Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 «Прикладная информатика»**

О Т Ч Е Т

по домашнему заданию №1

Название: Эквивалентность

Дисциплина: Прикладная теория цифровых автоматов

Вариант 5

Студент	ИУ6-44Б	18.05.2023	А.И. Гарифуллин
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Е.Ю. Гаврилова
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

1. Содержание

2. Введение

В настоящей работе выполнена реализация цифрового автомата «Бег 100 метров».

Существуют 2 способа реализации автомата: программный и аппаратный. Программная реализация выполняется на любом языке высокого уровня. Аппаратная реализация – предусматривает построение устройств памяти для запоминания текущего состояния автомата, в роли которых обычно используются триггеры.

В настоящей работе использован программный способ реализации цифрового автомата, так как этот способ подразумевает вариативность реализации, возможность отладки и тестирования в процессе разработки программы. К программам (в отличие от аппаратной реализации цифровых автоматов) можно добавлять новые функции по мере изменения целей, под которые она разрабатывается.

Задание (вариант 5)

Реализовать программно автомат, осуществляющий проверку автоматов на эквивалентность.

Цель работы

Закрепить навыки реализации конечных цифровых автоматов. Для реализации построенной цели необходимо выполнить следующие задачи.

Задачи

- Изучить задание в соответствии со своим вариантом;
- Описать автомат, соответствующий условию задачи;
- Изучить способы реализации цифровых автоматов;
- Выбрать один из способов реализации автоматов;
- Реализовать описанный цифровой автомат.

3. Основная часть

Проанализируем, что должен делать пользователь. Необходимо заполнить автомат (таблицу состояний и переходов). После нажать на кнопку “Проверить эквивалентность”. Получить ответ: является ли введенный автомат эквивалентным предложенному. Возможно сбросить ответ и введенный автомат.

На основе данного анализа составим конечный цифровой автомат.

Спецификация автомата

1. Состояния автомата:
   - q0 – начальное состояние автомата
   - q1 – введен автомат
   - q2 – программа выполняет алгоритм проверки
   - q3 – выдан ответ
2. Входные сигналы
   - a – введены данные
   - b – нажата кнопка “Проверить эквивалентность”
   - c – нажата кнопка “Сбросить”
   - d – программа вернула true
   - e – программа вернула false
3. Выходные сигналы
   - 0 – автоматы не эквивалентны
   - 1 – автоматы эквивалентны
   - 2 – ожидание

Полученный цифровой автомат

Составим таблицу, описывающую конечный автомат, составленный по условию задачи в результате проведенного анализа (таблица 1).

Таблица 1 – Таблица переходов результирующего автомата

Состояние	$\boldsymbol{\delta}$					$\boldsymbol{\lambda}$				
	a	b	c	d	e	a	b	c	d	e
q0	q1	q2	q0	-	-	2	2	2	-	-
q1	q1	q2	q0	-	-	2	2	2	-	-
q2	-	-	-	q3	q3	-	-	-	1	0
q3	q1	q2	q0	-	-	2	2	2	-	-

Продемонстрируем автомат в виде графа переходов (рисунок 1).

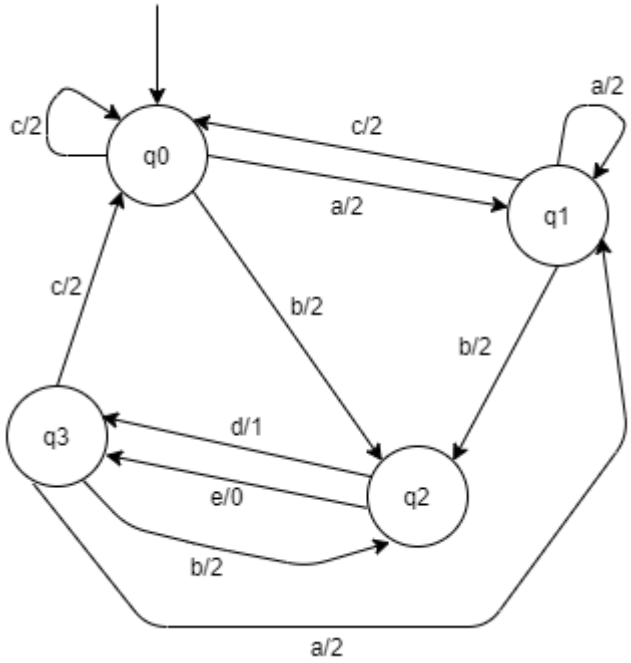

Рисунок 1 - Граф переходов результирующего автомата

Реализация результирующего автомата

Для реализации описанного цифрового автомата была разработана схема алгоритма (рисунок 2) программы, которая впоследствии была реализована на языке C++ с помощью фреймворка Qt.

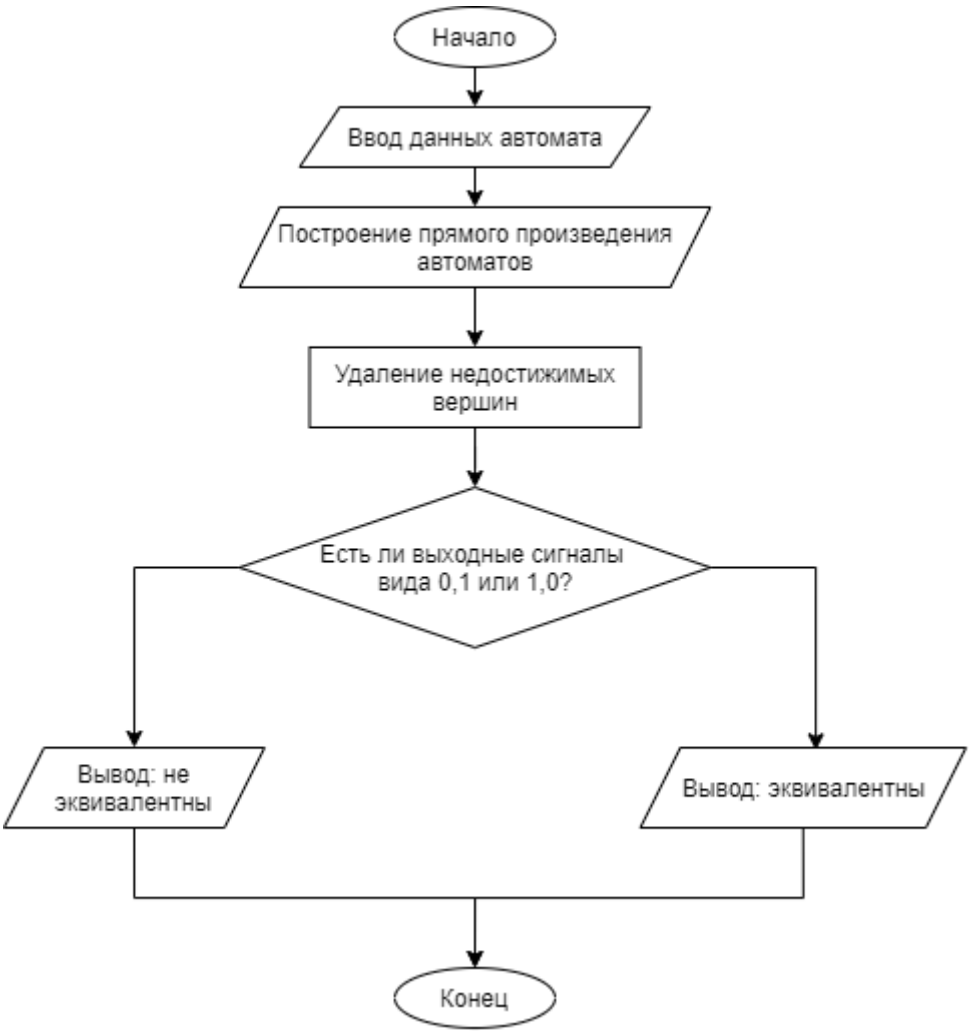

Рисунок 2 - Схема алгоритма программы, реализующей автомат

Код:

Файл mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();
```

```
    void on_pushButton_2_clicked();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Файл main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <bits/stdc++.h>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_clicked()
{
    char perehodS[4][4] ={
        {'0', 'a', '0', '1' },
        {'0', 'b', '1', '0' },
        {'1', 'a', '0', '1' },
        {'1', 'b', '0', '0' }
    };
    char perehodQ[6][4] = {
        {'0', 'a', '0', '1' },
        {'0', 'b', '1', '0' },
        {'1', 'a', '0', '1' },
        {'1', 'b', '0', '2' },
        {'2', 'a', '0', '1' },
        {'2', 'b', '1', '0' }
    };

    perehodQ[0][2] = ui->comboBox_7->currentIndex() + '0';
    perehodQ[0][3] = ui->comboBox->currentIndex() + '0';
    perehodQ[1][2] = ui->comboBox_10->currentIndex() + '0';
```

```
    perehodQ[1][3] = ui->comboBox_4->currentIndex() + '0';
    perehodQ[2][2] = ui->comboBox_8->currentIndex() + '0';
    perehodQ[2][3] = ui->comboBox_2->currentIndex() + '0';
    perehodQ[3][2] = ui->comboBox_11->currentIndex() + '0';
    perehodQ[3][3] = ui->comboBox_5->currentIndex() + '0';
    perehodQ[4][2] = ui->comboBox_9->currentIndex() + '0';
    perehodQ[4][3] = ui->comboBox_3->currentIndex() + '0';
    perehodQ[5][2] = ui->comboBox_12->currentIndex() + '0';
    perehodQ[5][3] = ui->comboBox_6->currentIndex() + '0';
    std::vector<std::vector<char>> v;

    for(int i = 0; i < 4; ++i)
    {
        if(i%2==0){
            for(int j = 0; j < 6; j = j + 2)
            {
                v.push_back({perehodS[i][0], perehodQ[j][0], perehodS[i][1],
perehodS[i][2], perehodQ[j][1], perehodQ[j][2], perehodS[i][3],
perehodQ[i][3]});
            }
        }
        else{
            for(int j = 1; j < 6; j = j + 2)
            {
                v.push_back({perehodS[i][0], perehodQ[j][0], perehodS[i][1],
perehodS[i][2], perehodQ[j][1], perehodQ[j][2], perehodS[i][3],
perehodQ[i][3]});
            }
        }
    }
    int sost[6] = {1,0,0,0,0,0};
    for(int i = 0; i < 12; ++i)
    {
        sost[(v[i][6]-'0')*3 + (v[i][7]-'0')] += 1;
    }

    bool otvet = true;
    for(int i = 0; i < 6; ++i)
    {
        if(sost[i] > 0)
        {
            for(int j = 0; j < 12; ++j)
            {
                if(((v[j][0]-'0')*3 + (v[j][1]-'0')) == i) {
                    if (v[j][3] != v[j][5]) otvet = false;
                }
            }
        }
    }
    if (otvet) ui->label_3->setText("Да, автоматы эквивалентны");
    else ui->label_3->setText("Нет, автоматы не эквивалентны");
}


void MainWindow::on_pushButton_2_clicked()
{
    ui->comboBox_7->setCurrentIndex(0);
    ui->comboBox->setCurrentIndex(0);
    ui->comboBox_10->setCurrentIndex(0);
    ui->comboBox_4->setCurrentIndex(0);
    ui->comboBox_8->setCurrentIndex(0);
    ui->comboBox_2->setCurrentIndex(0);
    ui->comboBox_11->setCurrentIndex(0);
```

```
      ui->comboBox_5->setCurrentIndex(0);
    ui->comboBox_9->setCurrentIndex(0);
    ui->comboBox_3->setCurrentIndex(0);
    ui->comboBox_12->setCurrentIndex(0);
    ui->comboBox_6->setCurrentIndex(0);
    ui->label_3->setText("");
}
```

Тестирование программы

Протестируем программу (рисунки 3-6).

В тестировании проверим все этапы условий, которые проходит программа.

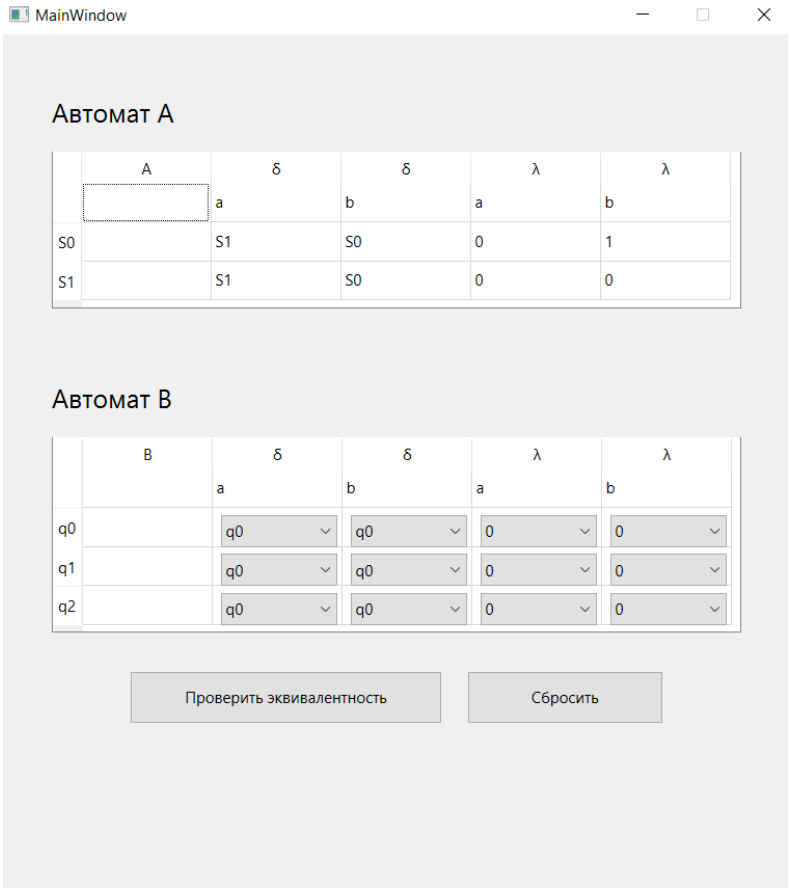

Рисунок 3 - До запуска

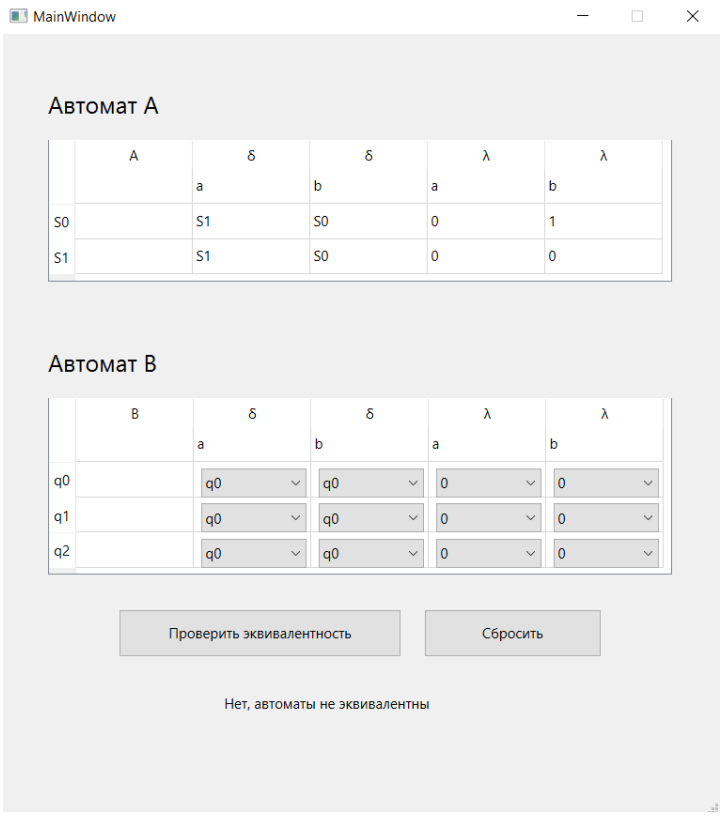

Рисунок 4 – После нажатия “Проверить эквивалентность” при вводе не эквивалентного автомата

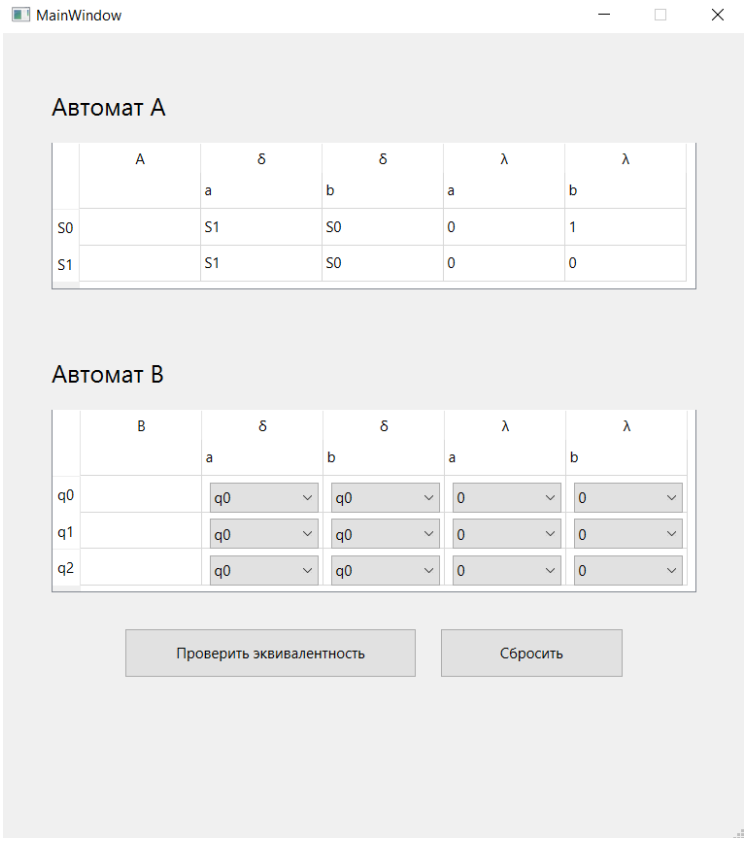

Рисунок 5 – При нажатии на кнопку “Сбросить”

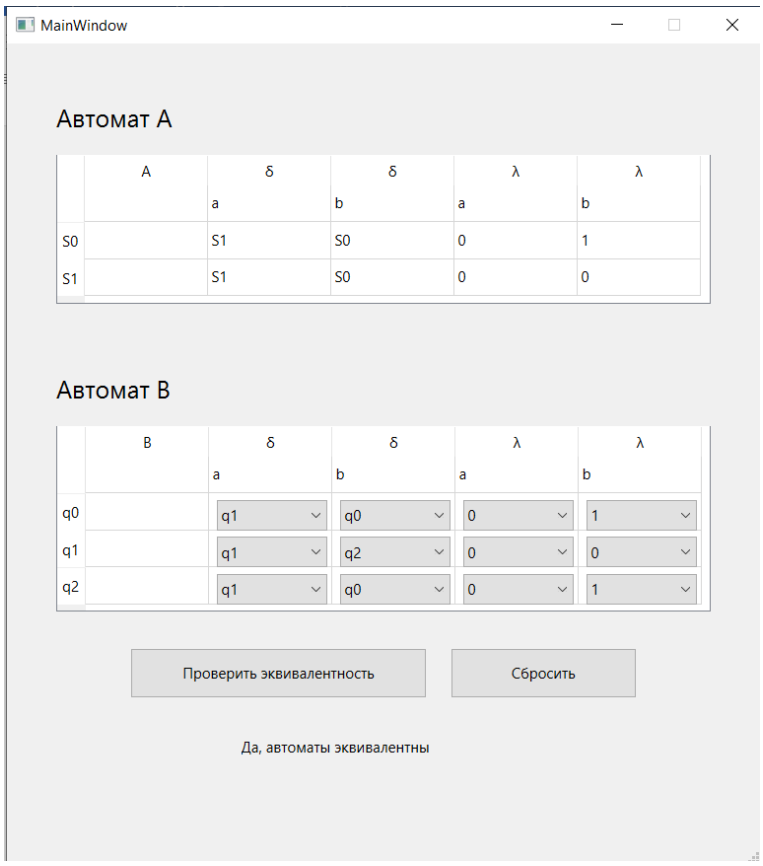

Рисунок 6 - После нажатия “Проверить эквивалентность” при вводе эквивалентного автомата

4. Заключение

В ходе выполнения домашнего задания спроектирован и реализован Был изучен программный способ реализации конечных цифровых автоматов. Был спроектирован автомат проверки эквивалентности двух автоматов. Создана программа для реализации данного автомата на языке C++ с помощью фреймворка Qt.