Resolución de Problemas y Algoritmos

Clase 18:
Resolución de problemas utilizando recursión

Charles Babbage
(1791-1871)

Dr. Diego R. García

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 si mismo, y

(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

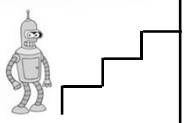
Problema propuesto: Hacer un planteo recursivo para subir una escalera la cual puede verse como "un simple escalón, o un escalón seguido de una escalera"

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

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: escribir un planteo recursivo para intentar 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

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.

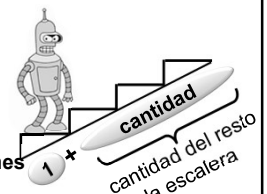
Problema propuesto: contar la cantidad de escalones.

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 (y técnicas) propuestas

1. **Al abordar el problema:**
   a) **Utilizar la técnica de división en sub-problemas (top-down o botton-up).**
   b) **Hacer ejemplos significativos particulares que ayuden a entender el problema (o cada sub-problema): esta técnica también se denomina particularización.**
   c) **Generalizar las ideas de los ejemplos particulares para cubrir todos los casos: esta técnica también se denomina generalización.**
2. **Buscar si existe alguna analogía con otros problemas ya resueltos para aprovechar la experiencia anterior.**
3. **Escribir un algoritmo.**
   Si se requiere una solución recursiva: realizar un planteo recursivo en el cual se distinga el "caso base", y el "caso general".
4. **Verificar con una traza que el algoritmo propuesto (o el planteo recursivo) sea correcto (usar como casos de prueba los ejemplos del punto 1) b).**
5. **Determinar para cada primitiva si se realizará una función o un procedimiento, e implementarlas en el lenguaje de programación (en RPA en Pascal).**
6. **Realizar la traza de cada primitiva implementada (con casos de prueba significativos).**

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 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 {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;
       {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 {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

<u>Observación</u>: en clase mostramos que había muchas formas (levemente) diferentes de implementar una función recursiva que respete el planteo.

- **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

<u>Tarea propuesta</u>: 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;
```


Tarea propuesta: suma dígitos

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

Ejemplos:

Planteo: suma dígitos de N

Caso base:

Caso general:

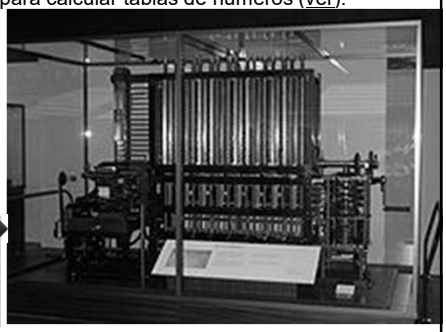
Verifiquemos con casos de prueba (los ejemplos).

Información adicional

Charles Babbage (1791-1871)

En 1812 diseñó y parcialmente implementó una máquina a vapor de diferencias mecánicas para calcular tablas de números (ver).

Parte de su máquina original.

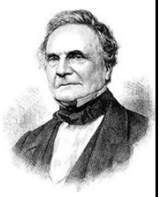


Babbage's difference engine, construida por el Museo de Ciencias Británico en 1991 usando el plano original.

Hay varios videos en YouTube que muestran la máquina funcionando

Charles Babbage (1791-1871)

Matemático y científico de la computación británico.



En 1812 intentó encontrar un método por el cual se pudieran **hacer cálculos automáticamente por una máquina a vapor**.

Eliminando así los errores debidos a la fatiga o aburrimiento que sufrían las personas encargadas de compilar las tablas matemáticas de la época.

Tres diversos factores parecían haberlo influido: su aberración al desorden, su conocimiento de tablas logarítmicas, y los trabajos de máquinas calculadoras realizadas por Blaise Pascal y Gottfried Leibniz.

http://es.wikipedia.org/wiki/Charles_Babbage

La máquina analítica de Babbage

Entre 1833 y 1842, Babbage diseño e intentó construir una máquina que fuese programable para hacer cualquier tipo de cálculo. El diseño se basaba en el telar de Joseph Jacquard, el cual usaba tarjetas perforadas para determinar como una costura debía ser realizada.

Se considera que la máquina analítica de Babbage fue la primera computadora. Un diseño inicial plenamente funcional de ella fue terminado en 1835. Pero la máquina nunca se construyó (hasta ahora…)

El Museo Británico tiene un proyecto para construirla y esperan terminar para 2021 (150 años después de la muerte de Babbage). Sería equivalente a una computadora con un reloj de 7 Hz y con 675 bytes de memoria.

http://www.sciencemuseum.org.uk/

Volver