Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт

по практической работе №1

«Основные сведении о языке Пролог и системе программирован. *Visual* *Prolog* »

по дисциплине «Экспертные Системы»

Вариант 2

|  |  |
| --- | --- |
| Выполнил: | Проверила: |
| студент гр. 820601 | Т. В. Тиханович |
| Шведов А. Р. |  |
|  |  |

Минск 2022

1. Цель работы

Целью данной работы являются изучение основных возможностей языка Пролог и системы программирования *Visual* *Prolog* и изучение механизмов управления в программах на языке Пролог.

1. Теоретическая часть
   1. Основные конструкции языка Пролог. Понятие предиката

Пролог – язык логического программирования, предназначенный для представления и обработки знаний о некоторой предметной области. В Прологе реализован так называемый декларативный подход, при котором задача описывается с помощью набора утверждений о некоторых объектах и правил обработки этих утверждений.

Основной конструкцией̆ языка Пролог является предикат. Предикат – это логическая функция от некоторого набора аргументов; при этом аргументы могут иметь любой вид, а функция принимает значение «ложь» или «истина». В языке Пролог имеется три вида предикатов: предикаты-факты, предикаты-правила и стандартные предикаты. Аргументами предикатов-фактов являются константы.

Предикаты-факты предназначены для записи некоторых утверждений, которые при выполнении программы считаются истинными. Пусть, например, требуется записать в программе утверждение: «Иван – отец Петра». В программе на Прологе его можно выразить фактом: *father*("Иван","Петр"). Здесь *father* – имя предиката (оно может быть любым другим). Этот предикат имеет два аргумента, оба - строковые.

Предикаты-правила записываются в виде: <Предикат1> истинен, если истинны <Предикат2>, <Предикат3>,...,<ПредикатN>. Здесь <Предикат1>  
называется головным (или заголовком), остальные – телом правила.  
Пусть в программе есть факты, задающие отношение «отец», а требуется составить правило, позволяющее найти по этим данным брата конкретного человека (или проверить, являются ли два конкретных человека братьями). Это правило можно сформулировать так:

«Один человек – брат другого, если эти люди разные и у них один и тот же отец». На Прологе это можно записать так:  
*brother* (*X*, *Y*):- *father* (*Z*, *X*), *father* (*Z*, *Y*), *X*<>*Y*.

Стандартные предикаты входят в состав самого языка Пролог. Простейшие из них: *write*(*X*) – вывод значения *X* на экран или на другое устройство, где *X* - переменная или константа; *nl* - переход в следующую строку на экране; *readln*(*X*) – ввод строковой переменной;

*readint*(*X*) – ввод целочисленной переменной; *readreal* (*X*) – ввод вещественной переменной.

* 1. Принципы работы программ на языке Пролог

Выполнение программы на Прологе заключается в доказательстве целевого предиката. Этот предикат обычно является правилом. Чтобы доказать правило (любое, а не только цель), требуется доказать все предикаты, составляющие его тело, т.е. найти факты, соответствующие этим предикатам. Для этого происходит согласование предиката с другим одноименным предикатом, т.е. сопоставление (унификация) соответствующих аргументов этих предикатов. Согласование предикатов выполняется успешно, если успешна унификация всех аргументов предикатов. При унификации аргументов возможны следующие основные случаи:

* сопоставление двух констант. Заканчивается успешно, если они равны, и неудачно, если они не равны.
* сопоставление константы и переменной, еще не имеющей значения (свободной). Заканчивается успешно, и переменная получает значение константы (становится связанной).
* сопоставление двух связанных (т.е. имеющих значения) переменных. Заканчивается успешно, если значения переменных равны, и неудачно, если они не равны.
* сопоставление двух переменных, одна из которых связана, а другая свободна (т.е. еще не получила значение). Заканчивается успешно, и свободная переменная принимает то же значение, что и связанная.
* сопоставление двух свободных переменных. Заканчивается успешно; если впоследствии одна из переменных получает значение, то и другой переменной присваивается то же значение.  
   Если согласование предикатов закончилось неудачно, то делается попытка согласования данного предиката с другим одноименным клозом. Если таких клозов нет, то происходит возврат (бэктрекинг) к ближайшей «развилке», т.е. к точке программы, в которой было возможно другое согласование предикатов.
  1. Механизмы управления в языке Пролог

Встроенным механизм поиска с возвратом в Прологе может принести к поиску не нужных решений, в результате чего теряется эффективность, например, когда желательно найти только одно решение. В других случаях может оказаться необходимым продолжать поиск дополнительных решений, даже если целевое утверждение уже согласовано.

    Пролог обеспечивает два инструментальных средства, которые дают возможность управлять механизмом поиска с возвратом: предикат *fail*, который используется для инициализации поиска с возвратом, и отсечение (обозначается как «!») – для запрета возможности возврата.

Пролог начинает поиск с возвратом, когда вызов завершается неудачно. В определенных ситуациях бывает необходимо инициализировать выполнение поиска с возвратом, чтобы найти другие решения. Пролог поддерживает специальный предикат *fail*, вызывающий неуспешное завершение, и, следовательно, инициализирует возврат. Действие предиката *fail* равносильно эффекту от сравнения 2 = 3 или другой невозможной подцели. *Fail* не может быть согласован (он всегда неуспешен), поэтому Пролог вынужден повторять поиск с возвратом. При поиске с возвратом он возвращается к последнему обращению, которое может произвести множественные решения. Такое обращение называют недетерминированным. Недетерминированное обращение является противоположностью детерминированному обращению, которое может произвести только одно решение. Предикат *fail* все время завершается неудачно, нет возможности для достижения подцели, расположенной после *fail*.

Пролог предусматривает возможность отсечения, которая используется для прерывания поиска с возвратом. Отсечение обозначается восклицательным знаком «!». Действует отсечение просто: через него невозможно совершить откат (поиск с возвратом). Отсечение помещается в программу таким же образом, как и подцель в теле правила. Однажды пройдя через отсечение, уже невозможно произвести откат к подцелям, расположенным в обрабатываемом предложении перед отсечением, и также невозможно возвратиться к другим предложениям, определяющим обрабатывающий предикат (предикат, содержащий отсечение). Существуют два основных случая применения отсечения.

* Если заранее известно, что определенные посылки никогда не приведут к осмысленным решениям (поиск решений в этом случае будет лишней тратой времени). При применении отсечения программа станет быстрее и экономичнее.
* Если отсечения требует сама логика программы для исключения из рассмотрения альтернативных подцелей.

1. Практическая часть
   1. Ход работы

Согласно заданию необходимо написать программу для выбора автомобиля из базы данных по различным признакам (например, по марке, цене, стране-производителю, году). Программа должна распознавать вопрос (что спрашивается) и выводить все подходящие автомобили.

Для создания программы была создана база данных, состоящая из предиката-факта ﻿*auto*(*string*, *real*, *string*, *integer*), принимающим параметры марка, стоимость, страна производства и год производства, а также предикат-правило ﻿*printAllAutos* ﻿(*string*, *real*, *string*, *integer*), выводящий на экран все автомобили, удовлетворяющие заданным параметрам. Для реализации интерфейса взаимодействия пользователя с программой были созданы предикаты-прафила ﻿*menu*(*integer*), принимающий номер выбранного пункта меню, а также ﻿*selectMenu*(), выводящий на экран доступные опции и вызывающий предикат *menu*(*integer*) с номером, выбранным пользователем.

Целевым предикатом является предикат *result.* Правило для данного предиката включает в себя только вызов предиката ﻿*selectMenu*().

Предикат ***selectMenu*()** выводит на экран список доступных опций с помощью конъюнкции стандартного предиката вывода ﻿*write*() и стандартного предиката перевода строки *nl*. Доступными опциями являются: 0 (выход), 1 (выбор авто по марке), 2 (выбор авто по цене), 3 (выбор авто по стране производства) и 4 (выбор авто по году производства). После вывода меню, с помощью конъюнкции со стандартным предикатом для считывания целого числа ﻿*readint*(), осуществляется запись выбранной опции в переменную *Option*. Далее значение переменной *Option* передаётся в предикат *menu*() для обработки пользовательского выбора. Если предикат *menu*() возвращает истинное значение, правило предиката *selectMenu()* заканчивается. В случае возвращения ложного значения, с помощью дизъюнкции с предикатом *selectMenu,* данный предикат выполняется рекурсивно.

Предикат ***menu()***возвращает истинное значение для параметра 0 (выход). Для остальных опций правила предиката включают в себя вывод строки о считывания параметра, соответствующего выбранной опции, с помощью стандартного предиката вывода *write*, считывания введённого значения в переменную *Brand* с помощью предиката *readln*() для 1-й опции, *Price* с помощью предиката *readreal*() для 2-й опции, *Country* помощью предиката *readln*() для 3-й опции и *Year* с помощью предиката *reading*() для 4-й опции, а также передачу введённых параметров в предикат ﻿*printAllAutos(string*, *real*, *string*, *integer).*

Предикат ***printAllAutos(﻿AutoBrand, AutoPrice, AutoCountry, AutoYear)***вызывает предикат-факт *auto*(*string*, *real*, *string*, *integer*) с полученными параметрами и в случае возвращения данным предикатом истинного значения, вызывает стандартный предикат *write*() для вывода на экран информации о автомобиле, а также стандартный предикат возврата *fail*, возвращающий исполнение на последнюю развилку, а именно, предикат *auto*(), для вывода на экран всех автомобилей из базы данных, удовлетворяющих переданным параметрам в предикат *printAllAutos(AutoBrand, AutoPrice, AutoCountry, AutoYear).* Таким образом предикат *printAllAutos()* всегда в конечном счёте возвращает ложное занчение, что используется для рекурсивного вызова предиката *selectMenu()*, описанного ниже.

Пример исполнения программы для вывода автомобилей по заданной марке представлен на рисунке 1.

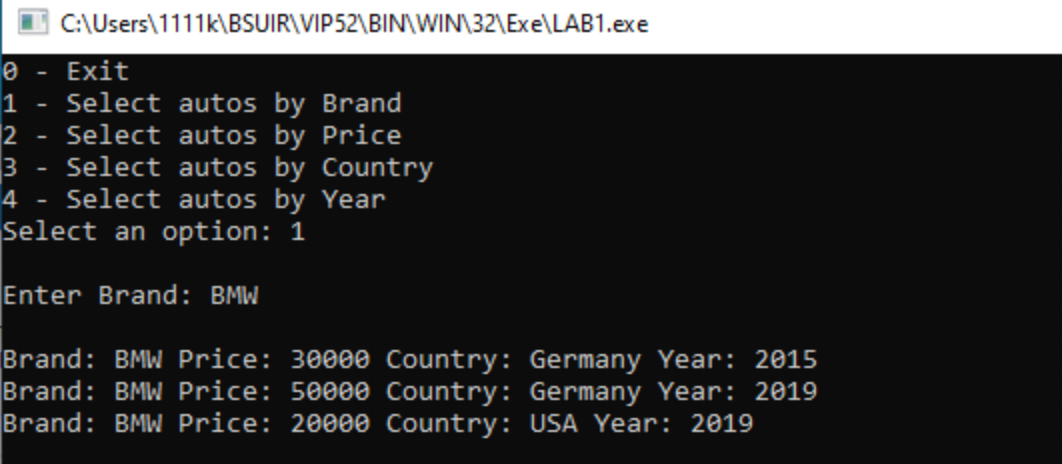


Рисунок 1 – Вывод автомобилей по заданной марке

Пример исполнения программы для вывода автомобилей по заданной цене представлен на рисунке 2.

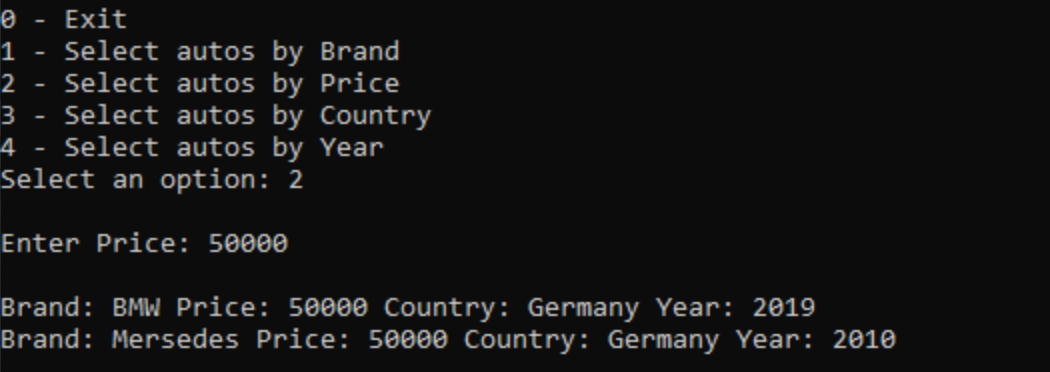


Рисунок 2 – Вывод автомобилей по заданной цене

Пример исполнения программы для вывода автомобилей по заданной стране производства представлен на рисунке 3.



Рисунок 3 – Вывод автомобилей по заданной стране производства

Пример исполнения программы для вывода автомобилей по заданному году производства представлен на рисунке 4.

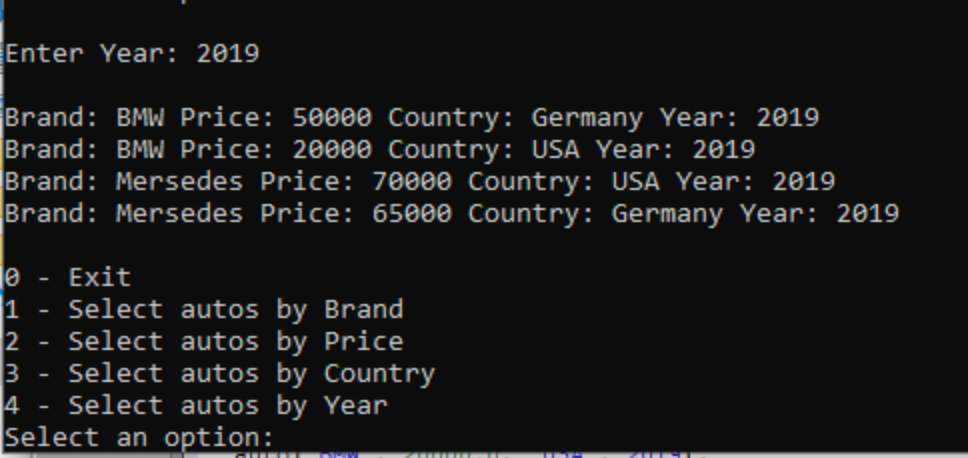


Рисунок 4 – Вывод автомобилей по заданному году производства

Ниже представлен код данной программы:

*﻿ predicates*

*nondeterm result*

*nondeterm menu(integer)*

*nondeterm selectMenu()*

*nondeterm auto(string, real, string, integer)*

*nondeterm printAllAutos(string, real, string, integer)*

*clauses*

*result:-*

*selectMenu().*

*selectMenu():-*

*write("0 - Exit"), nl,*

*write("1 - Select autos by Brand"), nl,*

*write("2 - Select autos by Price"), nl,*

*write("3 - Select autos by Country"), nl,*

*write("4 - Select autos by Year"), nl,*

*write("Select an option: "), readint(Option), nl,*

*menu(Option); nl,nl, selectMenu().*

*menu(0).*

*menu(1):-*

*write("Enter Brand: "),*

*readln(Brand),*

*printAllAutos(Brand, \_, \_, \_).*

*menu(2):-*

*write("Enter Price: "),*

*readreal(Price),*

*printAllAutos(\_, Price, \_, \_).*

*menu(3):-*

*write("Enter Country: "),*

*readln(Country),*

*printAllAutos(\_, \_, Country, \_).*

*menu(4):-*

*write("Enter Year: "),*

*readint(Year),*

*printAllAutos(\_, \_, \_, Year).*

*printAllAutos(AutoBrand, AutoPrice, AutoCountry, AutoYear):-*

*auto(AutoBrand, AutoPrice, AutoCountry, AutoYear), nl,*

*write("Brand: ", AutoBrand, " Price: ", AutoPrice, " Country: ", AutoCountry, " Year: ", AutoYear),*

*fail.*

*auto("BMW", 30000.0, "Germany", 2015).*

*auto("BMW", 50000.0, "Germany", 2019).*

*auto("BMW", 20000.0, "USA", 2019).*

*auto("Mersedes", 50000.0, "Germany", 2010).*

*auto("Mersedes", 70000.0, "USA", 2019).*

*auto("Mersedes", 65000.0, "Germany", 2019).*

*auto("LADA", 10000.0, "Russia", 2015).*

*auto("Citroen", 20000.0, "France", 2017).*

*goal*

*result.*

Вывод

В ходе выполнения данной лабораторной работы я изучил основные возможности языка Пролог и системы программирования *Visual* *Prolog*, изучил механизмы управления в программах на языке Пролог, а также приобрёл практические навыки их использования.