Resolución de Problemas y Algoritmos

Clase 13:
Primitivas como procedimientos.
Parámetros por valor y por referencia.

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

Conceptos: retornar el valor de una función

- En toda función, es necesario que al menos una vez se ejecute una **asignación** que en su parte izquierda tenga el **nombre** de la función.
- Es la forma que tiene la función de retornar un valor.
- Si esto no ocurre se considera **un error de programación** (a continuación se incluyen algunos ejemplos).

```
FUNCTION Cubo (N:integer):integer;
BEGIN
Cubo:= N*N*N;
END;
```
CORRECTO

```
FUNCTION Cubo (N:integer):integer;
VAR aux: integer;
BEGIN
aux:= N*N*N;
END;
```
MAL

INCORRECTO: falta la asignación de valor a la función.

Conceptos: retornar el valor de una función

- En toda función, es necesario que al menos una vez se ejecute una **asignación** que en su parte izquierda tenga el **nombre** de la función.
- Si esto no ocurre se considera **un error de programación** (a continuación se incluyen algunos ejemplos).

```
FUNCTION EsMayuscula (letra :char): boolean;
BEGIN
    IF ('A' <= letra) and (letra <= 'Z' )
    THEN EsMayuscula:=true
    ELSE EsMayuscula:=false
END;
```
CORRECTO

```
FUNCTION EsMayuscula (letra :char): boolean;
BEGIN
    IF ('A' <= letra) and (letra <= 'Z' )
    THEN EsMayuscula:=true
END;
```
MAL

Error de programación: cuando no es mayúscula no se ejecuta la asignación de valor a la función.

Conceptos: retornar el valor de una función

```
FUNCTION EsMayuscula (letra :char): boolean;
BEGIN
    EsMayuscula:=false;
    IF ('A' <= letra) and (letra <= 'Z' )
    THEN EsMayuscula:=true
END;
```
CORRECTO

```
FUNCTION dia_mes (mes,anio :integer): integer;
BEGIN {retorna la cantidad de días de un mes}
    dia_mes:=0;
    CASE MES OF
    11,4,6,9: dia_mes :=30;
    2: IF (anio mod 4=0) and (anio mod 100<>0) or (anio mod 400=0)
       THEN dia_mes := 29 ELSE dia_mes := 28;
    1,3,5,7,8,10,12: dia_mes :=31;
    END;
END;
```
CORRECTO

Puede haber muchas asignaciones que dan valor a la función, y pueden ejecutarse más de una.
Pero para que exista un error de programación no debe ejecutarse al menos una vez.

Importante: el nombre de una función NO ES UNA VARIABLE

Conceptos: retornar el valor de una función

Importante: el nombre de una función **no es una variable**. Algunas veces se crea una confusión porque se le "asigna" un valor. Pero no puede usarse como una variable.

```
function esMayuscula (c :char): boolean;
begin
 esMayuscula:= ('A' <= c) and (c <= 'Z' )
end;
```

```
program …
…
resultado:= esMayuscula(ch);
IF resultado = true
then ….
```

identificador := expresión

El nombre de una función a la izquierda del := se usa para darle valor a la función.

El nombre de una función usado en la expresión a la derecha del := es usado para llamar a la función

Conceptos: retornar el valor de una función

```
FUNCTION Cubo (N:integer):integer;
VAR aux,P: integer;
BEGIN
  Cubo := 1;
  FOR aux:= 1 TO 3
     DO Cubo := Cubo * N;
END;
```
MAL

INCORRECTO: "Cubo" no es una variable, es el nombre de la función. Si usa el nombre de la función en una expresión (como aquí) entonces ¡está llamando a la función!

identificador := expresión

El nombre de una función a la izquierda del := se usa para darle valor a la función.

El nombre de una función usado en la expresión a la derecha del := es usado para llamar a la función

Conceptos: retornar el valor de una función

```
FUNCTION Cubo (N:integer):integer;
VAR aux,P: integer;
BEGIN
  P:= 1;
  FOR aux:= 1 TO 3
     DO P := P * N;
  Cubo:= P;
END;
```

OK

identificador := expresión

El nombre de una función a la izquierda del := se usa para darle valor a la función.

El nombre de una función usado en la expresión a la derecha del := es usado para llamar a la función

Problema planteado

Problema: Escribir una función que retorne el dígito más significativo (dms) de un número entero. Realice además un programa de prueba que use esa función.

Recuerde la metodología propuesta:
ejemplos, solución, algoritmo y finalmente Pascal.

Ejemplos: dms(326)=3; dms(3)=3; dms(-14)=1; dms(0)=0

Solución:

Tomar el valor absoluto N del número ingresado.
Cuando N tiene un dígito, el dms(N) es N.
Si N tiene más de 1 dígito, se cumple la propiedad que dms(N)=dms(N div 10). Ej: dms(326)=dms(32)=dms(3).

Por lo tanto divido N sucesivamente por 10 hasta llegar a tener un numero de un dígito.

Conceptos: parámetros por valor

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR num, D :Integer;

FUNCTION digito_mas_significativo(N:integer): digito;
BEGIN
   if N < 0 then N:=-1*N;
   while (N >= 10) do N:=N div 10;
   digito_mas_significativo:= N;
END;

BEGIN
write('Ingrese un número:');
readln(num);
D:=digito_mas_significativo(num);
writeln('el D.M.S. de', num, 'es', D);
END.
```

Parámetros por valor: reciben una copia de los valores de los efectivos

Como "N" es un parámetro por valor, aunque en la función le asigne nuevos los valores, estos cambios no afectan a la variable "num" del parámetro efectivo.

Copiar la traza del pizarrón ☺

Conceptos: Parámetros por valor

PARÁMETROS

- **Formales**
  - por valor.
- **Efectivos**
  - si corresponde a un **parámetro formal por valor**, puede ser una de estas tres opciones:
    - un **valor**
    - una **expresión**
    - una **variable**

Ejemplos:

Parámetro formal por valor

```
FUNCTION digito_mas_significativo(N:integer): digito;
```

```
{…llamadas a la función…}
d:=digito_mas_significativo(123);
d:=digito_mas_significativo(num div 2);
d:=digito_mas_significativo(num);
```

Parámetros efectivos: valores, expresiones o variables

Conceptos: parámetros por valor

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR num, D :Integer;

FUNCTION digito_mas_significativo(N:integer): digito;
BEGIN
   if N < 0 then N:=-1*N;
   while (N >= 10) do N:=N div 10;
   digito_mas_significativo:= N;
END;

BEGIN
D:=digito_mas_significativo(236);
writeln('el D.M.S. de 236 es', D);
Num:=123;
D:=digito_mas_significativo(Num*7+Num);
writeln('el D.M.S. de', Num*7+num,' es', D);
END.
```

Parámetros por valor: reciben una copia de los valores de los efectivos

El parámetro efectivo puede ser tanto una variable, un valor, o una expresion (siempre que sea de un tipo asignación compatible con el del parámetro formal).

Copiar la traza del pizarrón ☺

Conceptos: Procedimientos (procedure)

Nombre del procedimiento

Parámetros formales

```
PROCEDURE Asteriscos( cant : INTEGER ) ;
{ Imprime "cant" asteriscos consecutivos }
VAR i :INTEGER ;
BEGIN
        FOR i := 1 TO cant  DO  write('*') ;
END ;
```

Variables Locales

- **En Pascal, además de las funciones, se pueden construir primitivas como "procedimientos" (PROCEDURE).**
- **No tienen un tipo asociado.**
- **Tampoco retornan obligatoriamente un valor.**
- **Al igual que las funciones pueden tener parámetros y también variables locales.**
- **Su invocación se realiza como una sentencia.**

```
PROGRAM ejemplos;
VAR tope,i: integer;
PROCEDURE Asteriscos( cant : INTEGER ) ;
{ Imprime "cant" asteriscos consecutivos }
VAR i :INTEGER ;
BEGIN
        FOR i := 1 TO cant DO write('*') ;
END ;
PROCEDURE Pausa;
BEGIN {Muestra mensaje y espera ENTER}
Asteriscos(40); writeln;
Writeln ('Presione ENTER para continuar'); Readln;
END ;
BEGIN
Pausa;
writeln('ingrese tope'); readln(tope);
FOR i:= 1 to tope DO begin Asteriscos(i); writeln; end;
END.
```

Variables globales

Sugerencia: copie el programa y ejecute en la máquina para ver la salida en pantalla.

Obs: no tiene parámetros

Llamadas a procedimiento

Parámetro efectivo

Conceptos: diferencias entre…

Funciones

- Se invocan desde una expresión
- Al regresar de la invocación se sigue ejecutando la sentencia de la llamada.
- Tiene un tipo asociado
- Aunque no tenga parámetros devuelve un valor que se usa en la expresión que la llama.

Procedimientos

- Se invocan como una sentencia.
- Al regresar de la invocación se ejecuta la sentencia siguiente a la llamada.

Cabeza (o encabezado) y Cuerpo

```
PROCEDURE Asteriscos( cant : INTEGER ) ;
{ Imprime "cant" asteriscos consecutivos }
VAR i :INTEGER ;
BEGIN
        FOR i := 1 TO cant DO write('*') ;
END ;
```

cabeza

cuerpo

```
FUNCTION Potencia(Base,Exp:integer):integer;
{calcula Base elevado a la Exp}
VAR aux,P: integer;
BEGIN
  P := 1;
  FOR aux:= 1 TO Exp DO P := P * Base;
  Potencia:= P;
END;
```

cabeza

cuerpo

Ejemplo

- Problema: Escriba una primitiva para multiplicar dos fracciones.

$$\frac{N1}{D1} \times \frac{N2}{D2} = \frac{N1 \times N2}{D1 \times D2} \qquad \frac{1}{3} \times \frac{5}{6} = \frac{5}{18}$$

- Se propone representar una fracción con su numerador y denominador en forma separada; y construir una primitiva "multiplicar fracciones" que retorne el numerador y el denominador del resultado.
- La primitiva tendrá 4 datos de entrada: los 2 numeradores y los 2 denominadores.
- y además 2 datos de salida: el numerador (N) y el denominador (D) del resultado .
- El cálculo de N y D será el sigientes:

```
N := N1 * N2;
D := D1 * D2;
```


Conceptos: Parámetros por referencia

Los parámetros formales pueden ser:

- **por valor: sólo permiten recibir valores y se los utiliza para entrada de datos.**
- **por referencia: cuando se antepone la palabra VAR. En este caso, se crea una referencia con el parámetro efectivo, y por lo tanto, permite salida de datos.**

4 parámetros formales por valor

2 parámetros formales por referencia

```
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
BEGIN
   N := N1 * N2;
   D := D1 * D2;
END;
```


Parámetros por valor

Parámetros por referencia

```
PROGRAM Prueba;
VAR num1, den1, num2, den2, Nres, Dres :Integer;
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
   BEGIN
      N := N1 * N2;
      D := D1 * D2;
   END;
BEGIN
 write('Ingrese 2 fracciones:');
 readln(num1,den1,num2,den2);
 MultFrac (num1,den1,num2,den2, Nres, Dres );
 writeln('Fracción resultado: ',Nres,'/',Dres);
 END.
```

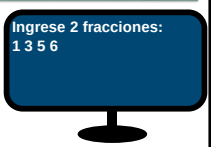
referencia

referencia

Sugerencia: pase a la máquina y ejecute.

Traza con parámetros por referencia

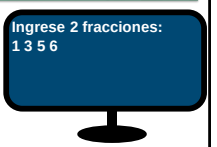
Prueba	
Num1	1
Den1	3
Num2	5
Den2	6
Nres	
Dres	



(1) Cuando comienza la ejecución de prueba, se crean 6 variables. Luego de la ejecución de **readln(num1,den1,num2,den2)**; se asignan valores a las cuatro primeras variables.

Traza con parámetros por referencia

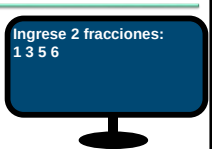
Prueba		MultFrac	
Num1	1	N1	1
Den1	3	D1	3
Num2	5	N2	5
Den2	6	D2	6
Nres		N	
Dres		D	



(2) Al llamar al procedimiento **MultFrac** se crean los parámetros por valor (N1,D1,N2,D2) y los parámetros por referencia (N y D). Luego se copian los valores para los parámetros por valor (indicado en la figura con la flecha punteada de simple punta). Además, se crea una referencia (indicada en la figura con una flecha de doble punta) entre el parámetro N y la variable Nres del programa, y entre el parámetro D y la variable Dres.

Traza con parámetros por referencia

Prueba		MultFrac	
Num1	1	N1	1
Den1	3	D1	3
Num2	5	N2	5
Den2	6	D2	6
Nres	5	N	
Dres	18	D	

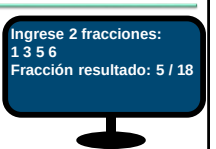


Todo cambio que haga sobre un parámetro formal por referencia, afectará directamente al parámetro efectivo vinculado.

(3) Al ejecutarse las asignaciones del cuerpo del procedimiento, se modifican los parámetros N y D, y al ser estos por referencia modifican directamente a las variables Nres, y Dres que estaban en los parámetros efectivos de la llamada al procedimiento.

Traza con parámetros por referencia

Prueba	
Num1	1
Den1	3
Num2	5
Den2	6
Nres	5
Dres	18



Todo cambio que haga sobre un parámetro formal por referencia, afectará directamente al parámetro efectivo vinculado.

(4) Al finalizar la ejecución del procedimiento, la memoria usada por el procedimiento se liberará. Pero dado que los valores asignados a los parámetros por referencia modificaron las variables Nres y Dres, entonces el resultado no se pierde y es mostrado en pantalla

Conceptos: Parámetros por valor y por referencia

PARÁMETROS

- **Formales**
  - por valor.
  - por referencia.
- **Efectivos**
  - si corresponde a un **parámetro formal por valor,** puede ser una de estas tres opciones:
    - un **valor**
    - una **expresión**
    - una **variable**
  - si corresponde a un **parámetro formal por referencia**, debe ser siempre: • una **variable**

Ejemplos:

```
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
…
```

parámetros formales

```
{…llamadas…}
MultiFrac (1,2,3,4, N,D);
MultiFrac (N,D, 2+2, trunc(2.3)+1, N1, D1);
```

parám. efectivos

Conceptos: compatibilidad de tipos entre parámetros

- En los parámetros por valor, se copia el valor del parámetro efectivo y se asigna este valor al parámetro formal. Cualquier modificación que realice sobre el formal no afectará nunca al valor que tiene el efectivo.
- El valor de un parámetro efectivo pasado por valor debe ser de asignación-compatible al tipo del parámetro formal.

- En los parámetros por referencia se crea un referencia entre el formal y el efectivo. Todo cambio en el formal afecta y cambia al efectivo.
- Si un procedimiento o función tiene un parámetro formal pasado por referencia, entonces el tipo del parámetro formal debe ser idéntico al tipo del parámetro efectivo.