Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Системы искусственного интеллекта**

(название дисциплины)

**Лабораторная работа № 3**

**Вариант № 7**

Выполнил

Студент группы \_\_\_ИВТАПбд-41\_\_\_\_

\_\_\_\_\_\_\_\_\_\_Карпова А.С.\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

Проверил(а):

ст. преподаватель кафедры «ВТ»

(должность)

\_\_\_\_\_\_\_\_\_Хайруллин И.Д.\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

Ульяновск

2024

**Оглавление**

[Задание 3](#_Toc185886419)

[Ход работы 3](#_Toc185886420)

[Тестирование 5](#_Toc185886421)

[Вывод 5](#_Toc185886422)

[Приложение. Исходный код 6](#_Toc185886423)

# Задание

Необходимо разработать программу на языке python, которая реализует предложенное вариантом задание. Предметную область можно выбрать из предложенного списка, либо выбрать свою.

В рамках данной лабораторной работы необходимо разработать систему управления для автоматизации заданного технического объекта. Для хранения базы правил и онтологии предметной области предлагается использовать Neo4j, либо любое другое хранилище на выбор (реляционная СУБД, Protege). Задание предполагает проектирование логической модели управления, построение и настройку правил управления, а также разработку симулятора для проверки работы системы управления. Для формирования условий срабатывания правил необходимо использовать фаззификацию на основе нечеткой логики, для формирования управляющих инструкций - дефаззификацию.

Также необходимо разработать минимально рабочий симулятор предметной области с дискретным программным управлением.

Вариант:

Система автоматического орошения спортивных полей: разработать систему, которая контролирует полив в зависимости от погоды, влажности почвы и времени суток.

# Ход работы

#### 1. В начале работы создаётся база данных SQLite (irrigation\_system.db) с тремя таблицами:

* WeatherConditions — для хранения информации о погодных условиях (температура, влажность, наличие дождя).
* SoilMoisture — для хранения данных о влажности почвы.
* IrrigationSettings — для сохранения настроек полива (время суток и количество воды).

#### 2. Генерируются и сохраняются в базу данных случайные значения:

* **Температура** (от 0 до 35°C) и **влажность** (от 0% до 100%).
* **Состояние дождя** (идёт или не идёт дождь).
* **Уровень влажности почвы** (от 5 до 80%).

#### 3. Последние данные из таблиц WeatherConditions и SoilMoisture извлекаются для обработки. Входные параметры классифицируются на три уровня:

* **Влажность:** low, medium, или high.
* **Температура:** low, medium, или high.
* Если идёт дождь, значения по умолчанию устанавливаются как high (влажность) и low (температура).

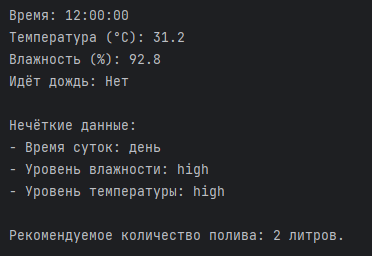
#### 4. Генерируется случайное время суток, которому соответствует одна из четырёх категорий: утро, день, вечер или ночь. Это время влияет на дальнейшее принятие решений о количестве полива.

#### 5. Используя нечёткие уровни влажности и температуры, а также категорию времени суток, определяется количество воды для полива:

* При низкой влажности и высокой температуре рекомендуется больше воды.
* При высокой влажности и низкой температуре полив уменьшается.
* Категория времени суток также учитывается: например, утром может требоваться больше воды, чем ночью.

#### 6. В процессе симуляции программа выводит на экран текущие параметры (рис. 1): температуру, влажность, состояние дождя и уровень влажности почвы. Отображает нечёткие данные, такие как уровень температуры, влажности и время суток. Показывает рекомендованное количество воды для полива (в литрах).

# Тестирование



*Рис 1. Вывод результата работы программы*

# Вывод

В данной лабораторной работе была разработана система управления, которая контролирует полив в зависимости от погоды, влажности почвы и времени суток.

# Приложение. Исходный код

|  |
| --- |
| import sqlite3 import random from datetime import datetime  # Функция для создания базы данных и таблиц def create\_database(db\_file):  conn = sqlite3.connect(db\_file)  cursor = conn.cursor()   cursor.execute('''  CREATE TABLE IF NOT EXISTS WeatherConditions (  id INTEGER PRIMARY KEY AUTOINCREMENT,  temperature REAL,  humidity REAL,  is\_raining BOOLEAN  );  ''')   cursor.execute('''  CREATE TABLE IF NOT EXISTS SoilMoisture (  id INTEGER PRIMARY KEY AUTOINCREMENT,  moisture\_level REAL  );  ''')   cursor.execute('''  CREATE TABLE IF NOT EXISTS IrrigationSettings (  id INTEGER PRIMARY KEY AUTOINCREMENT,  time\_of\_day TEXT,  irrigation\_amount REAL  );  ''')   conn.commit()  conn.close()  # Функция для генерации случайных данных и их добавления в базу данных def generate\_random\_data(db\_file):  conn = sqlite3.connect(db\_file)  cursor = conn.cursor()   # Случайные данные о погоде  temperature = round(random.uniform(0, 35), 1) # Температура от 10 до 35 градусов  humidity = round(random.uniform(0, 100), 1) # Влажность от 20% до 100%  is\_raining = random.choice([True, False]) # Идёт дождь или нет   cursor.execute('INSERT INTO WeatherConditions (temperature, humidity, is\_raining) VALUES (?, ?, ?)',  (temperature, humidity, is\_raining))   # Случайные данные о влажности почвы  moisture\_level = round(random.uniform(5, 80), 1) # Уровень влажности почвы от 5 до 80  cursor.execute('INSERT INTO SoilMoisture (moisture\_level) VALUES (?)', (moisture\_level,))   conn.commit()  conn.close()  # Функция для фаззификации входных данных def fuzzify(temperature, humidity, is\_raining):  if is\_raining:  return "high", "low" # Возвращаем значения по умолчанию при дожде   # Определяем уровень влажности  if humidity < 40:  humidity\_level = "low"  elif humidity < 70:  humidity\_level = "medium"  else:  humidity\_level = "high"   # Определяем уровень температуры  if temperature < 20:  temperature\_level = "low"  elif temperature < 30:  temperature\_level = "medium"  else:  temperature\_level = "high"   return humidity\_level, temperature\_level  # Функция для генерации случайного времени суток и его нечёткой категории def generate\_random\_time\_of\_day():  time\_categories = ["утро", "день", "вечер", "ночь"]  times = ["06:00:00", "12:00:00", "18:00:00", "00:00:00"]  index = random.randint(0, 3)  return times[index], time\_categories[index]  # Функция для дефаззификации с учётом времени суток def defuzzify(humidity\_level, temperature\_level, time\_of\_day\_category):  if time\_of\_day\_category == "утро":  if humidity\_level == "low" and temperature\_level == "high":  return 6 # Много полива утром  elif humidity\_level == "medium" and temperature\_level == "medium":  return 4 # Средний полив утром  elif humidity\_level == "high" and temperature\_level == "low":  return 2 # Мало полива утром  elif time\_of\_day\_category == "день":  if humidity\_level == "low" and temperature\_level == "high":  return 5 # Много полива днём  elif humidity\_level == "medium" and temperature\_level == "medium":  return 3 # Средний полив днём  elif humidity\_level == "high" and temperature\_level == "low":  return 1 # Мало полива днём  elif time\_of\_day\_category == "вечер":  if humidity\_level == "low" and temperature\_level == "high":  return 4 # Много полива вечером  elif humidity\_level == "medium" and temperature\_level == "medium":  return 3 # Средний полив вечером  elif humidity\_level == "high" and temperature\_level == "low":  return 2 # Мало полива вечером  else: # ночь  if humidity\_level == "low" and temperature\_level == "high":  return 3 # Много полива ночью  elif humidity\_level == "medium" and temperature\_level == "medium":  return 2 # Средний полив ночью  elif humidity\_level == "high" and temperature\_level == "low":  return 1 # Мало полива ночью   return 2 # Нормальное количество полива по умолчанию  # Симуляция процесса управления поливом def simulate\_irrigation(conn):  cursor = conn.cursor()   # Получаем последние данные о погоде и влажности почвы  cursor.execute("SELECT \* FROM WeatherConditions ORDER BY id DESC LIMIT 1")  weather\_conditions = cursor.fetchone()   cursor.execute("SELECT \* FROM SoilMoisture ORDER BY id DESC LIMIT 1")  soil\_moisture = cursor.fetchone()   # Фаззификация входных данных  humidity\_level, temperature\_level = fuzzify(weather\_conditions[1], weather\_conditions[2], weather\_conditions[3])   # Случайное время суток и его нечёткая категория  random\_time, time\_of\_day\_category = generate\_random\_time\_of\_day()   # Дефаззификация для получения количества полива с учётом времени суток  irrigation\_amount = defuzzify(humidity\_level, temperature\_level, time\_of\_day\_category)   # Вывод результата  print("\nВремя:", random\_time)  print("Температура (°C):", weather\_conditions[1])  print("Влажность (%):", weather\_conditions[2])  print("Идёт дождь:", "Да" if weather\_conditions[3] else "Нет")  print("\nНечёткие данные:")  print("- Время суток:", time\_of\_day\_category)  print("- Уровень влажности:", humidity\_level)  print("- Уровень температуры:", temperature\_level)  print(f"\nРекомендуемое количество полива: {irrigation\_amount} литров.")  # Основная функция для запуска всех операций def main():  database = "irrigation\_system.db"  create\_database(database)   generate\_random\_data(database)   connection = sqlite3.connect(database)  simulate\_irrigation(connection)  connection.close()  if \_\_name\_\_ == '\_\_main\_\_':  main() |