# Resolución de Problemas y Algoritmos

**Clase 4**
**Estructura de control condicional.**

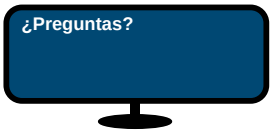

John von Neumann

**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 de las clases anteriores

- **Algoritmo. Primitiva. Traza.**
- **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.**




```
PROGRAM TextoPregunta;
CONST signo = CHR(168);
BEGIN
WRITE(signo,'Preguntas?');
readln {espera ENTER}
END.
```



## Intercambiar los valores de las variables

En una asignación: variable := expresión
1) primero se evalúa la expresión de derecha y se obtiene un valor,
2) luego modifica el valor de 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 valido y aún así tiene un error.**

**¿Qué casos de prueba usaría?**



## Intercambiar los valores de las variables

En una asignación: variable := expresión
1) primero se evalúa la expresión de derecha y se obtiene un valor,
2) luego modifica el valor de 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 enter
 read(a,b);
 a:= b;
 b:= a;
 writeln(a,' ',b);
END.
```

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

**Traza** de valores en memoria

| a | b |
|---|---|
| ? | ? |
| 1 | ? |
| 1 | 5 |
| 5 | 5 |
| 5 | 5 |



## Contenedores de elementos de cierto tipo

- **Las variables pueden pensarse como recipientes que pueden contener un elemento de cierto tipo .**
- **Por ejemplo un vaso es un recipiente pensado para contener elementos de tipo líquido (lo mismo una taza o una botella).**
- **Si tengo un vaso con una gaseosa, para poder tener ese mismo vaso con chocolatada, debo sacar la gaseosa y luego 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
1) primero se evalúa la expresión de derecha y se obtiene un valor,
2) luego se modifica el valor de variable, perdiéndose el 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.**

**¿Qué casos de prueba usaría?**

## Intercambiar los valores de las variables

En una asignación: variable := expresión
1) primero se evalúa la expresión de derecha y se obtiene un valor,
2) luego se modifica el valor de variable, perdiéndose el 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 enter
      read(a,b);
      aux:= a;
      a:= b;
      b:= aux;
      write(a,' ', b);
END.
```

**Traza** de valores en memoria

| a | b | aux |
|---|---|---|
| ? | ? | ? |
| 1 | ? | ? |
| 1 | 5 | ? |
| 1 | 5 | 1 |
| 5 | 5 | 1 |
| 1 | 5 | 1 |



## Metodología 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 …**

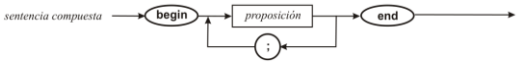


## Sentencias en Pascal

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

**Sentencia** (*statement*)
- **simple**: **a:=1**
- **compuesta**:

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





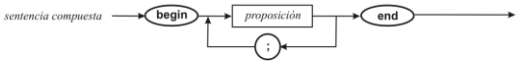

## Sentencia compuesta en Pascal

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.



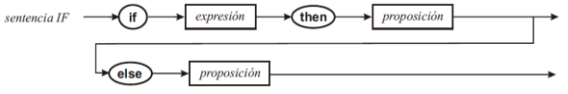



## Estructura de control condicional (IF-THEN-ELSE)

```
IF   expresión lógica (de tipo boolean)
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.






## Estructura de control condicional (IF-THEN-ELSE)

```
IF   expresión lógica (de tipo boolean)
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
```

**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**".

## 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**:

Escribamos el código fuente en Pascal en el pizarrón

```
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

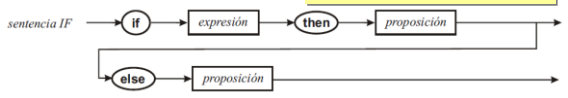
**Observe: no lleva ";" antes del ELSE**



## 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 caracter 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 caracter ASCII entre 'A' y 'Z' es una letra mayúscula, un caracter entre 'a' y 'z' es una letra minúscula.**

**Algoritmo:**

- **leer el caracter**
- **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 legra (ejemplos: 'G', 'g', '3' , '$')**



## No cantemos victoria antes de gloria…

```
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.');
END.
```

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.

## Un programa correcto

```
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 traza para el caso de prueba: 'G'



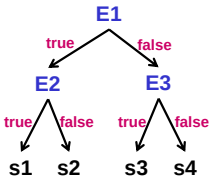
## Condicionales "anidados"

```
IF <exp. boolean>
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"



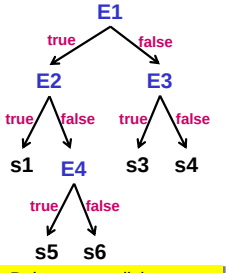

¿Bajo que condiciones se ejecutará la sentencia s3?

¿El resultado de E2 afecta a la ejecución de s3?

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



## El límite está en su imaginación




¿Bajo que condiciones se ejecutará la sentencia s6?

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



## ¿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

1) A = 20, B = 0, C = 100
2) A = 1, B = 0, C = 100
3) A = 1, B = 0, C = 1

¿Qué observa?

## ¿Tienen el mismo efecto?

Realice una traza con (i) A=20 y B=0; luego con (ii) A=1 y B=0.

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

```
IF( A > 10 )
  THEN BEGIN
         write(1);
         IF ( B = 0 )
           THEN write(2);
       END;
```

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

- En el recuadro de la izquierda (celeste) hay una secuencia de dos sentencias condicionales (**if-then**) que son independientes entre si (observe que están separadas por un ";" ).
- En cambio, en el recuadro de la derecha (amarillo), 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.



## ¿Tienen el mismo efecto?

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

```
IF A = B
THEN
    IF A = 5
  THEN WRITE('A es 5 ')
ELSE WRITE('DISTINTOS');
```

```
IF A = B
THEN
  BEGIN
    IF A = 5
    THEN WRITE('A es 5 ')
  END
ELSE WRITE('DISTINTOS');
```

- 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**".



## Nuevo problema propuesto

**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.**



## Problema propuesto: días 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*

**Definición**: un año es **bisiesto** si es múltiplo de 4 y no es múltiplo de 100 o es múltiplo de 400.
Ej. 2016, 2008 y 2000 son bisiestos, 2009, 2010 y 1900 no lo son.



## Problema propuesto: días 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**: (observe que si es mult. de 400 también es de 100 y de 4)

```
VAR anio:integer; bisiesto: boolean;
bisiesto := (anio mod 4=0) and not (anio mod 100=0) or (anio mod 400=0);
```

```
program dias_anio;
var anio: integer; bisiesto: boolean;
begin
write('ingrese año: '); readln (anio);
bisiesto := (anio mod 4=0) and
   not (anio mod 100=0) or (anio mod 400=0);
IF bisiesto
  THEN write('tiene 366 días')
  ELSE write('tiene 365 días');
end.
```

**Casos de prueba:**
4
100
400
1900
2000
2014
2015



## Calcular cuando un año es bisiesto

**Definición**: un año es **bisiesto** si es múltiplo de 4 y no es múltiplo de 100 o es múltiplo de 400.
Ej. 2004, 2008 y 2000 son bisiestos, 2009, 2010 y 1900 no lo son.

**Con una expresión**: (observe que si es mult. de 400 también es de 100 y de 4)

```
VAR anio:integer; bisiesto: boolean;
bisiesto := (anio mod 4=0) and not (anio mod 100=0) or (anio mod 400=0);
```

**Con condicionales:**

```
IF anio mod 4 = 0
  THEN IF anio mod 100 = 0
         THEN IF anio mod 400 = 0
                THEN bisiesto := true
                ELSE bisiesto := false
         ELSE bisiesto := true
  ELSE bisiesto := false
```

**Casos de prueba:**
4
100
400
1900
2000
2014
2015

## Información adicional



## Primeras computadoras: ENIAC & EDVAC

**ENIAC (Electronic Numerical Integrator And Computer) fue diseñada en 1943 por el físico John Mauchly, y el ingeniero electrónico John Eckert.**

**La programación era por hardware y reprogramarla costaba días. Esta tarea estaba a cargo de 6 mujeres con grandes habilidades matemáticas y lógicas, que iban inventando la programación a medida que la realizaban: Betty Snyder Holberton, Jean Jennings Bartik, Kathleen McNulty Mauchly Antonelli, Marlyn Wescoff Meltzer, Ruth Lichterman Teitelbaum y Frances Bilas Spence**

**Computadora ENIAC (1946) (http://es.wikipedia.org/wiki/ENIAC)**



## Primeras computadoras: ENIAC & EDVAC

**Eckert y Mauchly conscientes de las limitaciones de ENIAC empezaron con un nuevo diseño para una nueva computadora (EDVAC). Eckert propuso una memoria de mercurio para guardar tanto el programa como datos.**
**El matemático Goldstine también era parte del proyecto EDVAC, y en una charla casual (en una estación de tren) entusiasmó al matemático húngaro John von Neumann a formar parte del proyecto.**
**La arquitectura de EDVAC incorporó el concepto de programa almacenado en memoria y codificación en binario en lugar de decimal.**

**Computadora EDVAC (1945) (http://es.wikipedia.org/wiki/EDVAC)**



## El origen del término arquitectura von Neumann

**John von Neumann fue una persona muy inteligente y activa que trabajó en numerosas áreas: matemática, lógica, programación, energía atómica, física y computación teórica. [ver más]**

**El término arquitectura de von Neumann surgió a partir del primer reporte del diseño de la EDVAC que fue escrito (a mano) en 1945 por John von Neumann mientras regresaba en a su casa.**

**Luego envió el reporte por correo a Goldstine quien lo tipió y distribuyó apresuradamente entre sus colegas por todo el mundo, dejando como autoría solo el nombre de Von Neumann (sin incluir a Mauchly y Eckert).**




**John von Neumann**