Resolución de Problemas y Algoritmos

Clase 11:
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

Repaso :

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.

Ejemplo de una función

```
PROGRAM prueba; {prueba la función potencia}
VAR B, E, Pot :Integer;

FUNCTION Potencia(Base,Exponente:integer) : integer;
{utilidad: calcula "base" a la "exponente"}
VAR aux,resultado: integer;
BEGIN
  resultado := 1;
  FOR aux:= 1 TO Exponente DO
      resultado := resultado * Base;
  Potencia:= resultado;
END;

BEGIN
write('Ingrese base y exponente:'); readln(B,E);
Pot:=Potencia(B,E);
writeln(B,'a la ',E,'= ',pot);
END.
```

Variables globales
Tipo de la función
Parámetros
Comentario
Variables Locales
Asignación del resultado
Llamada a la función

Conceptos: parámetros formales y efectivos

```
PROGRAM Prueba_potencia;
VAR B,E, Pot :Integer;

FUNCTION Potencia (Base, Exponente:integer) : integer;
{utilidad: calcula "base" a la "exponente"}
VAR aux,resultado: integer;
BEGIN
  resultado := 1;
  FOR aux:= 1 TO Exponente DO resultado := resultado * Base;
  Potencia:= resultado;
END;

BEGIN
   write('Ingrese base y exponente:');
   readln(B,E);
   Pot:=Potencia(B,E);
   writeln(B,'a la',E,'=',pot);
   writeln('2 a la ', E+1, '=', potencia(2,E+1));
END.
```

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

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(32)=3; dms(-14)=1; dms(0)=0

Solución: Tomar el valor absoluto N del número ingresado. Para N con 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 N de un dígito. Cuando N tiene un dígito, el dms(N) es N.

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

```
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: parámetros por valor

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR N, 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(N);
  D:=digito_mas_significativo(N);
  writeln('el D.M.S. de', N, 'es', D);
END.
```

El nombre del parámetro puede ser igual a uno de una variable global.

Importante: Aunque tengan el mismo nombre, los cambios del parámetro N no afectarán a la variable global N.

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

Conceptos: Procedimientos (procedure)

- 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 y no desde una expresión.

Ejemplo:

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



```
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

Parámetros formales

Variables Locales

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

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 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.**
- **Podemos 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.**

```
NumRes := Num1 * Num2;
DenRes := Den1 * Den2;
```

- **Tendrá 4 datos de entrada: los 2 numeradores y los 2 denominadores.**
- **y además 2 datos de salida: el numerador y el denominador resultado.**

Conceptos: Parámetros por referencia

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

4 parámetros por valor

2 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 anteponen la palabra VAR. En este caso, se crea una referencia con el parámetro efectivo, y por lo tanto, permite salida de datos.**

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);
     MultiFrac (num1,den1,num2,den2, Nres, Dres );
 writeln('Fraccion resultado: ',Nres,'/',Dres);
END.
```

referencia referencia

Sugerencia: pase a la máquina y ejecute.

Conceptos: Parámetros en Funciones y Procedimientos

PARÁMETROS

- **Formales**
  - **Por valor** : <nombre/s>:<tipo>
  - **Por referencia:** VAR <nombre/s>:<tipo>
- **Efectivos**
  - si corresponde a un **parámetro formal por valor**, puede ser...
    - un **valor**
    - una **expresión**
    - una **variable**
  - si corresponde a un **parám. formal por referencia**, debe ser únicamente ...
    - una **variable**

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

Ejemplos:
p. formales
p. efectivos

```
...
MultiFrac (1,2,3,4, N,D);
MultiFrac (N,D, 2+2, trunc(2.3)+1, N1, D1);
```


Conceptos: compatibilidad entre parámetros por referencia

- **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.**

Por ejemplo, si hemos declarado:

```
PROCEDURE Calcula( VAR valor: real);
```

y se realiza la invocación:

```
Calcula(numero);
```

entonces numero debe ser de tipo idéntico a real.

Conceptos: compatibilidad entre parámetros por valor

- **En los parámetros por valor se crea una copia del valor del efectivo y se le asigna al 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.**

Por ejemplo, si hemos declarado:

```
PROCEDURE Calcula(valor:real);
```

y se realiza la invocación:

```
Calcula(numero);
```

entonces numero debe ser asignación compatible con real.