0.1 SISTEMA BINARIO DE NUMERACIÓN

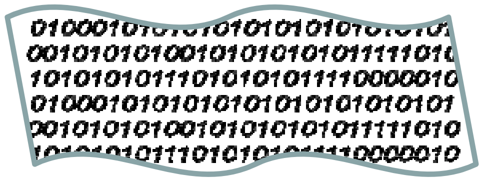
Mercè Rullán

Universidad Autónoma de Barcelona

1. Representación de la información en las computadoras

Un ordenador o computador es una máquina que recibe y procesa datos para convertirlos en información útil.

Los datos con los que trabaja pueden ser números, caracteres, audio, video, etc. Todo se reduce a procesar números que están codificados en forma de cadenas de **ceros y unos**.



La **tecnología** en la que se basan las computadoras (y en definitiva sus componentes físicos) es la responsable de que toda la información con la que trabaja un computador se codifique en un sistema de numeración denominado **sistema binario** en el que sólo utilizamos dos símbolos o estados: **0, 1**

Universitat Autònoma de Barcelona

2. Sistemas de numeración

Existen muchos sistemas de numeración pero recordaremos los siguientes:

- **sistema decimal**
- **sistema binario**
- **sistema hexadecimal**

y trabajaremos las formas de conversión de un sistema de numeración a otro.

2.1 Sistema decimal

- Es un sistema de numeración formado por **10 símbolos** o dígitos diferentes:

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- Es un sistema de numeración **posicional** (el orden de los dígitos es importante porque se asocia un peso a cada posición)

Ejemplo: 653

	6	**5**	**3**
(pesos)	10^2	10^1	10^0

$$653 = 6 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0$$

6 centenas, 5 decenas, 3 unidades

2.2 Sistema binario

- Es un sistema de numeración formado por **dos símbolos** o dígitos diferentes: **0, 1**
- También es un sistema de numeración **posicional**

Ejemplo:

	1	**1**	**0**	**1**
(pesos)	2^3	2^2	2^1	2^0

- Para calcular el valor en decimal de un número binario sólo hay que ir sumando los pesos en los que exista un 1:

$$(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13$$

Ejercicio

Calcular el valor decimal del siguiente número binario: $(101001)_2$

UAB Universitat Autònoma de Barcelona

Ejercicio (solución)

Calcular el valor decimal del siguiente número binario: $(101001)_2$

	1	0	1	0	0	1
pesos	2^5	2^4	2^3	2^2	2^1	2^0

$$(101001)_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 32 + 8 + 1 = 41$$

2.2 Sistema binario puro: rango de representación

- Binario puro corresponde a números que sólo pueden ser positivos.
- Si utilizamos n bits, podemos representar un total de 2^n valores distintos.
- Su rango de representación va desde el número 0 hasta el número (2^n - 1).

EJEMPLO:

n = 4 bits

16 combinaciones diferentes

desde el 0 al 15 = 2^4-1

Binario	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binario	Decimal
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

2.2 Sistema binario puro: rango de representación

- ✓ Si n = 3 , representamos de 0 a 7; en total $2^3 = 8$ valores diferentes
- ✓ Si n = 4 , representamos de 0 a 15; en total $2^4 = 16$ valores diferentes
- ✓ Si n = 5 , representamos de 0 a 31; en total $2^5 = 32$ valores diferentes
- ✓ Si n = 6 , representamos de 0 a 63; en total $2^6 = 64$ valores diferentes...

Ejemplo: ¿Cuántos bits necesitamos para representar en binario el número 48?

$$31 \leq 48 \leq 63$$

$\cancel{n=5}$ n=6

Para representar en binario el número 48_{10} necesitamos 6 bits: 110000

2.3 Sistema hexadecimal

- Es un sistema de numeración formado por 16 dígitos diferentes:

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- También es un sistema de numeración **posicional**

  3 A 9 F

  (pesos) 16^3 16^2 16^1 16^0

- Para calcular el valor decimal de un número hexadecimal sólo hay que ir sumando los pesos multiplicados por el valor respectivo que tengan en cada posición:

$(3A9F)_{16} = 3 \cdot 16^3 + 10 \cdot 16^2 + 9 \cdot 16^1 + 15 \cdot 16^0 = 15007_{10}$

BINARIO				HEXA
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

3. Cambios de base (HEXADECIMAL-BINARIO) (BINARIO-HEXADECIMAL)

- En el sistema HEXADECIMAL, cada dígito se puede representar en binario con 4 bits:

hexadecimal	**3**	**A**	**9**
binario	**0011**	**1010**	**1001**

- Para convertir un número de binario a HEXADECIMAL debemos agrupar los bits de 4 en 4 empezando por la derecha y obtener el valor resultante en hexadecimal:

binario	**100**	**1011**	**1010**	**0101**
hexadecimal	**4**	**B**	**A**	**5**

UAB
Universitat Autònoma de Barcelona

Ejercicio

BINARIO				HEXA
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Calcula la representación en hexadecimal:

$110110101001100_2 = ????_{16}$

Calcula la representación en binario:

$5F2B_{16} = ????_2$

Ejercicio (solución)

UAB
Universitat Autònoma de Barcelona

BINARIO				HEXA
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Calcula la representación en hexadecimal:

$$110110101001100_2 = 6D4C_{16}$$

Calcula la representación en binario:

$$5F2B_{16} = 0101111100101011_2$$

0101 1111 0010 1011

5. Cambios de base (DECIMAL- BINARIO)

- Dividimos por 2 el número en base decimal y los sucesivos cocientes se dividen de nuevo por 2, hasta obtener un cociente de valor 1.
- El número en base 2 está formado por el último cociente y los sucesivos restos, siendo el último cociente el bit más significativo.

Ejemplo: $18_{(10} = ?$

$$\boxed{}_{(10} = \boxed{} \cdot 2 + \boxed{}$$

$$18 = 9 \cdot 2 + 0$$
$$9 = 4 \cdot 2 + 1$$
$$4 = 2 \cdot 2 + 0$$
$$2 = 1 \cdot 2 + 0$$

$$18_{(10} = 10010_{(2}$$

Ejercicio

$43_{(10}$ = ¿número en base binaria?

UAB Universitat Autònoma de Barcelona

Ejercicio (solución)

$\mathbf{43_{(10}}$ **=** ¿número en base binaria?

$$43 = 21 \cdot 2 + 1$$

$$21 = 10 \cdot 2 + 1$$

$$10 = 5 \cdot 2 + 0$$

$$5 = 2 \cdot 2 + 1$$

$$2 = 1 \cdot 2 + 0$$

$\mathbf{43_{(10} = 101011}$

6. Suma y resta de números binarios

Combinaciones al sumar dos bits:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = **1 0** (se escribe 0 y hay acarreo de 1 que se suma a la siguiente etapa)

Combinaciones al restar dos bits:

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = **1 1** (se escribe 1 y hay acarreo de 1 que se suma al **sustraendo** de la siguiente etapa)

```
        1 1 1    → acarreos parciales que se suman
  1 0 1 0 0 1 0 1  = A
+   1 0 0 0 1 1 1  = B
-------------------
  1 1 1 0 1 1 0 0  = C
```

```
  1 0 1 0 0 1 0 1  = A
  1     1  1  1  1       → acarreos que se suman al sustraendo
-   1 0 0 0 1 1 1  = B
-------------------
  0 1 0 1 1 1 1 0  = C
```

Ejercicio

$$
\begin{array}{r}
10011011 = A \\
+ \quad 1010011 = B \\
\hline
\end{array}
$$

$$
\begin{array}{r}
10011001 = A \\
- \quad 1010011 = B \\
\hline
\end{array}
$$

Ejercicio (solución)

```
  0 0 1 0 0 1 1        → acarreo que se suma
  1 0 0 1 1 0 1 1 = A
+   1 0 1 0 0 1 1 = B
-------------------
  1 1 1 0 1 1 1 0
```

```
  1 0 0 1 1 0 0 1 = A
  1       1 1          → acarreo que se suma al sustraendo
-   1 0 1 0 0 1 1 = B
-------------------
  0 1 0 0 0 1 1 0
```

RESUMEN

- Representación de la información en las computadoras
- Sistemas de numeración (decimal, binario, hexadecimal)
- Sistema binario puro y rango de representación
- Cambios de base entre diferentes sistemas de numeración
- Suma y resta de números binarios

0.2 REPRESENTACIÓN DE ALGORITMOS EN PSEUDOCÓDIGO

Mercè Rullán

Universidad Autónoma de Barcelona

1. Representación de algoritmos: pseudocódigo

Algoritmo

- Procedimiento no ambiguo que resuelve un problema, de forma que indique una secuencia de operaciones a realizar para conseguir el resultado deseado.
- El algoritmo es independiente del lenguaje de programación que se utilizará para su implementación final.
- Existen diferentes formas de representación de los algoritmos: el lenguaje natural, **pseudocódigo**, diagramas de flujo y lenguajes de programación.

Pseudocódigo

- Es una forma de descripción **informal** de un algoritmo con una sintaxis cercana a los lenguajes de programación.
- Utiliza una mezcla de frases en lenguaje común, instrucciones de programación y palabras clave que definen las estructuras básicas.

2. Acciones y estructuras de control (esquema)

- ASIGNACIONES
- OPERADORES
  - Comparación
  - Lógicos
  - Aritméticos
- ESTRUCTURAS DE CONTROL
  - Selección (decisión)
    - Simple, Doble
    - Múltiple (Case)
  - Iteración (ciclos)
    - While
    - For
- PROCEDIMIENTOS

Universitat Autònoma de Barcelona

3. Asignaciones

La acción denominada **asignación** <= permite almacenar un valor en una variable.

El valor puede ser una constante o el resultado de la evaluación de una expresión.

Primero se evalúa la expresión de la derecha y luego se asigna el resultado a la variable de la izquierda.

<variable> <= <expresión>;

variable <= expresión;

Ejemplo

X <= 3; Y <= 2; Z <= 1;

Y <= 2X+Y+Z;

4. Operadores

Nos permiten realizar operaciones de diferentes tipos:

COMPARACIÓN

Operador	Significado
<	Menor que
>	Mayor que
=	Igual que
≤ (1)	Menor o igual que
≥ (2)	Mayor o igual que
≠(3)	Diferente

ARITMÉTICOS

Operador	Significado
+	Suma
-	Resta
*	Producto
/	División

LÓGICOS

Operador	Significado
or	Suma lógica
and	Producto lógico
not	Negación

(1) En algunos ejemplos también utilizamos el símbolo <= para la comparación “menor o igual que”
(2) En algunos ejemplos también utilizamos el símbolo >= para la comparación “mayor o igual que”
(3) En algunos ejemplos también utilizamos el símbolo /= para la comparación “diferente”

Ejercicio

Si tenemos la siguiente descripción en pseudocódigo:

1. X <= 2;
2. Y <= 3;
3. Z <= 4;
4. X <= X*Y;
5. Z <= X+Y;
6. Y <= X+Y+Z;

¿Qué valor tendrá la variable X después la línea 4?

¿Qué valor tendrá la variable Z después la línea 5?

¿Qué valor tendrá la variable Y después la línea 6?

Ejercicio (solución)

Si tenemos la siguiente descripción en pseudocódigo:

```
1.  X <= 2;
2.  Y <= 3;
3.  Z <= 4;
4.  X <= X*Y;
5.  Z <= X+Y;
6.  Y <= X+Y+Z;
```

¿Qué valor tendrá la variable X después la línea 4?

La variable X valdrá 6

¿Qué valor tendrá la variable Z después la línea 5?

La variable Z valdrá 9

¿Qué valor tendrá la variable Y después la línea 6?

La variable Y valdrá 18

5. Estructuras de control

La acciones que podemos describir mediante pseudocódigo se organizan en bloques de acciones que siguen unas estructuras básicas:

- ESTRUCTURAS DE **SELECCIÓN** (decisión)
  - · Simple
  - · Doble
  - · Múltiple (Case)

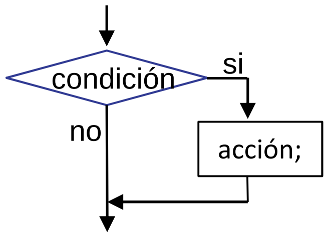
- ESTRUCTURAS DE **ITERACIÓN** (ciclos)
  - · While
  - · For

5.1 Estructuras de selección (decisión)

Las estructuras de selección nos permiten tomar decisiones condicionadas a la evaluación de una expresión (normalmente una condición lógica).

Simple

If *condición* **then** *acción/es*;
end if;

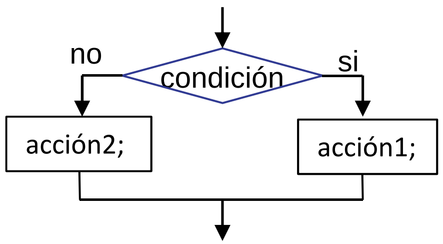


Ejemplo

If *(X es par) and (X ≥ 8)* **then** *acción/es*;
end if;

Doble

If *condición* **then** *acción1/es*;
else *acción2/es*;
end if;

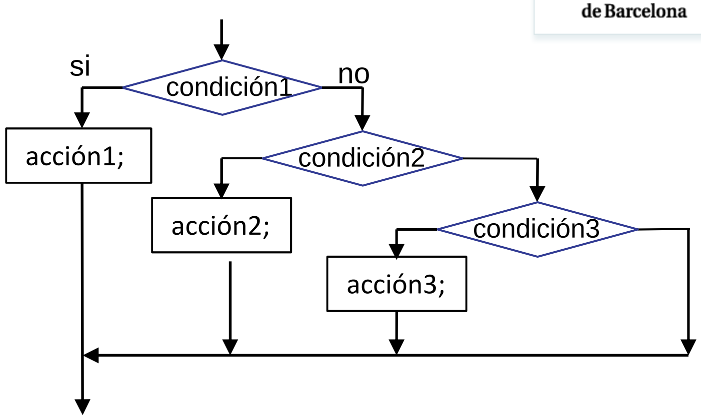


If *(X es par) and (X ≥ 8)* **then** *acción1/es*;
else *acción2/es*;
end if;

5.1 Estructuras de selección (decisión)

Múltiple

```
If  condición1 then acción1;
            elsif condición2  then acción2;
                        elsif condición3  then acción3;
                        end if;
            end if;
end if;
```



Ejemplo:

```
If  A=1 then Z <= X + Y;
          elsif A=2 then Z <= X - Y;
                    elsif  A >=3  then Z <= X * Y;
                    end if;
          end if;
end if;
```

Ejercicio

Si queremos realizar en pseudocódigo la operación y = X/2 , redondeando por defecto:

Análisis de los diferentes casos:

- Cuando X es par, el resultado de la división será exacto (tanto si X es positivo como negativo)

  x = 8 y = 8/2 = 4
  x = -8 y = -8/2 = -4

- Cuando X es impar, el redondeo deberá ser calculado de forma diferente:

  - si es negativo será (X+1)/2

  - si es positivo será (X-1)/2

  x = -7 y = (-7+1)/2= -3
  x = 7 y = (7-1)/2= 3

ESCRIBID el **pseudocódigo** de la operación y = X/2 , redondeando por defecto según el análisis de los diferentes casos que hemos planteado.

Ejercicio (resuelto)

Si queremos realizar en pseudocódigo la operación y = X/2 , redondeando por defecto:
Análisis de los diferentes casos:

- Cuando X es par, el resultado de la división será exacto (tanto si X es positivo como negativo)
  x = 8 y = 8/2 = 4
  x = -8 y = -8/2 = -4
- Cuando X es impar, el redondeo deberá ser calculado de forma diferente:
  - si es negativo será (X+1)/2 x = -7 y = (-7+1)/2= -3
  - si es positivo será (X-1)/2 x = 7 y = (7-1)/2= 3

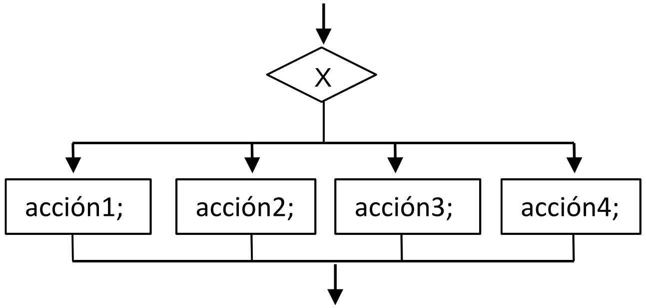
```
If (X es par) then y<= X/2;
              elsif (X<0) then y <= (X+1)/2;
               else y <= (X-1)/2;
             end if;
end if;
```

UAB
Universitat Autònoma de Barcelona

5.1 Estructuras de selección (decisión)

Case

```
case X is
    when “valor1" => acción1;
    when “valor2" => acción2;
    when “valor3" => acción3;
    when “valor4" => acción4;
end case;
```



Ejemplo: X es un dígito decimal y queremos obtener su representación en binario en la variable y

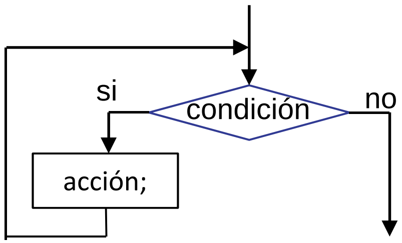
```
case x is
    when “0" =>  y <= 0000;
    when “1" =>  y <= 0001;
    ........
    when “9" => y <= 1001;
end case;
```

5.2 Estructuras de iteración (ciclos)

Las estructuras de iteración nos permiten repetir acciones condicionadas a la evaluación de una expresión.

While

While *condición* **loop**
    acción/es;
end loop;



For

for variable **in** min **to** max **loop**
    accion/es;
end loop;

UAB Universitat Autònoma de Barcelona

Ejercicio

Dados dos vectores de 8 posiciones:

a(0), a(1), ……a(7)
x(0), x(1), ……x(7)

ESCRIBID el **pseudocódigo** del cálculo:

$y = a(0) \cdot x(0) + a(1) \cdot x(1) + a(2) \cdot x(2) + \ldots\ldots\ldots + a(7) \cdot x(7)$

Ejercicio (resuelto)

Dados dos vectores de 8 posiciones:

a(0), a(1), ……a(7)
x(0), x(1), ……x(7)

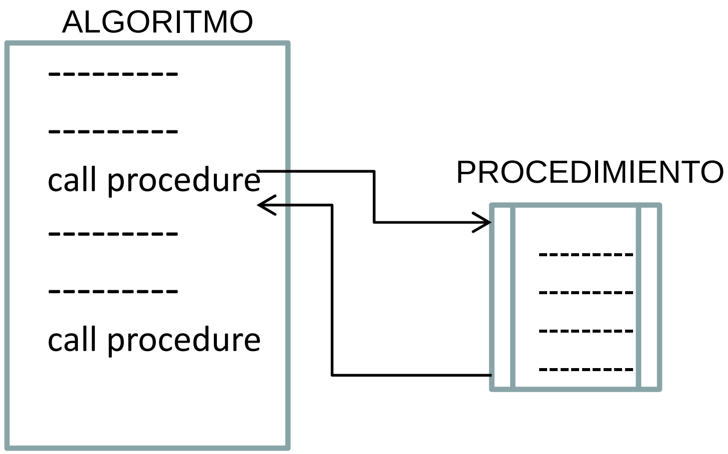
ESCRIBID el pseudocódigo del cálculo:

y = a(0) · x(0) + a(1) · x(1) + a(2) · x(2) + ……….+ a(7) · x(7)

```
acc <= 0;
for i in 0 to 7 loop
    acc <= acc+ a(i) · x(i);
end loop;
```

6. Procedimientos (subrutinas)

Procedure call *nombre (parámetros)*;



- conjunto de acciones a realizar que resuelven una tarea concreta
- a cada procedimiento se le asigna un **nombre** por el que puede ser llamado desde otra parte del programa principal
- al llamar a un procedimiento sólo debemos dar valor a las variables que intervienen en dicho procedimiento (**parámetros**) y dichos valores se asignarán al realizar la llamada al mismo
- facilita el diseño, estructura y comprensión de los algoritmos y nos evita repetir sentencias en el algoritmo principal

UAB
Universitat Autònoma de Barcelona

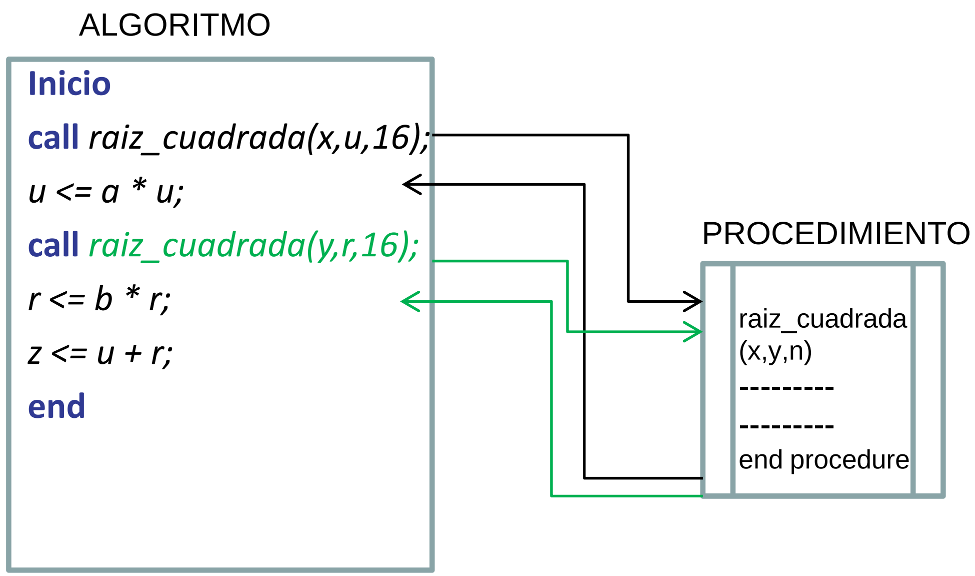
6. Procedimientos (subrutinas)

Ejemplo: **calcular $z = a \cdot \sqrt{x} + b \cdot \sqrt{y}$**

Procedure *raiz_cuadrada(x, y, n) is*

----
end procedure

Algoritmo
call *raiz_cuadrada(x, u, 16);*
*u <= a * u;*
call *raiz_cuadrada(y, r, 16);*
*r <= b * r;*
z <= u + r;



RESUMEN

- Representación de algoritmos: pseudocódigo
- Acciones y estructuras de control:
  - Asignaciones y operadores
  - Estructuras de control
    - Selección (decisión)
      - Simple, Doble
      - Múltiple (Case)
    - Iteración (ciclos)
      - While
      - For
- Procedimientos