МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Теория автоматов и формальных языков»

Лабораторная работа №1-2

«Дополнительные функции лексического анализатора»

Выполнил:

студент группы 1ПИб-02-2оп-22

Тихомиров Владислав Валерьевич

Проверил:

доцент, к.т.н.

Ганичева Оксана Георгиевна

Череповец, 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.

1. Тестовый код

#include <iostream>

using namespace std;

void main() {

// запускаем цикл for

for (int i = 0; i < 5; i++) {

// запускаем switch, который в качестве параметра берёт i

switch (i)

{

case 0:

cout << i << endl;

break;

case 1:

cout << i << endl;

break;

case 2:

cout << i << endl;

break;

case 3:

cout << i << endl;

break;

case 4:

cout << i << endl;

break;

case 5:

cout << i << endl;

break;

}

}

}

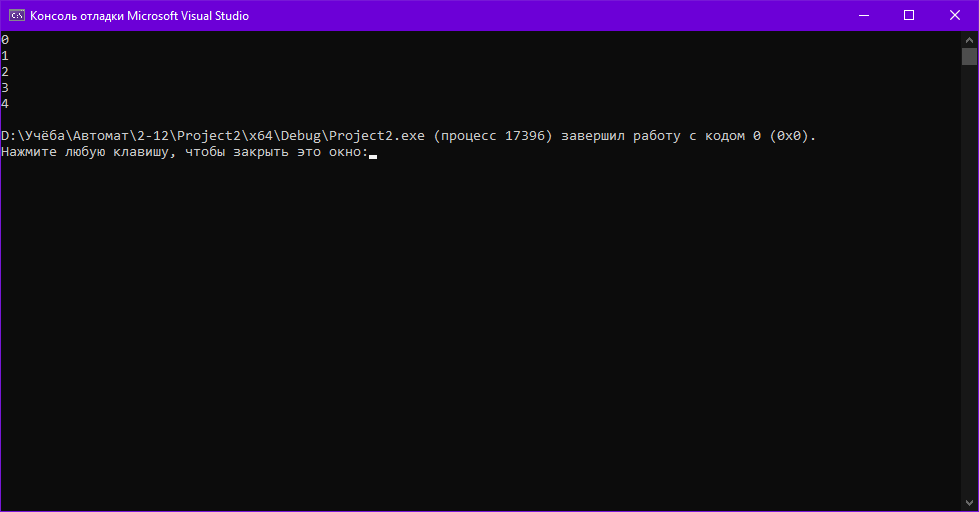


Рис. 1. Результат работы тестового кода

2. Формальное описание конечного автомата

А = {X, S, S0, F, δ}

X = {|\_|, /, v,\t,\n,\*}

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

S0 = {S0}

F = {S0, S2, S5, S6, S7}

δ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (S0, /) → S1 | (S1, /) → S2 | (S2, /) → S2 | (S3, /) → S3 | (S4, /) → S0 |
| (S0, \*) → S0 | (S1, \*) → S3 | (S2, \*) → S2 | (S3, \*) → S4 | (S4, \*) → S3 |
| (S0, v) → S0 | (S1, v) → S0 | (S2, v) → S2 | (S3, v) → S3 | (S4, v) → S3 |
| (S0, |\_|) → S5 | (S1, |\_|) → S0 | (S2, |\_|) → S2 | (S3, |\_|) → S3 | (S4, |\_|) → S3 |
| (S0, \t) → S6 | (S1, \t) → S0 | (S2, \t) → S2 | (S3, \t) → S3 | (S4, \t) → S3 |
| (S0, \n) → S7 | (S1, \n) → S0 | (S2, \n) → S0 | (S3, \n) → S3 | (S4, \n) → S3 |

|  |  |  |
| --- | --- | --- |
| (S5, /) → S1 | (S6, /) → S1 | (S7, /) → S1 |
| (S5, \*) → S5 | (S6, \*) → S6 | (S7, \*) → S7 |
| (S5, v) → S0 | (S6, v) → S0 | (S7, v) → S0 |
| (S5, |\_|) → S5 | (S6, |\_|) → S5 | (S7, |\_|) → S5 |
| (S5, \t) → S6 | (S6, \t) → S6 | (S7, \t) → S6 |
| (S5, \n) → S7 | (S6, \n) → S7 | (S7, \n) → S7 |

3. Граф конечного автомата

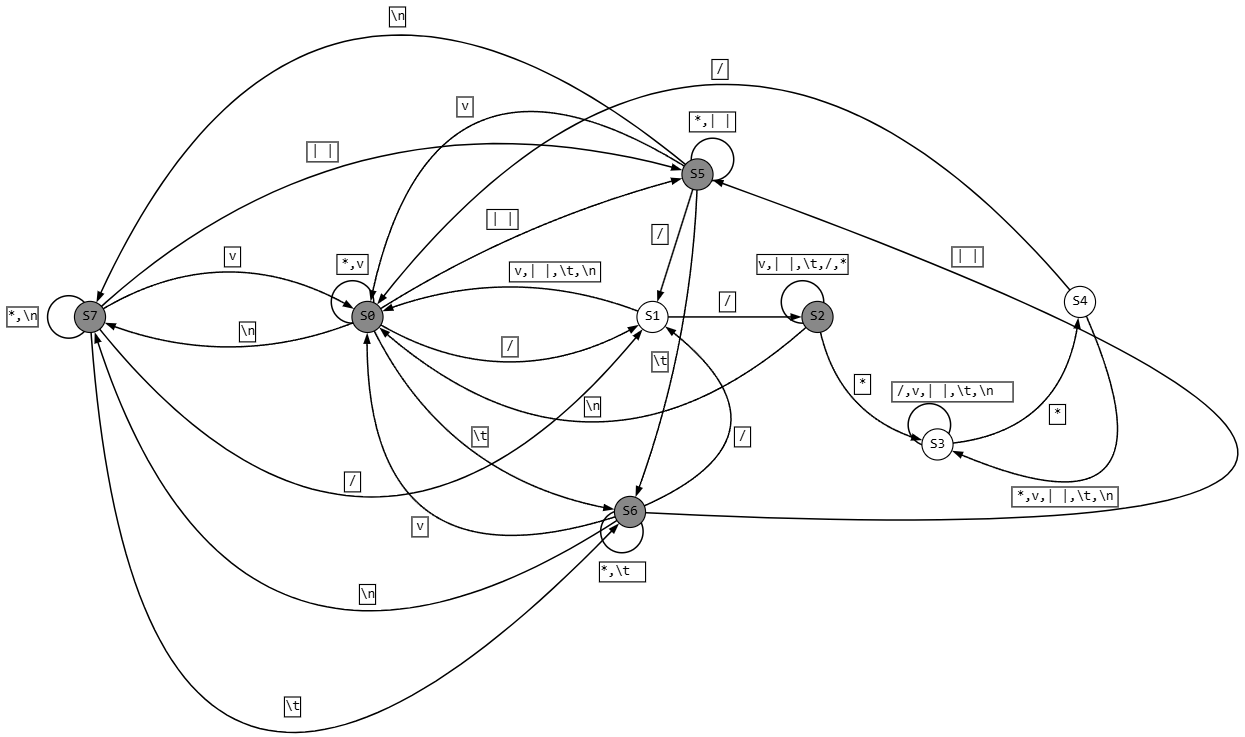
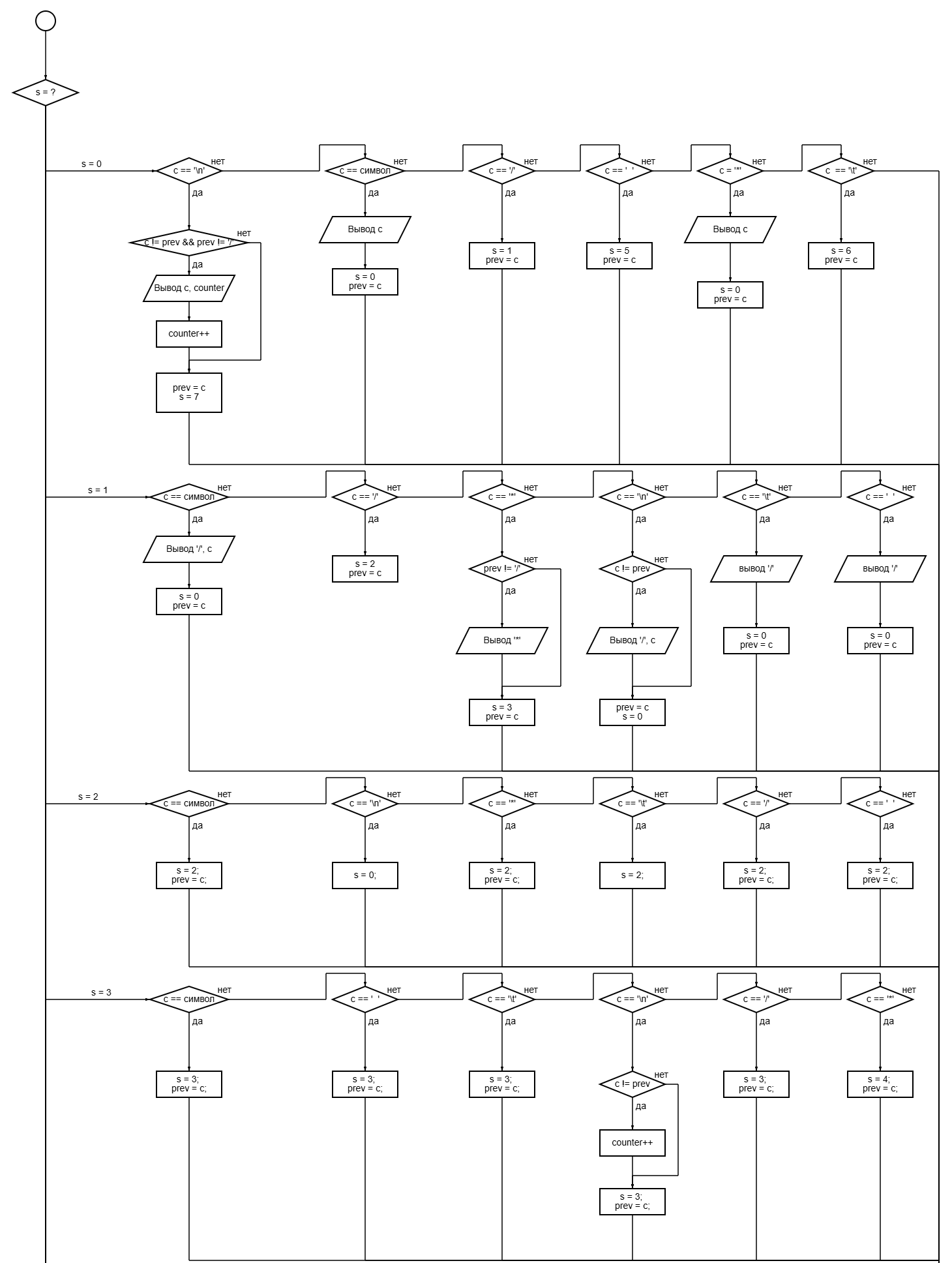
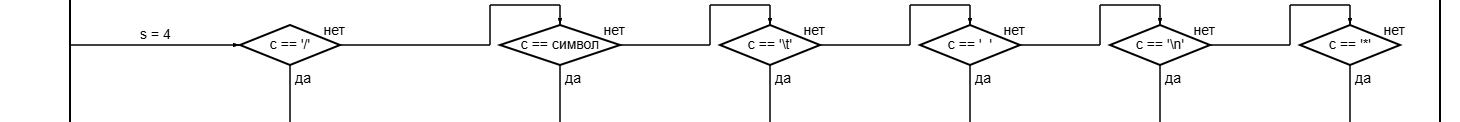


Рис. 2. Граф конечного автомата

4. Блок-схема switch





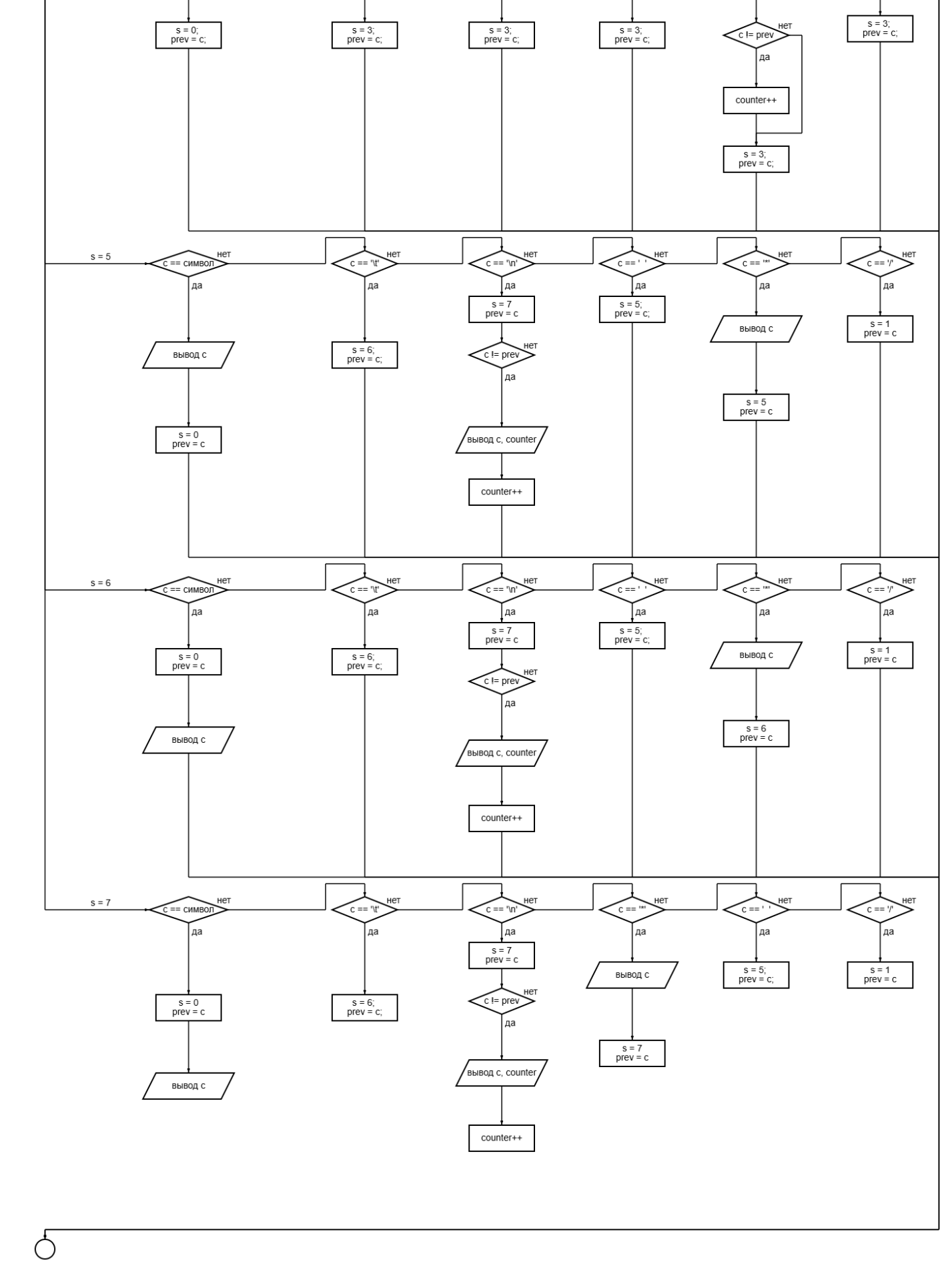


Рис. 3. Блок схема switch

5. Блок-схема программы

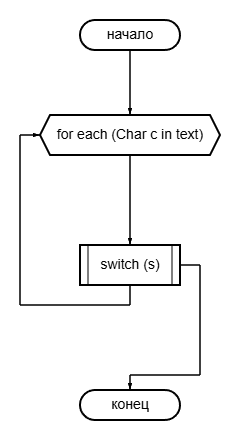


Рис. 4. Блок схема программы

6. Демонстрация работы программы

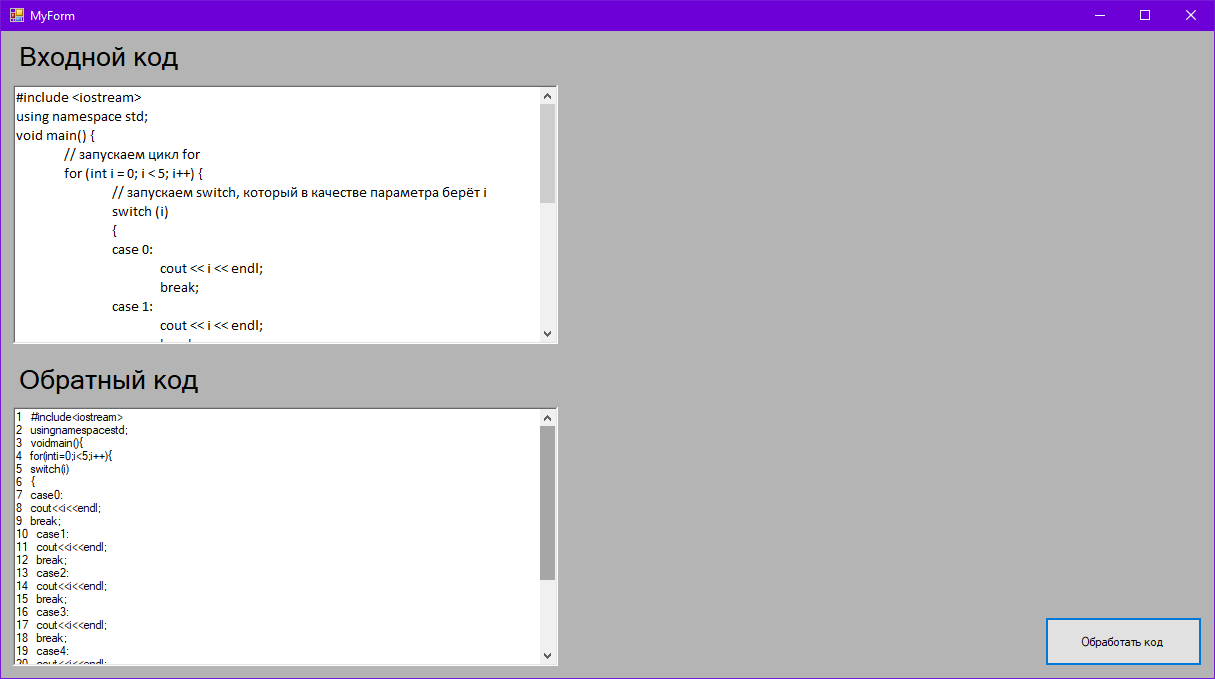


Рис. 5. Демонстрация работы программы

7. Код программы

#pragma once

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:

/// <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->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(193, 31);

this->label2->TabIndex = 3;

this->label2->Text = L"Обратный код";

//

// button1

//

this->button1->Location = System::Drawing::Point(1044, 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);

//

// 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(1213, 647);

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";

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;

// 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;

}

}

}

};

}

Вывод

В ходе лабораторной работы был построен конечный автомат для распознавания входной последовательности из входного файла и граф. Данный конечный автомат был смоделирован с помощью языка программирования C++. Его работа была продемонстрирована при помощи средств Visual Studio и пакета Microsoft Forms. Было создано визуальное приложение, которое позволило наглядно продемонстрировать работу автомата, создать пользовательский интерфейс.

Характеристики построенного автомата:

1. Детерменированный – следующее состояние определяется однозначно текущим состоянием и входным символом. Функция переходов имеет только одно результирующее состояние
2. Абстрактный – математическая модель дискретного устройства, которое в каждый момент времени находится в каком-либо одном состоянии из множества возможных
3. Конечный – число состояний конечно
4. Асинхронный – работает над последовательностью при ее поступлении на вход, что может произойти с различными интервалами времени