Resolución de Problemas y Algoritmos

Clase 21:

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

Problema propuesto

Escribir un procedimiento recursivo para calcular, tanto la suma como el producto de los dígitos de un número natural.

Casos de prueba: 1234 suma:10 prod: 24; 11111 suma:5 prod: 1

Planteo: suma y producto de los dígitos de N

caso base: si N tiene un solo dígito,
suma es N y producto es N

caso general: si N tiene más de un dígito, entonces
la suma es: el último dígito de N más la suma de los dígitos de "N sin último dígito"
y el producto es: último dígito de N por el producto de los dígitos de "N sin considerar su último dígito".


```
program prueba; {Prog. de prueba para factorial}
var  num,sumadig,proddig: integer;
 procedure Suma_y_producto(N:integer; suma, prod :integer;)
 begin {calcula suma y producto de los dígitos de N no negativo}
  if N div 10 =0
  then begin suma:= N; prod:=N; end {caso base}
  else begin {caso general}
            suma_y_producto(N div 10, suma, prod);
            suma:=suma+ (N mod 10);
            prod:=prod+ (N mod 10);
       end;
 end;
begin
writeln(' Ingrese un número no negativo');
repeat Readln(Num); until num>=0;
suma_y_producto(Num sumadig,proddig);
writeln(' N:', Num,' suma: ' ,sumadig,'prod: ',proddig );
end.
```


Ejemplo: factorial de un número

- **Las función factorial (!) puede definirse recursivamente de la siguiente manera:**

$$N! \begin{cases} 0! = 1 \\ N! = N * (N-1)! \end{cases}$$

- **Observe que tiene un caso base y otro caso que se define con una instancia más simple de sí mismo.**

Planteo Recursivo: Factorial de N

Caso Base: si N = 0 Factorial de N es 1

Caso General: si N > 0 entonces
Factorial de N es N * Factorial de N-1

Tarea

- **A continuación hay diferentes funciones recursivas que respetan el planteo anterior.**
- **Esto muestra (como ya hemos dicho) que no hay una única forma de resolver un problema.**
- **Haga una traza de cada una de las funciones con el mismo número (por ejemplo 3) para ver las diferencias en la ejecución de cada una de ellas.**
- **Se sugiere luego pasar a la máquina y agregar algunos "write" en su primitiva recursiva para "ver" una traza automática de como se modifican sus variables.**


```
program Factorial_Recursivo; {Prog. de prueba para factorial}
var  num: integer;
     function Factorial (N:integer):integer;
     var aux,nuevoN: integer;
     begin {calcula el factorial de un N no negativo}
      if N=0 then Factorial:=1 {caso base}
      else begin {caso general}
                nuevoN:=N-1;
                aux:= Factorial(nuevoN);
                Factorial:= N * aux;
           end;
     end;
begin
writeln(' Ingrese un número no negativo');
repeat Readln(Num); until num>=0;
writeln(' Factorial de ', Num,' es ' , factorial(Num));
end.
```

Otras dos formas de implementar la función

```
Function Factorial (N:integer):integer;
var aux: integer;
begin {calcula el factorial de un N no negativo}
 IF N=0
 THEN aux:=1 {caso base}
  ELSE aux:= N* Factorial(N-1); {caso general}
 Factorial:= aux;
end;
```

```
Function Factorial (N:integer):integer;

begin {calcula el factorial de un N no negativo}
 IF N=0
 THEN Factorial :=1 {caso base}
  ELSE Factorial := N* Factorial(N-1); {caso general}
end;
```


También puede agregar una validación de datos

Si no confía en la validación de datos antes de la llamada puede agregar una validación interna. Esto hará la función más robusta (nunca entrará en un ciclo infinito), pero no tan elegante.

```
Function Factorial (N:integer):integer;
{calcula el factorial de un N no negativo
 retorna -1 y un mensaje de error si es negativo N}
begin
 IF N=0
 THEN Factorial :=1 {caso base}
 ELSE IF N>0
       THEN Factorial := N* Factorial(N-1) {caso general}
       ELSE begin
              writeln(' ¡error! no existe factorial de negativo');
              Factorial :=-1 {para que la función retorne algo}
            end;
end;
```


Problema propuesto

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

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: 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.


```
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.
```