

## RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

### CLASE 4

Estructura de control condicional.

0 1 1 0 0  
 1 0 0 1 1  
 1 0 1 1 0  
 0 1 1 1 0  
 0 1 1 1 0  
 0 1 1 0 0  
 1 0 0 1 1  
 1 0 1 1 0  
 0 1 1 1 0  
 1 0 0 1 1  
 1 1 1  
 0 0 0  
 1

Luciano H. Tamargo  
<http://cs.uns.edu.ar/~lt>  
 Depto. de Ciencias e Ingeniería de la Computación  
 Universidad Nacional del Sur, Bahía Blanca  
 2016

### CONCEPTOS DE LA CLASE PASADA

- Algoritmo. Primitiva. Traza.
  - Programa. Código fuente.
- Lenguaje de programación.
  - Programa. Código fuente.
- Pascal:
  - Identificadores reservados y predefinidos
  - Constantes, variables y tipos de datos.
  - Primitivas: asignación (:=), read, readln, write, writeln
  - Tipos predefinidos: real, integer, char, boolean.
  - Expresiones. Operaciones y funciones predefinidas.

¿PREGUNTAS?

Resolución de Problemas y Algoritmos - 2016 2

### PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  - primero** se evalúa la expresión de derecha y se obtiene un valor,
  - luego** se modifica el valor de la variable, perdiéndose el valor anterior.
- Problema: Intercambiar los valores de dos variables **a** y **b** de forma tal que el valor de **a** quede en **b** y el de **b** quede en **a**.

MAL

```
PROGRAM IntercambiaMAL;
VAR a, b: integer;
BEGIN
  write('Ingrese 2 enteros');
  read(a,b);
  a:= b;
  b:= a;
  writeln(a, ' ', b);
END.
```

Observe que IntercambiaMAL es sintácticamente válido y aún así tiene un error.

### PRIMITIVA DE ASIGNACIÓN

- En una asignación: variable := expresión
  - primero** se evalúa la expresión de derecha y se obtiene un valor,
  - luego** se modifica el valor de la variable, perdiéndose el valor anterior.
- Problema: Intercambiar los valores de dos variables **a** y **b** de forma tal que el valor de **a** quede en **b** y el de **b** quede en **a**.

MAL

```
PROGRAM IntercambiaMAL;
VAR a, b: integer;
BEGIN
  write('Ingrese 2 enteros');
  read(a,b);
  a:= b;
  b:= a;
  writeln(a, ' ', b);
END.
```

Traza de valores en memoria		
	a	b
	?	?
←	1	?
←	1	5
←	5	5
←	5	5

Observe que IntercambiaMAL es sintácticamente válido y aún así tiene un error.

### CONTENEDORES DE ELEMENTOS DE CIERTO TIPO

- Las **variables** pueden pensarse como recipientes que pueden contener un cierto tipo de elemento.
- Por ejemplo**, un vaso es un recipiente pensado para contener elementos de tipo líquido.
- Si tengo un vaso con una gaseosa, para poder tener ese mismo vaso con chocolatada, debo sacar la gaseosa para poder colocar la chocolatada (y el vaso ya no tendrá gaseosa).
- Piense ahora como resuelve el siguiente problema de dos hermanos pequeños: por error la taza de Mateo tiene jugo y la de María chocolatada, pero los niños quieren tomar cada uno en su propia taza lo que tiene el otro. Usando los líquidos que ya están servidos en las tazas ¿Cómo hace para intercambiar los líquidos de taza?



### INTERCAMBIAR LOS VALORES DE LAS VARIABLES

- En una asignación: variable := expresión
  - primero** se evalúa la expresión de derecha y se obtiene un valor,
  - luego** se modifica el valor de la variable, perdiéndose el valor anterior.
- Problema: Intercambiar los valores de dos variables **a** y **b** de forma tal que el valor de **a** quede en **b** y el de **b** quede en **a**.

BIEN

```
PROGRAM IntercambiaBIEN;
VAR a, b, aux: integer;
BEGIN
  write('Ingrese 2 enteros');
  read(a,b);
  aux:= a;
  a:= b;
  b:= aux;
  write(a, ' ', b);
END.
```

Preservo el valor de a en aux.

# Resolución de Problemas y Algoritmos

### INTERCAMBIAR LOS VALORES DE LAS VARIABLES

- En una asignación: variable := expresión
  - primero** se evalúa la expresión de derecha y se obtiene un valor,
  - luego** se modifica el valor de la variable, perdiéndose el valor anterior.
- Problema: Intercambiar los valores de dos variables **a** y **b** de forma tal que el valor de **a** quede en **b** y el de **b** quede en **a**.

**BIEN**

Preservo el valor de a en aux.

```

PROGRAM IntercambiaBIEN;
VAR a, b, aux: integer;
BEGIN
  write('Ingrese 2 enteros');
  read(a,b);
  aux:= a;
  a:= b;
  b:= aux;
  write(a, ' ', b);
END.
                
```

a	b	aux
?	?	?
1	?	?
1	5	?
1	5	1
5	5	1
1	5	1

### METODOLOGIA GENERAL PROPUESTA

```

PROBLEMA
  ↓
SOLUCIÓN
  ↓
ALGORITMO
  ↓
verificación
  ↓
PROGRAMA
  ↓
verificación
                
```

Veremos a continuación como codificar en el lenguaje Pascal una estructura condicional de la forma:

**Si ...condición... entonces .... de lo contrario ...**

Resolución de Problemas y Algoritmos - 2016

### SENTENCIAS EN PASCAL

- Las sentencias (*statements*) en Pascal pueden ser **simples** o **compuestas**. En la jerga informática, la palabra *statement* se traduce al castellano a estas palabras: **sentencia, instrucción o proposición.**

Sentencia (statement)

- simple { a:=1
- compuesta {
 

```

BEGIN
  PrecioBase := 200;
  Iva:= Precio * 0.20;
  PrecioFinal:= PrecioBase+ iva
END
                    
```

### SENTENCIA COMPUESTA EN PASCAL

- Una sentencia compuesta comienza con **BEGIN** y termina con **END** y permite definir una secuencia de sentencias como si fuera una única sentencia.
- Por ejemplo, la siguiente es una sentencia compuesta, a su vez, por tres sentencias simples.

```

BEGIN
  PrecioBase := 200;
  Iva:= Precio * 0.20;
  PrecioFinal:= PrecioBase + iva
END
                
```

El punto y coma es un **separador de sentencias.**

En la última sentencia de una secuencia el ";" **no es necesario** ya que no hay otra para separar de la última.

Resolución de Problemas y Algoritmos - 2016

### ESTRUCTURA DE CONTROL CONDICIONAL (IF-THEN-ELSE)

```

IF {expresión lógica} THEN
  {Sentencia (simple o compuesta)}
ELSE
  {Sentencia (simple o compuesta)}
                
```

```

IF (A>0) and (A<maxint) THEN
  BEGIN
    a := a + 1;
    write(a);
  END
ELSE
  BEGIN
    a := 1;
    write('error');
  END
                
```

- Sintaxis:** ver el diagrama sintáctico.

Resolución de Problemas y Algoritmos - 2016

### ESTRUCTURA DE CONTROL CONDICIONAL (IF-THEN-ELSE)

```

IF {expresión lógica} THEN
  BEGIN
    a := a + 1;
    write(a);
  END
ELSE
  BEGIN
    a := 1;
    write('error');
  END
                
```

- Semántica:**
  - Si la evaluación de la **expresión lógica** da resultado **true**, entonces se ejecuta únicamente la sentencia que sigue al **"THEN"** (ya sea una sentencia simple o compuesta).
  - Si en cambio, la evaluación de la **expresión lógica** da **false**, entonces se ejecuta solamente la sentencia que sigue al **"ELSE"**.

Resolución de Problemas y Algoritmos - 2016

# Resolución de Problemas y Algoritmos

**PROBLEMA SIMPLE PROPUESTO**

**Problema:** Escriba un programa en Pascal para obtener el valor absoluto de un número.

- Solución:**  
Si el número es positivo o cero, el valor absoluto es el mismo número, de lo contrario es el número multiplicado por -1.

**Algoritmo:**  
 Leo Número  
 Si Número >= 0 entonces: val\_abs es Número  
 de lo contrario: val\_abs es Número \* -1  
 Muestro val\_abs en pantalla

**Verificación:**  
 Ejemplos significativos para **casos de prueba:**  
 un número positivo, uno negativo y cero.  
 Ejemplo: 3, 0 y -3

**PROGRAMA PARA VALOR ABSOLUTO**

```
PROGRAM valor_absoluto;
VAR
  numero, val_abs: real;
{Realiza el cálculo del valor absoluto de un número}
BEGIN
  write ('Ingrese un número');
  readln(numero);
  IF numero >= 0 THEN
    val_abs := numero
  ELSE
    val_abs := (-1) * numero;
  writeln(' Su valor absoluto es: ', val_abs);
END.
```

Realice trazas para los casos de prueba: 3, 0 y -3

Observe: no lleva "," antes del ELSE

**ESTRUCTURA DE CONTROL CONDICIONAL (IF-THEN)**

IF expresión lógica THEN Sentencia 1 (simple o compuesta)

IF expresión lógica THEN BEGIN ... END

- Sintaxis:** vea en el diagrama sintáctico que en la sentencia IF no es obligatorio un ELSE (es opcional).
- Semántica:** Si la evaluación de la expresión lógica da resultado **true**, entonces se ejecuta solamente la sentencia que sigue al "THEN" (sea simple o compuesta).
- Si en cambio la evaluación de la expresión lógica da **false**, se sigue con la ejecución de la sentencias que siguen al IF (si es que existen).

**OTRO PROGRAMA PARA VALOR ABSOLUTO**

Otra solución sin usar ELSE

```
PROGRAM valor_absoluto;
VAR
  numero, val_abs: real;
{Realiza el cálculo del valor absoluto de un número}
BEGIN
  write ('Ingrese un número'); readln(numero);
  val_abs:= numero;
  IF numero < 0 THEN
    val_abs := (-1) * numero;
  writeln(' Su valor absoluto es: ', val_abs);
END.
```

Realice trazas para los casos de prueba: 3, 0 y -3

**PROBLEMA SIMPLE PROPUESTO**

**Problema:** Considerando únicamente las letras a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z; escriba un programa que lea un carácter y distinga si se trata de una letra mayúscula o minúscula.

- Obs.:** para simplificar no incluimos las vocales con acentos ni la letra Ñ, pero lo haremos más adelante.
- Solución:** un carácter ASCII entre 'A' y 'Z' es una letra mayúscula, un carácter entre 'a' y 'z' es una letra minúscula.

**Algoritmo:**  
 leer el carácter  
 Si está entre 'A' y 'Z' entonces es una mayúscula  
 Si está entre 'a' y 'z' entonces es una minúscula

**Verificación:**  
 - casos de prueba: una mayúscula, una minúscula y un carácter que no sea una letra (ejemplos: 'G', 'g', '3', '\$').

**NO CANTEMOS VICTORIA**

```
PROGRAM Veamos;
VAR
  ch: char;
{Este programa intenta distinguir mayúsculas y minúsculas}
BEGIN
  write('Ingrese un caracter:');
  readln(ch);
  IF (ch >= 'A') and (ch <= 'Z') THEN
    writeln(ch, ' es una mayúscula.')
  ELSE
    writeln(ch, ' es una minúscula.');
```

Realice trazas para los casos de prueba: 'G', 'g', '3' y '\$'

La traza para '3' y para '\$' muestra que hay un ERROR. Usando ELSE, cualquier CHAR que no sea mayúscula se considera minúscula, lo cual es incorrecto.

# Resolución de Problemas y Algoritmos

**NO CANTEMOS VICTORIA**

```
PROGRAM mayucula_o_minuscula;
VAR
ch: char;
{Este programa permite distinguir mayúsculas y
minúsculas}
BEGIN
write('Ingrese un caracter:');
readln(ch);
IF (ch >= 'A') and (ch <= 'Z') THEN
writeln(ch, ' es una mayúscula. ');
IF (ch >= 'a') and (ch <= 'z') THEN
writeln(ch, ' es una minúscula. ');
END.
```

Realice trazas para los casos de prueba: 'G', 'g', '3' y '\$'

**ENCUENTRE EL ERROR...**

```
PROGRAM EncuentreError;
VAR
ch: char;
{Este programa tiene un error}
BEGIN
write('Ingrese un caracter:');
readln(ch);
IF (ch >= 'A') and (ch <= 'Z') THEN
writeln(ch, ' es una mayúscula. ');
IF (ch >= 'a') and (ch <= 'z') THEN
writeln(ch, ' es una minúscula. ');
ELSE
writeln(ch, 'no es una letra');
END.
```

Realice trazas para el caso de prueba: 'G'

**CONDICIONALES "ANIDADOS"**

```
IF <exp. boolean E1> THEN
{Sentencia
{simple o compuesta};
ELSE
{Sentencia
{simple o compuesta};

IF <exp. boolean E1> THEN
IF <exp. boolean E2> THEN
<sentencia>
ELSE
<sentencia>
ELSE
IF <exp. boolean E3> THEN
<sentencia>
ELSE
<sentencia>
```

**CONDICIONALES "ANIDADOS"**

```
IF <exp. boolean E1> THEN
IF <exp. boolean E2> THEN
<s1>
ELSE
<s2>
ELSE
IF <exp. boolean E3> THEN
<s3>
ELSE
<s4>
```

¿Bajo que condiciones se ejecutará la sentencia s3?  
 ¿El resultado de E2 afecta a la ejecución de s3?

**CONDICIONALES "ANIDADOS"**

```
IF <exp. boolean E1> THEN
IF <exp. boolean E2> THEN
<s1>
ELSE
IF <exp. boolean E4> THEN
<s5>
ELSE
<s6>
ELSE
IF <exp. boolean E3> THEN
<s3>
ELSE
<s4>
```

¿Bajo que condiciones se ejecutará la sentencia s6?

**¿TIENEN EL MISMO EFECTO?**

**Secuencia de condicionales**

```
IF ( A > 10 ) THEN
write(1);
IF ( B = 0 ) THEN
write(2);
IF ( C > 20 ) THEN
write(3);
```

**Condicionales ANIDADOS**

```
IF ( A > 10 ) THEN
write(1)
ELSE
IF ( B = 0 ) THEN
write(2)
ELSE
IF ( C > 20 ) THEN
write(3);
```

- Realice diferentes trazas con los siguientes casos de prueba
  - A = 20, B = 0, C = 100
  - A = 1, B = 0, C = 100
  - A = 1, B = 0, C = 1
- ¿Qué observa?

# Resolución de Problemas y Algoritmos

¿TIENEN EL MISMO EFECTO?

<pre>IF ( A &gt; 10 ) THEN   write(1); IF ( B = 0 ) THEN   write(2);</pre>	<pre>IF ( A &gt; 10 ) THEN   BEGIN     write(1)     IF ( B = 0 ) THEN       write(2);   END;</pre>
--	--

¿Por qué con A=1 y B=0 tienen diferente efecto?

- En el recuadro de la izquierda, hay una secuencia de dos sentencias condicionales (**if-then**) que son independientes entre sí (observe que están separadas por un “;”).
- En cambio, en el recuadro de la derecha, como hay un **begin-end**, el segundo **if-then** depende del primero ya que está “anidado” dentro del primero: se ejecutará solamente cuando el valor de A sea mayor a 10.

Resolución de Problemas y Algoritmos - 2016

¿TIENEN EL MISMO EFECTO?

- Realice una traza con A=5 y B=6

<pre>IF A = B THEN   write(1); IF A = 5 THEN   write('A es 5') ELSE   write('DISTINTOS')</pre>	<pre>IF A = B THEN   BEGIN     IF A = 5 THEN       WRITE('A es 5')     END   ELSE     WRITE('DISTINTOS');</pre>
--	---

- El “ELSE” siempre se corresponde con el “IF-THEN” anterior más cercano que no tenga ELSE. Por lo tanto, en el ejemplo de la izquierda el “ELSE” se corresponde con el “IF A=5 THEN”.
- Sin embargo, utilizando “BEGIN - END” puedo forzar y hacer que se corresponda con otro IF-THEN. Esto ocurre en el ejemplo del bloque de la derecha donde el “ELSE” se corresponde con el “IF A=B THEN”.

Resolución de Problemas y Algoritmos - 2016

NUEVO PROBLEMA PROPUESTO

**Problema:** Escriba un programa en Pascal que lea un carácter (CHAR) y diga si se trata de una letra mayúscula, minúscula, o si se trata de otro símbolo.

- Por ejemplo:
  - 'G', es una mayúscula
  - 'g', es una minúscula
  - '3', es otro símbolo
  - '\$', es otro símbolo
- Siguiendo la metodología propuesta, escriba un algoritmo y un programa en Pascal que resuelva el problema.
- Indique cuales son los casos de prueba que usó.

Resolución de Problemas y Algoritmos - 2016

PROBLEMA PROPUESTO: DIAS DE UN AÑO

**Problema:** Escribir un programa que dado un año, indique cuantos días tiene.

- Solución:**
  - En general son 365 días pero algunos años febrero tiene 29 días (años bisiestos) y son 366 ¿cuáles son años bisiestos? ¿por qué pasa esto?
  - Un año “astronómico” tiene 365 días 5 h 48 m 45,25 s
  - Un año calendario tiene 365 o 366 días (año bisiesto)

vea [http://es.wikipedia.org/wiki/Año\\_bisiesto](http://es.wikipedia.org/wiki/Año_bisiesto)

**Definición:** un año es bisiesto si es múltiplo de 4 y no es múltiplo de 100 o es múltiplo de 400.

**Ejemplo:** 2016, 2008 y 2000 son bisiestos, 2009, 2010 y 1900 no lo son.

Resolución de Problemas y Algoritmos - 2016

PROBLEMA PROPUESTO: DIAS DE UN AÑO

**Definición:** un año es bisiesto si es múltiplo de 4 y no es múltiplo de 100 o es múltiplo de 400.

**Con una expresión:** (Note que si es mult. de 400 también es de 100 y de 4)  
 VAR año: integer; bisiesto: boolean;  
 bisiesto := (año mod 4=0) and not (año mod 100=0) or (año mod 400=0);

<pre>PROGRAM dias_año; VAR año: integer;     bisiesto: boolean; BEGIN   write('ingrese año: ');   readln(año);   bisiesto := (año mod 4=0) and not(año mod 100=0) or (año mod 400=0);   IF bisiesto THEN     write('tiene 366 días')   ELSE     write('tiene 365 días'); END.</pre>	<b>Casos de prueba:</b> 4 100 400 1900 2000 2014 2016
---	--

Resolución de Problemas y Algoritmos - 2016

PROBLEMA PROPUESTO: DIAS DE UN AÑO

**Definición:** un año es bisiesto si es múltiplo de 4 y no es múltiplo de 100 o es múltiplo de 400.

**Con una expresión:** (Note que si es mult. de 400 también es de 100 y de 4)  
 VAR año: integer; bisiesto: boolean;  
 bisiesto := (año mod 4=0) and not (año mod 100=0) or (año mod 400=0);

- Con condicionales:

<pre>IF año mod 4 = 0 THEN   IF año mod 100 = 0 THEN     IF año mod 400 = 0 THEN       bisiesto := true     ELSE       bisiesto := false   ELSE     bisiesto := true ELSE   bisiesto := false</pre>	<b>Casos de prueba:</b> 4 100 400 1900 2000 2014 2016
---	--

Resolución de Problemas y Algoritmos - 2016