Resolución de Problemas y Algoritmos

Clase 16:
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

Reflexión sobre temas vistos

Los siguientes temas, vistos en clases anteriores, están todos relacionados y son muy importantes:

- **Diseño de la solución dividiendo el problema**
- **Funciones y procedimientos en Pascal**
- **Parámetros (por valor y por referencia)**
- **Entorno de referencia de los identificadores**
- **Visibilidad – Identificadores locales, globales, etc.**

¿Alguna pregunta?

Esta importancia va más allá de RPA, en su vida profesional es muy probable que trabaje en un equipo.

Procedimientos y parámetros en Pascal

MultiplicarFracciones tiene 6 parámetros formales: 4 por valor (para recibir datos) y 2 por referencia (para devolver datos)

```
PROCEDURE  MultiplicarFracciones
          ( Num1, Den1, Num2, Den2: INTEGER;
            VAR NumRes,DenRes: INTEGER);
BEGIN  {multiplica dos fracciones}
   NumRes := Num1 * Num2;
   DenRes := Den1 * Den2;
END;
```

Los parámetros por referencia ¿siempre van al final? Respuesta: no.

¿Puede un procedimiento tener sólo parámetros por valor?

```
PROCEDURE BAJAR_LINEAS(cant:INTEGER);
var v:integer; {Deja "cant" líneas en blanco en la pantalla}
BEGIN
FOR v:=1 TO cant DO writeln;
END;
```

¿Conoce uno predefinido que responda a esta pregunta? Realice la misma reflexión sobre todas las demás preguntas

Procedimientos y parámetros en Pascal

¿ Puede un procedimiento tener sólo parámetros por referencia?

```
PROCEDURE PasarAmayuscula(VAR L:char);
begin {Si recibe una minúscula, retorna una mayúscula}
  if (L >= 'a') and (L <= 'z')
  then L := chr(ord(L) - (ord('a') - ord ('A'))
end; {Si no recibe una minúscula, retorna lo mismo recibido}
```

¿ Puede un procedimiento no tener parámetros?

```
PROCEDURE PAUSA;
BEGIN {muestra mensaje y espera por un ENTER del usario}
  writeln(' pulse ENTER para continuar'); readln;
END;
```


Funciones y parámetros en Pascal

¿Puede una función no tener parámetros?

```
FUNCTION  leer_letra:CHAR;
var aux: char; {Esta función sin parámetros lee del buffer }
BEGIN       {hasta que el carácter leído sea una letra y la retorna}
 REPEAT
     read(aux)
 UNTIL (aux>='A') and (aux<='Z') or (aux>='a') and (aux<='z')
 leer_letra:= aux
END;
```

Llamada a la función:

```
ch:=leer_letra;
```


Funciones y parámetros en Pascal

¿ Puede una función tener parámetros por referencia?

```
FUNCTION  leer_letra(VAR error:boolean) :CHAR;
var aux: char; {retorna un carácter leído y false o true }
BEGIN          { false: cuando leyó una letra o true en otro caso }
read(aux);
 IF (aux>='A') and (aux<='Z') or (aux>='a') and (aux<='z')
     then error:=FALSE
     else error:=TRUE;
 leer_letra:= aux
END;
```

Llamada a la función:

```
ch:=leer_letra(hubo_error);
```

Procedimientos vs. Funciones

¿Puede un procedimiento tener un único dato de salida?

```
PROCEDURE EsVocal (letra :char; var ES: boolean);
BEGIN {primitiva para identificar letras vocales}
CASE letra OF {si letra es vocal  retorna true en ES}
 'A','E','I','O','U', 'a','e','i','o','u': Es:=true;
ELSE Es:=false; {o false en caso contrario}
END;
```

¿Qué diferencia hay con tener una función?

```
FUNCTION EsVocal (letra :char): boolean;
BEGIN {primitiva para identificar letras vocales}
CASE letra OF {si letra es vocal la función retorna true}
 'A','E','I','O','U', 'a','e','i','o','u': EsVocal:=true;
ELSE EsVocal:=false; {o false en caso contrario}
END;
```


Observación sobre la traducción al castellano

El nombre en Inglés para los parámetros en Pascal es:

- **formal parameters**, traducido como **"parámetros formales"**
- **actual parameters** que se puede traducir al castellano como
  1) **parámetros reales**
  2) **parámetros efectivos** (yo prefiero esta última para no confundir con un parámetro de tipo real)

IMPORTANTE no hay que confundir "*actual*" en inglés con "actual" en castellano que se escriben igual (se pronuncian diferente) y son dos cosas diferentes.

Es bastante común ver mal traducido *actual parameters* como "parámetros actuales" pero no es correcto (ver a continuación).

https://translate.google.com/?hl=en#en/es/The%20actual%20parameters%20are%20in%20the%20function%20call.

Observación sobre "actual parameters"

EN CASTELLANO:

actual *adj.* Presente. Activo, que obra. Que existe en el tiempo en que se habla.

TRADUCCIÓN A INGLÉS:

actual *ADJ* **1.** (= *de ahora*) [*situación, sistema, gobernante*] → current, present; [*sociedad*] → contemporary, present-day; [*moda*] → current, modern
el actual campeón de Europa → the reigning *o* current *o* present European champion
en el momento actual → at the present moment
la actual literatura francesa → French literature today, present-day French literature

---

EN INGLÉS:

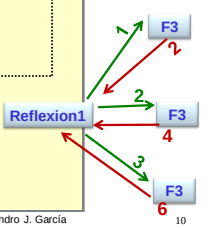
actual *adj* **1.** existing in reality or as a matter of fact
2. real or genuine

TRADUCCIÓN A CASTELLANO:

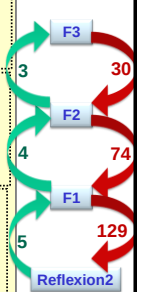
The actual number is much higher than that → El número real es mucho más grande
The film was based on actual events → La película estaba basada en hechos reales
Let's take an actual case/example → Tomemos un caso/ejemplo concreto
There is no actual contract → no hay contrato propiamente dicho

Para reflexionar sobre el pasaje de parámetros, las variables locales y globales, con respecto a la dinámica en la ejecución realice la traza de los siguientes programas:

```
program Reflexion1;
{Para reflexionar sobre la dinámica con una traza}
var i, tope, suma:integer;
   Function F3  (N:integer):integer;
   var local:integer; {F3 modifica local y N}
    begin
    local:= N; N:= N + local; F3:= N;
   end;
begin
tope:= 3;
Suma:=0; {F3 es llamada 3 veces}
for i:=1 to tope do suma:=suma+F3(i);
Writeln('La suma es ', suma);
end.
```




```
program reflexion2; var global:integer;
   Function F3  (N:integer):integer;
   var local:integer;
    begin  writeln('Entro a F3 con ', N);
    local:= 0;  N:= N * 10;  F3:= N + local;
    writeln('Salgo de F3 con ', N + local);  end;
   Function F2  (N:integer):integer;
   var local:integer;
    begin  writeln('Entro a F2 con ', N);
    local:= N+ F3( N - 1);  N:= N * 10;  F2:= N + local;
    writeln('Salgo de F2 con ', N + local);  end;
   Function F1  (N:integer):integer;
   var local:integer;
    begin  writeln('Entro a F1 con ', N);
    local:=N+F2( N - 1);  N:= N * 10;  F1:= N + local;
    writeln('Salgo de F1 con ', N + local);  end;
begin global:= 3 + F1(5) * 10; writeln(global); end.
```



(Repaso) Algoritmos recursivos

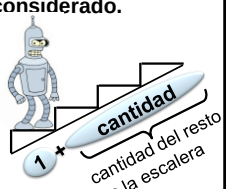
- **Un planteo recursiva será válido, si:**
  (a) existe un caso base que no se define en términos de si mismo, y en el caso general,
  (b) la referencia a sí mismo es relativamente más sencilla o reducida que el caso considerado.

Ejemplo: "Contar los escalones de una ecalera"

Planteo: Cantidad de escalones

(CB) si no hay escalones:
la cantidad es 0 (cero)

(CG) si hay escalones:
sube un escalón y
la cantidad es 1 + la cantidad del resto de la escalera



Problema propuesto

Escriba un programa que permita ingresar por teclado una secuencia de caracteres terminada en un punto (por ejemplo: "hola que tal.") y que la muestre por pantalla en orden inverso ("lat euq aloh"). Ejemplo:

> **Ingrese una cadena terminada en punto: un animal.**
> **Invertida queda así: lamina nu**

Planteo Recursivo para MostrarInvertida una secuencia S

caso base: si la secuencia S es solamente un "." mostrar el cartel "Invertida queda así:"

caso general: MostrarInvertida la secuencia S sin su primer elemento, y luego mostrar el primer elemento de S.

procedimiento recursivo: invertir buffer

```
procedure MostrarInvertida; {observe que no hay parámetros}
var caract: char;
begin
  read(caract); {leo el primer elemento de la secuencia}
   if caract = '.'
    then write(' Invertida queda así: ') {caso base}
   else begin {caso general}
            MostrarInvertida;  {llamada recursiva}
            write(caract); {imprime el primer elemento}
          end; {fin del caso general}
end;
```

Planteo Recursivo para MostrarInvertida una secuencia S

caso base: si la secuencia S es solamente un "." mostrar el cartel "Invertida queda así:"

caso general: MostrarInvertida la secuencia S sin su primer elemento, y luego mostrar el primer elemento de S.


```
program Invertir_Recursivo; {Programa de prueba}

procedure MostrarInvertida; {observe que no hay parámetros}
var caract: char;
begin
  read(caract); {leo el primer elemento de la secuencia}
   if caract = '.'
    then write(' Invertida queda así: ') {caso base}
   else begin {caso general}
            MostrarInvertida;  {llamada recursiva}
            write(caract); {imprime el primer elemento}
          end; {fin del caso general}
end;

begin
writeln(' Ingrese una cadena terminada en punto: ');
MostrarInvertida;
end.
```


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.**
5. **Realizar la traza de la primitiva en Pascal.**