МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| Кафедра математического и программного обеспечения |
| Теория автоматов и формальных языков |
|  |

ЛАБОРАТОРНАЯ РАБОТА №1-2

«Построение модели автомата Мили»

|  |  |
| --- | --- |
| Исполнитель: | студент  группы 1ПИб-02-3оп-23  Богданов  Ренат Алексеевич |
| Руководитель: | Ганичева  Оксана Георгиевна |
| Оценка: |  |
| Подпись: |  |

2024 год

Задание

1. Построить свой вариант автомата «Электронный калькулятор». Автомат должен принимать цифры, операторы и выполнять арифметические операции, а также сбрасывать вычисления.

По заданному описанию:

1. определить составные компоненты автомата Мили (Х, У, S, s0, функция переходов, функция выходов)
2. построить граф переходов в терминологии автомата Мили (см. лекцию. У вас он будет отличаться от представления в учебнике)
3. построить таблицу переходов
4. построить таблицу выходов
5. построить эквивалентный автомат Мура
6. рассмотреть произвольную входную последовательность (длиной не менее 15 символов) и построить протокол работы своего автомата для этой последовательности. Построить этот протокол работы и для автомата Мили, и для автомата Мура
7. Написать программу на языке высокого уровня, моделирующую работу автомата, построенного в 1-ой работе (калькулятор).

В программе должны быть предусмотрены:

1. реализация с использованием оператора выбора Switch (внешний выбор по состояниям, внутренний - по входным символам);
2. ввод исходной строки входных символов (произвольной длины);
3. отображение информации по входным символам, внутренним состояниям, выходным символам (протокол работы автомата - см. лекции).

Введение

В данной лабораторной работе рассматривается проектирование и реализация автомата, моделирующего работу простого калькулятора. Целью работы является создание программы, управляемой конечным автоматом, выполняющей базовые арифметические операции. В процессе выполнения работы будут изучены основные принципы работы конечных автоматов, такие как состояния, входные и выходные сигналы, а также переходы между состояниями в зависимости от входных данных.

Реализация программы будет выполнена на языке программирования C++ с использованием инструментов разработки Windows Forms для создания графического интерфейса. В программе будут реализованы автоматический и ручной режимы работы калькулятора, демонстрирующие работу конечного автомата в различных условиях.

Формальное описание модели

Автомат моделирует работу простого калькулятора, который принимает последовательность команд (ввод цифр, операторов и специальных сигналов) и выполняет соответствующие действия. Внутреннее состояние автомата меняется в зависимости от входных символов, а выходные сигналы отражают текущее состояние и операции, выполняемые калькулятором. Автомат поддерживает базовые арифметические операции и имеет состояния для принятия цифр, операторов, записи результатов и сброса вычислений. В программе предусмотрены как автоматический режим, где команды вводятся строкой, так и ручной режим, где команды выбираются из интерфейса пользователя.

Формально абстрактный автомат определяется как набор компонентов следующего вида: A= (S, X, Y, S0, δ, λ), где:

x – конечное непустое множество входных символов (входной алфавит);

y – конечное непустое множество выходных символов (выходной алфавит);

S – конечное непустое множество состояний;

s0 ∈ S – начальное состояние;

δ: S×x→S – функция переходов автомата из одного состояния в другое;

λ: S×x→y – функция выходов.

Закон функционирования автомата Мили: S(t+1) = δ(S(t), x(t)); y(t) = λ(S(t), x(t)), где: t – текущий момент времени; t+1 – следующий момент времени; S(t+1) – состояние автомата в следующем моменте времени; S(t), x(t), y(t) – элементы описания автомата в текущем моменте времени.

Модель автомата Мили:

Состояния автомата (S):

* S0 – начальное состояние;
* S1 – принятие цифры;
* S2 – запись результата в память;
* S3 – принятие знака операции;
* S4 – принятие знака “=”.

Входные сигналы (x):

* X1 – ввод цифры;
* X2 – ввод знака оператора;
* X3 – помещение оператора в память;
* X4 – подсчёт итога;
* X5 – сброс.

Выходные сигналы (y):

* Y1 – вывод на экран;
* Y2 – вывод знака оператора на экран;
* Y3 – запись операции в память;
* Y4 – вывод результата на экран;
* Y5 – очистка памяти.

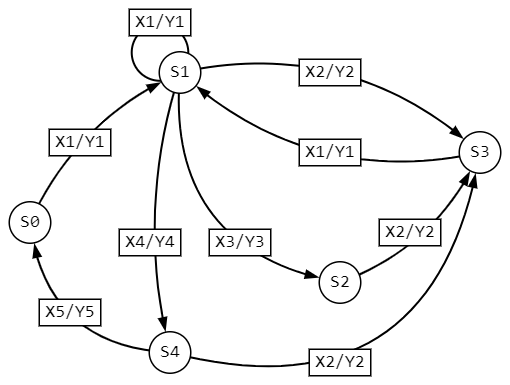


Рис. 1. Автомат устройства управления калькулятором (Автомат Мили)

Таблица 1. Переходы автомата

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ | S0 | S1 | S2 | S3 | S4 |
| X1 | S1 | S1 |  | S1 |  |
| X2 |  | S3 | S3 |  | S3 |
| X3 |  | S2 |  |  |  |
| X4 |  | S4 |  |  |  |
| X5 |  |  |  |  | S0 |

Таблица 2. Выходы автомата

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| λ | S0 | S1 | S2 | S3 | S4 |
| X1 | Y1 | Y1 |  | Y1 |  |
| X2 |  | Y3 | Y3 |  | Y3 |
| X3 |  | Y2 |  |  |  |
| X4 |  | Y4 |  |  |  |
| X5 |  |  |  |  | Y0 |

Таблица 3. Совмещённая таблица

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| δ/ λ | S0 | S1 | S2 | S3 | S4 |
| X1 | S1/Y1 | S1/Y1 |  | S1/Y1 |  |
| X2 |  | S3/Y3 | S3/Y3 |  | S3/Y3 |
| X3 |  | S2/Y2 |  |  |  |
| X4 |  | S4/Y4 |  |  |  |
| X5 |  |  |  |  | S0/Y0 |

Таблица 4. Протокол работы автомата Мили

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| S | S0 | S1 | S1 | S3 | S1 | S4 | S3 | S1 | S4 | S0 | S1 | S2 | S3 | S1 | S3 | S1 |
| X | X1 | X1 | X2 | X1 | X4 | X2 | X1 | X4 | X5 | X1 | X3 | X2 | X1 | X2 | X1 |  |
| Y | Y1 | Y1 | Y2 | Y1 | Y4 | Y2 | Y1 | Y4 | Y5 | Y1 | Y3 | Y2 | Y1 | Y2 | Y1 |  |

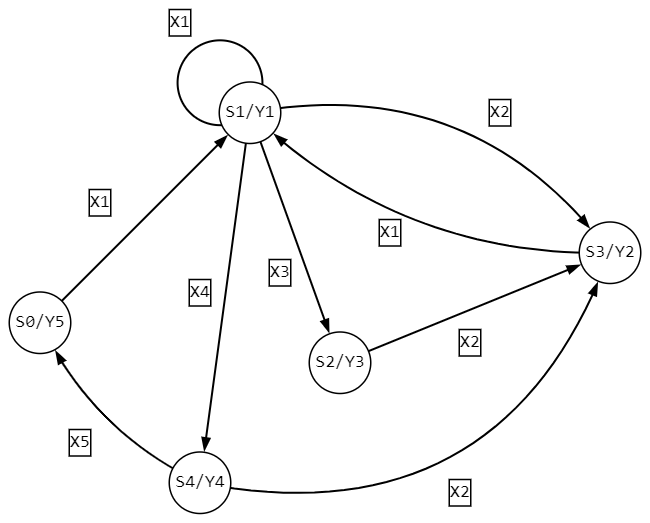


Рис 2. Автомат Мура

Таблица 5. Протокол работы автомата Мура

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| S | S0 | S1 | S1 | S3 | S1 | S4 | S3 | S1 | S4 | S0 | S1 | S2 | S3 | S1 | S3 | S1 |
| X | X1 | X1 | X2 | X1 | X4 | X2 | X1 | X4 | X5 | X1 | X3 | X2 | X1 | X2 | X1 |  |
| Y | Y1 | Y1 | Y2 | Y1 | Y4 | Y2 | Y1 | Y4 | Y5 | Y1 | Y3 | Y2 | Y1 | Y2 | Y1 |  |

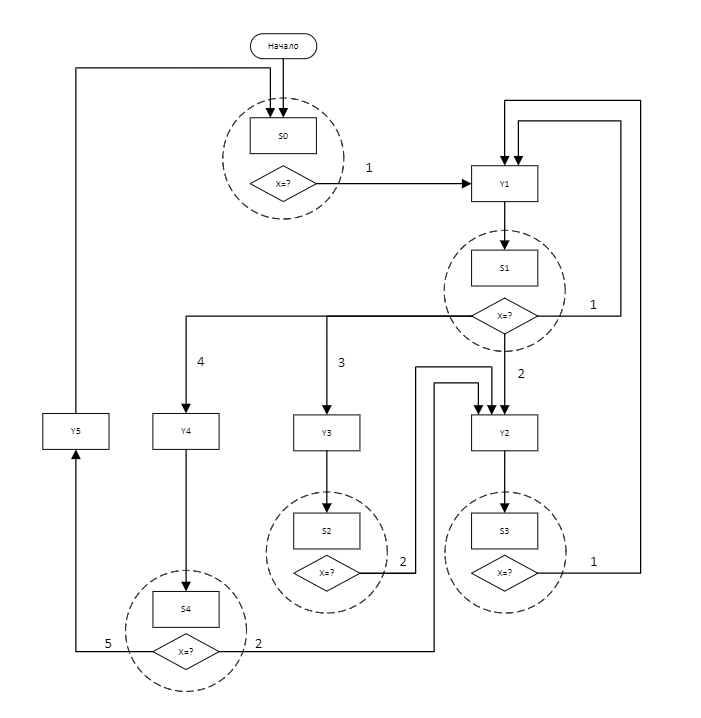


Рис. 3. Структурная схема автомата

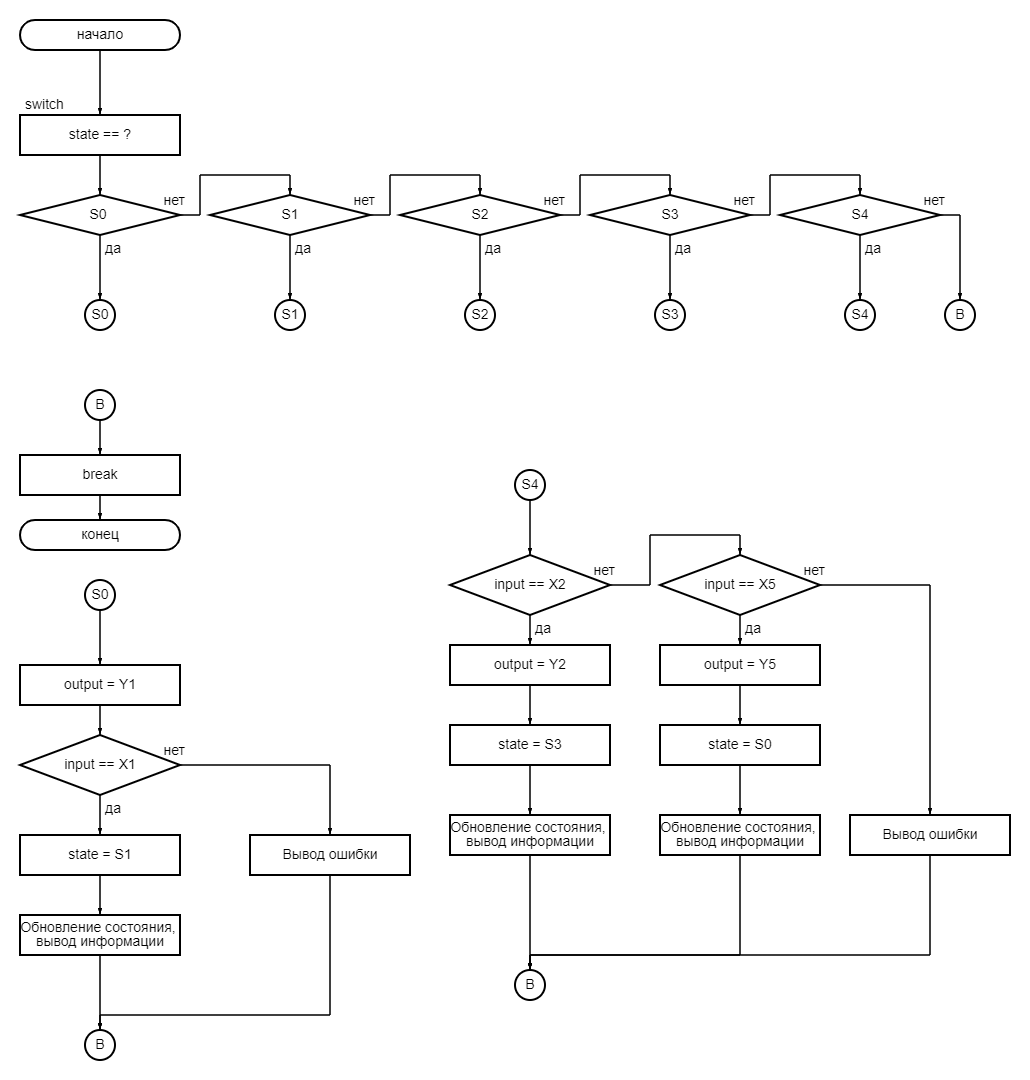


Рис. 4. Блок-схема оператора switch (часть 1)

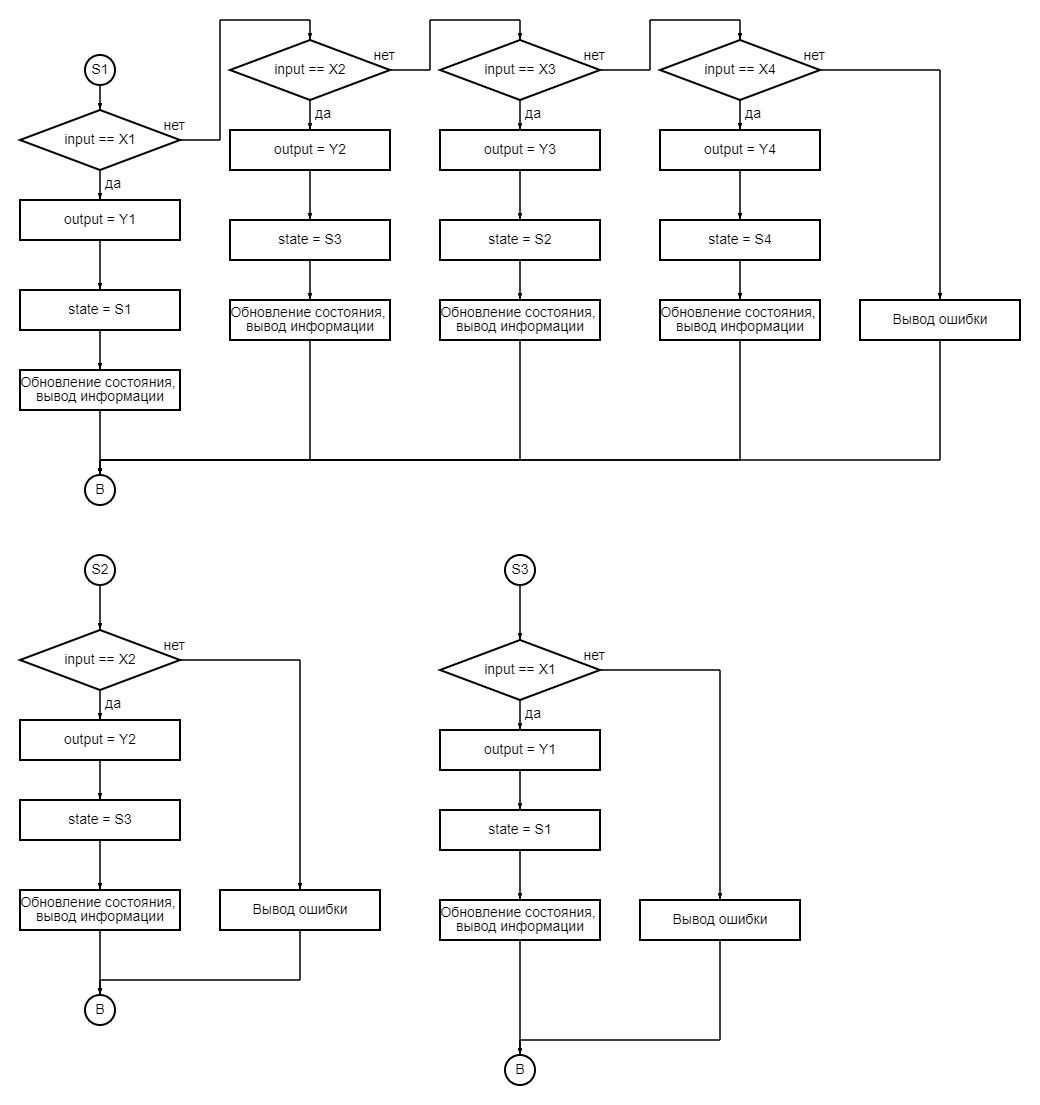


Рис. 5. Блок-схема оператора switch (часть 2)

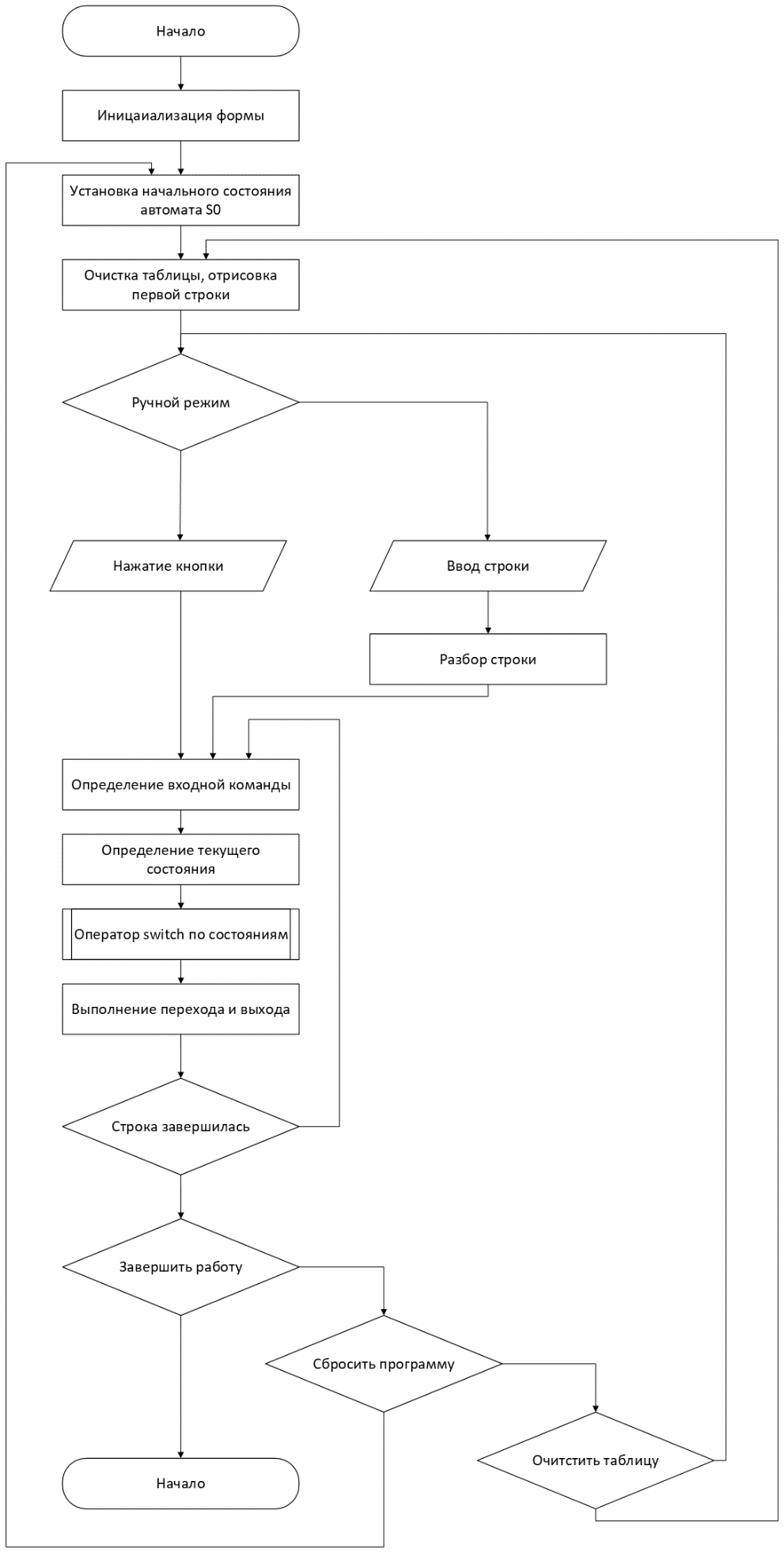


Рис. 6. Блок-схема программы

Программное моделирование

Код программы

CppCliWinForm.cpp

#include "pch.h"

#include "Form1.h"

using namespace CppCliWinForm;

[System::STAThreadAttribute]

int main(array<System::String^>^ args)

{

// Enabling Windows XP visual effects before any controls are created

System::Windows::Forms::Application::EnableVisualStyles();

System::Windows::Forms::Application::SetCompatibleTextRenderingDefault(false);

// Create the main window and run it

System::Windows::Forms::Application::Run(gcnew Form1());

return 0;

}

Form1.h

#pragma once

#include <string>

namespace CppCliWinForm {

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>

/// Summary for Form1

/// </summary>

public ref class Form1 : public System::Windows::Forms::Form

{

public:

Form1(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~Form1()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::PictureBox^ pictureBox1;

protected:

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

private: System::Windows::Forms::ToolTip^ toolTip1;

private: System::Windows::Forms::PictureBox^ pictureBox2;

private: System::Windows::Forms::Panel^ panel1;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ table\_s;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ table\_x;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ table\_y;

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

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Panel^ panel2;

private: System::Windows::Forms::Button^ button5;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::ComponentModel::IContainer^ components;

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

int currentState;

public:

enum class State {

S0, // Начальное состояние

S1, // Принятие цифры

S2, // Запись результата в память

S3, // Принятие знака операции

S4 // Принятие знака "="

};

enum class Input {

zi,

X1,

X2,

X3,

X4,

X5

};

enum class Output {

zo,

Y1,

Y2,

Y3,

Y4,

Y5

};

State state = State::S0;

Input input;

Output output;

private: System::Windows::Forms::PictureBox^ pictureBox3;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Button^ button6;

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

private: System::Windows::Forms::Button^ button7;

private: System::Windows::Forms::Button^ button8;

public:

bool isRunning = true;

private: System::Void ProcessInput(Input input) {

switch (state) {

case State::S0:

output = Output::Y1;

if (input == Input::X1) {

state = State::S1;

UpdateState();

}

else {

ShowErrorMessage();

}

break;

case State::S1:

if (input == Input::X1) {

output = Output::Y1;

state = State::S1;

UpdateState();

}

else if (input == Input::X2) {

output = Output::Y2;

state = State::S3;

UpdateState();

}

else if (input == Input::X3) {

output = Output::Y3;

state = State::S2;

UpdateState();

}

else if (input == Input::X4) {

output = Output::Y4;

state = State::S4;

UpdateState();

}

else {

ShowErrorMessage();

}

break;

case State::S2:

if (input == Input::X2) {

output = Output::Y2;

state = State::S3;

UpdateState();

}

else {

ShowErrorMessage();

}

break;

case State::S3:

if (input == Input::X1) {

output = Output::Y1;

state = State::S1;

UpdateState();

}

else {

ShowErrorMessage();

}

break;

case State::S4:

if (input == Input::X2) {

output = Output::Y2;

state = State::S3;

UpdateState();

}

else if (input == Input::X5) {

output = Output::Y5;

state = State::S0;

UpdateState();

}

else {

ShowErrorMessage();

}

break;

}

}

private: System::Void UpdateState() {

richTextBox1->AppendText("State: " + static\_cast<int>((int)state) + Environment::NewLine);

richTextBox1->AppendText("Output: " + static\_cast<int>((int)output) + Environment::NewLine);

richTextBox1->ScrollToCaret();

if (dataGridView1->Rows->Count > 0) {

int lastRowIndex = dataGridView1->Rows->Count - 1;

dataGridView1->Rows[lastRowIndex]->Cells["table\_x"]->Value = static\_cast<int>(input);

dataGridView1->Rows[lastRowIndex]->Cells["table\_y"]->Value = static\_cast<int>(output);

}

auto rowIndex = dataGridView1->Rows->Add();

dataGridView1->Rows[rowIndex]->Cells["table\_s"]->Value = static\_cast<int>(state);

dataGridView1->Rows[rowIndex]->Cells["table\_x"]->Value = "";

dataGridView1->Rows[rowIndex]->Cells["table\_y"]->Value = "";

dataGridView1->FirstDisplayedScrollingRowIndex = rowIndex;

}

private: System::Void ShowErrorMessage() {

richTextBox1->AppendText("Error! Try again" + Environment::NewLine);

richTextBox1->ScrollToCaret();

MessageBox::Show("Недопустимая команда!", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(Form1::typeid));

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

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

this->toolTip1 = (gcnew System::Windows::Forms::ToolTip(this->components));

this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());

this->panel1 = (gcnew System::Windows::Forms::Panel());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->table\_s = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_x = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->table\_y = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->button2 = (gcnew System::Windows::Forms::Button());

this->panel2 = (gcnew System::Windows::Forms::Panel());

this->button5 = (gcnew System::Windows::Forms::Button());

this->button4 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->pictureBox3 = (gcnew System::Windows::Forms::PictureBox());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->button6 = (gcnew System::Windows::Forms::Button());

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

this->button7 = (gcnew System::Windows::Forms::Button());

this->button8 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->BeginInit();

this->panel1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->panel2->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox3))->BeginInit();

this->SuspendLayout();

//

// pictureBox1

//

this->pictureBox1->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.Image")));

this->pictureBox1->Location = System::Drawing::Point(4, 3);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(435, 343);

this->pictureBox1->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureBox1->TabIndex = 0;

this->pictureBox1->TabStop = false;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Consolas", 15.75F, 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(262, 24);

this->label1->TabIndex = 1;

this->label1->Text = L"Автомат «Калькулятор»";

//

// pictureBox2

//

this->pictureBox2->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox2.Image")));

this->pictureBox2->Location = System::Drawing::Point(445, 3);

this->pictureBox2->Name = L"pictureBox2";

this->pictureBox2->Size = System::Drawing::Size(430, 343);

this->pictureBox2->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureBox2->TabIndex = 2;

this->pictureBox2->TabStop = false;

//

// panel1

//

this->panel1->Controls->Add(this->pictureBox1);

this->panel1->Controls->Add(this->pictureBox2);

this->panel1->Location = System::Drawing::Point(16, 36);

this->panel1->Name = L"panel1";

this->panel1->Size = System::Drawing::Size(879, 354);

this->panel1->TabIndex = 3;

//

// dataGridView1

//

this->dataGridView1->AllowUserToAddRows = false;

this->dataGridView1->AllowUserToDeleteRows = false;

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(3) {

this->table\_s,

this->table\_x, this->table\_y

});

this->dataGridView1->Location = System::Drawing::Point(901, 36);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->ReadOnly = true;

this->dataGridView1->Size = System::Drawing::Size(343, 510);

this->dataGridView1->TabIndex = 4;

this->dataGridView1->Rows->Add(static\_cast<int>(State::S0), "", "");

//

// table\_s

//

this->table\_s->HeaderText = L"S";

this->table\_s->Name = L"table\_s";

this->table\_s->ReadOnly = true;

//

// table\_x

//

this->table\_x->HeaderText = L"X";

this->table\_x->Name = L"table\_x";

this->table\_x->ReadOnly = true;

//

// table\_y

//

this->table\_y->HeaderText = L"Y";

this->table\_y->Name = L"table\_y";

this->table\_y->ReadOnly = true;

//

// button1

//

this->button1->Location = System::Drawing::Point(3, 0);

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

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 6;

this->button1->Text = L"X1";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &Form1::button1\_Click\_1);

//

// button2

//

this->button2->Location = System::Drawing::Point(84, 0);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(75, 23);

this->button2->TabIndex = 7;

this->button2->Text = L"X2";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &Form1::button2\_Click);

//

// panel2

//

this->panel2->Controls->Add(this->button5);

this->panel2->Controls->Add(this->button4);

this->panel2->Controls->Add(this->button3);

this->panel2->Controls->Add(this->button1);

this->panel2->Controls->Add(this->button2);

this->panel2->Location = System::Drawing::Point(461, 429);

this->panel2->Name = L"panel2";

this->panel2->Size = System::Drawing::Size(418, 34);

this->panel2->TabIndex = 8;

//

// button5

//

this->button5->Location = System::Drawing::Point(328, 0);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(75, 23);

this->button5->TabIndex = 10;

this->button5->Text = L"X5";

this->button5->UseVisualStyleBackColor = true;

this->button5->Click += gcnew System::EventHandler(this, &Form1::button5\_Click);

//

// button4

//

this->button4->Location = System::Drawing::Point(247, 0);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(75, 23);

this->button4->TabIndex = 9;

this->button4->Text = L"X4";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &Form1::button4\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(165, 0);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(75, 23);

this->button3->TabIndex = 8;

this->button3->Text = L"X3";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &Form1::button3\_Click);

//

// richTextBox1

//

this->richTextBox1->Location = System::Drawing::Point(20, 455);

this->richTextBox1->ReadOnly = true;

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(403, 117);

this->richTextBox1->TabIndex = 9;

this->richTextBox1->Text = L"";

this->richTextBox1->Font = (gcnew System::Drawing::Font(L"Consolas", 14.0F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

//

// pictureBox3

//

this->pictureBox3->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox3.Image")));

this->pictureBox3->Location = System::Drawing::Point(461, 469);

this->pictureBox3->Name = L"pictureBox3";

this->pictureBox3->Size = System::Drawing::Size(429, 103);

this->pictureBox3->TabIndex = 10;

this->pictureBox3->TabStop = false;

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(20, 429);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(403, 20);

this->textBox1->TabIndex = 11;

//

// button6

//

this->button6->Location = System::Drawing::Point(25, 399);

this->button6->Name = L"button6";

this->button6->Size = System::Drawing::Size(398, 23);

this->button6->TabIndex = 12;

this->button6->Text = L"Автоматический режим";

this->button6->UseVisualStyleBackColor = true;

this->button6->Click += gcnew System::EventHandler(this, &Form1::button6\_Click);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(460, 413);

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

this->label2->Size = System::Drawing::Size(79, 13);

this->label2->TabIndex = 13;

this->label2->Text = L"Ручной режим";

//

// button7

//

this->button7->Location = System::Drawing::Point(901, 548);

this->button7->Name = L"button7";

this->button7->Size = System::Drawing::Size(222, 23);

this->button7->TabIndex = 14;

this->button7->Text = L"Очистка";

this->button7->UseVisualStyleBackColor = true;

this->button7->Click += gcnew System::EventHandler(this, &Form1::button7\_Click);

//

// button8

//

this->button8->Location = System::Drawing::Point(1129, 548);

this->button8->Name = L"button8";

this->button8->Size = System::Drawing::Size(115, 23);

this->button8->TabIndex = 15;

this->button8->Text = L"Перезапуск";

this->button8->UseVisualStyleBackColor = true;

this->button8->Click += gcnew System::EventHandler(this, &Form1::button8\_Click);

//

// Form1

//

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

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

this->AutoSize = true;

this->ClientSize = System::Drawing::Size(1260, 609);

this->Controls->Add(this->button8);

this->Controls->Add(this->button7);

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

this->Controls->Add(this->button6);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->panel2);

this->Controls->Add(this->pictureBox3);

this->Controls->Add(this->richTextBox1);

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->panel1);

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

this->Name = L"Form1";

this->Text = L"Form1";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->EndInit();

this->panel1->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

this->panel2->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox3))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

input = Input::X1;

ProcessInput(input);

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

input = Input::X2;

ProcessInput(input);

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

input = Input::X3;

ProcessInput(input);

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

input = Input::X4;

ProcessInput(input);

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

input = Input::X5;

ProcessInput(input);

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

// Получаем введенную строку

String^ inputString = textBox1->Text;

// Разделяем строку по пробелам

array<String^>^ commands = inputString->Split(' ');

// Обрабатываем каждую команду

for each (String ^ command in commands) {

// Удаляем лишние пробелы и приводим к нижнему регистру для упрощения проверки

command = command->Trim()->ToLower();

if (command == "x1") {

input = Input::X1;

}

else if (command == "x2") {

input = Input::X2;

}

else if (command == "x3") {

input = Input::X3;

}

else if (command == "x4") {

input = Input::X4;

}

else if (command == "x5") {

input = Input::X5;

}

else {

richTextBox1->AppendText("Unknown command: " + command + Environment::NewLine);

MessageBox::Show("Недопустимая команда: " + command, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

continue;

}

ProcessInput(input);

}

}

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) {

// Очистка таблицы

dataGridView1->Rows->Clear();

// Добавление текущего состояния на первую строку

auto rowIndex = dataGridView1->Rows->Add();

dataGridView1->Rows[rowIndex]->Cells["table\_s"]->Value = static\_cast<int>(state);

dataGridView1->Rows[rowIndex]->Cells["table\_x"]->Value = "";

dataGridView1->Rows[rowIndex]->Cells["table\_y"]->Value = "";

}

private: System::Void button8\_Click(System::Object^ sender, System::EventArgs^ e) {

// Сброс состояния

state = State::S0;

input = Input::zi;

output = Output::zo;

// Очистка таблицы

dataGridView1->Rows->Clear();

// Добавление начального состояния на первую строку

auto rowIndex = dataGridView1->Rows->Add();

dataGridView1->Rows[rowIndex]->Cells["table\_s"]->Value = static\_cast<int>(state);

dataGridView1->Rows[rowIndex]->Cells["table\_x"]->Value = "";

dataGridView1->Rows[rowIndex]->Cells["table\_y"]->Value = "";

}

};

}

Примеры работы программы

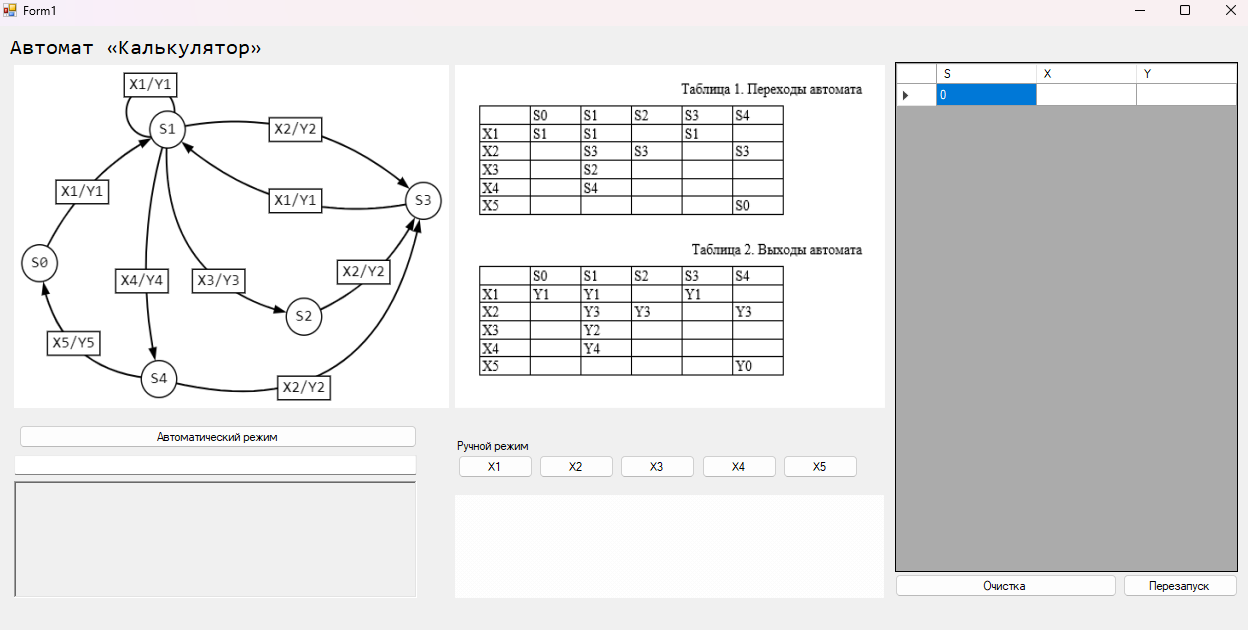


Рис. 7. Интерфейс программы

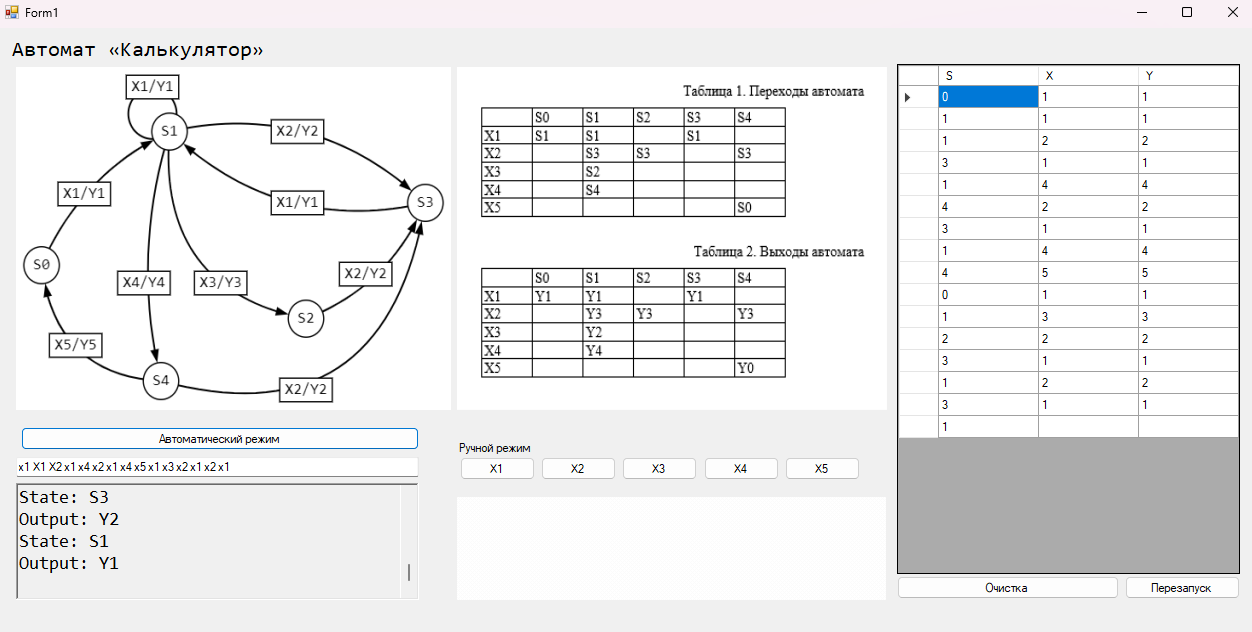


Рис. 8. Работа автомата в автоматическом режиме (протокол соответствует табл. 4)

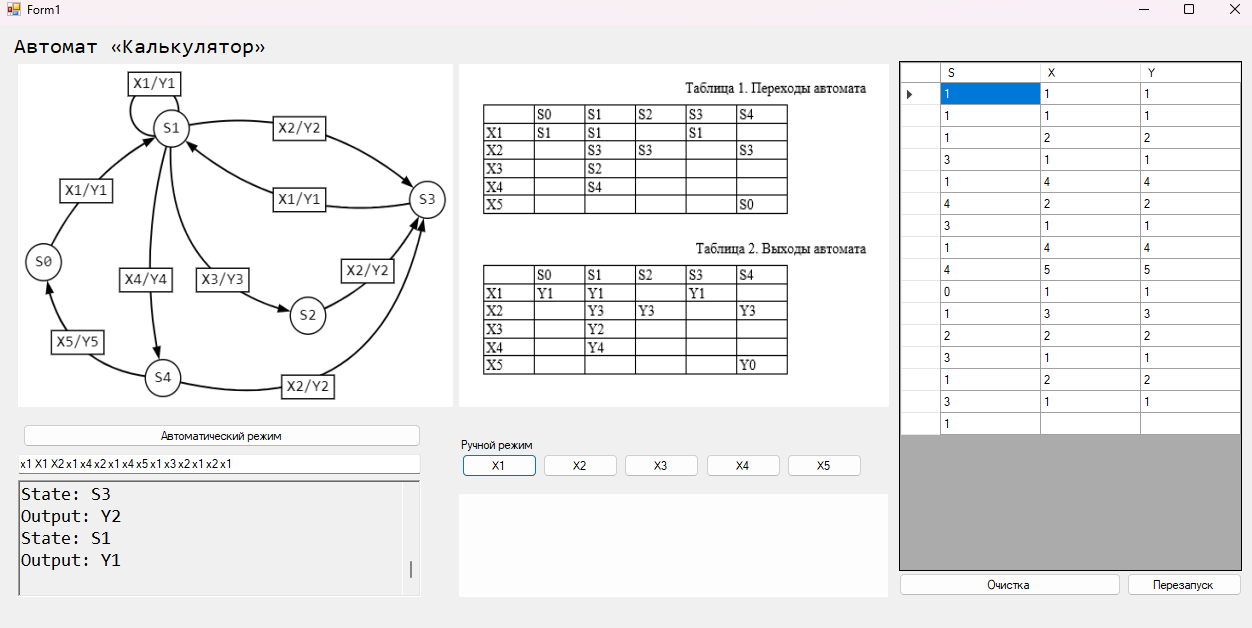


Рис. 9. Работа автомата в ручном режиме (протокол соответствует табл. 4)

Вывод

В ходе выполнения лабораторной работы была успешно реализована программа, моделирующая работу простого калькулятора с использованием автоматов Мили и Мура. Были изучены основные принципы работы автоматов, такие как состояния, входные и выходные сигналы, а также переходы между состояниями в зависимости от входных данных. Реализация программы на языке C++ с использованием Windows Forms позволила создать графический интерфейс, поддерживающий как автоматический, так и ручной режим работы калькулятора. Это позволило закрепить теоретические знания на практике и развить навыки программирования и моделирования систем.

Автомат, построенный в данной лабораторной работе, является автоматом Мили, который моделирует работу простого калькулятора. Характеристики данного автомата включают следующие аспекты:

1. детерминированным, так как для каждого входного сигнала есть только один вариант перехода;
2. асинхронным, так как входные сигналы поступают в любой момент времени;
3. частичными, так как присутствует неопределенность выходов, т.е. значения выходных символов при определенных состояниях автомата не определены и присутствуют ограничения на входе, т.е. не каждый входной символ допустим при определенных состояниях;
4. абстрактный, так как внешняя модель поведения автомата.