Resolución de Problemas y Algoritmos

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

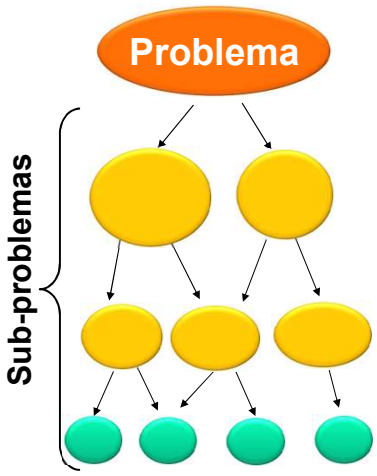
Dr. Diego R. García

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

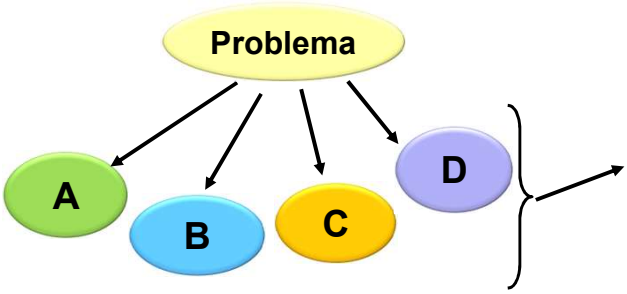
Resolución de problemas con primitivas

En lo que resta de la materia veremos:

1. **Técnicas para resolver problemas más complejos.**
2. **Cómo escribir algoritmos basados en primitivas y algoritmos que son primitivas.**
3. **Cómo implementar en Pascal primitivas (funciones y procedimientos) y poder usar las dos técnicas anteriores.**

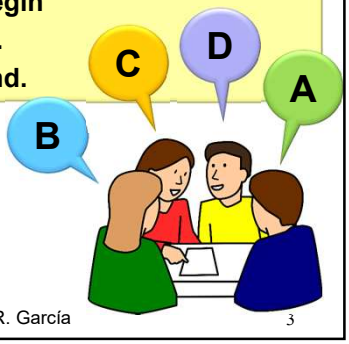


División de un problema en sub-problemas



Program SOLUCIÓN;
- Function A
- Procedure B
- Function C
- Procedure D

Begin
…
End.



Metología: Para resolver un problema complejo se propone:
1) **dividir** en sub-problemas,
2) **resolver** cada sub-problema y luego
3) **para cada parte implementar primitivas** en Pascal: como funciones o procedimientos

Resolución de Problemas y Algoritmos Dr. Diego R. García 3

Repaso de la clase pasada sobre funciones

En Pascal una función tiene:

1. Un **nombre** (con el cual se la invocará desde una expresión).
2. **Parámetros formales** (entre los cuales estarán los datos de entrada).
3. **Tipo del resultado** (determinará en que expresión podrá ser usada)
4. **Variables locales** (que son propias de la función).
5. Sentencias (también llamado "**cuerpo**" de la función).
6. **Asignación** de una expresión al **nombre** de la función (al menos una vez). Es la forma de retornar un valor.

```
FUNCTION Potencia (Base, Exponente: integer) : integer;
{retorna base elevado a la exponente}
VAR aux,resultado: integer;  // variables auxiliares
BEGIN
  resultado := 1;
  FOR aux:= 1 TO Exponente DO resultado := resultado * Base;
  Potencia:= resultado;
END;
```

1 2 3 4 5 6

Resolución de Problemas y Algoritmos Dr. Diego R. García 4

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: error de programación. "Cubo" no es una variable, es el nombre de la función. Si usa el nombre de la función en una expresión 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

CORRECTO:
Una forma de evitar el error de programación anterior es usar una variable local.
"P" si es una variable y puede usarse para almacenar y usar su valor.

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

Resolución de Problemas y Algoritmos Dr. Diego R. García 7

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:

hacer ejemplos/casos de prueba, buscar solución, escribir algoritmo, verificar con casos de prueba y finalmente pasar a 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.

Resolución de Problemas y Algoritmos Dr. Diego R. García 8

Conceptos: parámetros por valor

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR D:digito; Num:Integer;
FUNCTION digito_mas_significativo(N:integer): digito;
BEGIN {retorna el dígito más significativo de un número entero}
   if N < 0 then N:=-1*N; {calcula valor absoluto}
   while (N >= 10) do N:=N div 10; {divide x10 hasta que queda 1 dígito}
   digito_mas_significativo:= N; {retorna el dígito que quedó en N}
END;
BEGIN
write('Ingrese un número:');
readln(Num);
D:=digito_mas_significativo(Num);
writeln('el D.M.S. de', Num, 'es', D);
END.
```

Parámetro por valor: recibe una copia del valor del parámetro efectivo

Parámetro efectivo: se usa su valor para copiarlo al parámetro formal.

¿Podría la variable "Num" del programa llamarse "N"? ¿Qué ocurre si es así?

Copiar la traza del pizarrón ☺

Conceptos: parámetros por valor

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR D:digito; N:Integer;
FUNCTION digito_mas_significativo(N:integer): digito;
BEGIN {retorna el dígito más significativo de un número entero}
   if N < 0 then N:=-1*N; {calcula valor absoluto}
   while (N >= 10) do N:=N div 10; {divide x10 hasta que queda 1 dígito}
   digito_mas_significativo:= N; {retorna el dígito que quedó en N}
END;
BEGIN
write('Ingrese un número:');
readln(N);
D:=digito_mas_significativo(N);
writeln('el D.M.S. de', N, 'es', D);
END.
```

Las dos variables se llaman "N"

El parámetro "N" de la función es por valor, entonces aunque en la función le asigne nuevos valores, estos cambios no afectan a la variable "N" del programa.

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…}
num:=101;  {num es de tipo integer}
d:=digito_mas_significativo(123);
d:=digito_mas_significativo(num div 2);
d:=digito_mas_significativo(num);
```

Parámetros efectivos: valores, expresiones o variables, que sean asignación compatible con el parámetro formal.

Resolución de Problemas y Algoritmos Dr. Diego R. García 11

Conceptos: parámetros por valor

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

```
PROGRAM PruebaDMS;
TYPE digito = 0..9;
VAR N, D :Integer;
FUNCTION digito_mas_significativo(N:integer): digito;
BEGIN {retorna el dígito más significativo d
   if N < 0 then N:=-1*N; {calcula valor abs
   while (N >= 10) do N:=N div 10; {divide x
   digito_mas_significativo:= N; {retorna e
END;
BEGIN
D:=digito_mas_significativo(236);
writeln('el D.M.S. de 236 es', D);
N:=123;
D:=digito_mas_significativo(N*7+N);
writeln('el D.M.S. de', N*7+N,' es', D);
END.
```

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).

Hacer la traza ☺

Resolución de Problemas y Algoritmos Dr. Diego R. García 12

Conceptos: Procedimientos (procedure)

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

Parámetros formales

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.

```
Asteriscos(10) ; // imprime 10 * en pantalla
num:=5; asteriscos(num); // imprime 5 * en pantalla
read(num); asteriscos(num+1)); // imprime * en pantalla
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 13

```
PROGRAM  ejemplos;
VAR tope,i: integer;
```

Variables globales

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

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

```
PROCEDURE Pausa;
BEGIN {Muestra mensaje y espera ENTER}
Writeln ('Presione ENTER para continuar'); Readln;
END ;
```

Obs: no tiene parámetros

```
BEGIN
Pausa;
writeln('ingrese tope'); readln(tope);
FOR i:= 1 to tope DO begin Asteriscos(i); writeln; end;
pausa;
END.
```

Llamadas a procedimiento

Parámetro efectivo

Resolución de Problemas y Algoritmos Dr. Diego R. García 14

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: N1, D1 y N2, D2
- y además 2 datos de salida: el numerador (N) y el denominador (D) del resultado .
- El cálculo de N (numerador) y D (denominador) será de esta forma:

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

Conceptos: Parámetros por referencia

4 parámetros formales por valor

```
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
BEGIN {multiplica dos fracciones: N1/D1 y N2/D2.
   N := N1 * N2; {calcula el numerador}
   D := D1 * D2; {calcula el denominador}
END;
```

2 parámetros formales 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.**

Resolución de Problemas y Algoritmos Dr. Diego R. García 17

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 {multiplica dos fracciones: N1/D1 y N2/D2.
   N := N1 * N2; {calcula el numerador}
   D := D1 * D2; {calcula el denominador}
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

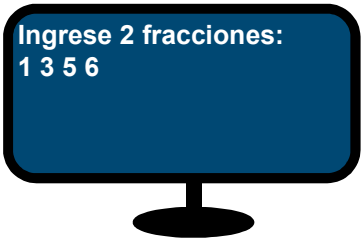
Copiar la traza del pizarrón ☺

Sugerencia: pase a la máquina y ejecute.

Resolución de Problemas y Algoritmos Dr. Diego R. García 18

Traza con parámetros por referencia

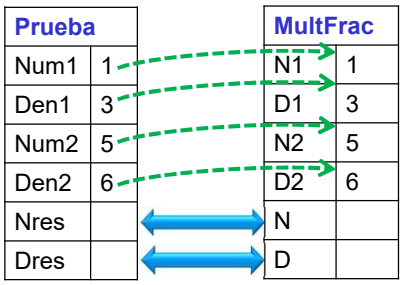
Prueba	
Num1	1
Den1	3
Num2	5
Den2	6
Nres	
Dres	

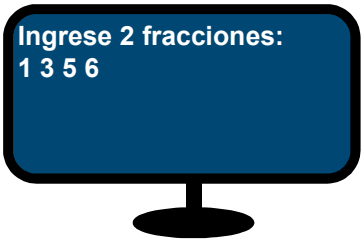


(1) Cuando comienza la ejecución de prueba, se crean 6 variables. Luego de la ejecución de **readln(num1,den1,num2,den2)**; se asignan valores a las cuatro primeras variables. Cosidere el caso de prueba: 1 3 5 6 (que representa a 1/3 y 5/6)

Resolución de Problemas y Algoritmos Dr. Diego R. García 19

Traza con parámetros por referencia

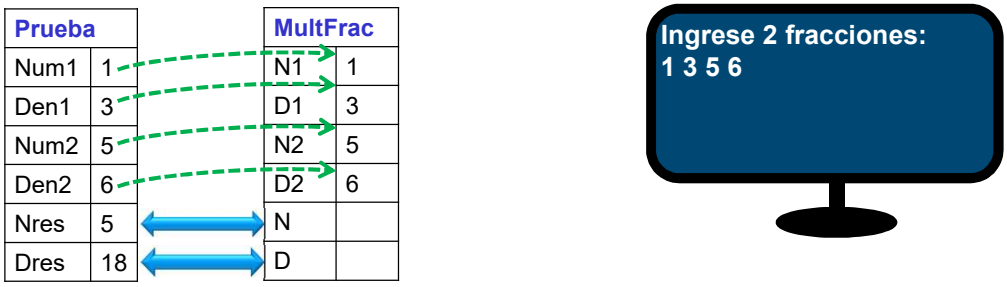




(2) Al llamar al procedimiento **MultFrac** se crean los parámetros por valor (N1,D1,N2,D2) y los parámetros por referencia (N y D). Luego se copian los valores para los parámetros por valor (indicado en la figura con la flecha punteada de simple punta). Además, se crea una referencia (indicada en la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

Resolución de Problemas y Algoritmos Dr. Diego R. García 20

Traza con parámetros por referencia

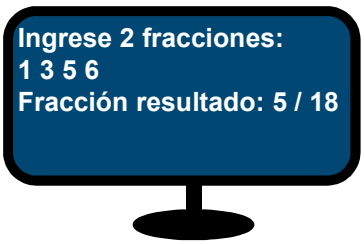


Todo cambio que haga sobre un parámetro formal por referencia, afectará directamente al parámetro efectivo vinculado.

(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: 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.

¿Cuándo usar función o procedimiento?

Para estas primitivas ¿función o procedimiento?

1. **Contar la cantidad de elementos de un archivo.**
2. **Mostrar todos los elementos de un archivo por pantalla.**
3. **Borrar un elemento de un archivo.**
4. **Agregar un elemento a un archivo.**
5. **Ver cuantas veces aparece un dígito en un número.**
6. **Ver cuantas veces aparece un elemento en un archivo.**
7. **Indicar si un elemento está en un archivo.**
8. **Esperar hasta que el usuario presione enter.**

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**; y debe ser de tipo **asignación-compatible con el tipo del parámetro formal**.
  - si corresponde a un **parámetro formal por referencia**, debe ser siempre: → una **variable**; y debe ser de **tipo idéntico al tipo del parámetro formal**.

Ejemplos:

```
PROCEDURE MultFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
...
{…llamadas…}
MultiFrac (1,2,3,4, N,D);
MultiFrac (N,D, 2+2, trunc(2.3)+1, N1, D1);
```

parámetros formales — **parám. efectivos**

Resolución de Problemas y Algoritmos Dr. Diego R. García 25

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.

Resolución de Problemas y Algoritmos Dr. Diego R. García 26

```
Program Ejemplo;
TYPE TipoElemento = Integer;
     TipoArch: FILE OF TipoElemento;
VAR F1:TipoArch;
    PROCEDURE mostrara ( VAR archi: TipoArch; separador:char);
    Var elemento: TipoEelemento;
    begin  {... muestra el contenido de un archivo de números enteros
                usando un "separador" enviado por parámetro…}
        Reset(archi);
         while not eof(archi) do  begin
            read(archi, elemento);  write(elemento,' ', separador ,' ' );
          end; {while}
        close(archi);
    End;
begin  {programa}
    assign(F1, 'mis-numeros.datos');
    mostrarA(F1, ';');
end. {fin del programa}
```

Con archivos es obligatorio usar parámetros por referencia

Procedimientos y parámetros en Pascal

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

```
PROCEDURE  MultiplicaFracciones  ( N1, D1, N2, D2: INTEGER;  VAR NR,DR: INTEGER);
BEGIN  {multiplica dos fracciones N1/D1 y N2/D2}
    NR := N1 * N2; {calcula el numerador}
    DR := D1 * D2; {calcula el denominador}
END;
```

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

```
PROCEDURE  Multiplica_2_Fracciones  (VAR NR,DR: INTEGER; N1, D1, N2, D2: INTEGER );
BEGIN  {multiplica dos fracciones.
Datos de entrada: N1/D1 y N2/D2.
Datos de salida: NR/DR}
    NR := N1 * N2; {calcula el numerador}
    DR := D1 * D2; {calcula el denominador}
END;
```

El programador debe tener cuidado con el orden en los parámetros efectivos. Por ello el programador creador de la primitiva debe documentar muy bien el código y cuales son los datos de entrada y salida.

En "EJEM" ¿cuáles son parámetros por valor?

```
PROCEDURE  EJEM  (VAR A: char; B, C: char; VAR D,E: real; F: real; G:integer );
…
```

Procedimientos y parámetros en Pascal

¿ Puede un procedimiento tener sólo parámetros por referencia?

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

¿ Puede un procedimiento no tener parámetros?

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

Resolución de Problemas y Algoritmos Dr. Diego R. García 29

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; END; {o false en caso contrario}
END;
```

¿Qué diferencia hay con tener una función como esta?

```
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; END {o false en caso contrario}
END;
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 30

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;
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 31

Funciones y parámetros en Pascal

¿ Puede una función tener parámetros por referencia?

```
TYPE   TipoElemento= integer; ARCHI: FILE OF TipoElemento;
```

```
FUNCTION  cantidad_elementos (VAR A:ARCHI ):integer;
var aux: TipoElemento; cant:integer;
BEGIN {retorna la cantidad de elementos de un archivo}
cant:= 0;
reset(A);
while not  eof(A) do { por cada elemento leído suma uno}
     begin  read(A,aux);  cant:=cant+1;    end;
cantidad_elementos := cant;
close(A);
END;
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 32

```
program Reflexion4; {El objetivo de este programa es hacer una traza y
reflexionar sobre el pasaje de parámetros por referencia. }
var v1,v2,v3,v4:integer; {quedó un poco compacto para que entre en una "hoja"}
procedure P3 (var R, C, Z:integer; N:integer);
  var local: integer;
  begin  writeln('Entro a P3 con ', R:9, N:9);
   local:= 3;  N:= local+N; R:=N; C:=0; Z:=R;
   writeln('Salgo de P3 con ', local, R:9, C:9, Z:9, N:9); end;
procedure P2 (var R, C, W:integer; N:integer);
  var local: integer;
  begin  writeln('Entro a P2 con ', R:9, N:9);
   local:= 2;  P3 (local,C,W,N+1); R:=local+N; C:=C+1;
   writeln('Salgo de P2 con ', local, R:9, C:9, W:9, N:9); end;
procedure P1 (var R, C, X:integer; N:integer);
  var local: integer;
  begin  writeln('Entro a P1 con ', R:9, N:9);
   local:= 1;  P2 (local,C,X,N+1); R:=local+N; C:=C+1;
   writeln('Salgo de P1 con ', local, R:9, C:9, X:9, N:9); end;
begin v1:=5; v4:=1; P1(v1,v2,v3,v4); write('finalizo con', v1,v2,v3,v4); end.
```

Tarea: Primero haga una traza en papel (bien prolija) y luego ejecute en su computadora para comparar. (Puede agregar mas "writeln"s.)

Preguntas para reflexionar

La siguientes preguntas son sobre el programa "reflexion4", (antes de responderlas tiene que hacer la traza)

- **(fácil): ¿cuáles son parámetros por referencia?**
- **La variable v2 no tiene valor al ser usada en el parámetro efectivo de la llamada a P1, ¿es un error de programación?**
- **La variable v1 si tiene valor ¿es un error? ¿es mejor?**
- **¿Qué ocurriría si en P3 no se hiciera C:=0?**