Лабораторная работа 4

**Механизм логического вывода в**

**продукционных интеллектуальных системах**

**Цель работы:** получить навыки в проектировании и программировании интерпретатора продукционных правил, работающего по принципу прямого или обратного логического вывода, а также научиться применять эффективные стратегии поиска решения в древовидном пространстве состояний.

**Предметная область**: 19. Поиск неисправностей легкового автомобиля.

**Задание к лабораторной работе**

Разработать машину логического вывода, реализующую заданный метод вывода.

Метод вывода – прямой.

**Описание стратегии управления логическим выводом и её применение к поиску решения в своей продукционной системе.**

Цепочка прямых рассуждений включает следующие типовые шаги:

• МЛВ принимает (вводит) одну или несколько исходных ситуаций (исходные

факты);

• для каждой ситуации МЛВ ищет в БП продукционные правила, условная часть которых содержит эту ситуацию;

• выбранные правила могут генерировать новые ситуации, которые добавляются к уже существующим;

• МЛВ обрабатывает каждую вновь сгенерированную ситуацию. При наличии хотя бы одной такой ситуации выполняются действия, начиная с пункта 2.

Рассуждения заканчиваются, когда нет необработанных ситуаций.

**Основные структуры данных экспертной системы.**

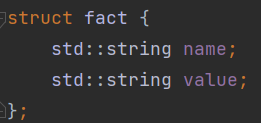
**Структура 1**. База фактов (БФ).

Хранит выведенные факты.

Например:

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Признак инициализации** | **Значение** |
| FROST | No |  |
| TEMP | In | 1 |
| WIND | No |  |
| SUN | No |  |
| PUBLIC | No |  |

1 факт представлен в виде структуры fact, которая имеет имя и значение.



База фактов представлена в виде списка фактов.



**Структура 2.** База правил (БП).

Хранит правила ЭС.

Например:

1. **Если** Мороз на улице = возрастает

**то** Число гуляющих = уменьшается.

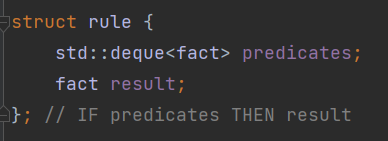
2. **Если** Мороз на улице = уменьшается

**то** Число гуляющих = возрастает.

3. **Если** Температура воздуха = уменьшается

**то** Мороз на улице = возрастает.

1 правило представлено в виде структуры rule, которая имеет предикат и результат. Предикат - это условная часть, которая реализована с помощью списка фактов. Результат - это заключение правила, который реализован с помощью факта.



База правил представлена в виде списка правил.



**Структура 3.** Очередь переменных (ОчП)

Она используется для организации обработки ситуаций, обозначаемых переменными, в порядке их возникновения.

В данной реализации алгоритма под очередь переменных явно не выделено структуры. Однако функция очереди переменных выполняется за счет особенностей реализации алгоритма. Также функция очереди переменных частично реализована в базе фактов.

**Алгоритм вывода на псевдокоде.**

**Начало**

Установить исходное состояние структур

Ввести исходную ситуацию (ИмП и ЗнП)

ИмП 🡪 ОчП

ЗнП 🡪 БФ

**Цикл** **пока** ОчП не пуста ?

Поиск в МасПУ переменной, имя

которой стоит в начале ОчП.

**Если** ИмП найдено в МасПУ ?

**то**

Номер найденного правила🡪УкПУ

Номер условия := 1

Номер условия 🡪 УкПУ

**Если** Среди ИмП условной части

найденного правила есть

неозначенные переменные?

**то** Запросить значения этих ИмП

Полученные ЗнП 🡪 БФ

Менять значение УкПУ

**всё**

**Если** Все условия найденного

правила истинны ?

**то**

ИмП из ТО-части правила🡪ОчП

ЗнП из ТО-части правила🡪 БФ

**всё**

**иначе**

Удалить ИмП из начала ОчП

**всё**

**Повторить**

**Конец.**

**Текст программы**

#include <iostream>

#include <string>

#include <deque>

#include <map>

#include <deque>

#include <optional>

#include <fstream>

struct fact {

std::string name;

std::string value;

};

struct rule {

std::deque<fact> predicates;

fact result;

}; // IF predicates THEN result

std::ostream& operator<<(std::ostream &out, const fact &fact\_) {

out << fact\_.name << " = " << std::boolalpha << fact\_.value;

return out;

}

std::ostream& operator<<(std::ostream &out, const std::deque<fact> &facts) {

for (fact f : facts) {

out << f << std::endl;

}

return out;

}

std::ostream& operator<<(std::ostream &out, const rule &rule\_) {

out << "IF ";

for (fact f : rule\_.predicates) {

out << f << "," << std::endl;

}

out << "THEN " << rule\_.result << std::endl;

return out;

}

std::ostream& operator<<(std::ostream &out, const std::deque<rule> &rule\_base) {

for (rule r : rule\_base) {

out << r << std::endl;

}

return out;

}

//bool getFactValue(const std::string& rule\_name, std::deque<fact>& facts) {

// for (fact& f : facts) {

// if (f.name == rule\_name) {

// return f;

// }

// }

// return {"Not found", false};

//}

bool operator==(const fact& f1, const fact& f2) {

if (f1.value == f2.value && f1.name == f2.name)

return true;

else

return false;

}

bool isContains(const std::deque<fact> &facts, const fact &given\_fact) {

for (const fact &f : facts) {

if (f == given\_fact) {

return true;

}

}

return false;

}

bool isContains(const std::deque<fact> &rule\_facts, const std::deque<fact> &fact\_base) {

if(rule\_facts.size() == 0) {

return false;

}

for (const fact &f : rule\_facts) {

if (!isContains(fact\_base, f)) {

return false;

}

}

return true;

}

bool isRightRule(const rule &r, const std::deque<fact> &fact\_base) {

return isContains(r.predicates, fact\_base);

}

void deleteRuleFromDeque(std::deque<rule> &rule\_base, std::string result\_name) {

std::erase\_if(rule\_base, [&result\_name](rule r) {

return r.result.name == result\_name;

});

}

void directLogic(std::deque<rule> &rule\_base, std::deque<fact> &fact\_base, const std::string &target\_name) {

bool end\_system{false};

while(!end\_system) { // Пока система не закончила работу

end\_system = true;

for (const auto &r: rule\_base) { // Пройти по всем правилам

if (isRightRule(r, fact\_base)) { // Если правило подходит под имеющийся набор фактов

std::cout << "Сработало правило: " << r;

if (target\_name == r.result.name) { // Если вывод правила совпадает с целью ЭС

std::cout << "Результат: " << r.result << std::endl << std::endl; // Вывести результат

return; // Конец

}

std::cout << "Промежуточный результат: " << r.result << std::endl << std::endl; // Вывод результата поддерева

fact\_base.push\_back(r.result); // Сохранение нового факта

deleteRuleFromDeque(rule\_base, r.result.name); // Удалить уже использованное поддерево из базы правил

end\_system = false; // Продолжить выполнение ЭС

}

}

}

std::cout << "Результат: не могу найти неисправность."; // Если ЭС закончилась, но не вывела результат

}

fact parseStringToFact(std::string str) {

std::string delimiter = " - ";

std::string first = str.substr(0, str.find(delimiter));

str.erase(0, str.find(delimiter) + delimiter.length());

std::string second = str;

// std::cout << first << std::endl << second << std::endl;

return {first, second};

}

void readRulesFromFile(std::deque<rule> &rule\_base, const std::string &file\_path) {

std::fstream in(file\_path);

std::string str;

std::string param;

rule new\_rule{};

while (!in.eof()) {

std::getline(in, str);

if (str == "если") {

param = "условие";

continue;

}

if (str == "то") {

param = "заключение";

continue;

}

if (str.empty()) {

rule\_base.push\_back(new\_rule);

new\_rule = {};

continue;

}

if (param == "условие") {

new\_rule.predicates.push\_back(parseStringToFact(str));

// std::cout << parseStringToFact(str) << std::endl;

}

if (param == "заключение") {

new\_rule.result = parseStringToFact(str);

// std::cout << parseStringToFact(str) << std::endl;

}

}

}

void readFactsFromFile(std::deque<fact> &fact\_base, const std::string &file\_path) {

std::fstream in(file\_path);

std::string str;

while (!in.eof()) {

std::getline(in, str);

fact\_base.push\_back(parseStringToFact(str));

}

}

void initRuleBase(std::deque<rule> &rule\_base, const std::string &path) {

readRulesFromFile(rule\_base, path);

}

void initFactBase(std::deque<fact> &fact\_base, const std::string &path) {

readFactsFromFile(fact\_base, path);

}

int main() {

system("chcp 65001");

std::deque<rule> rule\_base = {};

std::deque<fact> fact\_base = {};

initRuleBase(rule\_base, "../rules.txt"); // Чтение базы правил из файла

initFactBase(fact\_base, "../facts2.txt"); // Чтение безы фактов из файла

// std::cout << "База правил:" << std::endl << rule\_base << std::endl; // Вывод базы правил

std::cout << "База фактов:" << std::endl << fact\_base << std::endl; // Вывод базы фактов

std::string target\_name = "неисправность"; // Определение цели ЭС

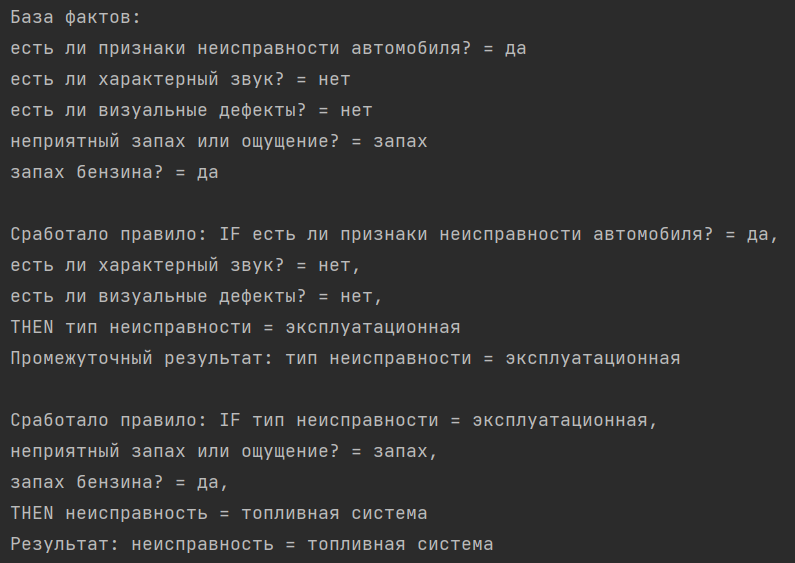
directLogic(rule\_base, fact\_base, target\_name); // Начало работы ЭС

return 0;

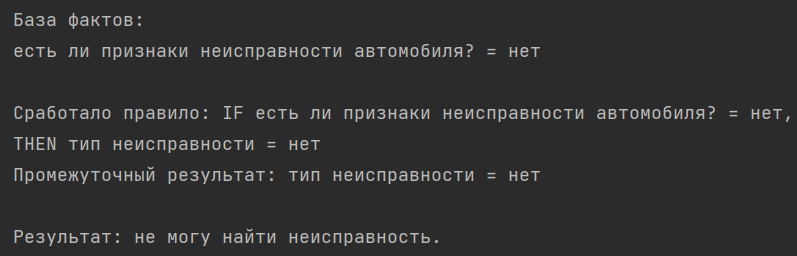
}

**Результаты выполнения**

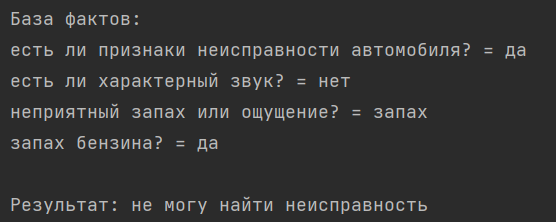
Пример 1:



Пример 2:



Пример 3:



**Содержимое файлов базы фактов и базы правил**

**База фактов для примера 1:**

есть ли признаки неисправности автомобиля? - да

есть ли характерный звук? - нет

есть ли визуальные дефекты? - нет

неприятный запах или ощущение? - запах

запах бензина? – да

**База фактов для примера 2:**

есть ли признаки неисправности автомобиля? - нет

**База фактов для примера 3:**

есть ли признаки неисправности автомобиля? - да

есть ли характерный звук? - нет

неприятный запах или ощущение? - запах

запах бензина? - да

**База правил**:

если

есть ли признаки неисправности автомобиля? - да

есть ли характерный звук? - да

то

тип неисправности - акустическая

если

есть ли признаки неисправности автомобиля? - да

есть ли характерный звук? - нет

есть ли визуальные дефекты? - да

то

тип неисправности - визуальная

если

есть ли признаки неисправности автомобиля? - да

есть ли характерный звук? - нет

есть ли визуальные дефекты? - нет

то

тип неисправности - эксплуатационная

если

есть ли признаки неисправности автомобиля? - нет

то

тип неисправности - нет

если

тип неисправности - акустическая

источник звука колеса? - да

тип звука - гул? - да

то

неисправность - ступичные подшипники

если

тип неисправности - акустическая

источник звука колеса? - да

тип звука - гул? - нет

тип звука - стук и скрип? - да

причина звука? - поворот руля

то

неисправность - наконечник рулевой тяги

если

тип неисправности - акустическая

источник звука колеса? - да

тип звука - гул? - нет

тип звука - стук и скрип? - да

причина звука? - езда по неровной поверхности

то

неисправность - амортизаторы

если

тип неисправности - акустическая

источник звука колеса? - да

тип звука - гул? - нет

тип звука - стук и скрип? - нет

то

неисправность - тормозные колодки

если

тип неисправности - акустическая

источник звука колеса? - нет

источник звука под капотом? - нет

то

неисправность - система выхлопа

если

тип неисправности - акустическая

источник звука колеса? - нет

источник звука под капотом? - да

тип звука - визг, гул и скрежет? - да

то

неисправность - коробка передач

если

тип неисправности - акустическая

источник звука колеса? - нет

источник звука под капотом? - да

тип звука - визг, гул и скрежет? - нет

тип звука - свист? - да

то

неисправность - ремень генератора

если

тип неисправности - акустическая

источник звука колеса? - нет

источник звука под капотом? - да

тип звука - визг, гул и скрежет? - нет

тип звука - свист? - нет

то

неисправность - водяная помпа системы охлаждения

дерево визуальных типов неисправности:

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - да

подтеки бензина? - да

подтеки рядом с топливным баком? - да

то

неисправность - топливный бак

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - да

подтеки бензина? - да

подтеки рядом с топливным баком? - нет

то

неисправность - система подачи топлива

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - да

подтеки бензина? - нет

подтеки тормозной жидкости? - да

спереди или сзади автомобиля? - спереди

то

неисправность - бак с тормозной жидкостью

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - да

подтеки бензина? - нет

подтеки тормозной жидкости? - да

спереди или сзади автомобиля? - сзади

то

неисправность - система подачи тормозной жидкости

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - да

подтеки бензина? - нет

подтеки тормозной жидкости? - нет

то

неисправность - емкость с моторным маслом

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - да

повреждение шины или колеса? - шины

то

неисправность - шина

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - да

повреждение шины или колеса? - колеса

то

неисправность - колесо

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - нет

есть ли неполадки со световыми приборами? - да

передние или задние? - передние

то

неисправность - передние световые приборы

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - нет

есть ли неполадки со световыми приборами? - да

передние или задние? - задние

то

неисправность - задние световые приборы

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - нет

есть ли неполадки со световыми приборами? - нет

есть ли электропитание? - да

работает ли световая индикация на приборной панели? - да

то

неисправность - электронные датчики

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - нет

есть ли неполадки со световыми приборами? - нет

есть ли электропитание? - да

работает ли световая индикация на приборной панели? - нет

то

неисправность - световая индикация приборной панели

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - нет

есть ли неполадки со световыми приборами? - нет

есть ли электропитание? - нет

есть ли повреждение проводки? - да

то

неисправность - электропроводка

если

тип неисправности - визуальная

есть ли подтеки технических жидкостей? - нет

есть ли повреждение шин или колес? - нет

есть ли неполадки со световыми приборами? - нет

есть ли электропитание? - нет

есть ли повреждение проводки? - нет

то

неисправность - аккумулятор

дерево эксплуатационных типов неисправности:

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - запах

запах бензина? - да

то

неисправность - топливная система

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - запах

запах бензина? - нет

запах горелого масла? - да

то

неисправность - система смазки

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - запах

запах бензина? - нет

запах горелого масла? - нет

сладковатый запах? - да

то

неисправность - бак с охлаждающей жидкостью

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - запах

запах бензина? - нет

запах горелого масла? - нет

сладковатый запах? - нет

запах гари? - да

то

неисправность - проводка

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - запах

запах бензина? - нет

запах горелого масла? - нет

сладковатый запах? - нет

запах гари? - нет

то

неисправность - катализатор

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - ощущение

рывки во время движения? - да

то

неисправность - система зажигания

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - ощущение

рывки во время движения? - нет

сильное проседание авто? - да

то

неисправность - ходовая система

если

тип неисправности - эксплуатационная

неприятный запах или ощущение? - ощущение

рывки во время движения? - нет

сильное проседание авто? - нет

то

неисправность - коробка передач