

Lab. 3: Más sobre ciclos básicos en Prolog

12 de mayo de 2019

Integrantes

- Leonardo Henry Añez Vladimirovna
- Gerson Oliva Rojas
- Pedro Luis Caricari Torrejón
- Erick Edwing Vidal Céspedes

Porcentaje Completado: 100 %

Comentario(s):

Source Code:

1. mostrarDivisoresDesc(n) : Procedimiento que muestra los divisores del entero n. Muestra desde el n a 1.

Python (Iterativo)

```
1 def mostrarDivisoresDesc(n):
2     i = n
3     while(i>=1):
4         if(n%i==0):
5             print(i)
6             i=i-1
```

Python (Recursivo)

```
1 def mostrarDivisoresDesc(n):
2     recursivo_mostrarDivisoresDesc(n,1)
3
4 def recursivo_mostrarDivisoresDesc(n,i):
5     if(i<=n):
6         recursivo_mostrarDivisoresDesc(n,i+1)
7         if(n%i==0):
8             print(i)
```

Prolog

```
1 mostrarDivisoresDesc:- read(N),
2     divisores(N,1).
3
4 divisores(N,I):- I>N,! .
5 divisores(N,I):- N mod I == 0,
6     I1 is I+1,
7     divisores(N,I1),
8     write(I),nl.
9 divisores(N,I):- I1 is I+1,
10    divisores(N,I1).
```

2. `mostrarDivisoresComunes(n, m)` : Procedimiento que muestra los divisores comunes de los entero n y m .
3. `mostrarDivisoresPares(n)` : Procedimiento que muestra los divisores pares de n .
4. `mostrarDivisoresImpares(n)` : Procedimiento que muestra los divisores impares de n .
5. `mostrarDivisores(n, a, b)` : Procedimiento que muestra los divisores de n , comprendidos entre a y b inclusive.
6. `primo(n)` : Función lógica que devuelve True, si el entero n es número primo.

Python (Iterativo)

```

1 def ite_primo(n):
2     if(n==1):
3         return False
4     i = 2
5     while(i<n):
6         if(n%i==0):
7             return False
8         i = i + 1
9     return True

```

Python (Recursivo)

```

1 def rec_primo(n):
2     return primo(n,2)
3
4 def primo(n,i):
5     if(n==1):
6         return False
7     if(i>n/2):
8         return True
9     if(n%i==0):
10        return False
11    return primo(n,i+1)

```

Prolog

```

1 primo:- read(N),
2     primo(N).
3
4 primo(N):- primo(N,2).
5 primo(N,I):- I > N // 2.
6 primo(N,I):- N mod I =\= 0,
7     I1 is I+1,
8     primo(N,I1).

```

7. `proximoPrimo(n)` : Función que devuelve, el siguiente primo después de n . Si n es primo, devuelve n .

Python (Iterativo)

```

1 def proximoPrimo(n):
2     # La condicion n<=10e6 es usada como un limite por la complejidad
3     # de un test de primalidad (el mejor por ahora AKS, O((log n)^6))
4     # y como no es el core de la materia no entramos en detalle :P
5
6     # Hacemos uso de la funcion primo(n)
7     while(primo(n)!=True and n<=1000000):
8         n = n + 1
9     return n

```

Python (Recursivo)

```

1 def proximoPrimo(n):
2     # De igual manera que el anterior usamos un limite
3     if(n>1000000):

```

```

4     return -1
5     if(primo(n)):
6         return n
7     return proximoPrimo(n+1)

```

Prolog

```

1 proximoPrimo(N,R):-
2     N =< 1000000,
3     primo(N),
4     R is N.
5 proximoPrimo(N,R):- N1 is (N+1),
6     proximoPrimo(N1,R).

```

8. anteriorPrimo(n) : Función que devuelve, el anterior primo antes de n. Si n es primo, devuelve n.

Python (Iterativo)

```

1 def anteriorPrimo(n):
2     # En caso de ser 1
3     if(n<2):
4         return -1
5     # Hacemos uso de la funcion primo(n)
6     while(primo(n)!=True and n>1):
7         n = n - 1
8     return n

```

Python (Recursivo)

```

1 def anteriorPrimo(n):
2     # De igual manera que el anterior usamos un limite
3     if(n<2):
4         return -1
5     if(primo(n)):
6         return n
7     return anteriorPrimo(n-1)

```

Prolog

```

1 anteriorPrimo(N,R):-
2     N > 2,
3     primo(N),
4     R is N.
5 anteriorPrimo(N,R):- N1 is (N-1),
6     anteriorPrimo(N1,R).

```

9. mostrarPrimos(a, b): Procedimiento que muestra los número primos entre a y b, inclusive.

Python (Iterativo)

```

1 def mostrarPrimos(a,b):
2     while(proximoPrimo(a)<=b):
3         a = proximoPrimo(a)
4         print(a)
5         a = a + 1

```

Python (Recursivo)

```

1 def mostrarPrimos(a,b):
2     if(proximoPrimo(a)<=b):
3         a = proximoPrimo(a)
4         print(a)
5         a=a+1
6         mostrarPrimos(a,b)

```

Prolog

```

1 mostrarPrimos(A,B):- A >= B,!.
2
3 mostrarPrimos(A,B):- proximoPrimo(A,R),

```

```

4         P is R,
5         B >= P,
6         X is R+1,
7         write(R),nl,
8         mostrarPrimos(X,B).
9

```

12. `mostrarFactoriales(n)`: Procedimiento que muestra los factoriales de 1 a n.

Python (Iterativo)

```

1 def mostrarFactoriales(n):
2     f = 1
3     i = 1
4     while(i<=n):
5         f = f*i
6         print(str(i)+"!="+str(f))
7         i = i + 1

```

Python (Recursivo)

```

1 def mostrarFactoriales(n):
2     if(n==0 or n==1):
3         print(str(n)+"!=1")
4         return 1
5     f = n*mostrarFactoriales(n-1)
6     print(str(n)+"!="+str(f))
7     return f

```

Prolog

```

1 mostrarFactorial:- read(N),mostrarFactorial(N,_).
2
3 mostrarFactorial(0,1):-!.
4 mostrarFactorial(N,R):- N1 is N-1,
5     mostrarFactorial(N1,R1),
6     R is N*R1,
7     write(N),
8     write("!="),
9     write(R),nl.

```

13. `mostrarCoefBin(n)` : Procedimiento que muestra los coeficientes binomiales de un binomio elevado a la n.

14. `mostrarFib(n)`: Procedimiento que muestra los primeros n términos de la secuencia de Fibonacci.

Python (Iterativo)

```
1 def ite_mostrarFib(n):
2     i = 1
3     a = 0
4     b = 1
5     while(i<n):
6         print(b)
7         f = a + b
8         a = b
9         b = f
10        i = i + 1
```

Python (Recursivo)

```
1 def fibo(n):
2     if(n==1):
3         return 1
4     if(n==2):
5         return 1
6     x = (fibo(n-1)+fibo(n-2))
7     return x
8
9 def mostrarFib(n):
10    for i in range(1,n+1):
11        print(fibo(i))
```

Prolog

```
1 fibo:-
2     read(N),
3     fib(N,_,_).
4
5 fib(1, 1, 0):- write(1),nl.
6 fib(X, Y1, Y2) :- X > 1,
7     X1 is X - 1,
8     fib(X1, Y2, Y3),
9     Y1 is Y2 + Y3,
10    write(Y1),nl.
```

1	3	1	0	0	0	0	200	0	3	1	0	1	0	0	180	0	0	1	0	1	3	0	150
1	1	0	1	0	0	0	100	0	1	0	1	1	0	0	80	0	0	0	1	1	1	0	70
-1	-0	0	0	1	0	0	-20	1	0	-0	-0	-1	-0	-0	20	1	0	0	0	-1	0	0	20
-0	-1	0	0	0	1	0	-10	0	-1	0	0	0	1	0	-10	-0	1	-0	-0	-0	-1	-0	10
-30	-50	0	0	0	0	1	0	0	-50	0	0	-30	0	1	600	0	0	0	0	-30	-50	1	1100
0	0	0.333333	0	0.333333	1	0	50	0	0	0.5	-0.5	0	1	0	40	'x1': 50.0, 'x2': 50.0, 'max':							
0	0	-0.333333	1	0.666667	0	0	20	0	0	-0.5	1.5	1	0	0	30								
1	0	0	0	-1	0	0	20	1	0	-0.5	1.5	0	0	0	50								
0	1	0.333333	0	0.333333	0	0	60	0	1	0.5	-0.5	0	0	0	50								
0	0	16.6667	0	-13.3333	0	1	3600	0	0	10	20	0	0	1	4000								
4000.0																							