МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИИТ. ПРОГРАМНАЯ ИНЖЕНЕРИЯ

ЛАБОРАТОРНАЯ РАБОТА №1-2

Дополнительные функции лексического анализатора

Исполнитель:

Студент Зернов В.А.
Группа 1ПИб-02-2оп-22

Руководитель:

Ганичева Оксана Георгиевна
Ф.И.О. преподавателя

Оценка
Подпись

2024 год

Задание

1. Написать программный код для своего варианта задания на 20-25 строк (допустимо. до 30 строк)

2. Проверить его работоспособность. В качестве доказательства сделать скриншот программы и результатов. Результат должен выводиться в оформленном виде.

3. Написать функцию лексического анализатора, выполняющую следующие действия: (работу этой функции проверять на примере написанного в п.1 рабочего кода)

1) удаление лишних пробелов во входном коде;

2) удаление комментариев из текста программы;

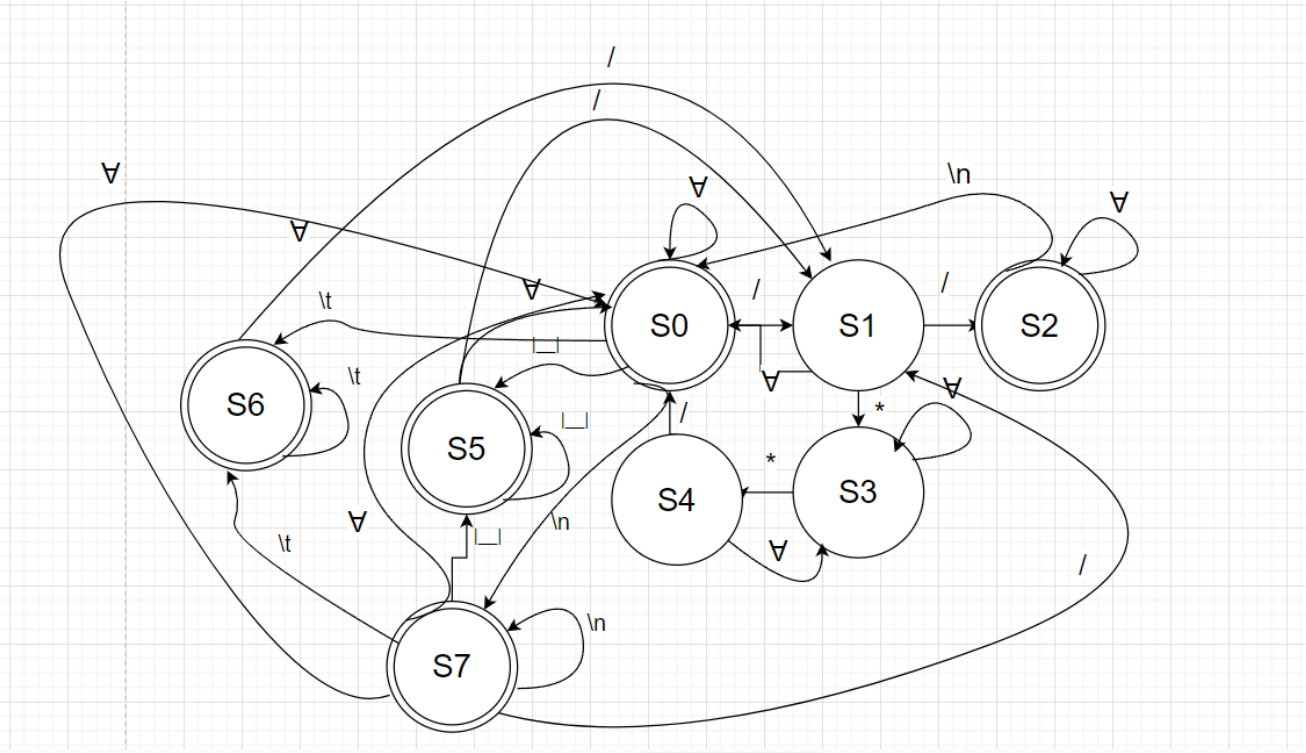
3) подсчет количества строк во входном тексте.

4) Составить блок-схему для этой функции.

4. Протестировать работу функции на других примерах.

Примечание: в отчет по работе включать все с п.1 по 3 и код программы с функцией п.3. Код программы должен быть с комментариями.

Граф конечного автомата



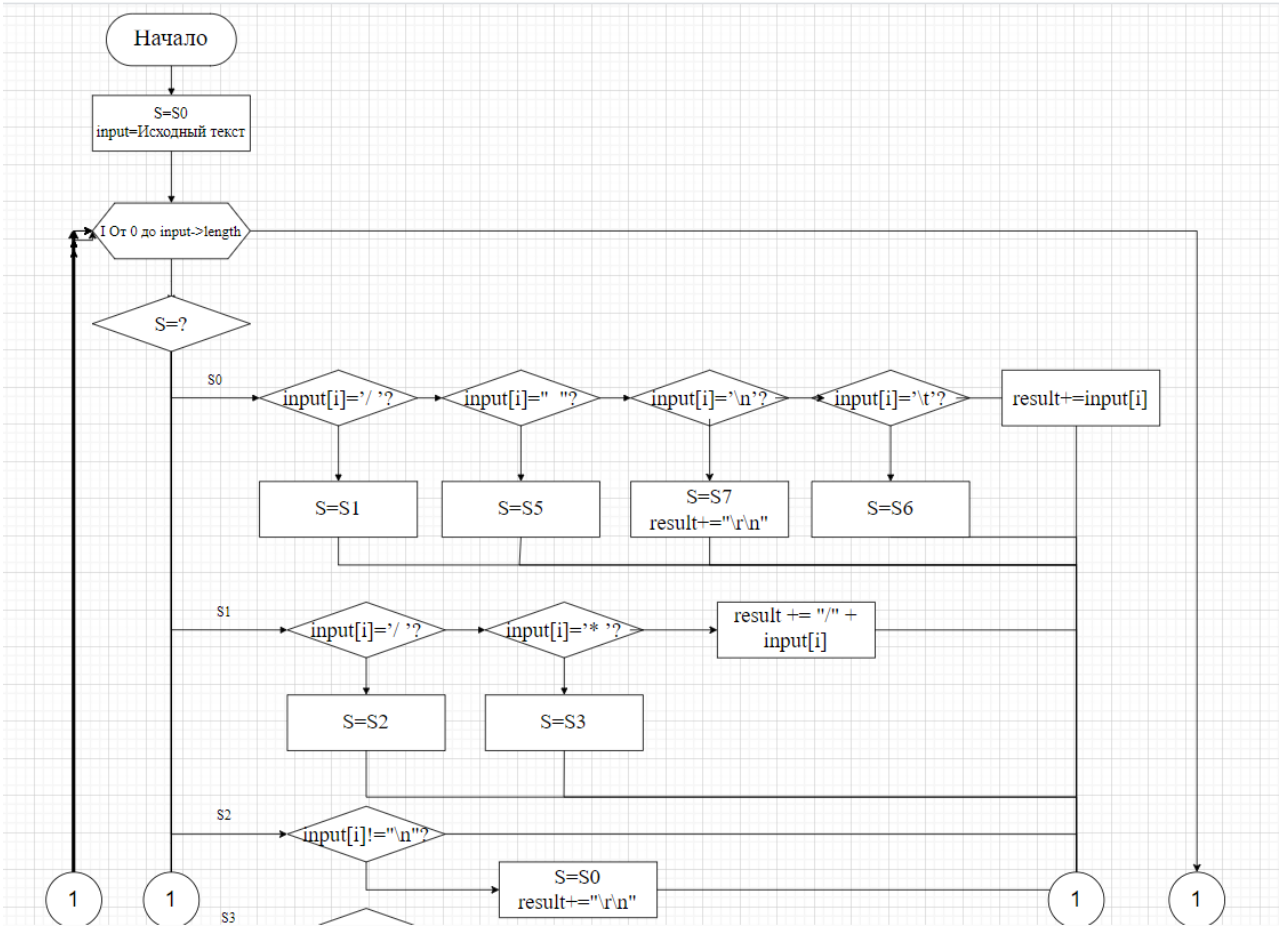
X – подходящий символ

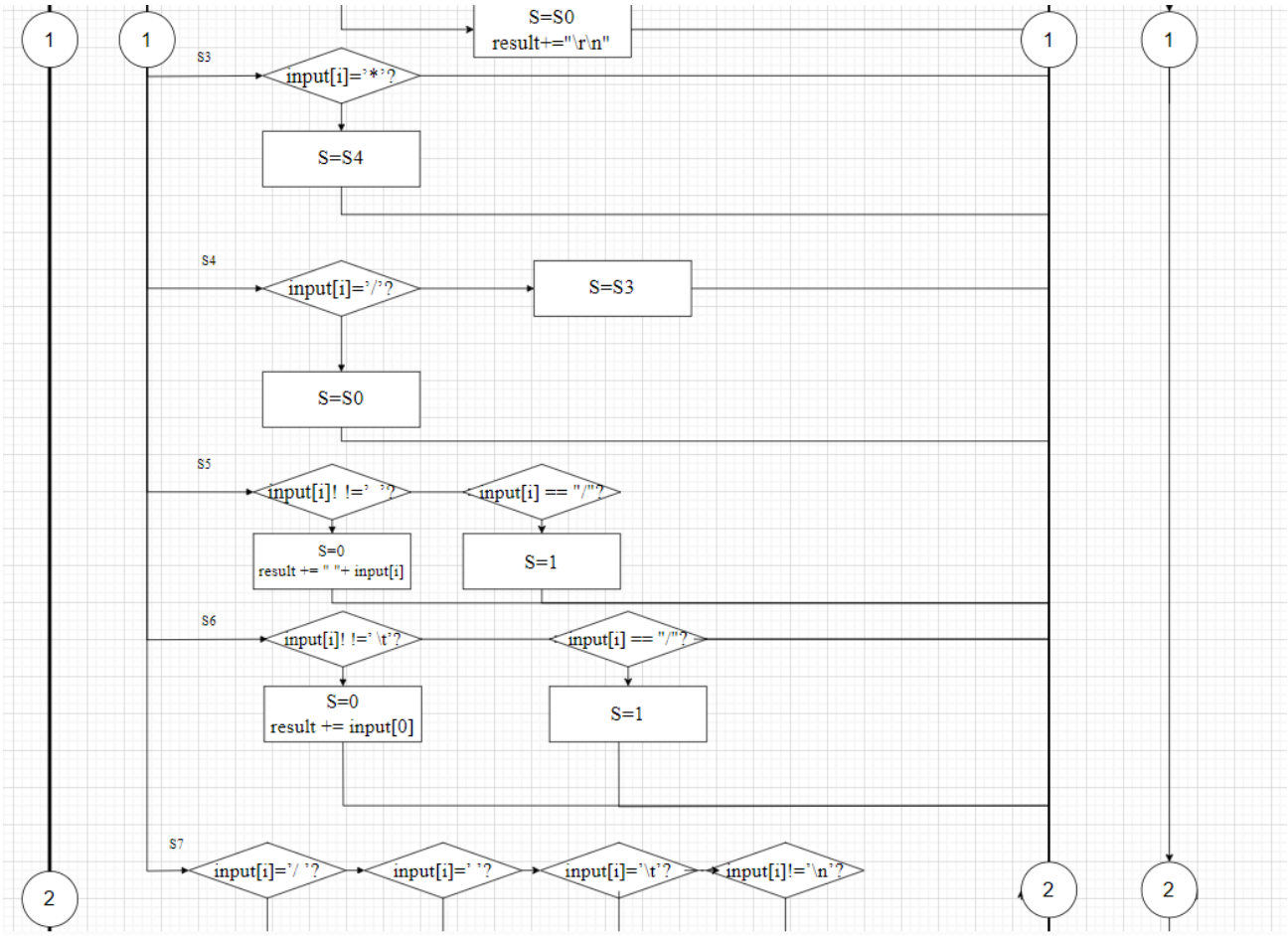
S={S0, S1, S2, S3, S4, S5, S6,S7}

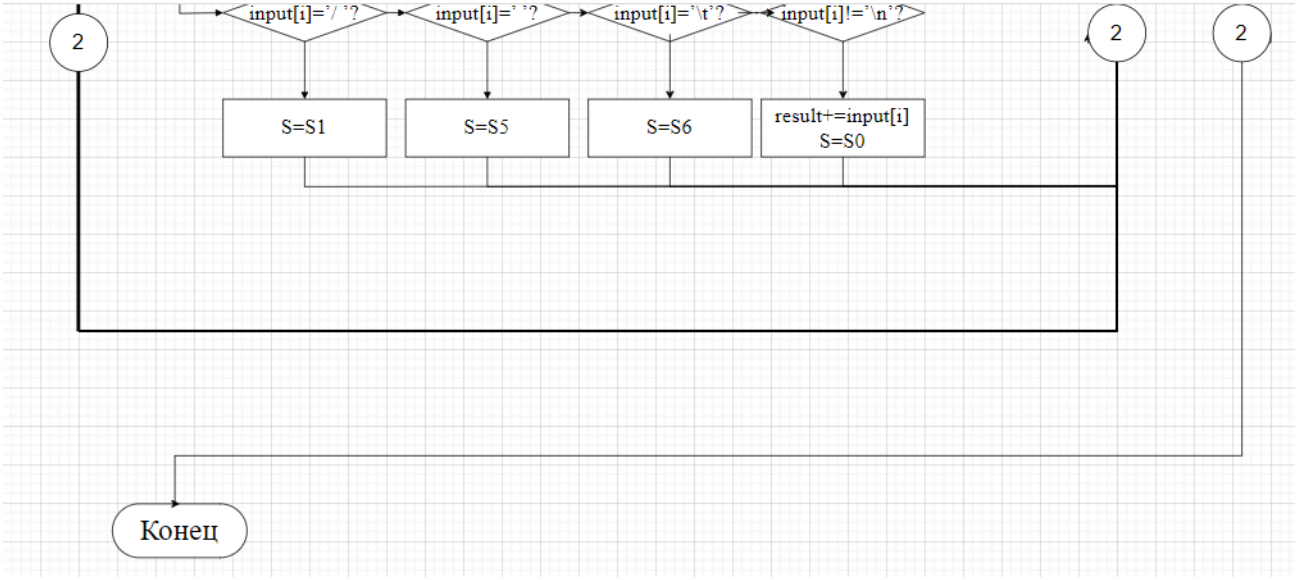
S0 = {S0}

F = {S0, S2, S5, S6};

Блок-схема


Начало
S=S0
input=Исходный текст
I От 0 до input->length
S=?
S0
input[i]='/ '?
input[i]=" "?
input[i]='\n'?
input[i]='\t'?
result+=input[i]
S=S1
S=S5
S=S7
result+="\r\n"
S=S6
S1
input[i]='/ '?
input[i]='* '?
result += "/" +
input[i]
S=S2
S=S3
S2
input[i]!="\n"?
S=S0
result+="\r\n"
1
1
1
1
S3

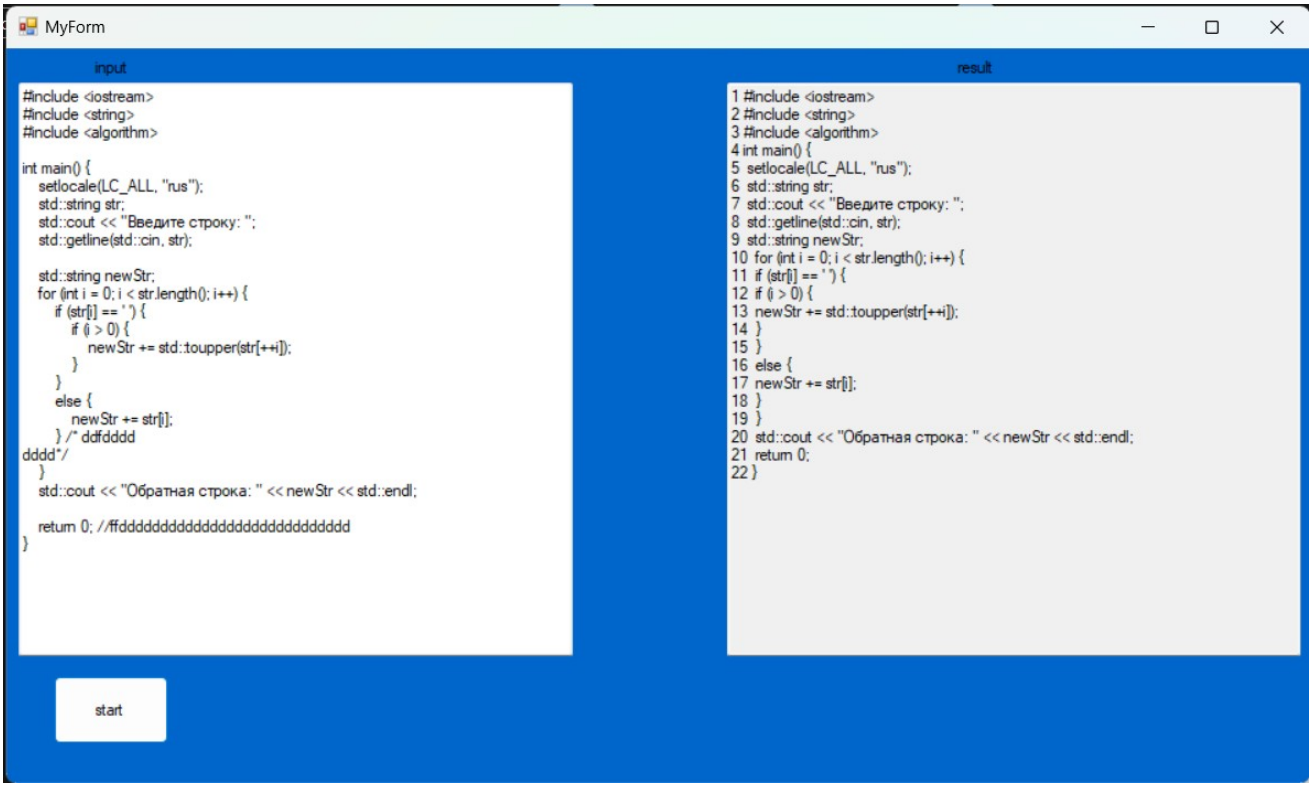

1
1
S=S0
result+="\r\n"
1
1
S3
input[i]='*'?
S=S4
S4
input[i]='/'?
S=S3
S=S0
S5
input[i]! !=' '?
input[i] == "/"?
S=0
result += " "+ input[i]
S=1
S6
input[i]! !='\t'?
input[i] == "/"?
S=0
result += input[0]
S=1
S7
input[i]='/ '?
input[i]=' '?
input[i]='\t'?
input[i]!='\n'?
2
2
2

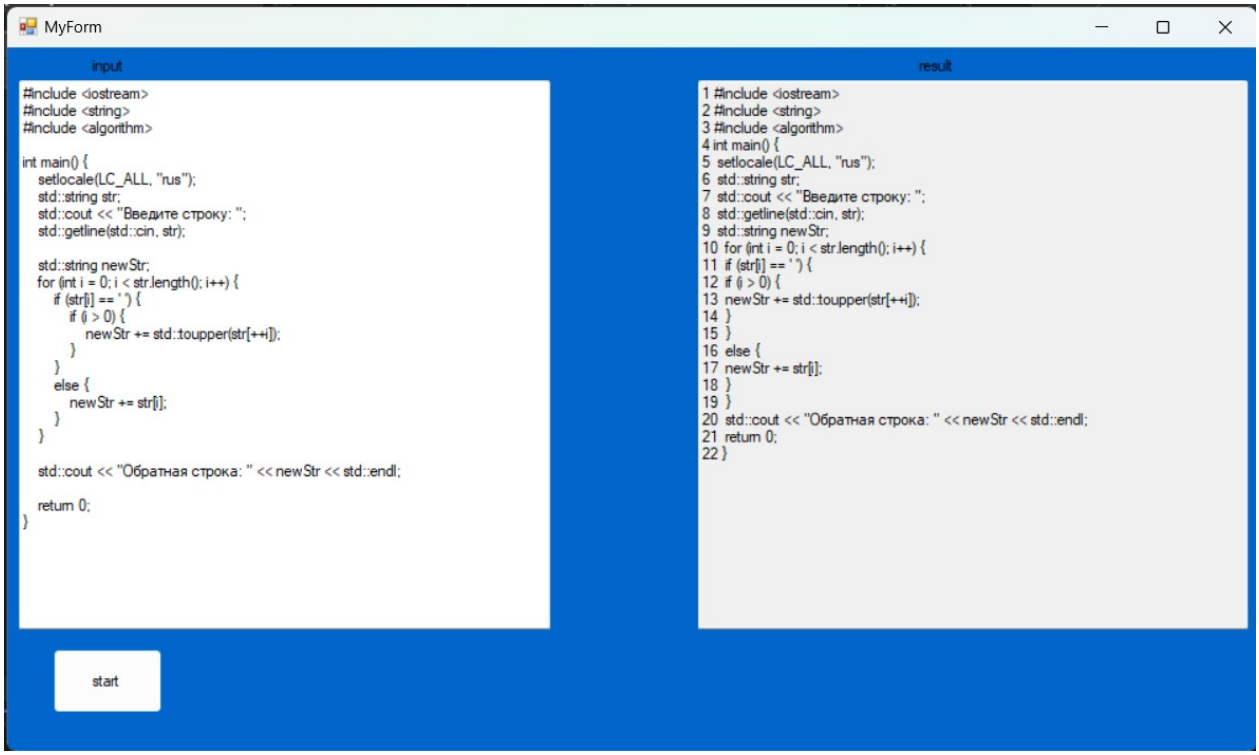

2
input[i]=' '?
input[i]=' '?
input[i]='\t'?
input[i]!='\n'?
2
2
S=S1
S=S5
S=S6
result+=input[i]
S=S0
Конец

Входной код

```cpp
#include <iostream>
#include <string>
#include <algorithm>

int main() {
    setlocale(LC_ALL, "rus");
    std::string str;
    std::cout << "Введите строку: ";
    std::getline(std::cin, str);

    std::string newStr;
    for (int i = 0; i < str.length(); i++) {
        if (str[i] == ' ') {
            if (i > 0) {
                newStr += std::toupper(str[++i]);
            }
        }
        else {
            newStr += str[i];
        }
    }

    std::cout << "Обратная строка: " << newStr << std::endl;

    return 0;

 }
```

Тестирование





Вывод

В ходе выполнения лабораторной работы был разработан конечный автомат, включая его программную реализацию. Были описаны его компоненты, построена блок-схема и граф, который наглядно отражает функционирование автомата в соответствии с предоставленным описанием. Затем было создано визуальное приложение, в котором автомат представлен в виде программы. Это приложение позволяет редактировать исходный текст, убирая лишние пробелы, комментарии, а также подсчитывает количество строк в исходном тексте.

Текст программы

```
#pragma once
#include "States.cpp"

namespace Lexer {

	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>
	/// ******* *** MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: ********* *** ***************
			//
		}

	protected:
		/// <summary>
		/// ************ *** *************** ********.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::TextBox^ input;
	protected:
	private: System::Windows::Forms::TextBox^ result;
	private: System::Windows::Forms::Button^ startProcessing;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Label^ label2;

	private:
		/// <summary>
		/// **************** ************ ****************.
		/// </summary>
		System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// ************ ****** *** ********** *************** * ** ********** 
		/// ************ ****** ******* * ******** ********** ****.
		/// </summary>
		void InitializeComponent(void)
		{
			this->input = (gcnew System::Windows::Forms::TextBox());
			this->result = (gcnew System::Windows::Forms::TextBox());
			this->startProcessing = (gcnew System::Windows::Forms::Button());
			this->label1 = (gcnew System::Windows::Forms::Label());
			this->label2 = (gcnew System::Windows::Forms::Label());
			this->SuspendLayout();
			//
			// input
			//
			this->input->Location = System::Drawing::Point(9, 25);
			this->input->Margin = System::Windows::Forms::Padding(2, 2, 2, 2);
			this->input->Multiline = true;
			this->input->Name = L"input";
			this->input->Size = System::Drawing::Size(402, 416);
			this->input->TabIndex = 0;
			//
			// result
			//
			this->result->Location = System::Drawing::Point(523, 25);
			this->result->Margin = System::Windows::Forms::Padding(2, 2, 2, 2);
			this->result->Multiline = true;
			this->result->Name = L"result";
			this->result->ReadOnly = true;
			this->result->Size = System::Drawing::Size(416, 416);
			this->result->TabIndex = 1;
			//
			// startProcessing
			//
			this->startProcessing->Location = System::Drawing::Point(34, 455);
			this->startProcessing->Margin = System::Windows::Forms::Padding(2, 2, 2, 2);
```

```
                        this->startProcessing->Name = L"startProcessing";
                        this->startProcessing->Size = System::Drawing::Size(84, 50);
                        this->startProcessing->TabIndex = 2;
                        this->startProcessing->Text = L"start";
                        this->startProcessing->UseVisualStyleBackColor = true;
                        this->startProcessing->Click += gcnew System::EventHandler(this,
&MyForm::startProcessing_Click);
                        //
                        // label1
                        //
                        this->label1->AutoSize = true;
                        this->label1->Location = System::Drawing::Point(62, 7);
                        this->label1->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);
                        this->label1->Name = L"label1";
                        this->label1->Size = System::Drawing::Size(30, 13);
                        this->label1->TabIndex = 3;
                        this->label1->Text = L"input";
                        //
                        // label2
                        //
                        this->label2->AutoSize = true;
                        this->label2->Location = System::Drawing::Point(688, 7);
                        this->label2->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);
                        this->label2->Name = L"label2";
                        this->label2->Size = System::Drawing::Size(32, 13);
                        this->label2->TabIndex = 4;
                        this->label2->Text = L"result";
                        //
                        // MyForm
                        //
                        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                        this->BackColor = System::Drawing::SystemColors::HotTrack;
                        this->ClientSize = System::Drawing::Size(946, 533);
                        this->Controls->Add(this->label2);
                        this->Controls->Add(this->label1);
                        this->Controls->Add(this->startProcessing);
                        this->Controls->Add(this->result);
                        this->Controls->Add(this->input);
                        this->Margin = System::Windows::Forms::Padding(2, 2, 2, 2);
                        this->Name = L"MyForm";
                        this->Text = L"MyForm";
                        this->ResumeLayout(false);
                        this->PerformLayout();

                }
#pragma endregion

        private: States currentState = States::S0;
        private: System::Void startProcessing_Click(System::Object^ sender, System::EventArgs^ e) {
                this->result->Text = "";
                String^ str = "";
                currentState = States::S0;
                for (int i = 0; i < this->input->Text->Length; i++) {
                        switch (currentState) {
                        case States::S0:
                                if (this->input->Text[i] == '/') {
                                        currentState = States::S1;
                                }
                                else if (this->input->Text[i] == ' ') {
                                        currentState = States::S5;
                                }
                                else if (this->input->Text[i] ==  '\t') {
                                        currentState = States::S6;
                                }
                                else if (this->input->Text[i] == '\n') {
                                        str += this->input->Text[i];
                                        currentState = States::S7;
                                }
                                else {
                                        str += this->input->Text[i];
                                }
                                break;
                        case States::S1:
                                if (this->input->Text[i] == '/') {
                                        currentState = States::S2;
                                }
                                else if (this->input->Text[i] == '*') {
                                        currentState = States::S3;
                                }
                                else {
                                        str +=  "/" + this->input->Text[i];
                                        currentState = States::S0;
                                }
                                break;
                        case States::S2:
                                if (this->input->Text[i] == '\n') {
                                        str += "\r\n";
                                        currentState = States::S0;
                                }
                                break;
```

```
                case States::S3:
                        if (this->input->Text[i] == '*') {
                                currentState = States::S4;
                        }
                        break;
                case States::S4:
                        if (this->input->Text[i] == '/') {
                                currentState = States::S0;
                        }
                        else {
                                currentState = States::S3;
                        }
                        break;
                case States::S5:
                        if (this->input->Text[i] == '/') {
                                currentState = States::S1;
                        }
                        else if (this->input->Text[i] != ' ') {
                                currentState = States::S0;
                                str += " ";
                                str += this->input->Text[i];
                        }
                        break;
                case States::S6:
                        if (this->input->Text[i] == '/') {
                                currentState = States::S1;
                        }
                        else if (this->input->Text[i] != '\t') {
                                currentState = States::S0;
                                str += this->input->Text[i];
                        }
                        break;
                case States::S7:
                        if (this->input->Text[i] == '\n' || this->input->Text[i] == '\r') {
                                currentState = States::S7;
                        }
                        else if (this->input->Text[i] == '/')
                                currentState = States::S1;
                        else if (this->input->Text[i] == '\t') {
                                currentState = States::S6;
                        }
                        else if (this->input->Text[i] == ' ') {
                                currentState = States::S5;
                        }
                        else {
                                str += this->input->Text[i];
                                currentState = States::S0;
                        }
                        break;
                }
        }
        int numberString = 1;
        for (int i = 0; i < str->Length; i++) {
                if (i == 0) {
                        this->result->Text += numberString.ToString() + " " + str[i];
                        numberString++;
                }
                else if (str[i] == '\n') {
                        this->result->Text += str[i] + numberString.ToString() + " ";
                        numberString++;
                }
                else {
                        this->result->Text += str[i];
                }
        }
}
};
};
```