МИНИСТЕРСТВО ОБРАЗОВАНИЯ

Белорусский государственный университет

информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Лабораторная работа № 1 по дисциплине

ЭКСПЕРТНЫЕ СИСТЕМЫ

на тему

«ОСНОВНЫЕ СВЕДЕНИЯ О ЯЗЫКЕ ПРОЛОГ   
И СИСТЕМЕ ПРОГРАММИРОВАНИЯ VISUAL PROLOG»

Вариант 4

Выполнили:

ст. гр.820601

Шведов А.Р.

Проверил:

Тиханович Т. В.

Минск 2022

# Цель работы

Изучение основных возможностей языка Пролог и системы программирования Visual Prolog. Изучение механизмов управления в программах на языке Пролог.

# Условие

Имеются следующие факты:

– о том, в какой стpане pасположено пpедпpиятие, напpимеp: raspol("Оптpон","Россия")

– о том, кто кому поставляет свою пpодукцию (с указанием стоимости поставляемой продукции), напpимеp:

postav("Элтоp","Счетмаш","тpансфоpматоp",200)

В данном случае это означает, что пpедпpиятие "Элтоp" пос­тавляет свою пpодукцию (тpансфоpматоpы) пpедпpиятию "Счетмаш" на сумму 200 тыс. ден. ед.

1. Программа запрашивает название страны и товара. На экран должен выводиться список всех зарубежных предприятий, у которых предприятия данной страны закупают указанный товар (с указанием страны и стоимости закупаемого товара).

2. Программа запрашивает название пpедпpиятия и товаpа. Если пpедпpиятие закупает данный товаp за гpаницей, то программа должна сообщать название предприятия-поставщика, страны-поставщика и стоимость поставок; в противном случае должен выводиться ответ "нет".

# Теоретическая часть

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 конструкцией языка Пролог является *предикат*. Предикат - это 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.

**Предикаты-факты**. Аргументами предикатов-фактов являются константы. Предикаты-факты предназначены для записи некоторых утверждений, которые при выполне­нии программы считаются истинными.

**Предикаты-правила.** Они записываются в виде: <Предикат1> исти­нен, если истинны <Предикат2>, <Предикат3>, ..., <ПредикатN>. Здесь <Предикат1> называется головным (или заголовком), остальные – телом правила.

**Стандартные предикаты** входят в состав самого языка Пролог.

Три основных раздела программы на Прологе:

* predicates - список имен предикатов и типы их аргументов;
* goal – целевой пре­дикат (цель);
* clauses - перечень клозов, т. е. правил и фактов.

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.

Стандартный предикат fail вызывает искусственный возврат (состоя­ние неудачи).

Если предикат имеет несколько клозов, то при доказательстве одного из них остальные сохраняются в памяти ЭВМ как нерассмотренные. В случае неудачи при доказательстве какого-либо предиката, происходит возврат к таким нерассмотренным клозам. Если требуется исключить такую возможность, то применяется стандартный предикат отсечения (!).

**4. Выполнение работы**

Код программы:

*predicates*

*nondeterm vyvod1*

*nondeterm vyvod2*

*nondeterm raspol (string, string)*

*nondeterm postav (string, string, string, integer)*

*nondeterm poisk (string, string)*

*nondeterm poisk2 (string, string)*

*nondeterm end(string, string)*

*nondeterm repeat*

*goal*

*vyvod1, vyvod2.*

*clauses*

*vyvod1:- repeat, write ("Country: "), readln (C),*

*write ("Product: "), readln (P),*

*poisk (C, P),*

*nl, write("Continue? "), readln(Prod), Prod="n", nl.*

*poisk (Coun, Pr):- raspol (Manuf, Coun),*

*postav(Sup, Manuf, Pr, Stoim),*

*raspol (Sup, Coun2),*

*Coun<>Coun2,*

*write ("Supplier", " ", Sup," from country ", Coun2, " supply the product by price ", Stoim),*

*nl, fail.*

*poisk(\_,\_).*

*vyvod2:- repeat, write ("Manufacturer: "), readln (M),*

*write ("Product: "), readln (P),*

*poisk2 (M, P),*

*nl, write("Continue? "), readln(Prod), Prod="n", nl.*

*poisk2 (Man, Pr):- end(Man, Pr), postav(Sup, Man, Pr, Stoim),*

*raspol (Man, Coun),*

*raspol (Sup, Coun2),*

*Coun<>Coun2,*

*write ("Supplier", " ", Sup, " from country ", Coun2, " supply the product by price ", Stoim),*

*nl, fail.*

*poisk2 (\_,\_).*

*end(Man,Pr):- postav(Sup, Man, Pr, Stoim), !.*

*end(\_,\_):- write("NO"), nl.*

*repeat.*

*repeat:- repeat.*

*raspol ("Red", "China").*

*raspol ("Green", "Poland").*

*raspol ("Blue", "Poland").*

*raspol ("Black", "Germany").*

*raspol ("Purple", "Russia").*

*postav("Black","Green","table", 300).*

*postav("Black","Purple","book", 100).*

*postav("Red","Purple","table", 280).*

*postav("Red","Blue","table", 240).*

*postav("Red","Blue","book", 240).*

*postav("Red","Black","phone", 400).*

*postav("Green","Blue","phone", 500).*

*postav("Red","Blue","phone", 500).*

**5. Вывод**

В ходе лабораторной работы были изучены основные возможности языка Пролог и системы программирования *Visual Prolog*. Также была реализована программа в *Visual Prolog* в соответствии с полученным вариантом.