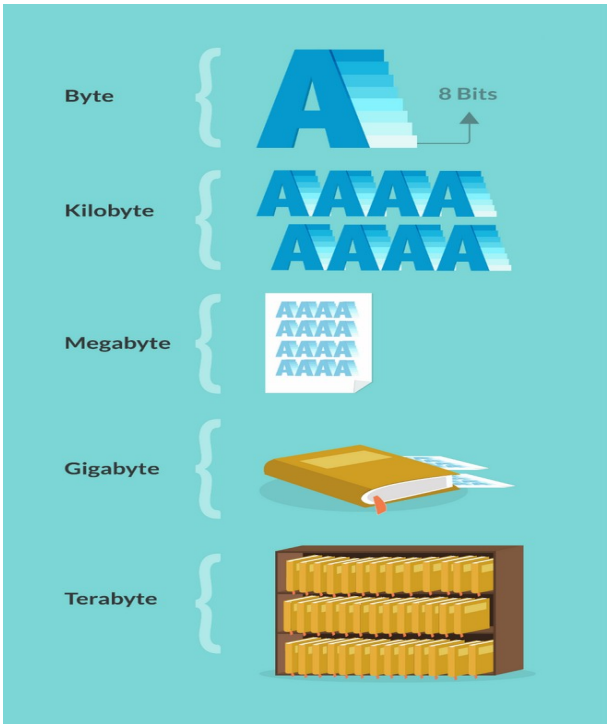
# Sumario

# 1. Medidas de almacenamiento de información.

Medidas de almacenamiento de información. Las unidades de medida más usadas son el Bit, Byte, Kilobyte, Megabyte, Gigabyte y Terabyte.

Por ejemplo puedes imaginar esto:

Tienes un libro muy grande, y una sola letra de ese libro representa un Byte. Esta letra está compuesta por (8) ocho partes y cada una de esas partes se llama Bit.




Si juntas varias letras (bytes) formarías palabras, y con las palabras un párrafo, que aquí contaría como un Kilobyte.

Con varios párrafos (Kilobytes) podrías conformar algunas páginas del libro, lo que podría ser un Megabyte.

Y uniendo todas las páginas (megabytes), tendrías el libro completo, que puedes imaginar que es Gigabyte.

Si unes ese libro a muchos otros libros (Gygabytes), tendrías una gran biblioteca que, en este caso, equivaldría a un Terabyte.

Aunque la capacidad de almacenamiento de cada una de las unidades de medida no es exactamente igual al ejemplo que te acabamos de dar, ya tienes una idea de cómo funcionan y se organizan. Equivalencias reales:

**Bit:** Es la unidad mínima de información empleada en informática.

**Byte (B):**

Equivale a 8 bits. C**on dos bytes guardas o procesas una letra.**

**Kilobyte (kB):** 1024 bytes forman un Kilobyte.

**Megabyte (MB):** Equivale a 1024 Kilobytes.

**Gigabyte (GB):** Es igual a 1024 Megabytes. Es la unidad de medida que se suele usar para determinar la capacidad de almacenamiento de las USB.

**Terabyte (TB):** Lo componen 1024 Gigabytes. Muchas veces esta medida determina la capacidad de almacenamiento de los discos duros. ¡Imagina la cantidad de archivos que podrías guardar!

# 2. Sistemas de numeración

## a) El sistema decimal.

Compuesto por los símbolos 0 al 9 (base 10), es el sistema numérico que utilizamos a diario.

## b) El sistema binario.

Es un sistema de numeración en el que los números se representan utilizando solamente dos cifras: cero (0) y uno (1). Se utilizan en las computadoras, debido a que estas trabajan internamente con dos niveles de voltaje, por lo cual es su sistema de numeración natural.

## c) El sistema octal

Su base es 8, esto significa que existen ocho símbolos (dígitos octales): 0, 1, 2, 3,4, 5, 6, 7. Se usa este sistema de numeración al tener un método de conversión casi inmediato a binario, reduce la cantidad de dígitos para representar una cantidad, con lo que se reduce los errores.

Cada dígito octal representa tres bits y tres bits pueden representarse mediante un dígito octal. La tabla muestra la relación entre un patrón de bits y un dígito octal.

| Octal | Decimal | Binario |
|---|---|---|
| 0 | 0 | 000 |
| 1 | 1 | 001 |
| 2 | 2 | 010 |
| 3 | 3 | 011 |
| 4 | 4 | 100 |
| 5 | 5 | 101 |
| 6 | 6 | 110 |
| 7 | 7 | 111 |

## d) El sistema hexadecimal.

El binario se usa para representar datos cuando éstos se almacenan dentro de un ordenador. Sin embargo, para la gente es difícil manipular los patrones de bits. Escribir una serie de números 0 y 1 es tedioso y propenso al error. La notación hexadecimal ayuda a las personas con estos dos problemas.

| DECIMAL | BINARIO | HEXADECIMAL |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

La notación hexadecimal tiene base 16, es decir hay 16 símbolos (dígitos hexadecimales): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. La importancia de la notación hexadecimal se hace evidente cuando se convierte binario a notación hexadecimal.

Cada dígito hexadecimal puede representar cuatro bits y cuatro bits pueden representarse mediante un dígito hexadecimal.

La notación hexadecimal se escribe en dos formas. Se añade una x minúscula (o mayúscula) antes de los dígitos para mostrar que la representación está en hexadecimal. Por ejemplo, **x**A34 representa un valor hexadecimal en esta convención.

Otra forma, es indicar la base del número (16) como el subíndice después de cada número. Por ejemplo, $A34_{16}$ muestra el mismo valor en la segunda convención.

# 3. Conversiones de sistemas de numeración

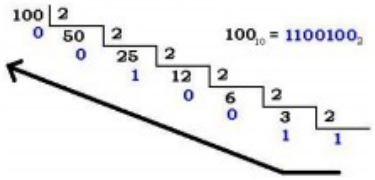
## a) Decimal a binario

Se divide el número del sistema decimal entre 2, cuyo resultado entero se vuelve a dividir entre 2, y asísucesivamente hasta que el dividendo sea menor que el divisor, 2. Es decir, cuando el número a dividir sea 1 finaliza la división.

A continuación se ordena desde el último cociente hasta el primer resto, simplemente se colocan en orden inverso

**Ejemplo**

Transformar el número decimal 100 en binario.




## b) Binario a decimal

Para realizar la conversión de binario a decimal, realice lo siguiente:

1. Empezamos por el lado derecho del número en binario. Multiplique cada dígito por 2 elevado a la potencia consecutiva (comenzando por la potencia $2^0$).

2. Después se hacen las multiplicaciones y tras sumarlas el resultante será el equivalente al sistema decimal.

**Ejemplos:**

***Nota:*** Los números de la parte superior del número binario indican la potencia a la que hay que elevar el número 2, solo tiene carácter informativo en el ejemplo.

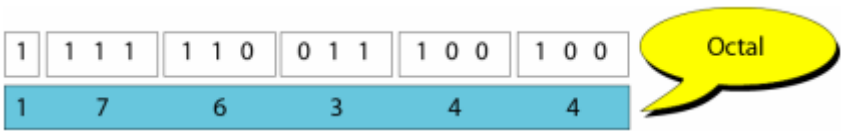
$$\overset{5\,4\,3\,2\,1\,0}{1\,1\,0\,1\,0\,1_2} = 1\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 32 + 16 + 0 + 4 + 0 + 1 = 53$$

$$\overset{7\,6\,5\,4\,3\,2\,1\,0}{1\,0\,0\,1\,0\,1\,1\,1_2} = 1\cdot 2^7 + 0\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 128 + 0 + 0 + 16 + 0 + 4 + 2 + 1 = 151$$

$$\overset{5\,4\,3\,2\,1\,0}{1\,1\,0\,1\,1\,1_2} = 1\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 32 + 16 + 0 + 4 + 2 + 1 = 55$$

## c) Binario a octal y octal a binario.

La conversión de un patrón de bits a notación octal se realiza mediante la organización del patrón en grupos de tres y la determinación del valor octal de cada grupo de tres bits. Para la conversión de octal a patrón de bits, se convierte cada dígito octal a su equivalente de tres bits




**Ejemplo 1**

Muestre el equivalente octal del patrón de bits 101110010.

**Solución**

Cada grupo de tres bits se traduce a un dígito octal. El equivalente es **o**562 o $562_8$.

**Ejemplo 2**

Muestre el equivalente octal del patrón de bits 1100010.

**Solución**

El patrón de bits se divide en grupos de tres bits (a partir de la derecha). En este caso, se añaden dos 0 más a la izquierda para hacer el número total de bits divisible entre 3. Así que usted tiene 001100010, lo cual se traduce a **o**l42 o $142_8$.

**Ejemplo 3**

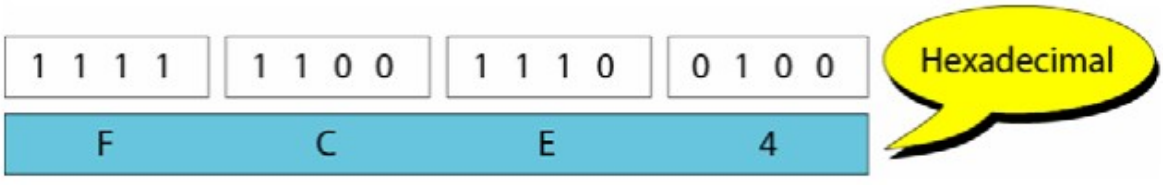
¿Cuál es el patrón de bits para 248?

**Solución**

Cada dígito octal se escribe como su patrón de bits equivalente para obtener 010100.

## d) Binario a hexadecimal y hexadecimal a binario.

La conversión de un patrón de bits a notación hexadecimal se realiza por medio de la organización del patrón en grupos de cuatro y luego hallar el valor hexadecimal para cada grupo de cuatro bits. Para una conversión de hexadecimal a patrón de bits se convierte cada dígito hexadecimal a su equivalente de cuatro bits.




**Ejemplo 1**

Determine el hexadecimal equivalente de $110011100010_2$.

**Solución**

Cada grupo de cuatro bits se traduce a un dígito hexadecimal. El equivalente es **x**CE2.

**Ejemplo 2**

Determine el hexadecimal equivalente del patrón de bits 0011100010.

**Solución**

El patrón de bits se divide en grupos de cuatro bits (a partir de la derecha). En este caso, se añaden dos 0 más a la izquierda para hacer el número total de bits divisible entre cuatro. Así el número binario 000011100010, se expresa en hexadecimal como **x**0E2.

**Ejemplo 3**

¿Cuál es la conversión de **x**24C a binario?

**Solución**

Cada dígito hexadecimal se escribe como su patrón de bits equivalente y se obtiene 001001001100.

# 4. Operaciones artísticas.

## a) Suma en números binarios.

La tabla de sumar, en binario, es mucho más sencilla que en decimal. Sólo hay que recordar cuatro combinaciones posibles.

| Suma |
| --- |
| 0 + 0 = 0 |
| 0 + 1 = 1 |
| 1 + 0 = 1 |
| 1 + 1 = 10 |

Ejemplo.

```
  100110101
+  11010101
-----------
 1000001010
```

Operamos como en el sistema decimal: comenzamos a sumar desde la derecha, en nuestro ejemplo, 1 + 1 = 10, entonces escribimos 0 en la fila del resultado y llevamos 1 (este "1" se llama arrastre). A continuación se suma el acarreo a la siguiente columna: 1 + 0 + 0 = 1, y seguimos hasta terminar todas la columnas (exactamente como en decimal).

## b) Producto de números binarios.

El algoritmo del producto en binario es igual que en números decimales; aunque se lleva cabo con más sencillez, ya que el 0 multiplicado por cualquier número da 0, y el 1 es el elemento neutro del producto.

| Multiplicación |
| --- |
| 0 x 0 = 0 |
| 0 x 1 = 0 |
| 1 x 0 = 0 |
| 1 x 1 = 1 |

Por ejemplo, multipliquemos 10110 por 1001:

```
    10110
     1001
---------
    10110
   00000
  00000
 10110
---------
 11000110
```

# 5. Operaciones lógicas.

## a) Operador AND

La función AND es equivalente a la conjunción "Y" de nuestra lengua, denominada también multiplicación lógica. Al aplicar esta función sobre dos variables (A y B), el resultado (S) será el siguiente: El resultado será verdadero si y sólo si, A y B son verdaderos. Es decir:

| A | B | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## b) Función NOT

Si la variable A es verdadera, el resultado será falso y viceversa:

| A | S |
|---|---|
| 1 | 0 |
| 0 | 1 |

## c) Función OR

La función OR es equivalente a la conjunción "O" de nuestra lengua, también denominada suma lógica. Al aplicar esta función sobre dos variables (A y B), el resultado (S) será el siguiente: Si al menos una de las dos variables tiene un valor verdadero (1), entonces el resultado será verdadero. Para esta función con dos variables son posibles cuatro combinaciones, o sea:

| A | B | S |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# 6. Representación de caracteres.

Un fragmento de texto en cualquier idioma es una secuencia de símbolos usados para representar una idea en ese idioma. (ej, ABC…Z, 0,1,2,3…9) .

Se puede representar cada símbolo (de lenguajes humanos) con un patrón de bits.

| | | | | | | |
|---|---|---|---|---|---|---|
| B | Y | T | E | b | y | t |
| 1000010 | 1011001 | 1010100 | 1000101 | 0110010 | 1001001 | 1000100 |

**¿Cuántos bits se necesitan en un patrón de bits para representar un símbolo en un idioma?**

Esto depende de cuantos símbolos haya en la secuencia (idioma). La longitud del patrón de bits que representa un símbolo en un idioma depende del número de símbolos usados en ese idioma. Mas símbolos significan un patrón de bits mas grande.

- Un patrón de bits de dos bits puede tomar combinaciones diferentes formas diferentes: 00, 01, 10 y 11 y cada una de estas representará un símbolo.

  00 la a 01 la b 10 la c 11 la d Necesitamos más bit.

- Un patrón de tres bits puede tomar ocho combinaciones diferentes diferentes: 000, 001, 010, 011, 100, 101, 110 y 111 y cada una de estas representará un símbolo.

  000 la a 001 la b 010 la C 011 la D

  100 la E 101 la F 110 la G 111 la H Necesitamos más bit.

- Cuantos vamos a necesitar pues depende de la cantidad de caracteres a codificar pero podemos pensar que con un patrón de bits de **n** bits puede tomar $\mathbf{2^n}$ combinaciones diferentes

- Ejemploç

  Para n=2 es $\mathbf{2^2 =}$ **4 combinaciones** diferentes.

## a) Códigos

Para comunicarnos con el ordenador, vamos a codificar nuestro alfabeto con secuencias 0 y 1, cuantos caracteres necesito codificar :

- Caracteres Caracteres Alfabéticos Alfabéticos: Mayúsculas Mayúsculas y minúsculas minúsculas del alfabeto alfabeto inglés.
- Caracteres Numéricos: Del cero al nueve.
- Caracteres Caracteres Especiales Especiales: { }, , #, $, %, &, _, +. -. , * /, \, ( ), , ?, !, [, ]
- Caracteres de Control: Representan órdenes de control al ordenador: EOL, EOT, SYNC, ESC, BEEP, CTRL
- Caracteres Gráficos: Permiten “dibujar” figuras o iconos elementales.

Generalmente nos referiremos referiremos en programación programación a estas clases como:

- Caracteres alfanuméricos: que abarcan las dos primeras.
- Caracteres de texto: que abarcan las tres primeras categorías.
- Con 7 bits se puede representar hasta 128 caracteres distintos por lo que los primeros códigos tuvieron ese tamaño.

**ASCCI American American Standard Standard Code for Information Information Interchange**

Usa 7 bits, puede representar representar hasta m = 128 caracteres caracteres distintos distintos.

Ejemplo:

| Carácter | ASCII 7 | bits Representación interna |
|---|---|---|
| '0' | $060_8$ | 0110 000 |
| '9' | $071_8$ | 011 1001 |
| 'A' | $101_8$ | 100 0001 |
| '(' | $050_8$ | 010 1000 |

| Binary | Dec | Hex | Glyph |
|---|---|---|---|
| 010 0000 | 32 | 20 | SP |
| 010 0001 | 33 | 21 | ! |
| 010 0010 | 34 | 22 | " |
| 010 0011 | 35 | 23 | # |
| 010 0100 | 36 | 24 | $ |
| 010 0101 | 37 | 25 | % |
| 010 0110 | 38 | 26 | & |
| 010 0111 | 39 | 27 | ' |
| 010 1000 | 40 | 28 | ( |
| 010 1001 | 41 | 29 | ) |
| 010 1010 | 42 | 2A | * |
| 010 1011 | 43 | 2B | + |
| 010 1100 | 44 | 2C | , |
| 010 1101 | 45 | 2D | - |
| 010 1110 | 46 | 2E | . |
| 010 1111 | 47 | 2F | / |
| 011 0000 | 48 | 30 | 0 |
| 011 0001 | 49 | 31 | 1 |
| 011 0010 | 50 | 32 | 2 |
| 011 0011 | 51 | 33 | 3 |
| 011 0100 | 52 | 34 | 4 |
| 011 0101 | 53 | 35 | 5 |
| 011 0110 | 54 | 36 | 6 |
| 011 0111 | 55 | 37 | 7 |
| 011 1000 | 56 | 38 | 8 |
| 011 1001 | 57 | 39 | 9 |

| Binary | Dec | Hex | Glyph |
|---|---|---|---|
| 011 1010 | 58 | 3A | : |
| 011 1011 | 59 | 3B | ; |
| 011 1100 | 60 | 3C | < |
| 011 1101 | 61 | 3D | = |
| 011 1110 | 62 | 3E | > |
| 011 1111 | 63 | 3F | ? |
| 100 0000 | 64 | 40 | @ |
| 100 0001 | 65 | 41 | A |
| 100 0010 | 66 | 42 | B |
| 100 0011 | 67 | 43 | C |
| 100 0100 | 68 | 44 | D |
| 100 0101 | 69 | 45 | E |
| 100 0110 | 70 | 46 | F |
| 100 0111 | 71 | 47 | G |
| 100 1000 | 72 | 48 | H |
| 100 1001 | 73 | 49 | I |
| 100 1010 | 74 | 4A | J |
| 100 1011 | 75 | 4B | K |
| 100 1100 | 76 | 4C | L |
| 100 1101 | 77 | 4D | M |
| 100 1110 | 78 | 4E | N |
| 100 1111 | 79 | 4F | O |
| 101 0000 | 80 | 50 | P |
| 101 0001 | 81 | 51 | Q |
| 101 0010 | 82 | 52 | R |
| 101 0011 | 83 | 53 | S |

| Binary | Dec | Hex | Glyph |
|---|---|---|---|
| 101 0100 | 84 | 54 | T |
| 101 0101 | 85 | 55 | U |
| 101 0110 | 86 | 56 | V |
| 101 0111 | 87 | 57 | W |
| 101 1000 | 88 | 58 | X |
| 101 1001 | 89 | 59 | Y |
| 101 1010 | 90 | 5A | Z |
| 101 1011 | 91 | 5B | [ |
| 101 1100 | 92 | 5C | \ |
| 101 1101 | 93 | 5D | ] |
| 101 1110 | 94 | 5E | ^ |
| 101 1111 | 95 | 5F | _ |
| 110 0000 | 96 | 60 | ` |
| 110 0001 | 97 | 61 | a |
| 110 0010 | 98 | 62 | b |
| 110 0011 | 99 | 63 | c |
| 110 0100 | 100 | 64 | d |
| 110 0101 | 101 | 65 | e |
| 110 0110 | 102 | 66 | f |
| 110 0111 | 103 | 67 | g |
| 110 1000 | 104 | 68 | h |
| 110 1001 | 105 | 69 | i |
| 110 1010 | 106 | 6A | j |
| 110 1011 | 107 | 6B | k |
| 110 1100 | 108 | 6C | l |

| Binary | Dec | Hex | Glyph |
|---|---|---|---|
| 110 1101 | 109 | 6D | m |
| 110 1110 | 110 | 6E | n |
| 110 1111 | 111 | 6F | o |
| 111 0000 | 112 | 70 | p |
| 111 0001 | 113 | 71 | q |
| 111 0010 | 114 | 72 | r |
| 111 0011 | 115 | 73 | s |
| 111 0100 | 116 | 74 | t |
| 111 0101 | 117 | 75 | u |
| 111 0110 | 118 | 76 | v |
| 111 0111 | 119 | 77 | w |
| 111 1000 | 120 | 78 | x |
| 111 1001 | 121 | 79 | y |
| 111 1010 | 122 | 7A | z |
| 111 1011 | 123 | 7B | { |
| 111 1100 | 124 | 7C | \| |
| 111 1101 | 125 | 7D | } |
| 111 1110 | 126 | 7E | ~ |

**ASCII extendido**: El tamaño de cada patrón es de 1 byte (8 bits).

**Unicode**

Ante la necesidad de un código de mayores capacidades, una coalición de fabricantes de hardware y software desarrollo un código que utiliza 16 bits y puede representar hasta 65 536 (216) símbolos.

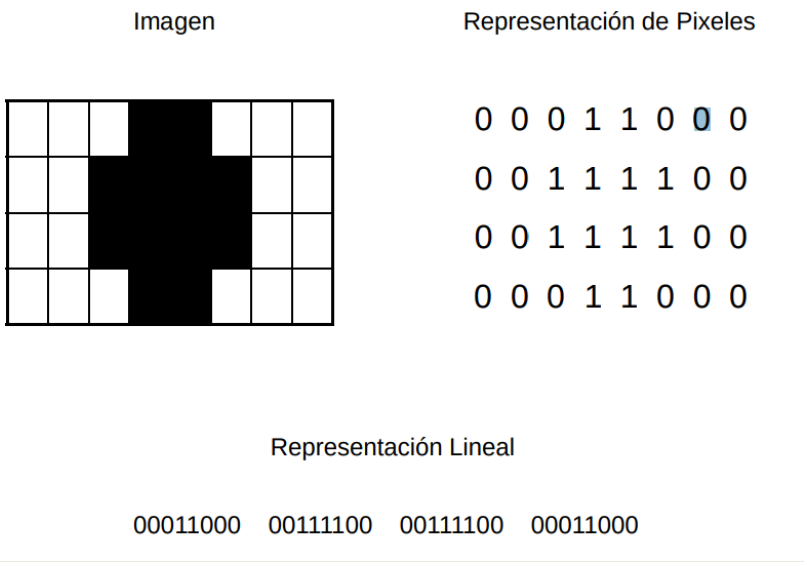
Diferentes seccione del código se asignan a los símbolos de distintos idiomas en el mundo.

# 7. Representación de imágenes.

Las imágenes se representan en un ordenador mediante uno de dos métodos: gráficos de mapa de bits o gráficos de vectores.

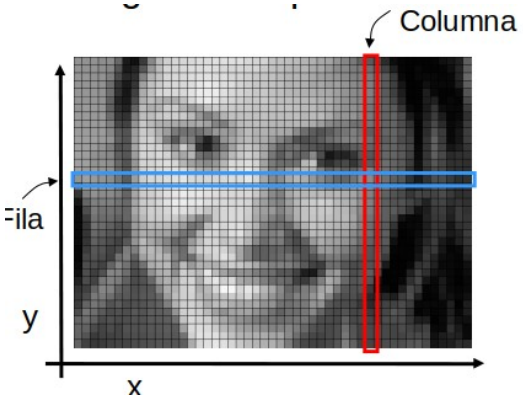
## a) Gráficos de Mapa de Bits.

Una imagen se divide en una matriz de pixeles. A cada pixel se le asigna un patrón de bits. El tamaño y el valor del patrón depende de la imagen, para una imagen formada solo por puntos blancos y negros, un patrón de un bit es suficiente para representar un pixel. Los almacenan en la computadora




**Un poco de nomenclatura**

- Nº de columnas de la matriz: **ancho** de la imagen (**width**).
- Nº de filas de la matriz: **alto** de la imagen (**height**).
- Eje horizontal: **eje x**.
- Eje vertical: **eje y**.
- Normalmente el tamaño de la imagen se expresa como: **ancho x alto**



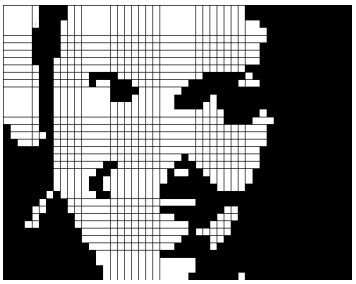

Ejemplo. Tamaños típicos:

320x240, 640x480,

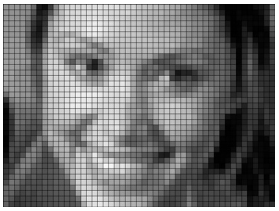
800x600, 1024x768, …

- si i es una imagen, i(x, y) será el valor del píxel en la columna x, fila y.
- Entre otros valores un píxel puede indicar:

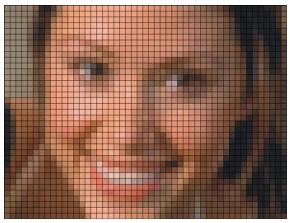
- Cada píxel representa el valor de una magnitud física.
- Cantidad de luz en un punto de una escena.
- Valor de color (cantidad de radiación en la frecuencia del rojo, verde y azul).
- Nivel de radiación infrarroja, rayos X, etc. En general, cualquier radiación electromagnética.
- Profundidad (distancia) de una escena en una dirección.
- Temperatura de cada punto de la escena.



- Imagen binaria: 1 píxel = 1 bit
  - 0 = negro; 1= blanco
- Imagen en escala de grises: 1 píxel = 1 byte
  - Permite 256 niveles de gris
  - 0 = negro; 255 = blanco



- Imagen en color: 1 píxel = 3 bytes
  - Cada píxel consta de 3 valores: (Rojo, Verde, Azul)
  - Un byte por color
  - 16,7 millones de colores posibles



## b) Representación de Gráficos de Vectores

Este método no guarda los patrones de bits. La imagen se descompone en una combinación de curvas y líneas. Cada curva o línea se representa por medio de una formula matemática. En este caso cada vez que se dibuja la imagen, la formula se vuelve a evaluar.