МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет «Информационных систем и технологий»

Кафедра «Вычислительная техника»

Дисциплина «Теория автоматов»

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

Тема «Программная реализация микропрограммного автомата с жесткой логикой: объединенная микропрограмма операций умножения и деления – вариант 1»



Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Д.В. Бахматов/

подпись инициалы, фамилия

Курс 2 Группа ИВТВМбд-21

Направление/ специальность 09.03.01 «Информатика и вычислительная техника»

Руководитель старший преподаватель

должность, ученая степень, ученое звание

Лылова Анна Вячеславовна

фамилия, имя, отчество

Дата сдачи:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Дата защиты:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

2020 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет «Информационных систем и технологий»

Кафедра «Вычислительная техника»

Дисциплина «Теория автоматов»

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

студенту ИВТВМбд-21 Бахматов Д.В.

группа фамилия, инициалы

Тема проекта (работы) «Программная реализация микропрограммного автомата с жесткой логикой: объединенная микропрограмма операций умножения и деления – вариант 1»

Срок сдачи законченного проекта (работы) «11» июня 2020г.

Исходные данные к проекту (работе) Объединенная микропрограмма операций умножения и деления,

(базовое предприятие, характер курсового проекта (работы):

варьирование реализаций(в зависимости от способа задания функции переходов и

задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)

выходов: троичная матрица и логические выражения), тип автомата (автомат Мили), язык реализации (C++), задание кафедры ВТ

Содержание пояснительной записки (перечень подлежащих разработке вопросов):

1. Титульный лист
2. Введение
3. Расширенное техническое задание
4. Анализ предметной области курсовой работы
5. Анализ исходного описания реализуемого микропрограммного автомата с жесткой логикой
6. Проектирование микропрограммного автомата с жесткой логикой:

1 этап – Составление схемы алгоритма (выдана в качестве исходного материала для выполнения курсовой работы)

2 этап – Построение отмеченной ГСА заданного микропрограммного автомата (Мура/Мили – согласно варианту задания)

3 этап – Построение графа переходов на основе полученной на предыдущем этапе отмеченной ГСА

4 этап – Проведение канонического метода структурного синтеза:

- Построение семиколоночной таблицы

- Построение логических функций (выражений) переходов и выходов

- Перевод полученных логических выражений в одно (два) заданное по варианту представление (троичная матрица, множество единичных наборов, граф (уже построен на предыдущих этапах))

1. Разработка формата протокола входных и выходных данных для разрабатываемого интерпретатора микропрограммного автомата с жесткой логикой:
2. Разработка интерпретатора микропрограммного автомата с жесткой логикой:

* алгоритм работы для каждой из процедур интерпретатора (способа реализации микропрограммного автомата с жесткой логикой, зависящего от способа его задания: логические выражения, троичная матрица, множество единичных наборов, граф;
* структурная и логическая взаимосвязь функциональных блоков (частей, модулей) программной реализации;
* количество строк кода для каждого способа интерпретации микропрограммного автомата с жесткой логикой;
* сложность реализации каждого способа интерпретации микропрограммного автомата с жесткой логикой;
* сложность модификации каждого способа интерпретации микропрограммного автомата с жесткой логикой
* и т.д. и т.п.

1. Разработка рабочей нагрузки (последовательностей X, которые обеспечивают покрытие всех переходов автомата)
2. Тестирование интерпретатора:

* прогон рабочей нагрузки,
* формирование протоколов

1. Сравнительный анализ данных полученных протоколов
2. Заключение (подведение итогов)
3. Список литературы
4. Приложение (исходные тексты программных модулей интерпретатора)

Перечень графического материала (с точным указанием обязательных чертежей)

С.А. Майоров, Г.И. Новиков, О.Ф. Немолочнов, С.И. Баранов, П.А. Шипилов, В.И. Скорубский, Г.А. Петухов, Б.Д. Тимченок Проектирование цифровых вычислительных машин. Под. ред. С.А. Майорова. Учебное пособие для студентов вузов. М., «Высш. школа», 1972, 344 с. с илл.Параграф 2.6. «Анализ операционных схем и микропрограмм»рис. 2.32 (стр. 67)

Руководитель Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ / А.В. Лылова/

должность подпись инициалы, фамилия

 «12» марта 2020г

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **/** Д.В. Бахматов **/**

подпись инициалы, фамилия

«12» марта 2020г

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовой проект (работу)**

студента Бахматова Данила Викторовича

фамилия, имя и отчество

Факультет ИСТ группа ИВТВМбд-21 курс 2

Дисциплина «Теория автоматов»

Тема проекта (работы) «Программная реализация микропрограммного автомата с жесткой логикой: объединенная микропрограмма операций умножения и деления – вариант 1»

Отмечаются следующие моменты: актуальность темы исследования; соответствие содержания и структуры курсовой работы ее теме; степень разработанности проблемы, наиболее интересно исследованные вопросы. Оценивается степень самостоятельности и инициативы студента; умение пользоваться различными источниками информации; уровень его теоретической подготовки; умение анализировать научные материалы, делать практические выводы; знание основных концепций, научной и специальной литературы по избранной теме. Содержится оценка проекта (работы) руководителем.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель\_\_\_Старший преподаватель\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ / А.В. Лылова/

должность, учёная степень, ученое звание подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

**Содержание**

[1. Введение 5](#_Toc41559118)

[2. Расширенное техническое задание 5](#_Toc41559119)

[3. Анализ предметной области курсовой работы 5](#_Toc41559120)

[4. Анализ исходного описания реализуемого микропрограммного автомата с жесткой логикой 5](#_Toc41559121)

[5. Проектирование микропрограммного автомата с жёсткой логикой 5](#_Toc41559122)

[6. Разработка формата протокола входных и выходных данных 5](#_Toc41559123)

[7. Разработка интерпретатора микропрограммного автомата с жёсткой логикой 5](#_Toc41559124)

[8. Разработка рабочей нагрузки 5](#_Toc41559125)

[9. Тестирование интерпретатора 5](#_Toc41559126)

[10. Сравнительный анализ данных полученных протоколов 5](#_Toc41559127)

[11. Заключение 5](#_Toc41559128)

[12. Список литературы 5](#_Toc41559129)

[13. Приложение. Листинг программы 5](#_Toc41559130)

* + - 1. **Введение**

Целью дисциплины «Теория автоматов» является изучение базовых понятий и принципов построения формальных грамматик и различных моделей автоматов. Любое цифровое устройство можно рассматривать как устройство, состоящее из двух частей: операционного и управляющего блоков. Операционный блок, например АЛУ, характеризуется совокупностью определённых в нём микроопераций, каждая из которых представляет собой некоторый выполняемый в данном операционном блоке акт передачи или преобразования информации. Часть цифрового вычислительного устройства, предназначенного для выработки последовательности управляющих функциональных сигналов, называется управляющим блоком или управляющим устройством.

Проектирование и реализация микропрограммных автоматов направлена на модернизацию информационных систем. Для повышения эффективности ЭВМ необходимо использование программных реализаций наиболее производительных и простых, создание которых невозможно без использования методов проектирования.

Целью работы является изучение методов проектирования и программной реализации микропрограммного автомата с жесткой логикой на основе исходных данных – объединенная микропрограмма операций умножения и деления.

* + - 1. **Расширенное техническое задание**

Требуется проанализировать исходные данные, на основе которых необходимо создать программную реализацию микропрограммного автомата с жесткой логикой (объединенная микропрограмма операций умножения и деления) на языке программирования высокого уровня C++.

На этапе проектирования необходимо провести анализ схемы алгоритма, на основе которого строится отмеченная ГСА по типу автомата Мили. Далее требуется построить граф переходов на основе полученной на предыдущем шаге отмеченной ГСА. На следующем этапе требуется провести канонический метод структурного синтеза, заключающийся в кодировании входных и выходных сигналов, а также состояний, построении семиколоночной таблицы и логических функций переходов и выходов на её основе.

• Состояния автомата должны иметь имена в формате Si, где i = 0..n (S0 – начальное и конечное состояния автомата);

• Условия в автомате (условные вершины в графе, входные сигналы автомата) кодируются как Xj, где j = 0..m;

• Действия в автомате (операторные вершины в графе, выходные сигналы автомата) кодируются как Yl, где l = 0..k;

• Функция возбуждения (функции переходов) должны иметь имена формата wz (переход с 0 на 1) и uz (переход с 1 на 0), где z = 0..v.

• процедура интерпретатора должна иметь имя следующего формата:

v№name\_avtv№var, где

v№ - порядковый номер из таблицы выдачи варианта,

name\_avt – имя типа автомата: Mili

№var – номер варианта в теме курсового проекта: 1

В данной работе процедура интерпретатора будет иметь имя: v9Miliv1

Реализованный интерпретатор микропрограммного автомата с жесткой логикой должен иметь строго заданный формат протокола входных и выходных данных. Программа должна выдавать ответ на любые входные данные: результат, если они корректны, сообщение об ошибке, если некорректны. Программная реализация должна иметь методы для двух способов задания: «Троичная матрица» и «Логическое выражение». После вывода результата программа должна выводить информацию о времени, затраченном на выполнение в зависимости от выбора способа задания.

* + - 1. **Анализ предметной области курсовой работы**

Управляющие автоматы с «жесткой» логикой представляют собой логические схемы, вырабатывающие распределенные во времени управляющие функциональные сигналы. В отличие от управляющих устройств с хранимой в памяти логикой у этих автоматов можно изменить логику работы только путем переделок схем автомата. Построение управляющих автоматов с «жесткой» логикой формализуется на основе интерпретации микропрограмм автоматами [1].

Работу операционного блока можно описать микропрограммой, например, на языке микроопераций или в виде графа. По микропрограмме строится соответствующий управляющий автомат типа Мура или Мили. Причем для выходных данных в автомате Мили необходимы входные данные, в то время как в автомате Мура выходные данные есть всегда.

При построении графов для автоматов Мура и Мили их внутренние состояния представляются вершинами графа. Внутренние переходы от одного состояния к другому изображаются направленными дугами. Для автоматов Мили и Мура значение входного сигнала, вызывающего этот переход из текущего состояния s(t) в последующее s(t+1), приписывается соответствующей дуге. Для автомата Мура значения выходных сигналов зависят только от внутреннего состояния и поэтому приписываются соответствующей вершине. Таким образом, на графах отображаются обе характеристические функции конечного автомата [2].

Последовательность действий автомата по формированию выходных сигналов и сигналов управления триггерами с учетом входных сигналов может быть задана с помощью алгоритма. Алгоритм фактически является формализованным представлением задачи по построению цифрового устройства, где определены группы выходных сигналов для инициализации устройств схемы (например, операционного устройства процессора в зависимости от поступления тех или иных входных сигналов ). Задавать цифровой автомат удобно с помощью графа. Графом называется непустое конечное множество узлов (вершин) вместе с множеством дуг (ветвей), соединяющих пары различных узлов. Граф обычно представляется в наглядной форме, при этом вершины изображаются точками или кругами, которые помечаются с целью идентификации, а ветви изображаются линиями, соединяющими соответствующие узлы.

Каждой операторной вершине ГСА автомата Мура соответствует определенное состояние – Si . Начальное состояние автомата соответствует началу ГСА, а точнее – входу в первую вершину ГСА. Первая вершина ГСА обычно соответствует логическому условию «Пуск», поэтому начальное состояние можно отметить на входе этой вершины. Поскольку этому состоянию не соответствует никакая операторная вершина, то и автомат в начальном состоянии не выдает никаких микрокоманд.

Для корректной работы автомата необходимо, чтобы после завершения выполнения алгоритма автомат вернулся в начальное состояние. Таким образом можно совместить начальное и конечное состояния автомата и обозначить их одинаково a0.

Часто дополнительно введенную операторную вершину, соответствующую микрокоманде «Операция выполнена», отмечают как конечное состояние автомата (S0), а так как конечное состояние автомата должно совпадать с начальным, в ГСА вводится еще одна дополнительная операторная вершина, которая отмечается состоянием S0, так же, как конечная. Такое преобразование ГСА имеет определенные преимущества: если автомат «свободен», он не «молчит», а выдает микрокоманду «Операция выполнена», что говорит о его готовности к работе.

В построении графа переходов его вершины соответствуют состояниям автомата, дуги – переходам из состояния Sm в состояние Sk. У вершин графа записываются микрокоманды, соответствующие состояниям, в начале дуги – логические условия, определяющие переход из состояния Sm в состояние Sk.

Граф-схема алгоритма (ГСА) есть конечный связный ориентированный граф G, удовлетворяющий следующим условиям:

• В графе G имеется два отмеченных узла – входной, из которого выходит не более одной стрелки, и выходной, не имеющий ни одной выходящей стрелки.

• Из каждого узла, отличного от входного и выходного, исходит либо одна стрелка (γ-узел), либо две стрелки (β-узел). Стрелки β-узла помечены – одна плюсом, другая минусом или одна 0, другая 1.

• Имеется два конечных множества функциональных элементов – множество преобразователей информации Q = {A1, A2,..., An } и множество распознавателей P = {p1 , p2 ,..., pm }.

• Каждому γ-узлу однозначно сопоставлен преобразователь AiÎQ , а каждому β-узлу – распознаватель ai ( p1, p2 ,...)Î P. В зависимости от алгоритма преобразователи и распознаватели могут повторяться в графе G.

Цифровые автоматы – это логические устройства, в которых помимо логических элементов имеются элементы памяти. Значение выходных сигналов такого устройства зависит не только от аргументов на входе в данный момент времени, но и от предыдущего состояния автомата, которое фиксируется элементами памяти. В качестве элементов памяти могут использоваться триггеры. Каждое внутреннее состояние цифрового автомата определяется исходным состоянием триггеров и последовательностью входных сигналов, действующих на входе в данный момент времени, поэтому такие устройства называются последовательностными схемами. К последовательностным схемам можно отнести – триггеры, счетчики, регистры. В общем случае структурная схема цифрового автомата может быть представлена в виде набора трех узлов – комбинационной схемы формирования выходных сигналов, комбинационной схемы формирования сигналов управления триггерами и, собственно, памяти.

Учитывая всё выше изложенное, можно выполнить синтез цифрового автомата следующим образом.

1. Кодирование входных сигналов в виде набора логических переменных.
2. Кодирование выходных сигналов в виде набора логических функций.
3. Кодирование состояний цифрового автомата.
4. Формирование кодированной таблицы переходов.
5. Выбор типа запоминающего элемента.
6. Составление логических выражений для логических функций, использованных для кодировки выходных сигналов.
7. Составление логических выражений для сигналов управления памятью.
8. Синтез логических схем для сформированных логических выражений.
9. Формирование выходных сигналов цифрового автомата на основании кодирующих их функций.

Для проектирования устройств с памятью применяется канонический метод структурного синтеза. Канонический метод структурного синтеза оперирует с элементарными автоматами, которые делят на два больших класса. Первый класс составляют элементарные автоматы с памятью (т. е. автоматы, имеющие более одного внутреннего состояния); такие автоматы называются элементами памяти или запоминающими элементами. Второй класс составляют автоматы без памяти (т. е. автоматы с одним внутренним состоянием), которые принято называть комбинационными, или логическими элементами.

Канонический метод структурного синтеза выполняется в четыре этапа:

1) Кодирование входных и выходных сигналов и состояний

2) Выбор элементов памяти

3) Выбор системы элементов

4) Построение функций переходов и выходов

Любую логическую функцию f (х1, х2 ..., хn) можно задать таблицей истинности, в левой части которой выписаны все возможные наборы значений ее аргументов х1, х2 ..., хn, а правая часть представляет собой столбец значений функций, соответствующих этим наборам. Набор значений переменных, на котором функция принимает значение f = 1, называется единичным набором функции f; множество всех единичных наборов - единичным множеством функции f. Аналогично набор значений, на котором f = 0, называется нулевым набором функции f, а множество нулевых наборов - нулевым множеством [3].

Число всех возможных различающихся наборов значений n переменных логической функции f (х1, х2 ..., хn) равно 2n (равно числу всех возможных двоичных векторов длины n). Число всех различных функций n переменных равно числу возможных расстановок нулей и единиц в столбце с 2n строками, т. е. |Р2 (n)| = 22n.

Троичная матрица представляет из себя структуру, в которой значения, нужные для вычисления заменяются на 0 или 1, а те, что не влияют на результат, заменяются прочерками.

C++ — компилируемый, статически типизированный язык программирования общего назначения.Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, наплатформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.[4]

* + - 1. **Анализ исходного описания реализуемого микропрограммного автомата с жесткой логикой**

Для программной реализации дана уже готовая блок-схема микропрограммного автомата согласно данному варианту [5] (объединенная микропрограмма операций умножения и деления). Ниже представлен рисунок данной блок-схемы (см. рисунок 4.1).

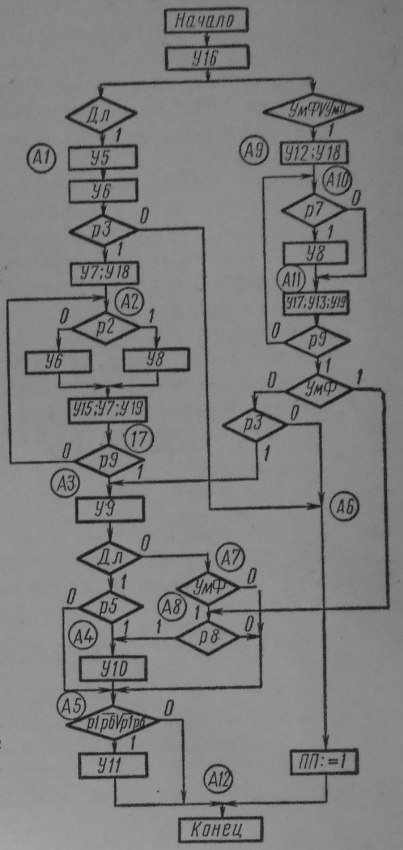


Рис. 4.1. Объединенная микропрограмма операций умножения и деления.

Будем называть микропрограммным автоматом конечный автомат Мили, синтезированный по графу алгоритма. Построение графа автомата Мили производиться в два этапа:

* + - * 1. Получение отмеченного графа алгоритма
        2. Построение графа автомата

Обычно при проектировании цифровых устройств предварительно составляется содержательный граф алгоритма, в котором внутри условных и операторных вершин записаны не элементы множества X и Y, а логические условия и микрооперации в содержательных терминах. После построения содержательного графа алгоритма логические условия и микрооперации кодируются символами x\_i, …, x\_1,…, x\_L и y\_i,…, y\_N соответственно. Кроме того, различным микрокомандам присваивается различные символы Y\_1,…, Y\_i.

Пример закодированного графа алгоритма изображен на рисунке 4.1.

Условимся в дальнейшем операторные вершины графа алгоритма обозначить символами микрокоманд, а условные вершины – символами логических условий. Если в проектируемом устройстве в каждый момент времени выполняется только одна операция из заданного множества операций, то с целью экономии оборудования выполнение всего множества операций возлагается на одну схему. Схема операционной части формируется путем объединения операционных схем, разработанных для отдельных операций.

При записи микроопераций, принадлежащих объединенному набору, разряды устройств нумеруются согласно скелету операционной части. Аналогично определяется объединенный набор информационных сигналов. В соответствии с правилами языка операционных схем по наборам микроопераций и информационных сигналов определяются шины, связывающие между собой операционные устройства.

* + - 1. **Проектирование микропрограммного автомата с жёсткой логикой**

В исходном описании автомата количество использующихся значений входных (условий) и выходных состояний не соответствует действительному. В схеме пропущены некоторые входные и выходные значения. Поэтому было принято решение о перекодировании значений X-ов и Y-ов (см. таблицу 5.1).

Далее необходимо построить отмеченную граф-схему алгоритма (см. рисунок 5.1) на основе схемы, которая изображена на рисунке 4.1.

В автомате Мили начало и конец помечаются состоянием S0, вершины, стоящие после операторных помечаются символом Sn (n=2,3..).

Таблица 5.1. Перекодирование х и y

| Исходный y | Новый y | Исходный x | Новый x |
| --- | --- | --- | --- |
| y16 | y1 | Дл | x1 |
| y5 | y2 | УмФ | x2 |
| y6 | y3 | УмЦ | x3 |
| y7 | y4 | p3 | x4 |
| y18 | y5 | p2 | x5 |
| y8 | y6 | p9 | x6 |
| y15 | y7 | p5 | x7 |
| y19 | y8 | p8 | x8 |
| y9 | y9 | p1!р6 v !p1p6 | x9 |
| y10 | y10 | p7 | x10 |
| y11 | y11 |  |  |
| y12 | y12 |  |  |
| y17 | y13 |  |  |
| y13 | y14 |  |  |
| ПП:=1 | y15 |  |  |

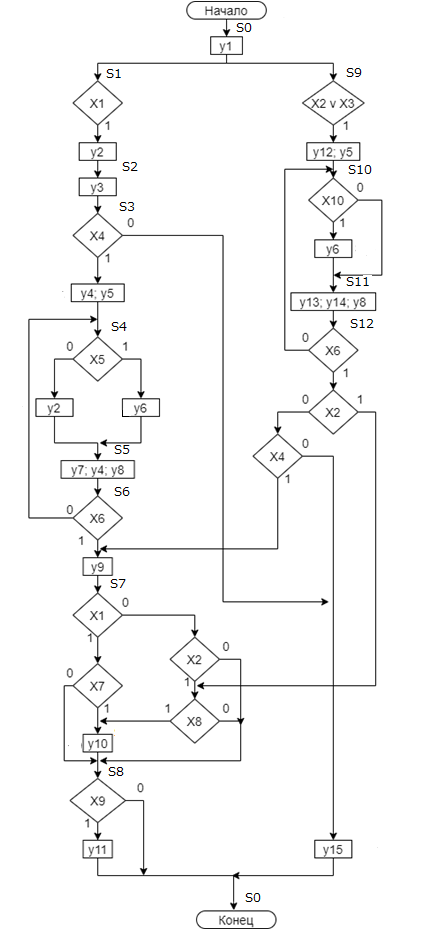
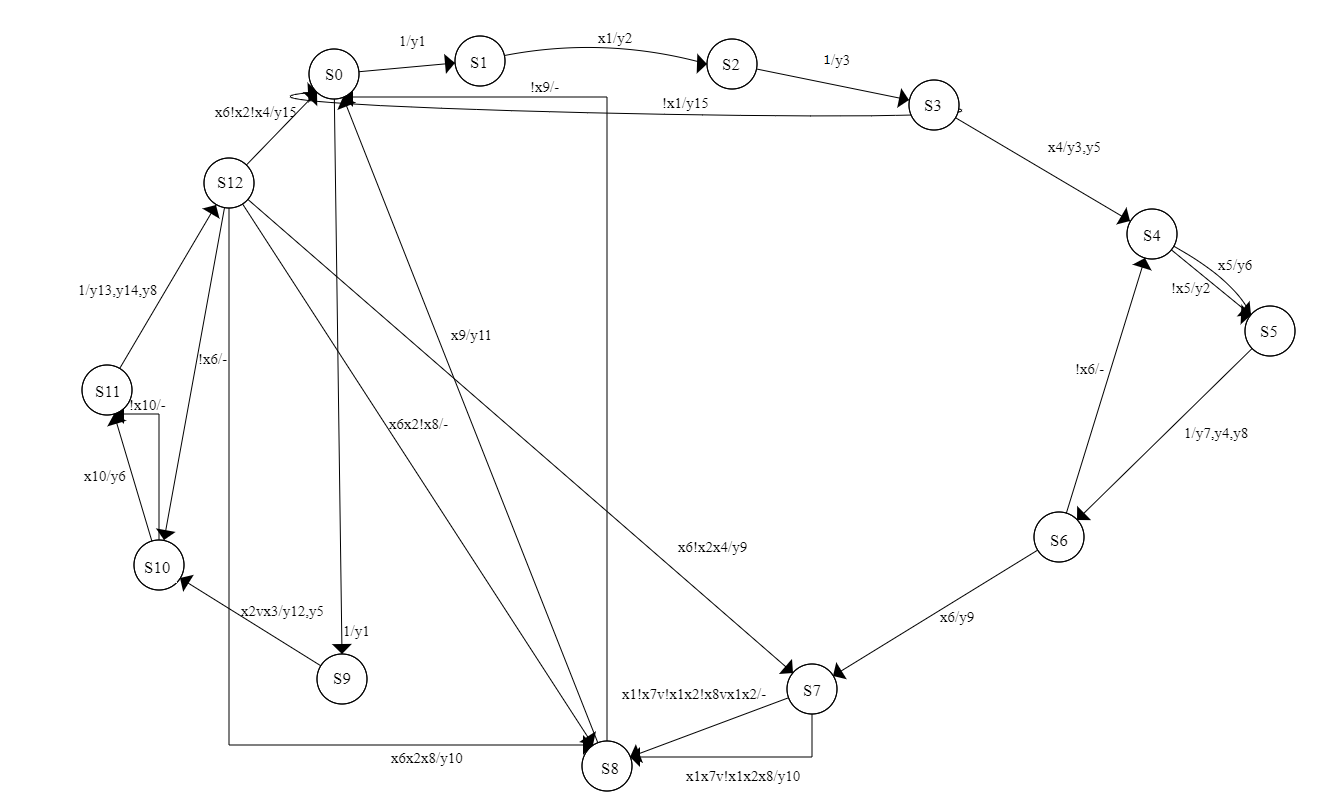


Рис. 5.1. Отмеченная граф-схема автомата.

После построения отмеченной ГСА, на ее основе строится граф переходов (см. рисунок 5.2).

Рис. 5.2. Граф переходов.

Далее необходимо произвести кодирование входных и выходных сигналов, а также состояний (S). Так как после построение схемы получилось 15 состояний (S0…S14), то для кодирования потребуется 4 бита (а3 а2 а1 а0), это вытекает из формулы:

Для наглядности кодирования состояний строится таблица 5.2.

Таблица 5.2. Кодирование состояний

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **a3** | **a2** | **a1** | **a0** |
| S0 | 0 | 0 | 0 | 0 |
| S1 | 0 | 0 | 0 | 1 |
| S2 | 0 | 0 | 1 | 0 |
| S3 | 0 | 0 | 1 | 1 |
| S4 | 0 | 1 | 0 | 0 |
| S5 | 0 | 1 | 0 | 1 |
| S6 | 0 | 1 | 1 | 0 |
| S7 | 0 | 1 | 1 | 1 |
| S8 | 1 | 0 | 0 | 0 |
| S9 | 1 | 0 | 0 | 1 |
| S10 | 1 | 0 | 1 | 0 |
| S11 | 1 | 0 | 1 | 1 |
| S12 | 1 | 1 | 0 | 0 |

На следующем этапе строится семиколоночная таблица 5.3, на основе графа переходов, изображенного на рисунке 5.2, в которой указываются все возможные переходы между состояниями, а также выходные сигналы при переходе и функции возбуждения wz – переход из 0 в 1, uz – переход из 1 в 0.

Таблица 5.3. Семиколоночная таблица

| **Нач. сост.** | **Код нач. состояния a3a2a1a0** | **Кон. сост.** | **Код кон. состояния a3a2a1a0** | **Входные сигналы** | **Вых.**  **сигналы** | **Функц.**  **возбужд.** |
| --- | --- | --- | --- | --- | --- | --- |
| S0 | 0000 | S1 | 0001 | x1 | y1 | w0 |
| S0 | 0000 | S9 | 1001 | x2vx3 | y1 | w0w3 |
| S1 | 0001 | S2 | 0010 | x1 | y2 | u0w1 |
| S2 | 0010 | S3 | 0011 | 1 | y3 | w0 |
| S3 | 0011 | S4 | 0100 | x4 | y4,y5 | u0u1w2 |
| S3 | 0011 | S0 | 0000 | !x4 | y15 | u0u1 |
| S4 | 0100 | S5 | 0101 | x5 | y6 | w0 |
| S4 | 0100 | S5 | 0101 | !x5 | y2 | w0 |
| S5 | 0101 | S6 | 0110 | 1 | y7y4y8 | u0w1 |
| S6 | 0110 | S4 | 0100 | !x6 | - | u1 |
| S6 | 0110 | S7 | 0111 | x6 | y9 | w0 |
| S7 | 0111 | S8 | 1000 | x1x7v!x1x2x8 | y10 | u0u1u2w3 |
| S7 | 0111 | S8 | 1000 | x1!x7v!x1x2!x8v!x1!x2 | - | u0u1u2w3 |
| S8 | 1000 | S0 | 0000 | !x9 | - | u3 |
| S8 | 1000 | S0 | 0000 | x9 | y11 | u3 |
| S9 | 1001 | S10 | 1010 | x2vx3 | y12y5 | u0w1 |
| S10 | 1010 | S11 | 1011 | x10 | y6 | w0 |
| S10 | 1010 | S11 | 1011 | !x10 | - | w0 |
| S11 | 1011 | S12 | 1100 | 1 | y13y14y8 | u0u1w2 |
| S12 | 1100 | S10 | 1010 | !x6 | - | u2w1 |
| S12 | 1100 | S7 | 0111 | x6!x2x4 | y9 | u3w0w1 |
| S12 | 1100 | S8 | 1000 | x6x2x8 | y10 | u2 |
| S12 | 1100 | S8 | 1000 | x6x2!x8 | - | u2 |
| S12 | 1100 | S8 | 1000 | x6x10!x8 | - | u2 |
| S12 | 1100 | S0 | 0000 | x6!x2!x4 | y15 | u2u3 |

Для построения функций выходов необходимо учитывать тип автомата. При построении функций выходов у автомата Мили учитывается код начального состояния и входные сигналы (в отличии от автомата Мура, где учитываются только коды начального состояния).

y1 = !a3!a2!a1!a0x1 v !a3!a2!a1!a0x2 v !a3!a2!a1!a0x3

y2 = !a3!a2!a1a0x1 v !a3a2!a1!a0!x5

y3 = !a3!a2a1!a0

y4 = !a3a2!a1a0 v !a3!a2a1a0x4

y5 = !a3!a2a1a0x4 v a3!a2!a1a0x2 v a3!a2!a1a0x3

y6 = !a3a2!a1!a0x5 v a3!a2a1!a0x10

y7 = !a3a2!a1a0

y8 = !a3a2!a1a0 v a3!a2a1a0

y9 = !a3a2a1!a0x6 v a3a2!a1!a0x6!x2x4

y10 = !a3a2a1a0x1x7 v !a3a2a1a0!x1x2x8 v a3a2!a1!a0x6x2x8

y11 = a3!a2!a1!a0x9

y12 = a3!a2!a1a0x2 v a3!a2!a1a0x3

y13 = a3!a2a1a0

y14 = a3!a2a1a0

y15 = !a3!a2a1a0!x4 v a3a2!a1!a0x6!x2!x4

Функции возбуждения зависят от начального состояния и входного сигнала всегда, независимо от типа автомата.

w0 = !a3!a2!a1!a0x1 v !a3a2!a1!a0x5 v a3!a2a1!a0x10 v a3!a2a1!a0!x10 v a3a2!a1!a0x6!x2x4 v !a3!a2!a1!a0x2 v !a3!a2!a1!a0x3 v !a3!a2a1!a0 v !a3a2!a1!a0!x5 v !a3a2a1!a0x6

w1 = !a3a2!a1a0 v a3!a2!a1a0x2 v a3!a2!a1a0x3 v a3a2!a1!a0!x6 v a3a2!a1!a0x6!x2x4

w2 = !a3!a2a1a0x4 v a3!a2a1a0

w3 = !a3!a2!a1!a0x2 v !a3!a2!a1!a0x3 v !a3a2a1a0x1!x7 v !a3a2a1a0x1x2!x8 v !a3a2a1a0!x1!x2 v !a3a2a1a0x1x7 v !a3a2a1a0!x1x2x8

u0 = !a3!a2a1a0x4 v !a3!a2a1a0!x4 v !a3a2!a1a0 v !a3a2a1a0x1x7 v !a3a2a1a0!x1x2x8 v !a3a2a1a0x1!x7 v !a3a2a1a0x1x2!x8 v !a3a2a1a0!x1!x2 v a3!a2a1a0 v a3!a2!a1a0x2 v a3!a2!a1a0x3 v !a3!a2!a1a0x1

u1 = !a3!a2a1a0x4 v !a3!a2a1a0!x4 v !a3a2a1!a0!x6 v !a3a2a1a0x1x7 v !a3a2a1a0!x1x2x8 v !a3a2a1a0x1!x7 v !a3a2a1a0x1x2!x8 v !a3a2a1a0!x1!x2 v a3!a2a1a0

u2 = !a3a2a1a0x1x7 v !a3a2a1a0!x1x2x8 v !a3a2a1a0x1!x7 v !a3a2a1a0x1x2!x8 v !a3a2a1a0!x1!x2 v a3a2!a1!a0!x6 v a3a2!a1!a0x6x2x8 v a3a2!a1!a0x6x10!x8 v a3a2!a1!a0x6!x2!x4 v a3a2!a1!a0x6x2!x8

u3 = a3!a2!a1!a0!x9 v a3a2!a1!a0x6!x2x4 v a3a2!a1!a0x6!x2!x4 v a3!a2!a1!a0x9

Троичные матрицы строятся на основе минимизированных логических выражений. Каждая дизъюнкция в логическом выражении образует новую строку матрицы. Каждая логическая переменная кодируется 0 и 1 если известна, а если неизвестна, то вместо нее ставится прочерк.

Полученные троичные матрицы:

y1 = [0000---------1, 0000--------1-, 0000-------1--]

y2 = [0001---------1, 0110-----0----]

y3 = [0010----------]

y4 = [0101----------, 0011------1---]

y5 = [0011------1---, 1000--------1-, 1000-------1--]

y6 = [0100-----1----, 10101---------]

y7 = [0101----------]

y8 = [0101----------, 1011----------]

y9 = [0110----1-----, 1100----1-1-0-]

y10 = [0111---1-----1, 0111--1-----10, 0011--1-1---1-]

y11 = [1000-1--------]

y12 = [1000--------1-, 1000-------1--]

y13 = [1011----------]

y14 = [1011----------]

y15 = [0011------0---, 1100----1-0-0-]

w0 = [0000---------1, 0001---------1, 0100-----1----, 1000-1--------, 10101---------, 10100---------, 1100----1-1-0-]

w1 = [0100-----0----, 0101----------, 1000--------1-, 1000-------1--, 1100----0-----, 1100----0-1-1-]

w2 = [0011------1---, 1011----------]

w3 = [0000---------1, 0111---0-----1, 0111--0-----11, 0111--------11, 0111---1-----1, 0111--1-----10]

u0 = [0011------1---, 0011------0---, 0101----------, 0111---1-----1, 0111--1-----10, 0111---0-----1, 0111--0-----11, 0111--------11, 1011----------]

u1 = [0001---------1, 0011------1---, 0011------0---, 0110----0-----, 0111---1-----1, 0111--1-----10, 0111---0-----1, 0111--0-----11, 0111--------11, 1011----------]

u2 = [0111---1-----1, 0111--1-----10, 0111---0-----1, 0111--0-----11, 0111--------11, 1100----0-----, 1100--1-1---1-, 11001-0-1-----, 1100----1-0-0-]

u3 = [1000-0--------, 1100----1-1-0-, 1100----1-0-0-]

* + - 1. **Разработка формата протокола входных и выходных данных**

После канонического метода структурного синтеза необходимо разработать протокол входных данных. На вход программы подается строка из 10 чисел, значение каждого из которых 0 или 1. Из этой строки удаляются лишние пробелы, проверяется ее длина и содержание. Полученная строка – входные сигналы x1-x10.

Если во входных данных ошибка, то выведется соответствующее ей сообщение. Если ошибок нет, то программа предложит на выбор 2 представления функций выходов и переходов: логические выражения или троичные матрицы. На вход ожидается «1» или «2».

При вводе «1», будет вызван метод, использующий троичные матрицы для прохода по автомату. При вводе «2», будет вызван метод, использующий логические выражения. Если команда введена некорректно, то будет выведено сообщение об ошибке.

При выводе предварительных результатов данные представлены в следующем формате:

• Состояние

• Выходные сигналы y1-y15

• Код состояния a3-a0

Время работы программы будет выведено после окончания вывода результатов.

* + - 1. **Разработка интерпретатора микропрограммного автомата с жёсткой логикой**

Программа интерпретатора реализована в виде класса, в котором есть поля для хранения входной строки, текущие значения выходных сигналов, кода состояния, функций возбуждения, значения троичных матриц. Так же в классе присутствует список, в котором хранится информация, сколько раз в каком состоянии был автомат, а также есть словарь, в котором хранятся коды состояний и их номера. А также у класса есть множество методов, используемых для работы автомата.

Программу интерпретатора можно разделить на три функциональных части:

* Взаимодействие с пользователем и проверка входных данных
* Подсчет функций выходов и переходов
* Переходы между состояниями и вывод информации

Блок-схема всей программы изображена на рисунке 7.1.

После ввода входных данных пользователю необходимо выбрать формат, в котором будут вычисляться выходные сигналы и функции переходов. На выбор ему предоставлен способ троичных матриц и граф-схема.

После ввода «1» или «2», вызывается метод v9Miliv1\_Matrix или v9Miliv1\_Logic, соответственно. Блок-схема метода v9Miliv1\_Matrix представлена на рисунке 7.2, а v9Miliv1\_Logic на рисунке 7.3.

В методе сначала проверяется входная строка на корректность, начинается отсчет времени работы метода и выводится начальное состояние. После находится цикл, в котором проверяется зацикленность состояния, очищаются текущие значения состояний и функций, далее производится одна итерация по переходу в новое состояние, подсчитываются новые значения состояния, выходных сигналов и функций возбуждения, выводится информация о текущем состоянии. После цикла выводится время работы метода.

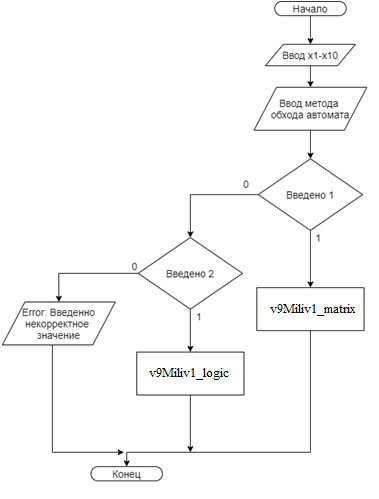


Рис. 7.1. Блок-схема программы интерпретатора.

В методе v9Miliv1\_logic вызывается метод LogicYWU, в котором производится вычисления значений y, w и u и устанавливаются новые значения выходных сигналов. Далее производится переход к новому состоянию в зависимости от подсчитанных функций возбуждений.

В методе v9Miliv1\_Matrix сначала подсчитываются функции возбуждения, для этого склеенная строка кода состояния и входной строки сравнивается с каждой строкой из троичных матриц возбуждения, если где-то строка совпала, то данная функция возбуждения имеет значение 1. Далее производится переход к новому состоянию в зависимости от подсчитанных функций возбуждений. После происходит подсчет таким же образом выходных значений в зависимости от текущего состояния.

Благодаря модульности программа легко модифицируема под новые методы и возможности, что также позволяет использовать методы из неё для реализации других автоматов.

Суммарно весь код основной программы занимает около 430 строк кода.

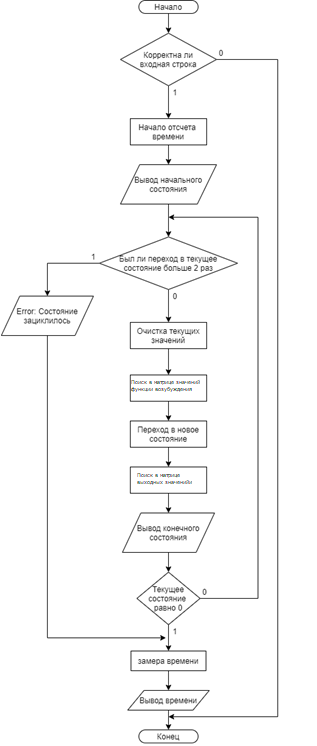
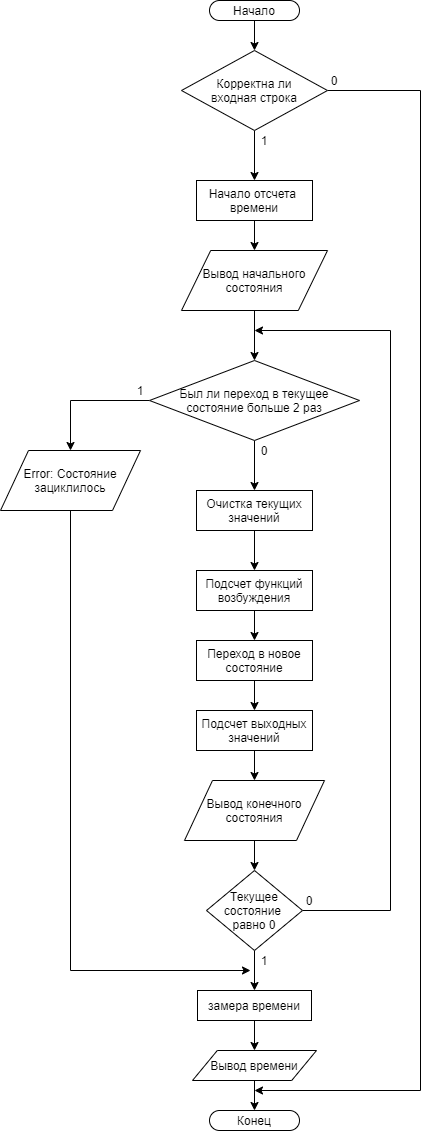


Рис. 7.2. Блок схема метода v9Miliv1\_matrix.

  
Рис. 7.3. Блок схема метода v9Murav1\_logic.

* + - 1. **Разработка рабочей нагрузки**

Для определения вариантов рабочей нагрузки аналитически подбираются входные сигналы для автомата с целью покрытия всех возможных вариантов автомата. Была проанализирована блок-схема автомата и выделен набор входных сигналов, позволяющий получить проход автомата по большинству состояний. Результаты представлены в таблице 8.1.

Таблица 8.1. Ожидаемые исходы работы автомата

|  |  |
| --- | --- |
| **Входные сигналы**  **x1x2x3x4x5x6x7x8x9x10** | **Ожидаемая работа программы** |
| 1000000000 | S0 – S1(y1) – S2(y2) – S3(y3) – S0 |
| 1001110000 | S0 – S1(y1) – S2(y2) – S3(y3) – S4(y4y5) – S6(y6) – S7(y7y4y8) – S8 – S0 |
| 1001000000 | S0 – S1(y1) – S2(y2) – S3(y3) – S4(y4y5) – S5(y2) - S6(y7y4y8) – S4 Состояние зациклилось |
| 1001100000 | S0 – S1(y1) – S2(y2) – S3(y3) – S4(y4y5) – S5(y2) - S6(y7y4y8) – S4 Состояни езациклилось |
| 1001111000 | S0 – S1(y1) – S2(y2) – S3(y3) – S4(y4y5) – S5(y6) - S6(y7y4y8) – S7(y9) – S8(y10) – S0 |
| 1001111010 | S0 – S1(y1) – S2(y2) – S3(y3) – S4(y4y5) – S5(y6) - S6(y7y4y8) – S7(y9) – S8(y10) – S0(y11) |
| 0100010010 | S0 – S9(y1) – S10(y12y5) – S11(y13y14y8) – S12(y11) – S8 – S0(11) |
| 0010000000 | S0 – S9(y1) – S10(y12y5) – S11 – S12(y13y14y8) – S10 Состояние зациклилось |
| 0010010001 | S0 – S9(y1) – S10(y12y5) – S11(y6) – S12(y13y14y8) – S0(y15) |
| 0100010110 | S0 – S9(y1) – S10(y12y5) – S11 – S12(y13y14y8) – S0(y15) |
| 0011011000 | S0 – S9(y1) – S10(y12y5) – S11 – S12(y13y14y8) – S8 – S0 |
| 0100010001 | S0 – S9(y1) – S10(y12y5) – S11(y6) – S12(y13y14y8) – S7 – S8 – S0 |
| 0100210010 | Error: Во входной строке присутствуют запрещенные символы |

* + - 1. **Тестирование интерпретатора**

Интерпретатор был протестирован на всех примерах из таблицы 8.1.На рисунках 9.1 – 9.2 изображены примеры вывода программы интерпретатора. Результаты были сведены в таблицу 9.1.

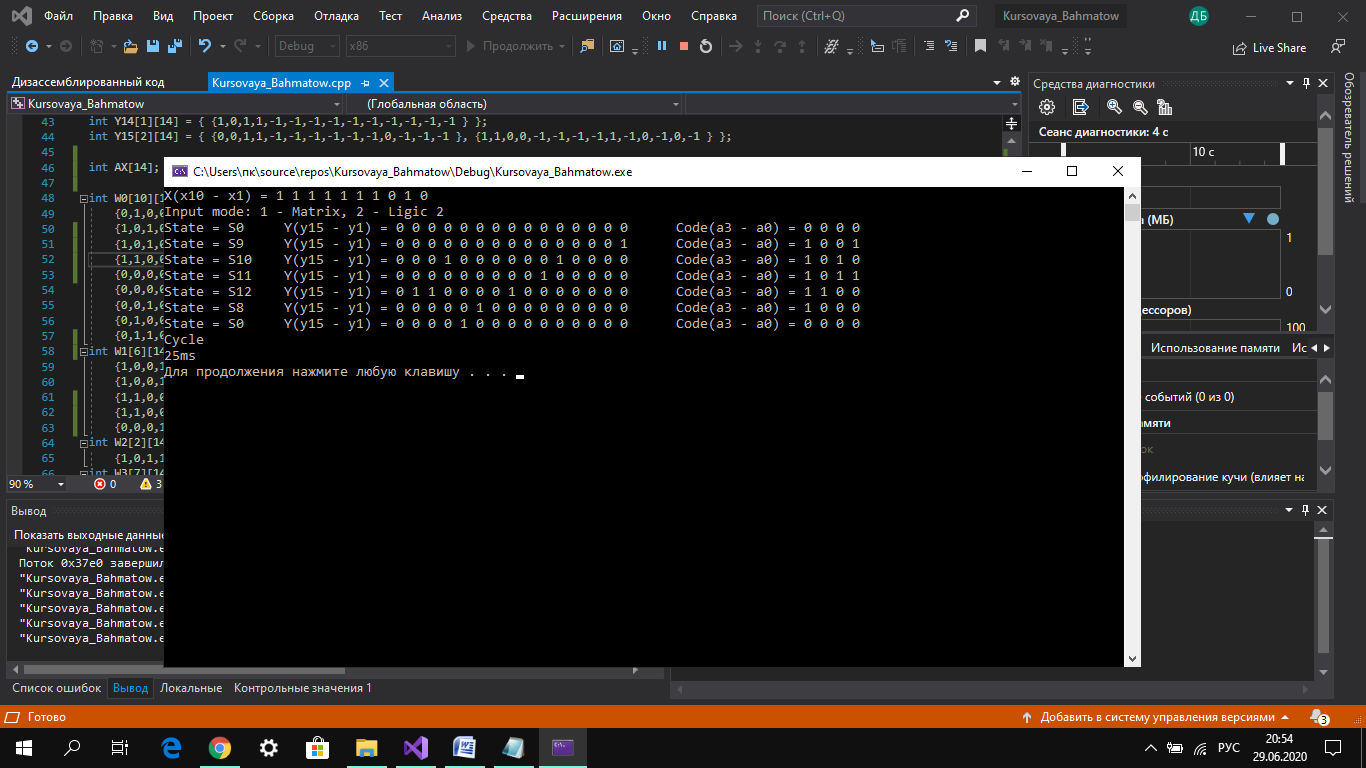


Рис. 9.1. Вывод программы интерпретатора при способе логических выражений.

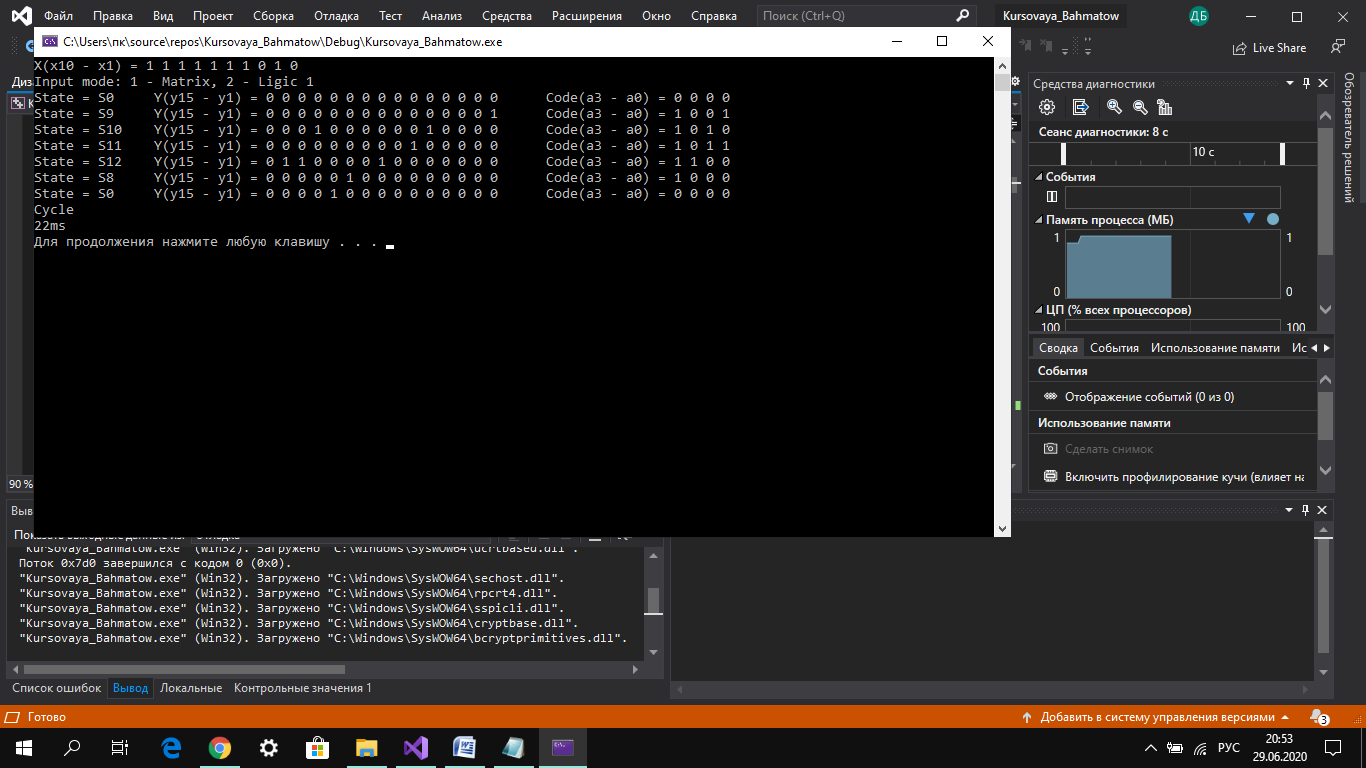


Рис. 9.2. Вывод программы интерпретатора при способе троичных матриц.

Таблица 9.1. Результаты работы программы

|  |  |  |  |
| --- | --- | --- | --- |
| **Входные сигналы**  **x10x9x8x7x6x5x4x3x2x1** | **Результат работы с использованием представления функций троичными матрицами**  **y: y15y14y13y12y11y10y9y8y7y6y5y4y3y2y1** | | **Результат работы с использованием представления функций логическими выражениями**  **y: y15y14y13y12y11y10y9y8y7y6y5y4y3y2y1** |
| 0000000001 | X(x10 - x1) = 0 0 0 0 0 0 0 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S0 Y(y15 - y1) = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  37ms | X(x10 - x1) = 0 0 0 0 0 0 0 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S0 Y(y15 - y1) = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  13ms | |
| 0000001001 | Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 0 0  Cycle  24ms | X(x10 - x1) = 0 0 0 0 0 0 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 0 0  Cycle  31ms | |
| 0000101001 | X(x10 - x1) = 0 0 0 0 1 0 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  27ms | X(x10 - x1) = 0 0 0 0 1 0 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  37ms | |
| 0000011001 | X(x10 - x1) = 0 0 0 0 0 1 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 0 0  Cycle  23ms | X(x10 - x1) = 0 0 0 0 0 1 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 0 0  Cycle  24ms | |
| 0001111001 | X(x10 - x1) = 0 0 0 1 1 1 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  33ms | X(x10 - x1) = 0 0 0 1 1 1 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  48ms | |
| 0101111001 | X(x10 - x1) = 0 1 0 1 1 1 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  40ms | X(x10 - x1) = 0 1 0 1 1 1 1 0 0 1  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S1 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 0 0 0 1  State = S2 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 Code(a3 - a0) = 0 0 1 0  State = S3 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 Code(a3 - a0) = 0 0 1 1  State = S4 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 Code(a3 - a0) = 0 1 0 0  State = S5 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 0 1 0 1  State = S6 Y(y15 - y1) = 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 Code(a3 - a0) = 0 1 1 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  33ms | |
| 0100100010 | X(x10 - x1) = 0 1 0 0 1 0 0 0 1 0  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  20ms | X(x10 - x1) = 0 1 0 0 1 0 0 0 1 0  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  32ms | |
| 0000000100 | X(x10 - x1) = 0 0 0 0 0 0 0 1 0 0  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S10 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 0  Cycle  17ms | X(x10 - x1) = 0 0 0 0 0 0 0 1 0 0  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S10 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 0  Cycle  23ms | |
| 1000100100 | X(x10 - x1) = 1 0 0 0 1 0 0 1 0 0  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S0 Y(y15 - y1) = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  23ms | X(x10 - x1) = 1 0 0 0 1 0 0 1 0 0  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S0 Y(y15 - y1) = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  20ms | |
| 0110100010 | X(x10 - x1) = 0 1 1 0 1 0 0 0 1 0  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S8 Y(y15 - y1) = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  25ms | X(x10 - x1) = 0 1 1 0 1 0 0 0 1 0  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S8 Y(y15 - y1) = 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  25ms | |
| 0000101100 | X(x10 - x1) = 0 0 0 0 1 0 1 1 0 0  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  20ms | X(x10 - x1) = 0 0 0 0 1 0 1 1 0 0  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S7 Y(y15 - y1) = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 1 1 1  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  22ms | |
| 0100010001 | X(x10 - x1) = 1 0 0 0 1 0 0 0 1 0  Input mode: 1 - Matrix, 2 - Ligic 1  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  21ms | X(x10 - x1) = 1 0 0 0 1 0 0 0 1 0  Input mode: 1 - Matrix, 2 - Ligic 2  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  State = S9 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 Code(a3 - a0) = 1 0 0 1  State = S10 Y(y15 - y1) = 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 Code(a3 - a0) = 1 0 1 0  State = S11 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Code(a3 - a0) = 1 0 1 1  State = S12 Y(y15 - y1) = 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 Code(a3 - a0) = 1 1 0 0  State = S8 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 1 0 0 0  State = S0 Y(y15 - y1) = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Code(a3 - a0) = 0 0 0 0  Cycle  37ms | |
| 0200010001 | Incorrect data | Incorrect data | |

* + - 1. **Сравнительный анализ данных полученных протоколов**

Был произведен анализ алгоритмов и данных, полученных в результате их работы. Результаты сравнения сведены в таблицу 10.1

Таблица 10.1. Сравнение методов

|  |  |  |
| --- | --- | --- |
| **Характеристики** | **Метод логических выражений** | **Метод троичных матриц** |
| Сложность реализации | Метод реализуется довольно сложно. Вся сложность заключается в вводе длинных цепочек | Метод реализуется немного проще. Вся сложность заключается в вводе длинных цепочек. |
| Сложность модификации | Благодаря модульности легко модифицируется | Благодаря модульности легко модифицируется |
| Длительность написания кода | 4 часа | 3 часа |
| Количество строк кода | 150 | 250 |
| Среднее время работы | 35.23 мс | 30.38 мс |
| Максимальное время работы | 48 мс | 40 мс |
| Минимальное время работы | 13 мс | 20 мс |

Результатом тестирования является обнаруженная незначительная разница во времени написания программы из-за ввода множества логических выражений, троичные матрицы были инициализированы немного быстрее. При этом метод троичных матриц работает быстрее из-за быстрой проверки строк.

Метод троичных матриц подходит для автоматов с большим количеством переходов и для дальнейшей модификации автомата в связи с более удобным форматом ввода логических выражений в код программы. Метод логических выражений был бы также удобен.

* + - 1. **Заключение**

В курсовой работе реализован микропрограммный автомат типа автомата Мили с жесткой логикой, а именно: объединенная микропрограмма операций умножения и деления.

В ходе выполнения были созданы:

* + 1. Отмеченная блок-схема алгоритма
    2. Граф автомата Мили
    3. Семиколоночная таблица
    4. Функции переходов и выходов вместе с их преобразованием в троичную матрицу
    5. Программная реализация микропрограммного автомата, использующего логические выражения или троичную матрицу

В процессе выполнения было улучшено владение языком С++ и умения реализации микропрограммных автоматов с жёсткой логикой. Также было закреплено умение создавать модульные и универсальные программы, не зависящие от количества входных данных.

* + - 1. **Список литературы**

1. Управляющие автоматы с «жёсткой» логикой [Электронный ресурс]. - https://studfiles.net/preview/5179828/ (дата обращения 10.05.2020)

2. Cперанский, Д.В. Моделирование, тестирование и диагностика цифровых устройств: учебное пособие / Д.В. Cперанский, Ю.А. Скобцов, В.Ю. Скобцов. - Электрон. дан. - Москва : , 2016. - 534 с. - Режим доступа: https://e.lanbook.com/book/100660.

3. Хаггарти, Род. Дискретная математика для программистов: учебное пособие для вузов / Род Хаггарти; пер. с англ. под ред. С. А. Кулешова с доп. А. А. Ковалева, В. А. Головешкина, М. В. Ульянова. - 2-e изд., доп. - Москва: Техносфера, 2005. - (Мир программирования; VIII / 02). - 399 с.: ил.

4. C Sharp[Электронный ресурс]. –

https://ru.wikipedia.org/wiki/C\_Sharp (дата обращения 10.05.2020)

5. С.А. Майоров, Г.И. Новиков, О.Ф. Немолочнов, С.И. Баранов, П.А. Шипилов, В.И. Скорубский, Г.А. Петухов, Б.Д. Тимченок Проектирование цифровых вычислительных машин. Под. ред. С.А. Майорова. Учебное пособие для студентов вузов. М., «Высш. школа», 1972, 344 с. с илл.

6. Теория автоматов [Электронный ресурс]. –

ru.wikipedia.org/wiki/Теория\_автоматов (дата обращения 12.05.2020)

7. Хопкрофт, Джон, Э., Мотвани, Раджив, Ульман, Джеффри, Д. Введение в теорию автоматов, языков и вычислений, 2-еизд.. : Пер. сангл. —М. : Издательский дом “Вильямс”, 2008. — 528 с.

8. Теория автоматов (часть I). Конспект лекций /Киров, Вятский государственный университет, 2010, 56с.

9. Князьков, В.С. Введение в теорию автоматов: учебное пособие / В.С. Князьков, Т.В. Волченская. - Электрон. дан. - Москва:, 2016. - 89 с. - Режим доступа: https://e.lanbook.com/book/100715.

10. Горбатов, Вячеслав Афанасьевич. Теория автоматов: учебник для вузов / Горбатов В. А., Горбатов А. В., Горбатова М. В. - Москва: АСТ: Астрель, 2008. - (Высшая школа). - 559 с.

11. Кузнецов, Олег Петрович. Дискретная математика для инженера: [для студ. и спец. в обл. вычисл. техники] / Кузнецов О. П. - 4-е изд., стер. - Санкт-Петербург [и др. ]: Лань, 2005. - 395 с.

12. Автомат Мура [Электронный ресурс]. –

https://ru.wikipedia.org/wiki/Автомат\_Мура (дата обращения 10.05.2020)

**Приложение. Листинг программы**

#include <iostream>

#include <fstream>

#include <chrono>

using namespace std;

void InputBool(bool Array[20], int Lenght);

void OutputBool(bool Array[20], int Lenght);

void ZeroBoolArray(bool Array[20], int Lenght);

void LogicYWU();

void ChangeA();

string OutputName();

void v9Miliv1\_logic();

void MergeAX();

bool FindInArr(int Arr[14], int Arr2[20][14], int Len1, int Len2);

void FindYWU();

void v9Mili\_matrix();

bool y[15];

bool x[10];

bool a[4];

bool w[4];

bool u[4];

string StateName[13] = { "S0 ", "S1 ", "S2 ", "S3 ", "S4 ", "S5 ", "S6 ", "S7 ", "S8 ", "S9 ", "S10", "S11", "S12" };

bool CycleCheck[13] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

bool CycleFlag = 0;

int Ylen[15] = {3, 2, 1, 2, 3, 2, 1, 2, 2, 3, 1, 2, 1, 1, 2};

int Wlen[4] = { 10,5,2,7 };

int Ulen[4] = { 12,9,10,4 };

int Y1[3][14] = { {0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,1 }, {0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 }, {0,0,0,0,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 } };

int Y2[2][14] = { {0,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1 }, {0,1,0,0,-1,-1,-1,-1,-1,0,-1,-1,-1,-1 }};

int Y3[1][14] = { {0,0,1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y4[2][14] = { {0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 }, {0,0,1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1 } };

int Y5[3][14] = { {0,0,1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1 }, {1,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 }, {1,0,0,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 } };

int Y6[2][14] = { {0,1,0,0,-1,-1,-1,-1,-1,1,-1,-1,-1,-1 }, {1,0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y7[1][14] = { {0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y8[2][14] = { {0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 }, {1,0,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y9[2][14] = { {0,1,1,0,-1,-1,-1,-1,1,-1,-1,-1,-1,-1 }, {1,0,0,1,-1,-1,-1,-1,0,-1,1,-1,1,-1 } };

int Y10[3][14] = { {0,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1 }, {0,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,1,0 }, {1,1,0,0,-1,-1,1,-1,1,-1,-1,-1,1,-1 } };

int Y11[1][14] = { {1,0,0,0,-1,1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y12[2][14] = { {1,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 }, {1,0,0,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 } };

int Y13[1][14] = { {1,0,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y14[1][14] = { {1,0,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int Y15[2][14] = { {0,0,1,1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1 }, {1,1,0,0,-1,-1,-1,-1,1,-1,0,-1,0,-1 } };

int AX[14];

int W0[10][14] = { {0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,1 },

{0,1,0,0,-1,-1,-1,-1,-1,1,-1,-1,-1,-1 },

{1,0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1 },

{1,0,1,0,0,-1,-1,-1,-1,-1,-1,-1,-1,-1 },

{1,1,0,0,-1,-1,-1,-1,1,-1,1,-1,0,-1 },

{0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 },

{0,0,0,0,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 },

{0,0,1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 },

{0,1,0,0,-1,-1,-1,-1,-1,0,-1,-1,-1,-1 },

{0,1,1,0,-1,-1,-1,-1,1,-1,-1,-1,-1,-1 } };

int W1[6][14] = { {0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 },

{1,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 },

{1,0,0,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 },

{1,1,0,0,-1,-1,-1,-1,0,-1,-1,-1,-1,-1 },

{1,1,0,0,-1,-1,-1,-1,1,-1,1,-1,0,-1 },

{0,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1 } };

int W2[2][14] = { {0,0,1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1 },

{1,0,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 } };

int W3[7][14] = { {0,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 },

{0,0,0,0,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 } ,

{0,1,1,1,-1,-1,-1,0,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,0,-1,-1,-1,-1,-1,1,1 },

{0,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,0,0 },

{0,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,1,0} };

int U0[12][14] = { {0,0,1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1},

{0,0,1,1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1},

{0,1,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1},

{0,1,1,1,-1,-1,-1,0,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,0,-1,-1,-1,-1,-1,1,1 },

{0,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,0,0 },

{0,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,1,0},

{1,0,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 },

{1,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1 },

{1,0,0,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1 },

{0,0,0,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1 } };

int U1[9][14] = { {0,1,1,0,-1,-1,-1,-1,0,-1,-1,-1,-1,-1},

{1,0,1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1 },

{0,1,1,1,-1,-1,-1,0,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,0,-1,-1,-1,-1,-1,1,1 },

{0,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,0,0 },

{0,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,1,0},

{0,0,1,1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1},

{0,0,1,1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1}, };

int U2[10][14] = { {0,1,1,1,-1,-1,-1,0,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,0,-1,-1,-1,-1,-1,1,1 },

{0,1,1,1,-1,-1,-1,-1,-1,-1,-1,-1,0,0 },

{0,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1 },

{0,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,1,0},

{1,1,0,0,-1,-1,-1,-1,0,-1,-1,-1,-1,-1 },

{1,1,0,0,-1,-1,1,-1,1,-1,-1,-1,1,-1 },

{1,1,0,0,1,-1,0,-1,1,-1,-1,-1,-1,-1 },

{1,1,0,0,-1,-1,-1,-1,1,-1,0,-1,0,-1 },

{1,1,0,0,-1,-1,0,-1,1,-1,-1,-1,1,-1 }, };

int U3[4][14] = { {1,0,0,0,-1,0,-1,-1,-1,-1,-1,-1,-1,-1 },

{1,0,0,0,-1,1,-1,-1,-1,-1,-1,-1,-1,-1 },

{1,1,0,0,-1,-1,-1,-1,1,-1,1,-1,0,-1 },

{1,1,0,0,-1,-1,-1,-1,1,-1,0,-1,0,-1 }, };

int main()

{

int n;

InputBool(x, 10);

cout << "X(x10 - x1) = ";

OutputBool(x, 10);

cout << endl;

cout << "Input mode: 1 - Matrix, 2 - Ligic ";

cin >> n;

if (x[0] && (x[1] || x[2]))

{

cout << "Incorrect data (x1 & (x2 | x3))\n";

system("pause");

exit;

}

if (n == 1) v9Mili\_matrix();

else if (n == 2) v9Miliv1\_logic();

else

{

cout << "Incorrect mode)\n";

system("pause");

exit;

}

system("pause");

}

void FindYWU()

{

y[0] = FindInArr(AX, Y1, 3, 14);

y[1] = FindInArr(AX, Y2, 2, 14);

y[2] = FindInArr(AX, Y3, 1, 14);

y[3] = FindInArr(AX, Y4, 2, 14);

y[4] = FindInArr(AX, Y5, 3, 14);

y[5] = FindInArr(AX, Y6, 2, 14);

y[6] = FindInArr(AX, Y7, 1, 14);

y[7] = FindInArr(AX, Y8, 2, 14);

y[8] = FindInArr(AX, Y9, 2, 14);

y[9] = FindInArr(AX, Y10, 3, 14);

y[10] = FindInArr(AX, Y11, 1, 14);

y[11] = FindInArr(AX, Y12, 2, 14);

y[12] = FindInArr(AX, Y13, 1, 14);

y[13] = FindInArr(AX, Y14, 1, 14);

y[14] = FindInArr(AX, Y15, 2, 14);

w[0] = FindInArr(AX, W0, 10, 14);

w[1] = FindInArr(AX, W1, 6, 14);

w[2] = FindInArr(AX, W2, 2, 14);

w[3] = FindInArr(AX, W3, 7, 14);

u[0] = FindInArr(AX, U0, 12, 14);

u[1] = FindInArr(AX, U1, 9, 14);

u[2] = FindInArr(AX, U2, 10, 14);

u[3] = FindInArr(AX, U3, 4, 14);

}

bool FindInArr(int Arr[14],int Arr2[20][14], int Len1, int Len2)

{

for (int i = 0; i < Len1; i++)

{

int k = 0;

for (int j = 0; j < Len2; j++)

{

if (Arr2[i][j] != -1)

{

if (Arr[j] == Arr2[i][j])

{

k++;

continue;

}

else

{

break;

}

}

else

{

k++;

}

}

if (k == 14) return 1;

}

return 0;

}

void v9Mili\_matrix()

{

auto begin = chrono::high\_resolution\_clock::now();

ZeroBoolArray(a, 4);

cout << "State = ";

OutputName();

cout << "Y(y15 - y1) = ";

OutputBool(y, 15);

cout << "Code(a3 - a0) = ";

OutputBool(a, 4);

cout << endl;

ZeroBoolArray(y, 15);

ZeroBoolArray(w, 4);

ZeroBoolArray(u, 4);

for (int i = 0; i < 20; i++)

{

MergeAX();

FindYWU();

ChangeA();

cout << "State = ";

OutputName();

cout << "Y(y15 - y1) = ";

OutputBool(y, 15);

cout << "Code(a3 - a0) = ";

OutputBool(a, 4);

if (CycleFlag == 1)

{

CycleFlag = 0;

cout << "\nCycle\n";

break;

}

ZeroBoolArray(y, 15);

ZeroBoolArray(w, 4);

ZeroBoolArray(u, 4);

cout << endl;

}

auto end = chrono::high\_resolution\_clock::now();

cout << chrono::duration\_cast<chrono::nanoseconds>(end - begin).count() / 1000000 << "ms" << endl;

}

void v9Miliv1\_logic()

{

auto begin = chrono::high\_resolution\_clock::now();

ZeroBoolArray(y, 15);

ZeroBoolArray(a, 4);

for (int i = 0; i < 20; i++)

{

cout << "State = ";

OutputName();

cout << "Y(y15 - y1) = ";

OutputBool(y, 15);

cout << "Code(a3 - a0) = ";

OutputBool(a, 4);

if (CycleFlag == 1)

{

CycleFlag = 0;

cout << "\nCycle\n";

break;

}

LogicYWU();

ChangeA();

cout << endl;

}

auto end = chrono::high\_resolution\_clock::now();

cout << chrono::duration\_cast<chrono::nanoseconds>(end - begin).count() / 1000000 << "ms" << endl;

}

void MergeAX()

{

int i = 0;

int k = 3;

while (k >= 0)

{

AX[i] = a[k];

k--;

i++;

}

k = 9;

while (k >= 0)

{

AX[i] = x[k];

k--;

i++;

}

}

void LogicYWU()

{

y[0] = !a[3] && !a[2] && !a[1] && !a[0] && x[0] || !a[3] && !a[2] && !a[1] && !a[0] && x[1]

|| !a[3] && !a[2] && !a[1] && !a[0] && x[2];

y[1] = !a[3] && !a[2] && !a[1] && a[0] && x[0] || !a[3] && a[2] && !a[1] && !a[0] && !x[4];

y[2] = !a[3] && !a[2] && a[1] && !a[0];

y[3] = !a[3] && a[2] && !a[1] && a[0] || !a[3] && !a[2] && a[1] && a[0] && x[3];

y[4] = !a[3] && !a[2] && a[1] && a[0] && x[3] || a[3] && !a[2] && !a[1] && a[0] && x[1]

|| a[3] && !a[2] && !a[1] && a[0] && x[2];

y[5] = !a[3] && a[2] && !a[1] && !a[0] && x[4] || a[3] && !a[2] && a[1] && !a[0] && x[9];

y[6] = !a[3] && a[2] && !a[1] && a[0];

y[7] = !a[3] && a[2] && !a[1] && a[0] || a[3] && !a[2] && a[1] && a[0];

y[8] = !a[3] && a[2] && a[1] && !a[0] && x[5] || a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && x[3];

y[9] = !a[3] && a[2] && a[1] && a[0] && x[0] && x[6] || !a[3] && a[2] && a[1] && a[0] && !x[0] && x[1] && x[7]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && x[1] && x[7];

y[10] = a[3] && !a[2] && !a[1] && !a[0] && x[8];

y[11] = a[3] && !a[2] && !a[1] && a[0] && x[1] || a[3] && !a[2] && !a[1] && a[0] && x[2];

y[12] = a[3] && !a[2] && a[1] && a[0];

y[13] = a[3] && !a[2] && a[1] && a[0];

y[14] = !a[3] && !a[2] && a[1] && a[0] && !x[3] || a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && !x[3];

w[0] = !a[3] && !a[2] && !a[1] && !a[0] && x[0]

|| !a[3] && a[2] && !a[1] && !a[0] && x[4]

|| a[3] && !a[2] && a[1] && !a[0] && x[9]

|| a[3] && !a[2] && a[1] && !a[0] && !x[9]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && x[3]

|| !a[3] && !a[2] && a[1] && !a[0]

|| !a[3] && !a[2] && !a[1] && !a[0] && x[1]

|| !a[3] && !a[2] && !a[1] && !a[0] && x[2]

|| !a[3] && a[2] && !a[1] && !a[0] && !x[4]

|| !a[3] && a[2] && a[1] && !a[0] && x[5];

w[1] = !a[3] && a[2] && !a[1] && a[0]

|| a[3] && !a[2] && !a[1] && a[0] && x[1]

|| a[3] && !a[2] && !a[1] && a[0] && x[2]

|| a[3] && a[2] && !a[1] && !a[0] && !x[5]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && x[3]

|| !a[3] && !a[2] && !a[1] && a[0] && x[0];

w[2] = !a[3] && !a[2] && a[1] && a[0] && x[3]

|| a[3] && !a[2] && a[1] && a[0];

w[3] =

!a[3] && !a[2] && !a[1] && !a[0] && (x[1] || x[2])

|| !a[3] && a[2] && a[1] && a[0] && x[0] && !x[6]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[1] && !x[7]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && !x[1]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[6]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && x[1] && x[7];

u[0] = !a[3] && !a[2] && a[1] && a[0] && x[3]

|| !a[3] && !a[2] && a[1] && a[0] && !x[3]

|| !a[3] && a[2] && !a[1] && a[0]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[6]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && x[1] && x[7]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && !x[6]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[1] && !x[7]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && !x[1]

|| a[3] && !a[2] && a[1] && a[0]

|| !a[3] && !a[2] && !a[1] && a[0] && x[0]

|| a[3] && !a[2] && !a[1] && a[0] && x[1]

|| a[3] && !a[2] && !a[1] && a[0] && x[2];

u[1] =

!a[3] && !a[2] && a[1] && a[0] && x[3]

|| !a[3] && !a[2] && a[1] && a[0] && !x[3]

|| !a[3] && a[2] && a[1] && !a[0] && !x[5]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[6]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && x[1] && x[7]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && !x[6]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[1] && !x[7]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && !x[1]

|| a[3] && !a[2] && a[1] && a[0];

u[2] = !a[3] && a[2] && a[1] && a[0] && x[0] && x[6]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && x[1] && x[7]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && !x[6]

|| !a[3] && a[2] && a[1] && a[0] && x[0] && x[1] && !x[7]

|| !a[3] && a[2] && a[1] && a[0] && !x[0] && !x[1]

|| a[3] && a[2] && !a[1] && !a[0] && !x[5]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && x[1] && x[7]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && x[9] && !x[7]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && !x[3]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && x[1] && !x[7];

u[3] = a[3] && !a[2] && !a[1] && !a[0] && !x[8]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && x[3]

|| a[3] && a[2] && !a[1] && !a[0] && x[5] && !x[1] && !x[3]

|| a[3] && !a[2] && !a[1] && !a[0] && x[8];

}

string OutputName()

{

int sum = 0;

if (a[0] == 1)

{

sum += 1;

}

if (a[1] == 1)

{

sum += 2;

}

if (a[2] == 1)

{

sum += 4;

}

if (a[3] == 1)

{

sum += 8;

}

if (CycleCheck[sum] == 1)

{

CycleFlag = 1;

}

CycleCheck[sum] = 1;

cout << StateName[sum] << " ";

return StateName[sum];

}

void ChangeA()

{

for (int i = 3; i >= 0; i--)

{

if (w[i] == 1)

{

a[i] = 1;

}

if (u[i] == 1)

{

a[i] = 0;

}

}

}

void ZeroBoolArray(bool Array[20], int Lenght)

{

for (int i = 0; i < Lenght; i++)

{

Array[i] = 0;

}

}

void InputBool(bool Array[20], int Lenght)

{

int tmp;

ifstream fin;

fin.open("C:\\TA\\Kurs.txt", ios::in);

for (int i = 0; i < Lenght; i++)

{

fin >> tmp;

if (tmp > 1 || tmp < 0)

{

cout << "Incorrect data\n";

system("pause");

exit;

}

Array[i] = tmp;

}

fin.close();

}

void OutputBool(bool Array[20], int Lenght)

{

for (int i = Lenght - 1; i >= 0; i--)

{

cout << Array[i] << " ";

}

cout << " ";

}