***Tips Harvard - Introduction to Computer Science***

***0 week***

nota: acá se escriben los pequeños detalles que se nos escapen del video, de interés en conocimiento, que no estén resumidos, contemplados o muy bien detallados en las notas de aprendisaje de las clases. Son tips.

... Ascii, de momento, lo interpreto como un sistema (mapeo estandar) que permite darnos la equivalencia, en números binarios (lenguaje de maquina), del conjunto de números, puntuaciones y textos (palabras) que usamos los humanos para entendernos entre sí; es decir, el sistema ASCII es quién asigna la equivalencia entre textos, puntuaciones y números (de la capacidad interpretativa del hombre cotidiano) con el lenguaje binario de las maquinas; dicho lo anterior, por ejemplo, es ASCII quien define que “HI!” en lenguaje maquina pueda ser descrito como: *01001000, 01001001 y 00100001 (donde cada conjunto de 8 bits -de 8 numeros binarios- equivalen a un caracter: “H”, “I”, “!”, respectivamente).*

dato: 8 bits equivalen a 1 byte.

El sistema ASCII es definido por Wikipedia así: ASCII, Código Estándar Americano para Intercambio de Información, es una codificación de caracteres estándar para la comunicación electrónica. Los códigos ASCII representan texto en computadoras, equipos de telecomunicaciones y otros dispositivos. La mayoría de los esquemas de codificación de caracteres modernos se basan en ASCII, aunque admiten muchos caracteres adicionales.

*tip: Con ocho bits, o un byte, podemos tener 256 valores diferentes (incluido cero). El valor más alto hasta el que podemos contar sería 255; 2x2x2x2x2x2x2x2 = 256*

*A qué se debe esa secuencia de “2x2x2x2x2x2x2x2”?*

*Si sabemos que por cada bit se pueden asumir, como mucho, dos posibilidades de dígitos los cuáles son: 0 y 1 (por tratarse de un lenguaje binario, base 2). Y sabemos, además, que un carácter está compuesto por 8 bits (1 byte). Entonces, cuantas combinaciones posibles de bits podemos hacer para saber el número máximo de caracteres que podemos generar, a partir de los bits, respetando en cada uno su propia combinación de binarios en los 8 bits de espacio disponible que tendría cada caracter.*

*Pues bien, en matemáticas, hay una herramienta lógica que nos permite saber con exactitud e inmediatamente cuántas posibilidades máximas de combinaciones hay en un conjunto de números determinados (en este caso, respetando la combinación de dígitos y estableciendo también el límite máximo de dígito por número, que puede tomar un número -nuestro caso, 0 y 1)... A esa herramienta lógica matemática se llama: “Potenciación”.*

*Cuántas formas de combinación independientes hay, entonces, en 8 espacios de número que sólo pueden contemplar dos dígitos por número: 0 y 1.*

***Respuesta: 256.***

de interes: Otros caracteres, como letras con acentos y símbolos en otros idiomas, **son parte de un estándar llamado Unicode**, que usa más bits que ASCII para acomodar todos estos caracteres. Es otro mapeo.

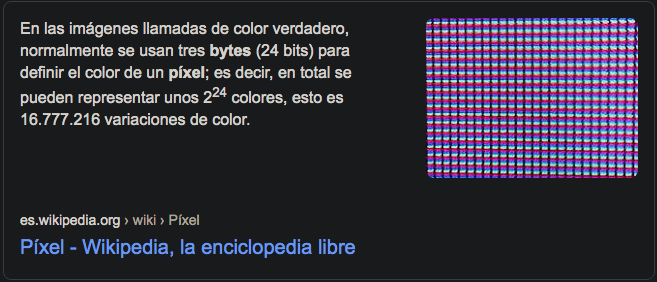
Aprende a leer en código binario: https://blog.makeitreal.camp/aprende-a-leer-en-codigo-binario/

***Imagenes, video & sonido***

*Las imagenes también están formadas por bits (o bytes) y hacen parte del mismo mapeo “Unicode”; pero, a diferencia de un caracter, cuando usted amplia mucho una imagen, usted se da cuenta de la existencia de una especie punticos (o cuadritos) que se empiezan a denotar, cada vez más, en la medida que amplia la imagen en cuestión (a este proceso se le llama* ***“pixelación”****); cada puntico de esos (o cuadritos) se les llama técnicamente:* ***“pixel”****. A partir de los pixeles es que se calcula, en datos, la cantidad de bits (o bytes) que tiene una imagen en su conjunto; es decir, conociendo los bits (o bytes) que tiene cada pixel. Los pixeles, como ya sabrá porqué, tienen más bits que un caracter: exactamente triplica en datos su peso en bits con relación a los de un caracter: se compone por 24bits (3 bytes).*

*Citemos lo siguiente: “Con solo bits, también podemos asignar números a colores. Hay muchos sistemas diferentes para representar colores, pero uno común es el* ***RGB (Red, Green, Blue)****, que representa diferentes colores al indicar la cantidad de rojo, verde y azul dentro de cada color” (o dentro de cada pixel).*

*A partir de lo anterior convengamos entonces que, cada pixel se compone por una combinación de estos 3 colores, donde cada color tiene tal independencia que se compone por un byte (8 bits). Dicho esto, al unir los tres colores (y sabiendo que cada color en un pixel está compuesto por 1 byte, 8 bits); entonces, como son tres colores por pixel (rojo, verde y azul), llegariamos a la conclusión de que cada pixel pesa en datos 3 bytes (es decir, 24bits). Lo cuál nos llevaría a hacer la siguiente pregunta: ¿cúantos combinaciones máximas posibles de color puede interpretar la maquina; es decir, cuántos colores se pueden originar a partir de la lectura binaria que tiene un computador si sabemos que cada pixel se compone por 24 bits (3 bytes)?* ***Respuesta:***



*224 = 2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2*

**datos interesantes:** La **resolución** de una imagen es la cantidad de píxeles que hay, horizontal y verticalmente, por lo que una imagen de alta resolución tendrá más píxeles y requerirá más bytes para almacenarse (por esto “pesan” más las imagenes HD o de alta calidad).

Los videos se componen de muchas imágenes, **que cambian varias veces por segundo para darnos la apariencia de movimiento**, como lo haría un libro animado antiguo .

La música también se puede representar con bits, **con asignaciones de números a notas y duraciones**, o asignaciones más complejas de bits a frecuencias de sonido en cada momento.

Los formatos de archivo, como JPEG y PNG, o documentos de Word o Excel, también se basan en algún estándar que algunos humanos han acordado, para representar información con bits.

***Algoritmo y Pseudocódigo***

*Antes de pasar directamente a lo que es programar o manipular un lenguaje de programación; debemos tener conocimiento, primeramente, de dos procesos:* ***Algoritmo y Pseudocódigo****, uno origina al otro respectivamente y ambos tienen la finalidad de desarrollar la lógica y ver el área de programación, propiamente, como una manera adicional de darle resolución lógica a nuestros problemas. (no como unos garabatos compuestos por puro código incomprensible para el entendimiento del hombre medio)*

*De manera salvajemente simplificada... el área del* ***Algoritmo****, más que ser un tema de único interés a la programación, está lejos de ser solamente eso. Cuando hablamos de Algoritmo podemos aplicarlo a cualquier área del conocimiento; pues, explicitamente se puede entender por “Algoritmo”, como ya lo mencionamos, una forma de resolución de problemas; esto a partir de un planteamiento inicial del mismo (esa introducción que nos da el planteamiento del problema, la entrada, técnicamente es llamada como: input). El Algoritmo como tal comprende todos los procesos que se lleven a cabo, paso a paso (sin subestimar ninguno, ojo), entre lo que ocurre en la declaración del problema inicial hasta su eventual posible solución (esta solución, la salida, es técnicamente conocida como: output).*

*En resumidas cuentas, el Algoritmo se reduce a ese puente que hay entre problema-solución y que se compone por un conjunto de instrucciones y operaciones sistematicas (y no-ambiguas) que se ejecutan, paso a paso, para llevar a cabo la consecución de un fin: darle solución al problema en cuestión.*

*Lo que viene sería la parte del* ***Pseudocódigo****, el cuál sí se enfoca o ya pretende direccionarse al área de la programación; pues, si bien no es código aún, de ésta intenta replicar su estructura lógica y de sintaxis (simulando uso de funciones, condicionales, expresiones booleanas y bucles); pero sin variar todavía, la manera interpretativa y de resolución del problema elaborada anteriormente con Algoritmos y en un lenguaje humano. (****Pseudocódigo*** *es la representación del* ***Algoritmo*** *en* ***programación).***