Tema# 3

Lab. 3: Más sobre ciclos básicos en Prolog

28 de abril de 2019

Integrantes

- Leonardo Henry Añez Vladmirovna
- 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)

```python
def mostrarDivisoresDesc(n):
  i = n
  while(i>=1):
    if(n%i==0):
      print(i)
    i=i-1
```

   Python (Recursivo)

```python
def mostrarDivisoresDesc(n):
  recursivo_mostrarDivisoresDesc(n,1)

def recursivo_mostrarDivisoresDesc(n,i):
  if(i<=n):
    recursivo_mostrarDivisoresDesc(n,i+1)
    if(n%i==0):
      print(i)
```

   Prolog

```
mostrarDivisoresDesc:- read(N),
          divisores(N,1).

divisores(N,I):- I>N,!.
divisores(N,I):- N mod I =:= 0,
         I1 is I+1,
         divisores(N,I1),
         write(I),nl.
divisores(N,I):- I1 is I+1,
         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)

```python
def ite_primo(n):
  if(n==1):
    return False
  i = 2
  while(i<n):
    if(n%i==0):
      return False
    i = i + 1
  return True
```

Python (Recursivo)

```python
def rec_primo(n):
        return primo(n,2)

def primo(n,i):
  if(n==1):
    return False
  if(i>n/2):
    return True
  if(n%i==0):
    return False
  return primo(n,i+1)
```

Prolog

```
primo:- read(N),
    primo(N).

primo(N):- primo(N,2).
primo(N,I):- I > N // 2.
primo(N,I):- N mod I =\= 0,
         I1 is I+1,
       primo(N,I1).
```

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

Python (Iterativo)

```python
def proximoPrimo(n):
  # La condicion n<=10e6 es usada como un limite por la complejidad
  # de un test de primalidad (el mejor por ahora AKS, O((log n)^6))
  # y como no es el core de la materia no entramos en detalle :P

  # Hacemos uso de la funcion primo(n)
  while(primo(n)!=True and n<=1000000):
    n = n + 1
  return n
```

Python (Recursivo)

```python
def proximoPrimo(n):
  # De igual manera que el anterior usamos un limite
  if(n>1000000):
```

```
    return -1
  if(primo(n)):
    return n
  return proximoPrimo(n+1)
```

Prolog

```
proximoPrimo(N,R):-
          N =< 1000000,
          primo(N),
          R is N.
proximoPrimo(N,R):- N1 is (N+1),
          proximoPrimo(N1,R).
```

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

Python (Iterativo)

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

Python (Recursivo)

```
def anteriorPrimo(n):
  # De igual manera que el anterior usamos un limite
  if(n<2):
    return -1
  if(primo(n)):
    return n
  return anteriorPrimo(n-1)
```

Prolog

```
anteriorPrimo(N,R):-
          N > 2,
          primo(N),
          R is N.
anteriorPrimo(N,R):- N1 is (N-1),
          anteriorPrimo(N1,R).
```

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

Python (Iterativo)

```
def mostrarPrimos(a,b):
        while(proximoPrimo(a)<=b):
                a = proximoPrimo(a)
                print(a)
                a = a + 1
```

Python (Recursivo)

```
def mostrarPrimos(a,b):
        if(proximoPrimo(a)<=b):
                a = proximoPrimo(a)
                print(a)
                a=a+1
                mostrarPrimos(a,b)
```

Prolog

```
mostrarPrimos(A,B):- A >= B,!.

mostrarPrimos(A,B):- proximoPrimo(A,R),
```

```
            P is R,
            B >= P,
            X is R+1,
            write(R),nl,
            mostrarPrimos(X,B).

```

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

Python (Iterativo)

```
def mostrarFactoriales(n):
  f = 1
  i = 1
  while(i<=n):
    f = f*i
    print(str(i)+"!="+str(f))
    i = i + 1
```

Python (Recursivo)

```
def mostrarFactoriales(n):
  if(n==0 or n==1):
    print(str(n)+"!=1")
    return 1
  f = n*mostrarFactoriales(n-1)
  print(str(n)+"!="+str(f))
  return f
```

Prolog

```
mostrarFactorial:- read(N),mostrarFactorial(N,_).

mostrarFactorial(0,1):-!.
mostrarFactorial(N,R):- N1 is N-1,
      mostrarFactorial(N1,R1),
      R is N*R1,
      write(N),
      write("!="),
      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)

```python
def ite_mostrarFib(n):
  i = 1
  a = 0
  b = 1
  while(i<n):
    print(b)
    f = a + b
    a = b
    b = f
    i = i + 1
```

Python (Recursivo)

```python
def fibo(n):
  if(n==1):
    return 1
  if(n==2):
    return 1
  x = (fibo(n-1)+fibo(n-2))
  return x

def mostrarFib(n):
  for i in range(1,n+1):
    print(fibo(i))
```

Prolog

```
fibo:-
  read(N),
  fib(N,_,_).

fib(1, 1, 0):- write(1),nl.
fib(X, Y1, Y2) :- X > 1,
          X1 is X - 1,
          fib(X1, Y2, Y3),
          Y1 is Y2 + Y3,
          write(Y1),nl.
```