Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Кафедра вычислительной математики и механики

**Лабораторная работа № 2**

**«Проектирование и реализация оболочки экспертной**

**системы, использующей продукционный способ**

**представления знаний»**

Выполнила:

студентка группы ИСТ-19-2б

Семёнова А.С.

Проверил:

ассистент кафедры ВММБ

Нетбай Г.В.

Пермь, 2022

**Цель:** сформировать у студентов способность построения учебного прототипа экспертной системы, основанной на знаниях, включающего основные компоненты ИИС.

**Описание:** в рамках данной работы необходимо реализовать оболочку ЭС, которая была бы способна работать с продукционной БЗ. Работа позволит углубленно познакомиться со способами взаимодействия компонентов ЭС, а также реализовать отдельные функциональные блоки:

1. База знаний;
2. Механизм логического вывода;
3. Компонента объяснения.

Таким образом, будет создан минимальный прототип оболочки ЭС, позволяющий осуществить логический вывод. Вариативность задания может заключаться в различном способе хранения правил:

1. Хранение в формате XML.

2. Хранение в формате JSON.

3. Хранение в формате YAML.

4. Хранение в формате Protocol Buffers.

Последовательность выполнения:

1. Взять предметную область и БЗ из лабораторной работы №1;

2. В текстовом виде описать БЗ в выбранной нотации;

3. Спроектировать интерфейсы модулей реализуемой оболочки ЭС;

4. Начать разработку в системе хранения контроля версий.

**Требования:**

1. Должно быть использовано несколько стратегий разрешения конфликтов;
2. Каждая компонента оболочки ЭС должна быть представлена в виде отдельного модуля системы;
3. Программный код должен соответствовать требованиям:

* код должен быть единообразно отформатирован;
* должны присутствовать комментарии.

1. Компонента объяснения не должна быть сильно связана с МЛВ, т.е. должна присутствовать так называемая «Рабочая память».

В результате выполнения лабораторной работы необходимо, чтобы были получены следующие результаты:

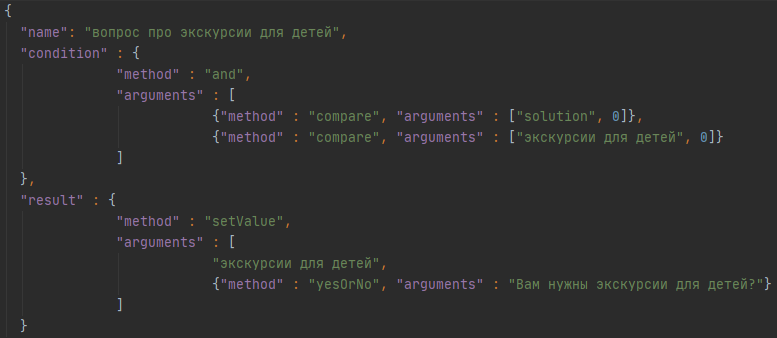
* программный код, хранящийся в системе контроля версий.

**Задача**

Задача лабораторной работы №2 заключается в написании программы, выполняющей роль продукционной системы. Был выбран язык Python. Программа должна выполнять действия в соответствии с правилами БЗ, сохранёнными в отдельный файл. Для хранения БЗ был выбран формат JSON.

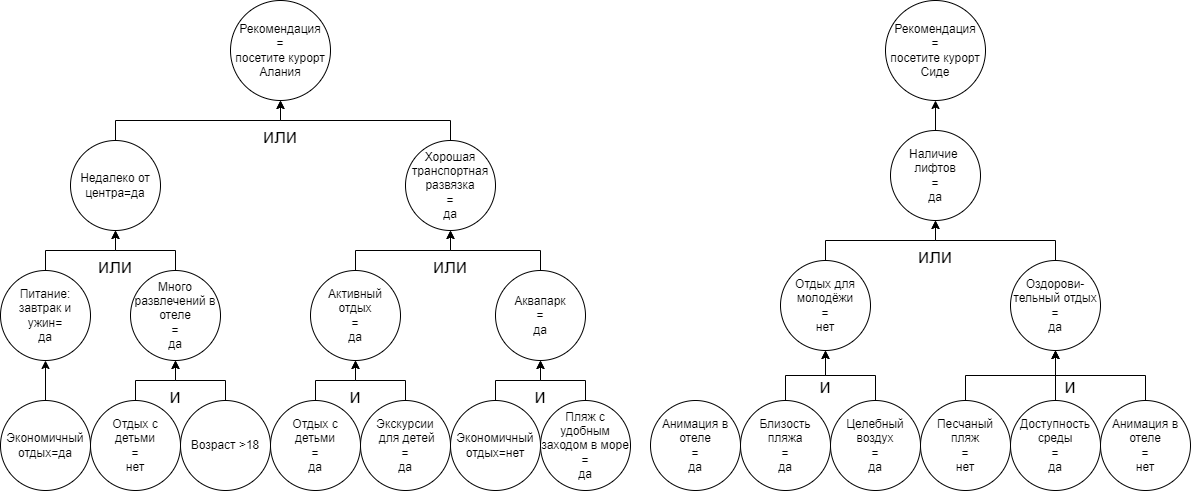
**Устройство БЗ в JSON**

В JSON хранится список ([…]) правил ({…}). Рассмотрим одно из правил:

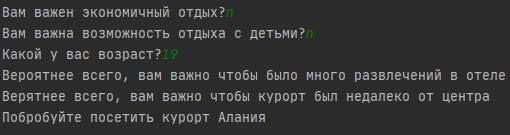


* Ключ “name”, соответствует названию правила.
* Ключ “condition”, объявляет блок с условиями (в том числе вложенные условия).
* Ключ “result”, объявляет блок действий при выполнении условия из “condition”.
* Ключ “method”, соответствует названию метода, который нужно использовать.
* Ключ “arguments”, соответствует параметрам, необходимым для выполнения “method.”

**Визуально оформленный граф**

****

**Пример запуска программы**

****

**Код оболочки на языке Python**

import json  
  
  
def And(\*params):  
 return all(params)  
  
  
def AskValue(question):  
 while True:  
 answer = input(question)  
 if answer.isdigit():  
 break  
 return answer  
  
  
def Biggest(param, value):  
 if not (param in variables):  
 variables[param] = 0  
 return float(variables[param]) > float(value)  
  
  
def Compare(param, value):  
 if not (param in variables):  
 variables[param] = 0  
 return variables[param] == value  
  
  
def Or(\*params):  
 return any(params)  
  
  
def Print(text):  
 print(text)  
  
  
def SetValue(param, value):  
 variables[param] = value  
 return variables[param]  
  
  
def YesOrNo(question):  
 answer = ""  
  
 while answer != "yes" and answer != "no":  
 answer = input(question)  
  
 if answer == "y":  
 answer = "yes"  
 if answer == "n":  
 answer = "no"  
 return answer  
  
  
methods = {"and": And, "askValue": AskValue, "biggest": Biggest, "compare": Compare, "or": Or, "print": Print, "setValue": SetValue, "yesOrNo": YesOrNo}  
variables = {"solution": 0}  
  
  
with open("rules.json", "r", encoding="utf-8") as rules:  
 data = json.load(rules)  
  
  
def ruleManager(part):  
 total\_arguments = []  
 if isinstance(part["arguments"], str):  
 total\_arguments.append(part["arguments"])  
 else:  
 for arg in part["arguments"]:  
 if not (isinstance(arg, dict)):  
 total\_arguments.append(arg)  
 else:  
 total\_arguments.append(ruleManager(arg))  
 return methods.get(part["method"])(\*total\_arguments)  
  
  
def ruleHandler():  
 i = len(data) - 1  
 skip = []  
 while variables["solution"] == 0 and i > 0:  
 if i not in skip:  
 if ruleManager(data[i]["condition"]):  
 ruleManager(data[i]["result"])  
 skip.append(i)  
 i = len(data)  
 i -= 1  
  
  
ruleHandler()