Década del 40. Primer sistema de codificación. Sistema mnemotécnico de codificación ENIAC.

Menciona a Kobol, de los 60, que aún sigue en uso.

Gran parte de los sistemas operativos (el Kernel) están escritos en C. Ya que combina el bajo nivel (hardware) y al mismo tiempo es de alto nivel.

Pascal es otro de los viejos, que se puede seguir usando.

Son lenguajes de propósito general. Se concibieron para programar cualquier tipo de programa. En los 80 empiezan a surgir lenguajes de nicho, no se pueden usar para todo: Ada, Perl, TCL.

En los 90 aparecen lenguajes que mezclan propósito general con nicho. Java aparece ahí como lenguaje embebido. Nacen como nichos, pero terminan sirviendo para muchas mas cosas. También surge Python, muy de moda hoy día por machine learning. PHP es de 1995. Ruby.

Desde los 2000, ActionScript. Era para hacer animaciones 2d o banners. Se lo conocía como flash. C#, GO, Swift.

Tipos de lenguajes

Compilados. Aquellos que a partir de código, se genera un programa (secuencia de 0 y 1) ejecutado por el procesador, ayudado por el sistema operativo.

Interpretados. Se parte de un código pero no se compila. Se genera un bitecode, un intermedio, que no es entendido por el procesador, falta el interprete. Es mas lento.

Java es interpretado, se ejecuta en Java Virtual Machine.

Compilado es mas rápido, aunque hace falta el compilador. Se debe hacer un programa para cada tipo de procesador.

Interpretado es mas lento, pero solo se escribe una vez, ya que el interprete traduce a las necesidades de cada procesador.

Entre ambos, están los híbridos. El interprete puede compilar también. Just in time.

Lenguajes tipados. Se almacenan datos pero se les debe decir de qué tipo son.

No tipados. El interprete consigue el tipo de dato autónomamente.

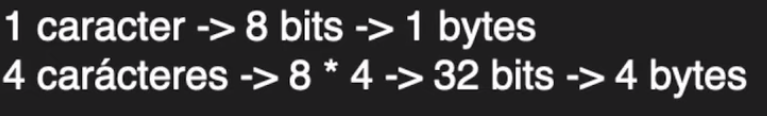
Tipos de datos

Tabla ASCII

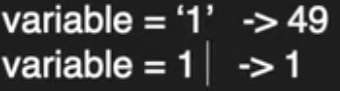
Primitivos:

Char= ‘2’;

String es un conjunto de caracteres. Cada letra se guarda un espacio de memoria distinto, consecutivos.



Int.



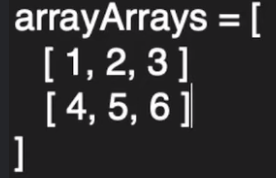
49 y 1 hacen referencia a la posición que ocupan en la tabla ASCII. O sea, es el espacio de memoria al que apunta la variable.

Boolean.

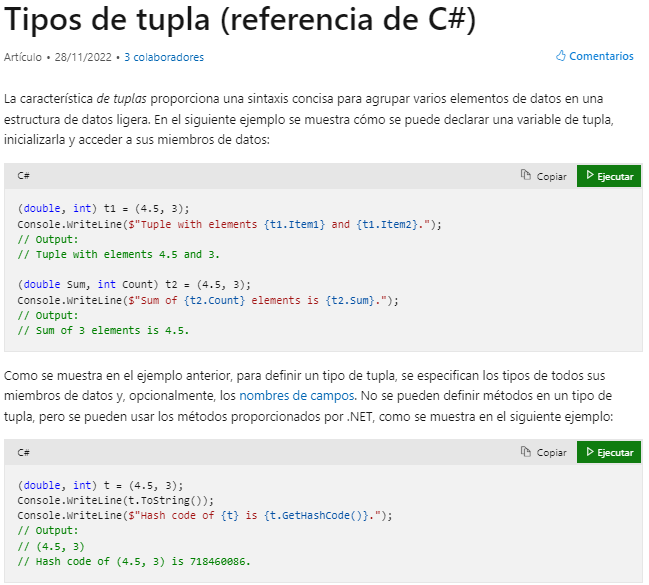
Tipos de datos complejos:

Array. Conjunto o colección de tipos de datos simples. Siempre es del mismo tipo de datos.

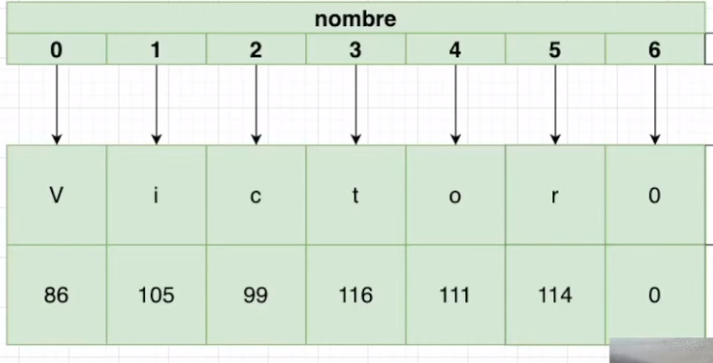
Array bidimensional es un array de arrays.

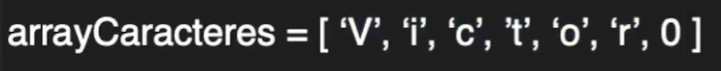


Tupla. Similar a un array. Los elementos de un array son mutables.



Habla también del array asociativo, también se llaman mapas. Casi igual a los diccionarios. Asocia Claves y valores.





Dice el muchacho que cuando guardamos un string, nos agrega un ‘0’ al final, que marca el fin del string.

Funciones

Cuando se pasa un parámetro por referencia se pasa una dirección de memoria (se la comparte), la cual podrá ser modificada en la función. No crea una copia (la copia se crea en la memoria. Ocupa el doble de espacio) de los valores pasados por parámetros como con los parámetros por valor.

Cuando se pasan objetos por referencia, se pasan siempre por referencia, de forma implícita.

Errores

Dice que es buena práctica agregar comentarios al código que expliquen lo que se va a hacer, aunque sin exagerar.

No usar el depurador y si printeos para ver por dónde va el programa.

Depuración

Busca errores de las Apps en tiempo de ejecución.

Pondera además, la importancia de la depuración remota.

Puntos de ruptura. Donde se le dice que debe detenerse.

Dice que el breakpoint se puede transformar en un watcher (expresiones de vigilancia, cre0). Es decir, funciona como breakpoint si se cumple alguna condición (suma==16), si no, pasa de largo y es como si no hubiera breakpoint.

Intro a la POO

Instanciar crea un espacio en memoria.

Privacidad, abstracción y encapsulación:

Los atributos por default son public.

Cuando usa setters prefiere poner siempre this.tipo=valor

Siempre usa el this, por mas que lo que se le pasa como parámetro se llame distinto.

La convención para los getters de los booleans no es getEstado sino IsEstado(), el setter sería SetEstado()

Herencia e interfaces:

Polimorfismo. Las clases hijas utilizan la misma función, pero funcionan de manera distinta. Ahí entran en juego los override.

También dice que el polimorfismo puede referir a cuando las funciones están sobrecargadas. Funciones con mismo nombre, pero distintos parámetros.

“Un método es una función dentro de una clase”.

Lenguajes compilados e interpretados

Compilador. Agarra un fichero de repositorio y lo traduce a código máquina o a otro lenguaje, por esto dice que hay 2 tipos de compiladores.

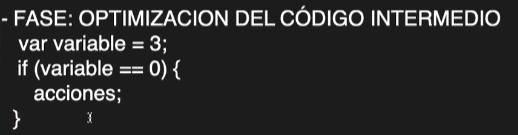
Analizador léxico. Lee cada carácter del código. Genera un string de tokens (secuencia de caracteres) o tokens.

Como parte del analizador léxico, detecta las palabras reservadas. Los identificadores, símbolos, números, etc.

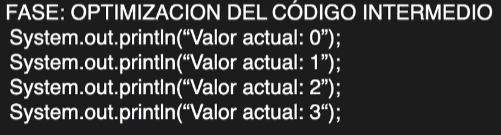
Análisis sintáctico. Debe analizar si los tokens analizados tienen coherencia.

Genera luego de esto un código intermedio. Ya no usa lo que codeamos, sino otro lenguaje, mas profundo: abstract syntax tree.

Algunos compiladores no tiene la fase siguiente, de optimización de código intermedio, algo mas óptimo que lo escrito por nosotros.



Ese código, correcto desde lo sintáctico y también desde lo léxico, jamás será ejecutado, ya que 3 es distinto de 0. Dice que algunos compiladores, directamente eliminan esa parte del código.



También, afirma, podría reemplazar un for por esta expresión de arriba, ya que haría mas eficiente al programa. “Unroll loops” se llama esta técnica.

Fase de generación de código. Genera o código máquina o un código para ser interpretado. Es la fase final. Si bien, dice que podría haber un optimizador de código generado.

Lenguaje compilado se ejecuta directamente en nuestro procesador a través del sistema operativo.

Lenguaje interpretado no se ejecutan directamente por el sistema operativo sino por una máquina virtual (o interprete). Java Virtual Machine (JVM) interpreta Java. Java genera un código para que lo lea JVM.

Lenguajes que son tanto interpretados como compilados. C# es uno de estos. Se genera código intermedio MSIL.

Just In Time o Jit Compiler: Si una parte del código se ejecuta muchas veces, la compila. Esto se hace para ganar rendimiento.

Los lenguajes compilados son mucho mas rápidos. Ya que los lenguajes interpretados usan un lenguaje adicional. Mas allá de esto, afirma que hoy en día no hay tanta diferencia de rendimientos entre lenguajes.

La ventaja de los lenguajes interpretados es que se pueden ejecutar en cualquier sistema operativo y plataforma. Eso no es así con los compilados.

Cuestiona JavaScript por no ser un lenguaje tipado. TypeScript lo convierte en tipado. TypeScript se compila a JavaScript.

