



Resolución de Problemas y Algoritmos

Clase 13
Lenguaje Pascal: estructura de bloques, entornos de referencia, visibilidad de identificadores. Pasaje de parámetros



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

(repasso) vimos en la clase anterior que ...

Las funciones predefinidas:

- Se utilizan en una expresión.
- Siempre retornan un valor de un tipo de Pascal.

Ejemplos:

- EOF(F): recibe un manejador y retorna boolean
- TRUNC(R): recibe real y retorna integer
- SQRT(R): recibe real y retorna real
- CHR(I): recibe integer y retorna char

En Pascal puedo construir mis propias funciones

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 2

(repasso) vimos en la clase anterior que ...

Los procedimientos predefinidos:

- Se los usa en una sentencia.
- Pueden tener 0 o más parámetros.

Ejemplos:

- Writeln
- Readln
- reset (F)
- Rewrite (F)
- Assign (F, nombre)

En Pascal puedo construir nuevos procedimientos

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 3

(repasso) Conceptos: funciones

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

```

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

Ejemplo de llamada:

```

...
IF esMayuscula('A') = TRUE
then ...
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 4

(repasso) procedimientos...

- Se invocan como sentencia.
- Al regresar de la invocación se ejecuta la sentencia siguiente a la llamada.
- No tienen un tipo asociado.
- Pueden no retornar un valor.

```

PROCEDURE Pausa;
    {Muestra un mensaje y espera ENTER}
BEGIN
    Write('Press ENTER to continue');
    Readln;
END;
    
```

```

PROCEDURE MultiplicarFracciones
    (N1, D1, N2, D2 : INTEGER;
    VAR NumRes,DenRes: INTEGER);
BEGIN
    NumRes := N1 * N2;
    DenRes := D1 * D2;
END;
    
```

Ejemplo de llamadas:

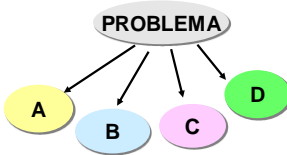
```

...
MultiplicarFracciones(2,3,4,5,N,D);
Write(Resultado: ',N','/','D);
Pausa;
...
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 5

División del problema en subproblemas

PROBLEMA



Program SOLUCIÓN;

Function A

Procedure B

Function C

Procedure D

Begin

End.

Metología: Para resolver un problema complejo se propone:

- 1) dividirlo en subproblemas,
- 2) resolver cada parte y luego
- 3) para cada parte implementar PRIMITIVAS en Pascal: como funciones o procedimientos

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 6

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.

Pascal es estructurado por bloques

```

PROGRAM MIPROGRAMA;
CONST ..... TYPE..... VAR .....
FUNCTION F(X:real):real;
CONST ..... TYPE.....
VAR .....
  PROCEDURE ...
  FUNCTION ...
BEGIN ...sentencias...END;

PROCEDURE P(Var X: char);
CONST..... TYPE.....
VAR .....
  PROCEDURE ...
  FUNCTION ...
BEGIN...sentencias...END;
BEGIN
  ...sentencias...
END.
    
```

- En Pascal, un **programa** constituye un **bloque** compuesto por:
 - Constantes, tipos, Variables,
 - funciones, procedimientos,
 - y sentencias.
- Cada **procedimiento** o **función** también constituye un **bloque** con:
 - parámetros
 - constantes, tipos, variables,
 - procedimientos, funciones,
 - y sentencias.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 7

En Pascal no hay límite en cantidad o anidamiento de bloques

```

PROGRAM PROGRAMA1;
PROCEDURE...
FUNCTION ...
PROCEDURE...
PROCEDURE...
FUNCTION ...
PROCEDURE...
{...puede incluir todos los proc. o fn. que quiera...}
BEGIN ... END.

PROGRAM PROGRAMA2;
PROCEDURE ...
  FUNCTION ...
    PROCEDURE ...
      BEGIN ..... END;
    BEGIN ..... END;
  BEGIN ..... END;
END.
    
```

{...y en c/bloque, todo el "anidamiento" que quiera}

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 8

Pascal: estructurado por bloques

- En un programa pueden incluirse tantos procedimientos y funciones como se desee.
- Cada uno de ellos puede a su vez tener sus bloques internos y así siguiendo.
- Esto permite implementar cualquier división del problema en subproblema que se diseñe.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 9

Estructurado por Bloques

- Elementos de un BLOQUE:**
 1. identificadores de constantes
 2. identificadores de tipos
 3. identificadores de variables
 4. identificadores parámetros formales (en proc. y fn.)
 5. identificadores de procedimientos
 6. identificadores de funciones
 7. sentencias
- Dentro de un mismo bloque no puede haber dos identificadores iguales para distintos elementos.
- Dos elementos pueden tener el mismo identificador si pertenecen a diferentes bloques.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 10

(P) Bloques e identificadores

```

PROGRAM simple; {para entender los conceptos}
Const Pi= 3.14; type Tdig=0..9; var A, B, C:CHAR;
PROCEDURE P1 (A:REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE(B) end;
PROCEDURE P2 (A:REAL);
var B, MIA: real;
FUNCTION F2 (A:REAL):REAL;
var B, DE_F2: REAL
begin B:= A; F2:= B + Pi; end;
begin B:= A; WRITE(F2(A)); P1(B) end;
BEGIN
P2(5); P1(10);
END.
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 11

(P) Bloques (demarcados) e identificadores

```

PROGRAM simple; {para entender los conceptos}
Const Pi= 3.14; type Tdig=0..9; var A, B, C:CHAR;
PROCEDURE P1 (A:REAL);
var B: REAL; F2: Tdig;
begin B:= A; WRITE(B) end;
PROCEDURE P2 (A:REAL);
var B, MIA: real;
FUNCTION F2 (A:REAL):REAL;
var B, DE_F2: REAL
begin B:= A; F2:= B + Pi; end;
begin B:= A; WRITE(F2(A)); P1(B) end;
BEGIN
P2(5); P1(10);
END.
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 12

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.

Preguntas sobre el programa "simple" simple

- ¿puedo llamar a P1 desde las sentencias de P2?
- ¿puedo llamar a F2 desde las sentencias de P2?
- ¿puedo llamar a F2 desde las sentencias de P1?
- ¿puedo llamar a P1 desde las sentencias de F2?
- ¡ **HAGA AHORA SUS PREGUNTAS !** (y copie las de sus compañeros)
- Pregunta más general: ¿desde qué lugar del programa puedo llamar a una función o proced.??
- ¿en qué bloques puedo usar la variable "MIA"?
- ¿y la variable DE_F2?
- ¿en qué bloques puedo usar una variable?
- **Todas las respuestas en la teoría que sigue a continuación...**

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 13

Vocabulario: declaración vs. referencia simple

Es importante distinguir entre:

1. La **declaración** de un identificador de constante, tipo, variable, parámetro, función, o procedimiento. **Ejemplos:**
`CONST pi=3.14; TYPE archi: FILE OF integer;`
`VAR precio: real; a_pagar: integer;`
`PROCEDURE recargo(precio, rec:real; var monto:real);`
`FUNCTION intereses(monto:integer):real`
2. La **referencia** (uso) de un identificador. **Ejemplos:**
`recargo(24,incremento,precio);`
`a_pagar := intereses(round(precio));`

En cada bloque, se declaran identificadores; y además, se hace referencia (usan) identificadores.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 14

Concepto: Entorno de referencia para un bloque B

El entorno de referencia de un bloque B simple
 está formado por:

- El **entorno local:** parámetros formales, constantes, tipos y variables declarados dentro de B y el nombre de los procedimientos y funciones declarados dentro del bloque B.
- El **entorno global:** conjunto de identificadores declarados en el bloque del programa principal
- El **entorno no-local:** conjunto de identificadores declarados en los bloques que contienen al bloque B, exceptuando al global
- El **entorno predefinido:** conjunto de identificadores ya declarados por el compilador de Pascal y disponible para todo programa (Ej: maxint, char, write, eof).

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 15

Conceptos: identificadores ocultos simple

Cuando se hace **referencia a un identificador:**

1. primero se busca en su entorno de referencia local,
2. luego en su entorno de referencia no local,
3. luego en su entorno de referencia global,
4. y finalmente en el entorno de referencia predefinido

Por lo anterior, **si hay identificadores iguales en diferentes entornos uno oculta al otro.**

1. Un identificador de nombre N en un entorno local **oculta** a todo identificador del mismo nombre N en otro entorno (no-local, global, predefinido)
2. Uno no-local N **oculta** a otro global de nombre N,
3. Un identificador global N **oculta** a uno predefinido N

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 16

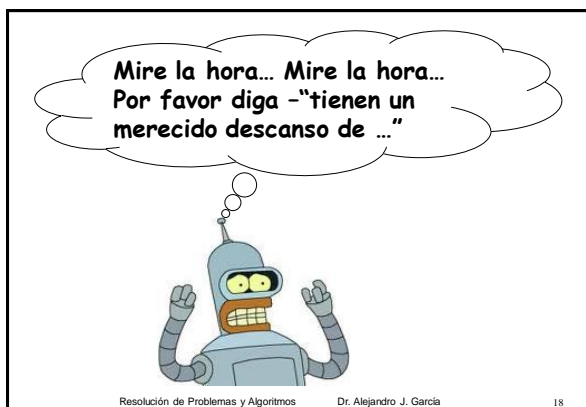
Conceptos y vocabulario simple

- Un identificador es **referenciable** en un bloque, si es parte de su entorno de referencia y no está oculto.
- Un identificador es **visible**, si es referenciable.
- El **alcance** de un identificador D, son aquellas sentencias (o bloques) del programa donde el identificador D es visible.

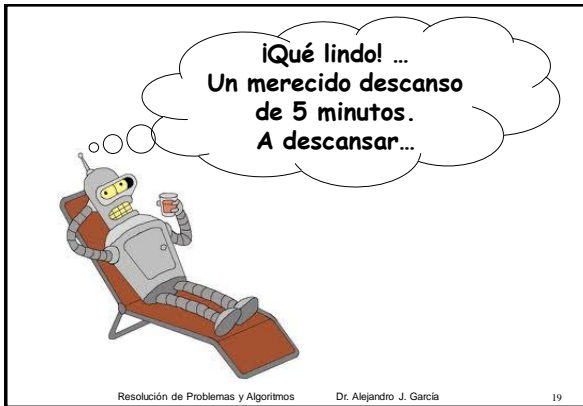
Ejercicios propuestos:

- Para cada bloque del programa **simple**, encuentre los identificadores visibles (referenciables).
- Indique el alcance del identificador P1 y el alcance de la variable global A.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 17



El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.



```

PROCEDURE Mayuscula ( Letra:char; VAR Mayu:integer);
Begin
Mayu:= chr(Ord(letra)-32);
end;
    
```

....
 carac1:='b';
 Mayuscula(carac1,carac2);
 writeln(car1,car2);
 ...

Parámetros formales
 Parámetros efectivos

copia valor
 referencia

Diferencias entre los tipos de parámetros formales:
P. por valor: el valor del parámetro efectivo **Carac1** se copia al formal correspondiente (**Letra**), las modificaciones a **Letra** no afectan a la variable **Carac1**.
P. por referencia: se crea un referencia entre **Carac2** y **Mayu**. Todo cambio en **Mayu** afecta y cambia a **Carac2**.

La llamada a **FUNCTION** debe realizarse desde una expresión:
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;
    
```

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

```

BEGIN
write('Ingresa base y exponente:');
readln(B,E);
Pot:=Potencia(B,E);
B:= B+1; E:=E+1;
writeln(pot,'<>', potencia(B,E));
END.
    
```

Parámetros efectivos

Conceptos: Parámetros en Funciones y Procedimientos

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 MultiFrac (N1,D1,N2,D2:integer; VAR N, D:integer);
    
```

Ejemplos:
 p. formales → MultiFrac (1,2,3,4, N,D);
 p. efectivos → MultiFrac (N,D, 2+2, trunc(2.3)+1, N1, D1);

Conceptos: compatibilidad entre parámetros

- 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 real.

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

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**.

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.

Traza con parámetros por valor o referencia

- Elementos de un BLOQUE:**
 1. identificadores de constantes
 2. identificadores de tipos
 3. identificadores de variables
 4. identificadores parámetros formales (en proc. y fn.)
 5. ...

Cambian sus valores en ejecución

Al realizar una traza hay que tener en cuenta que:

- cada bloque tiene sus propios identificadores,
- los parámetros formales reciben datos de los parámetros efectivos correspondientes, y
- los parámetros por referencia modifican los valores de sus correspondientes efectivos.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 25

Traza con llamadas a primitivas

- Para reflejar lo que sucede en ejecución (esto es **dinámicamente**) y poder hacer una traza, se usará una "tabla" (o "cajita"). También llamada Registro de Activación.
- Esa tabla contendrá un espacio por cada parámetro formal o cada variable local del mismo.
- Cada vez que se **invoca** un procedimiento o función se **crea** una nueva tabla (o "cajita").
- Cada vez que **finaliza** un procedimiento o función se **destruye** la tabla correspondiente.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 26

Traza con llamadas a primitivas

| |
|---------------------------|
| Parámetros por valor |
| Parámetros por referencia |
| Variables Locales |
| Resultado de la función |

Para cada bloque que se ejecuta.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 27

Ejemplo de traza con parámetros por valor

```

PROGRAM Ejemplo2; {ej. traza parámetros por valor}
VAR M,N: integer;
PROCEDURE EjemploValor (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,',');
EjemploValor(M,N);
Writeln('Al terminar M es ',M,' y N es ',N,',');
END.
    
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 28

Ejemplo de traza con parámetros por valor

```

PROGRAM Ejemplo2; {ej. traza parámetros por valor}
VAR M,N: integer;
PROCEDURE EjemploValor (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,',');
EjemploValor(M,N);
Writeln('Al terminar M es ',M,' y N es ',N,',');
END.
    
```

| | |
|---|----|
| M | ?? |
| N | ?? |

Al comenzar la ejecución se crea el bloque. Las variables aún no tienen valor.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 29

Ejemplo de traza con parámetros por valor

```

PROGRAM Ejemplo2; {ej. traza parámetros por valor}
VAR M,N: integer;
PROCEDURE EjemploValor (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,',');
EjemploValor(M,N);
Writeln('Al terminar M es ',M,' y N es ',N,',');
END.
    
```

| | |
|---|---|
| M | 0 |
| N | 9 |

Antes de llamar al procedimiento "EjemploValor" las variables M y N ya tienen valor asignado.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 30

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.

Ejemplo de traza con parámetros por valor

```
PROGRAM Ejemplo2; {ej. traza parámetros por valor}
VAR M,N: integer;
PROCEDURE EjemploValor (A, B: integer);
VAR Aux: integer;
BEGIN
write( A, B);
Aux := A ; A := aux + 1; B := Aux;
write( A, B, Aux);
END;
```

Al entrar al procedimiento se crea un nuevo bloque y los parámetros toman los valores que les fueron enviados. Las variables locales aún no tienen valor.

| | |
|-----|----|
| M | 0 |
| N | 9 |
| A | 0 |
| B | 9 |
| aux | ?? |

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 31

Ejemplo de traza con parámetros por valor

```
PROGRAM Ejemplo2; {ej. traza parámetros por valor}
VAR M,N: integer;
PROCEDURE EjemploValor (A, B: integer);
VAR Aux: integer;
BEGIN
write( A, B);
Aux := A ; A := aux + 1; B := Aux;
write( A, B, Aux);
END;
```

Antes de terminar el procedimiento los valores de A, B y aux cambiaron

| | |
|-----|---|
| M | 0 |
| N | 9 |
| A | 1 |
| B | 0 |
| aux | 0 |

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 32

Ejemplo de traza con parámetros por valor

```
PROGRAM Ejemplo2; {ej. traza parámetros por valor}
VAR M,N: integer;
PROCEDURE EjemploValor (A, B: integer);
VAR Aux: integer;
BEGIN
write( A, B);
Aux := A ; A := aux + 1; B := Aux;
write( A, B, Aux);
END;
```

Al volver del procedimiento "EjemploValor" las variables M y N conservan su valor asignado.

| | |
|---|---|
| M | 0 |
| N | 9 |

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 33

Traza del programa Ejemplo2

| | | | |
|-----|--------|-------|-----|
| (1) | (2) | (3) | (4) |
| M 0 | M 0 | M 0 | M 0 |
| N 9 | N 9 | N 9 | N 9 |
| | A 0 | A 1 | |
| | B 9 | B 0 | |
| | aux ?? | aux 0 | |

(1) Antes de llamar al procedimiento "EjemploValor"
 (2) Al entrar al procedimiento
 (3) Antes de salir del procedimiento
 (4) Después de salir del procedimiento

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 34

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.
```

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 35

Ejemplo de intercambio de valores

Cuando comienza el programa IntercambiarVariables.

| | |
|---|--|
| M | |
| N | |

IntercambiarVariables

Después de ejecutar las asignaciones M y N quedan con los siguientes valores:

| | |
|---|---|
| M | 5 |
| N | 9 |

IntercambiarVariables

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 36

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.

Ejemplo de intercambio de valores

Cuando se invoca al procedimiento **Intercambiar** debe crearse un bloque para ese procedimiento.

| | |
|---|---|
| M | 5 |
| N | 9 |

} IntercambiarVariables

| | |
|-----|--|
| A | |
| B | |
| Aux | |

} Intercambiar

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 37

Pasaje de parámetros por referencia

Cuando un parámetro es pasado por referencia, el parámetro formal **"es una referencia"** al parámetro real.

| | |
|---|---|
| M | 5 |
| N | 9 |

} IntercambiarVariables

| | |
|-----|---|
| A | ● |
| B | ● |
| Aux | |

} Intercambiar

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 38

Pasaje de parámetros por referencia

Cuando un parámetro es pasado por referencia, **cada** cambio en el parámetro formal **"afecta"** al parámetro real.

| | |
|---|---|
| M | 9 |
| N | 5 |

} IntercambiarVariables

| | |
|-----|---|
| A | ● |
| B | ● |
| Aux | 5 |

} Intercambiar

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 39

Ejemplo de intercambio de valores

Cuando termina la ejecución de **Intercambiar** queda el siguiente estado:

| | |
|---|---|
| M | 9 |
| N | 5 |

} IntercambiarVariables

En pantalla se imprimirá lo siguiente:

Inicialmente M es 5 y N es 9.
Al terminar M es 9 y N es 5.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 40

Problema propuesto (1)

```
PROGRAM Propuesto1;
VAR M,N: integer;
PROCEDURE EjemploValor_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,');
EjemploValor_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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 41

Problema propuesto (2)

```
PROGRAM Propuesto2;
VAR M,N: integer;
PROCEDURE EjemploRef (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,');
EjemploRef(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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 42

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"Resolución de Problemas y Algoritmos. Notas de Clase". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2012.