Programación: Sistematizar soluciones a problemas conocidos

Esfuerzo que viene hace muchos años (calculadora mecánica que ayuda a sistematizar operaciones matemáticas)

George Boole --> trató de cambiar la manera que vemos la matemática (decimal >> a 0 y 1) --> inventó el código binario. Se pueden hacer sumas restas y operaciones que no existen en la m. original

La matemática decimal es fáci de representar situaciones comunes

ADA LOVELACE --> tomó tarjetas perforadsa (agujero es 1, no agujero es 0): primeras piezas de software

1890: censo en USA con tarjetas perforadas

Segunda Guerra mundial: Alan Turing, interceptaba mensajes del ejército opuesto de manera encriptada y logra desencriptar mensajes.

1900s: dejamos de depender de aparatos gigantescos. Aparecen PCs, discos magnéticos impulsando a la informática a lugares dodnde todos podemos acceder

Usuario final : sistemas operativos gráficos y mouse

Programadores: lenguajes de programación de alto nivel (no sólo mensajes rudimentales a través de código binario, sino lenguajes más cercanos a lo que hablamos en el cotidiano, como C y Pascal)

Todo es código binario, formado a partir de Abstracción de capas (software, sistema operativo, lenguaje rudimentario, etc)

Para programar elegimos programas que se traducen capa a capa hasta llegar a la solución final (nos permite olvidar que hay capas por debajo ya que están ya solucionadas)

Vint Cerf y Bob Kahn: cómo hacemos para comunicar dos computadoras?

Hasta ese momento, grabar en un soporte (disco)

Ellos toman la aventura de inventar INTERNET.

Lo primero que hay que inventar es un Protocolo de comunicación para computadoras. Formato determinado y estandar mundial.

Internet funciona a través de modelo OSI (siete capas que aseguran comunicación entre compuatdoras=

7 - Aplicación --> cada aplicación muestra el mensaje que ya fue enviado

4 – Transporte (2 protocolos TCP IP)

3 - Red (2 protocolos TCP IP)

1 - Presentación --> capa física (mensaje desmenuzado en 0 y 1): cómo se transmite de un dispositivo a otro (por ej electricidad, mediante ondas magnéticas)

Cada dispositivo conectado tiene una dirección IP: basada en 4 dígitos, donde cada dígito es un nro entre 0 y 255.

[www.digitalhouse.com](http://www.digitalhouse.com) 🡪 ese pedido llega al servidor de DH (una computadora con dos características importantes: nunca se apaga y tiene instalado un webserver). Un web server es como un programa pendiente de que se “llame” a su página.

ISP: internet service provider: tiene instalada un servidor DNS (tipo páginas amarillas enormes que traducen digitahouse.com en IP de servidor). Si el DNS de fibertel no sabe, eleva su pedido a DNS con mayor autoridad. Ese pedido eventualmente pasa por la nube (telaraña gigante de muchas computadoras enrutando nuestro pedido). Siempre encuentra un camino óptimo. ((Arquitectura cliente-servidor))

**PHP: hypertext preprocessor**

Arquitectura cliente-servidor permite que usuario pida sitio web y obtenga html. Esto nos da una web estática: para cada página, una html en el servidor responde.

No es viable porque no puede haber un html para cada video de youtube por ej.

>>>> WEB DINAMICA: todos los videos tienen el mismo encabezado, comentarios, videos relacionados, un video. Por lo tanto se escribe el html con pequeños huecos: “acá viene el video” “acá vienen los comentarios”, etc. Todo eso depende de URL. Para eso aparecen los motores de procesamiento del lado del servidor. Un lenguaje de servidor toma el pedido y genera el html final

Ergo tenemos siempre el mismo html para todos los videos que a través de lógica de información, se va rellenando.

Web server trabaja con motor de procesamiento que accede a base de datos.

Motor de procesamiento: puede funcionar con java, phyton, PHP.

Tenemos que hacer que suceda la traducción entre php y el html final. Para replicar este ciclo, hay que instalar un web server (por ej Apache).

**Como escribir PHP**

Se ingresa mediante local host: url “localhost/inicio.php”

no se puede escribir el archivo en cualquier carpeta sino en ht docs

archivos ya no son html sino php

inicio.php en atom y se hace el código html como siempre

para escribir huequitos: etiqueta php (adentro no puede haber código html)

<?php

Echo “Hola Mundo”;

?>

Echo se usa para decir “agregá al html”

Echo date(“H:i:s”); 🡪 muestra la hora

Variables en php: $

<?php

$segundos = date(“s”);

If ($segundos > 30) {

Echo “Estamos en el final del minuto”;

}

Else {

Echo “Estamos en el principio”}

?>

PHP: nos ayuda a cambiar el contenido de acuerdo a algunas variables. Por ejemplo si está logeado, cambia botón de login por foto de perfil (web dinámica)

PHP: es un lenguaje que se ejecuta del lado del servidor, pensado para el desarrollo web.

Permite embeber html en nuestros archivos php

Es multiparadigma: enfocar nuestro código (por ej orientado a objetos)

Lenguaje “interpretado” (vs. Compilado: donde hay que tocar un botón de compilar): no requiere compilación. El intérprete ejecuta línea por línea.

Multiplataforma: funciona en cualquier SO.

**VARIABLES**

Sirve para el guardado de información, que necesitamos que retenga.

Las variables tienen etiquetas (nombres)

Por ej $nombre = “Dario”;

$ para guardar y llamar variables

= operador de asignación

; siempre terminar con punto y coma

$nombre= “Laura”;

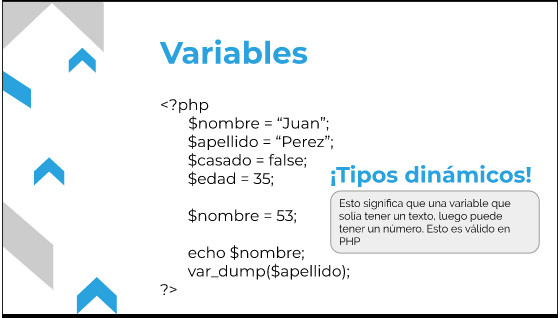
$edad= 32;

Función echo: escribi qué variable se quiere imprimir (echo $nombre;)

Var\_dump($nombre) 🡪 sirve para analizar una variable (string, largo, etc); se puede usar separando por comas varias

Si a una variable la vuelvo a escribir, se reasigna. Depende de dónde está el echo muestra una cosa o la otra

Ojo nombre de variable no puede comenzar con un número



TIPO DE VARIABLES

1. Integer: números enteros (positivos, negativos o 0)
2. Float: números decimales (positivos, negativos o 0, donde agregamos coma. En PHP se usa . y no coma)
3. String: cadenas de texto. Comillas dobles o simples.
4. Booleanos: valores de verdad (true or false)
5. Null. Representa variable sin valor. Existe y no está vacía.

Error unexpected: es porque olvidamos punto y coma

Dentro de una string pueden usarse variables, pero hay que usar COMILLAS DOBLES. Si usamos comillas simples, nos va a mostrar $lavariable en lugar de su valor.

Por ej: “hola mi nombre es Lala” (INTERPOLAR) vs ‘hola mi nombre es $nombre’

**ARRAY**

Nos deja almacenar un conjunto de valores, aun cuando son de distinto tipo

Arrays comienzan en posición 0

$miArray=[“lala”, “minblack”]; 🡪 se inicializa el array

$miArray[4] = “1987” 🡪 agrega un valor a la Array (corchetes antes del =), en la posición 4 (si no pongo nada se pone en la primera disponible, o sea en la 3)

$miArray[3] = “Abril”

Echo $miArray[1]; 🡪 devuelve Lala

No se puede hacer echo de un array entero

Símbolo => separa posición del valor. Sirve en el código para inicializar el array. Por ej

$miArray= [

1=> “lala”

2=>”minblack”

3=> [“Boca”, “DigitalHouse”] ------(arrays dentro de arrays)

]

**ARRAYS ASOCIATIVOS**

Permite que posiciones de una array no sean necesariamente numéricas, sino llamarlas nombre, apellido, edad. Es cómodo para representar en forma legible el uso de arrays.

Cada vez que citamos una posición, lo hacemos con una string.

$miArray= [

“nombre”=> “lala”,

“apellido”=>”minblack”,

“otros”=> [“Boca”, “DigitalHouse”]

];

Para agregar: $miArray[“edad”]=32;

Para llamar a arrays multidimensionales: $miArray[“otros”][1] 🡪 devuelve DigitalHouse

**OPERADORES**

Floats e integer se puden hacer operaciones matemáticas

Strings se pueden unir con concanetación

$numero1=31;

$numero2=58;

$resultado= $numero1 + $numero2;

\*\* operador de potencia

% operador de porcentaje, que hace división pero almacena el resto en lugar del resultado (útil para números pares) n%2=0 (números pares)

$nombre=”Laura”;

$apellido=”Mingrone”;

$texto= “Mi nombre es “ . $nombre . “ “ . $apellido (se concatena con puntos)

Arrays no se pueden usar dentro de strings, es por eso que se usa la concatenación