Resolución de Problemas y Algoritmos

Clase 17:
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

Recursión: 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 es correcto (con alguno de los ejemplos significativos).**
4. **Determinar si se realizará una función o un procedimiento recursivo, e implementarlo en Pascal (aquí ver detalles de implementación, por ejemplo parámetros).**
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 son pares.

Siguiendo la metodología….

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

Planteo: todos 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 pares en N sin su último dígito.

Implementación en Pascal

- **Como se ha dicho antes, para un planteo dado puede haber varias formas de implementar una primitiva que respete el planteo.**
- **A continuación se mostrarán diferentes formas de implementar funciones recursivas que respetan el planteo para "todos pares".**
- **Se mostrarán tres formas correctas pero claramente hay muchas más.**
- **Tarea propuesta: durante la clase (en el pizarrón) el planteo anterior fue implementado con una función, pero para practicar se sugiere también hacer un procedimiento.**

Una forma de implementar la función

```
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 begin {caso general: N tiene más de un dígito}
 {veo si es par el último dígito de N}
 ultimo_par:= (( N mod 10) mod 2) = 0;
 {veo (recursivamente) si son todos pares N sin su último dígito}
 anteriores_pares := todospares(N div 10);
{retorna true si el último es true y además anteriores es true }
 todospares := ultimo_par and anteriores_pares;
 end; {else}
end; {todospares}
```


Otra forma de implementar la función

- **La siguiente implementación también respeta el planteo. 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; {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}
```


Problema propuesto: cantidad de elementos

Escriba un planteo recursivo y luego un procedimiento que respete ese planteo para contar la cantidad de elementos de un archivo.

Ejemplos: 1 2 3 4 (4 elem.) 12 -34 (2 elem)
archivo vacio (0 elem.)

Planteo: cantidad de elementos de un archivo

Caso base: si el archivo está vacío
entonces la cantidad es 0 (cero)

Caso general: si el archivo no está vacío,
entonces la cantidad es 1 + la cantidad de elementos del archivo sin su primero elemento.

Implementación en Pascal

- **Como fue dicho antes no hay una única forma de escribir un procedimiento que respete el planteo.**
- **Hay que tener cuidado donde realiza "assign","reset" y "close" del archivo.**
- **Pregunta teórica:**
  ¿necesita hacer una primitiva recursiva diferente para cada tipo diferente de archivo?
  Escriba su respuesta y consulte sus dudas.
- **Tarea: En clase se mostró un procedimiento. Para practicar realice una función**


```
program prueba1;
type Telemento = integer; Tarchi = file of Telemento;
var A: Tarchi; cantidad: integer;
  Procedure contar (var F: Tarchi; var cant:integer);
  {cuenta los elementos de un archivo}
  var ele: telemento; aux:integer;
  begin
   if EOF(F) then cant:=0 {caso base}
   else begin read(F,ele); {caso recursivo}
              contar(F, aux);
              cant:= aux +1;
        end;
  end;
begin
assign (A, 'el-archivo');
reset(A); contar(A, cantidad); close(A);
writeln('cantidad de elementos: ',cantidad);
end.
```


Observaciones

- En el programa anterior (**prueba1**) **assign**, **reset** y **close** del archivo se realizan en el bloque principal.
- En el procedimiento recursivo "**contar**" la variable local "**aux**" es utilizada para almacenar la cantidad de elementos del "*archivo sin su primer elemento*".
- Realice la traza y verá que en cada llamada recursiva "**aux**" recibe la cantidad calculada por la invocación recursiva y luego el parámetro por referencia "**cant**" retorna "aux" + 1 a quién lo llamó.
- En el programa siguiente (**prueba2**) hay otra versión correcta del procedimiento recursivo que también respeta el planteo pero no usa "**aux**". Realice una traza para ver la diferencia en ejecución.


```
program prueba2;
type Telemento = integer; Tarchi = file of Telemento;
var A: Tarchi; cantidad: integer;
  Procedure contar (var F: Tarchi; var cant:integer);
  var ele: telemento; {cuenta los elementos de un archivo}
  begin
   if EOF(F) then cant:=0 {caso base}
   else begin read(F,ele); {caso recursivo}
              contar(F,cant);
              cant:=cant+1;
        end;
  end;
Begin
assign (A, 'el-archivo');
reset(A); contar(A, cantidad); close(A);
writeln('cantidad de elementos: ',cantidad);
end
```

Observaciones

- **En el programa siguiente (prueba3) hay otra versión correcta del procedimiento recursivo que también respeta el planteo.**
- **En este caso contar abre y cierra el archivo.**
- **Para hacer esto tiene su propio procedimiento interno que hace la tarea recursiva.**
- **Realice una traza para ver la diferencia en ejecución.**


```
program prueba;
type Telemento = integer; Tarchi = file of Telemento;
var A: Tarchi; cantidad: integer;
  Procedure contar (var F: Tarchi; var cant:integer);
    Procedure contar_rec (var F: Tarchi; var cant:integer);
    var ele: telemento; {cuenta los elementos de un archivo}
    begin
     if eof(F) then cant:=0 {caso base}
     else begin read(F,ele); {caso recursivo}
                contar_rec(F,cant);
                cant:=cant+1;
          end;
    end;
  begin {abre el archivo, llama al recursivo y cierra el archivo}
  reset(F); contar_rec(F, cantidad); close(F);
  end;
Begin assign (A, 'el-archivo'); contar(A, cantidad);
writeln('cantidad de elementos: ',cantidad); end.
```