|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт Информационных Технологий | | |
| Кафедра Вычислительной техники | | |

| **ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ №5**  «Реализация автомата» | |
| --- | --- |
|  | |
| **по дисциплине** | |
| **«**Теория формальных языков**»** | |
|  | |
| Выполнил студент группы ИКБО-43-23 | *Жаворонкова А.А.* |
| Принял ассистент | *Цынгалёв П.С.* |

| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2024 г. |  |
| --- | --- | --- |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2024 г. |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_heading=h.ixumht6r895f)

[1 ЦЕЛЬ 4](#_heading=h.hw8xg9o64yvh)

[2 ЗАДАНИЕ 5](#_heading=h.ooqn89hzco1s)

[2.1 Формулировка задания 5](#_heading=h.uqgkq1l1232u)

[2.2 Математическая модель 5](#_heading=h.ssl2ewlrsl5)

[2.3 Тестирование программы 6](#_heading=h.tdl9u6n4o1t)

[ЗАКЛЮЧЕНИЕ 7](#_heading=h.8le2bwtcmujs)

[ПРИЛОЖЕНИЯ 9](#_heading=h.2r1zhkwez3jp)

# ВВЕДЕНИЕ

Изучение теории формальных языков занимает важное место в образовательной программе по программной инженерии. Эта область, являющаяся неотъемлемой частью компьютерных наук и теоретической информатики, посвящена анализу структуры, свойств формальных языков и моделированию их обработки. В условиях современного мира, где постоянно растут объемы данных и необходимость в эффективных методах их обработки, хранения и передачи, понимание основ формальных языков становится особенно актуальным.

Формальные языки находят применение в различных областях: от описания синтаксиса языков программирования до разработки сетевых протоколов и создания компиляторов. Они позволяют формализовать взаимодействие как между человеком и компьютером, так и между различными компонентами программного обеспечения. Основу теории формальных языков составляют такие понятия, как грамматики, автоматы и алгебраические структуры, которые используются для описания синтаксиса и семантики языков.

В этом отчете будет рассмотрена программа преобразования недетерминированного конечного автомата (НКА) в детерминированный конечный автомат (ДКА).

# 1 ЦЕЛЬ

Графически представить недетерминированный конечный автомат, детерминированный конечный автомат и минимальный детерминированный конечный автомат (если возможно) согласно варианту. Реализовать автомат на выбранном ЯП согласно варианту.

# 2 ЗАДАНИЕ

## 2.1 Формулировка задания

На выбранном ЯП реализовать автомат согласно варианту.

Вариант: letter(letter|digit|\_)\*

## 2.2 Математическая модель

Создаем множества letters и digits для классификации символов строки как буквы, цифры или символы подчеркивания. Это необходимо для обработки каждого символа входной строки.

Разрабатываем функцию create\_automaton, которая генерирует автомат. В ней задаются:

- множество состояний states со значением {0, 1, 2, 3},

- начальное состояние initial\_state, равное 0,

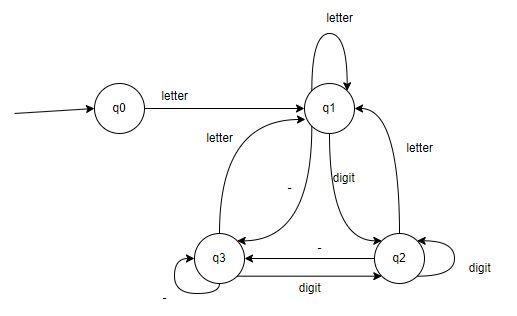
- множество конечных состояний final\_states со значением {1, 2, 3},

- словарь переходов transition\_function, описывающий правила изменения состояний в зависимости от типа входного символа.

Реализуем функцию process\_input, которая выполняет обработку строки. Она определяет тип каждого символа (буква, цифра, подчеркивание) и проверяет наличие допустимого перехода из текущего состояния. Если символ не подходит или переход отсутствует, функция возвращает False. После обработки всей строки проверяется, находится ли автомат в конечном состоянии. Если да, возвращается True.

В основной части программы автомат создается с помощью функции create\_automaton, затем строка анализируется функцией process\_input. По результатам анализа выводится сообщение: "Строка принята" или "Строка не принята".

Графическое представление недетерминированного конечного автомата приведено на рисунке 1.

**  
Рисунок 1 – Графическое представление недетерминированного конечного автомата**

НКА эквивалентен ДКА.

## 2.3 Тестирование программы

Проведём тестирование для данной программы на разных значениях. Результаты продемонстрируем на рисунке 2, 3.

 **Рисунок 2 - Первое тестирование программы**

 **Рисунок 3 - Второе тестирование программы**

# ЗАКЛЮЧЕНИЕ

В процессе выполнения практической работы были достигнуты следующие результаты:

Были реализованы: графическое представление недетерминированного конечного автомата, графическое представление детерминированного конечного автомата, графическое представление минимального детерминированного конечного автомата согласно варианту.

Реализован конечный автомат данного варианта на ЯП Python.

Проведено тестирование программы, и результаты представлены в отчете.

Таким образом, поставленная цель работы — Графически представить недетерминированный конечный автомат, детерминированный конечный автомат и минимальный детерминированный конечный автомат (если возможно) согласно варианту. Реализовать автомат на выбранном ЯП согласно варианту — успешно достигнута.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Свердлов С. З. Языки программирования и методы трансляции: учебное пособие. – Санкт-Петербург: Лань, 2019.
2. Малявко А. А. Формальные языки и компиляторы: учебное пособие для вузов. – М.: Юрайт, 2020.
3. Миронов С. В. Формальные языки и грамматики: учебное пособие для студентов факультета компьютерных наук и информационных технологий. – Саратов: СГУ, 2019.
4. Унгер А.Ю. Основы теории трансляции: учебник. – М.: МИРЭА – Российский технологический университет, 2022.
5. Антик М. И., Казанцева Л. В. Теория формальных языков в проектировании трансляторов: учебное пособие. – М.: МИРЭА, 2020.
6. Ахо А. В., Лам М. С., Сети Р., Ульман Дж. Д. Компиляторы: принципы, технологии и инструментарий. – М.: Вильямс, 2008.
7. Ишакова Е.Н. Теория языков программирования и методов трансляции: учебное пособие. – Оренбург: ИПК ГОУ ОГУ, 2007.

# ПРИЛОЖЕНИЯ

Приложение А – код на языке Python

**Приложение А**

Код на языке Python

*Листинг А.1 – main.py*

| import re  letters = set('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz')  digits = set('0123456789')  def create\_automaton():  states = {0, 1, 2, 3} # Состояния  initial\_state = 0 # Начальное состояние  final\_states = {1, 2, 3} # Конечные состояния  # Узлы состояний  transition\_function = {  (0, 'letter'): 1,  (1, 'letter'): 1,  (1, 'digit'): 2,  (1, '\_'): 3,  (2, 'digit'): 2,  (2, 'letter'): 1,  (2, '\_'): 3,  (3, '\_'): 3,  (3, 'letter'): 1,  (3, 'digit'): 2  }  return states, initial\_state, final\_states, transition\_function  def process\_input(input\_string, states, initial\_state, final\_states, transition\_function):  current\_state = initial\_state  for char in input\_string:  # Определяем тип символа: буква, цифра или подчеркивание  if char in letters:  char\_type = 'letter'  elif char in digits:  char\_type = 'digit'  elif char == '\_':  char\_type = '\_'  else:  return False # Если символ не подходит, сразу возвращаем False  # Переход к новому состоянию  if (current\_state, char\_type) in transition\_function:  current\_state = transition\_function[(current\_state, char\_type)]  else:  return False  return current\_state in final\_states  if \_\_name\_\_ == "\_\_main\_\_":  states, initial\_state, final\_states, transition\_function = create\_automaton()  input\_string = input("Введите строку для анализа: ")  if process\_input(input\_string, states, initial\_state, final\_states, transition\_function):  print("Строка принята")  else:  print("Строка не принята") |
| --- |