Trabajo práctico N°2

Curso K2055

TypeScript y Cobol

Grupo: 6

Marcos Ezequiel Benegas - 175.388-5

Ignacio Comes Kinderknecht - 211.745-9

Mateo Fernandez Cruz - 209.986-0

Ariel Pérez - 153.715-5

TypeScript

TypeScript fue desarrollado por Microsoft y lanzado inicialmente en octubre de 2012. Anders Hejlsberg, conocido por su trabajo en Turbo Pascal, Delphi y C#, lideró el equipo detrás de TypeScript.

La motivación principal detrás de TypeScript fue mejorar la experiencia de desarrollo de JavaScript al agregarle tipado estático opcional y características propias de lenguajes orientados a objetos, como clases, interfaces y módulos. Esto permitió a los desarrolladores identificar errores comunes durante la compilación en lugar de en tiempo de ejecución, haciendo que el código sea más fiable y menos propenso a los bugs.

TypeScript se basa en JavaScript estándar y compila a código JavaScript limpio y legible, lo que significa que cualquier programa JavaScript válido también es un programa TypeScript válido. Esto facilita la adopción gradual y permite a los desarrolladores migrar proyectos existentes de JavaScript a TypeScript de manera incremental.

A medida que JavaScript se volvió universal en el desarrollo web, especialmente con el auge de aplicaciones web complejas, surgieron desafíos importantes. JavaScript es un lenguaje dinámico y flexible, pero su carencia de un sistema de tipado estático puede llevar a errores difíciles de detectar hasta la fase de ejecución. Esto es problemático en proyectos grandes y equipos colaborativos donde la robustez y la mantenibilidad son críticas.

· **Conceptos Clave de TypeScript**:

* **Tipado Estático Opcional**: TypeScript introduce la capacidad de definir tipos estáticos para variables, parámetros de función, propiedades de objetos, etc. Esto permite atrapar errores durante la fase de compilación, mejorando así la robustez del código.
* **Compatibilidad con ECMAScript**: TypeScript sigue de cerca las especificaciones de ECMAScript, integrando las características más recientes y proporcionando soporte para versiones anteriores de JavaScript. Esto permite a los desarrolladores escribir código más limpio y expresivo utilizando características como funciones de flecha, desestructuración, literales de plantilla, etc.
* **Organización del Código:** En JavaScript, organizar el código en archivos separados y gestionar las dependencias es un poco tedioso. Sin embargo, TypeScript proporciona soporte integrado para módulos y espacios de nombres para organizar mejor el código. Los módulos permiten encapsular el código en archivos separados, facilitando la gestión y el mantenimiento de grandes bases de código.
* **Sistema de Tipos Avanzado**: Estas características mejoran las capacidades de comprobación estática de tipos de TypeScript, permitiendo a los desarrolladores escribir código más robusto y expresivo. Los genéricos permiten escribir código reutilizable que puede funcionar con distintos tipos. Los genéricos son como marcadores de tipos que se determinan en tiempo de ejecución basándose en los valores pasados a una función o una clase. Las uniones e intersecciones se utilizan para componer tipos y crear relaciones de tipos más complejas. Las uniones permiten combinar dos o más tipos en uno sólo que puede tener cualquiera de los tipos combinados. Las intersecciones permiten combinar dos o más tipos en sólo uno que debe satisfacer todos los tipos combinados.

Desde su lanzamiento, TypeScript ha experimentado un crecimiento significativo en su adopción. Grandes empresas como Google, Facebook y Airbnb lo utilizan en sus proyectos. Angular, el popular framework de Google, se reescribió en TypeScript, lo que ayudó a aumentar su popularidad.

TypeScript es compatible con una amplia gama de herramientas de desarrollo, como Visual Studio Code, que proporcionan características avanzadas como IntelliSense y depuración integrada. La comunidad de TypeScript es activa y contribuye con bibliotecas y herramientas que enriquecen el ecosistema.

[BNF de Typescript](https://github.com/matefernandezcc/SSL-Public/blob/main/02-Sort/Typescript/TYPESCRIPT.bnf)

[Codigo del benchmark en TS](https://github.com/matefernandezcc/SSL-Public/tree/main/02-Sort)

Cobol

La historia de COBOL (Common Business-Oriented Language) es notable por ser uno de los primeros lenguajes de programación de alto nivel y por su duradera influencia en la informática empresarial.

Fue diseñado en la década de 1950 por un consorcio de organizaciones gubernamentales y empresariales llamado Conferencia sobre Lenguajes de Sistemas de Datos (CODASYL). Derivado en parte de FLOW-MATIC, un lenguaje creado por la pionera de la informática, la Dra. Grace Hopper, COBOL se creó como parte de una iniciativa del Departamento de Defensa de EE. UU. que buscaba un lenguaje de programación que pudiera funcionar en distintos sistemas operativos (Linux, Windows, Unix, z/OS, etc.) y entornos de hardware.

La primera versión del lenguaje se publicó en 1960. Esta versión inicial ya incluía muchas de las características que lo hicieron popular entre las empresas, como la capacidad de procesar datos en formato tabular y la gestión de archivos.

Durante las décadas de 1960 y 1970, el COBOL se convirtió en el lenguaje predominante para el desarrollo de aplicaciones comerciales y gubernamentales, especialmente en sectores como la banca, las finanzas, los seguros y la administración pública. Su popularidad se debió en parte a su diseño orientado a negocios y su capacidad para manejar grandes volúmenes de datos de manera eficiente.

Aunque el COBOL ha sido criticado por su sintaxis verbosa y su estructura rígida en comparación con lenguajes más modernos, ha demostrado una notable capacidad de adaptación y modernización.

Muchas empresas han optado por modernizar sus sistemas existentes escritos en COBOL en lugar de reescribirlos por completo, aprovechando herramientas y técnicas para integrar nuevas tecnologías sin interrumpir las operaciones críticas.

El programa COBOL tiene una estructura jerárquica que comprende divisiones, secciones, párrafos, sentencias, verbos y cadenas de caracteres. La naturaleza divisional de un sistema COBOL (que consta de cuatro divisiones) permite una clara separación de las preocupaciones dentro de los programas COBOL.

COBOL fue finalmente estandarizado como lenguaje informático en 1968, tras lo cual los programadores de COBOL implementaron varias revisiones y modernizaciones, incluyendo COBOL-61, COBOL-68, COBOL-74 y COBOL-85. La iteración más reciente, COBOL 2002, intenta hacer que las aplicaciones COBOL sean más compatibles con las prácticas modernas de desarrollo de software mediante la introducción de características orientadas a objetos y otros paradigmas de programación avanzados en el lenguaje.

Conceptos clave de cobol

* Lenguaje estandarizado: la estandarización del lenguaje permitirá que se pueda compilar y ejecutar en máquinas IBM y otro tipo de computadoras
* Enfocado a negocios: el lenguaje estaba pensado para la automatización de procesos para las entidades financieras y además de su capacidad de trabajar con grandes volúmenes de datos y archivos grandes. Por esta razón los bancos, la defensa y empresas relacionadas con el sector financiero lo utilizaban
* Lenguaje Estructurado: el lenguaje tiene una estructura jerárquica formado por divisiones, secciones, párrafos, sentencias, verbos y cadenas de caracteres. a los usuarios se les hace fácil depurar el proyecto

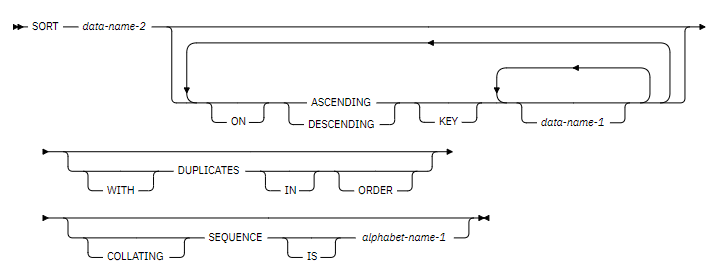
[BNF de cobol](https://github.com/matefernandezcc/SSL-Public/blob/main/02-Sort/Cobol/COBOL.bnf)

BNF completa de Cobol:  
<https://linguagemcobol.wordpress.com/bnf/>

Sort en Cobol

La sentencia SORT sirve par ordenar registros por un campo clave que le indiquemos, se puede elegir entre varias claves y definir si el orden será ascendente o descendente

se utiliza el siguiente formato para usar la sentencia SORT



### **1. SORT data-name-2**

* data-name-2 es el nombre del archivo de trabajo (SD file) que se utiliza para el proceso de ordenamiento. Es el archivo donde se almacenarán temporalmente los registros durante la operación de ordenamiento.

### **2. Claves de Ordenación (KEY)**

* **ON ASCENDING/DESCENDING KEY data-name-1**:
  + Aquí es donde se especifican las claves por las que se ordenarán los registros.
  + **ASCENDING**: Ordena los registros en orden ascendente (menor a mayor).
  + **DESCENDING**: Ordena los registros en orden descendente (mayor a menor).
  + **data-name-1**: Es el campo o campos del registro por los cuales se ordenarán los datos. Puede haber más de un campo especificado.

### **3. Duplicados (WITH DUPLICATES IN ORDER)**

* **WITH DUPLICATES IN ORDER**:
  + Indica que los registros con valores de clave duplicados deben mantenerse en el orden en el que se encuentran originalmente en el archivo de entrada. Si no se especifica, el orden de los duplicados puede no ser garantizado.

### **4. Secuencia de Colación (COLLATING SEQUENCE IS)**

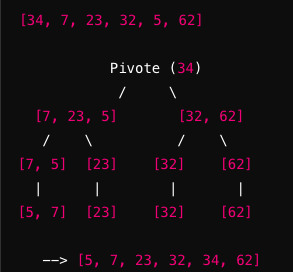
* **COLLATING SEQUENCE IS alphabet-name-1**:
  + Define una secuencia de colación personalizada para la comparación de claves durante el ordenamiento.
  + **alphabet-name-1**: Especifica un conjunto de reglas de colación, que pueden variar dependiendo del idioma o el conjunto de caracteres que se esté utilizando. Si no se especifica, se usa la secuencia de colación estándar del sistema.

### **5. Ubicación de la Sentencia SORT**

* **Colocación en el Código**:
  + La nota en la imagen indica que las sentencias SORT de formato 2 pueden aparecer en cualquier lugar dentro de la PROCEDURE DIVISION. Esto significa que la operación de ordenamiento puede ser invocada en diferentes puntos del flujo del programa.

Sort en Typescript

El algoritmo exacto que se usa depende del motor de JavaScript subyacente (por ejemplo, V8 en Chrome, Node.js). Por lo general, se utiliza una combinación de algoritmos eficientes como *QuickSort*, *MergeSort* o *TimSort*.



### **Explicación del Algoritmo QuickSort:**

1. **Seleccionar un pivote**: En cada paso del algoritmo, se selecciona un elemento como "pivote". Este pivote puede ser cualquier elemento del array, aunque comúnmente se elige el elemento central, el primero o el último.
2. **Dividir y conquistar**:
   * **Partición**: Se reorganizan los elementos en el array de tal manera que todos los elementos menores que el pivote queden a la izquierda de él, y todos los mayores a la derecha.
   * **Recursión**: Se aplica el mismo proceso recursivamente a los sub-arrays de la izquierda y derecha del pivote.
3. **Combinar**: Una vez que todos los sub-arrays están ordenados, se combinan para formar el array final ordenado.

Fuentes

<https://tomcopeland.blogs.com/cobol.html>

<https://thepower.education/blog/todo-sobre-cobol-el-lenguaje-de-programacion>

<https://www.ibm.com/es-es/topics/cobol>

<https://www.ibm.com/docs/es/cobol-linux-x86/1.2?topic=statements-sort-statement>

GitHub

https://github.com/matefernandezcc/SSL-Public/tree/main/02-Sort