**Федеральное агентство связи**

**Государственное бюджетное образовательное учреждение высшего**

**образование**

**Ордена Трудового Красного Знамени**

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

Кафедра информатики

**Отчет по лабораторной работе №5**

«Разработка программы с разветвлениями

“Расчёт значения exp(x)”»

по дисциплине «Информатика»

Подготовил студент

группы БВТ1903: Щитов В.М.

Проверил: Волков А.И.

Москва 2020

Содержание

[1 Постановка задачи 3](#_Toc37348918)

[2 Схемы алгоритмов 5](#_Toc37348919)

[3 Текст программы 19](#_Toc37348920)

[4 Результаты тестирования программы 25](#_Toc37348921)

[Вывод 28](#_Toc37348922)

# **1 Постановка задачи**

Разработать на Visual C++ приложение «Расчет значения exp(x)». Условие задачи предполагает следующие пункты его реализации:

1. Создать форму, имеющую вид, представленный на рисунке 1, приблизительный результат работы которой также представлен на рисунке 1.

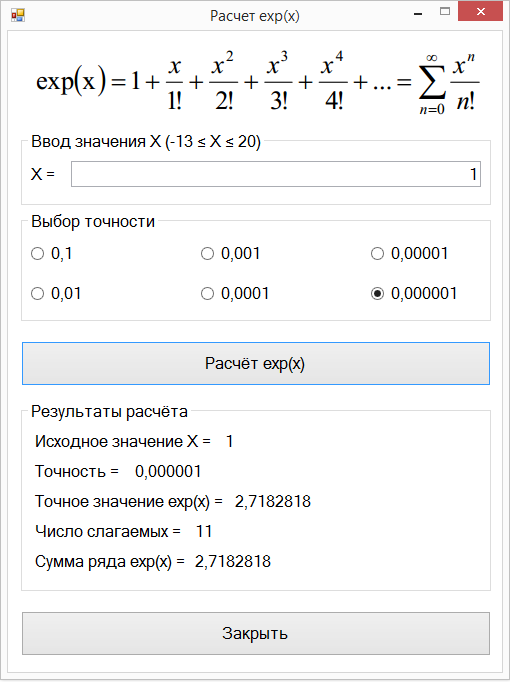


Рисунок 1 – Окно приложения «Расчет значения exp(x)».

1. Программа должна вычислять значение функции exp(x) с точностью до eps = 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001 путём разложения функции в ряд. При этом последнее слагаемое ряда не должно превышать eps.
2. Реализовать введение числового значения переменной **x** с помощью поля типа TextBox, . ограничить ввод букв и специальных символов, при этом для обеспечения ввода вещественных чисел должен корректно вводиться знак десятичного разделителя.
3. С помощью элементов управления RadioButton пользователю будет представлена возможность выбирать значение точности eps.
4. После нажатия на кнопку «Расчёт exp(x)» помимо вычисленного значения функции разложением в ряд должно выводиться также «точное» значение exp(x), вычисляемое с помощью метода Exp(x) класса System::Math, также выводимые значения должны округляться в зависимости от выбранной точности до предлагаемого значения.
5. Программа должна корректно отображать данные, при возникновении ошибок, обрабатывать их.
6. При изменении исходных данных результаты расчёта должны «очищаться».
7. Нажатие на кнопку «Выход» должно корректно завершать работу программы.

# **2 Схемы алгоритмов**

На рисунке 2 изображена схема функции funcLabelsCleanup, используемой для очистки результатов расчётов при изменении входных данных.



Рисунок 2 – Схема алгоритма функции funcLabelsCleanup

На рисунке 3 изображена схема функции txtBoxInputX\_TextChanged, отвечающей за ввод корректных данных в текстовое поле txtBoxInputX.



Рисунок 3.1 – Схема алгоритма функции txtBoxInputX\_TextChanged



Рисунок 3.2 – Схема алгоритма функции txtBoxInputX\_TextChanged



Рисунок 3.3 – Схема алгоритма функции txtBoxInputX\_TextChanged



Рисунок 3.4 – Схема алгоритма функции txtBoxInputX\_TextChanged

На рисунке 4 представлена схема алгоритма функции DigitOutput, который выполняет конвертирование данных из экспоненциальной формы записи в обычную и дописывает незначащие нули в зависимости от выбранной точности.



Рисунок 4 – Схема алгоритма функции DigitOutput

На рисунке 5 представлена схема алгоритма подфункции ActivateRadioButtons, которая активирует тот или иной RadioButton в зависимости от точности.



Рисунок 5 – Схема алгоритма функции ActivateRadioButtons

На рисунке 6 представлена схема алгоритма подфункции ChoiceOfPrecision, которая возвращает значение требуемой точности для проверки функции и нового подсчёта суммы ряда.



Рисунок 6 – Схема алгоритма функции ChoiceOfPrecision

На рисунке 7 представлена схема алгоритма функции funcExponenta, которая считает функцию exp(x) путём разложения функции в ряд. Все расчёты внутри функции выполняются согласно поставленным условиям.



Рисунок 7 – Схема алгоритма функции funcExponenta

На рисунке 8 представлена схема алгоритма функции verifications, который проверяет правильность выбранного значения точности. То есть если exp(x) является очень маленьким числом и выставлена маленькая точность вычислений, чтобы при округлении пользователю не выдавалось число «0», происходит проверка округления и выбор соответствующего значения точности. Из схемы видно, что она использует такие функции, как funcExponenta, ChoiceOfPrecision и ActivateRadioButtons.



Рисунок 8 – Схема алгоритма функции verifications

На рисунках 9 – 14 представлены схемы алгоритмов radioBtn\_CheckedChanged для обработки изменения активного RadioButton.



Рисунок 9 – Схема алгоритма функции radioBtn1\_CheckedChanged



Рисунок 10 – Схема алгоритма функции radioBtn2\_CheckedChanged



Рисунок 11 – Схема алгоритма функции radioBtn3\_CheckedChanged



Рисунок 12 – Схема алгоритма функции radioBtn4\_CheckedChanged



Рисунок 13 – Схема алгоритма функции radioBtn5\_CheckedChanged



Рисунок 14 – Схема алгоритма функции radioBtn6\_CheckedChanged

На рисунке 15 представлена схема алгоритма функции btnAction\_Click, которая выполняет все расчёты при нажатии на кнопку «Расчёт exp(x)».



Рисунок 15 – Схема алгоритма функции btnAction\_Click

На рисунке 16 представлена схема алгоритма функции btnClose\_Click, которая выполняется при нажатии на кнопку «Закрыть» и закрывает программу.



Рисунок 16 – Схема алгоритма функции btnClose\_Click

# **3 Текст программы**

Исходный текст программы представлен в ниже приведённом листинге.

private:

int x1;

double x3;

double precision = 0.1, x = 1; //точность и х

int UsedPrec = 1; //точность для функции Math::Round

System::String^ precisionText = "0,1";

int terms; //число слагаемых

double approximateExp; //сумма ряда

double mathExp; //точное значение через библиотеку

//Защита от дурака

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

if ((!Char::IsDigit(e->KeyChar)) && (e->KeyChar != '-' || txtBoxInputX->Text->IndexOf('-') != -1) && (e->KeyChar != ',' || txtBoxInputX->Text->IndexOf(',') != -1) && (e->KeyChar != (char)Keys::Back) && (e->KeyChar != '.' || txtBoxInputX->Text->IndexOf(',') != -1)) e->Handled = true;

if (e->KeyChar == '.') {

e->KeyChar = ',';

}

if ((e->KeyChar == (char)Keys::D0 || e->KeyChar == (char)Keys::NumPad0) && (txtBoxInputX->Text->Length >= 1) && (txtBoxInputX->Text[0] == '0' && txtBoxInputX->SelectionStart < 2)) e->Handled = true;

if (txtBoxInputX->Text->IndexOf('-') == 0) {

if (txtBoxInputX->SelectionStart == 0) {

e->Handled = true;

}

if ((txtBoxInputX->SelectionStart == 2) && (e->KeyChar == '0')) {

e->Handled = true;

}

if ((txtBoxInputX->SelectionStart == 1) && (e->KeyChar == ',')) {

e->Handled = true;

}

}

if (e->KeyChar == ',') {

x1 = 0;

if (txtBoxInputX->Text != "" && txtBoxInputX->Text[0] == '-') x1 = 1;

if (txtBoxInputX->Text->IndexOf(',') == -1 && txtBoxInputX->SelectionStart == x1) {

txtBoxInputX->Text = txtBoxInputX->Text->Insert(x1, "0");

txtBoxInputX->SelectionStart = x1 + 1;

}

}

if (e->KeyChar == '-' && txtBoxInputX->Text->IndexOf('-') == -1) txtBoxInputX->SelectionStart = 0;

if (txtBoxInputX->Text != (Convert::ToString(x)) && ((txtBoxInputX->Text != "," || txtBoxInputX->Text != "" || txtBoxInputX->Text != "-" || txtBoxInputX->Text != "-,") || txtBoxInputX->Text->IndexOf(',') != 0 || txtBoxInputX->Text->IndexOf('-') != 0)) {

x = 0;

}

else {

x = Convert::ToDouble(txtBoxInputX->Text);

funcLabelsCleanup();

}

}

//Функция очищения результатов

void funcLabelsCleanup() {

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

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

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

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

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

}

//Нормальное форматирование числа в строку

private: System::String^ DigitOutput(double num, int precision) {

String^ str2 = num.ToString("0.#######");

if ((str2->Length) - (str2->IndexOf(",") + 1) < (precision + 1)) {

int n = (precision + 1) - ((str2->Length) - (str2->IndexOf(","))) + 1;

while (n > 0) {

str2 = str2 + "0";

n = n - 1;

}

return str2;

}

else {

return str2;

}

}

//дополнительная подфункция для функции verifications()

//активирует RadioButton в зависимости от выбора точности

private: void ActivateRadioButtons(int precision) {

switch (precision) {

case 2:

radioBtn2->Checked = true;

break;

case 3:

radioBtn3->Checked = true;

break;

case 4:

radioBtn4->Checked = true;

break;

case 5:

radioBtn5->Checked = true;

break;

case 6:

radioBtn6->Checked = true;

break;

default:

MessageBox::Show(this, "Произошла ошибка, введите данные заново!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

break;

}

}

//дополнительная подфункция для функции verifications()

//возвращает точность для округления по х

private: double ChoiceOfPrecision(int precision) {

switch (precision) {

case 2:

return 0.01;

break;

case 3:

return 0.001;

break;

case 4:

return 0.0001;

break;

case 5:

return 0.00001;

break;

case 6:

return 0.000001;

break;

default:

MessageBox::Show(this, "Произошла ошибка, введите данные заново!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

break;

}

}

//Проверка на округление (чтобы exp(x) было не равно нулю)

private: void verifications(double num, int precision) {

if (num == 0) {

while (num == 0) {

precision = precision + 1;

num = Math::Exp(x);

num = Math::Round(num, (precision + 1));

}

double prec = ChoiceOfPrecision(precision);

double MyExp = funcExponenta(x, prec);

MyExp = Math::Round(MyExp, (precision + 1));

approximateExp = MyExp;

mathExp = num;

ActivateRadioButtons(precision);

}

}

//Вычисление суммы ряда

double funcExponenta(double inputX, double inputPrecision) {

double sum = 0, term = 1;

terms = 1;

sum = sum + term;

//Подсчёт суммы ряда в рекуррентном виде

while (fabs(term) > inputPrecision && inputX != 0) {

term = term \* (inputX / terms);

sum = sum + term;

terms = terms + 1;

}

return sum;

}

private: System::Void radioBtn1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (radioBtn1->Checked == true) {

precision = 0.1;

UsedPrec = 1;

precisionText = radioBtn1->Text;

}

else {

funcLabelsCleanup();

}

}

private: System::Void radioBtn2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (radioBtn2->Checked == true) {

precision = 0.01;

UsedPrec = 2;

precisionText = radioBtn2->Text;

}

else {

funcLabelsCleanup();

}

}

private: System::Void radioBtn3\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (radioBtn3->Checked == true) {

precision = 0.001;

UsedPrec = 3;

precisionText = radioBtn3->Text;

}

else {

funcLabelsCleanup();

}

}

private: System::Void radioBtn4\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (radioBtn4->Checked == true) {

precision = 0.0001;

UsedPrec = 4;

precisionText = radioBtn4->Text;

}

else {

funcLabelsCleanup();

}

}

private: System::Void radioBtn5\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (radioBtn5->Checked == true) {

precision = 0.00001;

UsedPrec = 5;

precisionText = radioBtn5->Text;

}

else {

funcLabelsCleanup();

}

}

private: System::Void radioBtn6\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (radioBtn6->Checked == true) {

precision = 0.000001;

UsedPrec = 6;

precisionText = radioBtn6->Text;

}

else {

funcLabelsCleanup();

}

}

//Обработчик кнопки расчета

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

if (txtBoxInputX->Text != "" && txtBoxInputX->Text != "-" && txtBoxInputX->Text != "-0" && txtBoxInputX->Text != "-0," && txtBoxInputX->Text != "-,") {

x = Convert::ToDouble(this->txtBoxInputX->Text);

if (x > 20 || x < -13) {

MessageBox::Show(this, "Введите данные в указанном диапазоне!", "Ошибка!",MessageBoxButtons::OK, MessageBoxIcon::Error, MessageBoxDefaultButton::Button1);

}

else {

approximateExp = funcExponenta(x, precision); //сумма ряда

mathExp = Math::Exp(x); //точное значение

approximateExp = Math::Round(approximateExp, (UsedPrec + 1)); //округление суммы ряда

mathExp = Math::Round(mathExp, (UsedPrec + 1)); //округление точного значения

verifications(mathExp, UsedPrec); //verification (проверка)

//Вывод данных в лейблы

this->lblTheSum->Text = DigitOutput(approximateExp, UsedPrec);

this->lblTerms->Text = Convert::ToString(terms);

this->lblMathExp->Text = DigitOutput(mathExp, UsedPrec);

this->lblInputPrecision->Text = precisionText;

this->lblInputX->Text = Convert::ToString(x);

}

}

else {

MessageBox::Show(this, "Введите корректные данные!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error, MessageBoxDefaultButton::Button1);

}

}

//Обработчик кнопки закрытия приложения

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

System::Windows::Forms::DialogResult result = MessageBox::Show(this, "Вы уверены, что хотите выйти?", "Выход", MessageBoxButtons::YesNo, MessageBoxIcon::Exclamation, MessageBoxDefaultButton::Button2);

if (result == System::Windows::Forms::DialogResult::Yes) {

this->Close();

}

}

# **4 Результаты тестирования программы**

На рисунке 17 представлен интерфейс рабочей программы и её корректная работа.

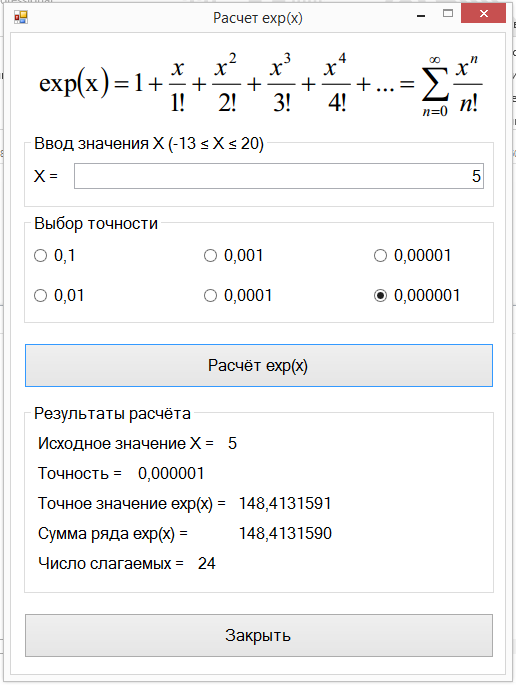


Рисунок 17 – Интерфейс программы

В таблице 1 представлены результаты тестирования программы.

Таблица 1 – Результаты тестирования программы

|  |  |
| --- | --- |
| **Введенные значения** | **Результат** |
| 1) X = 1, точность 0,000001 | Точное значение: 2,7182818  Сумма ряда: 2,7182818  Число слагаемых: 11 |
| 2) Х = 20, точность 0,000001 | Точное значение: 485165195,4097900  Сумма ряда: 485165195,4097900  Число слагаемых: 66 |
| 3) Х = -13, точность 0,000001 | Точное значение: 0,0000023  Сумма ряда: 0,0000023  Число слагаемых: 47 |
| 4) Х = -13, точность 0,01 | Автоматическое переключение на точность 0,00001.  Точное значение: 0,000002  Сумма ряда: 0,000003  Число слагаемых: 45 |
| 5) X = -14, точность 0,000001 | Вывод сообщения «Ошибка! Введите данные в указанном диапазоне!» |
| 6) X = 21, точность 0,000001 | Вывод сообщения «Ошибка! Введите данные в указанном диапазоне!» |
| 7) X = 0, точность 0,000001 | Точное значение: 1,000000  Сумма ряда: 1,000000  Число слагаемых: 1 |
| 8) Х = 0,001, точность 0,000001 | Точное значение: 1,0010005  Сумма ряда: 1, 0010005  Число слагаемых: 3 |
| 9)Х = -0,001, точность 0,000001 | Точное значение: 0,9990005  Сумма ряда: 0,9990005  Число слагаемых: 3 |

На рисунке 18 представлена корректная работа программы при пустом поле ввода значения Х.

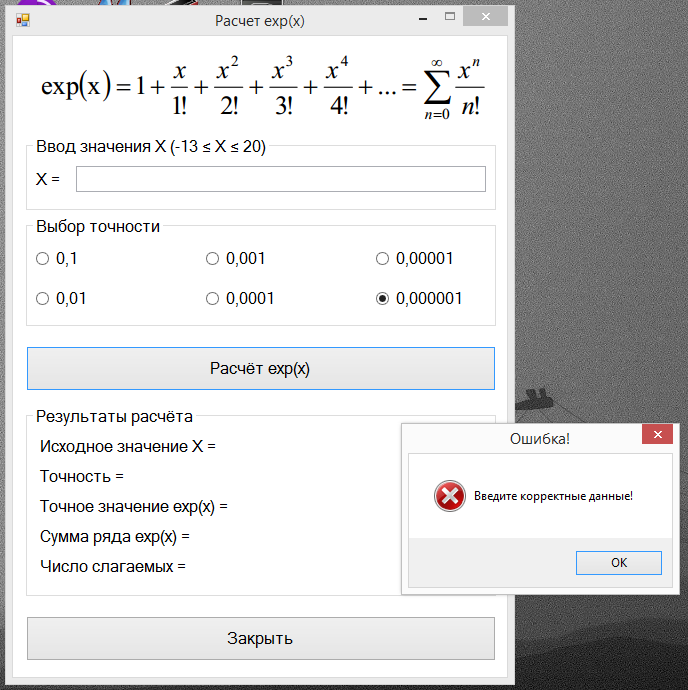


Рисунок 18 – Работа программы при некорректном вводе

## **Вывод**

В ходе данной лабораторной работы были получены практические знания, связанные с циклическими алгоритмами, которые потребовались для создания простейшей программы, вычисляющей сумму ряда функции exp(x). При этом были рассмотрены различные варианты возможной входной информации, предусмотрены некорректные значения вводимых данных.