Resolución de Problemas y Algoritmos

Clase 13
Procedimientos y Funciones en Pascal
Resolución de problemas por división y composición: construcción de primitivas.

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: parámetros formales y efectivos

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

FUNCTION Potencia (Base, Exponente:integer) : integer;
VAR aux,P: integer;
BEGIN
  P := 1;
  FOR aux:=1 TO Exponente DO P := P * Base;
  Potencia:= P;
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

Reciben una copia de los valores de los parám. efectivos

Parámetros efectivos

Conceptos: parámetros por valor

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

FUNCTION Potencia (Base, Exponente :integer) : integer;
VAR aux,P: integer;
BEGIN   P := 1;
  while (exponente > 0) DO
    begin P := P * Base; exponente:=exponente-1; end;
  Potencia:= P;
END;

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

END.
```

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

Como exponente es un parámetro por valor, aunque cambie los valores, estos cambios no afectan a la variable E

Conceptos: parámetros por valor

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

FUNCTION Potencia (Base, Exponente:integer) : integer;
VAR aux,P: integer;
BEGIN   P := 1;
  while (exponente > 0) DO
    begin P := P * Base; exponente:=exponente-1; end;
  Potencia:= P;
END;

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

Obs: pueden tener el mismo nombre.

Como exponente es un parámetro por valor, aunque cambie su valor, estos cambios no afectan a la variable global exponente (aunque se llamen igual).

Conceptos: Procedimientos (procedure)

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

Ejemplo:

```
PROCEDURE Pausa; {Muestra un mensaje y espera ENTER}
VAR i :INTEGER ;
BEGIN
writeln; FOR i:=1 TO 25 DO write(' ');
writeln ('Presione ENTER para continuar'); Readln;
END ;
```



```
PROGRAM ejemplos;
VAR tope,i: integer;
PROCEDURE Asteriscos( N : INTEGER) ;
{ Imprime N asteriscos consecutivos }
VAR i :INTEGER ;
BEGIN
        FOR i := 1 TO N DO write('*') ;
END ;
PROCEDURE Pausa; {Muestra mensaje y espera ENTER}
VAR i :INTEGER ;
BEGIN
writeln; FOR i:=1 TO 25 DO write(' ');
Writeln ('Presione ENTER para continuar'); Readln;
END ;
BEGIN
Asteriscos (40);
Pausa;
writeln('ingrese tope'); readln(tope);
FOR i:= 1 to tope DO Asteriscos(i);
END.
```

Variables globales

Parámetros formales

Variables Locales

Parámetros efectivos

Llamadas a procedimiento

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.

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 real 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.

Ejemplo con parámetros por valor

```
PROGRAM PorValor; {ejemplo con parámetros por valor}
VAR M,N: integer;
    PROCEDURE  EjemploxValor  ( A, B: integer);
    VAR Aux: integer;
     BEGIN
      write( A, B);
      Aux := A ;    A := aux + 1;    B := Aux;
      write( A, B, Aux);
     END;
BEGIN
 M:=0;
 N:=9;
 Writeln('Inicialmente M es ',M,' y N es ',N,'.');
 EjemploxValor(M,N);
 Writeln('Al terminar M es ',M,' y N es ',N,'.')
END.
```


Ejemplo con parámetros por referencia

```
PROGRAM PorValoryReferencia;
VAR M,N: integer;
 PROCEDURE EjemploxValor_y_Ref (A:integer; var B:integer);
 VAR Aux: integer;
     BEGIN
      write( A, B);
      Aux := A ;    A := aux + 1;    B := Aux;
      write( A, B, Aux);
     END;
BEGIN
 M:=0;
 N:=9;
 Writeln('Inicialmente M es ',M,' y N es ',N,'.');
 EjemploxValor_y_Ref(M,N);
 Writeln('Al terminar M es ',M,' y N es ',N,'.')
END.
```

Realice la traza de este programa donde ahora, hay un parámetro por valor y otro por referencia.

Ejemplo con parámetros por referencia

```
PROGRAM PorReferencia;
VAR M,N: integer;
     PROCEDURE EjemploxRef  (var A,B:integer);
     VAR Aux: integer;
     BEGIN
      Aux := A ;     writeln( 1, A, B);
      A := aux + B; writeln( 2, A, B);
      B := Aux + A; writeln( 3, A, B);
    END;
BEGIN
 M:=0;
 N:=9;
 Writeln('Inicialmente M es ',M,' y N es ',N,'.');
 EjemploxRef(M,N);
 Writeln('Al terminar M es ',M,' y N es ',N,'.')
END.
```

Realice la traza de este programa donde ahora, hay 2 parámetros por referencia

Ejemplo con parámetros por referencia

```
PROGRAM PorReferenciaDos;
VAR M,N: integer;
     PROCEDURE EjemploxRef  (var A,B:integer);
     VAR Aux: integer;
     BEGIN
      Aux := A ;     writeln( 1, A, B);
      A := aux + B; writeln( 2, A, B);
      B := Aux + A; writeln( 3, A, B);
    END;
BEGIN
 M:=0;
 N:=9;
 Writeln('Inicialmente M es ',M,' y N es ',N,'.');
 EjemploxRef(M,M);
 Writeln('Al terminar M es ',M,' y N es ',N,'.')
END.
```

Realice la traza de este programa donde ahora, hay 2 parámetros por referencia con el mismo parámetro efectivo M.

Ejemplo de traza con parámetros por referencia

```
PROGRAM IntercambiarVariables;
VAR M,N, aux: integer;
    PROCEDURE Intercambiar (VAR A, B: integer);
     VAR Aux: integer;
     BEGIN
       Aux := A;
       A := B;
       B := Aux
     END;
BEGIN
 M:=5;  N:=9;  aux: 100;
 Writeln('Inicialmente M es ',M,' y N es ',N,'.');
 Intercambiar(M,N);
 Writeln('Al terminar M es ',M,' y N es ',N,'.')
END.
```