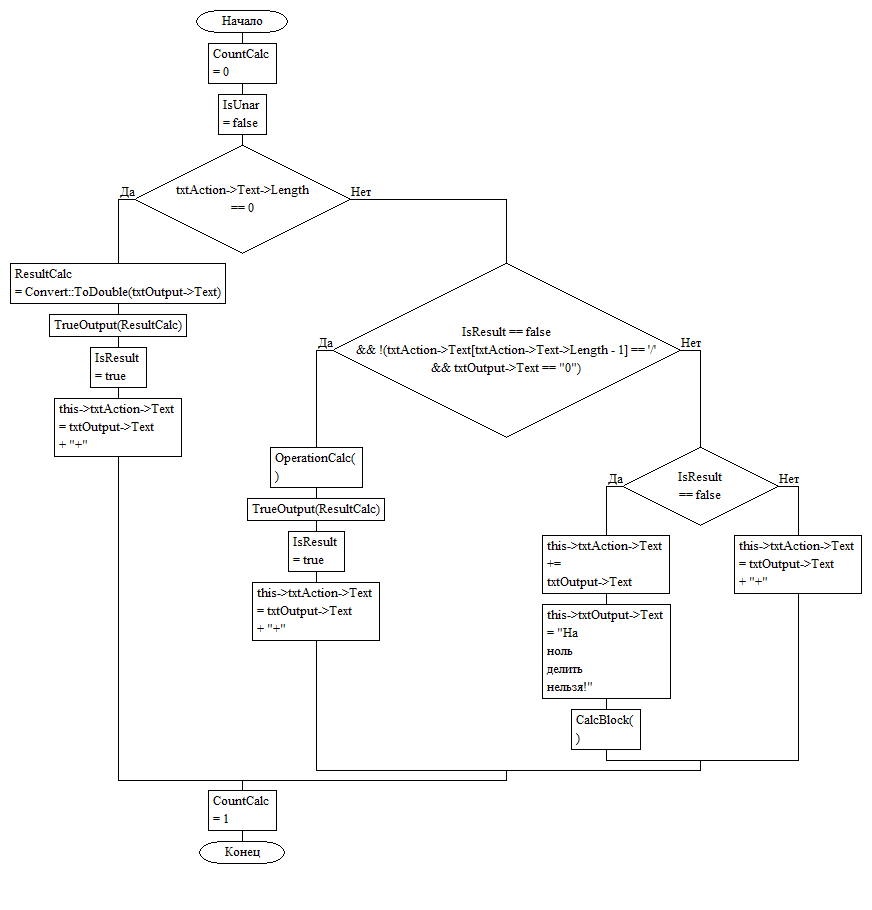


Рисунок 24 – Функция обработки события нажатия на кнопку «Сложение»

Схема алгоритма для кнопки «Вычитание» представлена на рисунке 25

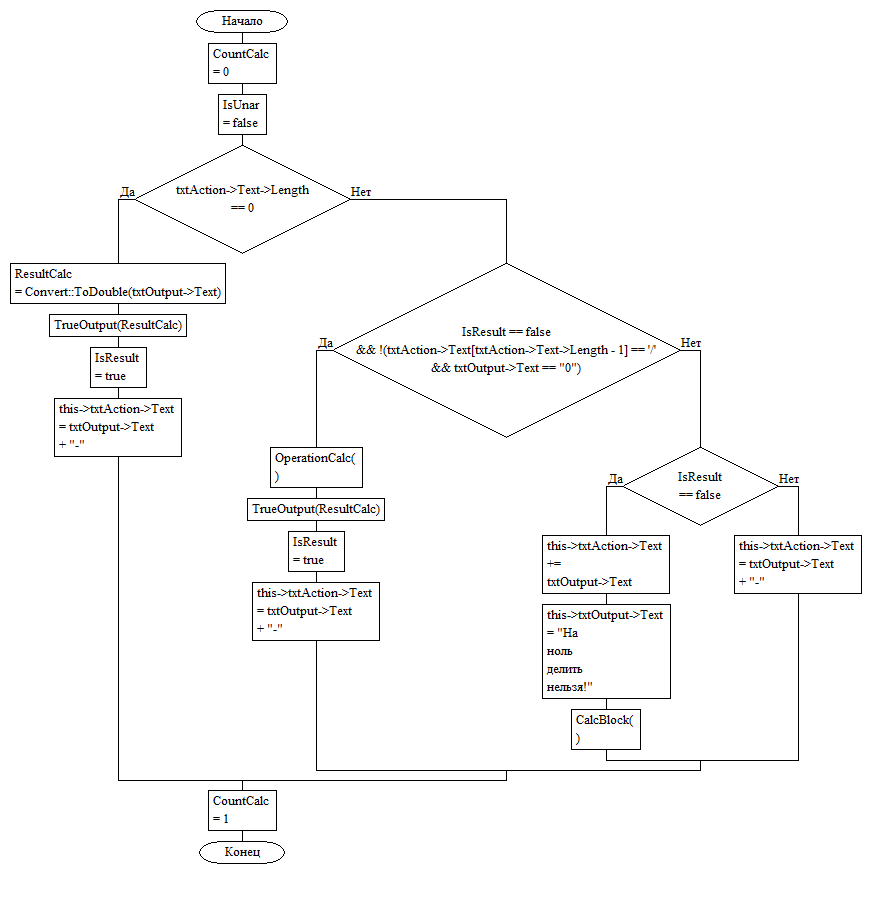


Рисунок 25 – Функция обработки события нажатия на кнопку «Вычитание»

Схема алгоритма для кнопки «Умножение» представлена на рисунке 26

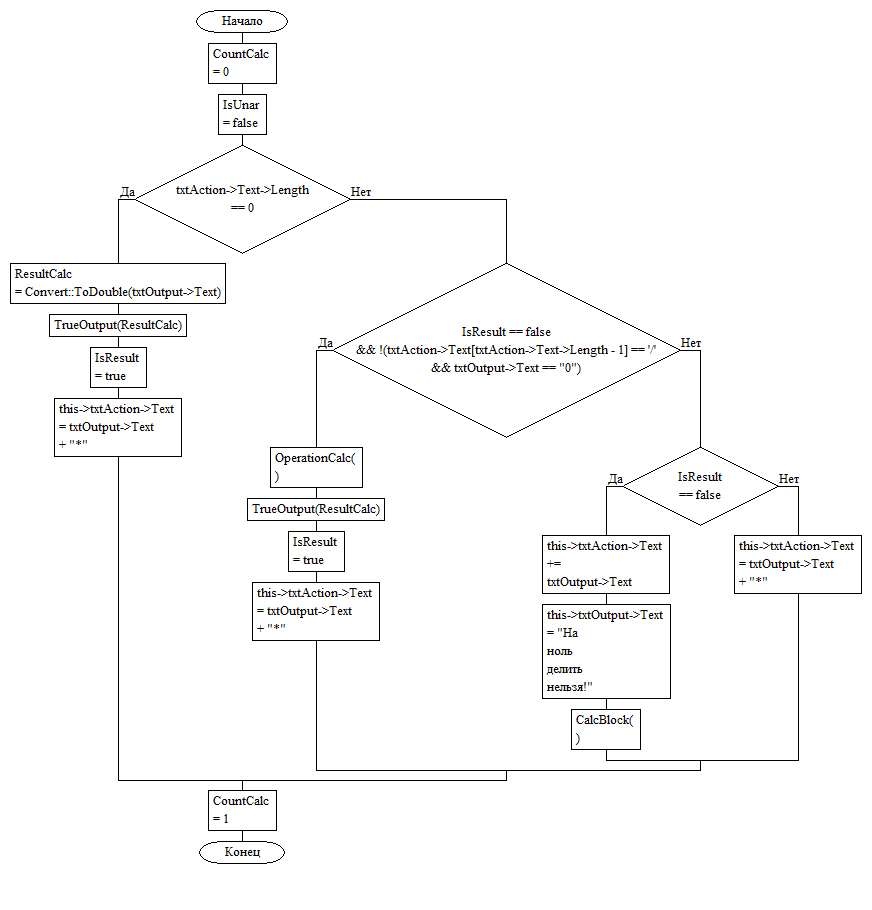


Рисунок 26 – Функция обработки события нажатия на кнопку «Умножение»

Схема алгоритма для кнопки «Деление» представлена на рисунке 27

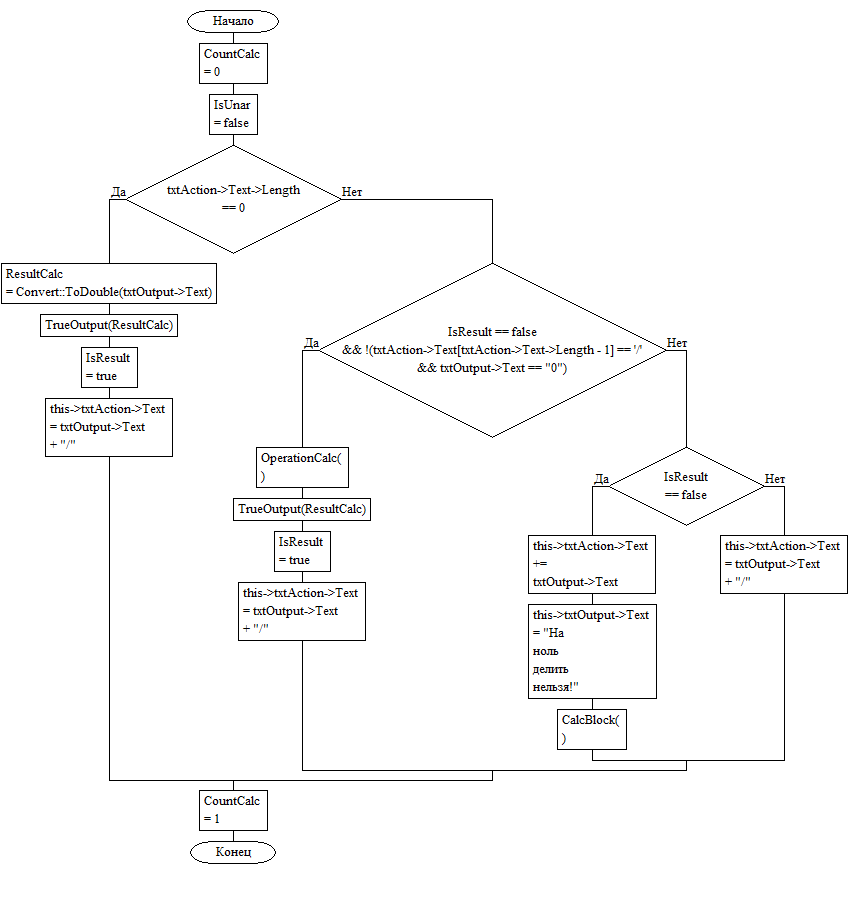


Рисунок 27 – Функция обработки события нажатия на кнопку «Деление»

Схема алгоритма для кнопки «Равно» представлена на рисунке 28

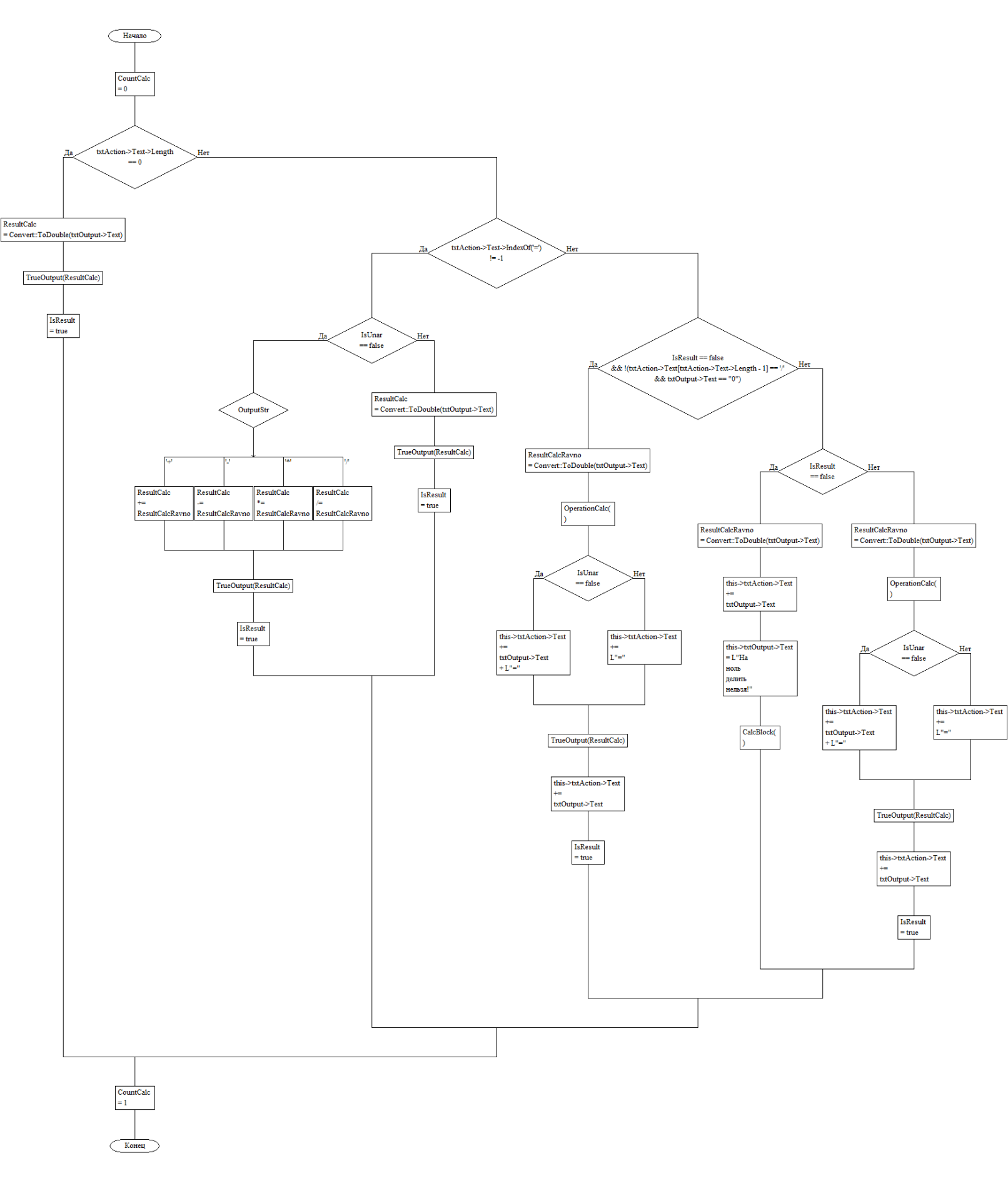


Рисунок 28 – Функция обработки события нажатия кнопки «Равно»

На рисунке 39 изображена блок-схема алгоритма события Form1\_Load

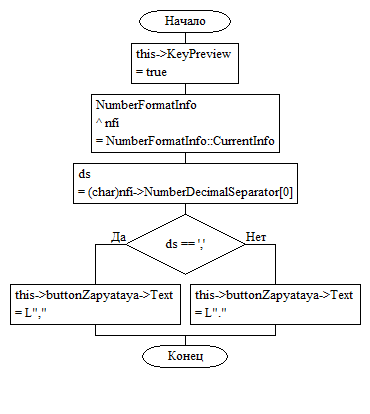


Рисунок 39

## 

## 3 Текст программы

Ниже представлен текст программы

#pragma endregion

// Процедура выбора между операциями

private: void OperationCalc()

{

// Считываем текущую строку

char OutputStr = txtAction->Text[txtAction->Text->Length-1];

switch (OutputStr) // В зависимости от её значения...

{

case '+': // если знак +

ResultCalc += Convert::ToDouble(txtOutput->Text); // складываем

break;

case '-': // если знак -

ResultCalc -= Convert::ToDouble(txtOutput->Text); // вычитаем

break;

case '\*': // если знак \*

ResultCalc \*= Convert::ToDouble(txtOutput->Text); // умножаем

break;

case '/': // если знак /

if (txtOutput->Text != "0") // если в знаменателе не 0

{

ResultCalc /= Convert::ToDouble(txtOutput->Text); // делим

}

break;

}

}

// Процедура блокирования кнопок в случая критической ситуации

private: void CalcBlock()

{

this->buttonBackspace->Enabled = false;

this->buttonExpon->Enabled = false;

this->buttonSqrt->Enabled = false;

this->buttonSqr2->Enabled = false;

this->button1delX->Enabled = false;

this->buttonPlusMinus->Enabled = false;

this->buttonZapyataya->Enabled = false;

this->buttonPlus->Enabled = false;

this->buttonMinus->Enabled = false;

this->buttonUmnoj->Enabled = false;

this->buttonDelit->Enabled = false;

this->buttonRavno->Enabled = false;

IsError = true; // ставим флаг ошибки в значение true

}

// Процедура разблокирования кнопок, когда критич. ситуация уже не угрожает работе

private: void CalcNonBlock()

{

this->buttonBackspace->Enabled = true;

this->buttonExpon->Enabled = true;

this->buttonSqrt->Enabled = true;

this->buttonSqr2->Enabled = true;

this->button1delX->Enabled = true;

this->buttonPlusMinus->Enabled = true;

this->buttonZapyataya->Enabled = true;

this->buttonPlus->Enabled = true;

this->buttonMinus->Enabled = true;

this->buttonUmnoj->Enabled = true;

this->buttonDelit->Enabled = true;

this->buttonRavno->Enabled = true;

IsError = false; // ставим флаг ошибки в значение false

}

private: void TrueOutput(double x)

{

this->txtOutput->Text = Convert::ToString(x); // конвертируем результат

//в строку

if (txtOutput->Text == L"∞") // если в строке встречен символ бесконечности

{

this->txtOutput->Text = L"Ошибка."; // вывести сообщение о переполнении

CalcBlock(); // блокируем кнопки

IsError = true; // ставим флаг ошибки в значение true

}

}

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

if (CountCalc > 0) // если до этого была операция, а не число

{

this->txtOutput->Text = "0"; // заменяем строку на 0

}

if (txtOutput->Text->Length == 1) // если длина строки равна 1

{

this->txtOutput->Text = "0"; // заменяем строку на 0

}

else

{

// удаляем последнюю цифру

this->txtOutput->Text = txtOutput->Text->Remove(txtOutput->Text->Length-1, 1);

}

}

private: System::Void buttonC1\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0) // если до этого была операция, а не число

{

if (IsError == true) // если возникла ошибка

{

this->txtAction->Text = L""; // обнуляем верхнюю строку

ResultCalc = 0; // обнуляем результат

CalcNonBlock(); // разблокируем кнопки

}

if (IsUnar == true) // если предыдущая операция унарная

{

this->txtAction->Text = L""; // обнуляем верхнюю строку

}

this->txtOutput->Text = L"0"; // обнуляем нижнюю строку

CountCalc = 0; // говорим, что теперь введено число, а не совершена операция

IsResult = false; // флаг реузультата ставим в false

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1) // если в строке 0 и её длина равна 1

{

this->txtOutput->Text = "1"; // замени строку на текущую цифру

}

else

{

this->txtOutput->Text = txtOutput->Text + "1"; // добавь текущую цифру к существующей строке

}

}

private: System::Void buttonC2\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "2";

}

else

{

this->txtOutput->Text = txtOutput->Text + "2";

}

}

private: System::Void buttonC3\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "3";

}

else

{

this->txtOutput->Text = txtOutput->Text + "3";

}

}

private: System::Void buttonC4\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "4";

}

else

{

this->txtOutput->Text = txtOutput->Text + "4";

}

}

private: System::Void buttonC5\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "5";

}

else

{

this->txtOutput->Text = txtOutput->Text + "5";

}

}

private: System::Void buttonC6\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "6";

}

else

{

this->txtOutput->Text = txtOutput->Text + "6";

}

}

private: System::Void buttonC7\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "7";

}

else

{

this->txtOutput->Text = txtOutput->Text + "7";

}

}

private: System::Void buttonC8\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "8";

}

else

{

this->txtOutput->Text = txtOutput->Text + "8";

}

}

private: System::Void buttonC9\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "9";

}

else

{

this->txtOutput->Text = txtOutput->Text + "9";

}

}

private: System::Void buttonC0\_Click(System::Object^ sender, System::EventArgs^ e) {

if (CountCalc > 0)

{

if (IsError == true)

{

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

ResultCalc = 0;

CalcNonBlock();

}

if (IsUnar == true)

{

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

}

this->txtOutput->Text = L"0";

CountCalc = 0;

IsResult = false;

}

if (txtOutput->Text == "0" && txtOutput->Text->Length == 1)

{

this->txtOutput->Text = "0";

}

else

{

this->txtOutput->Text = txtOutput->Text + "0";

}

}

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

this->txtOutput->Text = L"0";

}

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

this->txtOutput->Text = L"0";

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

ResultCalc = 0;

}

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

int k = 0; // объявляем и обнуляем счётчик

for (int i; i < txtOutput->Text->Length; i++) // запускаем цикл от начала до конца строки

{

if (txtOutput->Text[i] == ds) // если встречен десятичный разделитель

{

k++; // инкрементируем счётчик

}

}

if (k < 1) // если десятичный разделитель не был найден

{

if (ds == ',') // если десятичный разделитель запятая

{

this->txtOutput->Text = txtOutput->Text + L","; // вставляем в нижнюю строку запятую

}

else // если десятичный разделитель точка

{

this->txtOutput->Text = txtOutput->Text + L"."; // вставляем в нижнюю строку точку

}

}

}

private: System::Void buttonSqr2\_Click(System::Object^ sender, System::EventArgs^ e) {

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = true; // говорим, что эта операция унарная

// отображаем в верхней строке синтаксис операции возведения в квадрат

this->txtAction->Text = L"sqr(" + txtOutput->Text + L")";

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

ResultCalc \*= ResultCalc; // возводим входное значение в квадрат

TrueOutput(ResultCalc); // выводим результат

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = true; // говорим, что эта операция унарная

// отображаем в верхней строке синтаксис операции извлечения квадратного корня

this->txtAction->Text = L"√(" + txtOutput->Text + L")";

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

if (ResultCalc < 0) // если входное значение отрицательное

{

this->txtOutput->Text = L"Корень не существует!"; // выводим сообщение об ошибке

CalcBlock(); // блокируем кнопки

IsError = true; // ставим флаг ошибки в значение true

}

else // если входное значение >= 0

{

ResultCalc = Math::Sqrt(ResultCalc); // извлекаем из входного значения квадратный корень

TrueOutput(ResultCalc); // выводим результат

}

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

private: System::Void button1delX\_Click(System::Object^ sender, System::EventArgs^ e) {

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = true; // говорим, что эта операция унарная

// отображаем в верхней строке синтаксис операции 1/x

if (txtOutput->Text == "0") {

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

this->txtOutput->Text = L"На ноль делить нельзя!";}

else{

this->txtAction->Text = L"1/(" + txtOutput->Text + L")";

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

ResultCalc = 1/ResultCalc; // возводим входное значение в степень -1

TrueOutput(ResultCalc);} // выводим результат

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

if (IsUnar == false) // если текущая операция бинарная

{

double ResultPlusMinus = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

ResultPlusMinus = -ResultPlusMinus; // меняем знак входного значения на противоположный

txtOutput->Text = Convert::ToString(ResultPlusMinus); // выводим результат

}

else // если текущая операция унарная

{

// отображаем в верхней строке синтаксис смены знака

this->txtAction->Text = L"-(" + txtOutput->Text + L")";

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

ResultCalc = -ResultCalc; // меняем знак входного значения на противоположный

TrueOutput(ResultCalc); // выводим результат

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = true; // говорим, что эта операция унарная

// отображаем в верхней строке синтаксис нахождения экспоненты

this->txtAction->Text = L"exp(" + txtOutput->Text + L")";

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

ResultCalc = Math::Exp(ResultCalc); // находим экспоненту, аргументом передавая входное значение

TrueOutput(ResultCalc); // выводим результат

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = false; // говорим, что эта операция бинарная

if (txtAction->Text->Length == 0) // если верхняя строка нулевая

{

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "+"; // ставим в вехнюю строку входное значение и знак +

}

// иначе, если в нижней строке не результат и сейчас не ситуация деления на ноль

else if (IsResult == false && !(txtAction->Text[txtAction->Text->Length-1] == '/' && txtOutput->Text == "0"))

{

OperationCalc(); // запускаем функцию обработки простейших операций

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "+"; // ставим в вехнюю строку входное значение и знак +

}

// иначе, если это не результат, а значит ситуация деления на ноль

else if (IsResult == false)

{

this->txtAction->Text += txtOutput->Text; // добавляем в верхнюю строку входное значение

this->txtOutput->Text = "На ноль делить нельзя!"; // выводим сообщение об ошибке

CalcBlock(); // блокируем кнопки

}

else // в остальных случаях

{

this->txtAction->Text = txtOutput->Text + "+"; // ставим в вехнюю строку входное значение и знак +

}

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = false; // говорим, что эта операция бинарная

if (txtAction->Text->Length == 0) // если верхняя строка нулевая

{

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "-"; // ставим в вехнюю строку входное значение и знак -

}

// иначе, если в нижней строке не результат и сейчас не ситуация деления на ноль

else if (IsResult == false && !(txtAction->Text[txtAction->Text->Length-1] == '/' && txtOutput->Text == "0"))

{

OperationCalc(); // запускаем функцию обработки простейших операций

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "-"; // ставим в вехнюю строку входное значение и знак -

}

// иначе, если это не результат, а значит ситуация деления на ноль

else if (IsResult == false)

{

this->txtAction->Text += txtOutput->Text; // добавляем в верхнюю строку входное значение

this->txtOutput->Text = "На ноль делить нельзя!"; // выводим сообщение об ошибке

CalcBlock(); // блокируем кнопки

}

else // в остальных случаях

{

this->txtAction->Text = txtOutput->Text + "-"; // ставим в вехнюю строку входное значение и знак -

}

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = false; // говорим, что эта операция бинарная

if (txtAction->Text->Length == 0) // если верхняя строка нулевая

{

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "\*"; // ставим в вехнюю строку входное значение и знак \*

}

// иначе, если в нижней строке не результат и сейчас не ситуация деления на ноль

else if (IsResult == false && !(txtAction->Text[txtAction->Text->Length-1] == '/' && txtOutput->Text == "0"))

{

OperationCalc(); // запускаем функцию обработки простейших операций

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "\*"; // ставим в вехнюю строку входное значение и знак \*

}

// иначе, если это не результат, а значит ситуация деления на ноль

else if (IsResult == false)

{

this->txtAction->Text += txtOutput->Text; // добавляем в верхнюю строку входное значение

this->txtOutput->Text = "На ноль делить нельзя!"; // выводим сообщение об ошибке

CalcBlock(); // блокируем кнопки

}

else // в остальных случаях

{

this->txtAction->Text = txtOutput->Text + "\*"; // ставим в вехнюю строку входное значение и знак \*

}

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

IsUnar = false; // говорим, что эта операция бинарная

if (txtAction->Text->Length == 0) // если верхняя строка нулевая

{

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "/"; // ставим в вехнюю строку входное значение и знак /

}

// иначе, если в нижней строке не результат и сейчас не ситуация деления на ноль

else if (IsResult == false && !(txtAction->Text[txtAction->Text->Length-1] == '/' && txtOutput->Text == "0"))

{

OperationCalc(); // запускаем функцию обработки простейших операций

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг результата в true

this->txtAction->Text = txtOutput->Text + "/"; // ставим в вехнюю строку входное значение и знак /

}

// иначе, если это не результат, а значит ситуация деления на ноль

else if (IsResult == false)

{

this->txtAction->Text += txtOutput->Text; // добавляем в верхнюю строку входное значение

this->txtOutput->Text = "На ноль делить нельзя!"; // выводим сообщение об ошибке

CalcBlock(); // блокируем кнопки

}

else // в остальных случаях

{

this->txtAction->Text = txtOutput->Text + "/"; // ставим в вехнюю строку входное значение и знак /

}

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

CountCalc = 0; // говорим, что сейчас происходит операция, а не вставка цифры

// если строка верхняя строка нулевая или там уже есть знак равно

if (txtAction->Text->Length == 0 || txtAction->Text->IndexOf('=') != -1)

{

ResultCalc = Convert::ToDouble(txtOutput->Text); // конвертируем входное значение в Double

TrueOutput(ResultCalc); // выводим результат

IsResult = true; // ставим флаг реузльтат в true

}

// иначе, если в нижней строке не результат и сейчас не ситуация деления на ноль

else if (IsResult == false && !(txtAction->Text[txtAction->Text->Length-1] == '/' && txtOutput->Text == "0"))

{

OperationCalc(); // запускаем функцию обработки простейших операций

if (IsUnar == false) // если предыдущая операция была бинарной

{

this->txtAction->Text += txtOutput->Text + L"="; // добавляем к верхней строке текущей значение и знак равно

}

else // иначе, если предыдущая операция была унарной

{

this->txtAction->Text += L"="; // добавляем к верхней строке только знак равно

}

TrueOutput(ResultCalc); // выводим результат

this->txtAction->Text += txtOutput->Text; // добавляем к верхей строке полученный результат

IsResult = true; // ставим флаг результата в true

}

// иначе, если это не результат, а значит ситуация деления на ноль

else if (IsResult == false)

{

this->txtAction->Text += txtOutput->Text; // добавляем в верхнюю строку входящее значение

this->txtOutput->Text = L"На ноль делить нельзя!"; // выводим сообщение об ошибке

CalcBlock(); // блокируем кнопки

}

else // в остальных случаях

{

OperationCalc(); // запускаем функцию обработки простейших операций

if (IsUnar == false) // если предыдущая операция была бинарной

{

this->txtAction->Text += txtOutput->Text + L"="; // добавляем к верхней строке текущей значение и знак равно

}

else // иначе, если предыдущая операция была унарной

{

this->txtAction->Text += L"="; // добавляем к верхней строке только знак равно

}

TrueOutput(ResultCalc); // выводим результат

this->txtAction->Text += txtOutput->Text; // добавляем к верхей строке полученный результат

IsResult = true; // ставим флаг результата в true

}

IsUnar = true; // говорим, что последняя операция была унарной

CountCalc = 1; // переключаемся с операции на последующий ввод цифр

}

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

this->KeyPreview = true;

// Определение десятичного разделителя (точки или запятой)

// из региональных настроек.

NumberFormatInfo^ nfi = NumberFormatInfo::CurrentInfo;

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

// если десятичный разделитель - это запятая

if (ds == ',')

{

this->buttonZapyataya->Text = L","; // ставим на текст кнопки запятую

}

// иначе, если десятичный разделитель - это точка

else

{

this->buttonZapyataya->Text = L"."; // ставим на текст кнопки точку

}

}

**4 Результаты тестирования**  
В таблице 1 представлены результаты тестирования программы.

Таблица 1 - Тесты

|  |  |  |
| --- | --- | --- |
|  | Таблица 1 |  |
| **Введенные значения** | **Выбранная операция** | **Результат** |
| 1) Первое значение = 6,6 Второе значение = 7,7 | + | 14,3 |
| 2) Первое значение = -6 Второе значение = 2 | - | -8 |
| 3) Первое значение = 3 Второе значение = 2,1 | \* | 6,3 |
| 4) Первое значение = 10 Второе значение = 5 | / | 2 |
| 5) Входное значение = 5 | exp(x) | 148,413159102577 |
| 6) Входное значение = 98 | +/- | -98 |
| 7) Входное значение = 256 | sqrt(x) | 16 |
| 8) Входное значение = 36 | sqr(x) | 1296 |
| 9) Входное значение = 25 | 1/x | 0,04 |
| 10) Первое значение = -3 Второе значение = 0 | / | На ноль делить нельзя |
| 11) Входное значение = -14 | sqrt(x) | Корень не существует! |
| 12) Входное значение = 1234567890 | exp(x) | Переполнение |

## 

На рисунке 40 представлена корректная работа программы при попытке пользователя разделить число на 0.

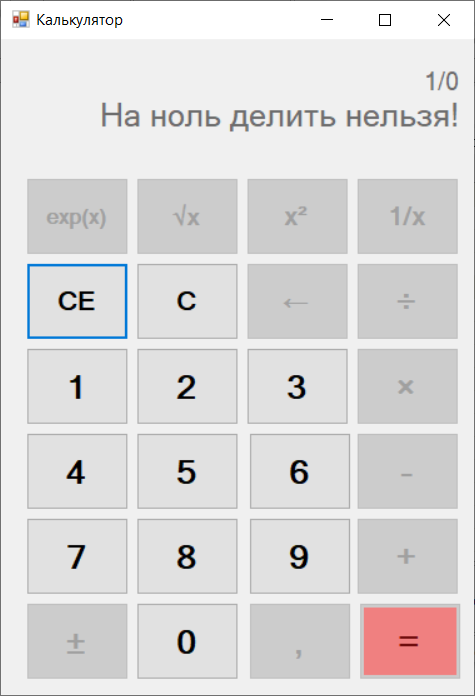


Рисунок 40 – Деление на ноль

На рисунке 41 представлена корректная работа программы при попытке пользователя извлечь корень из отрицательного числа.

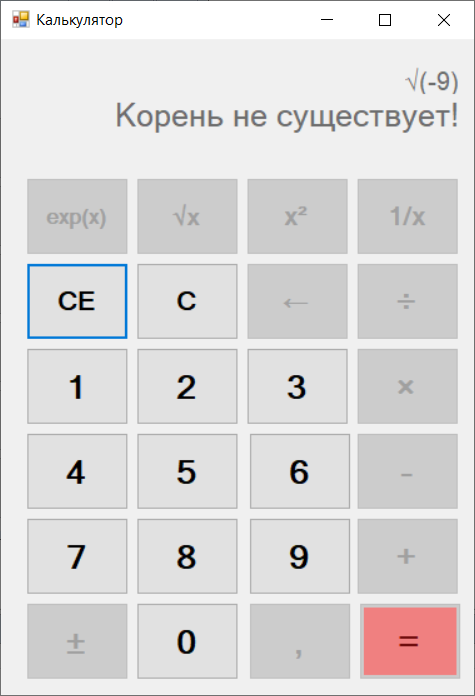


Рисунок 41 – Извлечение корня

## 5 Вывод

В ходе данной лабораторной работы была создана программа модели сложного калькулятора, которая умеет выполнять действия с вещественными числами и грамотно реагировать на попытку деления на ноль, извлечения корня из отрицательного числа.