



## Resolución de Problemas y Algoritmos

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

## MATERIAL ADICIONAL

Dado que para algunos alumnos, escribir un planteo recursivo suele resultar difícil cuando se tiene poca experiencia.

Se incluyen a continuación varios ejemplos de problemas con sus planteos como material de práctica adicional.

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

### Tarea propuesta

A continuación hay varios problemas propuestos con sus correspondientes planteos.

Para practicar se propone:

- 1) Sin mirar el planteo propuesto intente primero escribir su propio planteo para resolver el problema y luego comparar con el propuesto.
- 2) Escriba una función o procedimiento que siga su planteo.
- 3) Realice la traza.
- 4) Pase en la máquina y vea la ejecución
- 5) Consulte sus dudas

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

### Problemas propuestos

<1> Escriba un planteo recursivo y luego una primitiva que respete ese planteo para contar la cantidad de letras ingresadas por teclado, de una secuencia finalizada en un punto.

<2> Escriba un planteo recursivo y luego una primitiva en Pascal (que respete ese planteo) que indique cuando un dígito está presente en un número entero.

<3> Escriba un planteo recursivo y luego una primitiva en Pascal (que respete ese planteo) que indique si un elemento está presente o no en un archivo.

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

### Problema propuesto 1: cantidad de letras

Escriba un planteo recursivo y luego una función (o procedimiento) que respete ese planteo para contar la cantidad de letras ingresadas por teclado, de una secuencia finalizada en un punto.

Ejemplos: 1 2 3. (5 elem.) a. (1 elem) . (0 elem.)

**Planteo:** Cantidad de letras de una secuencia

**Caso base:** si la secuencia es solamente un “.” entonces la cantidad de letras es 0

**Caso general:** si la secuencia tiene más de un “.”, entonces la cantidad es 1 + la cantidad de letras de la secuencia sin su primero elemento.

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

### Problema propuesto 2: dígito presente

Escriba un planteo recursivo y luego una primitiva en Pascal (que respete ese planteo) que indique cuando un dígito está presente en un número entero.

Ejemplos: el 2 está presente en 12345, pero el 7 no.  
 el 0 está presente en 0, pero el 1 no.

**Planteo:** Dig presente en N

**Caso base:** si N tiene un único dígito entonces si N=Dig está presente, sino no lo está

**Caso general:** si N tiene más de un dígito, entonces Dig está presente si Dig es el último dígito de N o si está presente en N sin su último dígito.

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) 2014

### Problema propuesto 3: elemento presente

Escriba un planteo recursivo y luego una primitiva en Pascal (que respete ese planteo) que indique si un elemento está presente o no en un archivo.

Ejemplos: el 2 está presente en 1 2 3 4 5, pero el 7 no.

Ningún elemento está presente en el archivo vacío.

**Planteo:** Elem está presente en archivo

**Caso base:** si el archivo está vacío,

entonces no está el elemento

**Caso general:** si el archivo no está vacío, entonces el elemento está presente si Ele es igual al primer elemento, o está en el archivo sin su primero elemento.

Resolución de Problemas y Algoritmos

Dr. Alejandro J. García

7

### 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

Resolución de Problemas y Algoritmos

Dr. Alejandro J. García

8

### Tarea

- A continuación hay tres 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.

Resolución de Problemas y Algoritmos

Dr. Alejandro J. García

9

```
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.
```

Resolución de Problemas y Algoritmos

Dr. Alejandro J. García

10

### 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;
```

Resolución de Problemas y Algoritmos

Dr. Alejandro J. García

11

### 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;
```

Resolución de Problemas y Algoritmos

Dr. Alejandro J. García

12

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) 2014