# Отсечение и его использование.

*Лекция 4 (Часть 3).*

**Совместное использование рекурсии, отсечения и отката при организации циклов.**

***Специальности : 230105, 010501***

# Применение циклов в Пролог-программах.

**Основная сфера применения циклов в Прологе - организация повторяющихся действий. При организации циклов широко используются внелогические предикаты. Такие предикаты :**

**-Обладают четким процедурным смыслом при отсутствии декларативного;**

**-Всегда согласуются с БД и не могут быть пересогласованы;**

**-Обладают побочным эффектом;**

**-Реализуются либо на ассемблере, либо на алгоритмических языках (но не на Прологе !);**

**-Зависят от конкретной реализации Пролога.**

**Существует три основных вида внелогических предикатов : предикаты ввода/вывода, модификации программы, системные предикаты.**

**При организации циклов побочный эффект внелогических предикатов используется для циклического выбора очередной доказываемой цели, что, в частности, необходимо при написании оболочек.**

# Рекурсия и циклы.

**Для рекурсивных циклов характерно управление ходом циклических вычислений с помощью стандартных средств организации рекурсии в Прологе. В алгоритмических языках подобный цикл реализуется с помощью конструкции *while* :**

**Турбо-Паскаль :**

```
while not(Ch='A') do
   Ch:=Readkey;
```

**Турбо-Пролог :**

```
readloop:-readchar(Ch),
          not(Ch=65),
          readloop.
readloop.
```

# Циклы, управляемые отказами.

**Наряду с рекурсией, для управления ходом циклических вычислений может быть использован fail - откат ввиду неуспеха (отказ). Реализованные подобным образом циклы называются циклами, управляемыми отказом. Такие циклы бывают полезными лишь при совместном использовании с внелогическими предикатами. В алгоритмических языках данные циклы реализуются с помощью конструкции *repeat-until*. В качестве примера рассмотрим вариант записи представленного на плакате 2 цикла.**

**Турбо-Паскаль :**

```
repeat
      Ch :=Readkey;
until Ch='A'
```

**Турбо-Пролог :**

```
repeat.
repeat :- repeat.
readloop :-
   repeat,readchar(Ch),
   readloop(Ch),!.
readloop(Ch) :- Ch=65,!.
readloop(Ch) :- fail.
```

# repeat-циклы.

**Цикл, управляемый отказом и использующий предикат *repeat*, называется *repeat*-циклом.**

**Предикат *repeat* есть незавершающийся предикат, задаваемый с помощью представленной на плакате 4 минимальной рекурсивной процедуры.**

**В отличие от программы на плакате 2, в данной программе доказательство ЦУ *readloop(Ch)* приводит к неуспеху, если только Ch не есть ASCII - код символа ‘A’. Неуспех вызывает возврат к цели *repeat*, цель выполняется и затем считывается с клавиатуры следующий символ. Отсечение в определении предиката *readloop* гарантирует от позднейшего повторения цикла *repeat*.**

# Некоторые рекомендации по использованию repeat-циклов.

***repeat*-циклы применяются в Прологе для описания интерактивного взаимодействия с внешней средой путем повторяющегося ввода и (или)вывода.**

**В repeat-цикле обязательно должен присутствовать предикат, гарантированно приводящий к безуспешным вычислениям, определяющим продолжение итерации. В приведенном на плакате 4 примере это цель *readchar(Ch)*. Такой предикат вычисляется успешно лишь в момент выхода из цикла.**

**Можно сформулировать полезное эвристическое правило построения *repeat*-циклов : в теле правила, содержащего цель *repeat*, должно быть отсечение, предохраняющее от незавершающихся вычислений при возврате в цикл.**

# Разработка интерактивных программ с применением циклов.

**Наиболее распространенный случай использования циклов при программировании на Прологе - организация интерактивного взаимодействия пользователя и программы. В качестве примера рассмотрим простейшее меню.**

```
/* Использование рекурсивного цикла для организации меню */
menu:-
        makewindow(1,2,7,"Использование циклов для
                          организации меню",0,0,12,50),
          write("I - ввод списка сотрудников"),nl,
          write("M - модификация списка сотрудников"),nl,
          write("P - печать списка сотрудников"),nl,
          write("E - выход из программы"),nl,
          write("Введите литеру выбора"),nl,
          readchar(Ch),
          not(Ch=69),
          choice(Ch),
          menu.
menu.
```

# Применение repeat-цикла при организации меню.

**Рассмотрим альтернативный вариант определения предиката menu из приведенного на плакате 7 примера с помощью *repeat*-цикла.**

```
/* Использование цикла, управляемого отказом,
   для организации меню */
menu :-
        repeat,
        makewindow(1,2,7,"Использование циклов
                          для организации меню",0,0,12,50),
        write("I - ввод списка сотрудников"),nl,
        write("M - модификация списка сотрудников"),nl,
        write("P - печать списка сотрудников"),nl,
        write("E - выход из программы"),nl,
        write("Введите литеру выбора"),nl,
        readchar(Ch),menu(Ch),!.
menu(Ch) :- Ch=69,!.
menu(Ch) :- choice(Ch),fail.
```

**В отличие от программы на плакате 7, в приведенном здесь варианте организации того же меню доказательство ЦУ menu(Ch) приводит к неуспеху, если Ch не есть ASCII-код символа ‘E’. Неуспех вызывает возврат к цели repeat, цель выполняется, на экране отображается меню и затем считывается с клавиатуры очередной символ.**

# Организация ввода и вывода списков.

**Для организации ввода и вывода списков с учетом их рекурсивной природы используются рекурсивные циклы.**

**/* Пример : ввод списка произвольной длины */**

```
read_list([],0).
read_list([H|T],Len):-
     readint(H),Len1=Len-1,read_list(T,Len1).
```

**/* Печать заданного количества первых элементов списка */**

```
write_list(_,0).
write_list([H|T],Len):-write(H),nl,Len1=Len-1,
                  write_list(T,Len1).
```

# Выводы.

**Рекурсивные циклы предпочтительнее *repeat*-циклов, поскольку последние не имеют логической интерпретации.**

**На практике repeat-циклы часто необходимы при выполнении большого объема вычислений, особенно в реализациях Пролога без оптимизации остатка рекурсии и (или) без сборки мусора.**

**Обычно явный отказ приводит к требованию некоторого зависящего от реализации объема памяти.**

# Литература.


**Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог : Пер. с англ. - М.: Мир, 1990. С. 141, 147-153**

**Клоксин У., Меллиш К. Программирование на языке Пролог : Пер. с англ. - М.: Мир, 1987. С. 119**

**Маплас Дж. Реляционный язык Пролог и его применение : Пер. с англ. - М.:Наука, 1990. С. 129, 154, 173, 200-203**

**Доорс Дж. Пролог - язык программирования будущего : Пер. с англ. - М.: Финансы и статистика, 1990. С. 84-85**