МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

**«Вятский государственный университет»**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**Отчет по лабораторной работе №1**

**по дисциплине**

**«Логическое программирование»**

Выполнил студент группы ИВТм-1301-01 \_\_\_\_\_\_\_\_\_\_\_\_/Ощепков Н.Д./

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Ростовцев В.С./

Киров 2022

1. Цель работы

* ознакомиться с принципами логического программирования на языке Пролог;
* научиться составлять программы на языке Пролог.

1. Задание на лабораторную работу

Вариант №32

Разработать логическую программу и найти решение задачи?

Нужно для 4 дежурных - Антипова, Климова, Маркова и Лебедева - составить график дежурств на агитпункте с соблюдением следующих условий:

1) Если Лебедев не будет дежурить в понедельник, то в понедельник согласен дежурить Климов.

2) Ecли Климов не сможет дежурить ни в понедельник, ни в четверг, то Антипов будет дежурить в понедельник.

3) Если Марков не сможет дежурить в четверг, то Климов будет дежурить в среду.

4) Если Лебедев придет дежурить во вторник, то Климов не будет дежурить в понедельник.

5) Если Антипов не сможет дежурить в Понедельник, то Марков не сможет дежурить во вторник.

Каким должен быть график дежурств?

1. Структура программ Пролога

Программа, написанная на Прологе, состоит из пяти разделов:

* Domains - <описание доменов>
* Database - <описания предикатов динамической базы данных>
* Predicates - <описания предикатов>
* Goal - <целевое утверждение>
* Clauses - <утверждения, факты, правила>

Назначение этих разделов таково:

* раздел domains содержит определения доменов, которые описывают различные классы объектов, используемых в программе;
* раздел database содержит утверждения базы данных, которые являются предикатами динамической базы данных. Если программа такой базы данных не требует, то этот раздел может быть опущен;
* раздел predicates служит для описания используемых программой предикатов;
* в разделе goal формулируется назначение создаваемой программы. Составными частями при этом могут являться некоторые подцели, из которых формируется единая цель программы;
* раздел clauses заносятся факты и правила. Факт представляется именем предиката, за которым следуют аргументы, заключенные в круглые скобки. Заканчивается запись факта точкой. Все имена предикатов и константы должны начинаться со строчной буквы. Правило состоит из заголовка правила и тела правила. Заголовок представляет собой предикат, тело состоит из термов, которые могут быть связаны между собой словами or или and. Между телом и заголовком стоит слово if. Каждое правило должно заканчиваться точкой. Имена переменных должны начинаться с прописной буквы и могут содержать только буквы, цифры или знак подчеркивания.

1. Механизм отката
2. Листинг программного кода

day\_of\_duty(monday).

day\_of\_duty(tuesday).

day\_of\_duty(wednesday).

day\_of\_duty(thursday).

% day\_of\_duty(friday).

generation\_hypothesis(Duty\_K, Duty\_M, Duty\_A, Duty\_L):-

day\_of\_duty(Day\_K),

day\_of\_duty(Day\_M),

\+ Day\_M = Day\_K,

day\_of\_duty(Day\_A),

\+ Day\_A = Day\_M,

\+ Day\_A = Day\_K,

day\_of\_duty(Day\_L),

\+ Day\_L = Day\_A,

\+ Day\_L = Day\_M,

\+ Day\_L = Day\_K,

Duty\_K = duty(klimov, Day\_K),

Duty\_M = duty(markov, Day\_M),

Duty\_A = duty(antipov, Day\_A),

Duty\_L = duty(lebedev, Day\_L).

include(Day\_duty, Duty\_K, Duty\_M, Duty\_A, Duty\_L):-

Day\_duty = Duty\_K;

Day\_duty = Duty\_M;

Day\_duty = Duty\_A;

Day\_duty = Duty\_L.

% правила:

% 1) Если Лебедев не будет дежурить в понедельник, то в понедельник согласен дежурить Климов.

check\_condition(1, Duty\_K, Duty\_M, Duty\_A, Duty\_L):-

\+ include(duty(lebedev, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L),

include(duty(klimov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

check\_condition(1, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

include(duty(lebedev, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

% 2) Ecли Климов не сможет дежурить ни в понедельник, ни в четверг, то Антипов будет дежурить в понедельник.

check\_condition(2, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

\+ include(duty(klimov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L),

\+ include(duty(klimov, thursday), Duty\_K, Duty\_M, Duty\_A, Duty\_L),

include(duty(antipov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

check\_condition(2, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

include(duty(klimov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

check\_condition(2, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

include(duty(klimov, thursday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

% 3) Если Марков не сможет дежурить в четверг, то Климов будет дежурить в среду.

check\_condition(3, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

\+ include(duty(markov, thursday), Duty\_K, Duty\_M, Duty\_A, Duty\_L),

include(duty(klimov, wednesday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

check\_condition(3, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

include(duty(markov, thursday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

% 4) Если Лебедев придет дежурить во вторник, то Климов не будет дежурить в понедельник.

check\_condition(4, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

include(duty(lebedev, tuesday), Duty\_K, Duty\_M, Duty\_A, Duty\_L),

\+ include(duty(klimov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

check\_condition(4, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

\+ include(duty(lebedev, tuesday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

% 5) Если Антипов не сможет дежурить в Понедельник, то Марков не сможет дежурить во вторник.

check\_condition(5, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

\+ include(duty(antipov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L),

\+ include(duty(markov, tuesday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

check\_condition(5, Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

include(duty(antipov, monday), Duty\_K, Duty\_M, Duty\_A, Duty\_L).

% проверка:

check\_hypothesis(Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

check\_condition(1, Duty\_K, Duty\_M, Duty\_A, Duty\_L),

check\_condition(2, Duty\_K, Duty\_M, Duty\_A, Duty\_L),

check\_condition(3, Duty\_K, Duty\_M, Duty\_A, Duty\_L),

check\_condition(4, Duty\_K, Duty\_M, Duty\_A, Duty\_L),

check\_condition(5, Duty\_K, Duty\_M, Duty\_A, Duty\_L).

search(Duty\_K, Duty\_M, Duty\_A, Duty\_L) :-

generation\_hypothesis(Duty\_K, Duty\_M, Duty\_A, Duty\_L),

check\_hypothesis(Duty\_K, Duty\_M, Duty\_A, Duty\_L),

write(' '). % так вывод понятнее, через строчку.

1. Результат работы

Duty\_K = duty(klimov, monday),

Duty\_M = duty(markov, thursday),

Duty\_A = duty(antipov, tuesday),

Duty\_L = duty(lebedev, wednesday) ;

false.

1. Вывод

В результате выполнения лабораторной работы получены навыки написания программ на языке Prolog. В качестве диалекта пролога использовался SWI Prolog (GNU Prolog) версия для Unix систем в адаптированном под Windows варианте. Среди существенных отличий данного диалекта от Турбо-пролога нужно отметить отсутствие деления на секции (DOMAINS, PREDICATES, GOAL и т.д.) и ряд отличий в синтаксисе, например таких, как замена операции отрицания равенства с <> на \=.

Данная задача была решена без использования списков, входные посылки писались в качестве фактов в порядке их следования. Для решения задачи был выбран метод гипотез. Генерировалось предположение, а именно кто и когда дежурит, далее данное предположение проверялось установленными правилами и либо принималось, либо отвергалось.

Программа, написанная на прологе, позволяет решать подобные логические задачи без алгоритма, простым перебором посылок. Это существенно может сэкономить время при разработке решения, когда набор входных данных составляет сотни и тысячи посылок.