Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Часть 2

(Логическое программирование)

Методические указания к лабораторным работам и организации самостоятельной работы
для студентов направления 09.03.04
«Программная инженерия»
(уровень бакалавриата)

2018

Салмина Нина Юрьевна
Функциональное и логическое программирование. Часть 2 (Логическое программирование): Методические указания к лабораторным работам и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень бакалавриата) / Н.Ю. Салмина. – Томск, 2018. – 23 с.

Оглавление

1 Введение

Цель дисциплины – ознакомление студентов с основами логического программирования на примере языка Пролог, принципами структурирования знаний в виде фактов и правил, а также формирование у студентов профессиональных знаний и практических навыков по разработке и созданию интеллектуальных систем с помощью языка логического программирования Пролог.

Изучение данной части дисциплины включает в себя: теоретический раздел (изучение теоретического материала); практический раздел (выполнение лабораторных и контрольных работ); итоговый контроль результата изучения дисциплины. Данное пособие содержит в себе методические указания и варианты заданий для лабораторных работ, вопросы по организации самостоятельной работы по второй части дисциплины: Логическое программирование

2 Методические указания к проведению лабораторных работ

2.1 Лабораторная работа «Основы языка Пролог. Создание простейших функций»

Цель работы

Целью данной работы является знакомство со средой визуальной разработки Visual Prolog и получение первичных навыков работы в этой среде путем написания простейших процедур.

Рекомендации по подготовке к работе

Для работы в среде Visual Prolog необходимо создать проект.

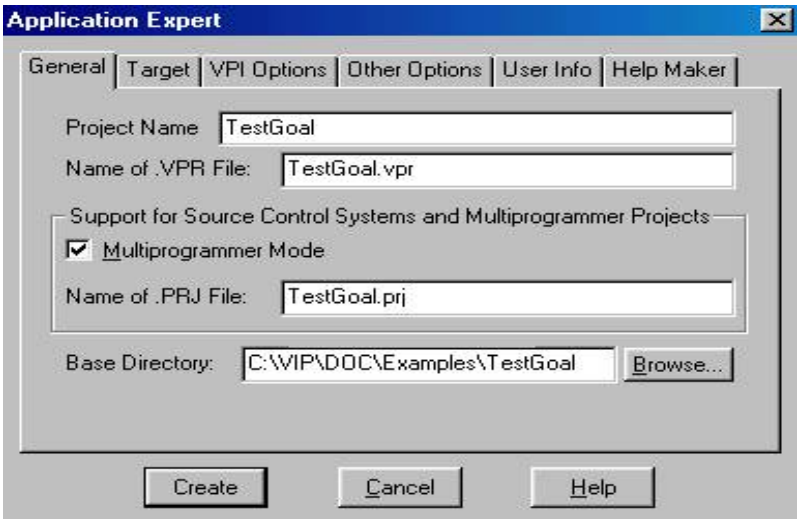
Для создания проекта требуется определить некоторые (не предопределенные) опции компилятора Visual Prolog. Для этого выполните следующие действия:

1. Запустите среду визуальной разработки Visual Prolog.

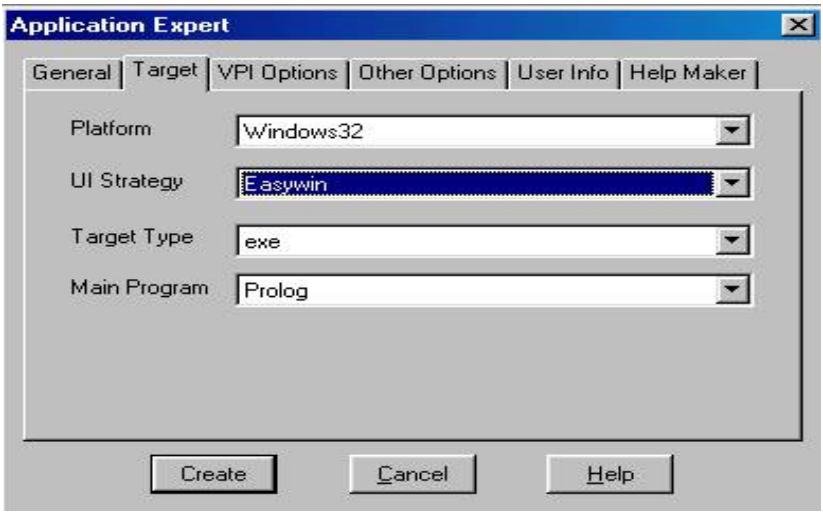
2. Создайте новый проект.

Выберите команду Project | New Project, активизируется диалоговое окно Application Expert.

3. Определите базовый каталог и имя проекта в поле Project Name. Щелкните мышью внутри поля Name of .VPR File. Также установите флажок Mulltiprogrammer Mode и щелкните мышью внутри поля Name of.PRJ File:



На вкладке Target рекомендуется выбрать следующие параметры:



Теперь нажмите кнопку Create для того, чтобы создать файлы проекта по умолчанию.

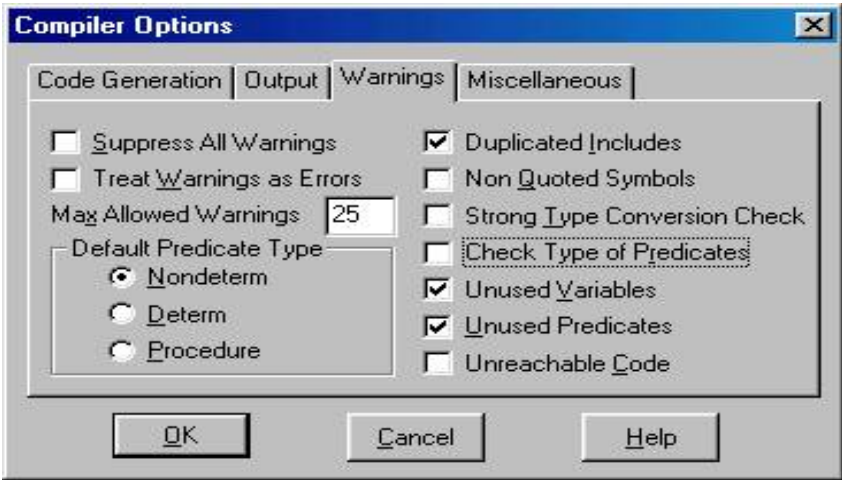
4. Установите требуемые опции компилятора для созданного проекта. Для активизации диалогового окна Compiler Options выберите команду Options | Project | Compiler Options. Откройте вкладку Warnings. Выполните следующие действия:

• установите переключатель Nondeterm. Это нужно для того, чтобы компилятор Visual Prolog принимал по умолчанию, что все определенные пользователем предикаты — недетерминированные (могут породить более одного решения);

• снимите флажки Not Quoted Symbols, Strong Type Conversion Check и Check Type of Predicates. Это будет подавлять некоторые возможные предупреждения компилятора;

•нажмите кнопку OK, чтобы сохранить установки опций компилятора.

В результате этих действий диалоговое окно Compiler Options будет выглядеть следующим образом:



После написания программы и определения цели необходимо запустить программу на выполнение. Для этого можно использовать команду Project | Test Goal (либо комбинацию клавиш <Ctrl>+<G>).

Более подробно работу в среде Visual Prolog – построение, компиляция и выполнение более сложных проектов можно посмотреть в файле «Работа в среде Visual Prolog.doc».

Задание на практику

1) Создать проект.
2) Написать предикат/процедуру, указанные в вашем варианте задания. При необходимости можно использовать предикаты работы со списками, рассмотренные на лекциях.
3) Протестировать работу написанных процедур.
4) Номера заданий указываются преподавателем.

Варианты заданий

1. Определите предикат translate(List1,List2) для преобразования списка цифр (целое число от 0 до 9) в список соответствующих слов, например, следующим образом:
   translate([3,5,1,3],[three, five, one, three]).
2. Напишите предикат p(V, L) - истинный тогда и только тогда, когда список L получается после удаления из списка V всех элементов, стоящих на нечетных местах, например:
   ?- p([0,a,b,c,d,e,f,g],X).
   X=[a,c,e,g]

3. Определите предикат ordered(List), который является истинным, если List –упорядоченный список чисел, например ordered([1,5,6,9,12]) – истина.
4. Определите предикат p(List1,List2,List3), который из двух списков (возможно разной длины) $[a_1, a_2, \ldots, a_n]$ и $[b_1, b_2, \ldots, b_n]$ создает список арифметических выражений $[a_1+b_1, a_2+b_2, \ldots, a_k+b_k]$, где k=min(n,m). Например, p([1,3,5,7],[2,4,6],[1+2,3+4,5+6]) – истина.
5. Определите отношение divideList(List, List1, List2) таким образом, чтобы элементы списка List попеременно распределялись между списками List1 и List2, причем List1 и List2 имели примерно одинаковую длину, например divideList([a,b,c,d,e],[a,c,e],[b,d]) – истина.
6. Определите предикат Split (Numbers, Pozitives, Negatives, Zeroes), который разбивает список чисел на три списка: положительные, отрицательные и нули, например, split([3,-1,0,5,-2],[3,5],[-1,-2],[0]) - истина.
7. Даны два упорядоченных по возрастанию списка чисел X и Y. Написать предикат p(X,Y,Z), который в качестве значения переменной Z выдает общий упорядоченный список элементов X и Y. Например,
   ?– p([1,3,5,7,8], [2,3,5,7],Z).
   Z=[1,2,3,3,5,5,7,7,8]
8. Определите предикат p (X, Y), вычисляющий знакочередую-щуюся сумму $Y = a_1 – a_2 + a_3 – a_4 + \ldots + (–1)^k a_k$ для списка X, имеющего вид $[a_1, a_2, \ldots, a_k]$.
9. Напишите предикат p(X, N, L) - истинный тогда и только тогда, когда L - список из N раз повторенных элементов X. Например:
   ?- p(a,5,X).
   X=[a,a,a,a,a]
10. Напишите предикат p(X, N, Y), осуществляющий циклическую перестановку элементов в любом списке на N позиций (знак N говорит о направлении сдвига). Подсказка: используйте рекурсию по N; базовые случаи: N=0, N=1, N= –1.
    ?– p([a,b,c,d],1,Y).
    Y=[d,a,b,c]
    ?– p([a,b,c,d,e,f],2,Y).
    Y=[e,f,a,b,c,d]
    ?– p([a,b,c,d],–1,Y).
    Y=[b,c,d,a]
11. Напишите предикат, который из данного списка строит список списков его элементов, например, [1,2,3,4,5] => [[1],[2],[3],[4],[5]].

12. Определите предикат p(U, V, L) - истинный тогда и только тогда, когда список L есть список всех элементов списка U, не содержащихся в списке V.
13. Напишите предикат, который проверяет, являются ли все элементы числового списка положительными.
14. Напишите предикат p(L, N) - истинный тогда и только тогда, когда N - предпоследний элемент списка L, имеющего не менее двух элементов.
15. Напишите предикат p(V, L) - истинный тогда и только тогда, когда список L получается после удаления из списка V всех элементов, стоящих на четных местах, например,
    ?- p([a,b,c,d,e,f,g],X).
    X=[a,c,e,g]
16. Напишите предикат p(S,X,Y), который выдает Yes тогда и только тогда, когда X и Y являются соседними элементами в списке S.
17. Существует следующее правило для високосных годов (годы с 366 днями):
    - год, делимый на 4, - високосный год (например, 1972);
    - но: если он делится на 100, это не високосный год (1900);
    - но: если он делится на 400, это - високосный год (2000).

    Напишите предикат p(N), который выясняет, является ли год N високосным.
18. Определите предикат p (N, S), вычисляющую сумму S = 1*2*3 + 4*5*6 +... + (n–2)*(n–1)*n, где n кратно 3.
19. Напишите новую версию процедуры "предок", которая вырабатывает список представителей всех промежуточных поколений, располагающихся между предком и потомком. Предположим, например, что Генри является отцом Джека, Джек - отцом Ричарда, Ричард - отцом Чарльза, а Чарльз - отцом Джейн. При запросе о том, является ли Генри предком Джейн, должен выдаваться список, характеризующий родственную связь этих людей, конкретно: [джек, ричард, чарльз].
20. Список, состоящий из целых чисел, иногда удобно представить в виде гистограммы. Напишите предикат p, который отображает список на экран в следующем виде (предполагается, что числа в списке не слишком велики):
    ?– p([3,4,6,5]).
    ***
    ****
    ******
    *****

21. Определите предикат p(X,Y), который меняет местами первый и последний элементы списка. Пример:
    ?- p([a,b,c,d,e,f,g],X).
    X=[g,b,c,d,e,f,a];
22. Напишите предикат prime(N), который определяет, является ли данное натуральное число N простым.
23. Определите возведение в целую степень через умножение и деление. Используйте рекурсию по показателю степени.
24. Последовательность чисел Фибоначчи 1,1,2,3,5,8,13… строится по следующему закону: первые два числа – единицы; любое следующее число есть сумма двух предыдущих. Напишите предикат, который вычисляет n-е число Фибоначчи.
25. Напишите предикат p(N, V, L) – истинный тогда и только тогда, когда список L получается после удаления N-го элемента из списка V.
26. Вычисление с днями недели. Обозначим дни как 0 = воскресенье, 1 = понедельник, ..., 6 = суббота. Определите предикат p(S, N, R), который вычисляет, какой будет день недели через N дней, если сегодня день недели S.
27. Напишите предикат p(L, N1, N2, R), который удаляет элементы списка L с позиции N1 до позиции N2 включительно.

2.2 Лабораторная работа «Разработка программ»

Цель работы

Целью данной работы является знакомство с рекурсией и принципами построения рекурсивных функций.

Рекомендации по подготовке к работе

В процессе выполнения лабораторной работы необходимо написать программу, выполняющую действия, описанные в вашем варианте задания. Количество необходимых предикатов, их входные и выходные параметры определить самостоятельно, исходя из задания.

Проверьте работу написанных предикатов на различных примерах. Обязательно проверяйте правильность работы программы при граничных значениях переменных. Также проанализируйте, как будет реагировать программа при неправильно или некорректно заданных значениях переменных.

При написании рекурсивных процедур проверяйте правила останова: в каких ситуациях мы должны завершать работу процедуры; в каком порядке записывать предложения процедуры – влияет ли порядок записи предложений на результат работы предиката.

Варианты заданий

1. Простейшая система кодирования сообщений заключается в замене каждой буквы сообщения на букву, находящуюся на N-й по отношению к ней позиции в алфавите. Например, для N=2 буква “а” заменяется буквой “с”, буква “у” — буквой “а” и т.д. Напишите процедуру для предиката *шифратор*, который берет шифруемое слово и целое число и выдает слово, представляющее шифр данного слова, полученный с помощью указанного метода. N может быть любое неотрицательное целое число.

2. Напишите предикат p(X,Y,W,R), осуществляющий замену элементов списка X на соответствующие элементы списка Y в списке W, например,
   ?- p([1,2],[10,20],[1,6,4,2,7,8,1],X).
   X=[10,6,4,20,7,8,10]

3. Определите предикат p(V, N, L) - истинный тогда и только тогда, когда L - список элементов списка V, встречающихся в нем не менее N раз. Проверьте работу этого предиката на примере списка
   [a, a, b, a, c, b, c, a, b, b, d, a, b] для N=1,2,5,0.

4. Написать программу, которая возвращает список (m1 m2 m3), состоящий из трех наибольших элементов исходного числового списка s: m1>=m2>=m3. Исходный список содержит не менее трех элементов.

5. Определить предикат p(X,Y), где X – одноуровневый список. Предикат должен подсчитывать количество вхождений в список каждого атома и выдавать результат в виде списка списков: X= [a, s, v, d, s, s, a, v] ==> Y=[[a, 2],[s, 3], [v, 2], [d, 1]]

6. Написать программу, осуществляющую перевод десятичного числа в любую заданную систему счисления (двоичную, восьмеричную и т.п.)

7. Определить предикат two_lists(A,B) где A и B – числовые списки. Предикат должен проверять, выполняется ли следующее соотношение между элементами списков: bi=ai+1; $a_{i+1}=2*b_i$

8. Определить предикат form(A,B,C,Y). A – это множество – список целых чисел, B и C – целые числа. Предикат должен формировать список Y трех подмножеств данного множества, определяемых следующим образом:
   a. Четные числа множества;
   b. Числа множества, являющиеся квадратами числа;
   c. Все числа b<=ai<=c.

   Например: form ([1, 2, 3, 4], 0, 2, X) => X=[[2, 4], [1, 4], [1, 2]]

9. Напишите программу, формирующую список простых чисел на заданном интервале.

10. Задан числовой список. Написать программу, подсчитывающую среднее значение элементов списка, за исключением максимального и минимального элементов.

11. Написать программу, осуществляющую перевод числа, представленного в любой заданной системе счисления (двоичной, восьмеричной и т.п.), в десятичную систему счисления.

12. Задан список вида [[A1, N1], [A2, N2],…[AK, NK]], где Ai – символьный атом, Ni – число. Написать программу, возвращающую список из двух символьных атомов исходного списка [Ai, Aj] с максимальным произведением Ni*Nj.

13. Напишите предикат p(S, L, N), который вычисляет, сколько раз список S входит в список L, как подсписок.
    Пример:
    ?- p([a,b,c],[a,a,b,c,d,a,b,b,c,c,a,b,c],N).
    N = 2

14. Последовательности Улама. Определим последовательность x0, x1, x2, ... следующим образом: x0 - произвольное нечетное число, отличное от единицы. При n > 0 имеем xn = u(xn–1)), где u - функция Улама, определяемая как
    u(x) =x/2, если x четно,
    u(x) = 3x + 1, если x нечетно.

Последовательность заканчивается, когда в ней встречается значение 1. Вот последовательность, которую мы получим, исходя из значения 7: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.
Напишите предикат p(N, L), который для данного нечетного N вычисляет L последовательность Улама, начинающуюся с N.

15. Напишите предикат p(X, Y, Q, S):
X ≡ [x1,x2,...,xn] и Y ≡ [y1,y2,...,ym] - числовые списки, Q - заданное число; предикат p - истинный тогда и только тогда, когда S есть сумма вида xi+yj, наиболее близкая к числу Q.

16. Для представления римских цифр используются символы: I - один, V - пять, X - десять, L - пятьдесят, C - сто, D - пятьсот, M - тысяча. Для изображения числа с помощью римских цифр используются общеизвестные правила; так, например, 482 - CDLXXXII, 1999 - MCMXCIX. . Напишите предикат p(N, R), который переводит целое число N из диапазона от 1 до 2000 в запись с римскими цифрами; например,
?– p(482, Y).
Y='CDLXXXII'

17. Натуральное число n называется совершенным, если сумма всех его делителей равна 2n. Найдите все совершенные числа, меньшие 1000 (их должно быть 3).

18. В старояпонском календаре был принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Подциклы обозначались названиями цветов: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла года носили названия животных: крыса, корова, тигр, заяц, дракон, змея, лошадь, овца, обезьяна, курица, собака и свинья. Например, 1984 год - год начала очередного цикла - назывался Годом Зеленой Крысы.
Составьте программу, которая по заданному номеру года нашей эры печатает его название в старояпонском календаре.

19. Определите предикат subsum(Set, Sum, SubSet) такой, что Set является списком чисел, SubSet - подмножеством этих чисел, а сумма чисел в SubSet равна Sum, например:
?- subsum([1,2,5,3,2],5,Sub).
Sub = [1,2,2];
Sub=[2,3];

Sub=[5];

20. Напишите предикат p(N, L) - истинный тогда и только тогда, когда список L содержит все последовательности (списки) из N нулей и единиц, в которых никакая цифра не повторяется три раза подряд (нет куска вида XXX).

21. Построить программу "сжать", назначение которой - преобразование английских слов в их "звуковой" код. Этот процесс предусматривает "сжатие" примерно одинаково звучащих слов в одинаковый их код - своего рода, аббревиатуру этих слов. Слова "сжимаются" в соответствии со следующими правилами:
    - первая буква слова сохраняется;
    - все последующие за ней гласные, а также буквы "h", "w" и "y" удаляются;
    - сдвоенные буквы заменяются одиночными;
    - закодированное слово состоит не более чем из четырех букв, остальные буквы удаляются.

    Примеры: сжать(barrington, brng); сжать(llewellyn, ln).

22. Определите предикат center(X, N, R) – истинный тогда и только тогда, когда атом R имеет длину N, и атом X располагается приблизительно в центре R, а остальные символы – пробелы (N не меньше длины X). Например,
    ?– center(ab, 3, X).
    X=’ ab‘
    ?– center(abc, 6, X).
    X=’ abc ‘

23. Напишите предикат, который формирует список делителей заданного натурального положительного числа.

2.3 Лабораторная работа «Графы и деревья»

Цель работы

Целью данной работы является работа с графами, представленными различными способами и написание предикатов для работы с ними.

Рекомендации по подготовке к работе

В процессе выполнения лабораторной работы необходимо написать программу определяющую для графа (дерева) заданную характеристику.

Решение различных задач обработки графов напрямую зависит от способа их представления. Так, для поиска пути на графе удобнее использовать вектора смежности. Если же мы решаем задачу построения остовного дерева, граф удобнее представлять в виде списка ребер и множества узлов графа.

Представление деревьев зависит от вида дерева: обычное или бинарное, и от вида решаемой задачи.

Продумайте алгоритм, который позволит решить поставленную задачу. При необходимости вы можете воспользоваться существующими алгоритмами просмотра, изменения, корректировки графов и деревьев.

Если в вашем варианте не оговорен способ представления графа (дерева), вы можете выбрать тот способ представления, который будет боле удобен для решения поставленной задачи.

Варианты заданий

1. Дан неориентированный граф. Написать программу, которая находит в графе максимальный цикл и выдает его в виде списка вершин. Если в графе нет циклов, функция должна сообщать об этом.
2. Написать программу, определяющую, связан ли рассматриваемый неориентированный граф.
3. Задан неориентированный граф, у которого для каждой дуги задана ее длина: ((a b 12) (s d 3) …). Написать программу, определяющую кратчайший путь между указанными двумя вершинами.
4. Написать программу, подсчитывающую количество циклов в неориентированном графе.
5. Написать программу, которая проверяет, является ли граф гамильтоновым, и если да, то найти гамильтонов цикл. Цикл в графе называется гамильтоновым, если он содержит все вершины графа ровно по одному разу; граф с таким циклом называется гамильтоновым.
6. Написать программу, которая определяет, существует ли путь из A в B в ориентированном графе. Если путь существует, выдать его в качестве результата работы.
7. Определите программу, аргументом которой является дерево (не обязательно бинарное!). Функция должна вернуть ветвь с максимальным количеством листьев.

8. Написать программу, которая будет строить список смежностей по представлению графа, заданному в виде матрицы смежностей.
9. Написать программу, которая ищет заданную вершину в дереве и возвращает список, содержащий предка искомой вершины и ее потомков: (предок (потомок1 потомок2 …)).
10. Напишите программу, выясняющую, лежат ли две заданные вершины дерева на одном пути.
11. Напишите программу, определяющую эйлеровый путь, начинающийся с заданной вершины неориентированного графа. Путь называется эйлеровым, если проходит все ребра графа по одному разу. Если такой путь не существует – программа должна сообщать об этом.
12. Написать функцию, которая проверяет, является ли заданный граф деревом (имеет ли циклы)?
13. Напишите программу, определяющую компоненты связности данного графа. Результатом работы программы должен быть список списков (компонента связности – список вершин).
14. Напишите программу, на вход которой подаются два дерева. Программа должна проверять, изоморфны ли данные деревья (Два дерева называются изоморфными, если можно отобразить одно из них в другое, изменением порядка ветвей в поддеревьях).
15. Определить функцию, аргументом которой является дерево, подсчитывающую количество листьев в дереве.
16. Определить отношение substree(A,B,T1,T2), выполнимое, если двоичное дерево T2 получено из двоичного дерева T1 заменой всех вхождений элемента A на элемент B.
17. Написать функцию, на вход которой подается дерево, определяющую максимальную глубину дерева.
18. Стяжение ветви. Определить функцию, аргументами которой является дерево и две его вершины. Функция должна склеивать два заданных узла, если они соседние и выдавать сообщение об ошибке в противном случае.
19. Задан граф, у которого для каждой дуги задана ее длина: ((a b 12) (s d 3) …). Написать программу, которая находит две самые удаленные друг от друга вершины.
20. Написать программу, которая ищет середину кратчайшего пути между двумя заданными вершинами графа. Функция должна возвращать: список из имени вершины (если между исходными вершинами нечетное количество вершин), список из двух вершин (если четное) и пустой список, если заданные вершины – соседи.
21. Определите для бинарного дерева отношение ordered(Tree), выполненное, если дерево Tree является упорядоченным деревом

целых чисел, т. е. число, стоящее в любой вершине дерева, больше любого элемента в левом поддереве и меньше любого элемента в правом поддереве. Например, дерево tree(nil, 5, tree(nil, 6, tree(tree(nil, 8, nil), 10, nil))) является упорядоченным.

22. Напишите программу, формирующую для ориентированного графа G его транзитивное замыкание G*. Дуга (a b) принадлежит G*, если существует ориентированный путь из a в b.
23. Напишите предикат p(T,X,Y,R), который из бинарного дерева T делает бинарное дерево R, совпадающее с T, за исключением того, что вершины дерева, содержащие также в списке X, меняются на соответствующие (по порядку) вершины из списка Y.
    Например:
    ?– p(tree(nil, 5, tree(nil, 6, tree(tree(nil, 8, nil), 10, nil))), [5,8,7], [50,80,70], R).
    R= tree(nil, 50, tree(nil, 6, tree(tree(nil, 80, nil), 10, nil)))

2.4 Лабораторная работа «Работа с базой фактов»

Цель работы

Целью данной работы является получение навыков работы с базой фактов: поиск, удаление и корректировка требуемой информации.

Рекомендации по подготовке к работе

В процессе выполнения лабораторной работы необходимо написать программу работы с БД фактов, выполняющую заданное действие. Исходный файл с фактами можно создавать любым способом – программно или вручную. При необходимости можно создать несколько файлов: например, в одном файле могут находиться предикаты, неизменяемые в процессе работы программы, а в другом – факты, которые программа будет добавлять или изменять в процессе работы.

При создании базы фактов помните, что строковые данные в файле должны заключаться в кавычки.

Количество фактов в файле должно быть достаточным для тестирования работы программы!

Порядок проведения работы

1) Создайте файл с набором предикатов, структура которых определена в вашем задании. Набор фактов должен быть достаточным, чтобы проверить работу вашей программы!
2) Напишите программу, выполняющую заданные действия с набором фактов. При необходимости используйте вспомогательные предикаты.

Варианты заданий

1. В файле хранится база фактов о людях в виде предиката: person(<фамилия>,<имя>,<пол>,<дата рождения>). Необходимо отсортировать и вывести на печать людей по возрасту. Выводить: Фамилия дата_рождения.
2. В файле хранится база фактов о полетах в следующем виде: fly (<рейс>,<пункт вылета>,<пункт прилета>,<время вылета>,<время прилета>). По заданным X=«пункт_вылета» и Y= «пункт_прилета» сформировать список возможных рейсов и отсортировать его по времени перелета.
3. Результаты соревнований хранятся в виде базы фактов: матч(<команда1>,N1,<команда2>,N2), где Ni – количество голов, забитых командой i. Необходимо сформировать рейтинговую таблицу команд, если победа приносит команде 3 очка, а ничья – 1 очко. В случае одинакового количества очков, рейтинг выше у той команды, которая забила больше голов.
4. В файле хранится база фактов о животных в виде предиката: животное(<название>,<ареал_обитания>,<популяция>). Написать программу, которая бы позволяло править данные файла, вводя количество родившихся или умерших животных, а также формировать список вымирающих животных по заданной границе минимальной популяции.
5. В файле хранится база фактов о людях в виде предиката: person(<фамилия>,<имя>,<пол>,<дата рождения>). Сформировать и сохранить в другом файле список семей: семья (фамилия, имя_мужа, имя_жены, список_детей). Будем считать, что в список детей попадают люди возрастом менее 18 лет.
6. В файле хранится база фактов о полетах в следующем виде: fly(<рейс>,<пункт_вылета>,<пункт_прилета>,<цена>). Для заданных пункта отправления и пункта назначения выдать список всех возможных перелетов (учесть возможность не только прямых рейсов, а и перелетов с пересадками).

7. В файле хранится таблица рейтинга спортсменов в виде: спортсмен(<фамилия>,<количество набранных баллов>). Написать программу, позволяющую вводить результаты соревнований в виде: спортсмен – занятое место. Корректировать файл рейтинга спортсменов, учитывая, что за 20-е место дается 1 балл, за 19-е – 2 балла, …, за 1-е место – 20 баллов.
8. В файле хранится информация расписания экзаменов в виде: экзамен(<группа>,<предмет>,<преподаватель>,<дата>). Написать программу, позволяющую выбирать информацию по экзаменам для группы/преподавателя, сортируя ее по дате.
9. В файле хранится информация о музыкальных инструментах в следующем виде: инструмент(<название>,<группа>,<жанр>). Под группами инструментов подразумеваются: духовые, струнные, клавишные, ударные и т.д. «Жанр» - здесь хранится информация в виде списка, в каких жанрах музыки (поп, рок, классика и т.п.) может быть использован данный инструмент. Написать программу, формирующую: 1) список инструментов, используемых в заданном жанре; 2) какие группы инструментов используются в заданном жанре.
10. Рейтинг группы по дисциплине хранится в базе в виде следующего предиката: рейтинг(<фамилия>,[Б1,Б2,…]). Здесь Бі – баллы, набранные студентом. Написать программу, позволяющую добавлять баллы в таблицу рейтинга, а также сортировать группу по суммарному рейтингу.
11. В файле хранится база фактов о полетах в следующем виде: fly(<рейс>,<пункт_вылета>,<пункт_прилета>,L). Здесь L – список дней, по которым летает данный рейс, например, L=[ежедневно], L=[пон,среда,птн]. Для заданных пункта отправления, пункта назначения и дня вылета выдать все возможные перелеты (возможно, с пересадками).
12. В файле хранятся результаты сессии группы в следующем виде: студент(<фамилия>,[[Экз1,Оценка1],…[ЭкзN,ОценкаN]]). Написать программу, выдающую список отличников, хорошистов, студентов с тройками и должников (оценку по не сданному экзамену можно отображать как «неуд»).
13. В БД фактов (в файлах) хранятся данные по преподавателям и дисциплинам в следующих предикатах: лектор (<фамилия>,<дисциплина>) и студенты(<дисциплина>,<группа>). Написать программу, позволяющую: 1) для заданной группы формировать список преподавателей, ведущих у нее занятия; 2) изменять в предикате «лектор» информацию по преподавателю или по

названию дисциплины. Учесть, что при изменении названия дисциплины необходимо корректировать и предикат «студенты».

14. В файле хранится информация о музыкальных альбомах в следующем виде: музыка(<исполнитель/группа>,<альбом>,<время_звучания>, <год_выпуска>). Написать программу, позволяющую добавлять новые факты, а также формировать список альбомов по заданному исполнителю, сортируя по году выпуска.
15. В файлах хранится информация об актерах и фильмах в следующем виде: актер (<фамилия>, <название_фильма>), фильм(<название_фильма>,<жанр>,<год_выпуска>). Написать программу, формирующую список актеров по жанрам (если актер снимался в фильмах разных жанров, то определять его жанр исходя из максимального количества ролей в фильмах одного жанра).
16. В файле хранится информация о приеме врачей в поликлинике в следующем виде: врач(<фамилия_врача>,<фамилия_пациента>,t). Здесь t – время начала приема (под прием отводится полчаса). Написать программу, позволяющую записаться новому пациенту на прием. Для этого сначала вывести на экран свободные часы по запрашиваемому врачу, ввести фамилию пациента и выбранное им время, добавить новый факт в БД.
17. В файле хранится информация о книгах в следующем виде: книга(<автор>,<название>,<год_издания>,<цена>). Написать программу, позволяющую формировать список книг заданного автора, сортируя их по году выпуска и цене одновременно.
18. В файле хранится информация о соревнованиях по прыжкам в длину: спорсмен(<фамилия>,<попытка>,<длина_прыжка>). У каждого спорсмена есть три попытки. Написать программу, формирующую список результатов – спорсмен + лучшая попытка (максимальная длина), сортируя его по длине прыжка.
19. В файле хранится информация о цветах в следующем виде: цветок(<название цветка>,<цвет>,<высота>,<срок жизни>). Здесь «срок жизни» цветка может принимать значения «однолетник», «двухлетник», «многолетник». Написать программу, осуществляющую поиск цветов по цвету или сроку жизни. Выдаваемый список должен быть отсортирован по высоте цветов.
20. В файле хранится информация о товаре на складе в следующем виде: товар(<наименование>,<дата_поступления>,<срок_хранения>). Написать программу, формирующую список просроченного товара, сортируя его по времени просрочки.

3 Методические указания для организации самостоятельной работы

3.1 Общие положения

Цель самостоятельной работы по дисциплине – проработка лекционного материала, самостоятельное изучение некоторых разделов курса, подготовка к лабораторным работам, тестам и контрольной работе.

Самостоятельная работа студента по дисциплине «Функциональное и логическое программирование» включает следующие виды его активности:

1. проработка лекционного материала;
2. изучение тем теоретической части дисциплины, вынесенных для самостоятельной проработки;
3. подготовка к лабораторным работам;
4. подготовка к тестам;
5. подготовка к контрольной работе;
6. подготовка к экзамену.

3.2 Проработка лекционного материала

При проработке лекционного материала необходимо:

а) отработать прослушанную лекцию (прочитать конспект, прочитать учебное пособие и сопоставить записи с конспектом, просмотреть слайды) и восполнить пробелы, если они имелись (например, если вы что-то не поняли или не успели записать);

б) перед каждой последующей лекцией прочитать предыдущую, дабы не тратилось много времени на восстановление контекста изучения дисциплины при продолжающейся теме.

Данный вид деятельности ориентирован как на закрепление материала, так и на подготовку к тестовым заданиям и контрольным работам.

Содержание лекций:

Концепция и особенности логического программирования. Основы языка Пролог: термы, факты, предикаты. Программа на языке Пролог. Переменные и константы. Сложные термы: структуры, списки. Обратите внимание, что в Прологе элементы одного списка должны иметь одинаковый тип данных!

Объекты данных. Сопоставление: унификация термов. Декларативный смысл пролог-программ. Процедурная семантика. Порядок предложений и целей. Взаимосвязь между Прологом и логикой. Поиск решения на Прологе, понятие резольвенты, завершение поиска.

Понятие рекурсии. Рекурсивное определение правил. Терминальная ветвь, рекурсивная ветвь. Рекурсия и эффективность. Итерации.

Списки: представление списка, операции над списками, вложенные списки. Бинарные деревья. Операции над структурами данных. Встроенные предикаты. Отсечение: общее правило применения отсечения. Ввод и вывод. Работа с файлами. Циклы и повторения.

Работа с множествами. Сортировка. Графы: представление графов, поиск пути на графе. Отображение деревьев. Стратегии решения задач: поиск в глубину, поиск в ширину.

Работа с динамической базой фактов: добавление, удаление, корректировка фактов. Выбор информации из набора фактов. Сохранение данных.

3.3 Самостоятельное изучение тем теоретической части курса

3.3.1 Грамматики

Необходимо рассмотреть существующие виды грамматик: регулярные грамматики, контекстно-свободные грамматики. Структура любой грамматики включает в себя: множество нетерминальных символов, множество терминальных символов, множество правил вывода и начальный символ. Обратите внимание, что грамматики различаются между собой прежде всего структурой правил вывода.

Чаще всего Пролог-реализацию грамматик рассматривают как систему распознавания некоторого формального языка, описанного заданной грамматикой. Цепочки терминальных символов, представляющие собой предложение формального языка, на Прологе можно записывать в виде списков символов.

Наиболее широко в различных приложениях используются контекстно-свободные грамматики (КС-грамматики). В основе Пролог-реализации КС-грамматик лежит применение предиката append, с помощью которого исходный список разбивается на два подсписка, которые далее также разбиваются на подсписки. Процесс успешно завершается, когда в текущем списке остается только один терминальный символ.

Для того, чтобы программа гарантированно заканчивала работу (находила решение), необходимо использовать не леворекурсивную КС-грамматику. При использовании леворекурсивной грамматики программа может уйти в бесконечную рекурсию!

3.3.1 Вычислительные задачи, головоломки

Для лучшего понимания работы Пролог-программ необходимо ознакомиться с примерами реализации различных вычислительных задач, таких как: преобразования систем счисления, вычисление различных рядов и сумм, приближенное решение уравнений и методов интегрирования.

Несмотря на то, что логическое программирование изначально предназначено для обработки символьной информации, решение вычислительных задач позволяет глубже понять механизм работы рекурсии. В существующих алгоритмах математических вычислений зачастую уже выделены граничные условия (правило останова) и рекурсивные условия. При программировании вычислительных задач необходимо уделять особое внимание проверкам на отрицательность/положительность значений переменных, а также на использование отсечения.

Для более глубокого понимания представления знаний, формализации различных логических утверждений, необходимо изучить способы решения различных головоломок. Головоломки могут быть сведены к решению следующих проблем:

- установление соответствия между множествами;

- составление логических уравнений;

- составление расписаний.

Решение задачи установления соответствия между множествами находится последовательным исключением недопустимых комбинаций. Процесс исключения продолжается до тех пор, пока не останется только та комбинация, которая отвечает условиям задачи.

Пролог-решение головоломок типа составления логических уравнений предполагает наличие процедур выполнения логических операций конъюнкции, дизъюнкции и отрицания и задания интерпретации, в которой данная формула была бы выполнима.

Задачи составления расписаний можно решать как путем установления соответствия между множествами, так и составлением логических уравнений.

4 Рекомендуемая литература

Цуканова, Н.И. Теория и практика логического программирования на языке Visual Prolog 7.Учебное пособие для вузов [Электронный ресурс] : учеб. пособие / Н.И. Цуканова, Т.А. Дмитриева. — Электрон. дан. — Москва: Горячая линия-Телеком, 2013. — 232 с. — ЭБС ЛАНЬ. – Режим доступа: https://e.lanbook.com/book/11847.