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 №3-4
«Регулярные множества и регулярные выражения»

Выполнил:
студент группы 1ПИб-02-2оп-22
Зернов Владислав Александрович
Проверил:
доцент, к.т.н.
Ганичева Оксана Георгиевна

Череповец, 2023 год

Задание 1.

1. Построить конечный автомат с входным алфавитом V = {a, b}, распознающий:

е) Все цепочки, заканчивающиеся кодом aabba

Регулярное выражение: (a+b)*aabba

Примеры строк: aabba, aaabba, baabba, abababababaababbaabba и т. д.

$A = \{X, S, S_0, F, \delta\}$

$X = \{a, b\}$

$S = \{S_0, S_1, S_2, S_3, S_4, S_5\}$

$S_0 = \{S_0\}$

$F = \{S_5\}$

δ:

$(S_0,$	a)	→	$(S_1,$	a)	→	$(S_2,$	a)	→	$(S_3,$	a)	→
S_1			S_2			S_2			S_1		
$(S_0,$	b)	→	$(S_1,$	b)	→	$(S_2,$	b)	→	$(S_3,$	b)	→
S_0			S_0			S_3			S_4		

$(S_4,$	a)	→	$(S_5,$	a)	→						
S_5			S_0								
$(S_4,$	b)	→	$(S_5,$	b)	→						
S_0			S_0								

Граф:

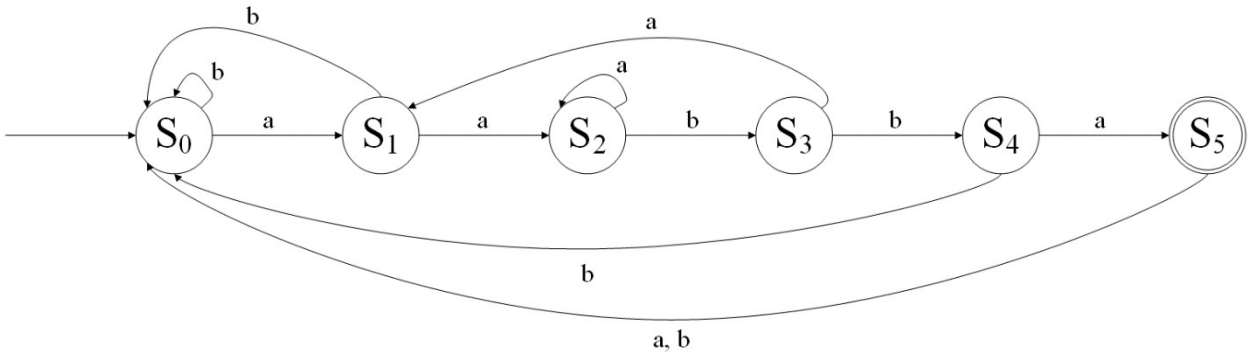

Рис. 1. Граф конечного автомата, соответвующего выражению (a+b)*aabba

ж) Все цепочки, в которых за каждым a непосредственно следует b

Регулярное выражение: (b*(ab+)*)*

Примеры строк: ε, b, bb, bbb, ab, bab, babb, babab, abab, bbbbbbabbbbbabbbbbabb и т. д.

A = {X, S, S_0, F, δ}

X = {a, b}

S = {S_0, S_1, S_2, S_3}

S_0 = {S_0}

F = {S_2}

δ:

(S_0, a) → S_1 (S_1, a) → S_3 (S_2, a) → S_1 (S_3, a) → S_3

(S_0, b) → S_0 (S_1, b) → S_2 (S_2, b) → S_2 (S_3, b) → S_3

Граф:

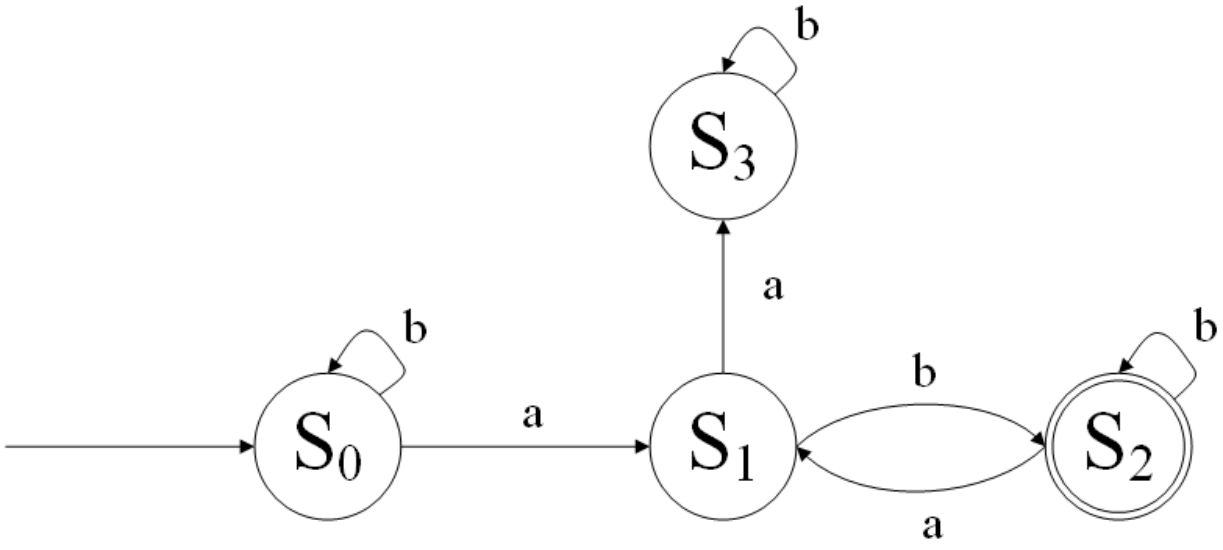

Рис. 2. Граф конечного автомата, соответвующего выражению (b*(ab+)*)*

2. Построить конечный автомат с входным алфавитом V = {a, b, c}, распознающий:

а) Все цепочки, в которых за каждым a когда-нибудь в будущем следует b

Регулярное выражение: (b+c)*(a(a+c)*b(b+c)*)*

Примеры строк: ε, b, c, bc, cb, bcbcbbcb, ab, acb, cacbc, bccbbcccbbcaccccccbacbc и т. д.

A = {X, S, S_0, F, δ}

X = {a, b, c}

S = {S_0, S_1, S_2}

S_0 = {S_0}

F = {S_2}

δ:

(S_0, a) → (S_1, a) → (S_2, a) → S_1 S_1 S_1

(S_0, b) → (S_1, b) → (S_2, b) → S_0 S_2 S_2

(S_0, c) → (S_1, c) → (S_2, c) → S_0 S_1 S_2

Граф:

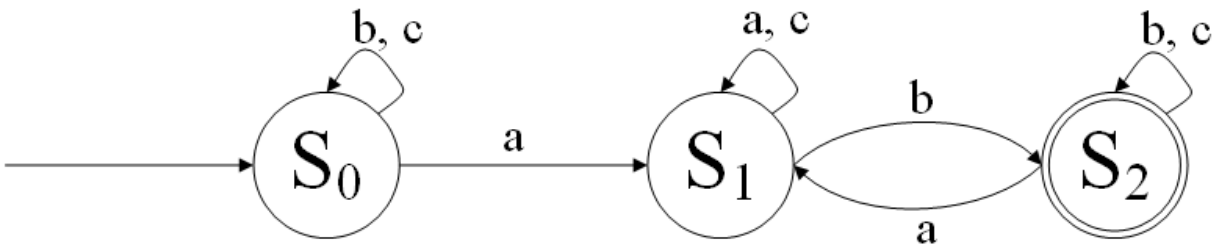

Рис. 3. Граф конечного автомата, соответвующего выражению (b+c)*(a(a+c)*b(b+c)*)*

б) Все цепочки, в которых 2 последние буквы не совпадают

Регулярное выражение: (a+b+c)*((a(b+c))+(b(a+c))+(c(a+b)))

Примеры строк: ab, ba, ac, ca, bc, cb, abc, abcbc, abbabcbccbabacbcbcb и т. д.

A = {X, S, S_0, F, δ}

X = {a, b, c}

S = {S_0, S_1, S_2, S_3, S_4, S_5, S_6}

S_0 = {S_0}

F = {S_4, S_5, S_6}

δ:

(S_0,	a)→	(S_1,	a)→	(S_2,	a)→	(S_3,	a)→	(S_4,	a)→	(S_5,	a)→	(S_6,	a)→
S_1		S_1		S_4		S_4		S_1		S_4		S_4	
(S_0,	b)→	(S_1,	b)→	(S_2,	b)→	(S_3,	b)→	(S_4,	b)→	(S_5,	b)→	(S_6,	b)→
S_2		S_5		S_2		S_5		S_5		S_2		S_5	
(S_0,	c)→	(S_1,	c)→	(S_2,	c)→	(S_3,	c)→	(S_4,	c)→	(S_5,	c)→	(S_6,	c)→
S_3		S_6		S_6		S_3		S_6		S_6		S_3	

Граф:

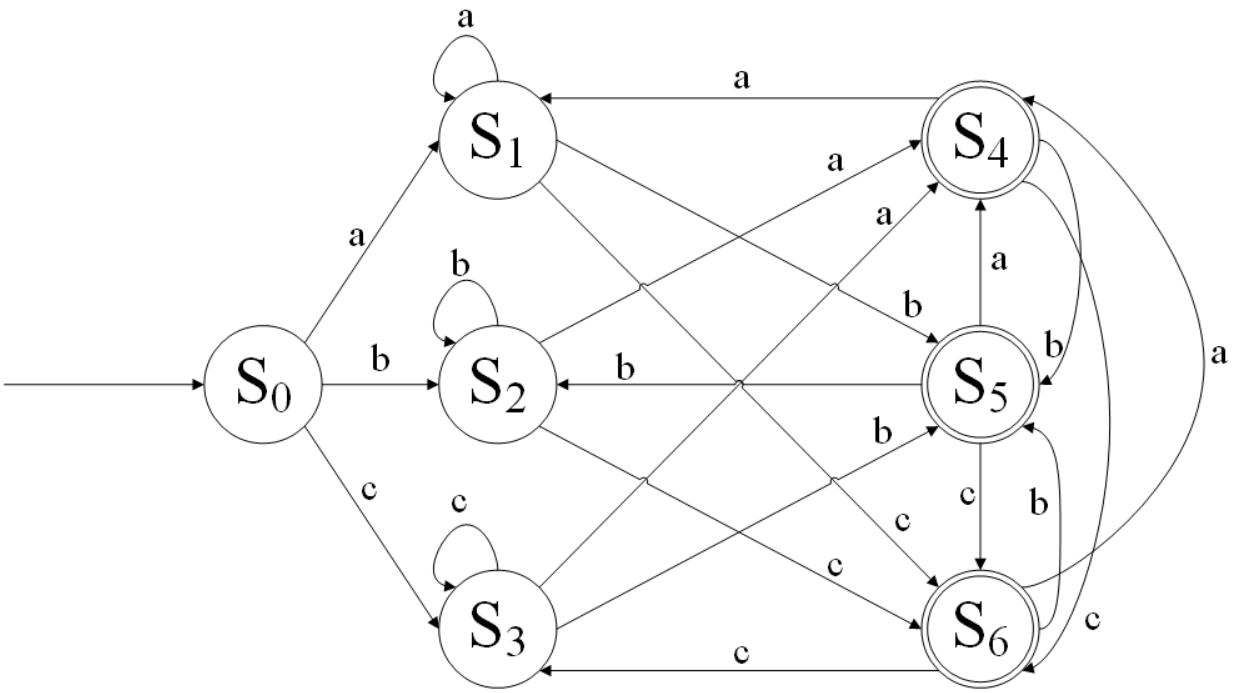

Рис. 4. Граф конечного автомата, соответвующего выражению (a+b+c)*((a(b+c))+(b(a+c))+(c(a+b)))

в) Все цепочки, начинающиеся и заканчивающиеся различными символами

Регулярное выражение: (a(a+b+c)*(b+c))+(b(a+b+c)*(a+c))+(c(a+b+c)*(a+b))

Примеры строк: ab, ba, ac, ca, bc, cb, abc, abcbc, abbabcbccbabacbcbcb и т. д.

A = {X, S, S_0, F, δ}

X = {a, b, c}

S = {S_0, S_1, S_2, S_3, S_4, S_5, S_6}

S_0 = {S_0}

F = {S_4, S_5, S_6}

δ:

(S_0, a)→	(S_1, a)→	(S_2, a)→	(S_3, a)→	(S_4, a)→	(S_5, a)→	(S_6, a)→
S_1	S_1	S_5	S_6	S_1	S_5	S_6

(S_0,	b)→	(S_1,	b)→	(S_2,	b)→	(S_3,	b)→	(S_4,	b)→	(S_5,	b)→	(S_6,	b)→
S_2		S_4		S_2		S_6		S_4		S_2		S_6	
(S_0,	c)→	(S_1,	c)→	(S_2,	c)→	(S_3,	c)→	(S_4,	c)→	(S_5,	c)→	(S_6,	c)→
S_3		S_4		S_5		S_3		S_4		S_5		S_3	

Граф:

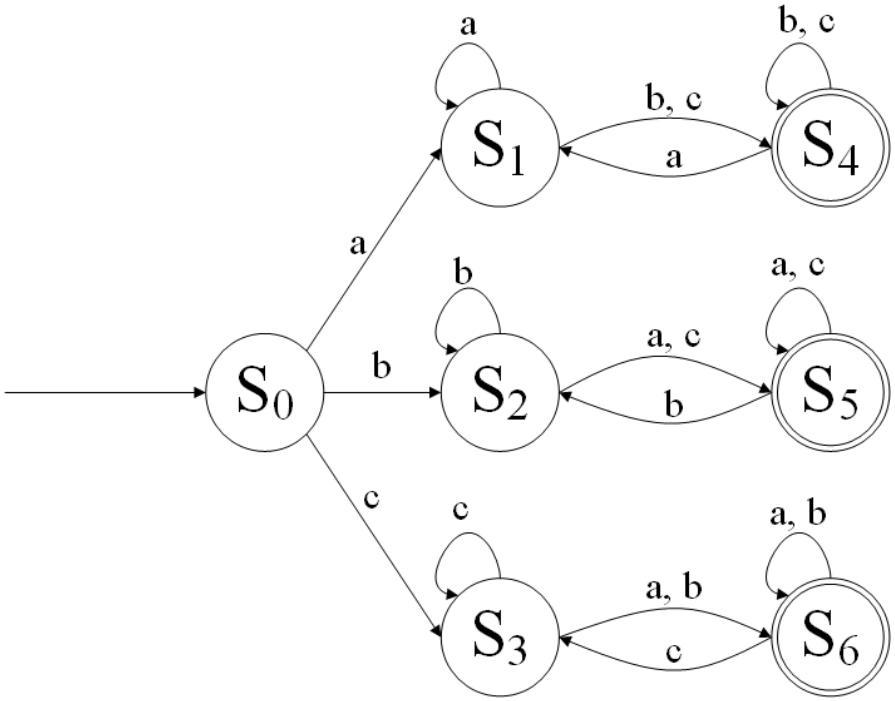

Рис. 5. Граф конечного автомата, соответвующего выражению (a(a+b+c)*(b+c))+(b(a+b+c)*(a+c))+(c(a+b+c)*(a+b))

г) Все цепочки, включающие по крайней мере 1 символ a и 1 символ b

Регулярное выражение:

(a+b+c)*((a(a+b+c)*b(a+b+c)*)+(b(a+b+c)*a(a+b+c)*))(a+b+c)*

Примеры строк: ab, ba, acb, bca, cabc, aabb, ccccccccccbccccccaccccc и т. д.

A = {X, S, S_0, F, δ}

X = {a, b, c}

$S = \{S_0, S_1, S_2, S_3\}$

$S_0 = \{S_0\}$

$F = \{S_3\}$

δ:

$(S_0, a) \rightarrow S_1$ $(S_1, a) \rightarrow S_1$ $(S_2, a) \rightarrow S_3$ $(S_3, a) \rightarrow S_3$
$(S_0, b) \rightarrow S_2$ $(S_1, b) \rightarrow S_3$ $(S_2, b) \rightarrow S_2$ $(S_3, b) \rightarrow S_3$
$(S_0, c) \rightarrow S_0$ $(S_1, c) \rightarrow S_1$ $(S_2, c) \rightarrow S_2$ $(S_3, c) \rightarrow S_3$

Граф:

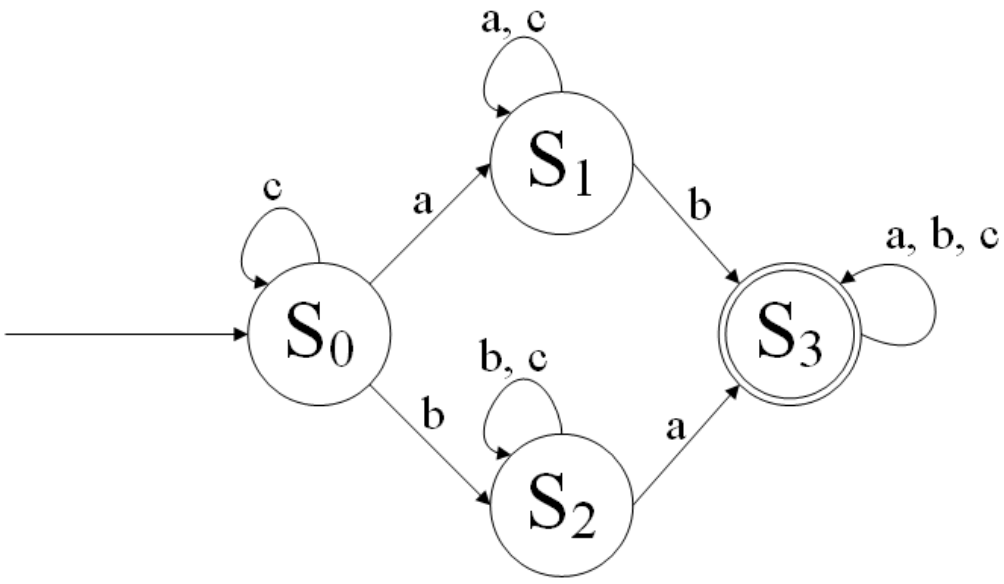

Рис. 6. Граф конечного автомата, соответвующего выражению (a+b+c)*((a(a+b+c)*b(a+b+c)*)+(b(a+b+c)*a(a+b+c)*))(a+b+c)*

Задание 2.

19. Построить автоматы, распознающие языки, задаваемые регулярными выражениями

Регулярное выражение: a*b*

Описание: все цепочки, в которых сначала идет подряд любое количество символов a, а затем подряд любое количество символов b

Примеры строк: ε, a, b, ab, aab, abb, aabb, aaaaaaaaaaaabbbbbbbbbbbbb и т. д.

A = {X, S, S_0, F, δ}

X = {a, b}

S = {S_0, S_1, S_2}

S_0 = {S_0}

F = {S_0, S_1}

δ:

(S_0, a) → S_0 (S_1, a) → S_2 (S_2, a) → S_2
(S_0, b) → S_1 (S_1, b) → S_1 (S_2, b) → S_2

Граф:

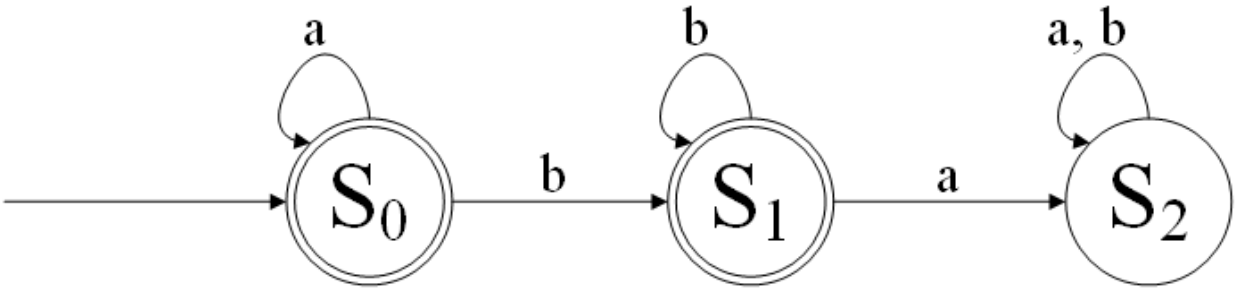

Рис. 7. Граф конечного автомата, соответвующего выражению a*b*

Регулярное выражение: a*a*

Описание: все цепочки, состоящие из любого количества символов a

Примеры строк: ε, a, aa, aaa, aaaa, aaaaa, aaaaaaaaaaaaaaaaaa и т. д.

A = {X, S, S_0, F, δ}

X = {a}

$S = \{S_0\}$

$S_0 = \{S_0\}$

$F = \{S_0\}$

$\delta: (S_0, a) \rightarrow S_0$

Граф:

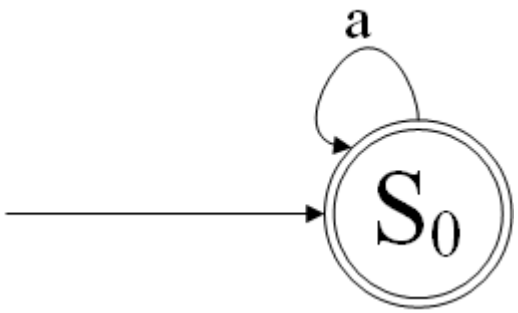

Рис. 8. Граф конечного автомата, соответвующего выражению a*a*

Регулярное выражение: a*+b*

Описание: все цепочки, состоящие из любого количества символов a или из любого количества символов b

Примеры строк: ε, a, b, aa, bb, aaa, bbb, aaaaaaaaaaaa, bbbbbbbbbb и т. д.

$A = \{X, S, S_0, F, \delta\}$

$X = \{a, b\}$

$S = \{S_0, S_1, S_2, S_3\}$

$S_0 = \{S_0\}$

$F = \{S_1, S_2\}$

δ:

$(S_0, a) \rightarrow S_1 \quad (S_1, a) \rightarrow S_1 \quad (S_2, a) \rightarrow S_3 \quad (S_3, \ a) \ \rightarrow$

$(S_0, b) \rightarrow S_2 \quad (S_1, b) \rightarrow S_3 \quad (S_2, b) \rightarrow S_2 \quad (S_3, b) \rightarrow S_3$

Граф:

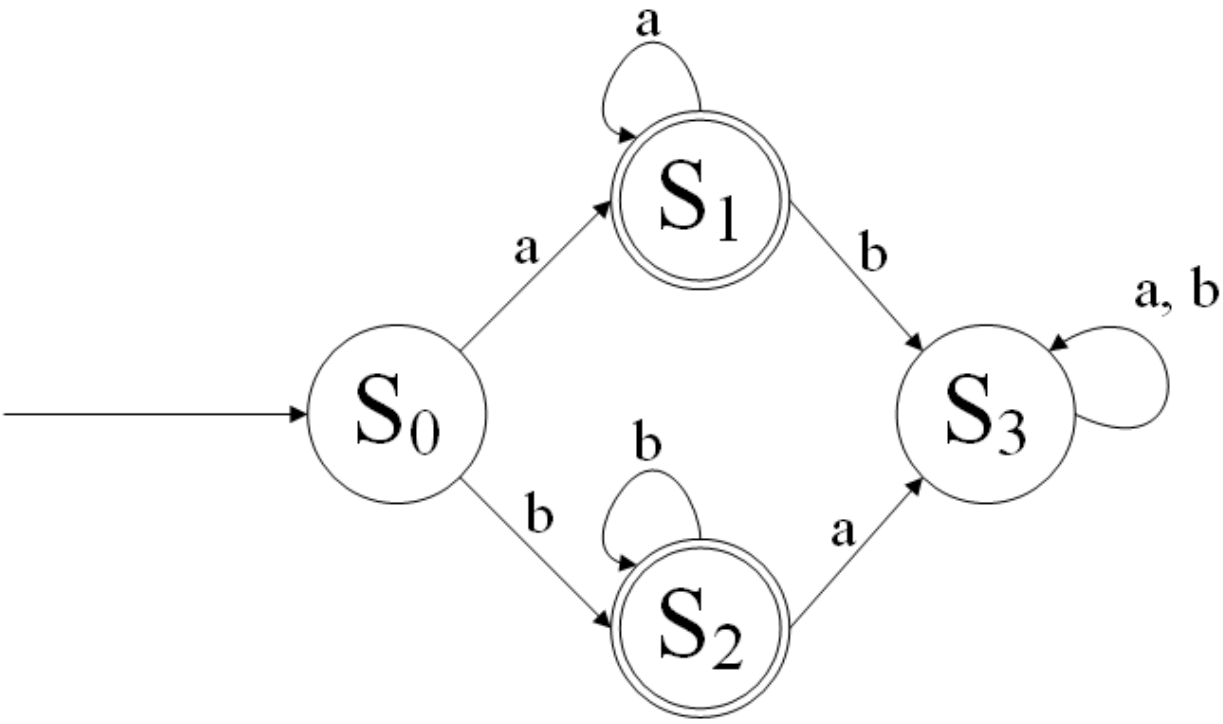

Рис. 9. Граф конечного автомата, соответвующего выражению a*+b*

Регулярное выражение: (a+b)*

Описание: все цепочки, состоящие из любого количества символов a и/или b

Примеры строк: ε, a, b, aa, bb, ab, ba, aab, bab, aabb, bbaa, abab, baba, abaabbababababaabbaba и т. д.

A = {X, S, S_0, F, δ}

X = {a, b}

S = {S_0}

S_0 = {S_0}

F = {S_0}

δ: (S_0, a) → S_0 (S_0, b) → S_0

Граф:

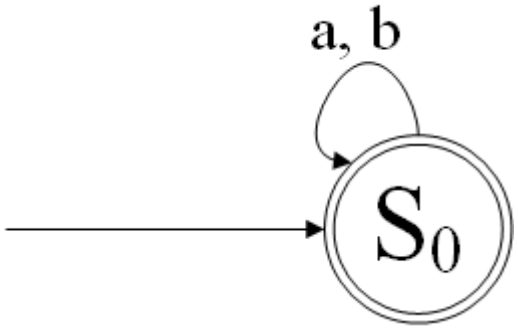

Рис. 10. Граф конечного автомата, соответвующего выражению (a+b)*

Регулярное выражение: (a*b*)*

Описание: все цепочки, состоящие из любого количества символов a и/или b

Примеры строк: ε, a, b, aa, bb, aaa, bbb, aaaaaaaaaaaa, bbbbbbbbbb и т. д.

A = {X, S, S_0, F, δ}

X = {a, b}

S = {S_0}

S_0 = {S_0}

F = {S_0}

δ: (S_0, a) → S_0 (S_0, b) → S_0

Граф:

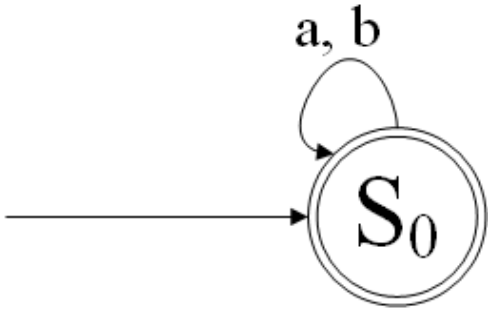

Рис. 11. Граф конечного автомата, соответвующего выражению (a*b*)*

Регулярное выражение: (a*+b*)*

Описание: все цепочки, состоящие из любого количества символов a и/или b

Примеры строк: ε, a, b, aa, bb, aaa, bbb, aaaaaaaaaaaa, bbbbbbbbbb и т. д.

A = {X, S, S_0, F, δ}

X = {a, b}

S = {S_0}

S_0 = {S_0}

F = {S_0}

δ: (S_0, a) → S_0 (S_0, b) → S_0

Граф:

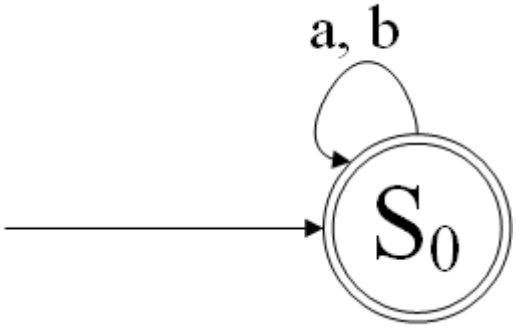

Рис. 12. Граф конечного автомата, соответвующего выражению (a*+b*)*

Регулярное выражение: a*b+b*a

Описание: все цепочки, состоящие из любого количества символов a и затем символа b или состоящие из любого количества символов b и затем символа a

Примеры строк: ε, a, b, ab, ba, aab, bba, aaaaaaaaaaab, bbbbbbbbbbbba и т. д.

$A = \{X, S, S_0, F, \delta\}$

$X = \{a, b\}$

$S = \{S_0, S_1, S_2, S_3\}$

$S_0 = \{S_0\}$

$F = \{S_3\}$

δ:

$(S_0, a) \rightarrow S_1$ $(S_1, a) \rightarrow S_1$ $(S_2, a) \rightarrow S_3$ $(S_3, a) \rightarrow S_3$
$(S_0, b) \rightarrow S_2$ $(S_1, b) \rightarrow S_3$ $(S_2, b) \rightarrow S_2$ $(S_3, b) \rightarrow S_3$

Граф:

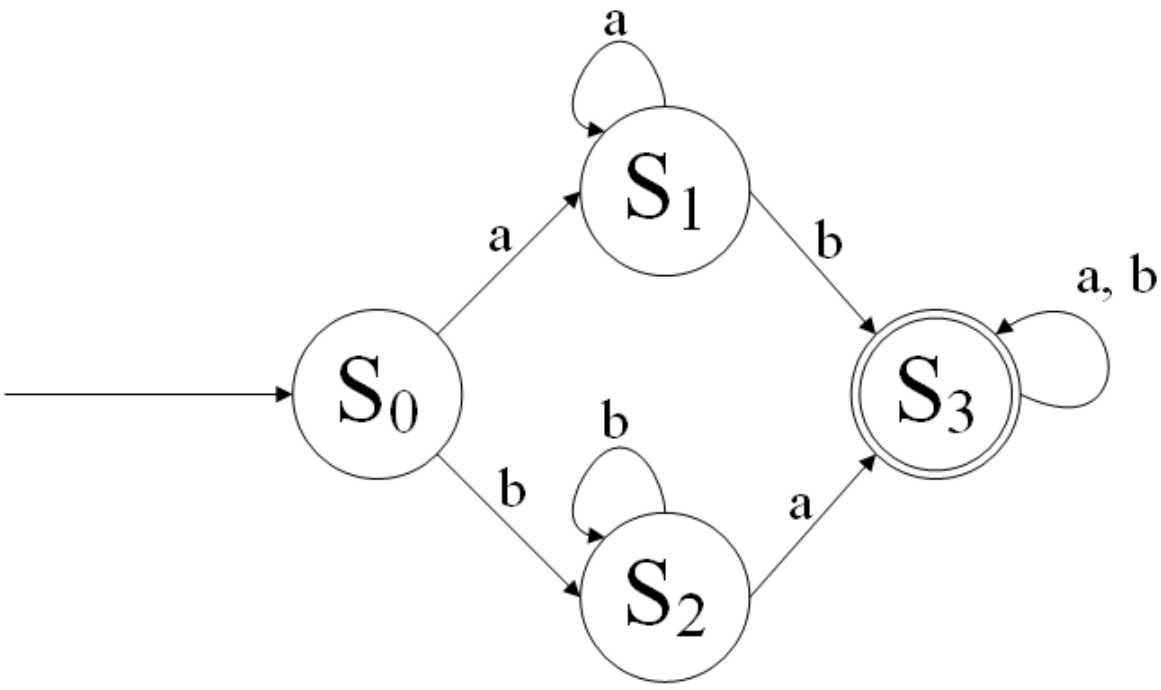

Рис. 13. Граф конечного автомата, соответвующего выражению a*b+b*a

21. Построить регулярное выражение, задающее множество всех таких слов над словарем {a, b, c}, и конечные автоматы, распознающие соответствующие языки, в которых за символом b:

а) обязательно стоит символ c

Регулярное выражение: (a+c)*(bc(a+c)*)*

Примеры строк: ε, a, c, ac, acca, abc, bcbc, cbcbc, acbcbcabccacaaccabca и т. д.

A = {X, S, S_0, F, δ}

X = {a, b, c}

S = {S_0, S_1, S_2, S_3}

S_0 = {S_0}

F = {S_2}

δ:

(S_0, a) → S_0 (S_1, a) → S_3 (S_2, a) → S_2 (S_3, a) → S_3
(S_0, b) → S_1 (S_1, b) → S_3 (S_2, b) → S_1 (S_3, b) → S_3
(S_0, c) → S_0 (S_1, c) → S_2 (S_2, c) → S_2 (S_3, c) → S_3

Граф:

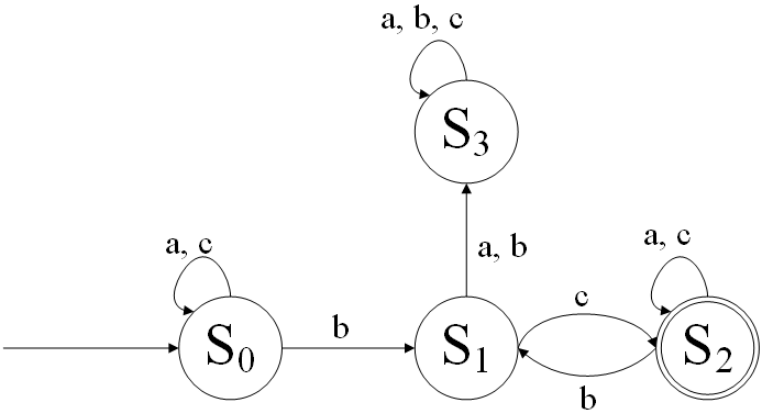

Рис. 14. Граф конечного автомата, соответвующего выражению (a+c)*(bc(a+c)*)*

б) не может стоять символ c

Регулярное выражение: (a+c)*((b*a*)+(b*a(a+c)*))*

Примеры строк: ε, a, c, ac, acca, b, ba, bb, bbbbbb, bababbabbaba, ccccccccbbaccccccbaccccc и т. д.

A = {X, S, S_0, F, δ}

X = {a, b, c}

S = {S_0, S_1, S_2}

S_0 = {S_0}

F = {S_1}

δ:

(S_0, a) → S_0 (S_1, a) → S_1 (S_2, a) → S_2
(S_0, b) → S_1 (S_1, b) → S_1 (S_2, b) → S_2
(S_0, c) → S_0 (S_1, c) → S_2 (S_2, c) → S_2

Граф:

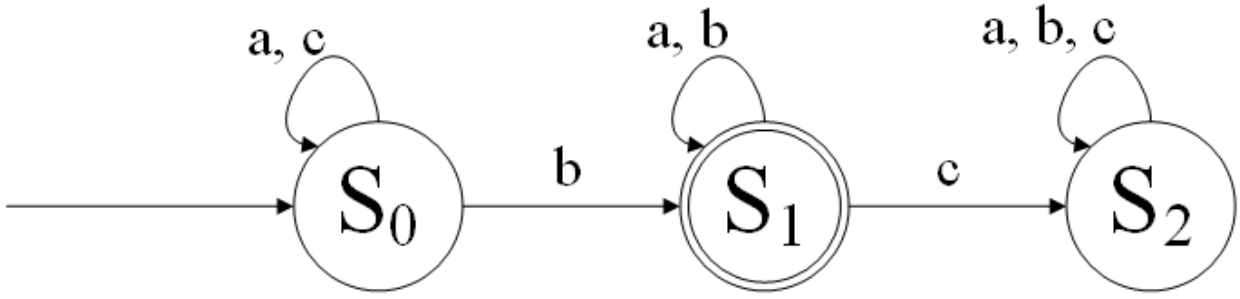

Рис. 15. Граф конечного автомата, соответвующего выражению (a+c)*((b*a*)+(b*a(a+c)*))*

Задание 3.

1. Регулярное выражение

Построить регулярное выражение, которое определяет язык, соответствующий варианту:

Вариант	Алфавит языка	Описание языка
14	1, 0	Множество слов, в которых четное число символов 0

Примечание: в данном задании решение для общего случая с использованием регулярного выражения затруднительно, поэтому решение должно быть частичным, т. е. предусматривать только некоторые случаи. Решение должно охватывать не менее 4-5 вариантов таких слов.

Регулярное выражение: (1*0·1*·0)*

Примеры строк: 100, 10101, 1010, 1001, 11011011, 101101, 101010101 и т. д.

2. Формальное описание конечного автомата

Для заданного регулярного выражения построить детерминированный конечный автомат.

$A = \{X, S, S_0, F, \delta\}$

$X = \{0, 1\}$

$S = \{S_0, S_1\}$

$S_0 = \{S_0\}$

$F = \{S_0\}$

δ:

$(S_0, 0) \rightarrow S_1$ $\quad$ $(S_1, 0) \rightarrow S_0$
$(S_0, 1) \rightarrow S_0$ $\quad$ $(S_1, 1) \rightarrow S_1$

3. Граф конечного автомата

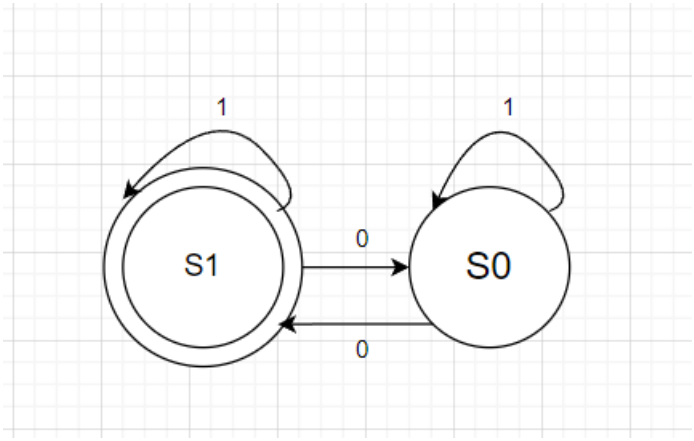

Рис. 16. Граф конечного автомата, соответствующего выражению (1*0·1*·0)*

4. Описание и блок-схема программы

На основе конечного автомата написать программу для распознавания строк, принадлежащих языку, определяемому регулярным выражением.

Представление результатов работы программы

1. В автоматическом режиме:
   1.1. Входная строка – содержит произвольное количество строк в заданном алфавите
   1.2. Две таблицы – в которые записываются правильные и неправильные строки из входного множества
2. В пошаговом режиме для заданного слова осуществляется разбор по шагам:
   2.1. Во входной строке – одно слово
   2.2. В результате – отображается последовательность команд конечного автомата

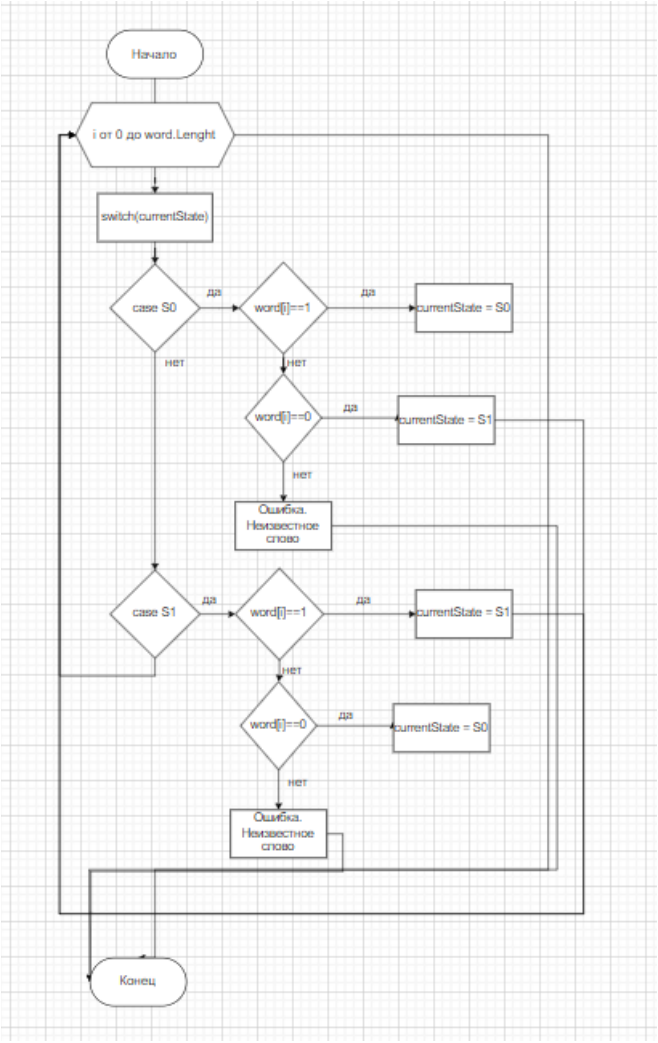

Рис. 18. Блок-схема автомата

5. Демонстрация работы программы

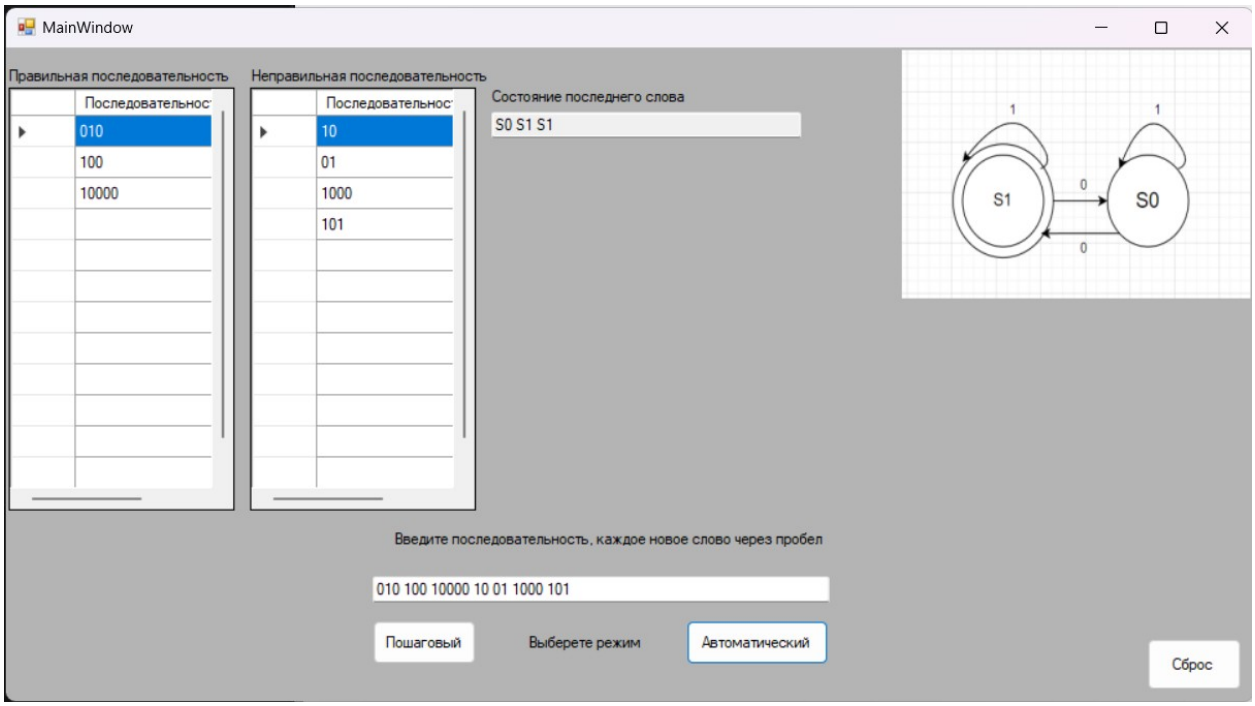

Рис. 19. Работа программы в автоматическом режиме

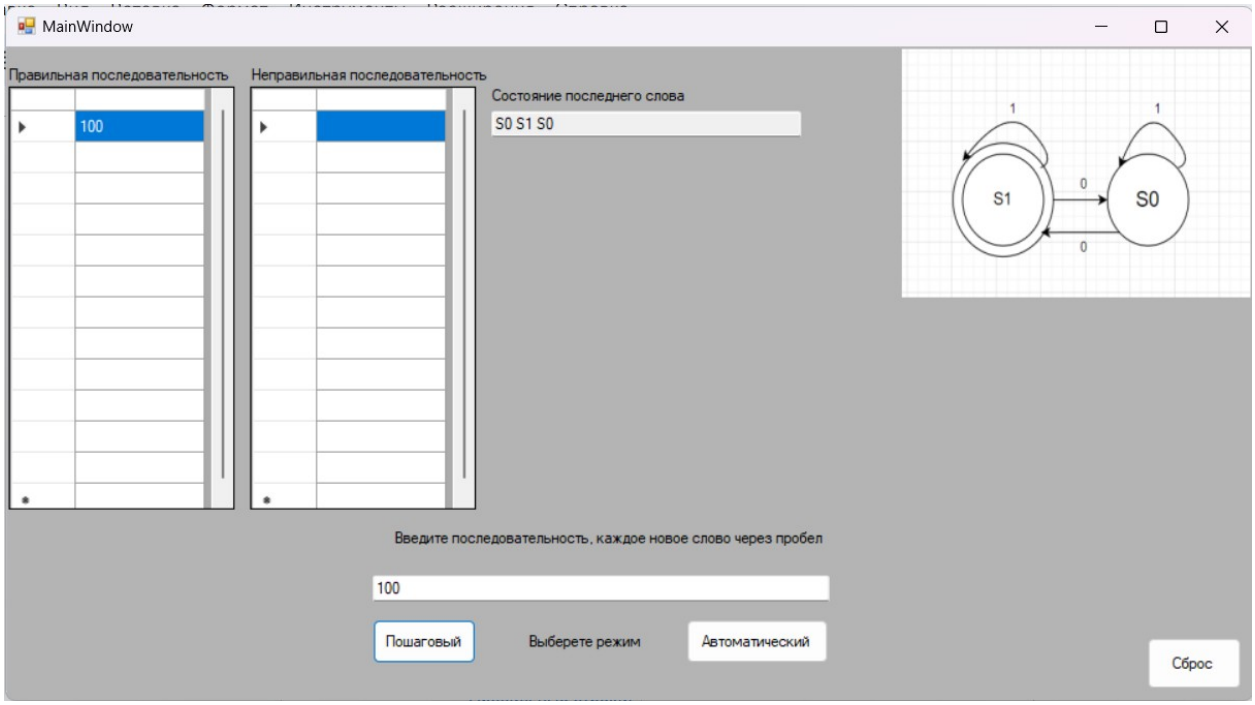

Рис. 20. Работа программы в пошаговом режиме

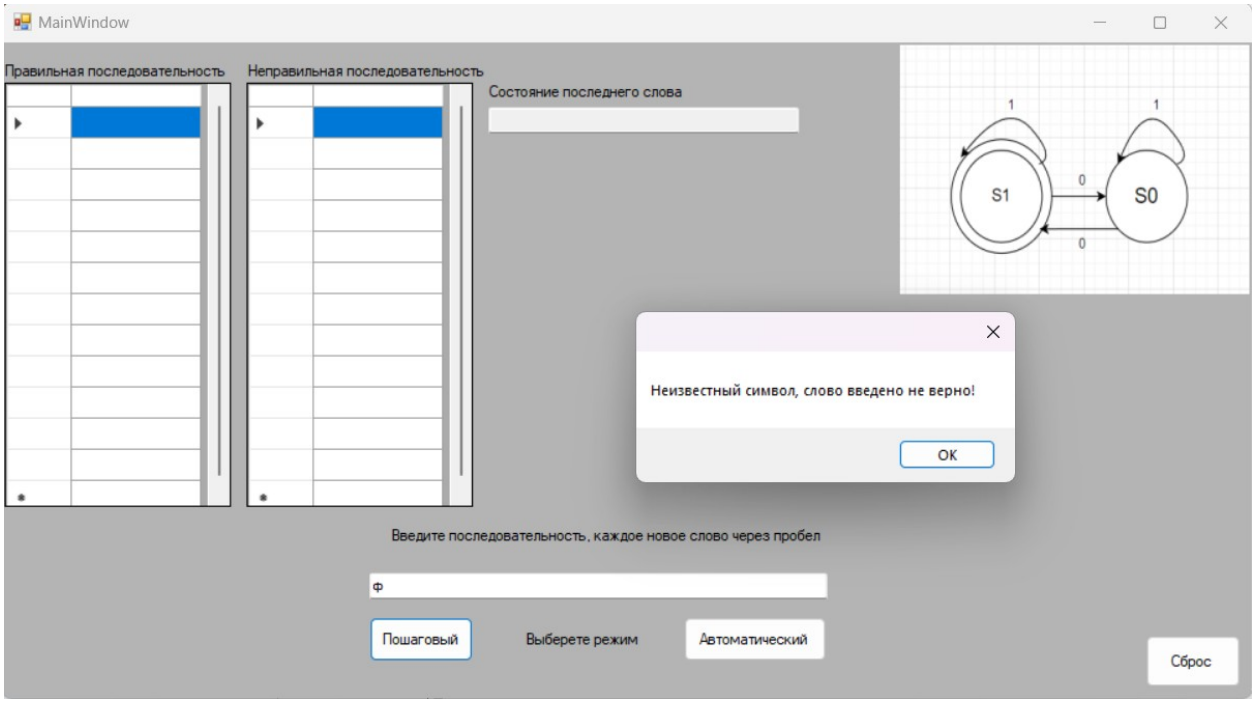

Рис. 21. Ошибка при вводе символа не из алфавита

6. Текст программы

Файл с описанием формы MyForm.cpp:

```
#include "MainWindow.h"

using namespace System;
using namespace System::Windows::Forms;

[STAThreadAttribute]
int main(array<String^>^ args)
{
	Application::SetCompatibleTextRenderingDefault(false);
	Application::EnableVisualStyles();
	Regex::MainWindow mainWindow;
	Application::Run(% mainWindow);
}
```

Файл State.cpp

```
enum class State {

	S0,

	S1

};
```

Файл Windows Forms:

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

namespace Regex {

	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 MainWindow
	/// </summary>
	public ref class MainWindow : public System::Windows::Forms::Form
	{
	public:
		MainWindow(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MainWindow()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::PictureBox^ imgAuto;
	private: System::Windows::Forms::TextBox^ input;
	private: System::Windows::Forms::Button^ step;
	private: System::Windows::Forms::Button^ autoStep;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Label^ label2;
	private: System::Windows::Forms::DataGridView^ correctInputs;
	private: System::Windows::Forms::DataGridViewTextBoxColumn^ Sequence;
	private: System::Windows::Forms::Label^ label3;
	private: System::Windows::Forms::Label^ label4;
	private: System::Windows::Forms::DataGridView^ unCorrectInputs;

	private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn1;
	private: System::Windows::Forms::TextBox^ stateCurrentWord;
	private: System::Windows::Forms::Label^ label5;
	private: System::Windows::Forms::Button^ reset;




	protected:

	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->imgAuto = (gcnew System::Windows::Forms::PictureBox());
			this->input = (gcnew System::Windows::Forms::TextBox());
```

```
                this->step = (gcnew System::Windows::Forms::Button());
                this->autoStep = (gcnew System::Windows::Forms::Button());
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->correctInputs = (gcnew System::Windows::Forms::DataGridView());
                this->Sequence = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->label4 = (gcnew System::Windows::Forms::Label());
                this->unCorrectInputs = (gcnew System::Windows::Forms::DataGridView());
                this->dataGridViewTextBoxColumn1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->stateCurrentWord = (gcnew System::Windows::Forms::TextBox());
                this->label5 = (gcnew System::Windows::Forms::Label());
                this->reset = (gcnew System::Windows::Forms::Button());
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->imgAuto))->BeginInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->correctInputs))->BeginInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->unCorrectInputs))->BeginInit();
                this->SuspendLayout();
                //
                // imgAuto
                //
                this->imgAuto->Location = System::Drawing::Point(926, 2);
                this->imgAuto->Name = L"imgAuto";
                this->imgAuto->Size = System::Drawing::Size(360, 236);
                this->imgAuto->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                this->imgAuto->TabIndex = 0;
                this->imgAuto->TabStop = false;
                //
                // input
                //
                this->input->Location = System::Drawing::Point(379, 503);
                this->input->Name = L"input";
                this->input->Size = System::Drawing::Size(471, 22);
                this->input->TabIndex = 1;
                //
                // step
                //
                this->step->Location = System::Drawing::Point(379, 545);
                this->step->Name = L"step";
                this->step->Size = System::Drawing::Size(106, 42);
                this->step->TabIndex = 2;
                this->step->Text = L"Пошаговый";
                this->step->UseVisualStyleBackColor = true;
                this->step->Click += gcnew System::EventHandler(this, &MainWindow::step_Click);
                //
                // autoStep
                //
                this->autoStep->Location = System::Drawing::Point(703, 545);
                this->autoStep->Name = L"autoStep";
                this->autoStep->Size = System::Drawing::Size(147, 42);
                this->autoStep->TabIndex = 3;
                this->autoStep->Text = L"Автоматический";
                this->autoStep->UseVisualStyleBackColor = true;
                this->autoStep->Click += gcnew System::EventHandler(this, &MainWindow::autoStep_Click);
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(536, 558);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(120, 16);
                this->label1->TabIndex = 4;
                this->label1->Text = L"Выберете режим ";
                //
                // label2
                //
                this->label2->AutoSize = true;
                this->label2->Location = System::Drawing::Point(398, 459);
                this->label2->Name = L"label2";
                this->label2->Size = System::Drawing::Size(437, 16);
                this->label2->TabIndex = 5;
                this->label2->Text = L"Введите последовательность, каждое новое слово через пробел";
                //
                // correctInputs
                //
                this->correctInputs->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                this->correctInputs->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(1) { this-
>Sequence });
                this->correctInputs->Location = System::Drawing::Point(2, 38);
                this->correctInputs->Name = L"correctInputs";
                this->correctInputs->RowHeadersWidth = 51;
                this->correctInputs->RowTemplate->Height = 24;
                this->correctInputs->Size = System::Drawing::Size(233, 402);
                this->correctInputs->TabIndex = 6;
                //
                // Sequence
                //
                this->Sequence->HeaderText = L"Последовательность";
                this->Sequence->MinimumWidth = 6;
                this->Sequence->Name = L"Sequence";
```

```
                                        this->Sequence->Width = 150;
                                        //
                                        // label3
                                        //
                                        this->label3->AutoSize = true;
                                        this->label3->Location = System::Drawing::Point(-1, 19);
                                        this->label3->Name = L"label3";
                                        this->label3->Size = System::Drawing::Size(228, 16);
                                        this->label3->TabIndex = 7;
                                        this->label3->Text = L"Правильная последовательность";
                                        //
                                        // label4
                                        //
                                        this->label4->AutoSize = true;
                                        this->label4->Location = System::Drawing::Point(249, 19);
                                        this->label4->Name = L"label4";
                                        this->label4->Size = System::Drawing::Size(244, 16);
                                        this->label4->TabIndex = 9;
                                        this->label4->Text = L"Неправильная последовательность";
                                        //
                                        // unCorrectInputs
                                        //
                                        this->unCorrectInputs->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                                        this->unCorrectInputs->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(1)
{ this->dataGridViewTextBoxColumn1 });
                                        this->unCorrectInputs->Location = System::Drawing::Point(252, 38);
                                        this->unCorrectInputs->Name = L"unCorrectInputs";
                                        this->unCorrectInputs->RowHeadersWidth = 51;
                                        this->unCorrectInputs->RowTemplate->Height = 24;
                                        this->unCorrectInputs->Size = System::Drawing::Size(233, 402);
                                        this->unCorrectInputs->TabIndex = 8;
                                        //
                                        // dataGridViewTextBoxColumn1
                                        //
                                        this->dataGridViewTextBoxColumn1->HeaderText = L"Последовательность";
                                        this->dataGridViewTextBoxColumn1->MinimumWidth = 6;
                                        this->dataGridViewTextBoxColumn1->Name = L"dataGridViewTextBoxColumn1";
                                        this->dataGridViewTextBoxColumn1->Width = 150;
                                        //
                                        // stateCurrentWord
                                        //
                                        this->stateCurrentWord->Location = System::Drawing::Point(502, 62);
                                        this->stateCurrentWord->Name = L"stateCurrentWord";
                                        this->stateCurrentWord->ReadOnly = true;
                                        this->stateCurrentWord->Size = System::Drawing::Size(318, 22);
                                        this->stateCurrentWord->TabIndex = 10;
                                        //
                                        // label5
                                        //
                                        this->label5->AutoSize = true;
                                        this->label5->Location = System::Drawing::Point(499, 38);
                                        this->label5->Name = L"label5";
                                        this->label5->Size = System::Drawing::Size(199, 16);
                                        this->label5->TabIndex = 11;
                                        this->label5->Text = L"Состояние последнего слова";
                                        //
                                        // reset
                                        //
                                        this->reset->Location = System::Drawing::Point(1179, 562);
                                        this->reset->Name = L"reset";
                                        this->reset->Size = System::Drawing::Size(97, 49);
                                        this->reset->TabIndex = 12;
                                        this->reset->Text = L"Сброс";
                                        this->reset->UseVisualStyleBackColor = true;
                                        this->reset->Click += gcnew System::EventHandler(this, &MainWindow::reset_Click);
                                        //
                                        // MainWindow
                                        //
                                        this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
                                        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                                        this->BackColor = System::Drawing::SystemColors::ActiveBorder;
                                        this->ClientSize = System::Drawing::Size(1288, 623);
                                        this->Controls->Add(this->reset);
                                        this->Controls->Add(this->label5);
                                        this->Controls->Add(this->stateCurrentWord);
                                        this->Controls->Add(this->label4);
                                        this->Controls->Add(this->unCorrectInputs);
                                        this->Controls->Add(this->label3);
                                        this->Controls->Add(this->correctInputs);
                                        this->Controls->Add(this->label2);
                                        this->Controls->Add(this->label1);
                                        this->Controls->Add(this->autoStep);
                                        this->Controls->Add(this->step);
                                        this->Controls->Add(this->input);
                                        this->Controls->Add(this->imgAuto);
                                        this->Name = L"MainWindow";
                                        this->Text = L"MainWindow";
                                        this->Load += gcnew System::EventHandler(this, &MainWindow::MainWindow_Load);
                                        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->imgAuto))->EndInit();
```

```
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->correctInputs))->EndInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->unCorrectInputs))->EndInit();
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion

        private: Image^ autoPath = Image::FromFile("C:\\Users\\Влад\\Desktop\\Университет\\Теория автоматов\\Regex\\automat.png");

        private: array<String^>^ X;
        private: int indexX = 0;
        private: int length = 0;
        private: State currentState = State::S0;
        private: int indexCorrectDataGrid = 0;
        private: int indexUncorrectDataGrid = 0;


        private: System::Void MainWindow_Load(System::Object^ sender, System::EventArgs^ e) {
            initPictures();
            initDataGrids();
        }

        private: void initPictures() {
            this->imgAuto->Image = autoPath;
        }
        private: void initDataGrids() {
            this->correctInputs->RowCount = 13;
            this->unCorrectInputs->RowCount = 13;
            this->indexCorrectDataGrid = 0;
            this->indexUncorrectDataGrid = 0;
        }
        private: System::Void step_Click(System::Object^ sender, System::EventArgs^ e) {
            if (this->input->Text->Length > 0 && (this->indexX == 0 || this->length != this->input->Text->Split()->Length)) {
                this->X = this->input->Text->Split();
                this->length = X->Length;
            }
            if (this->indexX == this->length) {
                MessageBox::Show("Слова закончились");
            }
            else {
                CheckWord(X[indexX++]);
            }
        }

        private: void CheckWord(String^ word) {
            String^ outputState;
            for (int i = 0; i < word->Length; i++)
            {
                switch (currentState) {
                    case State::S0:
                        if (word[i] == '1') {
                            outputState += "S0 ";
                            currentState = State::S0;
                        }
                        else if (word[i] == '0') {
                            outputState += "S1 ";
                            currentState = State::S1;
                        }
                        else {
                            MessageBox::Show("Неизвестный символ, слово введено не верно!");
                            return;
                        }
                        break;
                    case State::S1:
                        if (word[i] == '1') {
                            outputState += "S1 ";
                            currentState = State::S1;
                        }
                        else if (word[i] == '0') {
                            outputState += "S0 ";
                            currentState = State::S0;
                        }
                        else {
                            MessageBox::Show("Неизвестный символ, слово введено не верно!");
                            return;
                        }
                        break;
                    default:
                        break;
                }
            }
            if (currentState == State::S0) {
                this->correctInputs->Rows[indexCorrectDataGrid++]->Cells[0]->Value = word;
            }
            else {
                this->unCorrectInputs->Rows[indexUncorrectDataGrid++]->Cells[0]->Value = word;
            }
            this->stateCurrentWord->Text = outputState;
            currentState = State::S0;
```

```
	}
	private: System::Void reset_Click(System::Object^ sender, System::EventArgs^ e) {
		this->input->Text = "";
		this->indexX = 0;
		this->stateCurrentWord->Text = "";
		this->correctInputs->ColumnCount = 0;
		this->correctInputs->ColumnCount = 1;
		this->unCorrectInputs->ColumnCount = 0;
		this->unCorrectInputs->ColumnCount = 1;
		initDataGrids();
	}
	private: System::Void autoStep_Click(System::Object^ sender, System::EventArgs^ e) {
		for (int i = 0; i < this->input->Text->Split()->Length; i++)
		{
			step_Click(sender, e);
		}
	}
};
}
```

Вывод

В ходе выполнения лабораторной работы были решены задания из учебника Ю. Г. Карпова «Теория автоматов»: построены регулярные выражения, а также конечные автоматы, их распознающие. Также были построены графы для этих конечных автоматов.

Также было выполнено индивидуальное задание по вариантам: построено регулярное выражение, состоящее из символов 0 и 1, задающее множество слов, в которых четное число символов 0. Был построен конечный автомат для распознавания данного выражения и граф. Данный конечный автомат был смоделирован с помощью языка программирования C++.

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
5. Распознаватель – автомат распознает строки и принимает либо отвергает их