

Resolución de Problemas y Algoritmos

Clase 20: Resolución de problemas utilizando recursión



Dr. Alejandro J. García
http://cs.uns.edu.ar/~ajg



Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

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 sí mismo, y
 (b) 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).

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 2

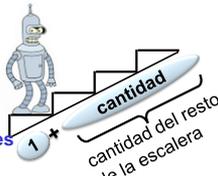
Concepto: planteo recursivo

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 sí mismo, y
 (b) 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.



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 3

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 sí 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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 4

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.

Planteo: son todos dígitos pares en N Verifiquemos con casos de prueba (los ejemplos).

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 el pizarrón que respete el planteo.
 ¿Será única? ¿Se puede hacer en otro lenguaje de programación?

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 5

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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 6

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c) 31/05/2016

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 {caso base: N tiene un dígito}
then 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;
{returna 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 {caso base: N tiene un dígito}
then 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 {caso base: N tiene un dígito}
then 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; {función todospares}
    
```

Implementación en Pascal

Observación: en clase mostramos que había por lo menos 96 formas diferentes de implementar una función recursiva que respete 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_sin_ultimo:boolean;
begin
if (N div 10) = 0 {caso base: N tiene un dígito}
then resultado:= (N mod 2) = 0
else {caso general: N tiene más de un dígito}
begin
todos_dig_par(N div 10, result_sin_ultimo); // llamada recursiva
if ((N mod 10) mod 2) = 0) and result_sin_ultimo
then resultado:=true
else resultado:= false
end; {else}
end;
    
```

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 “Resolución de Problemas y Algoritmos. Notas de Clase”. Alejandro J. García. Universidad Nacional del Sur. (c) 31/05/2016