**ТИТУЛЬНЫЙ ЛИСТ КУРСОВОГО ПРОЕКТА**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ   
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (МАИ)

Кафедра №806 «Вычислительная математика и программирование»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

«ФУНДАМЕНТАЛЬНАЯ ИНФОРМАТИКА И АРХИТЕКТУРА КОМПЬЮТЕРОВ»

на тему:

**«Решение задач по теме Машина Тьюринга/Диаграммы Тьюринга/Нормальные алгоритмы Маркова/Конечные автоматы»**

Студент Волков А. А.

Группа М8О-101БВ-24

Руководитель Дубинин А.В.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты «20» декабря 2024 г.

Москва, 2024

Содержание

[ВВЕДЕНИЕ 3](#_Toc185259153)

[Теоретическая часть 5](#_Toc185259154)

[Машина Тьюринга 5](#_Toc185259155)

[Диаграммы Тьюринга 6](#_Toc185259156)

[Нормальные Алгоритмы Маркова 9](#_Toc185259157)

[Конечные автоматы 11](#_Toc185259158)

[Практическая часть 14](#_Toc185259159)

[1. Машина Тьюринга 14](#_Toc185259160)

[1.1. Описание программы 14](#_Toc185259161)

[1.2. Идея алгоритма: 14](#_Toc185259162)

[1.3. Сценарий выполнения. 14](#_Toc185259163)

[Тесты: 14](#_Toc185259164)

[2. Диаграмма Тьюринга 15](#_Toc185259165)

[2.1. Описание программы: 15](#_Toc185259166)

[2.2. Идея алгоритма: 15](#_Toc185259167)

[2.3. Сценарий выполнения: 15](#_Toc185259168)

[Тесты: 19](#_Toc185259169)

[3. Нормальный алгоритм Маркова 20](#_Toc185259170)

[3.1. Описание программы: 20](#_Toc185259171)

[3.2. Идея алгоритма: 20](#_Toc185259172)

[3.3. Сценарий выполнения: 20](#_Toc185259173)

[Тесты: 21](#_Toc185259174)

[4. Конечные автоматы 22](#_Toc185259175)

[4.1. Описание программы: 22](#_Toc185259176)

[4.2. Идея алгоритма: 22](#_Toc185259177)

[4.3. Сценарий выполнения: 22](#_Toc185259178)

[Тесты: 24](#_Toc185259179)

[Заключение 25](#_Toc185259180)

[Список источников 26](#_Toc185259181)

# ВВЕДЕНИЕ

Теория вычислимости и формальных языков — это одна из фундаментальных областей информатики, исследующая природу вычислений, их ограничения и возможности различных вычислительных моделей. Данная дисциплина играет ключевую роль в понимании принципов функционирования современных вычислительных систем и создании алгоритмов.

В рамках этой курсовой работы рассматриваются классические вычислительные модели, такие как машина Тьюринга, диаграммы Тьюринга, нормальные алгоритмы Маркова и конечные автоматы. Эти модели, несмотря на их абстрактный характер, являются мощным инструментом для анализа и формализации процессов вычислений.

**Актуальность** темы обусловлена несколькими ключевыми аспектами:

* Формализация алгоритмов. Вычислительные модели предоставляют строгие математические определения алгоритмов и позволяют формально доказывать их свойства, корректность и эффективность.
* Изучение границ вычислимости. Понимание возможностей и ограничений различных моделей вычислений позволяет определить классы задач, которые принципиально могут быть решены алгоритмически, а также выделить неразрешимые проблемы.
* Практическое применение. Анализ вычислительных моделей способствует разработке более эффективных алгоритмов и решению прикладных задач в различных областях науки и техники.

**Цель работы** состоит в изучении математических основ моделей вычислений, освоении методов построения и анализа алгоритмов, а также решении задач, связанных с их использованием.

**Задачи** данной работы включают:

* исследование теоретических основ и принципов работы указанных моделей;
* практическую реализацию и анализ задач, решаемых с их помощью;
* выявление ограничений и возможностей каждой из рассматриваемых моделей.

# Теоретическая часть

## Машина Тьюринга

Машина Тьюринга — это абстрактная модель вычислений, предложенная Аланом Тьюрингом в 1936 году для формализации понятия алгоритма и исследования пределов вычислимости. Основная идея заключается в использовании максимально простой конструкции, которая, тем не менее, способна моделировать любые вычисления, выполняемые современными компьютерами.

**Основные компоненты МТ:**

1. **Рабочая лента.** Лента считается бесконечной в обе стороны и разделена на ячейки, каждая из которых может содержать символ из фиксированного алфавита.
2. **Головка чтения/записи.** В каждый момент времени головка находится над одной из ячеек ленты. Она способна считывать знак, записывать новый символ и перемещаться по ленте влево или вправо.
3. **Управляющее устройство.** Оно находится в одном из конечного множества состояний и определяет действия машины в зависимости от текущего состояния и символа, расположенного под головкой.

**Ключевые понятия:**

1. **Ситуация.** Состояние машины описывается парой S=(z, k), где z — текущее содержимое ленты, а k — номер ячейки, над которой находится головка.
2. **Конфигурация.** Конфигурация C = (S, q) включает ситуацию S и текущее состояние q.

Каждая конфигурация определяет следующее действие машины, переходя в новую конфигурацию, пока не будет достигнуто завершающее состояние.

**Принцип работы:**

1. На начальном этапе на ленте записана входная строка, остальные ячейки содержат пустой символ b. Головка располагается над первой ячейкой входной строки, а управляющее устройство находится в состоянии q0.
2. Машина пошагово выполняет команды, заданные функцией переходов δ. Каждая команда определяет:
   * новое состояние q′, в которое перейдёт управляющее устройство;
   * символ a′, который заменит текущий символ под головкой;
   * направление движения головки: влево (L), вправо (R) или остаётся на месте (N).
3. Работа завершается, если управляющее устройство переходит в одно из конечных состояний. Результат вычислений остаётся на ленте.

## Диаграммы Тьюринга

**Диаграмма Тьюринга (ДТ)** представляет собой графическое отображение работы Машины Тьюринга (МТ), где с помощью графических элементов (точек и стрелок) отображаются состояния и переходы между ними. Такой подход значительно облегчает восприятие алгоритмов и операций машины, являясь наглядной альтернативой формальному табличному представлению.

**1. Элементарные Машины Тьюринга**

Каждый элементарный шаг, выполняемый Машиной Тьюринга, может быть представлен с помощью базовых символов диаграммы:

1. **l** — сдвиг головки влево.
2. **r** — сдвиг головки вправо.
3. **λ, a₁, ..., aₚ** — запись символов на ленту, где λ — это пустое место, а a1, …, ap — символы рабочего алфавита.

Дополнительно могут использоваться более сложные символы:

1. **L** — сдвиг влево до первого пустого символа (λ).
2. **R** — сдвиг вправо до первого пустого символа.
3. **K** — копирование предыдущего символа или слова.

**2. Структура Диаграмм Тьюринга**

* **Точка** — представляет состояние машины Тьюринга.
* **Стрелка** — указывает на переход между состояниями.
* **Символ МТ** — отображается как «точка — буква — точка» и обозначает переход с условием, соответствующим символу на ленте.
* **Развилка** — условие перехода от одного состояния к нескольким другим в зависимости от считанного символа.
* **Повторение** — переход в то же состояние, что используется для отображения цикличности или повторяющихся операций.

**3. Принцип построения Диаграммы Тьюринга**

Диаграмму Тьюринга строят слева направо. Каждый символ Машины Тьюринга соответствует определенному начальному и конечному состоянию, которые изображаются точками, соединенными стрелками. Если после состояния 1 должно следовать состояние 2, то стрелка от точки состояния 1 направляется к точке состояния 2. Условие перехода указывается над стрелкой, что позволяет отобразить логику работы машины.

**4. Моделирование машин Тьюринга с помощью диаграмм**

Для того чтобы диаграмма моделировала работу Машины Тьюринга, выполняются следующие шаги:

1. **Кодирование алфавитов:** Алфавит, использованный в Машине Тьюринга (А), должен быть отображен с помощью алфавита диаграммы (A₁). Это гарантирует, что все символы корректно отображаются и используются в процессе работы.
2. **Соответствие состояниям:** Каждому состоянию q Машины Тьюринга будет соответствовать одна или несколько точек на диаграмме D. Это позволяет отобразить переходы между состояниями и выполнение операций.
3. **Конфигурации и начальная/конечная ситуация:** Конфигурации, такие как начальная ситуация C₀, должны отображаться в виде начальной конфигурации на диаграмме C₀₁. Важно также гарантировать, что машина остановится в том же состоянии, что и оригинальная МТ.

**5. Эквивалентность МТ и Диаграмм Тьюринга**

* **Эквивалентность программы и диаграммы Тьюринга:** Существует теорема, утверждающая, что каждой программе P, задающей Машину Тьюринга T = (A, Q, P, q0), можно составить диаграмму D, которая будет эффективно моделировать работу машины Тьюринга. Для каждой строки программы строится соответствующий элемент диаграммы, отражающий переходы между состояниями. Завершающим элементом будет правая точка предыдущего элемента диаграммы, что завершает выполнение.
* **Эквивалентность диаграммы и программы:** Теорема утверждает, что каждой диаграмме Тьюринга D можно сопоставить программу P, так, чтобы программа моделировала МТ, заданную диаграммой. Для этого на диаграмме заменяются все неэлементарные операции на элементарные, и каждая точка диаграммы получает соответствующий индекс и описание. Затем строится программа, которая будет выполнять последовательность операций, аналогичную описанным в диаграмме.

**6. Применение Машин и Диаграмм Тьюринга**

Машины Тьюринга (МТ) и Диаграммы Тьюринга (ДТ) — универсальные модели вычислений, которые находят применение в следующих областях:

* **Исследования в теории вычислимости и сложности:**
  + МТ используются для анализа задач на предмет их вычислимости, то есть можно ли решить задачу алгоритмически.
  + Исследование классов сложности (P, NP и др.) базируется на моделях, эквивалентных МТ.
* **Формальная верификация алгоритмов:** Диаграммы Тьюринга помогают визуализировать и верифицировать работу алгоритмов, показывая переходы между состояниями.
* **Моделирование программ и прототипирование:** МТ используются для формального доказательства того, что программное обеспечение может реализовать заданную функцию.

**7. Модификации Машин Тьюринга и Диаграмм Тьюринга**

Несмотря на различные модификации Машин Тьюринга, такие как многоленточные, недетерминированные, с полубесконечными и ограниченными лентами, вычислительная мощность этих вариантов не выходит за пределы базовой Машины Тьюринга. Это подтверждается теоремой Чёрча-Тьюринга, которая утверждает, что все эти вариации эквивалентны базовой МТ с точки зрения вычислительных возможностей.

*Примечание*: Диаграммы Тьюринга остаются мощным инструментом для визуализации и анализа таких машин. Эти модификации отображаются в диаграммах с учетом особенностей каждой вариации, например, многоленточная МТ будет иметь несколько головок и переходов, в то время как недетерминированная МТ будет отображать несколько возможных путей переходов в одном состоянии.

## Нормальные Алгоритмы Маркова

Нормальные Алгоритмы Маркова (НАМ) представляют собой теоретическую модель вычислений, предложенную в 1950 году академиком Андреем Марковым

**1. Определение Нормального Алгоритма Маркова**

НАМ можно рассматривать как систему детерминистических текстовых замен, которые задают вычисления по приоритету операций. Каждый алгоритм состоит из упорядоченного списка правил вида:

α→β, где:

* α — строка символов (левая часть правила),
* β — строка символов (правая часть правила).

Правила применяются к строке символов последовательно. Если левая часть правила α найдена в строке, то она заменяется на правую часть β. Этот процесс продолжается, пока не будет выполнено завершение алгоритма, которое происходит в одном из следующих случаев:

* Если все правила оказываются неприменимыми (то есть нет возможности для замены),
* Если применилось завершительное правило, помеченное точкой.

**2. Принцип Применения Правил**

Основные принципы применения правил в НАМ следующие:

1. **Приоритет применения:** Если несколько правил могут быть применены одновременно, выбирается правило, которое встречается первым в списке (то есть имеет более высокий приоритет).
2. **Первое вхождение:** Если одно и то же правило может быть применено к нескольким частям строки, замена происходит только в первой найденной позиции слева.

**3. Сравнение НАМ с Машинами Тьюринга и Диаграммами Тьюринга**

Хотя НАМ представляет собой отличную по форме модель вычислений, она эквивалентна другим моделям, таким как **Машины Тьюринга (МТ)** и **Диаграммы Тьюринга (ДТ)**, с точки зрения вычислительной мощности. Все три модели могут решать одни и те же задачи и обладают одинаковыми вычислительными возможностями. Таким образом, можно преобразовать НАМ в эквивалентную Машину Тьюринга или диаграмму Тьюринга, и наоборот.

* **МТ** и **ДТ** — это более сложные, но эквивалентные модели, использующие ленту для хранения данных и описания переходов состояний.
* **НАМ** работает по принципу текстовых замен и не использует ленту, что делает её менее удобной для описания некоторых типов вычислений, но теоретически она обладает той же вычислительной мощностью, что и МТ.

**4. Практическое Применение НАМ**

Несмотря на теоретическую эквивалентность с Машинами Тьюринга, НАМ редко используется на практике для решения реальных задач. Это объясняется следующими факторами:

* **Сложность отладки:** Алгоритмы на основе НАМ трудны в отладке и неэффективны в сравнении с императивными языками программирования, такими как C++, Python, Java.
* **Низкая эффективность:** Для разработки программного обеспечения, например, для сортировки чисел или обработки данных, использование НАМ было бы крайне неудобным и непрактичным.
* **Теоретическая роль:** Основное значение НАМ заключается в его теоретической применимости, в частности, для изучения вычислимости и формальных языков.

**5. Структура и Мета-Символы**

Для упрощения записи и представления алгоритмов часто используются **мета-символы** — символы, которые не входят в заданный алфавит, но служат для сокращения записи и улучшения понимания правил. Мета-символы помогают структурировать и оптимизировать набор правил, облегчая чтение и запись алгоритмов, особенно при работе с большими и сложными системами замен.

## ****Конечные автоматы****

**Конечный автомат (КА)** — это абстрактная модель вычислений, в которой система может находиться в одном из конечного числа состояний и переходить между ними под воздействием входных символов. В отличие от машины Тьюринга (МТ), КА не имеет памяти в виде ленты и может лишь последовательно считывать входные символы, не возвращаясь к предыдущим. Их

«память» сводится к текущему состоянию.

#### Типы Конечных Автоматов

1. **Детерминированный конечный автомат (ДКА):** Для каждого состояния и входного символа существует ровно один переход.
2. **Недетерминированный конечный автомат (НКА):** Для каждого состояния и входного символа может существовать несколько возможных переходов или отсутствие перехода. НКА принимает входную строку, если существует хотя бы один путь, ведущий в допускающее состояние.

#### Работа Конечного Автомата

* Автомат начинается в начальном состоянии q0 и считывает входную строку символ за символом.
* Для каждого символа применяется функция переходов, которая определяет, в какое следующее состояние перейти.
* Если после обработки всей строки автомат находится в одном из допускающих состояний, строка считается принятой. В противном случае — отклоненной.

#### Применение Конечных Автоматов

1. **Валидация ввода:** КА используются для проверки данных на соответствие заданным формам (например, проверка адреса электронной почты или номера телефона).
2. **Лексический анализ:** В компиляторах КА используются для разбиения исходного кода на токены (лексемы).
3. **Поиск текста:** Программы для поиска текста (например, с использованием регулярных выражений) основаны на КА.
4. **Системы управления:** КА моделируют системы управления, такие как светофоры, автоматы для продажи напитков, лифты и т. д.

**Сравнение Конечных Автоматов с Другими Моделями Вычислений**

1. ***Машины Тьюринга (МТ) и Диаграммы Тьюринга (ДТ):***
   * МТ и ДТ эквивалентны по своей вычислительной мощности и могут моделировать любые вычисления. Оба используют ленту как память для хранения данных и имеют возможность переходить между состояниями на основе содержимого ленты.
   * В отличие от них, КА не имеют ленты и ограничены лишь последовательно считывать входные символы. Это делает КА менее мощными в плане вычислительных возможностей, так как они не могут моделировать задачи, требующие произвольного доступа к данным (например, вычисления с произвольным количеством памяти).
2. ***Нормальные Алгоритмы Маркова (НАМ):***
   * НАМ, как и КА, являются абстрактными моделями вычислений, но НАМ основываются на текстовых заменах (заменах строк символов) и не используют память в виде ленты. НАМ могут решать те же задачи, что и МТ, но их использование на практике также ограничено из-за сложности реализации и отладки.
   * КА проще и имеют явную связь с реальными задачами, такими как валидация ввода и лексический анализ, в то время как НАМ ориентированы больше на теоретические исследования в области вычислимости.

# Практическая часть

# 1. Машина Тьюринга

1.1. Описание программы**:**

Выделение разряды второго двоичного числа по маске, заданной первым числом. Таким образом весь алфавит, с которым будет работать программа будет включать в себя 0, 1 и λ.

## 1.2. Идея алгоритма:

1. Копирование первого числа.
2. Копирование второго числа.
3. Замена бита маски третьего числа, битом четвертого числа, находящегося на таком же отдалении с конца.
4. Удаление ведущих нулей.
5. Сдвиг полученного результата, чтобы между ним и входными данным осталась одна λ.

## 1.3. Сценарий выполнения.

1. Реализовать копирование двоичного числа
2. Реализовать копирование двоичного числа, которое стоит перед другим двоичным числом
3. Реализовать побитовое выделение разрядов числа.
4. Посмотреть, как работает программа при числах разной длины и учесть эти моменты при копировании и побитовом выделении (например, двузначные и трехзначные числа).
5. Реализовать удаление ведущих нулей.
6. Реализовать удаление лишних пробелов.
7. Реализовать алгоритм для сдвига получившегося результата на расстояние одной λ между входными данными.
8. Реализовать алгоритм, который двигает сдвигает головку в новое начальное положение (первая λ после результата).

## Тесты:

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый результат | Комментарий |
| 111 101 | 101 | Одинаковая длина |
| 1111 100 | 100 | Маска больше |
| 111 1011 | 011 | Выделяемое число больше |
| 101 100 | 10 | В маске имеются нули |
| 111 001 | 1 | Удаление ведущих нулей |
| 11111 00000000000 | 0 | Выделяемо только из нулей |

# 2. Диаграмма Тьюринга

## 2.1. Описание программы:

Вычисление наибольшего общего делителя двух чисел в десятичной системе счисления. Таким образом весь алфавит, который нам понадобиться будет включать в себя 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и λ.

## 2.2. Идея алгоритма:

* 1. Копируем первое число.
  2. Копируем второе число.
  3. Копируем, уже скопированное первое число.
  4. Копируем, уже скопированное второе число.
  5. Вычитаем числа.
  6. На место большего числа записываем результат вычисления.

## 2.3. Сценарий выполнения:

1. Копирование первого числа
2. Удаление ведущих нулей копированного первого числа и проверка, что это не ноль.
3. Копирование второго числа.
4. Удаление ведущих нулей копированного первого числа и проверка, что это не ноль.
5. Повторное копирование первого и второго чисел на позиции пятого и шестого чисел.
6. Вычитание из пятого числа шестого.
7. Если результат положительный, то переход к пункту 11. Иначе переход к пункту 8.
8. Удаление пятого и шестого чисел.
9. Копирование четвертого числа на позицию пятого.
10. Копирование третьего числа на позицию шестого. Переход в пункт 6.
11. Запись результата вычитания на место большего из третьего и четвёртого чисел.
12. Переходим в пункт 5. Продолжаем это выполнение, пока результат вычитания не станет равен нулю.
13. Если результат вычитания равен нулю, то записанные числа на третью и четвёртые позиции являются итоговым результатом.
14. Удаление ведущих нулей получившегося результата.
15. Сдвиг результата на расстояние одной λ до входных данных.
16. Перевод головки на конечное положение (первое λ после результата).

Диаграмма основной машины «gcd»:

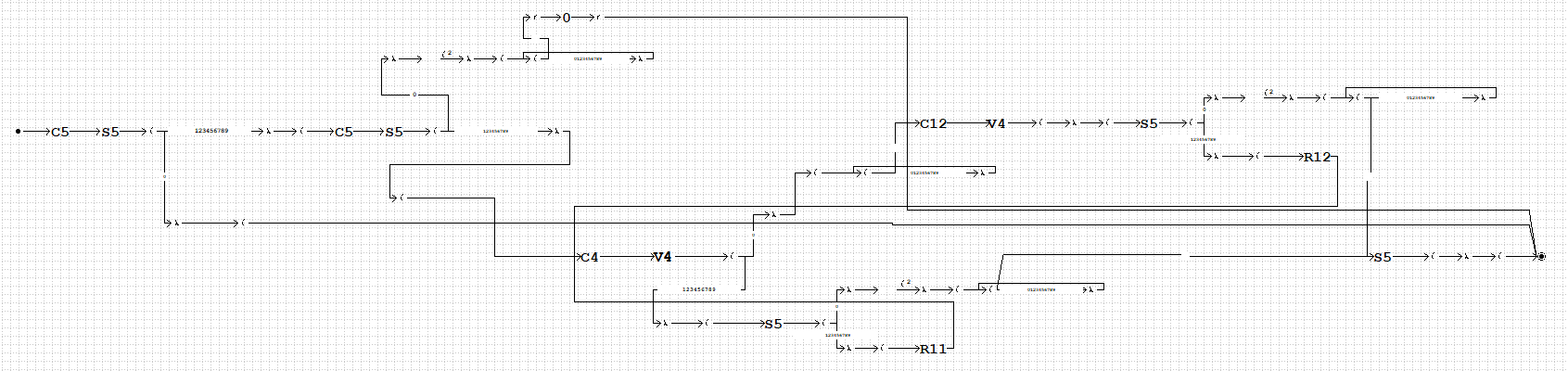


Диаграмма подмашины «copy»:

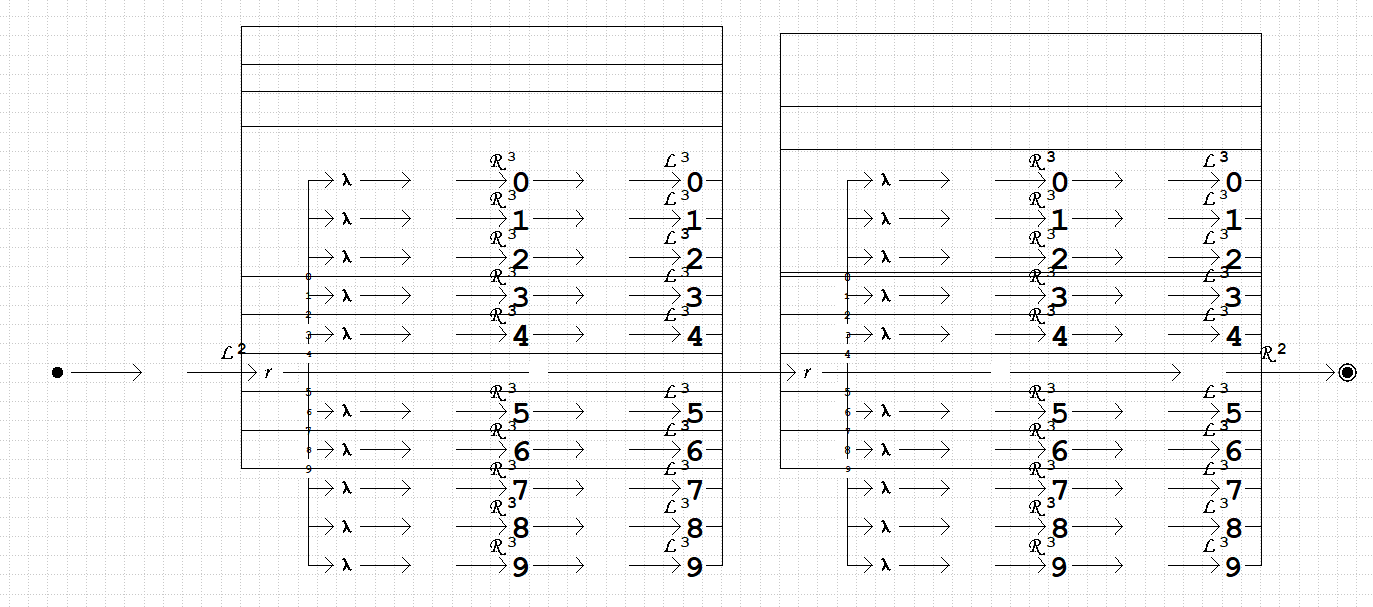


Диаграмма подмашины «copy1»:

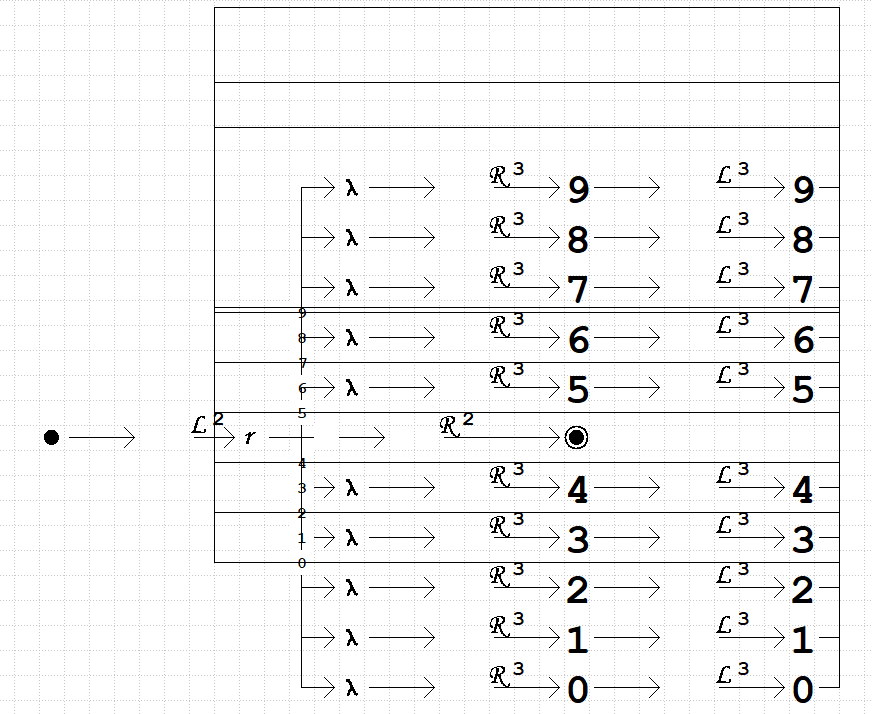


Диаграмма подмашины «copy\_reverse»:

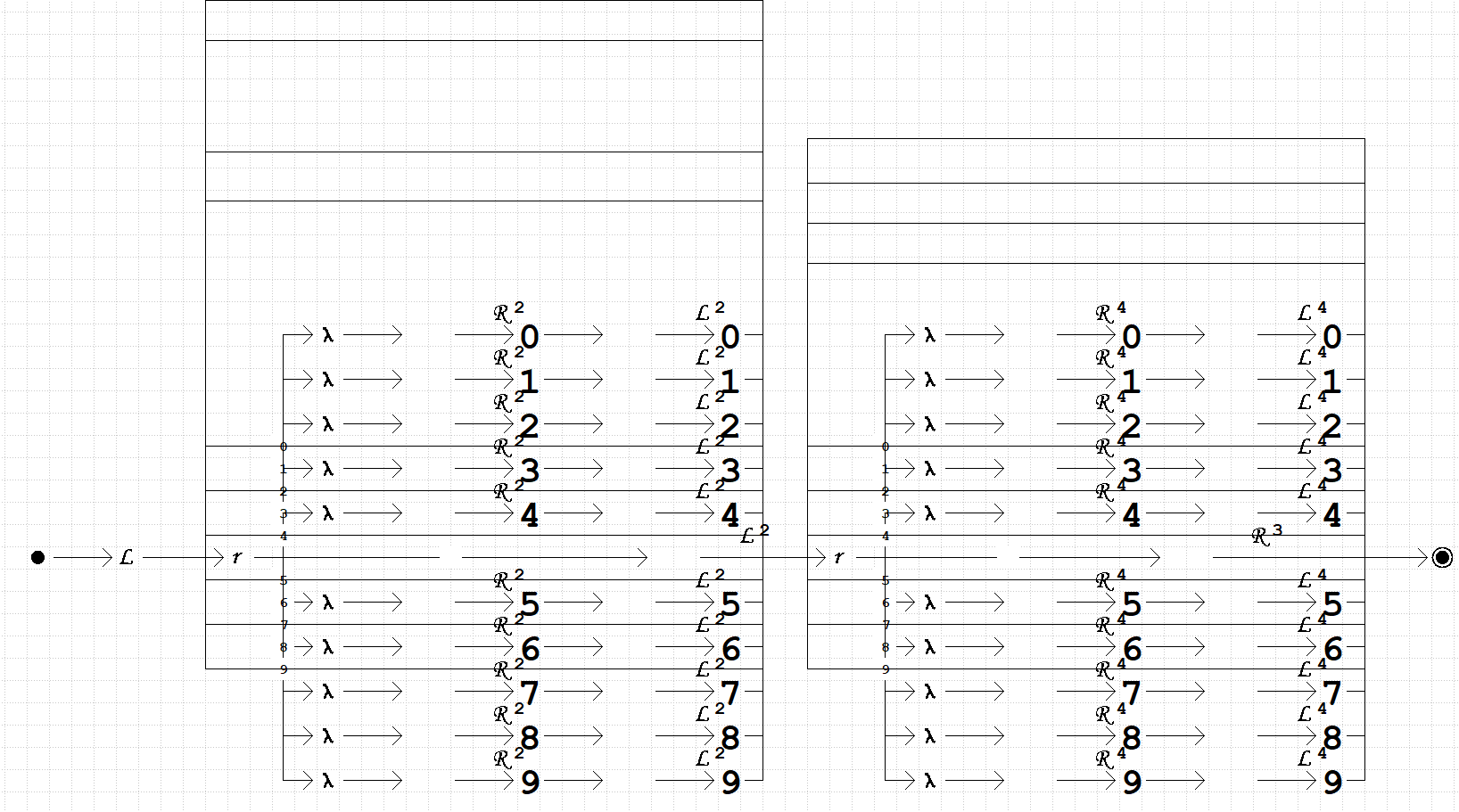


Диаграмма подмашины «replace\_one»:

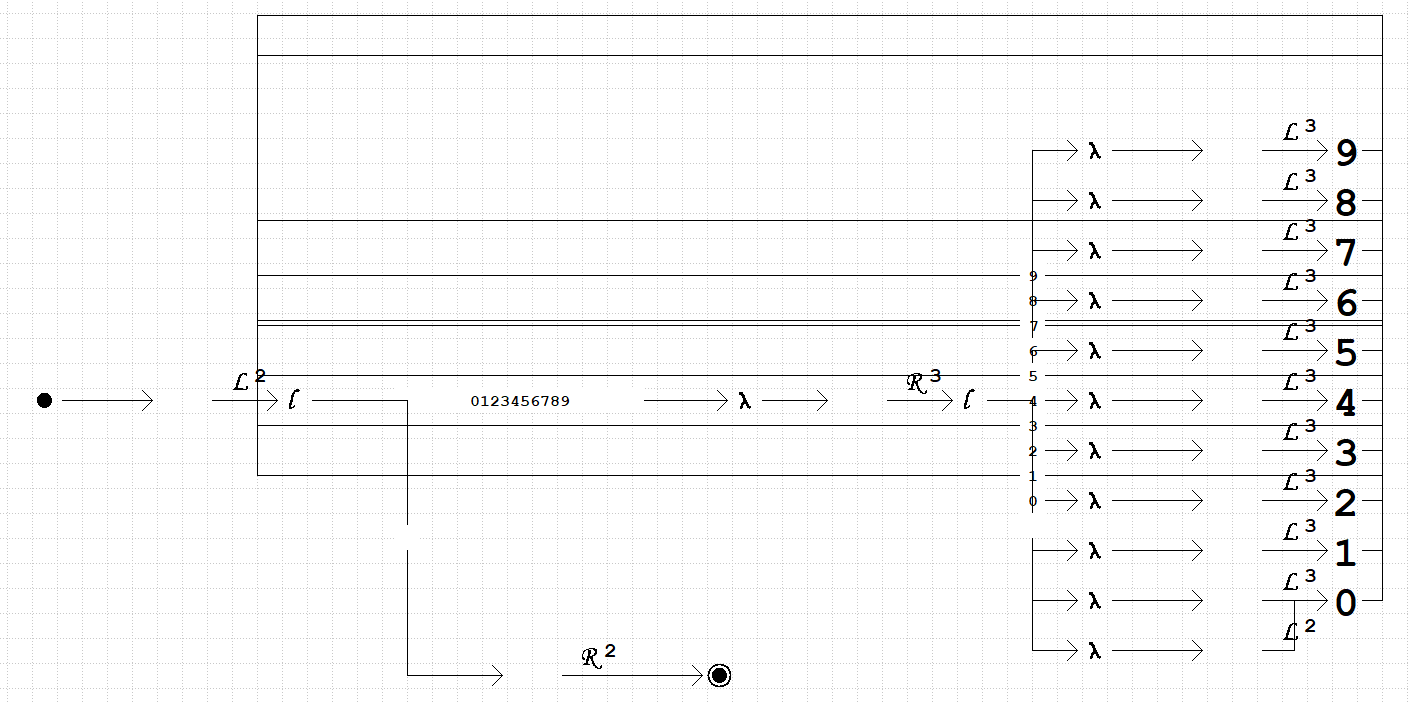


Диаграмма подмашины «replace\_two»:

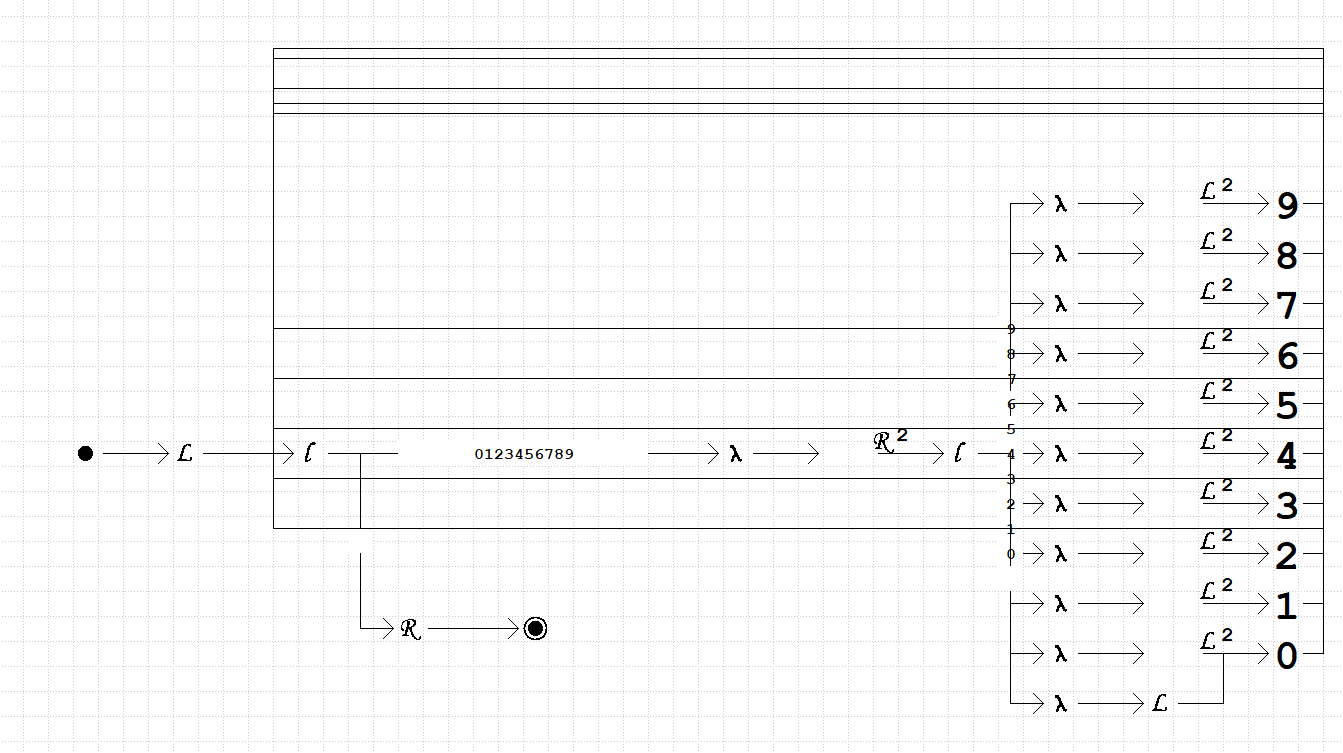


Диаграмма подмашины «sdvig»:

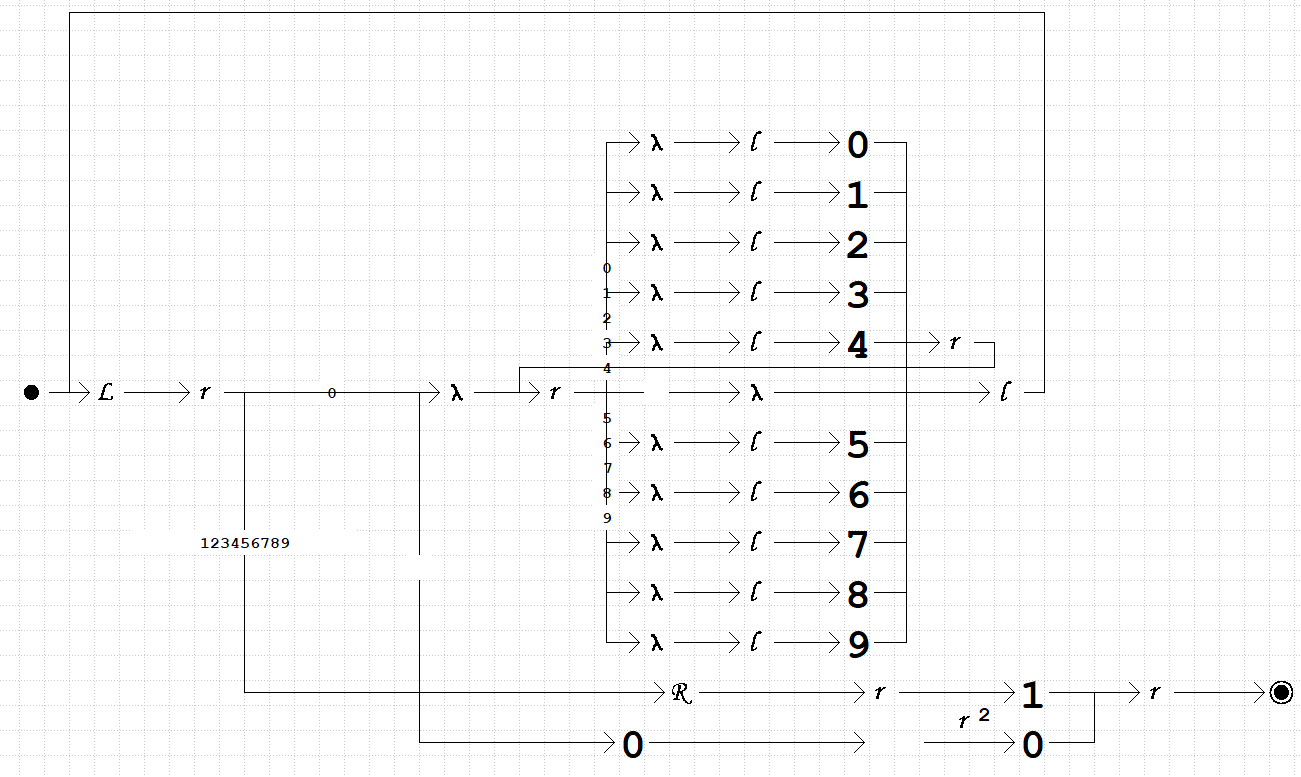
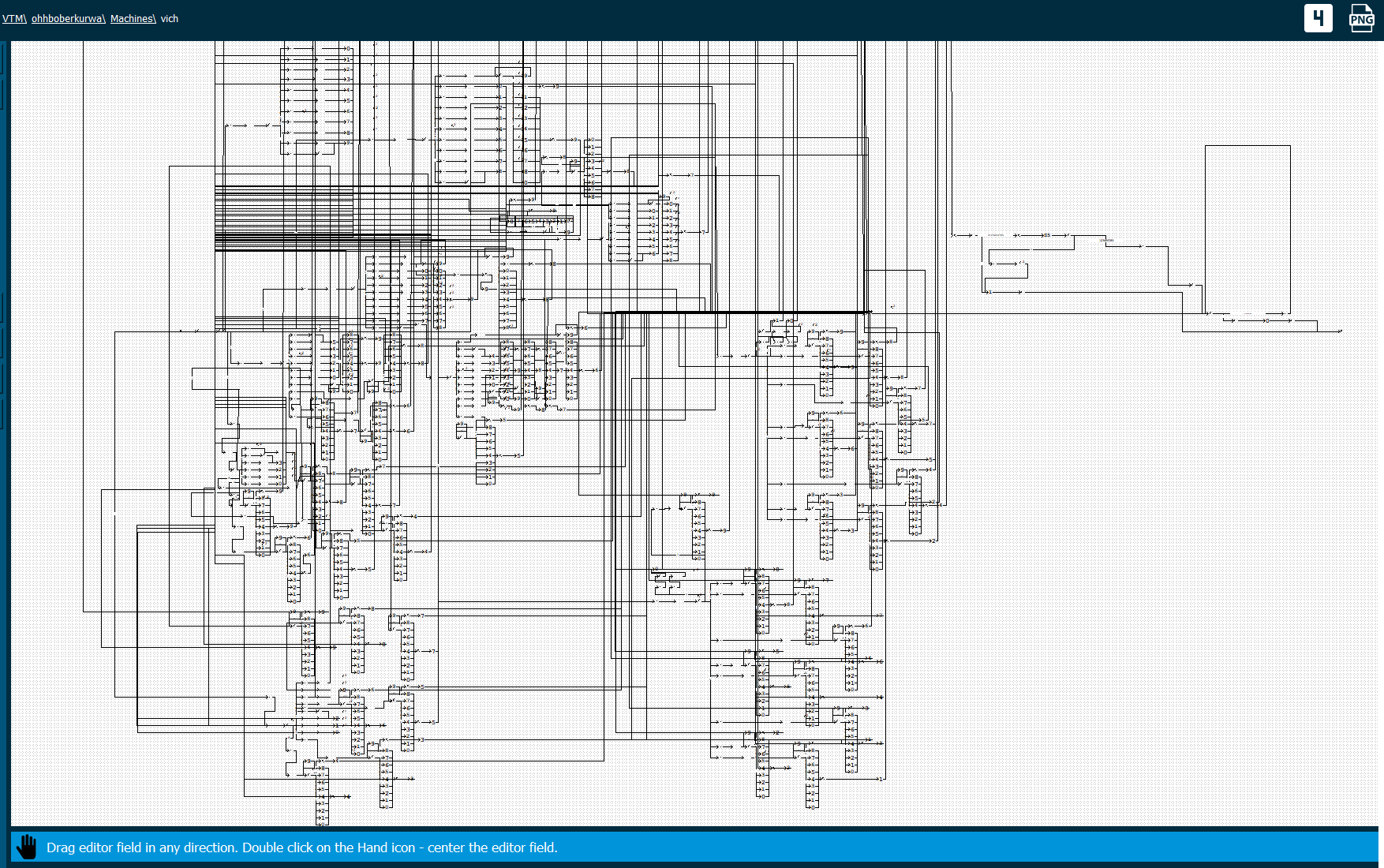


Диаграмма подмашины «vich»:



## Тесты:

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый результат | Комментарий |
| 2 1 | 1 | - |
| 48 0 | 0 | Одно из чисел ноль |
| 48 012 | 12 | Одно из чисел начинается с нуля |
| 3 12 | 3 | Второе число больше первого |

# 3. Нормальный алгоритм Маркова

## 3.1. Описание программы:

Входное слово представляет собой троичное число без знака. Составьте алгоритм реверса числа. Алфавит: {0, 1, 2, \*, . , #, |, &, \_}.

## 3.2. Идея алгоритма:

* 1. Выделение первой слева цифры.
  2. Перемещение цифры в конец записи.
  3. Ограничение цифры символом |.
  4. Удаление разделителей.
  5. Удаление ведущих нулей.

## 3.3. Сценарий выполнения:

0\*0->00\*

0\*1->10\*

0\*2->20\*

1\*0->01\*

1\*1->11\*

1\*2->21\*

2\*0->02\*

2\*1->12\*

2\*2->22\*

0\*->#|0

1\*->#|1

2\*->#|2

0#->#0

1#->#1

2#->#2

#0->0\*

#1->1\*

#2->2\*

&->.

#|->\_|

\_|0|->\_|

\_|1|->1-|

\_|2|->2-|

-|0|->0-|

-|1|->1-|

-|2|->2-|

\_|0 ->0&

\_|1 ->1-|

\_|2 ->2-|

-|->.

->#

* 1. Пишем правила для выделения первой слева цифры.
  2. Перемещаем выделенную цифру слева направо.
  3. При достижении конца слова, отделяем цифру символом |.
  4. Возвращаемся в пункт 1 и выполняем всё заново, пока есть цифры слева, не имеющие выделения | перед ними.
  5. Из получившегося слова удаляем все |, тем самым получаем число.
  6. Если имеются ведущие нули, то удаляем их.

## Тесты:

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый результат | Комментарий |
| 2102 | 2012 | - |
| 100 | 1 | Ведущие нули |
| 011 | 110 | Ведущие нули во входных данных |

# 4. Конечные автоматы

## 4.1. Описание программы:

Выделить все десятичные числа от 17 до 77 по модулю и распечатать их значение в словесной форме по-французски.

## 4.2. Идея алгоритма:

* 1. Считывание потока символов из стандартного ввода.
  2. Проверка каждого символа для определения текущего состояния конечного автомата
  3. Распознавание чисел:
     + - Если число находится в пределах от 17 до 77, программа запоминает его первую и вторую цифры.
       - Программа проверяет знак числа: положительное или отрицательное
  4. Преобразование чисел в текстовое представление на французском:
     + - Использование специальных правил для чисел 17−19 и 70−77, так как они отличаются в системе французского счета.
  5. Печать результата в текстовом формате.

## 4.3. Сценарий выполнения:

1. ***Инициализация:*** 
   1. Создается переменная state типа int и инициализируется значением 0. Она будет хранить текущее состояние конечного автомата.
   2. Создается переменная minus\_flag типа int и инициализируется значением 1. Она указывает, положительное или отрицательное число обрабатывается.
   3. Создаются переменные digFirst и digSecond типа int, инициализируемые значением 0. Они хранят первую и вторую цифры текущего числа.
   4. Создается переменная ch типа int. Она используется для хранения текущего символа, считанного из входного потока.
2. ***Цикл по символам входного потока:*** 
   1. **Обработка начального состояния (state = 0)**:
      1. Если текущий символ ch равен '+' или '-', программа переходит в состояние 6 (обработка знака) и устанавливает соответствующее значение minus\_flag.
      2. Если ch — цифра от '0' до '7', программа переходит в состояние, соответствующее первой цифре числа (state = determine\_state(ch)), и сохраняет цифру в digFirst.
      3. Если ch является разделителем (например, пробел, табуляция или новая строка), состояние остается в 0.
      4. Во всех остальных случаях состояние изменяется на 7 (пропуск).
   2. **Обработка состояний первой цифры (state = 1, 2, 3, 4):**
      1. Если текущий символ — цифра от '0' до '9', программа переходит в состояние 8 (вывод числа) и сохраняет цифру в digSecond.
      2. Если ch является разделителем, состояние сбрасывается в 0.
      3. Иначе программа переходит в состояние 7 (пропуск).
   3. Обработка состояния знака (state = 6):
      1. Если текущий символ — цифра от '0' до '7', программа переходит в состояние, соответствующее первой цифре (state = determine\_state(ch)), и сохраняет цифру в digFirst.
      2. Если ch является разделителем, состояние сбрасывается в 000 с установкой minus\_flag = 1
   4. Обработка состояния вывода числа (state = 8):
      1. Если текущий символ — разделитель:
         1. Проверяется, находится ли число в диапазоне от 17 до 77. Если да:
            1. Если minus\_flag равен 0, выводится строка "moins ".
            2. Для чисел с первой цифрой 7 вызывается функция lastDigit\_to\_Word(digSecond), которая возвращает текстовую строку для чисел от 70 до 77.
            3. Для остальных чисел вызываются функции firstDigit\_to\_Word(digFirst) и secondDigit\_to\_Word(digSecond), которые формируют текстовую строку для числа.
         2. Состояние сбрасывается в 0, а переменные digFirst, digSecond и minus\_flag очищаются.
      2. Если текущий символ не является разделителем, состояние изменяется на 7 (пропуск).
   5. Обработка состояния пропуска (state = 7):
      * Если текущий символ — разделитель, состояние сбрасывается в 000 и переменные очищаются.
3. ***Обработка конца файла (ch = EOF):*** После завершения цикла состояние и переменные проверяются на наличие незавершенного числа. Если число находится в допустимом диапазоне, оно обрабатывается так же, как в состоянии вывода.
4. ***Вывод результата:*** Для каждого числа, удовлетворяющего условиям, на экран выводится его текстовая форма, при необходимости с указанием отрицательного знака (moins).

## Тесты:

|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| 0 17 -56 89 арйщфш EOF | dix-sept moins cinquante-six |
| EOF |  |
| 19EOF | dix-neuf |
| 74ad 0000076 EOF | soixante-seize |
| -00063 asasdasd EOF | moins soixante-trois |

# Заключение

Данная курсовая работа представляла собой исследование фундаментальных моделей вычислений, играющих ключевую роль в теоретической информатике и разработке алгоритмов. В рамках работы были детально рассмотрены четыре модели: машина Тьюринга, диаграммы Тьюринга, нормальные алгоритмы Маркова и конечные автоматы. Теоретический анализ каждой модели включал описание ее структуры, детальное объяснение принципов функционирования и оценку вычислительных возможностей.

Практическая часть работы была сосредоточена на реализации конкретных алгоритмов с использованием каждой из изученных моделей. Этот практический подход позволил не только применить полученные теоретические знания, но и значительно углубить понимание нюансов работы каждой модели.

Все разработанные программы подверглись тщательному тестированию на разнообразных наборах входных данных для подтверждения их корректности и эффективности. Процесс тестирования являлся неотъемлемой частью работы, гарантирующей надежность реализованных алгоритмов.

Одним из важных результатов курсовой работы стало практическое подтверждение эквивалентности машины Тьюринга, диаграмм Тьюринга и нормальных алгоритмов Маркова как Тьюринг-полных моделей вычислений. Это означает, что эти модели обладают одинаковой вычислительной мощностью и способны решать любую задачу, вычислимую в принципе. В отличие от них, конечные автоматы имеют ограниченные возможности и эффективны преимущественно для решения задач распознавания относительно простых шаблонов.

Таким образом, данная курсовая работа дала глубокое понимание различных моделей вычислений, их возможностей и ограничений. Полученные навыки анализа и построения алгоритмов, а также опыт практической реализации и тестирования программ, являются ценным багажом для дальнейшего изучения теории вычислимости, разработки более сложных алгоритмов и успешной профессиональной деятельности в области информатики.

# Список источников

1. Гайсарян, С. С., Зайцев, В. Е. Курс информатики. Издание 13-е, предварительное, переработанное и дополненное с приложением электронной хрестоматии
2. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. — М.: «Вильямс», 2002
3. Булос Дж., Джеффри Р. Вычислимость и логика. — М.: Мир, 1994.
4. Манин Ю.И. Вычислимое и невычислимое. — М.: Советское радио, 1980.
5. Минский М. Вычисления и автоматы. — М.: Мир, 1971.
6. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ = Introduction to Algorithms, Third Edition. — 3-е. — М.: [«Вильямс»](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2013