RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

CLASE 19

Resolución de problemas utilizando **recursión**

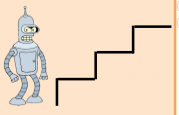
Luciano H. Tamargo
http://cs.uns.edu.ar/~lt
Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca
2016

CONCEPTO: PLANTEO RECURSIVO

La forma de resolver un problema puede plantearse de manera recursiva si se indica:
(a) **un caso base** que **no** se define en términos de si mismo, y
(b) **un caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

Problema propuesto: Hacer un planteo recursivo para subir una escalera la cual puede verse como "un simple escalón, o un escalón seguido de una escalera"

Planteo recursivo: **subir una escalera**
Caso base:
si hay un solo escalón, subo el escalón.
Caso general: si hay más de un escalón,
primero subo un escalón,
y luego **subir una escalera**.



CONCEPTO: PLANTEO RECURSIVO

Un **planteo recursivo** es una solución a un problema donde:
(a) se indica **un caso base** que **no** se define en términos de si mismo, y además,
(b) se indica **un caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

Problema propuesto: escribir un planteo recursivo para resolver una secuencia de ejercicios (un práctico).

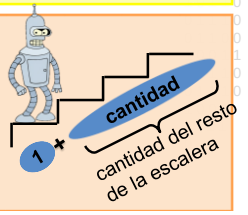
Planteo recursivo: **Resolver una secuencia de ejercicios**
Caso base: si queda un solo ejercicio, intentar resolverlo.
Caso general: si queda más de un ejercicio, intentar resolver el primer ejercicio y luego **Resolver la secuencia de ejercicios restante** (sin considerar al primero).

CONCEPTO: PLANTEO RECURSIVO

Un **planteo recursivo** es una solución a un problema donde:
(a) se indica **un caso base** que **no** se define en términos de si mismo, y además,
(b) se indica **un caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

Problema propuesto: contar la cantidad de escalones.

Planteo: **Cantidad de escalones**
(CB) si hay un solo escalón:
la cantidad de escalones es 1
(CG) si hay más de un escalón,
subo un escalón, y la cantidad de escalones es:
1 + la **cantidad de escalones** del resto de la escalera.



METODOLOGÍA PROPUESTA

1. Identificar **ejemplos significativos** que ayuden a entender el problema y su solución.
2. Realizar un **planteo recursivo** en el cual se distinga el "caso base", y el "caso general" (donde se define en términos de si mismo pero para una instancia más simple/reducida/menor).
3. **Verificar que el planteo sea correcto** (con alguno de los ejemplos significativos).
4. Determinar si se realizará una **función** o un **procedimiento recursivo**, e implementarlo en Pascal.
5. Realizar la **traza** de la primitiva en Pascal.

PROBLEMA PROPUESTO: TODOS PARES

Escriba un planteo recursivo y luego una función (que respete ese planteo) que indique si todos los dígitos de un número entero son pares. Siguiendo la metodología….

Ejemplos:
246, 440, 0, y 8 tienen todos dígitos pares.
21, 12468 y 5 no tienen todos dígitos pares.

Verifiquemos con casos de prueba (los ejemplos).

Planteo: son todos dígitos pares en N
Caso base: si N tiene un único dígito entonces si N es par, son todos pares, de lo contrario no lo son.
Caso general: si N tiene más de un dígito, entonces son todos pares en N si: el último dígito de N es par y además **son todos dígitos pares en N sin su último dígito**.

Realicemos una función en Pascal en el pizarrón que respete el planteo. ¿Será única?

IMPLEMENTACIÓN EN PASCAL

- Dado un planteo recursivo, puede haber varias formas de implementar una primitiva (función o procedimiento) recursiva que respete dicho planteo.
- A continuación se mostrará como implementar el planteo recursivo anterior (**son todos dígitos pares en N**) con tres funciones recursivas (levemente diferentes una de otra) pero todas respetando el planteo dado.

UNA FORMA DE IMPLEMENTAR LA FUNCIÓN RECURSIVA

```
FUNCTION todospares(N:integer):boolean;
{Función recursiva que recibe un entero N y retorna
 true si todos los dígitos de N son pares o falso en caso contrario}
VAR
ultimo_par, anteriores_pares: boolean;
BEGIN
IF (N div 10) = 0 THEN {caso base: N tiene un dígito}
 IF N mod 2 =0 THEN
  todospares := true
 ELSE todospares := false
ELSE  {caso general: N tiene más de un dígito}
 BEGIN
  ultimo_par:=((N mod 10)mod 2)=0; {¿par el último díg.de N?}
  anteriores_pares := todospares(N div 10);
                      {veo si son todos pares N sin su último dígito}
  todospares := ultimo_par and anteriores_pares;
              {retorna true si el último es true y además anteriores es true}
 END; {else}
END; {función todospares}
```


OTRA FORMA DE IMPLEMENTAR LA FUNCIÓN

- La siguiente implementación también respeta el planteo. Realice una traza para ver las diferencias.

```
FUNCTION todospares(N:integer):boolean;
{Función recursiva que recibe un entero N y retorna
 true si todos los dígitos de N son pares o falso en caso contrario}
BEGIN
IF (N div 10) = 0 THEN {caso base: N tiene un dígito}
  todospares := N mod 2 =0
ELSE {caso general: N tiene más de un dígito}
  IF (((N mod 10) mod 2) = 0) THEN
    todospares:= todospares(N div 10)
  ELSE
    todospares:=false;
END; {todospares}
```


OTRA FORMA DE IMPLEMENTAR LA FUNCIÓN

- La siguiente función recursiva también respeta el planteo, pero tiene una implementación un poco más compacta. Realice una traza para comprobarlo.

```
FUNCTION todospares(N:integer):boolean;
{Función recursiva que recibe un entero N y retorna
 true si todos los dígitos de N son pares o falso en caso contrario}
BEGIN
IF (N div 10) = 0 THEN {caso base: N tiene un dígito}
  todospares := N mod 2 =0
ELSE {caso general: N tiene más de un dígito}
  todospares:=(((N mod 10) mod 2) = 0) and
              todospares(N div 10);
END; {todospares}
```


IMPLEMENTACIÓN EN PASCAL

- **Observación**: en clase mostramos que había varias formas diferentes de implementar una función recursiva que respeta el planteo (más de una por alumno presente).
- Por ejemplo, para implementar "N tiene un solo dígito" se puede usar cualquiera de estas expresiones equivalentes:

```
(N div 10) = 0 {N tiene un dígito}
abs(N) < 10 {N tiene un dígito}
(N > -10) and (N < -10) {N tiene un dígito}
```

- "Si N es par, entonces la función retorna true, de lo contrario false" puede hacerse de estas dos formas:

```
IF N mod 2 = 0 THEN
 todospares:=true
ELSE
 todospares:=false
```

```
todospares:=N mod 2=0
```


IMPLEMENTACIÓN EN PASCAL

Tarea propuesta: implementar un procedimiento que resuelva el problema siguiendo lo establecido en el planteo "**son todos dígitos pares en N**". Escriba además un programa de prueba para este procedimiento y realice trazas con ejemplos significativos (esto lo ayudará además a practicar el concepto de parámetro por referencia) .

- El siguiente procedimiento recursivo también respeta el planteo. Realice una traza para comprobarlo.

PROCEDIMIENTO RECURSIVO

```
PROCEDURE todos_dig_par(N:integer; VAR resultado:
boolean);
{Procedimiento recursivo que recibe un entero N y retorna
 true si todos los dígitos de N son pares o falso en caso contrario}
VAR
Result_ultimo, result_resto: boolean;
BEGIN
IF (N div 10) = 0 THEN {caso base: N tiene un dígito}
 resultado:= (N mod 2) = 0
ELSE   {caso general: N tiene más de un dígito}
 BEGIN
  Result_ultimo := (N mod 10) mod 2)=0;
  todos_dig_par(N div 10, result_resto);// llamada
                                        // recursiva
  resultado:=result_ultimo and result_resto;
 END; {else}
END; {función todospares}
```