минобрнауки россии

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| наименование института (факультета) |
| Кафедра математического и программного обеспечения ЭВМ |
| наименование кафедры |
| Теория автоматов и формальных языков |
| наименование дисциплины в соответствии с учебным планом |

ЛАБОРАТОРНАЯ РАБОТА №3\_4

РАСПОЗНАВАНИЕ ЗАДАННЫХ СЛОВ ЯЗЫКА ПРОГРАММИРОВАНИЯ

|  |  |
| --- | --- |
| Исполнитель |  |
| студент | 1ПИб-02-2оп-22 |
|  | группа |
|  | Тихомиров В.В. |
|  | Фамилия, имя, отчество |
| Руководитель | Ганичева О.Г. |
|  | Ф.И.О. преподавателя |
| Оценка |  |
| Подпись |  |

2024 год

# Задание

1. Построить синтаксические диаграммы (конечный автомат) для распознавания заданных слов языка программирования по вариантам:

1- ключевые слова и знаки отношений;

2-идентификаторы и целые константы;

3-константы целые, вещественные и с плавающей точкой;

4-константы с плавающей точкой и ключевые слова;

5-идентификаторы и знаки операций;

6- знаки операций, операторы отношений и разделители;

7- разделители и константы с плавающей точкой

***Применительно к курсовой работе в качестве заданных слов берутся все слова из программного кода, который был написан в работе 1 , в соответствие с вариантом задания. - это доработка кода***

2. Написать программу для распознавания   слов языка С++ по своему варианту задания (ключевые слова, идентификаторы, знаки операций, знаки пунктуации, целые константы, вещественные константы, числа с фиксированной и плавающей точкой...).  Правильные слова должны заноситься в таблицу. На словах с ошибкой - выдаваться сообщение об ошибке.

3. Составить блок-схему алгоритма решения задачи.

4. Оформить отчет (задание, блок-схема алгоритма, код программы с комментариями, результаты тестирования программы (не менее 10 тестовых заданий), скриншоты результатов)

# Ход работы

Конечный автомат для задания:

Константы целые, вещественные и с плавающей точкой

Формальное описание конечного автомата:

А = {X, S, S0, F, δ}

X = {+, -, число, v, e, E, .}

S = {S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12}

S0 = {S0}

F = {S1, S3, S6}

δ:

|  |
| --- |
| (S0, -) → S1 |
| (S0, Число) → S1 |
| (S0, /) → S8 |
| (S0, Другой) → S2 |
| (S1, Число) → S1 |
| (S1, ') → S3 |
| (S1, Пробел ; : ,) → S0 |
| (S1, Другой) → S7 |
| (S2, Пробел) → S0 |
| (S2, Другой) → S2 |
| (S3, Число) → S3 |
| (S3, Пробел ; : ,) → S0 |
| (S3, e/E) → S4 |
| (S3, Другой) → S7 |
| (S4, Число) → S6 |
| (S4, +/-) → S5 |
| (S4, Другой) → S7 |
| (S5, Число) → S6 |
| (S5, Другой) → S7 |
| (S6, Число) → S6 |
| (S6, Пробел ; : ,) → S0 |
| (S6, Другой) → S0 |
| (S7, Любой) → S0 |
| (S8, /) → S9 |
| (S8, \*) → S10 |
| (S9, \n) → S0 |
| (S9, Другой) → S9 |
| (S10, \*) → S11 |
| (S10, Другой) → S10 |
| (S11, /) → S12 |
| (S11, Другой) → S10 |

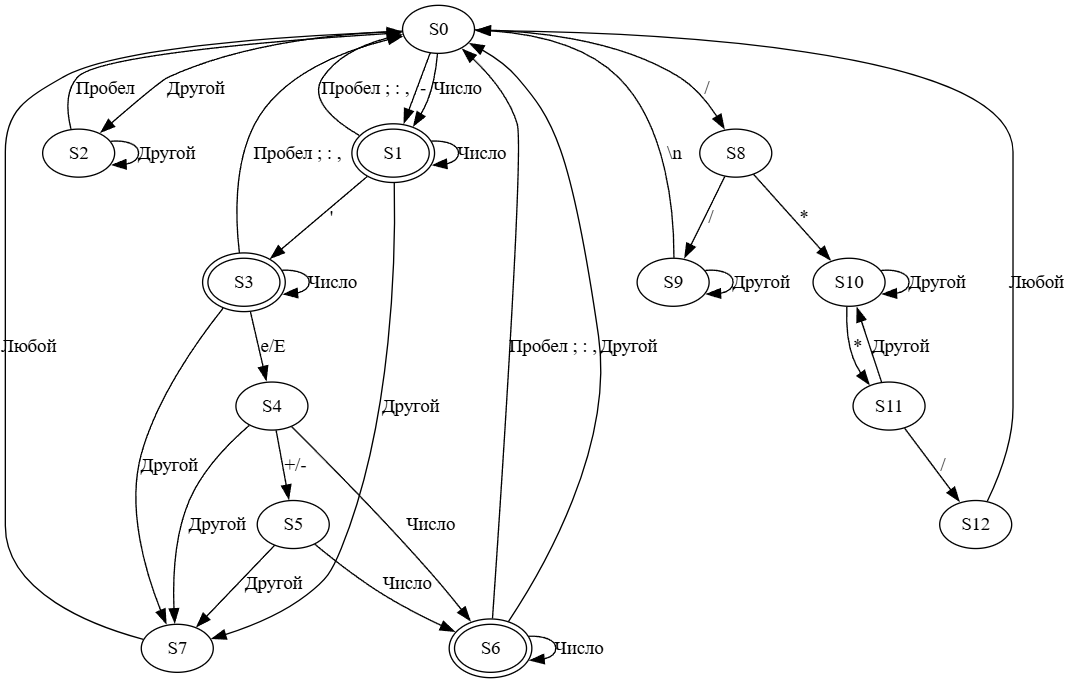


Рис. 1. Граф конечного автомата

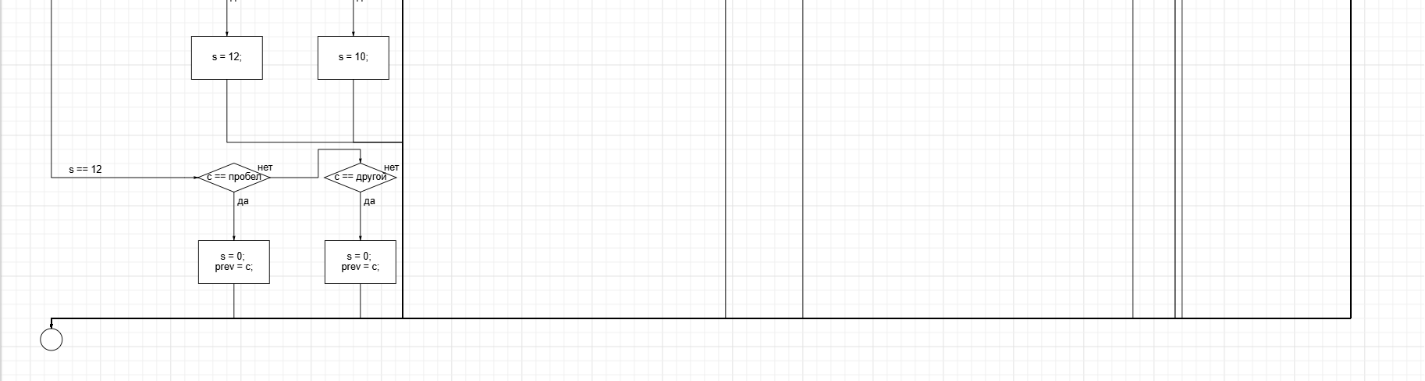
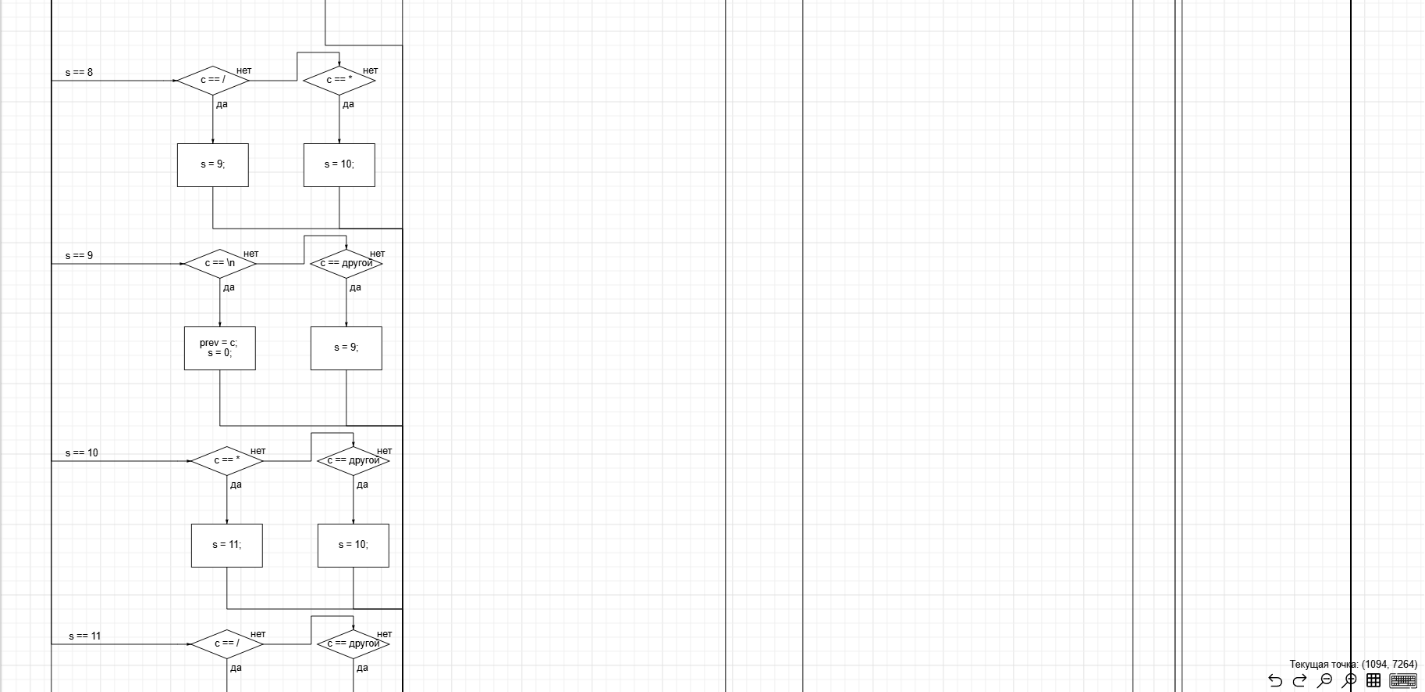
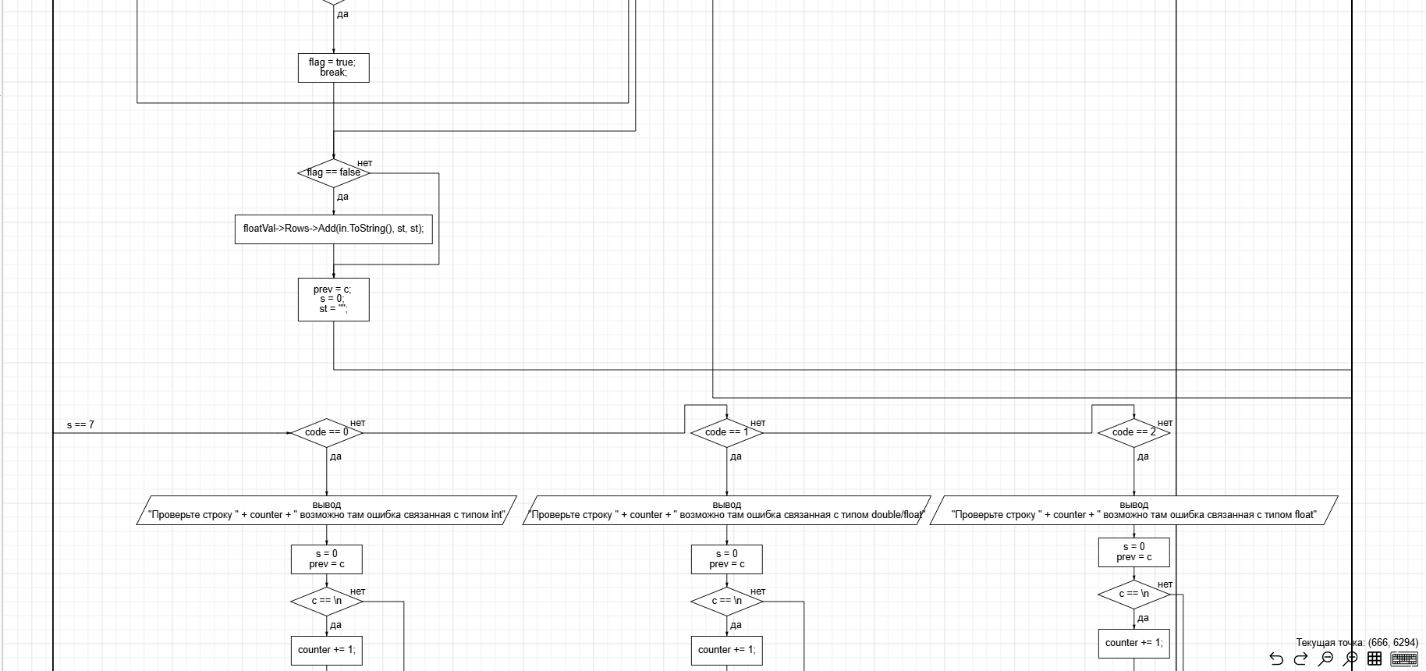
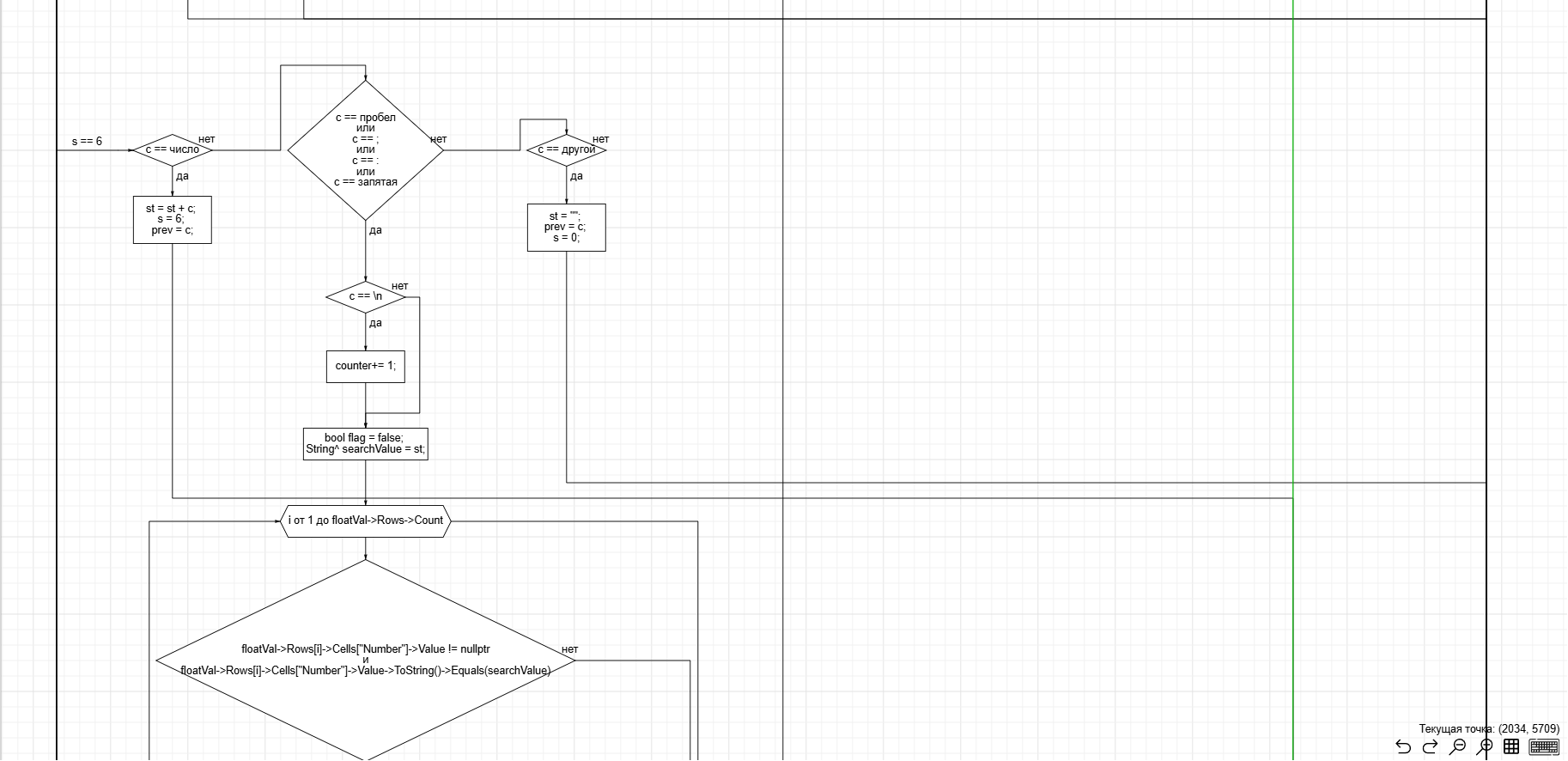
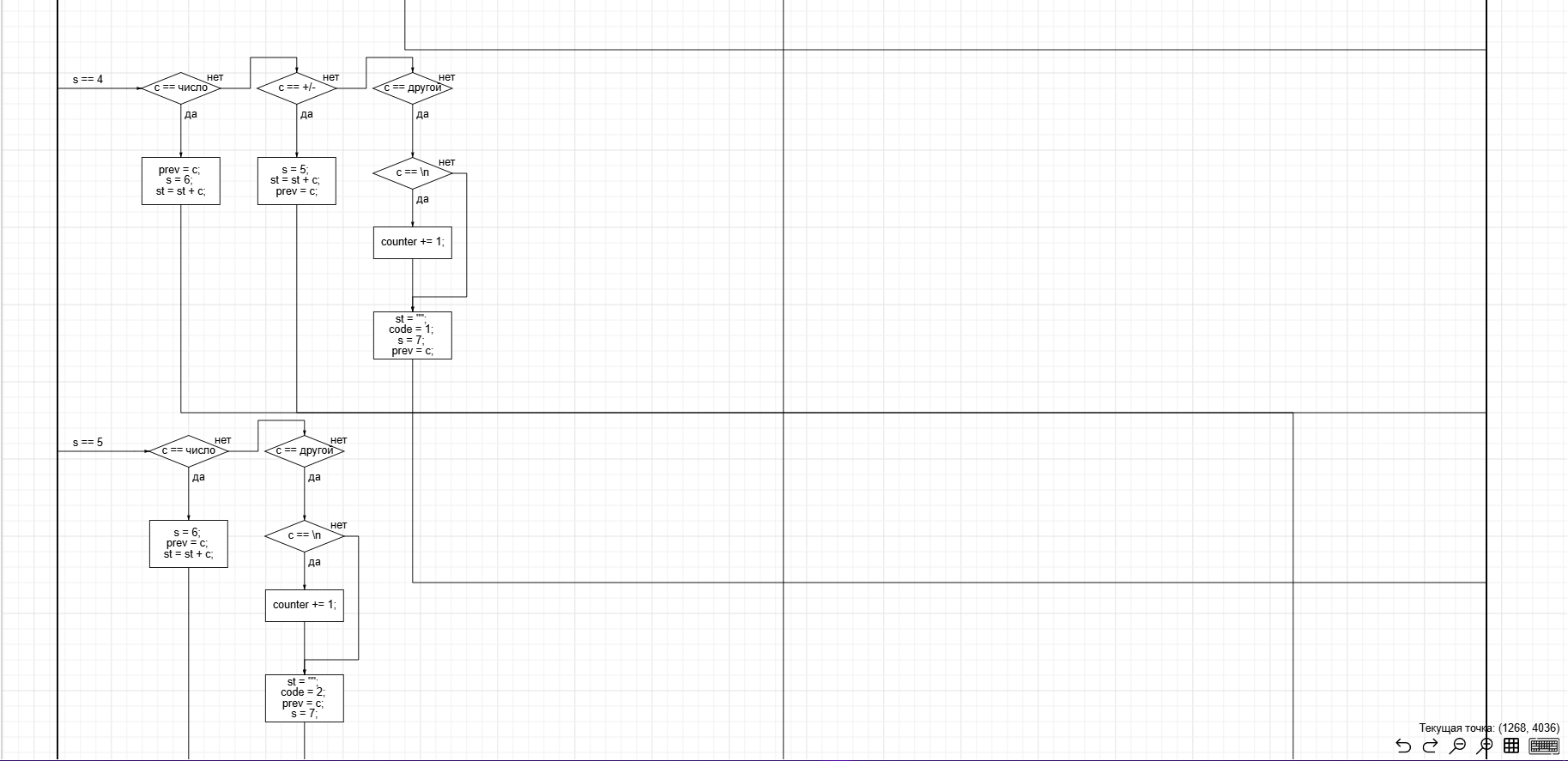
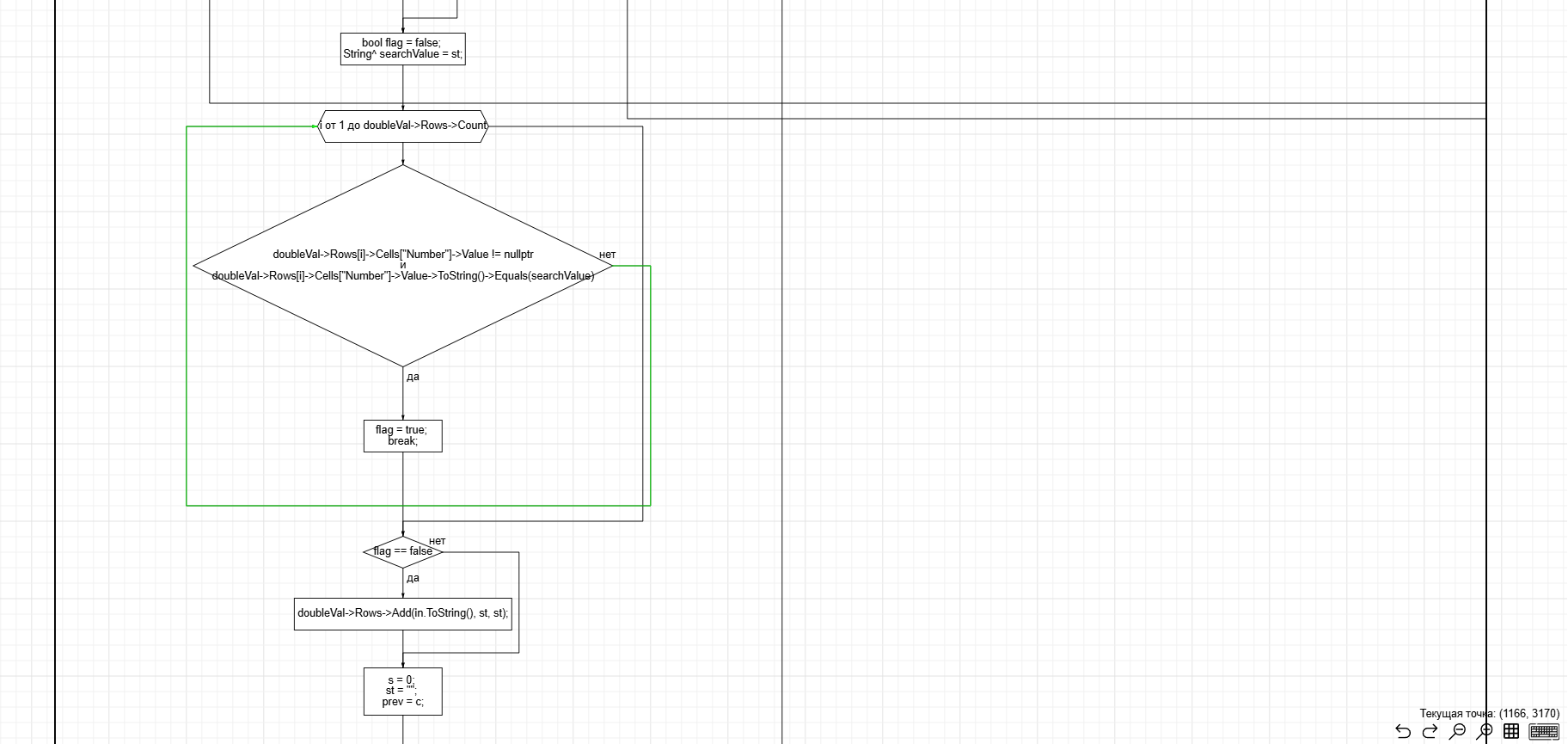
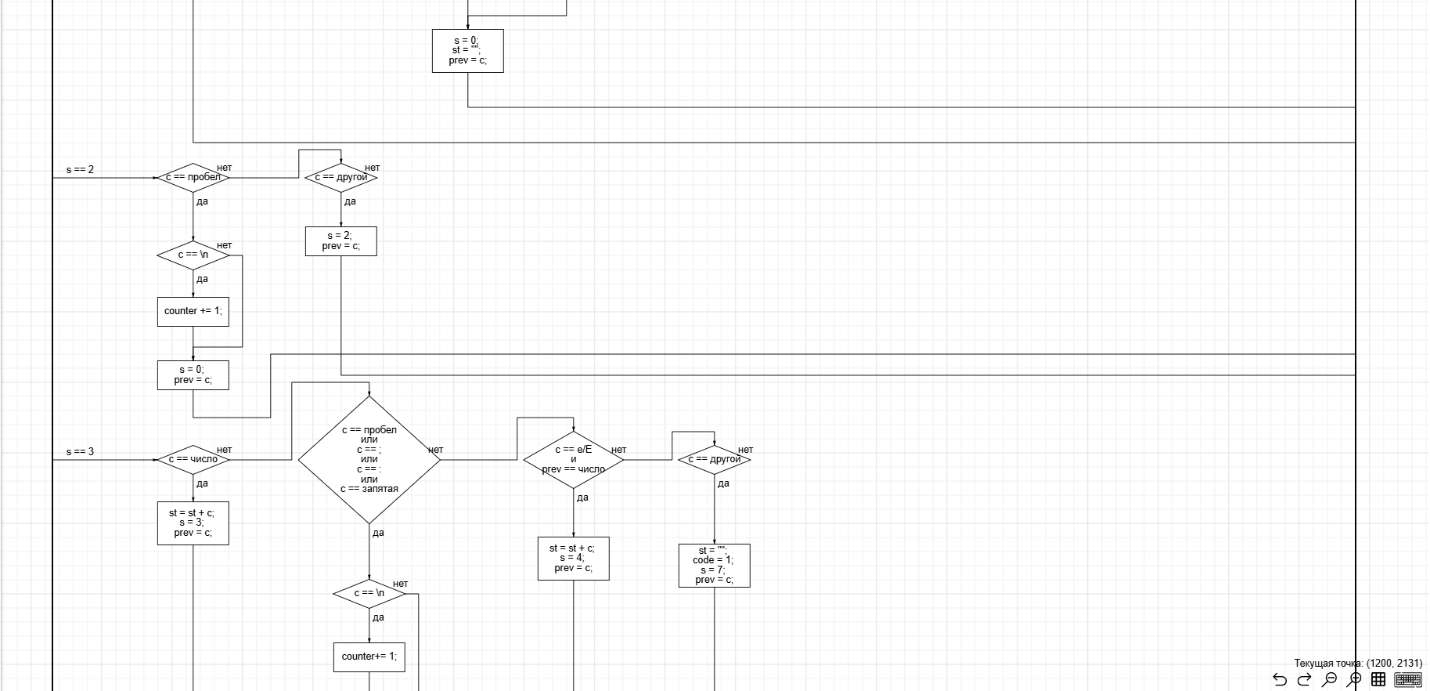
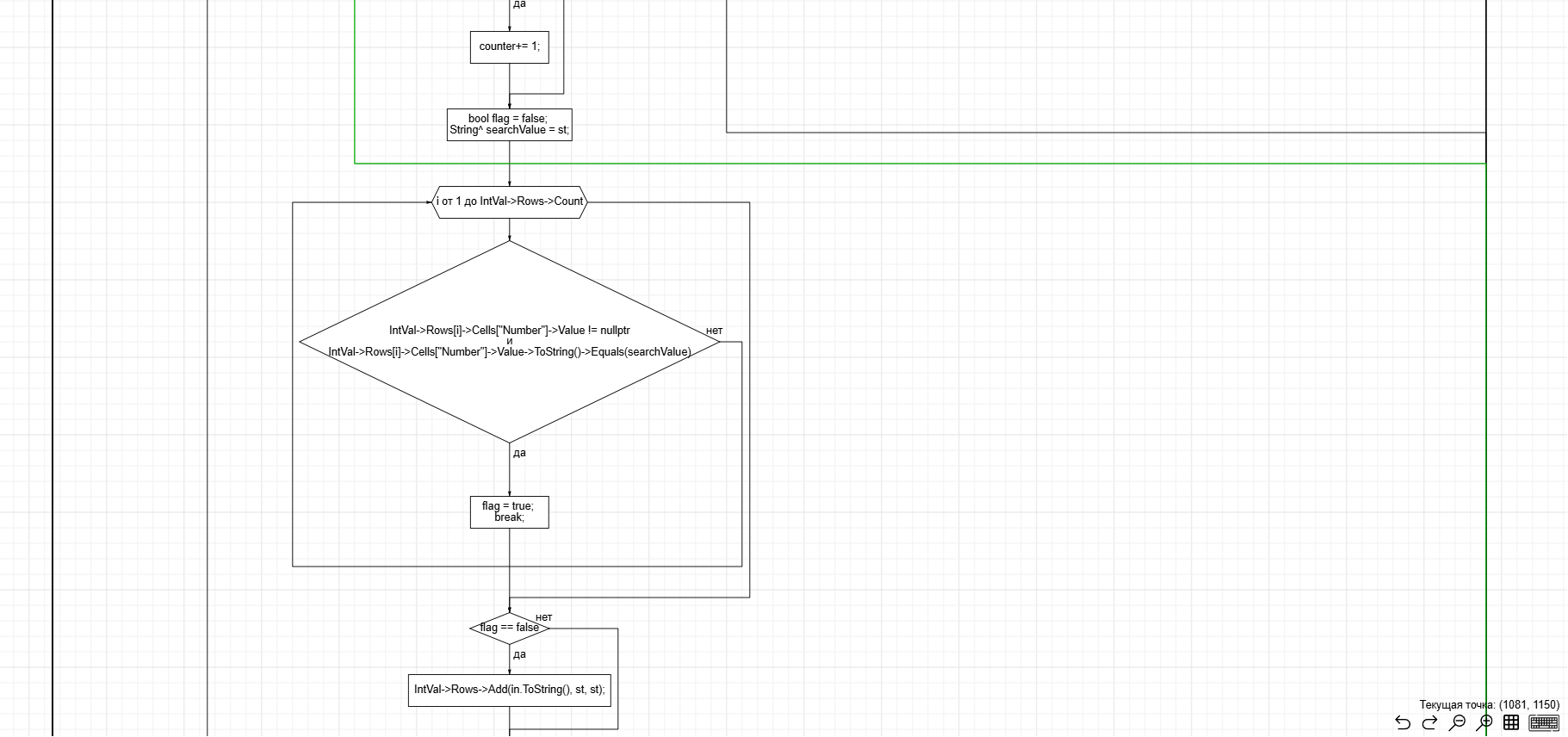
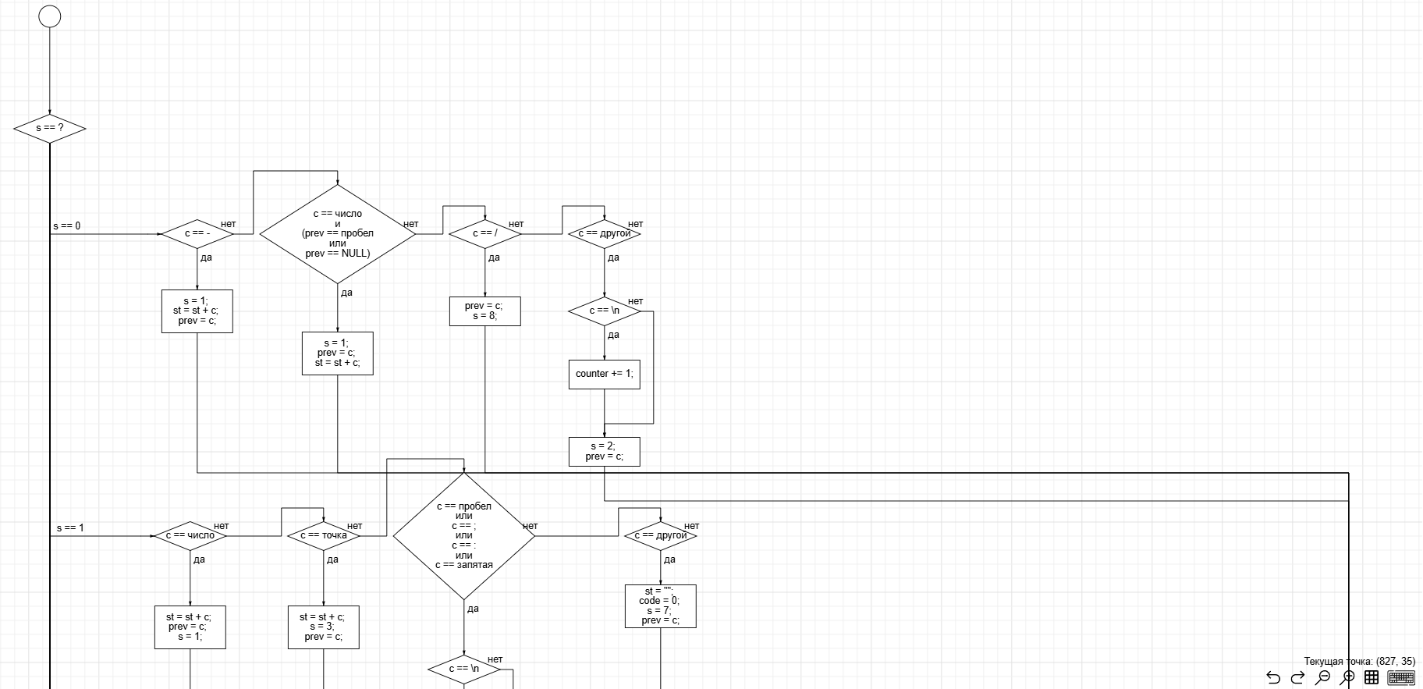


Рис. 2. Блок схема switch

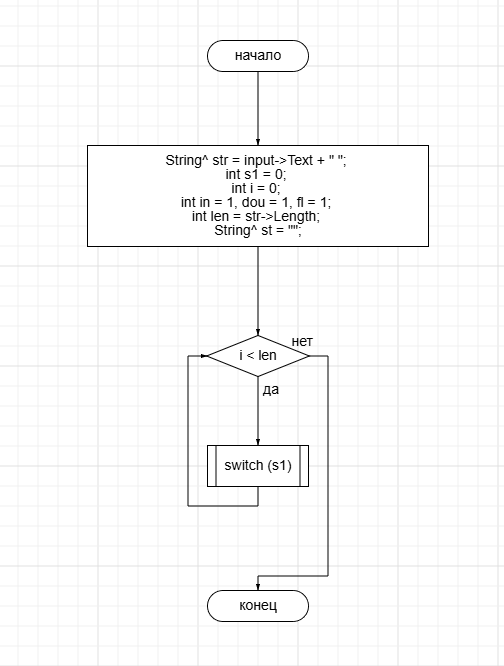


Рис. 3. Блок-схема алгоритма работы программы

Входной код

#include <iostream>

using namespace std;

void main() {

// инициализируем переменные

int d = 21;

int ds = 13;

double s = 54.32;

double ss = 2.1;

float a = 2.3e+1;

float f = 4.3e-1;

// выводим их значение на экран

for (int i = 0; i < 3; i++) {

switch (i)

{

case 0:

cout << d << endl;

cout << d + ds << endl;

break;

case 1:

cout << s << endl;

cout << s + ss << endl;

break;

case 2:

cout << a << endl;

cout << a + f << endl;

break;

default:

cout << "Error" << endl;

break;

}

}

}

Тесты программы

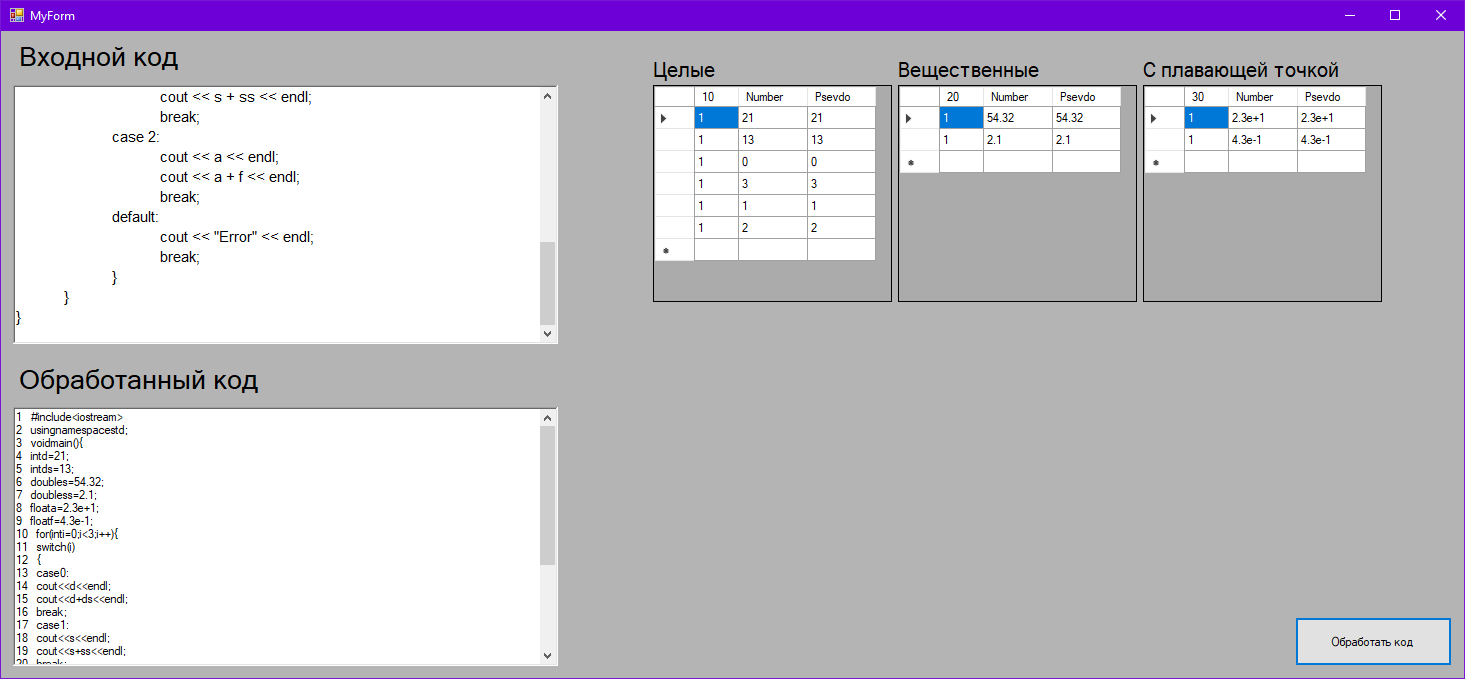


Рис. 4. Тест работы программы

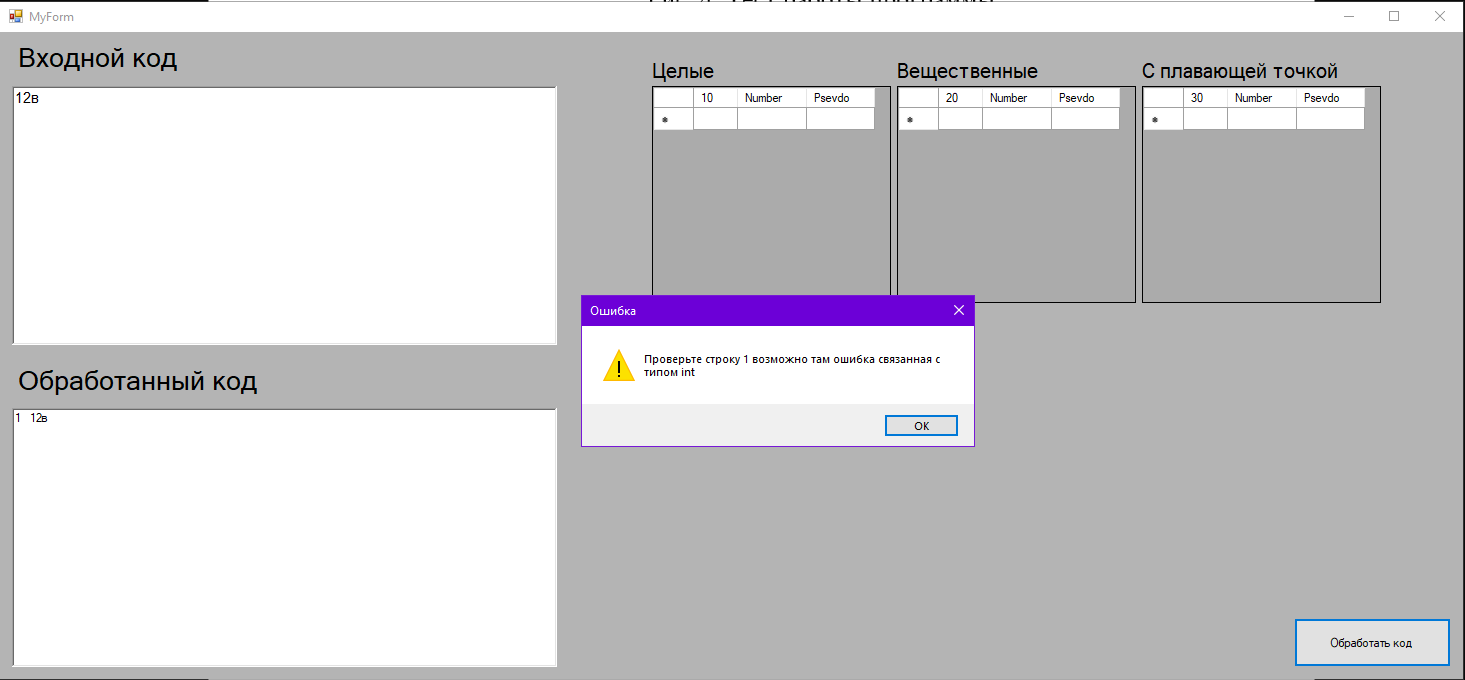


Рис. 5. Тест обработчика ошибок

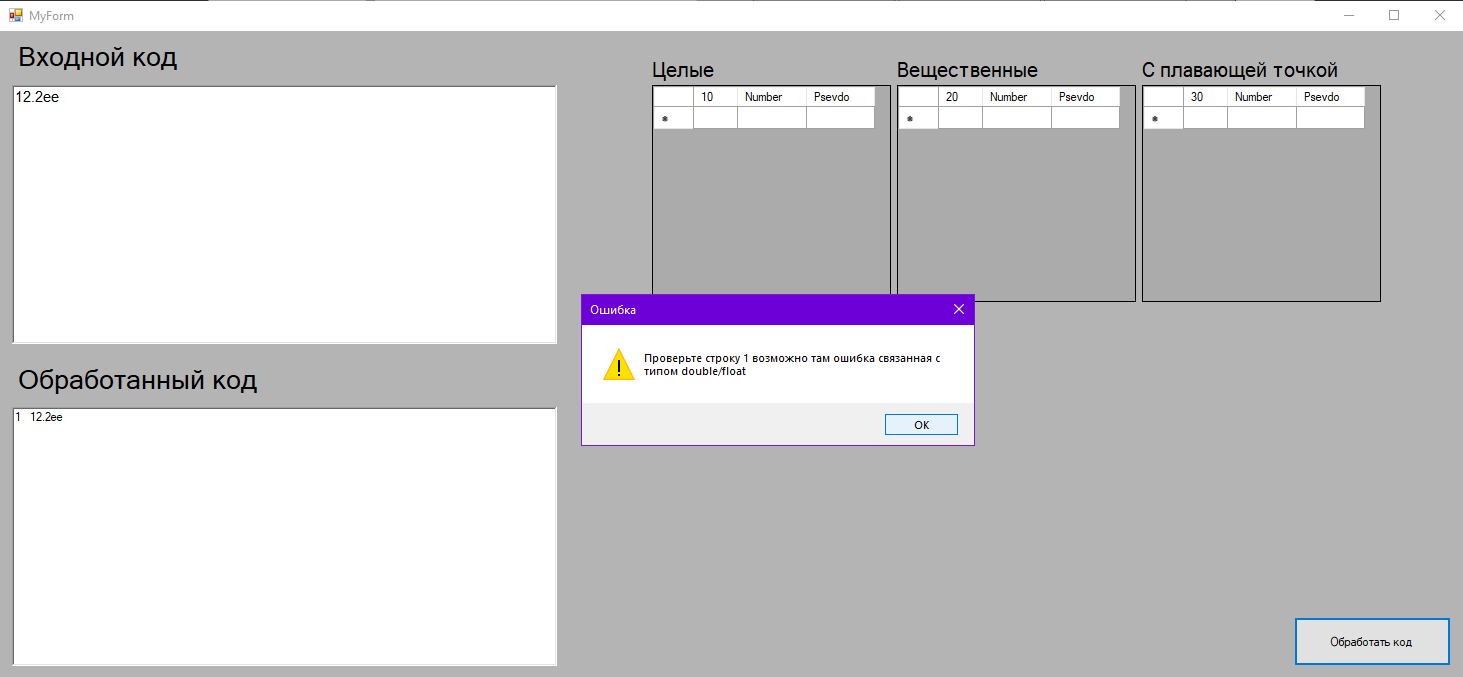


Рис. 6. Тест обработчика других ошибок

Код программы

#pragma once

#include <cctype>

namespace Project1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::RichTextBox^ input;

private: System::Windows::Forms::RichTextBox^ output;

protected:

protected:

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::DataGridView^ IntVal;

private: System::Windows::Forms::DataGridView^ floatVal;

private: System::Windows::Forms::DataGridView^ doubleVal;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->input = (gcnew System::Windows::Forms::RichTextBox());

this->output = (gcnew System::Windows::Forms::RichTextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->IntVal = (gcnew System::Windows::Forms::DataGridView());

this->floatVal = (gcnew System::Windows::Forms::DataGridView());

this->doubleVal = (gcnew System::Windows::Forms::DataGridView());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->IntVal))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->floatVal))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->doubleVal))->BeginInit();

this->SuspendLayout();

//

// input

//

this->input->Location = System::Drawing::Point(12, 54);

this->input->Name = L"input";

this->input->Size = System::Drawing::Size(545, 259);

this->input->TabIndex = 0;

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

//

// output

//

this->output->Location = System::Drawing::Point(12, 376);

this->output->Name = L"output";

this->output->Size = System::Drawing::Size(545, 259);

this->output->TabIndex = 1;

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

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 20, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(12, 9);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(173, 31);

this->label1->TabIndex = 2;

this->label1->Text = L"Входной код";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 20, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(12, 332);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(253, 31);

this->label2->TabIndex = 3;

this->label2->Text = L"Обработанный код";

//

// button1

//

this->button1->Location = System::Drawing::Point(1294, 586);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(157, 49);

this->button1->TabIndex = 4;

this->button1->Text = L"Обработать код";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// IntVal

//

this->IntVal->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->IntVal->Location = System::Drawing::Point(652, 54);

this->IntVal->Name = L"IntVal";

this->IntVal->Size = System::Drawing::Size(239, 217);

this->IntVal->TabIndex = 5;

//

// floatVal

//

this->floatVal->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->floatVal->Location = System::Drawing::Point(1142, 54);

this->floatVal->Name = L"floatVal";

this->floatVal->Size = System::Drawing::Size(239, 217);

this->floatVal->TabIndex = 6;

//

// doubleVal

//

this->doubleVal->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->doubleVal->Location = System::Drawing::Point(897, 54);

this->doubleVal->Name = L"doubleVal";

this->doubleVal->Size = System::Drawing::Size(239, 217);

this->doubleVal->TabIndex = 7;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(647, 26);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(74, 25);

this->label3->TabIndex = 8;

this->label3->Text = L"Целые";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(892, 26);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(153, 25);

this->label4->TabIndex = 9;

this->label4->Text = L"Вещественные";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(1137, 26);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(208, 25);

this->label5->TabIndex = 10;

this->label5->Text = L"С плавающей точкой";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::SystemColors::ActiveBorder;

this->ClientSize = System::Drawing::Size(1463, 647);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->doubleVal);

this->Controls->Add(this->floatVal);

this->Controls->Add(this->IntVal);

this->Controls->Add(this->button1);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->output);

this->Controls->Add(this->input);

this->Name = L"MyForm";

this->Text = L"MyForm";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->IntVal))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->floatVal))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->doubleVal))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

// Очистим поле для обратного кода

output->Clear();

// Получим текст из поля записи

String^ text = input->Text;

text = text + " ";

// Создадим переменную для хранения состояний

int s = 0;

bool closed = 1;

int counter = 1;

output->AppendText(counter.ToString() + " ");

counter++;

char prev = '/';

// Обойдём каждый символ кода

for each (Char c in text) {

switch (s)

{

case 0:

if (c == '\n') { // если c - это переход на следующую строку

if (c != prev && prev != '/') {

output->AppendText(c.ToString());

output->AppendText(counter.ToString() + " ");

counter++;

}

prev = c;

s = 7;

}

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != '\t' && c != '\*') {// если символ - это буква

s = 0;

output->AppendText(c.ToString());

prev = c;

}

if (c == '/') {// если с = /

s = 1;

prev = c;

}

if (c == ' ') { // если c - это пробел

s = 5;

prev = c;

}

if (c == '\*') {

s = 0;

output->AppendText(c.ToString());

prev = c;

}

if (c == '\t') {

s = 6;

prev = c;

}

break;

case 1:

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') { // если c - это символ

s = 0;

output->AppendText("/");

output->AppendText(c.ToString());

prev = c;

}

else if (c == '/') {

s = 2;

if (c != prev) {

output->AppendText(c.ToString());

}

/\*output->AppendText("\n");

output->AppendText(counter.ToString() + " ");\*/

prev = c;

}

else if (c == '\*') {

if (prev != '/') {

output->AppendText("/");

}

s = 3;

prev = c;

}

else if (c == '\n') {

s = 0;

if (c != prev) {

output->AppendText("/");

output->AppendText(c.ToString());

output->AppendText(counter.ToString() + " ");

counter++;

}

prev = c;

}

else if (c == '\t') {

output->AppendText("/");

s = 0;

prev = c;

}

else if (' ') {

output->AppendText("/");

s = 0;

prev = c;

}

break;

case 2:

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') {

s = 2;

prev = c;

}

else if (c == '\n') {

s = 0;

}

else if (c == '\*') {

s = 2;

prev = c;

}

else if (c == '\t') {

s = 2;

}

else if (c == '/') {

s = 2;

prev = c;

}

else if (c == ' ') {

s = 2;

prev = c;

}

break;

case 3:

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') {

s = 3;

prev = c;

}

else if (c == ' ') {

s = 3;

prev = c;

}

else if ('\t') {

s = 3;

prev = c;

}

else if ('\n') {

s = 3;

if (c != prev) {

counter++;

}

prev = c;

}

else if (c == '/') {

s = 3;

prev = c;

}

if (c == '\*') {

s = 4;

prev = c;

}

break;

case 4:

if (c == '/') {

s = 0;

prev = c;

}

else if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') {

s = 3;

prev = c;

}

else if (c == ' ') {

s = 3;

prev = c;

}

else if (c == '\t') {

s = 3;

prev = c;

}

else if (c == '\n') {

s = 3;

if (c != prev) {

counter++;

}

prev = c;

}

else if (c == '\*') {

s = 3;

prev = c;

}

break;

case 5:

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') {

s = 0;

output->AppendText(c.ToString());

prev = c;

}

else if (c == '\t') {

s = 6;

prev = c;

}

else if (c == '\n') {

s = 7;

prev = c;

if (c != prev) {

output->AppendText(c.ToString());

output->AppendText(counter.ToString() + " ");

counter++;

}

}

else if (c == ' ') {

s = 5;

prev = c;

}

else if (c == '\*') {

s = 5;

output->AppendText(c.ToString());

prev = c;

}

else if (c == '/') {

s = 1;

prev = c;

}

break;

case 6:

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') {

s = 0;

output->AppendText(c.ToString());

prev = c;

}

else if (c == '\t') {

s = 6;

prev = c;

}

else if (c == '\n') {

s = 7;

if (c != prev) {

output->AppendText(c.ToString());

output->AppendText(counter.ToString() + " ");

counter++;

}

prev = c;

}

else if (c == ' ') {

s = 5;

prev = c;

}

else if (c == '\*') {

s = 6;

output->AppendText(c.ToString());

prev = c;

}

else if (c == '/') {

s = 1;

prev = c;

}

break;

case 7:

if (!Char::IsWhiteSpace(c) && c != '\n' && c != '/' && c != '\r' && c != 't' && c != '\*') {

s = 0;

output->AppendText(c.ToString());

prev = c;

}

else if (c == '\t') {

s = 6;

prev = c;

}

else if (c == '\n') {

s = 7;

if (c != prev) {

output->AppendText(c.ToString());

output->AppendText(counter.ToString() + " ");

counter++;

}

prev = c;

}

else if (c == '\*') {

s = 7;

output->AppendText(c.ToString());

prev = c;

}

else if (c == ' ') {

s = 5;

prev = c;

}

else if (c == '/') {

s = 1;

prev = c;

}

break;

default:

break;

}

}

s = 0;

IntVal->Rows->Clear();

if (IntVal->Columns->Count == 0) {

DataGridViewTextBoxColumn^ column1 = gcnew DataGridViewTextBoxColumn();

column1->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column1->Name = "10";

IntVal->Columns->Add(column1);

DataGridViewTextBoxColumn^ column2 = gcnew DataGridViewTextBoxColumn();

column2->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column2->Name = "Number";

IntVal->Columns->Add(column2);

DataGridViewTextBoxColumn^ column3 = gcnew DataGridViewTextBoxColumn();

column3->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column3->Name = "Psevdo";

IntVal->Columns->Add(column3);

}

doubleVal->Rows->Clear();

if (doubleVal->Columns->Count == 0) {

DataGridViewTextBoxColumn^ column1 = gcnew DataGridViewTextBoxColumn();

column1->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column1->Name = "20";

doubleVal->Columns->Add(column1);

DataGridViewTextBoxColumn^ column2 = gcnew DataGridViewTextBoxColumn();

column2->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column2->Name = "Number";

doubleVal->Columns->Add(column2);

DataGridViewTextBoxColumn^ column3 = gcnew DataGridViewTextBoxColumn();

column3->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column3->Name = "Psevdo";

doubleVal->Columns->Add(column3);

}

floatVal->Rows->Clear();

if (floatVal->Columns->Count == 0) {

DataGridViewTextBoxColumn^ column1 = gcnew DataGridViewTextBoxColumn();

column1->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column1->Name = "30";

floatVal->Columns->Add(column1);

DataGridViewTextBoxColumn^ column2 = gcnew DataGridViewTextBoxColumn();

column2->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column2->Name = "Number";

floatVal->Columns->Add(column2);

DataGridViewTextBoxColumn^ column3 = gcnew DataGridViewTextBoxColumn();

column3->AutoSizeMode = DataGridViewAutoSizeColumnMode::AllCells;

column3->Name = "Psevdo";

floatVal->Columns->Add(column3);

}

prev = NULL;

int code = 0;

counter = 1;

int in = 1, dou = 1, fl = 1;

String^ st = "";

for each (Char c in text) {

switch (s)

{

case 0:// проверка на число

if (c == '-') {

s = 1;

st = st + c;

prev = c;

}

else if ((c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6'

|| c == '7' || c == '8' || c == '9') && (Char::IsWhiteSpace(prev) || prev == NULL)) {

s = 1;

prev = c;

st = st + c;

}

else if (c == '/') {

prev = c;

s = 8;

}

else {

if (c == '\n') {

counter += 1;

}

s = 2;

prev = c;

}

break;

case 1:

if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6'

|| c == '7' || c == '8' || c == '9') {

st = st + c;

prev = c;

s = 1;

}

else if (c == '.') {

st = st + c;

s = 3;

prev = c;

}

else if (Char::IsWhiteSpace(c) || c == ';' || c == ':' || c == ',') {

if (c == '\n') {

counter += 1;

}

bool flag = false;

String^ searchValue = st;

for (int i = 0; i < IntVal->Rows->Count; i++) {

if (IntVal->Rows[i]->Cells["Number"]->Value != nullptr && IntVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue)) {

flag = true;

break;

}

}

if (!flag) {

IntVal->Rows->Add(in.ToString(), st, st);

}

s = 0;

st = "";

prev = c;

}

else {

st = "";

code = 0;

s = 7;

prev = c;

}

break;

case 2:

if (Char::IsWhiteSpace(c)) {

if (c == '\n') {

counter += 1;

}

s = 0;

prev = c;

}

else {

s = 2;

prev = c;

}

break;

case 3:

if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6'

|| c == '7' || c == '8' || c == '9') {

st = st + c;

s = 3;

prev = c;

}

else if (Char::IsWhiteSpace(c) || c == ';' || c == ':' || c == ',') {

if (c == '\n') {

counter += 1;

}

if (prev != '.') {

bool flag = false;

String^ searchValue = st;

for (int i = 0; i < doubleVal->Rows->Count; i++) {

if (doubleVal->Rows[i]->Cells["Number"]->Value != nullptr && doubleVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue)) {

flag = true;

break;

}

}

if (!flag) {

doubleVal->Rows->Add(in.ToString(), st, st);

}

}

prev = c;

s = 0;

st = "";

}

else if ((c == 'e' || c == 'E') && (prev == '0' || prev == '1' || prev == '2' || prev == '3' || prev == '4' || prev == '5' || prev == '6'

|| prev == '7' || prev == '8' || prev == '9')) {

st = st + c;

s = 4;

prev = c;

}

else {

st = "";

code = 1;

s = 7;

prev = c;

}

break;

case 4:

if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6'

|| c == '7' || c == '8' || c == '9') {

prev = c;

s = 6;

st = st + c;

}

else if (c == '+' || c == '-') {

s = 5;

st = st + c;

prev = c;

}

else {

if (c == '\n') {

counter += 1;

}

st = "";

code = 1;

s = 7;

prev = c;

}

break;

case 5:

if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6'

|| c == '7' || c == '8' || c == '9') {

s = 6;

prev = c;

st = st + c;

}

else {

if (c == '\n') {

counter += 1;

}

st = "";

code = 2;

prev = c;

s = 7;

}

break;

case 6:

if (c == '0' || c == '1' || c == '2' || c == '3' || c == '4' || c == '5' || c == '6'

|| c == '7' || c == '8' || c == '9') {

st = st + c;

s = 6;

prev = c;

}

else if (Char::IsWhiteSpace(c) || c == ';' || c == ':' || c == ',') {

if (c == '\n') {

counter += 1;

}

bool flag = false;

String^ searchValue = st;

for (int i = 0; i < floatVal->Rows->Count; i++) {

if (floatVal->Rows[i]->Cells["Number"]->Value != nullptr && floatVal->Rows[i]->Cells["Number"]->Value->ToString()->Equals(searchValue)) {

flag = true;

break;

}

}

if (!flag) {

floatVal->Rows->Add(in.ToString(), st, st);

}

prev = c;

s = 0;

st = "";

}

else {

st = "";

prev = c;

s = 0;

}

break;

case 7:

if (code == 0) {

MessageBox::Show("Проверьте строку " + counter + " возможно там ошибка связанная с типом int", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

s = 0;

if (c == '\n') {

counter += 1;

}

prev = c;

}

if (code == 1) {

MessageBox::Show("Проверьте строку " + counter + " возможно там ошибка связанная с типом double/float", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

s = 0;

if (c == '\n') {

counter += 1;

}

prev = c;

}

if (code == 2) {

MessageBox::Show("Проверьте строку " + counter + " возможно там ошибка связанная с типом float", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Warning);

s = 0;

if (c == '\n') {

counter += 1;

}

prev = c;

}

break;

case 8:

if (c == '/') {

s = 9;

}

else if (c == '\*') {

s = 10;

}

break;

case 9:

if (c == '\n') {

prev = c;

s = 0;

}

else {

s = 9;

}

break;

case 10:

if (c == '\*') {

s = 11;

}

else {

s = 10;

}

break;

case 11:

if (c == '/') {

//prev = ' ';

s = 12;

}

else {

s = 10;

}

break;

case 12:

if (Char::IsWhiteSpace(c)) {

s = 0;

prev = c;

}

else {

s = 0;

prev = c;

}

break;

default:

break;

}

}

}

};

}

# Заключение

Лабораторная работа была посвящена расширению функциональности лексического анализатора. В работе описаны компоненты автомата, построена блок-схема и граф, наглядно иллюстрирующий его функционирование. Также было разработано визуальное приложение, представляющее автомат в виде программы, осуществляющей распознавание элементов входного кода согласно варианту.

Основной функционал программы был проверен на работоспособность, что подтверждено скриншотом программы и ее результатов.

В процессе выполнения работы были изучены основы лексического анализа, а также применены навыки программирования на языке C++. Была разработана и успешно протестирована функция лексического анализатора на различных примерах, что позволяет утверждать о ее корректной работе.