

Aplicatii Prolog – Functii recursive
1. Suma numerelor din intervalul [1,n]

$$suma(n) = \begin{cases} 0, n = 0 \\ n + suma(n-1), n > 0 \end{cases}$$

main.pro – v1 folosind *if*

```
implement main
  open core
class predicates
  suma : (integer N, integer S) procedure (i,o).
clauses
  suma(N, S) :-
    if N=0 then
      S=0
    else
      suma(N-1, S1),
      S = N + S1
    end if.
clauses
  run() :-
    console::init(),
    X=10,
    suma(X,S),
    stdio::writef("Suma numerelor din intervalul [1, %] = %", X, S),
    succeed().
end implement main
goal
  mainExe::run(main::run).
```

main.pro – v2 folosind Prolog

```
implement main
  open core
class predicates
  suma : (integer N, integer S) procedure (i,o).
clauses
  suma(0, S) :- S=0, !.
  suma(N, S) :-
    suma(N-1, S1),
```

```

    S = N + S1.
clauses
    run():-
        console::init(),
        X=10,
        suma(X,S),
        stdio::writef("Suma numerelor din intervalul [1,%] = %",X,S),
        succeed().
end implement main
goal
    mainExe::run(main::run).
```

Rulare

```
Suma numerelor din intervalul [1,10] = 55
```

2. 2 la puterea n (2^n)

$$2^n = \begin{cases} 1, & n = 0 \\ 2 \cdot 2^{n-1}, & n > 0 \end{cases}$$

main.pro – v1 folosind Prolog

```

implement main
    open core
class predicates
    doiLaPutere : (integer N, integer P) procedure (i,o).
clauses
    doiLaPutere(0,P) :- P=1, !.
    doiLaPutere(N, P) :-
        doiLaPutere(N-1, P1),
        P = 2 * P1.
clauses
    run():-
        console::init(),
        X=8,
        doiLaPutere(X,DoiLaPutere),
        stdio::writef("2^% = %",X,DoiLaPutere),
        succeed().
end implement main
goal
    mainExe::run(main::run).
```

main.pro – v2 folosind *if*

```
implement main
  open core
class predicates
  doiLaPutere : (integer N, integer P) procedure (i,o).
clauses
doiLaPutere(N, P) :-
  if N=0 then
    P=1
  else
    doiLaPutere(N-1, P1),
    P = 2 * P1
  end if.
clauses
run() :-
  console::init(),
  X=8,
  doiLaPutere(X,DoiLaPutere),
  stdio::writef("2^% = %",X,DoiLaPutere),
  succeed().
end implement main
goal
mainExe::run(main::run).
```

Rulare

```
2^8 = 256
```

3. Suma cifrelor unui numar

$$sumaCifre(n) = \begin{cases} 0, n = 0 \\ ultimaCifra + sumaCifre(numarFaraUltimaCifra), n > 0 \end{cases}$$

$$sumaCifre(n) = \begin{cases} 0, n = 0 \\ n \bmod 10 + sumaCifre(n \div 10), n > 0 \end{cases}$$

main.pro – v1 folosind *if*

```
implement main
  open core
class predicates
  sumaCifre : (integer N, integer S) procedure (i,o).
clauses
sumaCifre(N, S) :-
  if N=0 then
```

```

        S=0
    else
        sumaCifre(N div 10, S1),
        S = N mod 10 + S1
    end if.
clauses
run():-
    console::init(),
    X=12345,
    sumaCifre(X,SumaCifre),
    stdio::writef("SumaCifre(%) = %",X, SumaCifre),
    succeed().
end implement main
goal
mainExe::run(main::run).
```

main.pro – v2 folosind Prolog

```

implement main
    open core
class predicates
    sumaCifre : (integer N, integer S) procedure (i,o).
clauses
    sumaCifre(0, S) :- S=0 , !.
    sumaCifre(N, S) :-
        sumaCifre(N div 10, S1),
        S = N mod 10 + S1.
clauses
    run():-
        console::init(),
        X=12345,
        sumaCifre(X,DoiLaPutere),
        stdio::writef("SumaCifre(%) = %",X,DoiLaPutere),
        succeed().
end implement main
goal
mainExe::run(main::run).
```

Rulare

```
SumaCifre(12345) = 15
```

4. Numere Fibonacci

$$Fibonacci(n) = \begin{cases} 0, n = 0 \\ 1, n = 1 \\ Fibonacci(n-2) + Fibonacci(n-1), n \geq 2 \end{cases}$$

main.pro – v1 folosind **if**

```
implement main
  open core
class predicates
  fibonacci : (integer N, integer F) procedure (i,o).
clauses
  fibonacci(N, F) :-
    if N=0 or N=1 then
      F=N
    else
      fibonacci(N-1, F1),
      fibonacci(N-2, F2),
      F = F1 + F2
    end if.
clauses
  run() :-
    console::init(),
    N=6,
    fibonacci(N,Fibo),
    stdio::writef("Fibo(%) = %",N,Fibo),
    succeed().
end implement main
goal
  mainExe::run(main::run).
```

main.pro – v2 folosind **Prolog**

```
implement main
  open core
class predicates
  fibonacci : (integer N, integer F) procedure (i,o).
clauses
  fibonacci(0, F) :- F=0, !.
  fibonacci(1, F) :- F=1, !.
  fibonacci(N, F) :-
    fibonacci(N-1, F1),
```

```

        fibonacci(N-2, F2),
        F = F1 + F2.
clauses
    run() :-
        console::init(),
        N=6,
        fibonacci(N,Fibo),
        stdio::writef("Fibo(%) = %",N,Fibo),
        succeed().
end implement main
goal
    mainExe::run(main::run).
```

Rulare

```
Fibo(6) = 8
```

5. Suma de factoriale

$$S = 1! + 2! + 3! + \dots + n!$$

main.pro folosind *if*

```

implement main
    open core
class predicates
    factorial : (integer N, integer F) procedure (i,o).
    suma : (integer N, integer S) procedure (i,o).
clauses
    factorial(N,F) :-
        if N=0 then
            F=1
        else
            factorial(N-1,F1),
            F = N*F1
        end if.
    suma(N,S) :-
        if N=0 then
            S=0
        else
            factorial(N,F1),
            suma(N-1,S1),
            S = F1 + S1,
            stdio::writef("%! + ",N)
        end if.
```

```
clauses
  run():-
    console::init(),
    suma(5,S),
    stdio::writef(" = %",S),
    succeed().
end implement main
goal
  mainExe::run(main::run).
```

Rulare

```
1! + 2! + 3! + 4! + 5! + = 153
```

Tema:

Sa se implementeze problema 6 fara if. (**folosind Prolog**)

6. Afisarea a mai multor solutii ale problemelor anterioare

De exemplu, vrem sa afisam primele 10 puteri ale lui 2

main.pro **folosind Prolog**

```
implement main
  open core
class predicates
  doiLaPutere : (integer N, integer P) procedure (i,o).
  afisare : (integer N) procedure (i).
clauses
  doiLaPutere(0,P) :- P=1, !.
  doiLaPutere(N, P) :-
    doiLaPutere(N-1, P1),
    P = 2 * P1.
  afisare(0) :-
    doiLaPutere(0,P),
    stdio::writef("2^0 = %\n",P),
    !.
  afisare(N) :-
    afisare(N-1),
    doiLaPutere(N,P),
    stdio::writef("2^% = %\n",0,P).
clauses
  run():-
    console::init(),
    afisare(10),
```

```
        succeed().  
end implement main  
goal  
    mainExe::run(main::run).
```

Rulare

```
2^0 = 1  
2^1 = 2  
2^2 = 4  
2^3 = 8  
2^4 = 16  
2^5 = 32  
2^6 = 64  
2^7 = 128  
2^8 = 256  
2^9 = 512  
2^10 = 1024
```

Tema:

1. Rescrieti problema 6 utilizand instructiunea **if**.
2. In mod similar, afisati si pentru restul problemelor din laborator mai multe solutii, utilizand o functie recursiva de afisare.