



Taller de Programación



AGENDA



Recursión



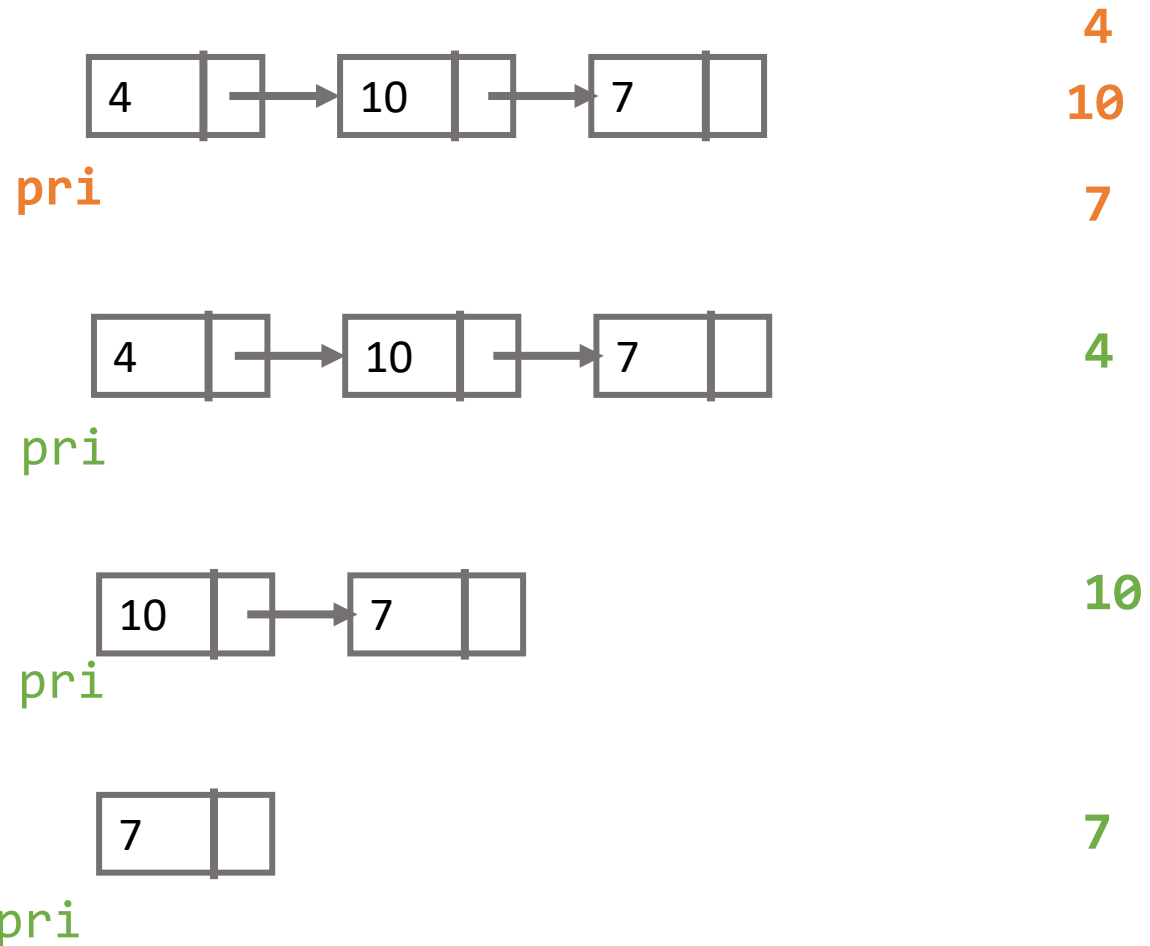
Recursión - MOTIVACION



Suponga que debe realizar un módulo que imprima una los elementos de una lista de enteros.

```
Procedure imprimir (pri:lista);  
Begin  
  while (pri <> nil) do  
    begin  
      write(pri^.dato);  
      pri:= pri^.sig;  
    end;  
End;
```

Se presenta el mismo problema cada vez más chico hasta llegar a una instancia que no se debe resolver nada





Recursión - MOTIVACION



Suponga que debe realizar un módulo que retorne el factorial de un número entero recibido. $n = n * (n-1)$ n veces

```
Procedure factorial (num:integer; var fac:integer);
Var
  i:integer;

Begin
  fac:= 1;
  for i:= num downto 1 do
    begin
      fac:= fac * i;
    end;
  End;
```

Se presenta el mismo problema cada vez más chico hasta llegar a una instancia que se resuelve de manera directa

$$5 = 5 * 4 * 3 * 2 * 1 = 120$$

$$\text{factorial (5)} = 5 * \text{factorial(4)}$$

$$\text{factorial (4)} = 4 * \text{factorial(3)}$$

$$\text{factorial (3)} = 3 * \text{factorial(2)}$$

$$\text{factorial (2)} = 2 * \text{factorial(1)}$$

$$\text{factorial (1)} = 1$$

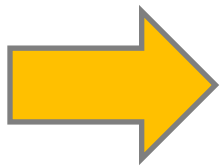


RECURSIÓN - DEFINICION

Existen un conjunto de problemas que pueden resolverse siempre de la misma manera con la característica que el problema debe ir “achicandose” en cada instancia a resolver, hasta que en alguna instancia la solución es “trivial”.



La **recursividad** es una técnica de resolución de problemas que consiste en dividir un problema en instancias más pequeñas del mismo problema (también llamados subproblemas) hasta que obtengamos un subproblema lo suficientemente pequeño que tenga una solución trivial o directa.



La recursividad consiste en resolver un problema por medio de un módulo (procedimientos o funciones) que se llama a sí mismo, evitando el uso de bucles y otros iteradores.

Cuando el problema se va achicando llega a un punto que no puede achicarse más, esa instancia se denomina **caso base**.

Hay problemas en los cuales debe realizarse alguna tarea cuando se alcanza el caso base y otros que no.

Hay problemas que pueden tener más de un caso base.



RECURSIÓN - EJEMPLOS

Suponga que debe realizar un módulo que imprima los elementos de una lista de enteros que recibe como parámetro.

SOLUCIÓN ITERATIVA

```
Procedure imprimir (pri:lista);  
Begin  
  while (pri <> nil) do  
    begin  
      write (pri^.dato);  
      pri:= pri^.sig;  
    end;  
End;
```

Cómo achíco el problema?

Hasta cuando achíco el problema?

Qué hago cuando llego al caso base?

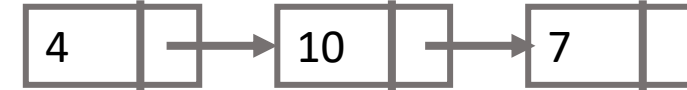
SOLUCIÓN RECURSIVA

```
Procedure imprimir (pri:lista);  
Begin  
  if (pri <> nil) do  
    begin  
      write (pri^.dato);  
      pri:= pri^.sig;  
      imprimir (pri);  
    end;  
End;
```

Cómo funciona?



RECURSIÓN – EJEMPLOS – Cómo funciona?



```
Procedure imprimir (pri:lista);
Begin
  if (pri <> nil) then
    begin
      write (pri^.dato);
      pri:= pri^.sig;
      imprimir (pri);
    end;
End;
```

*Cuál es la diferencia
con la solución
secuencial?*

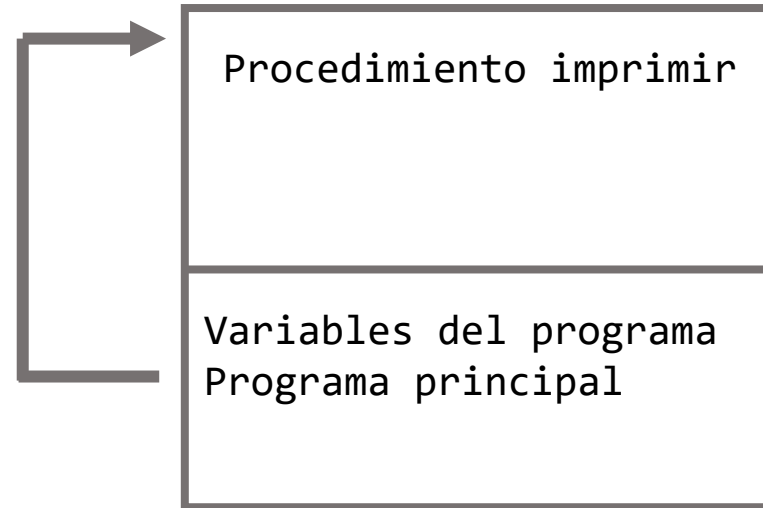
Procedimiento imprimir	pri= 4	4	3
Procedimiento imprimir	pri= 10	10	3
Procedimiento imprimir	pri= 7	7	3
Procedimiento imprimir	pri= nil	En este caso no se hace nada	
Variables del programa Programa principal			



RECURSIÓN – EJEMPLOS – Cómo funcionan?

SOLUCIÓN ITERATIVA

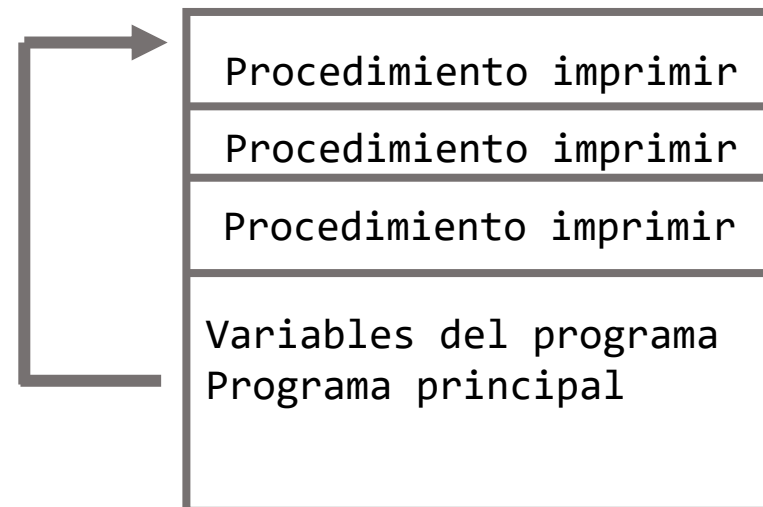
```
Procedure imprimir (pri:lista);  
Begin  
  while (pri <> nil) do  
    begin  
      write (pri^.dato);  
      pri:= pri^.sig;  
    end;  
  End;
```



Cuál cree que es más eficiente en cuanto al uso de la memoria?

SOLUCIÓN RECURSIVA

```
Procedure imprimir (pri:lista);  
Begin  
  IF (pri <> nil) then  
    begin  
      write (pri^.dato);  
      pri:= pri^.sig;  
      imprimir (pri);  
    end;  
  End;
```



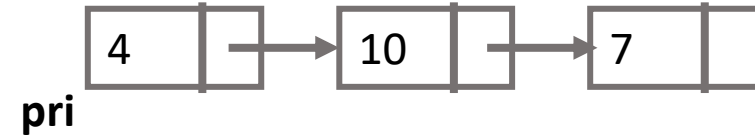
Qué pasa con los parámetros?




RECURSIÓN – EJEMPLOS – Cómo funciona?

SOLUCIÓN RECURSIVA

```
Procedure imprimir (pri:lista);
Begin
  if (pri <> nil) then
    begin
      write (pri^.dato);
      pri:= pri^.sig;
      imprimir (pri);
    end;
End;
```



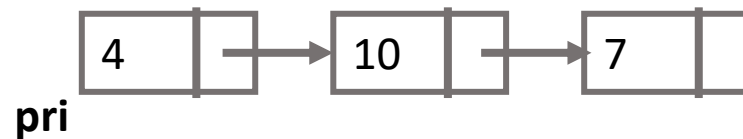
	Procedimiento imprimir	pri= 4	4	3
	Procedimiento imprimir	pri= 10	10	3
	Procedimiento imprimir	pri= 7	7	3
	Procedimiento imprimir	pri= nil	En este caso no se hace nada	
	Variables del programa Programa principal pri=4			




RECURSIÓN – EJEMPLOS – Cómo funciona?

SOLUCIÓN RECURSIVA

```
Procedure imprimir (VAR pri:lista);
Begin
  IF (pri <> nil) then
    begin
      write (pri^.dato);
      pri:= pri^.sig;
      imprimir (pri);
    end;
End;
```





Procedimiento imprimir	pri= 4	4	3
Procedimiento imprimir	pri= 10	10	3
Procedimiento imprimir	pri= 7	7	3
Procedimiento imprimir	pri= nil	En este caso no se hace nada	
Variables del programa Programa principal pri = nil			



RECURSIÓN - EJEMPLOS

Suponga que debe realizar un módulo que calcular la potencia de un número x a la n , que es $= x^n = x * x * x$ (n veces).

SOLUCIÓN ITERATIVA

```
Procedure potencia (x,n:integer;  
                  var pot:integer);
```

```
Var  
  i:integer;  
Begin  
  if (n = 0) then pot:= 1  
  else if (n = 1) then pot:= x  
  else begin  
    pot:= 1;  
    for i:= 1 to n do  
      pot:= pot * x;  
    end;  
  End;
```

Con una
función?

Cómo lo
pienso
recursivo?

SOLUCIÓN ITERATIVA

```
Function potencia (x,n:integer):integer;  
Var
```

```
  i,pot:integer;  
Begin  
  if (n = 0) then pot:= 1  
  else if (n = 1) then pot:= x  
  else begin  
    pot:= 1;  
    for i:= 1 to n do  
      pot:= pot * x;  
    end;  
  potencia:=pot;  
End;
```



RECURSIÓN - EJEMPLOS

Suponga que debe realizar un módulo que calcular la potencia de un número x a la n , que es $= x^n = x * x * x$ (n veces).

SOLUCIÓN RECURSIVA

```
Procedure potencia (x,n:integer;  
                  var pot:integer);
```

```
Var
```

```
  i:integer;
```

```
Begin
```

```
  if (n = 0) then pot:= 1
```

```
  else if (n = 1) then pot:= x
```

```
  else
```

```
    begin
```

```
      potencia (x, (n-1), pot);
```

```
      pot:= pot * n;
```

```
    end;
```

```
End;
```

Con una
función?

Cuántos caso
base hay?

SOLUCIÓN RECURSIVA

```
Function potencia (x,n:integer):integer;
```

```
Begin
```

```
  if (n = 0) then potencia:= 1
```

```
  else if (n = 1) then potencia:= x
```

```
  else
```

```
    potencia:= x * potencia(x, n-1));
```

```
  end;
```

```
End;
```

Cómo
funciona?



RECURSIÓN - Características

SOLUCIÓN RECURSIVA

```
Function potencia (x,n:integer);
```

```
Begin
```

```
  if (n = 0) then potencia:= 1
```

```
  else if (n = 1) then potencia:= x
```

```
  else
```

```
    potencia:= x * potencia(x, n-1));
```

```
  end;
```

```
End;
```

Supongamos $x = 4$ $n=3$



potencia x= 4,n=3

4 * **potencia** (4,2)

164

potencia x= 4,n=2

4 * **potencia** (4,1)

potencia x= 4,n=1

4

Alguna vez entrará
por el caso (n=0)?