МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт компьютерных наук
Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №1

По предмету: “СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА”

Разработка экспертной системы с использованием прямой цепочки рассуждений

Студент АС-21-1 _____ Станиславчук С.М.

(подпись, дата)

Руководитель

Профессор _____ Сараев П.В.

(подпись, дата)

Липецк, 2024 г.

Цель работы

Получение навыков проектирования и разработки экспертной системы на всех этапах ее создания.

Задача: разработать экспертную систему, которая поможет начинающему разработчику выбрать подходящий язык программирования для его проекта. Система должна учитывать различные факторы, такие как тип проекта, требования к производительности, простота использования, поддержка сообщества и другие критерии.

Задание кафедры

1. Разработайте оболочку экспертной системы, реализующую прямую цепочку рассуждений, на любом языке программирования.
2. Правила должны формироваться в следующем виде:

ЕСЛИ Объект1=Значение1 ТО Объект2=Значение2

3. Допускается использование логической связки "И".

Например,

ЕСЛИ объект1=значение1 И объект2=значение2 ТО объект3=значение3

4. База правил должна быть отделена от программного модуля и представлять собой отдельный текстовый файл.
5. Должна быть реализована возможность редактирования базы правил (изменения, дополнения, удаления правил).
6. В программе должны быть предусмотрены средства для отображения состояния рабочей базы данных.
7. Задайте стартовую ситуацию – начальное состояние рабочей базы данных.
8. Если ни одно из правил не может сработать, программа должна запрашивать у пользователя новые сведения о некотором объекте. Если таких сведений нет, программа завершает работу.
9. Выберите предметную область, которая знакома вам (разработка программного обеспечения, образовательный процесс, игра и т.д.). Составьте базу правил для этой области. Протестируйте работу вашей программы на этой базе правил.

Экспертная система на основе данной стартовой ситуации и своей базы правил должна сделать определенный вывод и отобразить его на экране.

P.S.

Допускается использование пакетов языка Python для составления базы правил и проведения логического выводов.

Срок сдачи:
30.09.2024 г.

Материалы для сдачи работы:

1. Пояснительная записка

База правил:

ЕСЛИ область_разработки=веб-разработка И совместимость=низкая_важность ТО приоритет=JavaScript

ЕСЛИ скорость=важна И время_разработки=не_важно ТО приоритет=C++

ЕСЛИ скорость=важна И область_разработки=драйверы ТО приоритет=C

ЕСЛИ скорость=не_важна И область_разработки=математика ТО приоритет=Python

ЕСЛИ область_разработки=мобильные_приложения И совместимость=высокая_важность ТО приоритет=Kotlin

ЕСЛИ скорость=важна И область_разработки=игры ТО приоритет=C#

ЕСЛИ время_разработки=важно И область_разработки=научные_исследования ТО приоритет=R

ЕСЛИ совместимость=низкая_важность И область_разработки=встраиваемые_системы ТО приоритет=Rust

ЕСЛИ скорость=не_важна И область_разработки=базы_данных ТО приоритет=SQL И удалить=совместимость

ЕСЛИ область_разработки=десктопные_приложения И совместимость=высокая_важность ТО приоритет=Java

ЕСЛИ скорость=важна И область_разработки=обработка_данных ТО приоритет=Go

ЕСЛИ время_разработки=не_важно И область_разработки=автоматизация ТО приоритет=Shell

ЕСЛИ совместимость=низкая_важность И область_разработки=искусственный_интеллект ТО приоритет=Julia

ЕСЛИ скорость=важна И область_разработки=веб-разработка ТО приоритет=TypeScript

ЕСЛИ скорость=не_важна И область_разработки=разработка_программного_обеспечения ТО приоритет=PHP

ЕСЛИ время_разработки=важно И область_разработки=облачные_технологии ТО приоритет=Scala

Программная реализация

```
import subprocess

from rich.columns import Columns
from rich.console import Console
from rich.panel import Panel
from rich.prompt import Prompt
from rich.table import Table


# Functions for working with rules and facts
def load_rules(file_path):
    rules = []
    with open(file_path, "r", encoding="utf-8") as file:
        for line in file:
            if line.strip():
                condition, result = line.strip().split(" ТО ")
                conditions = condition.replace("ЕСЛИ ", "").split(" И ")
                result_actions = result.split(" И ")
                rules.append((conditions, result_actions))
    return rules


def load_start_values(file_path):
    facts = {}
    with open(file_path, "r", encoding="utf-8") as file:
        for line in file:
            key, value = line.strip().split(": ")
            facts[key] = value
    return facts


def check_conditions(conditions, facts):
    for condition in conditions:
        obj, val = condition.split("=")
        if facts.get(obj, "").lower() != val.lower():
            return False
    return True


# Inference engine
def inference_engine(rules, facts):
    applied_rules = []
    applicable = False
```

```
    for conditions, result_actions in rules:
        if check_conditions(conditions, facts):
            for action in result_actions:
                if action.startswith("удалить="):
                    obj_to_remove = action.split("=")[1]
                    if obj_to_remove in facts:
                        del facts[obj_to_remove]
                        applied_rules.append(f"{conditions} ->
удалить={obj_to_remove}")
                else:
                    result_obj, result_val = action.split("=")
                    if facts.get(result_obj) != result_val:
                        facts[result_obj] = result_val
                        applied_rules.append(
                            f"{conditions} -> {result_obj} = {result_val}"
                        )
            applicable = True
            break  # Execute only one condition at a time
    return facts, applied_rules, applicable


# Display facts table
def display_facts(console, facts):
    table = Table(title="Facts")
    table.add_column("Object", justify="left")
    table.add_column("Value", justify="left")
    for obj, val in facts.items():
        table.add_row(obj, val)
    return table


# Display rules
def display_rules(console, applied_rules):
    table = Table(title="Applied Rules")
    table.add_column("Rule", justify="center")
    for rule in applied_rules:
        table.add_row(rule)
    return table


# Display help instructions
def display_help(console):
    help_table = Table(title="Help Instructions")
    help_table.add_column("Action", justify="left")
    help_table.add_column("Description", justify="left")
```

```
    help_table.add_row("1", "Run Inference")
    help_table.add_row("2", "Add Fact")
    help_table.add_row("3", "Edit Fact")
    help_table.add_row("4", "Remove Fact")
    help_table.add_row("5", "Clear Facts")
    help_table.add_row("6", "Open Rules Editor")
    help_table.add_row("7", "Exit")
    return help_table


# Clear facts and rules
def clear_facts(facts):
    facts.clear()


# TUI interface
def run():
    console = Console()
    rules_file = "bz.txt"
    rules = load_rules(rules_file)
    facts = load_start_values("start_val.txt")

    while True:
        console.clear()
        console.print(Panel("Expert System TUI"), justify="center")

        facts_table = display_facts(console, facts)
        help_table = display_help(console)

        columns = Columns([facts_table, help_table])
        console.print(columns)

        action = Prompt.ask(
            "Choose an action: ",
            choices=["1", "2", "3", "4", "5", "6", "7"],
        )

        if action == "1":
            facts, applied_rules, applicable = inference_engine(rules,
facts)
            console.print(facts_table)
            console.print(display_rules(console, applied_rules))
            if not applicable:
                console.print(
```

```
                    "No applicable rules. Please add new facts.",
style="bold red"
                )

        elif action == "2":
            obj = Prompt.ask("Enter fact name")
            val = Prompt.ask("Enter fact value")
            if obj and val:
                facts[obj] = val
            else:
                console.print(
                    "Error: Both object and value must be provided.",
style="bold red"
                )

        elif action == "3":
            obj = Prompt.ask("Enter fact name to edit")
            if obj in facts:
                val = Prompt.ask("Enter new fact value")
                facts[obj] = val
            else:
                console.print("Error: Fact not found.", style="bold red")

        elif action == "4":
            obj = Prompt.ask("Enter fact name to remove")
            if obj in facts:
                del facts[obj]
            else:
                console.print("Error: Fact not found.", style="bold red")

        elif action == "5":
            clear_facts(facts)

        elif action == "6":
            try:
                subprocess.call(["kitty -e nvim bz.txt"], shell=True)
            except FileNotFoundError:
                console.print("Error: rules.py file not found.",
style="bold red")

        elif action == "7":
            break

if __name__ == "__main__":
    run()
```

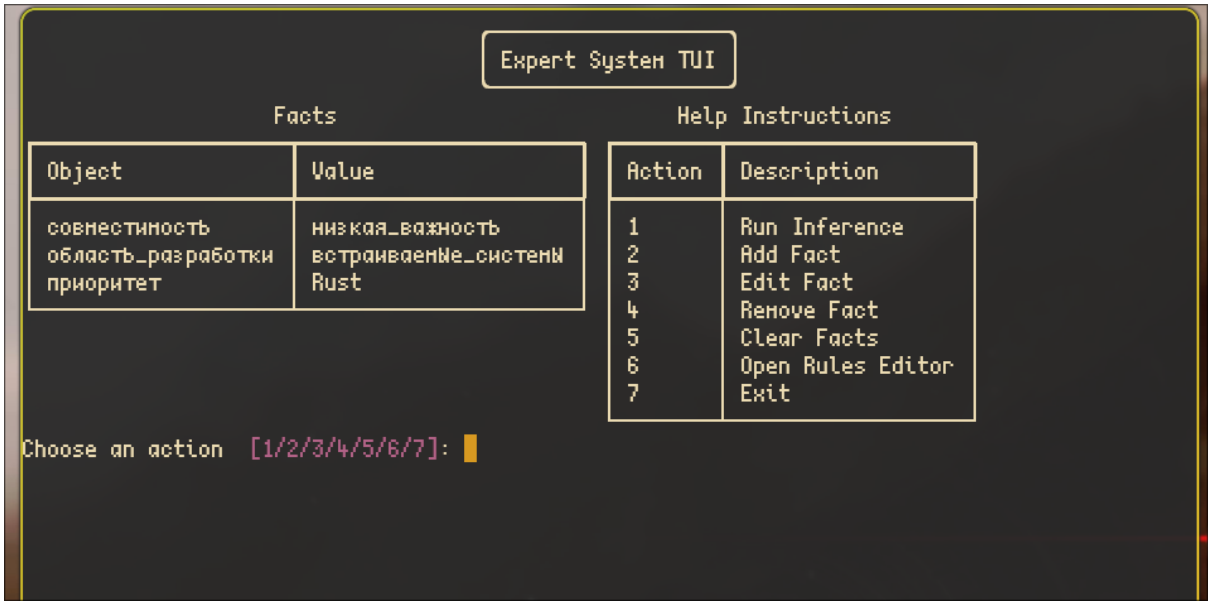

Рисунок 1 — Пример выполнеия программы

```
ЕСЛИ область_разработки=веб-разработка И совместимость=низкая_важность ТО приоритет=JavaScript
ЕСЛИ скорость=важна И время_разработки=не_важно ТО приоритет=C++
ЕСЛИ скорость=важна И область_разработки=встраиваемые_системы ТО приоритет=C
ЕСЛИ скорость=не_важна И область_разработки=математика ТО приоритет=Python
ЕСЛИ область_разработки=мобильные_приложения И совместимость=высокая_важность ТО приоритет=Kotlin
ЕСЛИ скорость=важна И область_разработки=игры ТО приоритет=C#
ЕСЛИ время_разработки=важно И область_разработки=научные_исследования ТО приоритет=R
ЕСЛИ совместимость=низкая_важность И область_разработки=встраиваемые_системы ТО приоритет=Rust
ЕСЛИ скорость=не_важна И область_разработки=базы_данных ТО приоритет=SQL И удалить=совместимость
ЕСЛИ область_разработки=десктопные_приложения И совместимость=высокая_важность ТО приоритет=Java
ЕСЛИ скорость=важна И область_разработки=обработка_данных ТО приоритет=Go
ЕСЛИ время_разработки=не_важно И область_разработки=автоматизация ТО приоритет=Shell
ЕСЛИ совместимость=низкая_важность И область_разработки=искусственный_интеллект ТО приоритет=Julia
ЕСЛИ скорость=важна И область_разработки=веб-разработка ТО приоритет=TypeScript
ЕСЛИ скорость=не_важна И область_разработки=разработка_программного_обеспечения ТО приоритет=PHP
ЕСЛИ время_разработки=важно И область_разработки=облачные_технологии ТО приоритет=Scala
```

Рисунок 2 — База правил в отдельном текстовом файле .txt с приоритетами

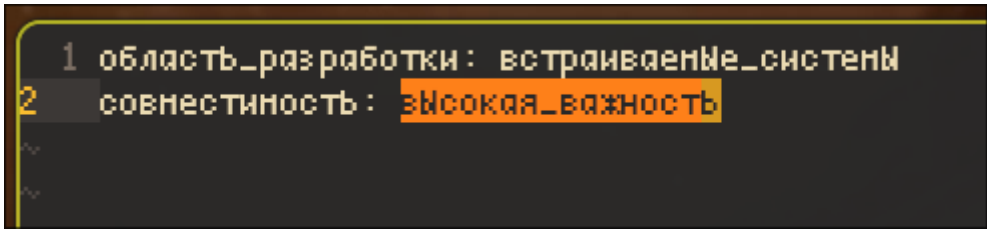

Рисунок 3 — Начальное состояние фактов

Вывод:

В ходе выполнения лабораторной работы получил навыки проектирования и разработки экспертной системы на всех этапах ее создания.