МИНИСТЕРСТВО ОБРАЗОВАНИЯ

Белорусский государственный университет

информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Лабораторная работа № 2 по дисциплине

ЭКСПЕРТНЫЕ СИСТЕМЫ

на тему

«ОБРАБОТКА СПИСКОВ В ПРОГРАММАХ НА ПРОЛОГЕ»

Вариант 4

Выполнили:

ст. гр. 820601

Гурин Н. С.

Кохнович М. О.

Проверил:

Тиханович Т. В.

Минск 2022

# Цель работы

Изучение возможностей представлений и обработки данных в программах на языке Пролог с использованием списков.

# Условие

1. Составить программу поиска номера заданного элемента в списке. Напpимеp, если введен список [1,5,2,7,4] и число 2, то должен быть получен ответ 3.

2. Имеются следующие факты:

- о том, какие предприятия расположены в указанной стране, напpимеp:

raspol("Россия",["Оптрон","Счетмаш","Кварц"])

raspol("Польша",["Элтор"])

raspol("США",["IBM","Apple"])

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

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

postav("Счетмаш","Оптpон","осциллогpаф",500)

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).

# Теоретическая часть

Рекурсия – это вызов предиката из тела самого предиката. В лабораторной работе 1 уже был рассмотрен пример рекурсии: использование предиката repeat.

Рекурсия обычно применяется при обработке списков, строк (например, для поиска и замены подстроки), при вычислениях (например, вычисление сумм, факториала) и в ряде других случаев.

Примечание. Можно сказать, что рекурсивные вызовы предикатов в рекурсивных программах образуют стек.

Необходимо обратить внимание, что для уменьшения переменной на единицу используется запись N1=N-1. Запись типа N=N-1 в Прологе ***недопустима***, так как она рассматривается как предикат *сравнения* N и N-1; очевидно, что он всегда принимает значение "ложь". Присвоить связанной переменной новое значение в Прологе невозможно.

Список представляет собой основную структуру данных в Прологе. Список - это последовательность из произвольного числа элементов не­которого типа. Ограничений на длину списка нет. Тип данных "список" объявляется в программе на Прологе следующим образом:

domains

списковый\_тип = тип\*

где "тип" - тип элементов списка; это может быть как стандартный тип, так и нестандартный, заданный пользователем и объявленный в разделе domains ранее.

Ниже приведен пример описания предиката, аргументом которого является список целых чисел:

domains

int\_list = integer\*

predicates

nondeterm numbers (int\_list)

Ввод списков с клавиатуры выполняется стандартным предикатом readterm (<тип>, <имя\_переменной>), где <тип> - списковый тип, который должен быть предварительно объявлен в разделе domains. Например, для ввода списка, объявление которого (int\_list) показано выше, можно использовать следующий предикат: readterm (int\_list, L). При вводе список необходимо набирать точно так же, как он записывается, т.е. в квадратных скобках, через запятую, без пробелов. Если элементы списка - строки, то они должны заключаться в двойные кавычки.

Вывод списка на экран выполняется (в простейшем случае) предика­том write (L), где L - список.

В языке Пролог для удобства обработки списков введены понятия головы и хвоста списка. Голова - это первый элемент списка; хвост - вся остальная часть списка. Таким образом, *хвост списка* сам является *списком*. Для представления списка в виде головы и хвоста используется следующая запись: [H|T], где H - пере­менная, связываемая с головой (первым элементом) списка, а T - пере­менная, связываемая с хвостом списка (т.е. также со списком).

Пустой список обозначается так: []. Запись вида [X] обозначает список из одного элемента.

Рекомендации по составлению программ со списками:

* в операциях со списками практически всегда используется рекурсия;
* обычно предикаты для реализации операций со списками име­ют несколько клозов. Первый из них относится к некоторому простейшему случаю (например, к операции с пустым списком, со списком из одного элемента, с головой списка). Последующие клозы относятся к бо­лее общим случаям и имеют следующее назначение: если список не соответству­ет простейшему случаю, рассмотренному в первом клозе, то его следует уменьшить на один элемент (обычно - исключить голову) и применить ре­курсию к укороченному списку;
* предикаты, предназначенные для получения одного списка из другого, *всегда* имеют *не менее двух аргументов*: один - исходный спи­сок, другой - список-результат.

К числу основных операций со списками можно отнести следующие: проверка принадлежности к списку, добавление элемента в список, удаление эле­мента из списка, присоединение одного списка к другому, вывод элемен­тов списка на экран. На основе этих операций реализуются другие, бо­лее сложные операции со списками.

**4. Выполнение работы**

Код программы:

*domains*

*int\_list=integer\**

*string\_list=string\**

*predicates*

*nondeterm task1*

*nondeterm task2*

*nondeterm find(int\_list, integer, integer)*

*nondeterm find2(string, string)*

*nondeterm find3(string\_list, string, string)*

*nondeterm check(string\_list, string)*

*nondeterm raspol(string, string\_list)*

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

*nondeterm repeat*

*goal*

*task1, task2.*

*clauses*

*task1:- repeat,*

*write("Enter array:"), nl,*

*readterm(int\_list, L),*

*write("Enter elem:"), nl,*

*readint(E),*

*find(L, E, I),*

*write("Index: ", I), nl,*

*write("Continue?"),*

*readln(N),*

*N = "n", nl.*

*find([H|T], E, I):- H <> E, find(T, E, II), I = II + 1.*

*find([H|T], E, I):- H = E, I = 0.*

*find([], E, 0).*

*task2:- repeat,*

*write("Enter country:"), nl,*

*readln(C),*

*write("Enter product"), nl,*

*readln(P),*

*find2(C, P),*

*write("Continue? "),*

*readln(N),*

*N = "n", nl.*

*find2(Country, Product):- raspol(Country, Enterprises),*

*find3(Enterprises, Product, Country),*

*nl, fail.*

*find2(\_,\_).*

*find3([H|T], Product, Country):- find3(T, Product, Country),*

*postav(Provider, H, Product, Price),*

*raspol(Country, Enterprises),*

*check(Enterprises, Provider),*

*write(Provider, " provides ", H, " with ", Product, " by ", Price), nl, fail.*

*find3([], Product, Country):- !.*

*find3(\_,\_,\_).*

*check([H|T], Enterprise):- H <> Enterprise, check(T, Enterprise).*

*check([], Enterprise).*

*raspol("Russia", ["Blue", "White", "Red"]).*

*raspol("Belarus", ["BlackRed", "Green"]).*

*raspol("Poland", ["White", "DarkRed" ]).*

*raspol("German", ["Black", "Yellow", "Orange"]).*

*postav("Green", "Blue", "table", 100).*

*postav("Green", "Blue", "book", 200).*

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

*postav("BlackRed", "Orange", "book", 400).*

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

*postav("Black", "Green", "table", 600).*

*postav("Black", "White", "phone", 700).*

*postav("Orange", "White", "table", 800).*

*postav("Yellow", "Red", "phone", 900).*

*postav("Green", "DarkRed", "book", 111).*

*postav("White", "Yellow", "table", 222).*

*postav("Blue", "DarkRed", "table", 333).*

*postav("White", "Orange", "phone", 444).*

*postav("White", "BlackRed", "book", 555).*

*repeat.*

*repeat:- repeat.*

**5. Вывод**

В ходе лабораторной работы были изучены возможности представления и обработки данных в программах на языке Пролог с использованием списков. Также были изучены особенности применения рекурсии в реализации программ на языке Пролог.