

## RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

### CLASE 18

#### Resolución de problemas utilizando **recursión**

0 1 1 0 0  
 1 0 0 1 1  
 1 0 1 1 0  
 0 1 1 1 0  
 0 1 1 0 0  
 1 0 0 1 1  
 1 0 1 1 0  
 0 1 1 1 0  
 1 0 0 1 1  
 1 1 1  
 0 0  
 1

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

### SOLUCIONES RECURSIVAS

- Considere un escenario donde debemos programar un robot para subir una escalera, y se dispone de estas primitivas:
  - **hay-un-solo-escalón**: retorna TRUE o FALSE
  - **hay-más-de-un-escalón**: retorna TRUE o FALSE
  - **subir-escalón**: hace al robot subir un escalón.

Observe el dibujo de la izquierda, donde el robot tiene una escalera por delante, si ejecuto **subir-escalón** el robot tendrá una escalera por delante (reducida en un escalón):

### SOLUCIONES RECURSIVAS

Una **escalera** puede verse entonces como: **un único escalón, o un escalón seguido de una escalera.**

**Observación:** en las tres situaciones de arriba, el robot siempre tiene una escalera por delante.

### SOLUCIONES RECURSIVAS

**Algoritmo: subir-una-escalera**  
 Si **hay-un-solo-escalón** entonces:  
 - **subir-escalón**  
 Si **hay-más-de-un-escalón** entonces:  
 - **subir-escalón**  
 - **subir-una-escalera**

### CONCEPTOS: ALGORITMOS RECURSIVOS

- **Recursión** es la forma en la cual se especifica un proceso **basado en su propia definición.**
- Un **algoritmo** es **recursivo** si se define en términos de sí mismo.
- Un algoritmo no debe entrar en una ejecución infinita, por lo tanto, un algoritmo recursivo **será válido**, si:
  - a) **existe un caso base** que **no** se define en términos de sí mismo, y
  - b) existe un caso general donde la **referencia a sí mismo es sobre una instancia más sencilla (o reducida)** que el caso considerado.

### ALGORITMOS RECURSIVOS

- Un algoritmo recursivo **será válido**, si:
  - a) **existe un caso base** que **no** se define en términos de sí mismo, y
  - b) existe un caso general donde la **referencia a sí mismo es sobre una instancia más sencilla (o reducida)** que el caso considerado.

**Algoritmo: subir-una-escalera**  
 Si **hay-un-solo-escalón** entonces:  
 - **subir-escalón**  
 Si **hay-más-de-un-escalón** entonces:  
 - **subir-escalón**  
 - **subir-una-escalera** **OK**

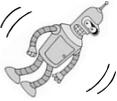
# Resolución de Problemas y Algoritmos

**ALGORITMOS RECURSIVOS**

- Un algoritmo recursivo **será válido**, si:
  - existe un caso base** que **no** se define en términos de sí mismo, y
  - existe un caso general donde la **referencia a sí mismo es sobre una instancia más sencilla (o reducida)** que el caso considerado.

¿Por qué es incorrecto?

**Algoritmo 2 subir-escalera**  
 -subir-escalón  
 -subir-escalera **MAL**



Falla porque no hay indicado un **caso base** (a).  
 ¿Termina de ejecutarse?

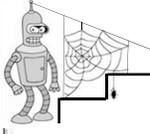
Resolución de Problemas y Algoritmos - 2016 7

**ALGORITMOS RECURSIVOS**

- Un algoritmo recursivo **será válido**, si:
  - existe un caso base** que **no** se define en términos de sí mismo, y
  - existe un caso general donde la **referencia a sí mismo es relativamente más sencilla o reducida** que el caso considerado.

¿Por qué es incorrecto?

**Algoritmo 3 subir-la-escalera**  
 Si **hay-más-de-un-escalón** entonces:  
 - subir-la-escalera  
 - subir-escalón **MAL**



Falla (b): la referencia a sí mismo **NO** es relativamente más sencilla o reducida (es igual).  
 ¿Termina de ejecutarse? ¿sube un escalón?

Resolución de Problemas y Algoritmos - 2016 8

**CONCEPTO: PLANTEO RECURSIVO**

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

- un **caso base** que **no** se define en términos de sí mismo, y
- un **caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

- De esta forma, al utilizar este tipo de planteo, el problema queda dividido en dos sub-problemas:
  - caso base (también llamado caso trivial) y
  - caso general (también llamado caso recursivo).
- Para indicar como resolver un problema de manera recursiva en RPA usaremos un planteo recursivo.

Resolución de Problemas y Algoritmos - 2016 9

**CONCEPTO: PLANTEO RECURSIVO**

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

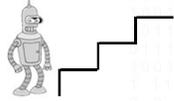
- un **caso base** que **no** se define en términos de sí mismo, y
- un **caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

- Observación:** Una escalera puede verse como "un único 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** que tiene un escalón menos.



Resolución de Problemas y Algoritmos - 2016 10

**CONCEPTO: PLANTEO RECURSIVO**

Un **planteo recursivo** es una solución a un problema donde:

- se indica un **caso base** que **no** se define en términos de sí mismo, y además,
- se indica un **caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

**Problema propuesto:** controlar un conjunto de 1 o más solicitudes de beca para asistir a un evento.

**Planteo recursivo: Controlar un conjunto de solicitudes**

Caso base:  
 si hay una sola solicitud, controlar dicha solicitud

Caso general: si hay más de una solicitud  
 Controlar una solicitud y luego **controlar un conjunto de solicitudes** sin considerar la ya controlada.

Resolución de Problemas y Algoritmos - 2016 11

**CONCEPTO: PLANTEO RECURSIVO**

Un **planteo recursivo** es una solución a un problema donde:

- se indica un **caso base** que **no** se define en términos de sí mismo, y además,
- se indica un **caso general** donde se hace **referencia a sí mismo**, pero con una instancia reducida del problema.

- Generalmente los trabajos de impresión (jobs) son enviados a una cola de impresión (queue) antes de ser impresos.
- Se puede escribir una solución recursiva para el siguiente problema.

**Problema propuesto:** imprimir todos los documentos de una cola de impresión que puede tener 1 o más documentos.

Resolución de Problemas y Algoritmos - 2016 12

**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 sí 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:** imprimir todos los documentos de una cola de impresión que puede tener 1 o más documentos.

**Planteo recursivo: Imprimir trabajos de la cola de impresión Q**  
 Caso base:  
 si hay un único trabajo en Q, enviar a la impresora  
 Caso general: si hay más de un trabajo en Q,  
 sacar el primero de Q y enviarlo a la impresora,  
 y luego **imprimir trabajos de la cola de impresión Q**.

**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 sí 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 (¡para el 29!):** escribir un planteo para comer un plato de ñoquis (que no está vacío).

**Planteo recursivo: Comer un plato de ñoquis**  
 Caso base:  
 si hay un solo ñoqui en el plato,  
 comer un ñoqui.  
 Caso general: si hay más de un ñoqui en el plato,  
 comer un ñoqui y luego **comer un plato de ñoquis**.



**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 sí 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:** (para acompañar los ñoquis) escribir un planteo para tomar un vaso de una bebida de a sorbos (que no está vacío).

**Planteo recursivo: tomar un vaso de bebida**  
 Caso base:  
 Caso general:

**SOLUCIONES RECURSIVAS EN PASCAL**

- Los **procedimientos y funciones** de Pascal nos permitirán **implementar soluciones recursivas**.
- A continuación, para mostrar como hacer funciones y procedimientos recursivos, **vamos a usar ejemplos un poco más simples** que los problemas asociados a programar el comportamiento de robots.
- Como verá en otras materias y en su vida profesional, "recursión" es una **herramienta mucho más general y poderosa** que la "iteración".
- Verá por ejemplo que hay **lenguajes de programación que solo tienen recursión**.

**METODOLOGÍA PROPUESTA**

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

**PROBLEMA PROPUESTO:  $2^N$**

- Escribir una función recursiva en Pascal para computar la función  $2^N$  (N no negativo).
- Para N no negativo, la función  $2^N$  puede definirse recursivamente de la siguiente manera:

$$2^N \begin{cases} 2^0 = 1 & (\text{si } N=0) \\ 2^N = 2 * 2^{N-1} & (\text{si } N>0) \end{cases}$$

Observe que ya está dividido en 2 casos.

**Planteo recursivo:  $2^N$**   
 • Caso base: si  $N=0$  entonces  $2^N$  es 1  
 • Caso general: Si  $N>0$  entonces  $2^N$  es  $2 * 2^{N-1}$

# Resolución de Problemas y Algoritmos

**PROBLEMA PROPUESTO:  $2^N$**

```

FUNCTION dosAlaN(N:integer):integer;
{Otra versión de una función recursiva para 2 a la N (que no usa variables auxiliares)}
BEGIN
IF (N = 0) THEN
  dosAlaN :=1 {caso base}
ELSE
  dosAlaN := 2 * dosAlaN(N-1); {c. general}
END;
    
```

Esta es una forma de implementar en Pascal la función recursiva que respeta el planteo recursivo (no es la única).

**Planteo recursivo:  $2^N$**

- Caso base: si  $N=0$  entonces  $2^N$  es 1
- Caso general: Si  $N>0$  entonces  $2^N$  es  $2 * 2^{N-1}$

Resolución de Problemas y Algoritmos - 2016 19

```

PROGRAM Prueba2;{Prueba otra versión de 2 a la N}
VAR exp, r: integer;

FUNCTION dosAlaN(N:integer):integer;
{Otra versión de una función recursiva para 2 a la N (que no usa variables auxiliares)}
BEGIN
IF (N = 0) THEN
  dosAlaN :=1 {caso base}
ELSE
  dosAlaN := 2 * dosAlaN(N-1); {c. general}
END;

{El programa valida la entrada y llama a la función recursiva}
BEGIN
writeln(' Ingrese un exponente >= 0');
repeat readln(exp) until exp >= 0;
r:= dosAlaN(exp); writeln('2 a la',exp,'es', r);
END.
    
```

Resolución de Problemas y Algoritmos - 2016 20

```

PROGRAM Prueba2;{Prueba otra versión de 2 a la N}
VAR exp, r: integer;

FUNCTION dosAlaN(N:integer):integer;
{Otra versión de una función recursiva para 2 a la N (que no usa variables auxiliares)}
BEGIN
IF (N = 0) THEN
  dosAlaN :=1 {caso base}
ELSE
  dosAlaN := 2 * dosAlaN(N-1); {c.
    
```

Realice trazas para  $exp=0$  y  $exp=3$ .  
¿Cuántas veces se llama a la función dosAlaN con respecto al valor de  $exp$ ?

{El programa valida la entrada y llama a la función recursiva}

```

BEGIN
writeln(' Ingrese un exponente >= 0');
repeat readln(exp) until exp >= 0;
r:= dosAlaN(exp); writeln('2 a la',exp,'es', r);
END.
    
```

Resolución de Problemas y Algoritmos - 2016 21

**PROBLEMA PROPUESTO:  $2^N$**

```

FUNCTION dosAlaN(N:integer):integer;
VAR aux, nuevoN:integer;
BEGIN {función recursiva para 2 a la N}
IF (N = 0) THEN
  dosAlaN :=1 {caso base}
ELSE
  BEGIN {caso general}
    nuevoN:=N-1;
    aux:= dosAlaN(nuevoN);
    dosAlaN:=2 * aux;
  END;
END;
    
```

Esta es una forma de implementar en Pascal la función recursiva que respeta el planteo recursivo (no es la única).

**Planteo recursivo:  $2^N$**

- Caso base: si  $N=0$  entonces  $2^N$  es 1
- Caso general: Si  $N>0$  entonces  $2^N$  es  $2 * 2^{N-1}$

Resolución de Problemas y Algoritmos - 2016 22

```

PROGRAM Prueba;{Prueba otra versión de 2 a la N}
VAR exp, r: integer;

FUNCTION dosAlaN(N:integer):integer;
VAR aux, nuevoN:integer;
BEGIN {función recursiva para 2 a la N}
IF (N = 0) THEN
  dosAlaN :=1 {caso base}
ELSE
  BEGIN {caso general}
    nuevoN:=N-1;
    aux:= dosAlaN(nuevoN);
    dosAlaN:=2 * aux;
  END;
END;

BEGIN
writeln(' Ingrese un exponente >= 0');
repeat readln(exp) until exp >= 0;
r:= dosAlaN(exp); writeln('2 a la',exp,'es', r);
END.
    
```

Resolución de Problemas y Algoritmos - 2016 23

```

PROGRAM Prueba;{Prueba otra versión de 2 a la N}
VAR exp, r: integer;

FUNCTION dosAlaN(N:integer):integer;
VAR aux, nuevoN:integer;
BEGIN {función recursiva para 2 a la N}
IF (N = 0) THEN
  dosAlaN :=1 {caso base}
ELSE
  BEGIN {caso general}
    nuevoN:=N-1;
    aux:= dosAlaN(nuevoN);
    dosAlaN:=2 * aux;
  END;
END;

BEGIN
writeln(' Ingrese un exponente >= 0');
repeat readln(exp) until exp >= 0;
r:= dosAlaN(exp); writeln('2 a la',exp,'es', r);
END.
    
```

Realice trazas para  $exp=0$  y  $exp=3$ .  
¿Cuántas veces se llama a la función dosAlaN con respecto al valor de  $exp$ ?

Resolución de Problemas y Algoritmos - 2016 24

# Resolución de Problemas y Algoritmos

