**The laboratory work 5**

A recursion procedure is a procedure that calls itself until it reaches some condition that is going to stop it. This condition is usually called a border condition. A recursion rule always consists of at least two parts one of which is non-recursive. This part defines a border condition.

In a recursive procedure, there is no problem storing the results of its execution because any computed values can be passed from one call to another as arguments of a recursively called predicate. Recursion is an effective way to solve problems that contain a subproblem of the same type.

**Example1.**

The database includes the following facts:

roditel(ivan,oleg).

roditel(inna,oleg).

roditel(oleg,dima).

roditel(oleg,marina).

It is required to create a recursive rule predot and define all ancestors and their descendants.

**Solution:**

DOMAINS

name=string

PREDICATES

roditel(name,name)

predok(name,name)

CLAUSES

roditel(ivan,oleg).

roditel(inna,oleg).

roditel(oleg,dima).

roditel(oleg,marina).

predok(X,Z):-roditel(X,Z). *% non-recursive part of the rule*

predok(X,Z):-roditel(X,Y),  *% recursive part of the rule*

predok(Y,Z).

GOAL

predok(X,Y),

write("Рredok -",X," Еgo potomok-",Y), nl, fail.

**Results of the program’s execution:**

Рredok -ivan Еgo potomok-oleg

Рredok -inna Еgo potomok-oleg

Рredok -oleg Еgo potomok-dima

Рredok -oleg Еgo potomok-marina

Рredok -ivan Еgo potomok-dima

Рredok -ivan Еgo potomok-marina

Рredok -inna Еgo potomok-dima

Рredok -inna Еgo potomok-marina

**Example 2.** Solving a factorial

**Solution:**

PREDICATES

fact(integer, integer)

CLAUSES

fact(0,1):-!. *% A factorial of zero equals 1*

fact(N,F):- N1=N-1, *% decreasing N by one,*

fact(N1,F1)*, % calculating a factorial of a new number,*

F=N\*F1. *% then multiply it by N*

GOAL

write("N="),readint(N),fact(N,F),write("F=",F),nl.

**Results of the program’s execution:**

The 1st case:

N=0

F=1

The 2nd case:

N=1

F=1

The 3rd case:

N=4

F=24

**Example 3**

Creating a program for calculating Y=Xn, X, n – целые числа

**Solution:**

Create a rule **stepen** that consists of 3 parts:

The 1st part (non-recursive) defines Х0=1.

The 2nd part (recursive) calculates Хn for a positive number n.

The 3rd par (recursive) calculates Хn for a negative n (a necessary condition Х<>0 is added)

PREDICATES

stepen(real,real,real)

CLAUSES

stepen(X,0,1):-!.

stepen(X,N,Y):-N>0,N1=N-1,stepen(X,N1,Y1),Y=Y1\*X,!.

stepen(X,N,Y):-X<>0,K=-N,stepen(X,K,Z),Y=1/Z.

GOAL

write("X="),readreal(X),

write("N="),readreal(N),

stepen(X,N,Y),write("Y=",Y),nl.

**Results of the program’s execution:**

The 1st case:

X=3

N=2

Y=9

The 2nd case:

X=2

N=-2

Y=0.25

**Example 4**. Hanoi tower

There are three bars: A, B and C. On the rod A, N disks of different diameters are put on each other in order of decreasing diameters. You need to move the discs from the rod A to the rod C using B as an auxiliary, if you can move only one disk and you cannot put a larger disk on a smaller one.

**The solution:**

We create a rule move that defines an order of moving disks.

The 1st (non-recursive) part of the rule determines the action if there is 1 disc on the rod.  
The 2nd (recursive) part of the rule moves first the upper N-1 disk to the rod B, using C as an auxiliary, then the remaining disk to the rod C and finally the disks from the rod B to C, using A as an auxiliary disk.

PREDICATES

move(integer,char,char,char)

CLAUSES

move(1,A,B,C):-

write("Move a disk from ",A," to  ",C),nl,!.

move(N,A,B,C):-

M=N-1,move(M,A,C,B),

write("Move a disk from ",A," to  ",C),nl,

move(M,B,A,C).

GOAL

write("Hanoi tower"), nl,

write("A number of disks:"), readint(N),nl,

move(N,'A','B','C').

**Results of the program’s execution:**

Hanoi tower

A number of disks: 3

Transfer disk from A to C  
 Transfer disk from A to B  
 Transfer disk from C to B  
 Transfer disk from A to C   
 Transfer disk from B to A  
 Transfer disk from B to C

Transfer disk from A to C

**Assignments**:

1. Find the sum 1+2+3+..+N
2. Find the sum of integer even numbers from 2 to N
3. Find the sum of integer odd numbers from 1 to N
4. Find the multiplication of: 2\*4\*6\*...\*26
5. Find the multiplication of: 1\*3\*5\*...\*11