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

* научиться составлять программы на языке Пролог с использованием списков.

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. Списки в языке Пролог

Связный список — структура данных, состоящая из узлов. Узел содержит данные и ссылку (указатель, связку) на один или два соседних узла. Списки языка Prolog являются односвязными, т.е. каждый узел содержит лишь одну ссылку.

В языке Prolog программист не сталкивается с явной работой с указателями в узлах, однако ему нужно иметь общее представление о списках, т.к. являясь основной структурой данных в функциональных и логических языках, они обладают рядом существенных отличий от массивов, используемых в императивных языках (таких как С++, Java, Pascal). В частности, элемент данных может быть очень быстро добавлен или удален из начала односвязного списка. Однако операция произвольного доступа (обращения к n-ному элементу) в списках выполняется гораздо дольше чем в массивах, т.к. требует n операций перехода по ссылкам.

При работе с односвязными списками необходимо выделять первый узел (называемый головой списка), остальные узлы (составляющие хвост списка) можно получить, передвигаясь по указателям вплоть до последнего узла. Хвост списка является таким же списком, как и исходный, поэтому обрабатывается аналогичным образом (рекурсивно).

Некоторые функции списков:

* sum\_list(List, Sum) — вычисление суммы элементов списка
* nth0(Index, List, Elem) — функция получения элемента списка с заданным индексом. Индексация начинается с нуля. Если нужного элемента нет — функция завершается неудачей.
* member(Elem, List) — выполняет поиск значения в списке. Завершается удачей если элемент найден.
* min\_list(List,MinElem) — вычисление наименьшего элемента списка
* reverse(List, ReverseList) — функция переворота списка
* sublist(Sub, List) — завершается удачей если все элементы списка Sub встречаются в списке List в точно таком же порядке.
* delete(InputList, Elem, ResultList) — функция удаления всех элементов с заданным значением из списка
* append(List1, List2, List1AndList2)— функция объединения двух списков
* unique(List) — проверка того, что ни один элемент списка не повторяется дважды
* rangConcat(List1, List2, List1AndList2) — объединение двух отсортированных списков так, чтобы в результате получился отсортированный список.
* length(List, Length) — вычисляет длину списка

1. Листинг программного кода

day\_of\_duty(monday).

day\_of\_duty(tuesday).

day\_of\_duty(wednesday).

day\_of\_duty(thursday).

day\_of\_duty(friday).

no\_rep([]).

no\_rep([First|Others]):-

\+ member(First, Others),

no\_rep(Others).

generation\_hypothesis(Hypothesis):-

day\_of\_duty(Day\_K),

day\_of\_duty(Day\_M),

day\_of\_duty(Day\_A),

day\_of\_duty(Day\_L),

no\_rep([Day\_K, Day\_M, Day\_A, Day\_L]), % Нужны только разные дни!

Hypothesis = [

duty(klimov, Day\_K),

duty(markov, Day\_M),

duty(antipov, Day\_A),

duty(lebedev, Day\_L)

].

% правила:

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

check\_condition(1, Hypothesis):-

\+ member(duty(lebedev, monday), Hypothesis),

member(duty(klimov, monday), Hypothesis).

check\_condition(1, Hypothesis) :-

member(duty(lebedev, monday), Hypothesis).

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

check\_condition(2, Hypothesis):-

\+ member(duty(klimov, monday), Hypothesis),

\+ member(duty(klimov, thursday), Hypothesis),

member(duty(antipov, monday), Hypothesis).

check\_condition(2, Hypothesis) :-

member(duty(klimov, monday), Hypothesis).

check\_condition(2, Hypothesis) :-

member(duty(klimov, thursday), Hypothesis).

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

check\_condition(3, Hypothesis):-

\+ member(duty(markov, thursday), Hypothesis),

member(duty(klimov, wednesday), Hypothesis).

check\_condition(3, Hypothesis) :-

member(duty(markov, thursday), Hypothesis).

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

check\_condition(4, Hypothesis):-

member(duty(lebedev, tuesday), Hypothesis),

\+ member(duty(klimov, monday), Hypothesis).

check\_condition(4, Hypothesis):-

\+ member(duty(lebedev, tuesday), Hypothesis).

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

check\_condition(5, Hypothesis):-

\+ member(duty(antipov, monday), Hypothesis),

\+ member(duty(markov, tuesday), Hypothesis).

check\_condition(5, Hypothesis) :-

member(duty(antipov, monday), Hypothesis).

% проверка:

check\_hypothesis(Hypothesis):-

check\_condition(1, Hypothesis),

check\_condition(2, Hypothesis),

check\_condition(3, Hypothesis),

check\_condition(4, Hypothesis),

check\_condition(5, Hypothesis).

search(Hypothesis):-

generation\_hypothesis(Hypothesis),

check\_hypothesis(Hypothesis).

1. Результат работы

D = [duty(klimov, monday), duty(markov, thursday), duty(antipov, tuesday), duty(lebedev, wednesday)];

false.

1. Вывод

В результате выполнения лабораторной работы получены навыки написания программ на языке Prolog с использованием списков. В качестве диалекта пролога использовался SWI Prolog.

В ходе выполнения работы были использованы связанные объекты (списки), над которыми выполнялись такие операции как:

* no\_rep – отвергались списки с повторными значениями;
* member – проверка, содержится ли указанный элемент в списке.