Resolución de Problemas y Algoritmos

Clase 19:
Resolución de problemas utilizando recursión

Ada Byron King

Dr. Diego R. García

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

Problema propuesto: cantidad de elementos

Escriba un planteo recursivo y luego una función que respete ese planteo para contar la cantidad de dígitos de un número.

Ejemplos: 1234 (tiene 4 dígitos)
12 y -34 (tienen 2 dígitos)
2 y 0 (tienen 1 dígito)

Planteo: cantidad de dígitos de N

Caso base: si N tiene un dígito entonces la cantidad es 1

Caso general: si N tiene más de un dígito entonces la cantidad es 1 + cantidad de dígitos de N sin uno de sus dígitos.

Tarea: escribir una función que respeta este planteo.

Resolución de Problemas y Algoritmos Dr. Diego R. García 2

Problema propuesto: mostrar elementos

Escriba un planteo recursivo y un procedimiento que respete ese planteo para mostrar por pantalla uno a uno los elementos de un archivo, e indicar "fin de archivo" al llegar al final.

Planteo: mostrar elementos de un archivo A

Caso base:

si el archivo está vacío

entonces mostrar "fin de archivo"

Caso general:

si el archivo no está vacío,

entonces mostrar el primer elemento y luego mostrar los elementos del archivo A sin su primer elemento.

Tarea: Escriba el procedimiento que hicimos en el pizarrón.

Resolución de Problemas y Algoritmos Dr. Diego R. García 3

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 (tiene 4 elementos)

12, -34 (tiene 2 elementos)

archivo vacío (tiene 0 elementos)

Planteo: cantidad de elementos de un archivo A

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 A sin su primer elemento.

Resolución de Problemas y Algoritmos Dr. Diego R. García 4

```
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 {caso general}
                read(F,ele); {leo el primero elemento}
                contar(F, aux); {llamo con F sin su primer elemento}
                cant:= aux +1;
         end;
  end;
begin
assign (A, 'el-archivo');
reset(A); contar(A, cantidad); close(A);
writeln('cantidad de elementos: ',cantidad);
end.
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 5

```
program prueba1;
type Telemento = integer; Tarchi = file of Telemento;
var A: Tarchi; cantidad: integer;
  Procedure contar (var F: Tarchi; var cant:integer);
  var ele: telemento; aux:integer;
  begin
   reset(A);
   if  EOF(F) then cant:=0 {caso base}
   else  begin {caso general}
                read(F,ele); {leo el primero elemento}
                contar(F, aux); {llamo con F sin su primer elemento}
                cant:= aux +1;
         end;
   close(A);
  end;
begin
assign (A, 'el-archivo');
contar(A, cantidad); writeln('cantidad de elementos: ',cantidad);
end.
```

¿Qué pasaría si hago reset y close dentro del procedimiento recursivo?

Resolución de Problemas y Algoritmos Dr. Diego R. García 6

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.**
- **<u>Pregunta teórica:</u>**
  ¿necesita hacer una primitiva recursiva diferente para cada tipo diferente de archivo?
  Escriba su respuesta y consulte sus dudas.

- **<u>Tarea</u>: (para practicar) Realice una función recursiva que respete el planteo anterior y cuente la cantidad de elementos de un archivo.**

Resolución de Problemas y Algoritmos Dr. Diego R. García 7

Observaciones

- En el programa anterior (**prueba1**) **assign**, **reset** y **close** del archivo se realizan en el bloque principal. Vea por ejemplo lo que pasa en estos dos casos que está mal implementado:

```
Procedure contar (var F: Tarchi; …
{cuenta los elementos de un archivo}
var ele: telemento; aux:integer;
begin
reset(F);
if  EOF(F) then …
```

MAL

```
…
if  EOF(F) then cant:=0
else  begin {caso general}
        reset(F);
        read(F,ele);
        contar(F, aux);
        cant:= aux +1;
end;
```

MAL

- En cualquiera de los dos ejemplos anteriores cada vez que se llama recursivamente se ejecuta nuevamente reset(F), con lo cual se vuelve a comenzar a leer del primer elemento y se produce una ejecución infinita, ya que nunca se reduce el archivo en un elemento (no respeta el planteo).

Resolución de Problemas y Algoritmos Dr. Diego R. García 8

Observaciones

- En el programa anterior (**prueba1**), 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);
  {cuenta los elementos de un archivo}
  var ele: telemento;
   begin
   if  EOF(F) then cant:=0 {caso base}
   else  begin {caso general}
                 read(F,ele); {leo el primero elemento}
                 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.
```

Observe que cambia en la traza si no uso la variable local "aux"

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 prueba3;
type Telemento = integer; Tarchi = file of Telemento;
var A: Tarchi; cantidad_elem: integer;
 Procedure contar (var F: Tarchi; var cantidad: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_elem);
writeln('cantidad de elementos: ',cantidad_elem); end.
```

Problema propuesto

Escriba un planteo recursivo y luego un procedimiento que respete ese planteo para contar la cantidad de apariciones de un elemento E en un archivo F.

Ejemplo: el 3 está 2 veces en F: 4 3 4 3 2

Planteo: Cantidad de apariciones de E en F

Caso base: Si F está vacío,

la cantidad de apariciones de E en F es 0.

Caso general: Si F no está vació entonces

la cantidad de apariciones de E en F, es la cantidad de apariciones de E en F sin su primer elemento,

más uno si el primer elemento de F es E.

Resolución de Problemas y Algoritmos Dr. Diego R. García 13

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

Resolución de Problemas y Algoritmos Dr. Diego R. García 14

Problema propuesto: sumar elementos

Escriba un planteo recursivo y luego una función que respete ese planteo para sumar los elementos de un archivo.

Ejemplos: 1, 2, 3, 4 (suma 10)

12, -34 (suma -22)

archivo vacío (suma 0)

Planteo: suma de elementos de un archivo A

Caso base: si el archivo está vacío

entonces la suma es 0 (cero)

Caso general: si el archivo no está vacío,

entonces la suma es el primer elemento + la suma de elementos del archivo A sin su primer elemento.

Resolución de Problemas y Algoritmos Dr. Diego R. García 15

Información adicional

Ada Augusta Byron King (1815-1852) wiki

Ada es en un sentido metafórico nuestra madre. Diferentes historiadores concuerdan en que fue la primera programadora de la historia.
Ada vivió en una época en la cual la sociedad en su conjunto no veía con agrado a las mujeres que se dedicaban a la ciencia.
A pesar de esto, Ada igual se dedicaba a las matemáticas, y se enteró de los esfuerzos de Babbage y se interesó en su máquina analítica.
Promovió activamente la máquina y escribió varios programas.

Escribió el primer algoritmo para una computadora, el cuál contenía dos bucles. También escribió algoritmos que usaban variables.

Hoy se reconoce a Ada como la primera persona en describir un lenguaje de programación, y la madre de la programación informática.

Resolución de Problemas y Algoritmos Dr. Diego R. García 17

Lady Ada Augusta Byron King, condesa de Lovelace

Lady Ada Lovelace también preveía la capacidad de las computadoras para ir más allá de los simples cálculos de números.

Mientras que otros, incluido el propio Babbage, se centraban únicamente en las capacidades numéricas.

Su trabajo fue olvidado por muchos años, atribuyéndole el papel de transcriptora de las notas de Babbage.

Sin embargo, sus notas sobre programación fueron aprovechadas mucho después.

Volver

Resolución de Problemas y Algoritmos Dr. Diego R. García 18