МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»  
(Самарский университет)

Лабораторная работа №3

по дисциплине «Логическое программирование»

**«Получение структурированной информации из базы данных»**

Вариант №2

Студент группы 6313-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.С.Балакин

Проверил ассистент кафедры   
программных систем Д.С. Оплачко

Дата \_\_\_. \_\_.2022

Оценка \_\_\_\_\_\_\_\_\_\_\_

Самара 2022

Содержание

[1. Задание 3](#_Toc99045548)

[2. Теоретический материал 3](#_Toc99045549)

[2.1 Основы логического программирования 3](#_Toc99045550)

[2.2 Использование дизъюнкции и отрицания 5](#_Toc99045551)

[2.3 Управление поиском решения 6](#_Toc99045552)

[2.4 Процедурность Пролога 6](#_Toc99045553)

[2.5 Использование списков в Прологе 7](#_Toc99045554)

[2.6 Использование составных термов 8](#_Toc99045555)

[3 Тип задачи 8](#_Toc99045556)

[4 Способы решения задачи 8](#_Toc99045557)

[5 Работа программы 8](#_Toc99045558)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 11](#_Toc99045559)

[ПРИЛОЖЕНИЕ А 12](#_Toc99045560)

# Задание

Предметная область – учебная группа факультета. Каждая учебная

группа может быть описана структурой: название факультета, код

специальности, номер группы, состав группы. Состав группы может быть описан списком структур, описывающих отдельного студента: фамилия, имя, отчество, обучение на военной кафедре, сводная ведомость. Сводная ведомость может быть описана списком из следующих структур: предмет, оценка.

Реализовать следующие типы запросов:

* Подсчитать общее число студентов на указанном факультете;
* Найти группу, у которой больше всего отличников;
* Найти все предметы, которые изучала определенная группа;
* Найти всех студентов группы, имеющих двойки;
* Найти все группы, изучавшие определенный предмет.

# Теоретический материал

# 2.1 Основы логического программирования

Язык программирования Пролог (PROgramming LOGic) предполагает получение решения задачи при помощи логического вывода из ранее известных фактов. Программа на языке Пролог не является последовательностью действий – она представляет собой набор фактов и правил, обеспечивающих получение логических заключений из данных фактов. Поэтому Пролог считается декларативным языком программирования. Пролог базируется на фразах (предложениях) Хорна, являющихся подмножеством формальной системы, называемой логикой предикатов. Пролог использует упрощенную версию синтаксиса логики предикатов, он прост для понимания и очень близок к естественному языку. Пролог имеет механизм вывода, который основан на сопоставлении образцов. С помощью подбора ответов на запросы Пролог извлекает хранящуюся информацию. Пролог пытается ответить на запрос, запрашивая информацию, о которой уже известно, что она истинна. Одной из важнейших особенностей Пролога является то, что он ищет не только ответ на поставленный вопрос, но и все возможные альтернативные решения. Вместо обычной работы программы на процедурном языке от начала и до конца, Пролог может возвращаться назад и просматривать все остальные пути при решении всех частей задачи. Программист на Прологе описывает объекты и отношения, а также правила, при которых эти отношения являются истинными. Объекты рассуждения в Прологе называются термами – синтаксическими объектами одной из следующих категорий:

• константы;

• переменные;

• функции (составные термы или структуры), состоящие из имени функции и списка аргументов-термов, имена функций начинаются со строчной буквы.

Константа в Прологе служит для обозначения имен собственных и начинается со строчной буквы. Переменная в Прологе служит для обозначения объекта на который нельзя сослаться по имени. Пролог не имеет оператора присваивания. Переменные в Прологе инициализируются при сопоствлении с константами в фактах и правилах. До инициализации переменная свободна, после присвоения ей значения она становится связанной. Переменная остается связанной только то время, которое необходимо для получения решения по запросу, затем Пролог освобождает ее и ищет другое решение. 5 Переменные в Прологе предназначены для установления соответствия между термами предикатов, действующих в пределах одной фразы (предложения), а не местом памяти для хранения данных. Переменная начинается с прописной буквы или знаков подчеркивания. В Прологе программист свободен в выборе имен констант, переменных, функций и предикатов. Исключения составляют резервированные имена и числовые константы. Переменные от констант отличаются первой буквой имени: у констант она строчная, у переменных – заглавная буква или символ подчеркивания. Область действия имени представляет собой часть программы, где это имя имеет один и тот же смысл:

• для переменной областью действия является предложение (факт, правило или цель), содержащее данную переменную;

• для остальных имен (констант, функций или предикатов) – вся программа.

Специальным знаком «\_» обозначается анонимная переменная, которая используется тогда, когда конкретное значение переменной не существенно для данного предложения. Анонимные переменные не отличаются от обычных при поиске соответствий, но не принимают значений и не появляются в ответах. Различные вхождения знака подчеркивания означают различные анонимные переменные. Отношения между объектами в Прологе называются фактами. Факт соответствует фразе Хорна, состящей из одного положительного литерала. Факт – это простейшая разновидность предложения Пролога. Любой факт имеет соответствующее значение истинности и определяет отношение между термами. Факт является простым предикатом, который записывается в виде функционального терма, состоящего из имени отношения и объектов, заключенных в круглые скобки, например: мать (мария, анна). отец (иван, анна). Точка, стоящая после предиката, указывает на то, что рассматриваемое выражение является фактом. Вторым типом предложений Пролога является вопрос или цель. Цель – это средство формулировки задачи, которую должна решать программа.

# 2.2 Использование дизъюнкции и отрицания

Чистый Пролог разрешает применять в правилах и целях только конъюнкцию, однако, язык, используемый на практике, допускает применение дизъюнкции и отрицания в телах правил и целях. Для достижения цели, содержащей дизъюнкцию, Пролог–система сначала пытается удовлетворить левую часть дизъюнкции, а если это не удается, то переходит к поиску решения для правой части дизъюнкции. Аналогичные действия производятся при выполнении тела правил, содержащих дизъюнкцию. Для обозначения дизъюнкции используется символ « ; ». В Прологе отрицание имеет имя «not» и для представления отрицания какого-либо предиката P используется запись not(P). Цель not(P) достижима тогда и только тогда, когда не удовлетворяется предикат (цель) P. При этом переменным значения не присваиваются. В самом деле, если достигается P, то не достигается not(P), значит надо стереть все 8 присваивания, приводящие к данному результату. Наоборот, если P не достигается, то переменные не принимают никаких значений.

# 2.3 Управление поиском решения

Встроенный в Пролог механизм поиска с возвратом может привести к поиску ненужных решений, в результате чего снижается эффективность программы в случае, если надо найти только одно решение. В других случаях бывает необходимо продолжить поиск, даже если решение найдено. Пролог обеспечивает два встроенных предиката, которые дают возможность управлять механизмом поиска с возвратом: предикат fail – используется для инициализации поиска с возвратом и предикат отсечения ! – используется для запрета возврата.

# 2.4 Процедурность Пролога

Пролог – декларативный язык. Описывая задачу в терминах фактов и правил, программист предоставляет Прологу самому искать способ решения. В процедурных языках программист должен сам писать процедуры и функции, которые подробно «объясняют» компьютеру, какие шаги надо сделать для решения задачи. Тем не менее, рассмотрим Пролог с точки зрения процедурного программирования:

1. Факты и правила можно рассматривать как определения процедур.
2. Использование правил для условного ветвления программы. Правило, в отличие от процедуры, позволяет задавать множество альтернативных определений одной и той же процедуры. Поэтому, правило можно считать аналогом оператора case в Паскале.
3. В правиле может быть выполнено сравнение, как в условных операторах.
4. Отсечение можно считать аналогом go to.
5. Возврат вычисленного значения производится аналогично процедурам. В Прологе это делается путем связывания свободных переменных при сопоставлении цели с фактами и правилами.

# 2.5 Использование списков в Прологе

Список – это упорядоченный набор объектов одного и того же типа. Элементами списка могут быть целые числа, действительные числа, символы, строки, символические имена и структуры. Порядок расположения элементов в списке играет важную роль: те же самые элементы списка, упорядоченные иным способом, представляют уже совсем другой список. Совокупность элементов списка заключается в квадратные скобки ([]), элементы друг от друга отделяются запятыми. Список может содержать произвольное число элементов, единственным ограничением является объем оперативной памяти. Количество элементов в списке называется его длиной. Список может содержать один элемент и даже не содержать ни одного элемента. Список, не содержащий элементов, называется пустым или нулевым списком. 12 Непустой список можно рассматривать как список, состоящий из двух частей: головы – первого элемента списка; и хвоста – остальной части списка. Голова является элементом списка, хвост является списком. Голова списка – это неделимое значение, хвост представляет собой список, составленный из того, что осталось от исходного списка в результате «отделения головы». Этот новый список обычно можно делить и дальше. Если список состоит из одного элемента, то его можно разделить на голову, которой будет этот самый элемент, и хвост, являющийся пустым списком. Пустой список нельзя разделить на голову и хвост. Операция деления списка на голову и хвост обозначается при помощи вертикальной черты (|): [Head | Tail]. Head здесь является переменной для обозначения головы списка, переменная Tail обозначает хвост списка (для имен головы и хвоста списка пригодны любые допустимые Прологом имена). Данная операция также присоединяет элемент в начало списка, например, для того , чтобы присоединить X к списку S следует написать [X | S]. Отличительной особенностью описания списков является наличие звездочки (\*) после имени домена элементов.

# 2.6 Использование составных термов

В Прологе функциональный терм или предикат можно рассматривать как структуру данных, подобную записи в языке Паскаль. Терм, представляющий совокупность термов, называется составным термом. Предикаты, записанные в виде составного терма, называются составной структурой данных. Составные структуры данных в Турбо-Прологе объявляются в разделе domains. Если термы структуры относятся к одному и тому же типу доменов, то этот объект называется однодоменной структурой данных. Если термы структуры относятся к разным типам доменов, то такая структура данных называется многодоменной структурой данных. Использование доменной структуры упрощает структуру предиката

# 3 Тип задачи

Создание базы данных на основе Prolog и создание запросов к ней.

# 4 Способы решения задачи

Записываем данные в задаче в виде фактов и правил, и записываем запросы к базе данных.

# 5 Работа программы

В программе присутствует 14 правил и 5 фактов. 9 фактов – В программе присутствует 14 правил и 5 фактов. 3 факта описывают базу данных, остальные два – вспомогательные. 9 правил – описывают запросы к базе данных, остальные 5 – вспомогательные.

Также в программе присутствуют 7 предикатов:

* g(string,integer,integer,studenti) – устанавливает группу, studenti – массив объектов student.
* q11(list,integer) – вспомогательный предикат для подсчета элементов, list – массив целых чисел. Также существует еще одна его вариация для подсчета массива строк.
* member(student,studenti). does(a,a) – – вспомогательный предикат для определения принадлежности элемента массиву студентов, student – объект, выражающийся фамилией, именем, отчеством, обучении на военной кафедре а также списком сданных предметов (string,string,string,string,predmeti). Также существует еще 3 его вариации для определения принадлежности массивам предметов, строк и целых чисел.
* zadolznosti(slist) – запрос, возвращающий всех двоечников.
* dvoika(string) – вспомогательный запрос, находящий студента у которого есть двойка за предмет.
* bestStudent(list) – запрос, возвращающий группу отличников.
* pyat(integer) - вспомогательный запрос, находящий группу у которой больше всего отличников.
* sveddom(slist,integer) – запрос, возвращающий предметы, по которым учатся определенная группа
* pred(string,integer) - вспомогательный запрос, находящий предметы, у определенной группы.
* remove\_duplicates(slist,slist) – вспомогательный запрос, убирающий дубликаты из массива строк. Существует еще одна вариация с целыми числами.
* pred\_Gropus(list,string) – запрос, возвращающий список групп, у которых существует заданный предмет.
* chislo\_grupp(string,integer) – запрос, считающий кол-во студентов на заданном факультете.
* count(string,string) – вспомогательный запрос подсчета студентов.

Результатом работы программы является выборки данных из базы данных (Рисунок 1-5).

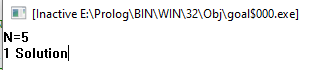


Рисунок 1 - пример работы программы при первом запросе

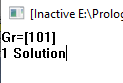


Рисунок 2 - пример работы программы при втором запросе

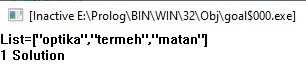


Рисунок 3 - пример работы программы при третьем запросе

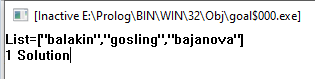


Рисунок 4 - пример работы программы при четвертом запросе

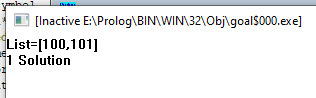


Рисунок 5 - пример работы программы при пятом запросе

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания к лабораторным работам [Электронный ресурс].URL: http://virtual6.ssau.ru/Moodle/file.php/979/Metodicheskie\_ukazanija\_k\_lr\_po\_prologu.pdf (дата обращения: 21.02.2022).
2. Список(Информатика) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA\_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) (дата обращения: 21.02.2022).
3. Логическое программирование на Prolog для чайников [Электронный ресурс]. URL: https://habr.com/ru/post/552318/ (дата обращения: 21.02.2022).
4. Урок 9. Списки и их рекурсивная обработка [Электронный ресурс]. URL: https://www.youtube.com/watch?v=Rk2ot8qq3Ss&t=429s (дата обращения: 21.02.2022)

# ПРИЛОЖЕНИЕ А

Исходный код программы.

DOMAINS

predmet=v(string,integer)

predmeti=predmet\*

student=s(string,string,string,string,predmeti)

studenti=student\*

list=integer\*

slist=string\*

PREDICATES

nondeterm g(string,integer,integer,studenti).

nondeterm q11(list,integer).

nondeterm q11(slist,integer).

nondeterm member(student,studenti).

nondeterm member(predmet,predmeti).

nondeterm member(string,slist).

nondeterm member(integer,list).

nondeterm zadolznosti(slist).

nondeterm dvoika(string).

nondeterm bestStudent(list).

nondeterm pyat(integer).

nondeterm sveddom(slist,integer).

nondeterm pred(string,integer).

nondeterm remove\_duplicates(slist,slist).

nondeterm remove\_duplicates(list,list).

nondeterm pred\_Gropus(list,string).

nondeterm chislo\_grupp(string,integer).

nondeterm count(string,string).

CLAUSES

g(iTFacultet,100,1,[s(ivan, sergeevich, balakin,y,[v(matan,5),v(algebra,4),v(russian,2)]),s(vlad, sergeevich, bezgin,y,[v(matan,3),v(geology,4),v(informatic,4)]),s(ryan, ivanovich, gosling,n,[v(matan,4),v(driving,5),v(english,2)])]).

g(iTFacultet,101,2,[s(jora, nikolavich, bomjihin,n,[v(optika,5),v(termeh,5),v(matan,5)]),s(kiril, ilich, petrov,y,[v(termeh,5),v(matan,5)])]).

g(uRfacultet,102,3,[s(anastasia, radionovna, bajanova,n,[v(prolog,3),v(vbasic,2)]),s(anna, petrovna, sidyakina,n,[v(prolog,3),v(vbasic,5)])]).

q11([],0).

q11([\_|Tail],N):-q11(Tail,N1),N=N1+1.

count(Surname,F):-g(F,\_,\_,S),member(s(\_,\_,Surname,\_,\_),S).

chislo\_grupp(F,N):-findall(S,count(S,F),List),q11(List,N).

remove\_duplicates([], []):-!.

remove\_duplicates([X|Xs], Ys):- member(X, Xs),!, remove\_duplicates(Xs, Ys).

remove\_duplicates([X|Xs], [X|Ys]):- !, remove\_duplicates(Xs, Ys).

member(H,[H|\_]).

member(H,[\_|Tail]):-member(H,Tail).

pyat(Gr):-g(\_,Gr,\_,S),member(s(\_,\_,\_,\_,Ocenki),S), not(member(v(\_,4),Ocenki)),not(member(v(\_,3),Ocenki)),not(member(v(\_,2),Ocenki)).

bestStudent(List1):-findall(Gr,pyat(Gr),List),remove\_duplicates(List,List1).

pred(Pred,Gr):-g(\_,Gr,\_,S),member(s(\_,\_,\_,\_,Ocenki),S),member(v(Pred,\_),Ocenki).

sveddom(List1,Gr):-findall(Pred,pred(Pred,Gr),List),remove\_duplicates(List,List1).

dvoika(Surname):-g(\_,\_,\_,Students),member(s(\_,\_,Surname,\_,Ocenki),Students),member(v(\_,2),Ocenki).

zadolznosti(List1):-findall(Surname,dvoika(Surname),List),remove\_duplicates(List,List1).

pred\_Gropus(List1,Pred):-findall(Gr,pred(Pred,Gr),List),remove\_duplicates(List,List1).

goal

%bestStudent(Gr).

%zadolznosti(List).

%sveddom(List,101).

%pred\_Gropus(List,matan).

chislo\_grupp(iTFacultet,N).