МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

УТВЕРЖДАЮ:
Директор института
_____ С.И. Лукьянов
«26» сентября 2018 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

ТЕОРИЯ АВТОМАТОВ

Направление подготовки (специальность)
09.03.01 Информатика и вычислительная техника

Направленность (профиль/ специализация) программы
Программное обеспечение средств вычислительной техники и
автоматизированных систем

Уровень высшего образования – бакалавриат

Программа подготовки – академический бакалавриат

Форма обучения
Очная

Институт	*энергетики и автоматизированных систем*
Кафедра	*вычислительной техники и программирования*
Курс	*2*
Семестр	*4*

Магнитогорск
2018 г.

Рабочая программа составлена на основе ФГОС ВО по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника, утвержденного приказом МО и Н РФ от 12.01.2016 № 5.

Рабочая программа рассмотрена и одобрена на заседании кафедры вычислительной техники и программирования «05» сентября 2018 г., протокол № *1*.

Зав. кафедрой _____ О.С. Логунова

Рабочая программа одобрена методической комиссией института энергетики и автоматизированных систем «26» сентября 2018 г., протокол № 1.

Председатель _____ С.И. Лукьянов

Рабочая программа составлена ст. преподавателем каф. вычислительной техники и программирования
–

_____ М.В. Зарецким

Рецензент:

начальник отдела инновационных разработок ЗАО «КонсОМ-СКС», канд. техн. наук

_____ А.Н. Панов

Лист регистрации изменений и дополнений

№ п/п	Раздел программы	Краткое содержание изменения/дополнения	Дата. № протокола заседания кафедры	Подпись зав. кафедрой
1	8	Корректировка списка рекомендуемой литературы	2,09,2019, протокол №1	
2	9	Обновление ссылки на перечень программного обеспечения	2,09,2019, протокол №1	

3

1 Цели освоения дисциплины (модуля)

Целями освоения дисциплины «Теория автоматов» являются:

- освоение принципов автоматной парадигмы программирования;
- освоение взаимосвязей автоматной и объектно-ориентированной парадигм программирования;
- освоение современных методов проектирования программных продуктов на основе автоматной парадигмы.

Для достижения поставленных целей в курсе «Функциональное программирование» решаются задачи:

- изучение языковых средств моделирования автоматов;
- освоение методов проектирования программных систем в автоматной парадигме;
- изучение современных применений автоматной парадигмы программирования.

2 Место дисциплины (модуля) в структуре образовательной программы подготовки бакалавра (магистра, специалиста)

Дисциплина «Теория автоматов» входит в вариативную часть блока 1 образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения следующих дисциплин:

- математики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, позволят обучающимся осмысленно выполнять операции над векторами и матрицами, применять понятие функции, грамотно строить суперпозиции функций;
- математической логики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, позволят обучающимся выполнять операции предикатами, строить формальные описания функционирования многокомпонентных систем;
- информатики (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для работы с информационными потоками на профессиональном уровне;
- прикладного программирования (базовая часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении данной дисциплины, являются основой для освоения методологии разработки программ в автоматной парадигме.

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для изучения следующих дисциплин:

- паттерное программирование (вариативная часть блока 1 образовательной программы). Знания, умения и навыки, полученные при изучении дисциплины «Теория автоматов», позволят обучающимся понять паттерны программирования, как реализацию концепции конечных автоматов;
- Scada-системы (вариативная часть блока 1 образовательной программы). Знания, умения и владения, полученные при изучении дисциплины «Теория автоматов» позволят обучающимся выполнять моделирование данных систем в автоматной парадигме.

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Теория автоматов» обучающийся должен обладать следующими компетенциями:

Структурный элемент компетенции	Планируемые результаты обучения

Структурный элемент компетенции	Планируемые результаты обучения
ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.	
Знать	– принципы синтеза цифровых автоматов, основные понятия автоматного программирования; – способы программного задания цифровых автоматов; – общие методы структурного синтеза автоматов, принципы моделирования предметной области в автоматной парадигме.
Уметь	– использовать методы синтеза цифровых автоматов, использовать методы проектирования автоматных программ; – строить распознаватели и преобразователи, сложные схемы взаимодействия автоматов; – разрабатывать многокомпонентные недетерминированные системы.
Владеть	– навыками реализации автоматных моделей на языках программирования высокого уровня; – навыками проектирования и реализации сложных автоматных моделей на языках программирования высокого уровня; – навыками реализации недетерминированных моделей, сочетающих автоматную и объектно-ориентированную парадигму моделирования.

4 Структура и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 4 зачетные единицы 144 акад. часа, в том числе:

- контактная работа – 54,15 акад. часов:
  - аудиторная – 51 акад. час;
  - внеаудиторная – 3,15 акад. часа
- самостоятельная работа – 54,15 акад. часа;
- подготовка к экзамену – 35,7 акад. часа

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
1. Раздел 1. Теория абстрактных автоматов	4							
1.1. Тема. Области применения автоматного подхода. Парадигма автоматного программирования. Абстрактные и структурные автоматы. Автоматы Мили и Мура. Модель дискретного преобразования В.М. Глушкова. Программная реализация автомата	4	2	2		6	Самостоятельное изучение учебной и научной литературы.	Беседа – обсуждение. Устный опрос.	*ПК-2 – зув*
1.2. Тема. Стандартные способы задания цифровых автоматов с памятью. Программная реализация	4	2	2		6	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув*
Итого по разделу	**4**	**4**	**4**		**12**		**Проверка индивидуальных заданий**	

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
2. Раздел. Теория структурных автоматов	4							
2.1. Тема. Методы структурного синтеза конечных автоматов. Программная реализация конечных автоматов.	4	2	4/2И		6	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув*
2.2. Тема Недетерминированные автоматы.	4	2	4/2И		6	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув*
Итого по разделу	**4**	**4**	**8/4И**		**12**		**Проверка индивидуальных заданий**	
3. Раздел. Теория формальных грамматик.								
3.1. Тема. Классификация формальных грамматик по Н. Хомскому. Примеры формальных грамматик и их применения.	4	2	6		6	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув*
3.2. Тема. Регулярные выражения.	4	2	6/2И		6	Самостоятельное изучение учебной и научной литературы.	Беседа – обсуждение. Анализ программного кода.	*ПК-2 – зув*

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа (в акад. часах)	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код и структурный элемент компетенции
		лекции	лаборат. занятия	практич. занятия				
						Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Устный опрос.	
Итого по разделу	**4**	**4**	**12/2И**		**12**		**Проверка индивидуальных заданий**	
4. Раздел. Автоматная парадигма программирования	4							
4.1. Тема. Программирование с явным выделением состояний.	4	5	10/8И		6	Самостоятельное изучение учебной и научной литературы. Подготовка к лабораторному занятию. Выполнение лабораторной работы.	Беседа – обсуждение. Анализ программного кода. Устный опрос.	*ПК-2 – зув*
Итого по разделу		**5**	**10/8И**		**18,15**		**Проверка индивидуальных заданий**	
Итого за семестр		**17**	**34/14И**		**54,15**		**Экзамен**	
Итого по дисциплине		**17**	**34/14И**		**54,15**			

5 Образовательные и информационные технологии

1. **Традиционные образовательные технологии** ориентируются на организацию образовательного процесса, предполагающую прямую трансляцию знаний от преподавателя к студенту (преимущественно на основе объяснительно-иллюстративных методов обучения). Учебная деятельность студента носит в таких условиях, как правило, репродуктивный характер.

Формы учебных занятий с использованием традиционных технологий:

Информационная лекция – последовательное изложение материала в дисциплинарной логике, осуществляемое преимущественно вербальными средствами (монолог преподавателя).

Семинар – беседа преподавателя и студентов, обсуждение заранее подготовленных сообщений по каждому вопросу плана занятия с единым для всех перечнем рекомендуемой обязательной и дополнительной литературы.

Практическое занятие, посвященное освоению конкретных умений и навыков по предложенному алгоритму.

Лабораторная работа – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

2. **Технологии проблемного обучения** – организация образовательного процесса, которая предполагает постановку проблемных вопросов, создание учебных проблемных ситуаций для стимулирования активной познавательной деятельности студентов.

3. **Интерактивные технологии** – организация образовательного процесса, которая предполагает активное и нелинейное взаимодействие всех участников, достижение на этой основе личностно значимого для них образовательного результата. Наряду со специализированными технологиями такого рода принцип интерактивности прослеживается в большинстве современных образовательных технологий. Интерактивность подразумевает субъект - субъектные отношения в ходе образовательного процесса и, как следствие, формирование саморазвивающейся информационно-ресурсной среды.

Формы учебных занятий с использованием специализированных интерактивных технологий:

Лекция «обратной связи» – лекция–провокация (изложение материала с заранее запланированными ошибками), лекция-беседа, лекция-дискуссия, лекция–пресс-конференция.

4. **Информационно-коммуникационные образовательные технологии** – организация образовательного процесса, основанная на применении специализированных программных сред и технических средств работы с информацией.

6 Учебно-методическое обеспечение самостоятельной работы обучающихся

Задание к лабораторной работе по теме:

Области применения автоматного подхода. Парадигма автоматного программирования. Абстрактные и структурные автоматы. Автоматы Мили и Мура. Модель дискретного преобразования В.М. Глушкова. Программная реализация автомата

Для автомата, заданного таблицей, постройте диаграмму Мура. Задайте этот автомат системой булевых функций:

1.

q	0	1	2	3
x				
0	1;1	3;0	2;0	2;0
1	2;1	2;0	3;0	3;0

2.

q	0	1	2	3
x				
0	1;0	2;0	2;1	3;0
1	3;0	3;1	2;1	1;0

3.

q	0	1	2	3
x				
0	0;1	1;1	2;1	3;1
1	0;0	0;1	3;1	2;1

4.

q	0	1	2	3
x				
0	1;0	3;1	2;0	1;0
1	3;0	1;1	0;1	3;1

5.

q	0	1	2	3
x				
0	1;1	3;1	2;0	1;0
1	3;0	1;0	0;1	3;1

6.

q	0	1	2	3
x				
0	0;0	3;1	2;1	1;0
1	3;0	1;0	0;1	3;1

7.

q	0	1	2	3
x				
0	1;0	3;1	2;1	1;0
1	3;0	1;2	0;1	3;1

8.

q	0	1	2	3
x				
0	0;0	1;1	3;1	2;0
1	2;0	0;1	3;1	1;0

9.

q	0	1	2	3
x				
0	3;0	2;0	1;1	0;1
1	0;0	1;1	2;0	3;0

10.

q	0	1	2	3
x				
0	2;1	2;1	1;1	0;1
1	0;1	1;1	2;0	3;0

Задание к лабораторной работе по теме:

Стандартные способы задания цифровых автоматов с памятью. Программная реализация.

Дан программный код на языке Python. Описать функционирование автомата в конкретном примере модификации функкции jump_to.

```
from __future__ import generators
import sys

def math_gen(n):    # Iterative function becomes a generator
   from math import sin
   while 1:
      yield n
      n = abs(sin(n))*31

# Jump targets not state-sensitive, only to simplify example
def jump_to(val):
   if    0 <= val < 10: return 'ONES'
   elif 10 <= val < 20: return 'TENS'
   elif 20 <= val < 30: return 'TWENTIES'
   else:                return 'OUT_OF_RANGE'

def get_ones(iter):
   global cargo
   while 1:
      print( "\nONES State:      ",)
      while jump_to(cargo)=='ONES':
         print( "@ %2.1f  " % cargo,)
         cargo = iter.next()
      yield (jump_to(cargo), cargo)

def get_tens(iter):
   global cargo
   while 1:
      print "\nTENS State:      ",
      while jump_to(cargo)=='TENS':
         print "#%2.1f  " % cargo,
         cargo = iter.next()
      yield (jump_to(cargo), cargo)

def get_twenties(iter):
   global cargo
   while 1:
      print "\nTWENTIES State:  ",
      while jump_to(cargo)=='TWENTIES':
         print "*%2.1f  " % cargo,
         cargo = iter.next()
      yield (jump_to(cargo), cargo)
```

```
def exit(iter):
    jump = raw_input('\n\n[co-routine for jump?] ').upper()
    print ("...Jumping into middle of", jump)
    yield (jump, iter.next())
    print( "\nExiting from exit()...")
    sys.exit()

def scheduler(gendct, start):
    global cargo
    coroutine = start
    while 1:
        (coroutine, cargo) = gendct[coroutine].next()
if __name__ == "__main__":
    num_stream = math_gen(1)
    cargo = num_stream.next()
    gendct = {'ONES'        : get_ones(num_stream),
              'TENS'        : get_tens(num_stream),
              'TWENTIES'    : get_twenties(num_stream),
              'OUT_OF_RANGE': exit(num_stream)         }
    scheduler(gendct, jump_to(cargo))
```

1.
```
def jump_to(val):
    if    0 <= val < 15: return 'Less 15'
    elif 15 <= val < 30: return 'Less 30'
   elif 30 <= val < 50: return 'Less 50'
   else:                return 'Something else'
```

2.
```
def jump_to(val):
    if    0 <= val < 45: return 'Less 45'
    elif 45 <= val < 70: return 'Less 70'
   elif 70 <= val < 90: return 'Less 90'
   else:                return 'Something else'
```

3.
```
def jump_to(val):
    if    0 <= val < 100: return 'Less 100'
    elif 100 <= val < 300: return 'Less 300'
  elif 300 <= val < 500: return 'Less 500'
   else:                return 'Something else'
```

4.
```
def jump_to(val):
    if    0 <= val < 200: return 'Less 200'
    elif 200 <= val < 300: return 'Less 300'
  elif 30 0<= val < 500: return 'Less 500'
   else:                return 'Something else'
```

5.
```
def jump_to(val):
    if    0 <= val < 300: return 'Less 300'
    elif 300 <= val < 500: return 'Less 500'
```

```
    elif 500 <= val < 700: return 'Less 700'
     else:              return 'Something else'
```

6.

```
def jump_to(val):
    if    0 <= val < 400: return 'Less 400'
    elif 400 <= val < 500: return 'Less 500'
   elif 500 <= val < 700: return 'Less 700'
    else:              return 'Something else'
```

7.

```
def jump_to(val):
    if    0 <= val < 400: return 'Less 300'
    elif 400 <= val < 600: return 'Less 600'
   elif 600 <= val < 700: return 'Less 700'
    else:              return 'Something else'
```

8.

```
def jump_to(val):
    if    0 <= val < 500: return 'Less 500'
    elif 500 <= val < 600: return 'Less 600'
   elif 600 <= val < 800: return 'Less 800'
    else:              return 'Something else'
```

9.

```
def jump_to(val):
    if    0 <= val < 600: return 'Less 600'
    elif 600 <= val < 700: return 'Less 700'
   elif 700 <= val < 800: return 'Less 800'
    else:              return 'Something else'
```

10.

```
def jump_to(val):
    if    0 <= val < 700: return 'Less 700'
    elif 700 <= val < 800: return 'Less 800'
   elif 800 <= val < 900: return 'Less 900'
    else:              return 'Something else'
```

Задание к лабораторной работе по теме:

Методы структурного синтеза конечных автоматов. Программная реализация конечных автоматов.

Дан класс «Конечный автомат» и программа, использующая данный класс. Выполните обращение к данному классу со своим вариантом. Проанализируйте программный код на языке Python и результаты работы программы. Модифицируйте программный код таким образом, чтобы он мог корректно работать с Вашим вариантом.

```
class StateMachine:
   def __init__(self):
      self.handlers = {}
      self.startState = None
      self.endStates = []
```

```
    def add_state(self, name, handler, end_state=0):
        name = name.upper()
        self.handlers[name] = handler
        if end_state:
            self.endStates.append(name)

    def set_start(self, name):
        self.startState = name.upper()

    def run(self, cargo):
        try:
            handler = self.handlers[self.startState]
        except:
            raise InitializationError("must call .set_start() before .run()")
        if not self.endStates:
            raise  InitializationError("at least one state must be an end_state")

        while True:
            (newState, cargo) = handler(cargo)
            if newState.upper() in self.endStates:
                print("reached ", newState)
                break
            else:
                handler = self.handlers[newState.upper()]

from statemachine import StateMachine

positive_adjectives = ["great","super", "fun", "entertaining", "easy"]
negative_adjectives = ["boring", "difficult", "ugly", "bad"]

def start_transitions(txt):
    splitted_txt = txt.split(None,1)
    word, txt = splitted_txt if len(splitted_txt) > 1 else (txt,"")
    if word == "Python":
        newState = "Python_state"
    else:
        newState = "error_state"
    return (newState, txt)

def python_state_transitions(txt):
    splitted_txt = txt.split(None,1)
    word, txt = splitted_txt if len(splitted_txt) > 1 else (txt,"")
    if word == "is":
        newState = "is_state"
    else:
        newState = "error_state"
    return (newState, txt)

def is_state_transitions(txt):
    splitted_txt = txt.split(None,1)
    word, txt = splitted_txt if len(splitted_txt) > 1 else (txt,"")
    if word == "not":
        newState = "not_state"
```

```
    elif word in positive_adjectives:
        newState = "pos_state"
    elif word in negative_adjectives:
        newState = "neg_state"
    else:
        newState = "error_state"
    return (newState, txt)

def not_state_transitions(txt):
    splitted_txt = txt.split(None,1)
    word, txt = splitted_txt if len(splitted_txt) > 1 else (txt,"")
    if word in positive_adjectives:
        newState = "neg_state"
    elif word in negative_adjectives:
        newState = "pos_state"
    else:
        newState = "error_state"
    return (newState, txt)

def neg_state(txt):
    print("Hallo")
    return ("neg_state", "")

1.
if __name__== "__main__":
    m = StateMachine()
    m.add_state("Start", start_transitions)
    m.add_state("Java_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("Java is interesting")

2.
if __name__== "__main__":
    m = StateMachine()
    m.add_state("OOPS!", start_transitions)
    m.add_state("FORTRAN_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("FORTRAN is old")
```

3.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("Run!", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is not difficult")
    m.run("LISP is interesting")
```

4.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("Let us begin!", start_transitions)
    m.add_state("C#_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is not difficult")
    m.run("C# is useful")
```

5.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("New!", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("JCL is ugly")
```

6.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("Stand up!", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
```

```
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("MongoDB is strange")
```

7.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("New!", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("Java Script is useful")
```

8.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("GO ON!", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("PHP is beautiful!")
```

9.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("Let Us Start", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("JSON is very useful!")
```

10.

```
if __name__== "__main__":
    m = StateMachine()
    m.add_state("Go quickly!", start_transitions)
    m.add_state("Python_state", python_state_transitions)
    m.add_state("is_state", is_state_transitions)
    m.add_state("not_state", not_state_transitions)
    m.add_state("neg_state", None, end_state=1)
    m.add_state("pos_state", None, end_state=1)
    m.add_state("error_state", None, end_state=1)
    m.set_start("Start")
    m.run("Python is great")
    m.run("Python is difficult")
    m.run("PL/I  was too difficult!")
```

Задание к лабораторной работе по теме:
Недетерминированные автоматы.

Изобразить недетерминированный источник, соответствующий недетерминированному автомату, заданному таблицей переходов с входным алфавитом {a,b} и множеством внутренних состояний{(1,,2,3,4,5}.

1.

Q	1	2	3	4	5
A					
a	Ø	1	2,3,4	1	3,4
b	3,4	3,5	Ø	3,5	2,4

2.

Q	1	2	3	4	5
A					
a	1,5	1,2,3,4,5	Ø	1	3
b	Ø	2,4	2,3,4	Ø	4,5

3.

Q	1	2	3	4	5
A					
a	Ø	Ø	4	Ø	1,2,4
b	3,4,5	1,3	2,3,5	1,2	2,3,5

4.

Q	1	2	3	4	5
A					
a	1,2,3,4,5	2,3,4	Ø	Ø	1,3,5
b	4	3	1,5	2,4	4

5.

Q	1	2	3	4	5
A					
a	Ø	2,3	Ø	3,4	4,5
b	1,2,3	1,4,5	1,5	Ø	2,3

6.

Q	1	2	3	4	5
A					
a	1,2,3	Ø	3,4	1,2	Ø
b	3	2,3,4	Ø	2,5	3,4

7.

Q	1	2	3	4	5
A					
a	3,5	3,5	1,2,4	1,2,4	3,4
b	1,2,5	Ø	3,5	Ø	1,2

8.

Q	1	2	3	4	5
A					
a	1,2	Ø	3,5	3,4	1,2,3,5
b	2,4	1,3,5	Ø	Ø	1,2

9.

Q	1	2	3	4	5
A					
a	2,3,4,5	3,4	Ø	Ø	2,4,5
b	3	2,5	3	1,5	3

10.

Q	1	2	3	4	5
A					
a	Ø	2	1,3,4,5	5	Ø
b	1,3,4	3,4	2,5	Ø	2,4

Задание к лабораторной работе по теме:

Классификация формальных грамматик по Н. Хомскому. Примеры формальных грамматик и их применения

1.

Описать язык, порождаемый грамматикой:

F → aF H, F → aR, RH → bRc, cH → Hc, R → bc.

2.

Описать язык, порождаемый грамматикой:

F → aF H, F → aR, RH → bRG, GH → HG, R → bG, G → c.

3.

Описать язык, порождаемый грамматикой:

F → aF H, F → abc, bH → bbc, cH → Hc.

4.

писать язык, порождаемый грамматикой:

S → aT bc, T b → bT , T c → U bcc, bU → U b, aU → aaT ,

5.

Описать язык, порождаемый грамматикой:

S → aU a, T b → bT , T a → U baa, bU → U b, aU → aaT ,aU → ab.

6.
Описать язык, порождаемый грамматикой:
S → aSBC, S → abC, CB → BC, bB → bb, bC → bc, cC → cc.

7.
Описать язык, порождаемый грамматикой:
S → aSA, S → aT , T A → bT a, aA → Aa, T → ba.

8.
Описать язык, порождаемый грамматикой:
S → T aS, S → U, T a → aaT , T U → U, U → a.

9.
Описать язык, порождаемый грамматикой:
S → KT , T → aT B, T → ε, aB → Baa, KB → Ka, K → a.

10.
Описать язык, порождаемый грамматикой:
F → GJ, G → KGT , G → KT , KT → aT K, Ka → aK,
T a → aT , KJ → Ja, T J → Ca, T C → Ca, C → a.

Задание к лабораторной работе по теме:
Регулярные выражения

Задан класс, реализующий конечный автомат для разбора регулярных выражений. Воспользоваться этим классом для работы с текстами, содержащими HTML-разметку. Из представленных текстов надо извлечь содержательную информацию и информацию о форматировании страницы. Если в тексте нет содержательной информации, программа должна об этом сообщить пользователю. Программы для обращения к классу написать самостоятельно.

```
class Rule_Parse_FSM:

    def __init__(self, input_str):
        self.input_str = input_str
        self.current_state = S_NEW_GROUP
        self.group_current_level = 0
        self.current_group = RuleGroup(None, self.group_current_level, None)
        self.current_char = ''

    def run(self):
        for c in self.input_str:
            if not self.process_next(c):
                print("skip '{}' in {}".format(c, self.current_state))

    def process_next(self, achar):
        self.current_char = achar
        frozen_state = self.current_state
        for transition in FSM_MAP:
            if transition['src'] == frozen_state:
                if self.iterate_re_evaluators(achar, transition):
                    return True
        return False
```

```
            def iterate_re_evaluators(self, achar, transition):
                condition = transition['condition_re_compiled']
                if condition.match(achar):
                    self.update_state(
                        transition['dst'], transition['callback'])
                    return True
                return False

            def update_state(self, new_state, callback):
                print("{} -> {} : {}".format(self.current_char,
                                             self.current_state,
                                             new_state))
                self.current_state = new_state
                callback(self)
```

```
        1.
displayAlign: 'center',
       "HTML-CSS": {
          styles: {'.MathJax_Display': {"margin": 0}},
          linebreaks: { automatic: true },
          scale: 90
       }
```

```
        2.
MathJax.Hub.Config({
       tex2jax: {
          inlineMath: [ ['$','$'], ["\\(","\\)"] ],
          displayMath: [ ['$$','$$'], ["\\[","\\]"] ],
       }
```

```
        3.
<a  href="https://www.python-training-courses.com"><img  style="width:  150px;"  alt="Python
Training Courses"
                    src="images/toronto_building.jpg"><br><br>Python Training Courses in To-
ronto, Canada</a>
```

```
        4.
</div><div id="sidebar-b"><img style="width: 150px;" alt="Python-Logo"
                    src="images/python-logo.png"><br><br>
<!-- Dieses Tag dort einfügen, wo die +1-Schaltfläche dargestellt werden soll -->
```

```
        5.
<br>
<!-- Place this tag where you want the widget to render. -->
<a href="https://plus.google.com/+BerndKleinPy">Bernd Klein on Google</a>
<br><br>
```

```
        6.
input type="text" value=""  style="width: 130px;" id="googlebox"/>
<input type="button" value="Go" onclick="google()"/>
<br><br>
```

```
        7.
```

<Python Tutorial in Deutsch
<h3>Python 3</h3>This is a tutorial in Python3,
but this chapter of our course is available in a version for Python 2.x as well:
Python Online Course in Python 2.x<p>

8.

<h3>Skilled Python Programmers</h3>
You are looking for experienced Python developers or programmers? We can help you, please
contact us.

9.

<h3>Data Protection Declaration</h3>
Data Protection Declaration

10.

<h3>News</h3>

<p class="update">
April 2018:
A completely new chapter <a href="python_sets_example.php">

Задание к лабораторной работе по теме:
Программирование с явным выделением состояний.

Сформулируйте поставленную задачу в автоматной парадигме. Выделите состояния. Определите переходы из состояния в состояние.

1. Снежинка имеет форму правильного многоугольника. Падает вниз. Под воздействием ветра может лететь вверх или в сторону. Может слипаться с другими снежинками. Комок снежинок имеет форму шара. Снежинка может таять. Изобразить снегопад, метель оттепель.
2. Схематично изображенный самолет может стоять на стоянке, выруливать по рулежным дорожкам на взлетно-посадочную полосу, разгоняться и взлетать, лететь по прямой траектории, снижаться, совершать посадку и уходить по рулежным дорожкам с взлетно-посадочной полосы на стоянку.
3. Схематично изображенный вертолет может стоять на стоянке, вертикально взлетать, лететь по прямой траектории, к нему может быть прицеплен груз на внешней подвеске. Вертолет может перевозить груз на внешней подвеске, устанавливать груз на указанное место, приземляться.
4. Заготовка проходит между двумя вращающимися валками, при этом уменьшается ее высота (обжатие), увеличивается ширина (уширение) и длина. Прошедшую между валками заготовку поворачивают на 90° (кантовка) и направляют в обратном направлении. Процесс продолжается до тех пор, пока не будут достигнуты требуемые параметры заготовки.
5. Схематично изображенный грузовик может привозить груз к подъемному крану и отвозить груз от подъемного крана. Схематично изображенный подъемный кран разгружает и загружает грузовики.
6. Схематично изображенный надувной шар может быть сплющенным, его можно надувать (его радиус увеличивается), при излишнем надувании он может лопнуть, он может лопнуть, если его проткнут иголкой. Шары можно соединять в гирлянду, отделять от гирлянды.
7. Схематично изображенная ветряная мельница вращает лопасти. Если ветра нет, лопасти неподвижны. При очень сильном ветре лопасти отваливаются.

8. Схематично изображенный катер может стоять у пристани, в катер может быть помещен груз. Катер может отчаливать, двигаться по фарватеру, причаливать, проводить выгрузку.
9. Схематично изображенный трамвай может стоять в депо, ехать от остановки до остановки по прямой и с поворотами, останавливаться на остановках, разворачиваться на конечных остановках.
10. Схематично изображенный автомобиль может стоять у тротуара, двигаться по прямой, перестраиваться из ряда в ряд, поворачивать. При перестроениях и поворотах у автомобиля должен быть включен соответствующий световой указатель. При нарушении правил автомобиль может быть удален из транспортного потока. Светофор поочередно включает в каждом направлении красный, желтый и зеленый свет

Индивидуальные задания к разделу 1.

Для автомата, заданного таблицей, постройте диаграмму Мура. Задайте этот автомат системой булевых функций:

1.

q	0	1	2	3
x				
0	1;2	3;0	2;0	2;0
1	2;1	2;0	3;0	3;1

2.

q	0	1	2	3
x				
0	1;0	2;2	2;1	3;0
1	3;0	3;1	2;3	1;0

3.

q	0	1	2	3
x				
0	0;1	1;3	2;1	3;1
1	0;0	0;1	3;1	2;3

4.

q	0	1	2	3
x				
0	1;0	3;2	2;0	1;0
1	3;0	1;1	0;1	3;1

5.

q	0	1	2	3
x				
0	1;1	3;1	2;0	1;0
1	3;0	1;0	2;1	3;1

6.

q	0	1	2	3
x				
0	0;1	3;1	2;1	1;0

1	3;0	1;0	0;1	3;2

7.

q	0	1	2	3
x				
0	1;0	3;1	2;1	1;0
1	3;0	1;2	0;1	3;1

8.

q	0	1	2	3
x				
0	0;1	1;1	3;1	2;0
1	2;0	0;1	3;1	1;3

9.

q	0	1	2	3
x				
0	3;2	2;0	1;1	0;1
1	0;0	1;1	2;1	3;0

10.

q	0	1	2	3
x				
0	2;3	2;1	1;1	0;1
1	0;1	1;3	2;0	3;0

Индивидуальные задания к разделу 2.

Изобразить недетерминированный источник, соответствующий недетерминированному автомату, заданному таблицей переходов с входным алфавитом {a,b} и множеством внутренних состояний{(1,,2,3,4,5}.

1.

Q	1	2	3	4	5
A					
a	Ø	1	2,3,4	1	3,4
b	3,4	3,7	2,1	Ø	2,4

2.

Q	1	2	3	4	5
A					
a	Ø	1,2,3,4,5	Ø	1	3
b	1,3	2,4	2,3,4	Ø	4,5

3.

Q	1	2	3	4	5
A					
a	Ø	3,4,5	4	Ø	1,2,4

b	Ø	1,3	2,3,5	1,2	2,3,5

4.

Q	1	2	3	4	5
A					
a	1,2,3,4,5	2,3,4	4	Ø	1,3,5
b	Ø	3	1,5	2,4	4

5.

Q	1	2	3	4	5
A					
a	Ø	1,3	Ø	3,4	4,5
b	1,2,3	1,4,5	2,5	Ø	2,3

6.

Q	1	2	3	4	5
A					
a	1,2,4	Ø	3,4	1,2	Ø
b	3	2,3,4	Ø	2,5	3,5

7.

Q	1	2	3	4	5
A					
a	3,5	3,5	1,2,4	1,2,4	3,4
b	Ø	1,5	3,5	Ø	1,2

8.

Q	1	2	3	4	5
A					
a	1,3	Ø	3,5	3,4	1,2,3,5
b	2,4	1,3,5	Ø	2,5	1,2

9.

Q	1	2	3	4	5
A					
a	Ø	3,4	2,4,5	Ø	2,4,5
b	3	2,5	3	1,5	3

10.

Q	1	2	3	4	5
A					
a	Ø	2	1,3,5	5	Ø
b	1,3,4	3,5	2,5	Ø	2,4

Индивидуальные задания к разделу 3.

Задан класс, реализующий конечный автомат для разбора регулярных выражений. Воспользоваться этим классом для работы с текстами, содержащими HTML-разметку. Из представленных текстов надо извлечь содержательную информацию и информацию о форматировании страницы. Если в тексте нет содержательной информации, программа

должна об этом сообщить пользователю. Программы для обращения к классу написать самостоятельно.

```
class Rule_Parse_FSM:

    def __init__(self, input_str):
        self.input_str = input_str
        self.current_state = S_NEW_GROUP
        self.group_current_level = 0
        self.current_group = RuleGroup(None, self.group_current_level, None)
        self.current_char = ''

    def run(self):
        for c in self.input_str:
            if not self.process_next(c):
                print("skip '{}' in {}".format(c, self.current_state))

    def process_next(self, achar):
        self.current_char = achar
        frozen_state = self.current_state
        for transition in FSM_MAP:
            if transition['src'] == frozen_state:
                if self.iterate_re_evaluators(achar, transition):
                    return True
        return False

    def iterate_re_evaluators(self, achar, transition):
        condition = transition['condition_re_compiled']
        if condition.match(achar):
            self.update_state(
                transition['dst'], transition['callback'])
            return True
        return False

    def update_state(self, new_state, callback):
        print("{} -> {} : {}".format(self.current_char,
                                     self.current_state,
                                     new_state))
        self.current_state = new_state
        callback(self)
```

1.

br>
October 2017:
In our chapter on
Polynomials we demonstrate how easily and beautifully a class
for the creation and manipulation of polynomial functions can be written in Python.

2.

<January - March 2017:
We extended our chapters on
Generators and Decorators

3.

```
<b>May 2016:</b><br>
New chapter on <a href="python3_decorators.php">Decorators</a>.
In combination with our chapter on <a href="python3_memoization.php">
Memoisation and Decorators</a> it belongs to the most extensive treatisises
 on the topic of decoration à la Python!
<br><br>
```

4.

```
<ul>
<li><a href="python3_classes_and_type.php">Classes and Class Creation</a>
<li><a href="python3_road_to_metaclasses.php">On the Road to Metaclasses</a>
<li><a href="python3_metaclasses.php">Metaclasses</a>
<li><a href="python3_count_function_calls.php">Count Method Calls Using a Metaclass</a>
</ul>
```

5.

```
<b>August 2015:</b><br>
We added a chapter on <a href="python3_slots.php">Slots</a>
and another about the difference between <a href="python3_classes_and_type.php">type and classes</a>.
<br><br>
```

6.

```
<br><br>
<b>July 2014:</b><br>
An introduction into using database interfaces in Python for
<a href="sql_python.php">SQL, MySQL and SQLite</a>
<br><br>
```

7.

```
<br><br>
<b>March 2014:</b><br>
We are currently completely revising the chapter on object oriented
programming. It's more or less complete rewrite. The
<a href="python3_object_oriented_programming_old.php">old version dealing with OOP</a>
can still be accessed, though we recommend to work through the new ones.
<br>
```

8.

```
<div class="intro2">
<h3>Ads for training classes</h3>
<div class="textmenu">
This website is ad-free! There are no paid-for ads.
The only things advertised here are the book by Bernd Klein, the author of
this tutorial, and the training classes given by the author.
</div>
```

9.

```
<h3>Tutorial in hard copy</h3>
```

There is no PDF version available, but you can create it yourself.
Use "Print to File" and you will get a nicely formatted version of a chapter.

10.

The interaction between Python and the Linux Shell is another
topic of our advanced section. This chapter is followed by Forks and Forking.

You can learn more about threads and threading in our Introduction into Threads.
We show how to find the active IP addresses in a local network by using forks.

Индивидуальные задания к разделу 4.

Выполнить автоматное моделирование неформально поставленной задачи:

1. Младенец умеет: есть, пить, плакать, когда голоден, когда хочет пить, плакать, когда надо сменить пеленки, Кроме того, он может плакать из солидарности с другим младенцем (когда слышит плач более 3 минут), плакать просто так. Если младенец плачет более 5 минут, пеленки обязательно будут мокрые. Младенец начинает плакать в случайный момент времени. Мать умеет: кормить младенца, поить младенца, менять ему пеленки, безошибочно определять причину плача, успокаивать одновременно несколько младенцев. Метода стирки у нее нет, поэтому при нехватке пеленок ей придется использовать в этом качестве простыни, наволочки и все, что под руку попадется. Отец умеет: стирать и гладить пеленки. Делать он это может только 1 раз в сутки.
2. Кошка умеет: охотиться на мышей (вероятность поимки мыши — p), воровать у хозяина мясо (вероятность успеха — q), воровать у хозяина молоко (вероятность успеха — r). Мышь умеет: прятаться от кошки, прятаться от хозяина, воровать у хозяина муку (вероятность успеха — s), воровать у хозяина крупу (вероятность успеха — t). Хозяин умеет: отгонять кошку от еды, ловить мышь в капкан, покупать все перечисленные ранее продукты. В произвольные моменты времени хозяин отлучается из дому.
3. Крестьянин выращивает в поле зерно. Птицы поедают зерно (m граммов в день). Кроме того, они поедают жучков (n жучков в день). Жучки поедают зерно (s граммов в день). Если крестьянин уничтожит или прогонит всех птиц, жучки, не имея естественных врагов, будут беспрепятственно съедать зерно. Вероятность для крестьянина прогнать птицу — p, вероятность для птицы съесть жучка — q
4. Медицинская страховая компания работает с N клиентами. За каждого клиента она получает ежемесячно $w_i \left(i = 1,..,N\right)$ рублей. Вероятность заболеть в течение года для каждого клиента равна $p_i \left(i = 1,...,N\right)$. Стоимость его излечения является равномерно распределенной случайной величиной в диапазоне от 1000 до 100000 рублей.
5. Процессинговый центр обслуживает расчеты по дебетовым карточкам. Существует вероятность p попытки получения суммы S неправомерным путем. С вероятностью q эта попытка будет пресечена, а злоумышленник наказан, в противном случае процессинговый центр заплатит банку штраф в размере $2S$.
6. Системный администратор компьютерной сети умеет определять пользователям права доступа, изымать вредные записи, брать плату за работу в сети, лишать нарушителей прав доступа, брать штрафы за нарушения. Добросовестный пользователь может вно-

сить записи в отведенный ему раздел, стирать свои записи, платить за пользование, извлекать доходы или нести убытки из-за невостребованности информации или деятельности хакеров. За 1 килобайт пользователь платит K рублей, использование 1 килобайта информации приносит N рублей дохода, если информация окажется востребованной (вероятность — p). Хакер может все то, что и добросовестный пользователь. Кроме того, он может с вероятностью q определять чужие пароли и переадресовывать счета за работу в сети другим пользователям (как добросовестным, так и хакерам). В случае обнаружения недобросовестного поведения, он может быть подвергнут штрафу в размере десятикратно превышающем доход от недобросовестной деятельности.

7. Спамер рассылает по сети Internet недобросовестную рекламу товара (услуги) ценой в G рублей. С вероятностью p_i i-й пользователь купит рекламируемый товар (или услугу), q_i — не отреагирует на спам, с вероятностью r_i заставит спамера заплатить штраф, величина которого указывается далее. За каждую покупку спамер получает 1% комиссионных, при изобличении платит штраф, в 10 раз превышающий размер потенциальных комиссионных (то есть комиссионных, которые бы он получил при условии, что все адресаты купят рекламируемый товар).
8. Дистрибьютор высококачественной компьютерной техники работает с сетью дилеров. Он покупает у производителей продукцию n наименований по цене k_i за единицу и поставляет ее дилерам с наценкой в $p\%$. Добросовестные дилеры торгуют только фирменным товаром с оговоренной в соглашении с дистрибьютором наценкой в $q\%$. Недобросовестные дилеры могут поступать так же, как и добросовестные. Кроме того, они могут под видом фирменных продавать компьютеры низкого качества с наценкой $\boldsymbol{r\% > q\%}$. В случае обнаружения подлога недобросовестный дилер возмещает обманутым клиентам разницу в цене и платит штраф в размере утроенной неправомерно полученной суммы. Покупатель не имеет возможности отличить добросовестного дилера от недобросовестного. Количество денег у покупателя задается случайным образом. Решение о покупке принимается случайным образом при наличии у покупателя достаточной суммы свободных денег
9. В библиотеке имеются книги N наименований. Количество экземпляров каждой из этих книг $k_i\left(i = 1,..., N\right)$. В библиотеку записано M читателей. Каждый из читателей может одновременно держать у себя не более L книг. Книги выдаются на срок до 30 дней. Считаем, что читатель с одинаковой вероятностью может запросить любую из книг. Если запрошенной книги нет в наличии, читателя ставят в очередь на ее получение. Когда читатель возвращает книгу, она достается первому в очереди. Если читатель возвращает книгу после указанного срока, он лишается права получить книги в течение 60 дней. Длительность нахождения книги у читателя является случайной величиной.
10. Фирма распространяет лицензионное программное обеспечение и занимается его сопровождением. За каждую инсталляцию программного обеспечения фирма получает k рублей прибыли. В случае возникновения проблем с использованием установленного программного обеспечения фирма должна в течение 2 часов с момента получения заявки направить специалиста для решения проблемы. Вероятность возникновения в течение суток проблемы, требующей вмешательства специалиста, равна p. Специалист устраняет проблему за время, являющееся случайной величиной, равномерно распределенной в интервале $\left[0, T\right]$. В том случае, когда специалист опаздывает, фирма платит штраф в размере r рублей за каждый час опоздания. Каждый специалист получает s рублей в месяц

7 Оценочные средства для проведения промежуточной аттестации

а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
ПК-2. Обладает способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования.		
Знать	– принципы синтеза цифровых автоматов, основные понятия автоматного программирования; – способы программного задания цифровых автоматов; – общие методы структурного синтеза автоматов, принципы моделирования предметной области в автоматной парадигме.	Список теоретических вопросов: – Функционально полные системы элементарных логических функций. Канонические формы представления логических функций. СДНФ, СКНФ. – Задача анализа и синтеза логических функций. Минимизация функций алгебры логики. Этапы минимизации. – Минимизация логических функций методом Квайна. – Числовое и геометрическое представление функций алгебры логики. – Минимизация логических функций методом Квайна-мак-Класки. – Минимизация логических функций методом Карно. – Методика выполнения арифметических действий в D кодах. – Основные понятия и определения: абстрактные и структурные автоматы, конечные автоматы, полностью определенные и частичные автоматы, синхронные и асинхронные автоматы. – Автоматы Мили и Мура. Закон функционирования и способы задания автоматов Мура. – Автоматы Мили и Мура. Закон функционирования и способы задания автоматов Мили. – Эквивалентные автоматы. Преобразование автомата Мура в автомат Мили. – Эквивалентные автоматы. Преобразование автомата Мили в автомат Мура. – Совмещенная модель автомата (С автомат). Закон функционирования и способы задания С автоматов. – Последовательное соединение автоматов. Таблицы переходов и выходов результирующего автомата. Пример. – Параллельное соединение автоматов. Таблицы переходов и выходов результирующего автомата. Пример.

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
		– Соединение автоматов с обратной связью. Таблицы переходов и выходов результирующего автомата. Пример. – Задача структурного синтеза автоматов. Теорема о структурной полноте. – Кодирование состояний автомата и сложность комбинационных схем, реализующих функции выходов и возбуждения элементов памяти. – Принцип микропрограммного управления. Модель дискретного преобразователя Глушкова. – Функции операционного и управляющего автоматов. – Функциональная микропрограмма. Язык функционального микропрограммирования. – Классы микроопераций. Функциональная и структурная совместимость микроопераций. – Содержательный и закодированный графы микропрограммы. – Структурная организация операционных автоматов. Структурный базис. – Программные системы, управляемые автоматами. – Автоматы и алгоритмы дискретной математики. – Языковые средства автоматного программирования. – Синтез автоматной и объектно-ориентированной парадигм программирования. – Автоматы и UML. – Автоматы и регулярные выражения.
Уметь	– использовать методы синтеза цифровых автоматов, использовать методы проектирования автоматных программ; – строить распознаватели и преобразователи, сложные схемы взаимодействия автоматов; разрабатывать многокомпонентные недетерминированные системы.	Список практических заданий: – разработать в автоматной парадигме систему регулирования движения на перекрестке с учетом следующих факторов: интенсивность движения, состояние дорожного полотна, время суток, день недели, необходимость незамедлительно пропускать транспорт, оборудованный спецсигналами. Макет системы должен быть программно реализован. – разработать в автоматной парадигме систему диспетчеризации работы ремонтной бригады с учетом следующих факторов: состояние оборудования, интенсивность использования оборудования, система приоритетов при ремонте оборудования. Макет системы должен быть программно реализован.

Структурный элемент компетенции	Планируемые результаты обучения	Оценочные средства
Владеть	– навыками реализации автоматных моделей на языках программирования высокого уровня; – навыками проектирования и реализации сложных автоматных моделей на языках программирования высокого уровня; – навыками реализации недетерминированных моделей, сочетающих автоматную и объектно-ориентированную парадигму моделирования	Список комплексных заданий: – разработать в автоматной парадигме систему регулирования движения в небольшом районе. Система должна учитывать ситуацию на всех перекрестках; – разработать в автоматной парадигме систему диспетчеризации работы ремонтного предприятия, в котором имеется N бригад.

б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:

Промежуточная аттестация по дисциплине «Теория автоматов» включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме экзамена и в форме выполнения и защиты курсовой работы.

Экзамен по данной дисциплине проводится в устной форме по экзаменационным билетам, каждый из которых включает два теоретических вопроса и одно практическое задание.

Показатели и критерии оценивания экзамена:

– на оценку **«отлично»** (5 баллов) – обучающийся демонстрирует высокий уровень сформированности компетенций, всестороннее, систематическое и глубокое знание учебного материала, свободно выполняет практические задания, свободно оперирует знаниями, умениями, применяет их в ситуациях повышенной сложности.

– на оценку **«хорошо»** (4 балла) – обучающийся демонстрирует средний уровень сформированности компетенций: основные знания, умения освоены, но допускаются незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.

– на оценку **«удовлетворительно»** (3 балла) – обучающийся демонстрирует пороговый уровень сформированности компетенций: в ходе контрольных мероприятий допускаются ошибки, проявляется отсутствие отдельных знаний, умений, навыков, обучающийся испытывает значительные затруднения при оперировании знаниями и умениями при их переносе на новые ситуации.

– на оценку **«неудовлетворительно»** (2 балла) – обучающийся демонстрирует знания не более 20% теоретического материала, допускает существенные ошибки, не может показать интеллектуальные навыки решения простых задач.

– на оценку **«неудовлетворительно»** (1 балл) – обучающийся не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения простых задач.

8 Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) Основная литература:

1. Шелехов В.И. Язык и технология автоматного программирования. [Интернет-ресурс]. Режим доступа: http://is.ifmo.ru/works/2014/shelekhov-language-and-technology.pdf
2. Теория автоматов. [Электронный ресурс]. Режим доступа: https://www.hse.ru/ba/isct/courses/184761615.html

б) Дополнительная литература:

1. Поликарпова, Н.И., Шалыто А.А. Автоматное программирование [Текст]. / Н.И. Поликарпова, А.А. Шалыто. – СПб. : Питер, 2010. – 176 с.
2. Хопкрофт, Дж. Введение в теорию автоматов, языков и вычислений [Текст]. / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. – М.; СПб. ; Киев : Издательский дом «Вильямс», 2008.
3. Тишин В.В. Теория автоматов / В.В. Тишин. – СПб.: БХВ-Петербург, 2008. – 352 с.
4. Галушкина Ю.И Конспект лекций по дискретной математике / Ю.И. Галушкина, А.Н. Марьямов. М.: Айрис-пресс, 2007. – 176с.

в) Методические указания:

1. Ходашинский И.А. Теория автоматов и формальных языков./ Ходашинский И.А. Томск: ТУСУР, 2012. [Электронный ресурс]. Режим доступа: http://aoi.tusur.ru/upload/methodical_materials/Labor_1_4_file__693_4648.pdf

г) Программное обеспечение и Интернет-ресурсы:

Программное обеспечение: лицензионное программное обеспечение: операционная система; офисные программы; математические пакет, статистические пакеты, установленные на каждом персональном компьютере вычислительного центра ФГБОУ ВПО «МГТУ».

Программное обеспечение: лицензионное программное обеспечение: операционная система MS Windows 2007; MS Office 2010; PacketTracerre, установленные на каждом персональном компьютере вычислительного центра ФГБОУ ВПО «МГТУ».

Перечень лицензионного программного обеспечения по ссылке:
http://sps.vuz.magtu.ru/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2FShared%20Documents%2F%D0%9F%D0%BE%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0%20%D0%BA%20%D0%B0%D0%BA%D0%BA%D1%80%D0%B5%D0%B4%D0%B8%D1%82%D0%B0%D1%86%D0%B8%D0%B8%202020%2F%D0%A1%D0%B0%D0%BC%D0%BE%D0%BE%D0%B1%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%202019%D0%B3%2F%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5%20%D0%9F%D0%9E&InitialTabId=Ribbon.Document&VisibilityContext=WSSTabPersistence

Официальные сайты промышленных предприятий и организаций: http://www.mmk.ru , http://www.magtu.ru , и т.п.; разработчиков программных продуктов: http://www.statsoft.ru , http://www.microsoft.com , http://www.netacad.com и т.п.

9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

Тип и название аудитории	Оснащение аудитории
Лекционная аудитория	Мультимедийные средства хранения, передачи и представления информации
Компьютерный класс	Персональные компьютеры с установленными: пакетом MS Office; Visual Studio; Python 3 (Anaconda). Должен быть обеспечен доступ в Интернет, позволяющий использовать интерактивные программные продукты (интерактивные отладчики программного кода). Должен быть обеспечен доступ в электронную информационно-образовательную среду университета.
Аудитории для самостоятельной работы: компьютерные классы; читальные залы библиотеки	Все классы УИТ и АСУ с персональными компьютерами, выходом в Интернет и с доступом в электронную информационно-образовательную среду университета
Аудиторий для групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации	Ауд. 282 и классы УИТ и АСУ
Помещения для самостоятельной работы обучающихся, оснащенных компьютерной техникой с возможностью подключения к сети «Интернет» и наличием доступа в электронную информационно-образовательную среду организации	Классы УИТ и АСУ
Помещения для хранения и профилактического обслуживания учебного оборудования	Центр информационных технологий – ауд. 379