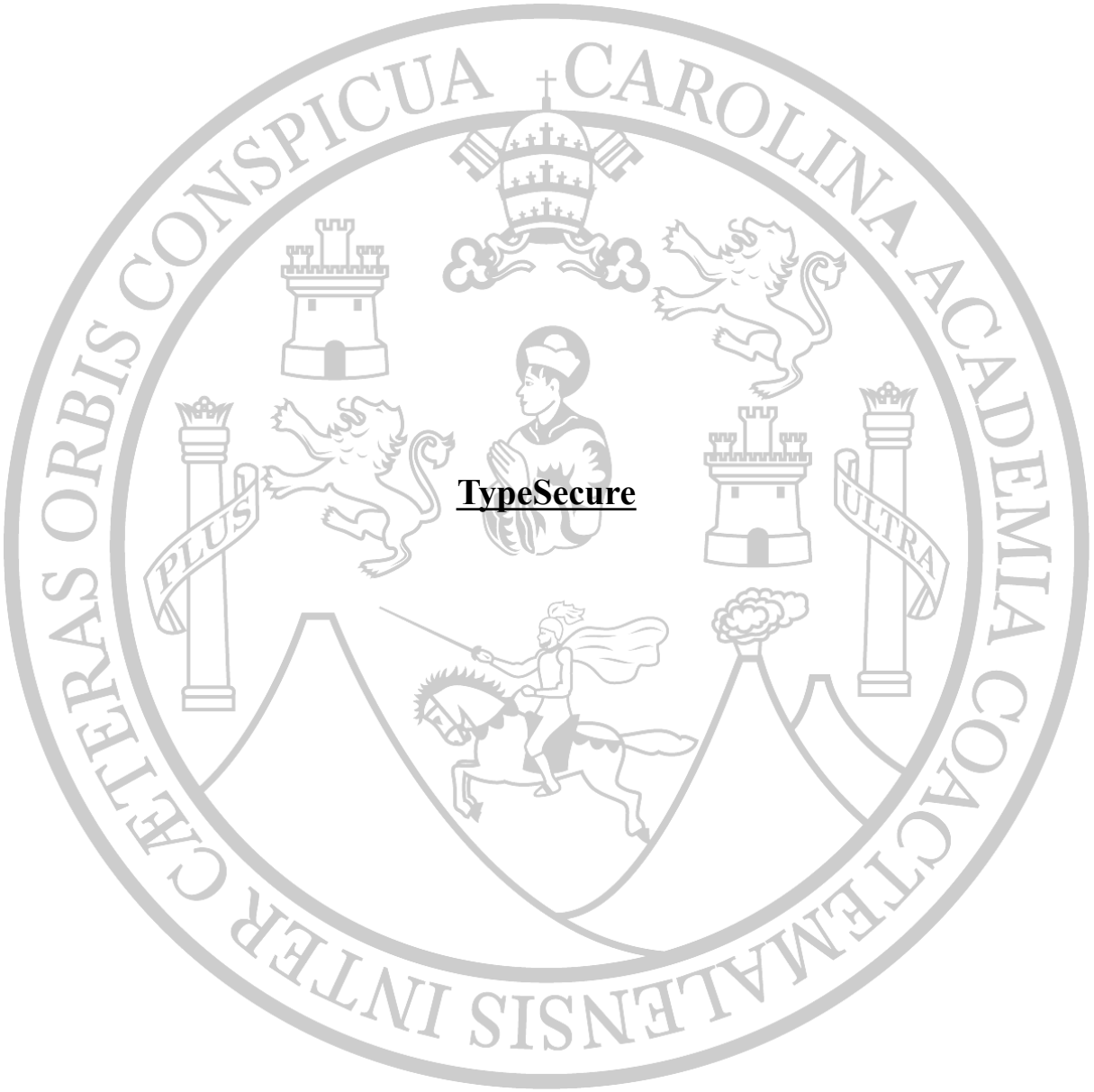
Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División Ciencias de la Ingeniería
Organización de lenguajes y compiladores 1



TypeSecure

Nombre:
Pedro Ricardo Gordillo González

Registro Académico:
202031683

Quetzaltenango, mayo de 2023

Manual de usuario

Requisitos del sistema

Requerimientos de hardware

- Equipo, teclado, mouse, monitor.
- Espacio en disco: 124 MB para JRE; 2 MB para Java Update
- RAM: 128 MB
- Procesador: Mínimo Pentium 2 a 266 MHz
- Exploradores: Internet Explorer 9 y superior, Firefox o el de su preferencia

Requerimientos de software

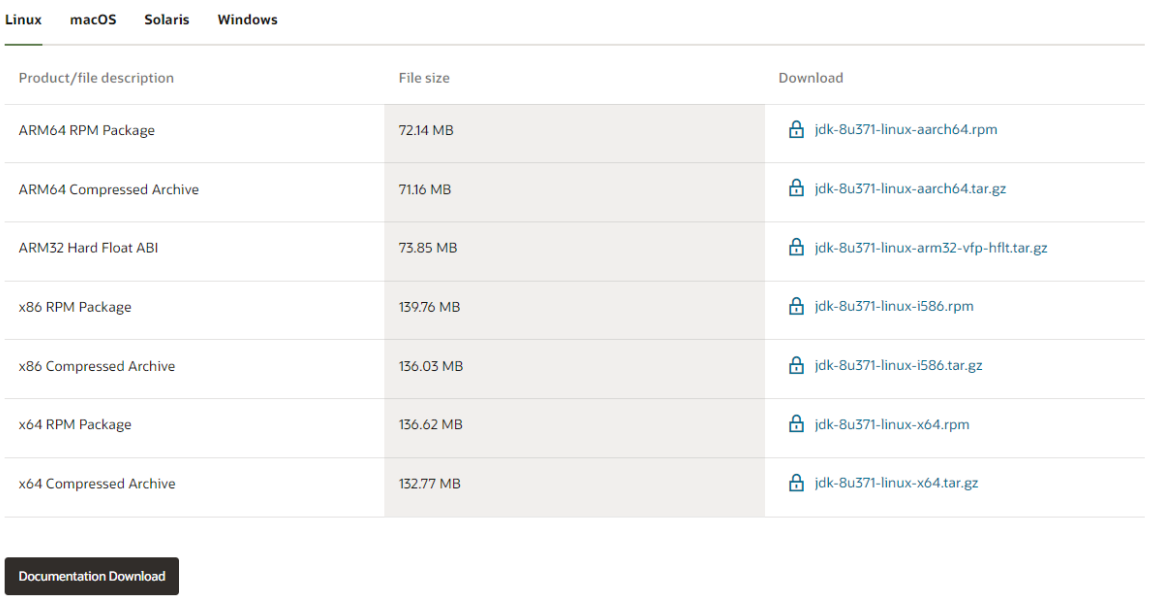
Sistema operativo

- Windows (Windows 7 en adelante), Mac OS X, o Linux

- Instalación JDK Java

Para ejecutar la aplicación se necesita tener instalado un jdk de java. A continuación hay una guía de instalación.

Ingresa al siguiente link:
https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html



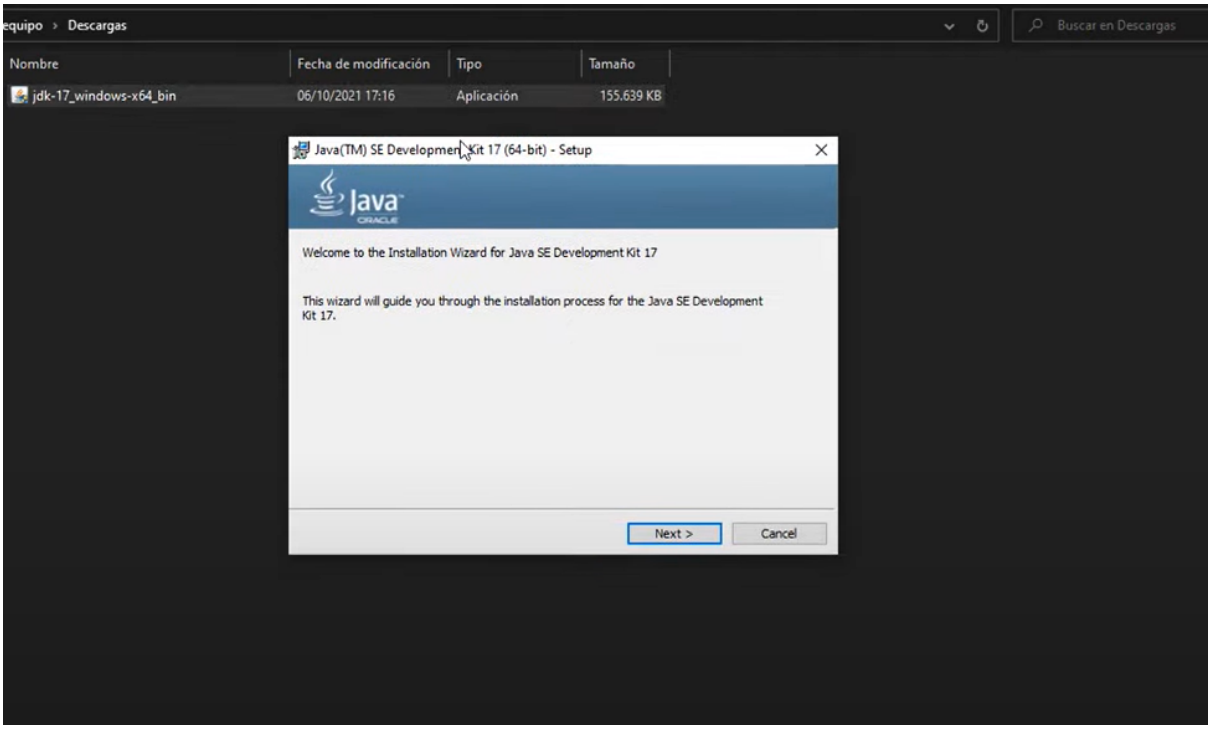
Linux macOS Solaris Windows

Product/file description	File size	Download
ARM64 RPM Package	72.14 MB	jdk-8u371-linux-aarch64.rpm
ARM64 Compressed Archive	71.16 MB	jdk-8u371-linux-aarch64.tar.gz
ARM32 Hard Float ABI	73.85 MB	jdk-8u371-linux-arm32-vfp-hflt.tar.gz
x86 RPM Package	139.76 MB	jdk-8u371-linux-i586.rpm
x86 Compressed Archive	136.03 MB	jdk-8u371-linux-i586.tar.gz
x64 RPM Package	136.62 MB	jdk-8u371-linux-x64.rpm
x64 Compressed Archive	132.77 MB	jdk-8u371-linux-x64.tar.gz

Documentation Download

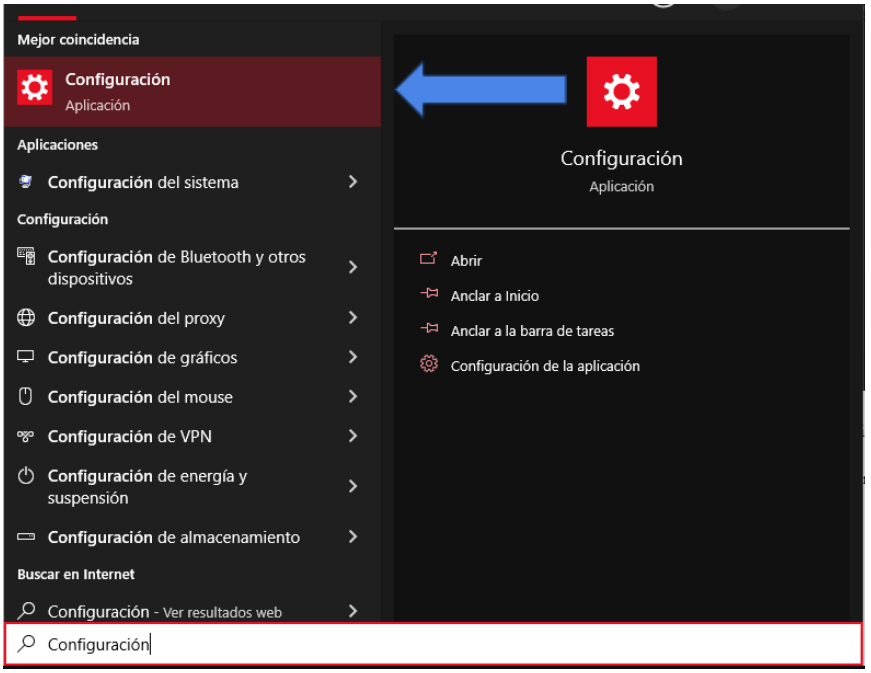
Descarga la versión que se adapte al sistema operativo que utiliza.

Ejecuta la aplicación que se descargó, en este caso se realiza en un sistema operativo windows.

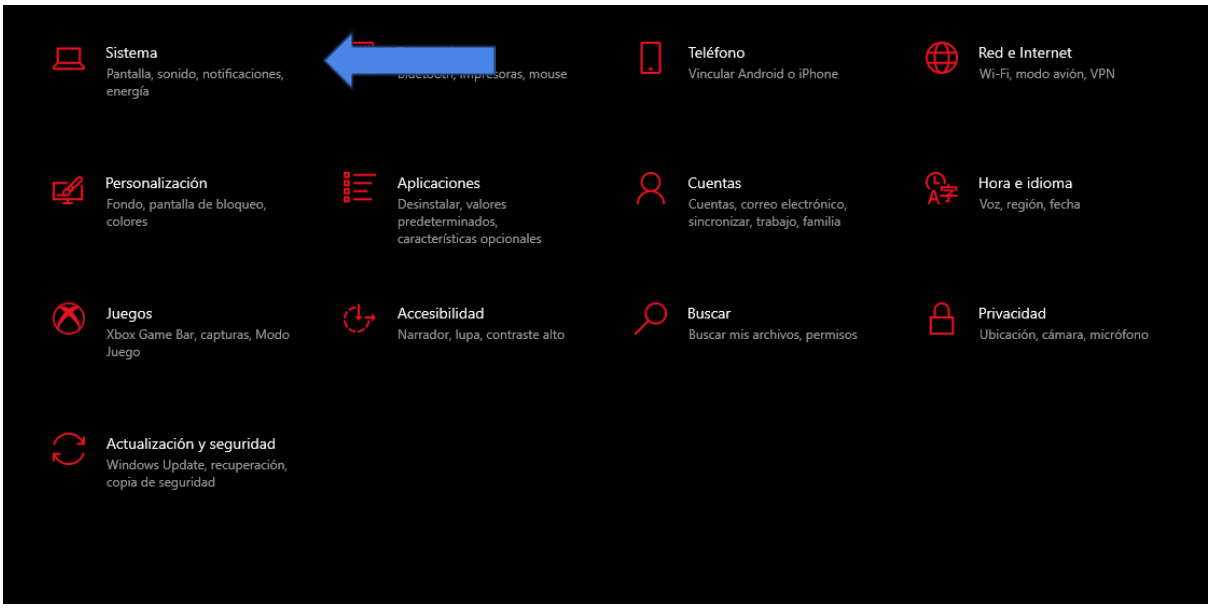


Dar click en el botón “Next” y sigue los pasos que se indican en la ventana de instalación.

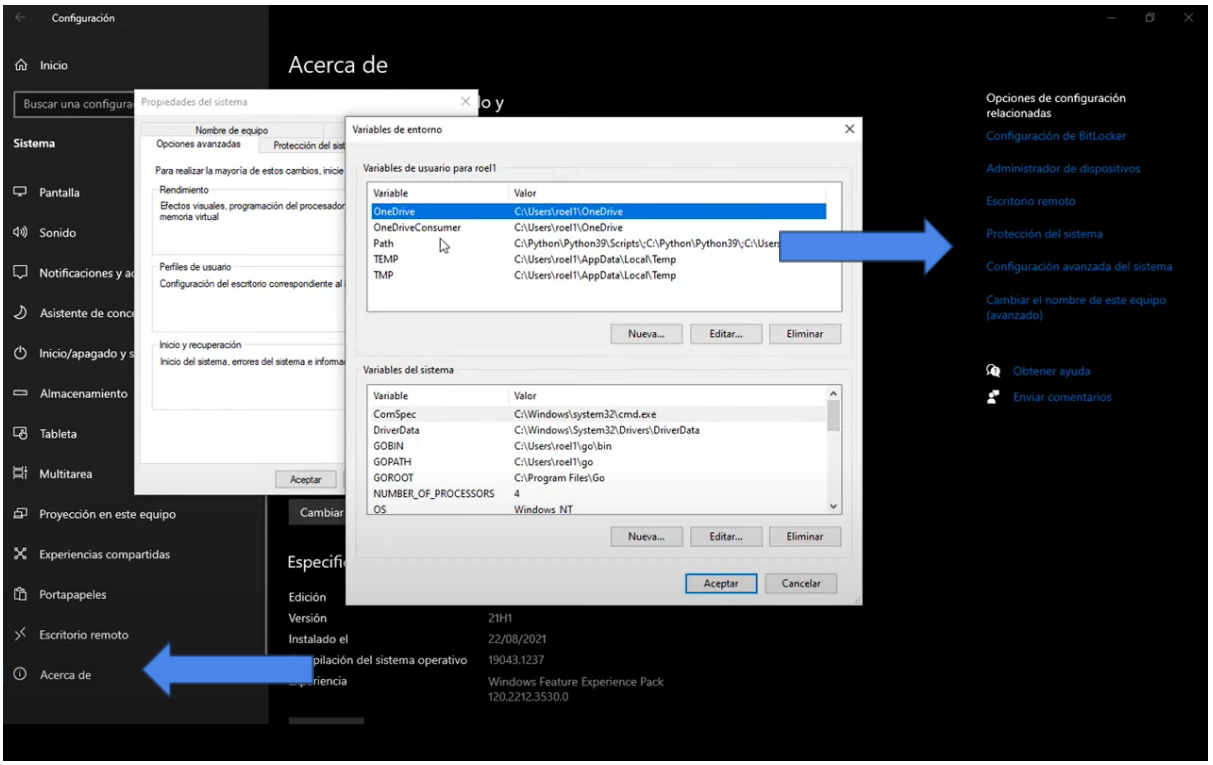
Para configurar la variable de entorno, presiona la tecla Windows y coloca “Configuración” y selecciona la primera opción.



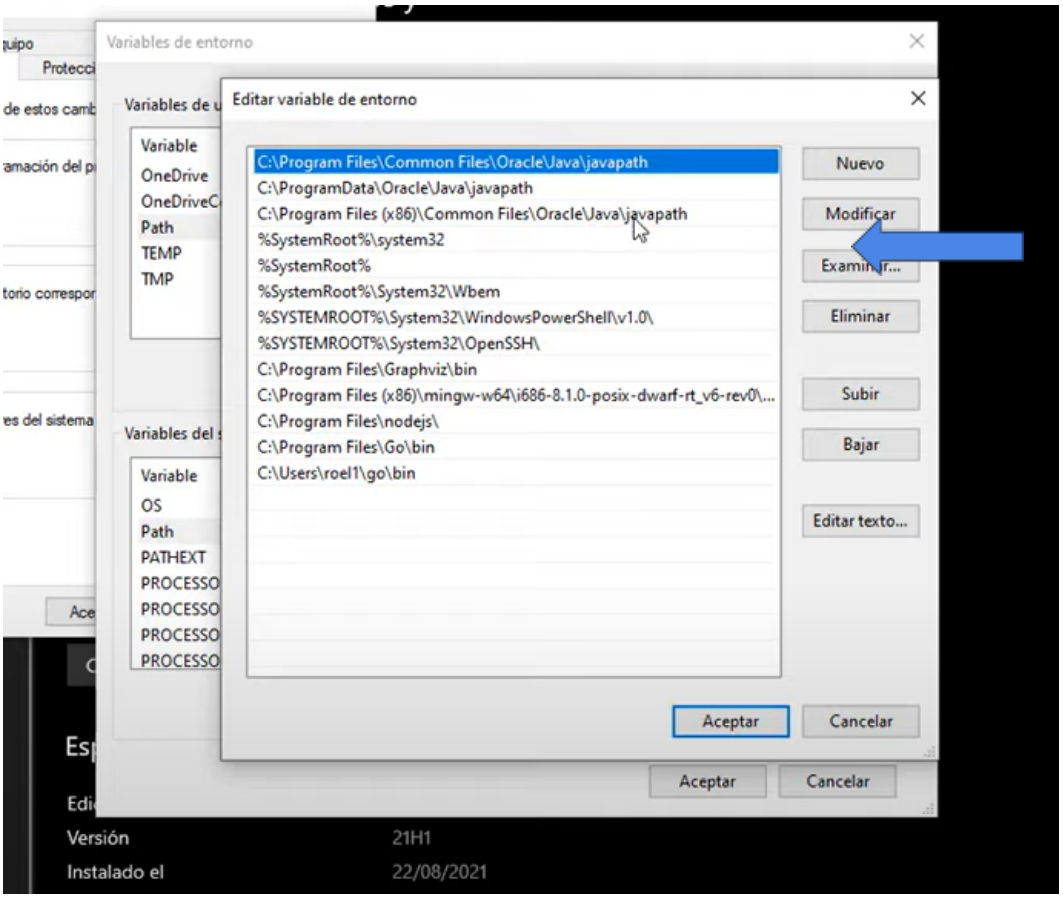
Ingresa a la opción de Sistema.



Ingresa a la opción de “Acerca de”, luego selecciona la opción de “Configuración avanzada del sistema”, y cuando se muestre una ventana, selecciona la opción de “Variables de entorno”.

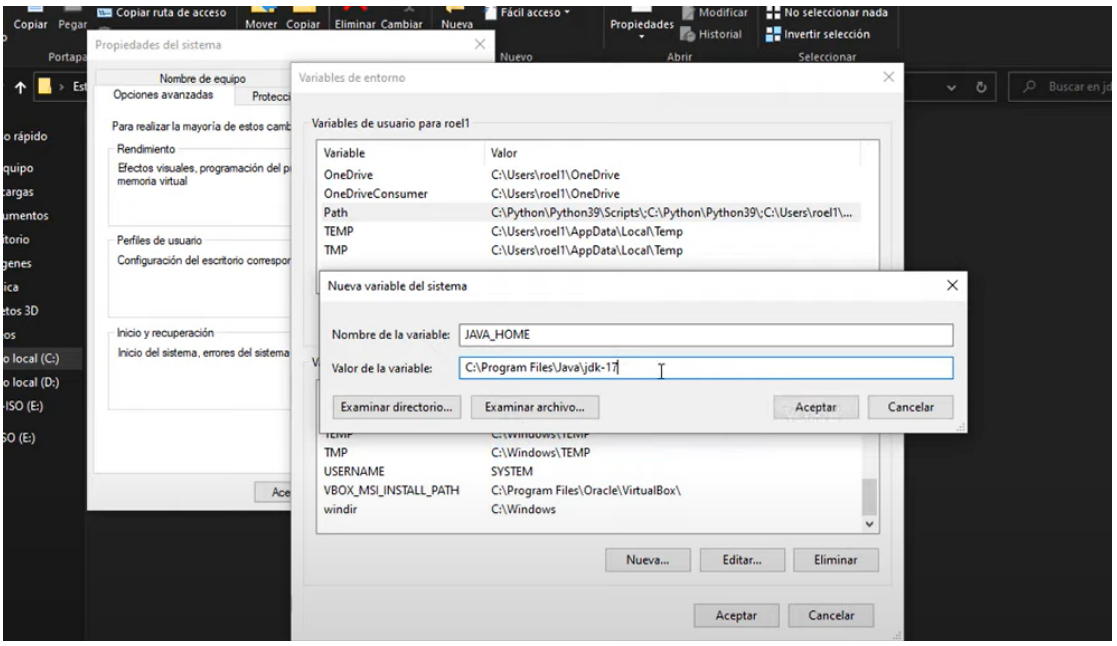


Luego en la parte de variables del sistema, busca la opción de Path y da doble click.
Si aparece el path de java, no habría que realizar más.



Si no aparece, debes agregar el path. Ve a donde instalaste java, comúnmente se instala en el disco local, copia el link, de la carpeta de los binarios. Por ejemplo, esta sería la ruta si java se instaló en el disco local C, “C:\Program Files\Java\jdk-18.0.2.1\bin”.
Al tener el path, le debes dar click a la opción de “Nuevo” y agregar el path, a continuación dar en “Aceptar”.

Luego de ello se debe de agregar el “Java Home”, para ello debes dar click en “Nuevo”, luego en la parte de “Nombre de la variable” ingresar el nombre “JAVA_HOME” y en el valor de la variable debes agregar el path de la ruta donde se instalo java, por ejemplo, este sería el path a agregar, “C:\Program Files\Java\jdk-18.0.2.1”.



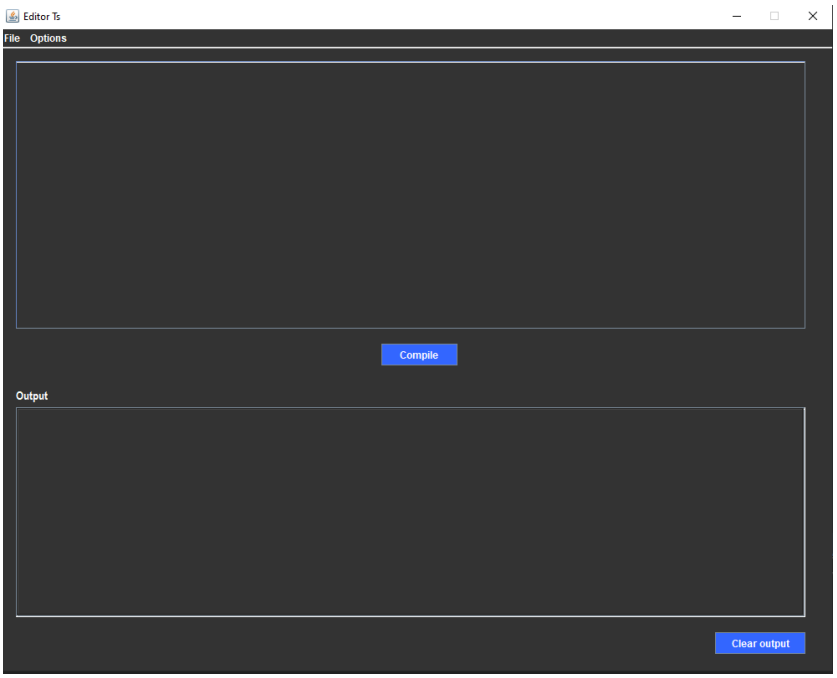
Con esto ya se tendría instalado el JDK de Java.

Ejecución de la aplicación

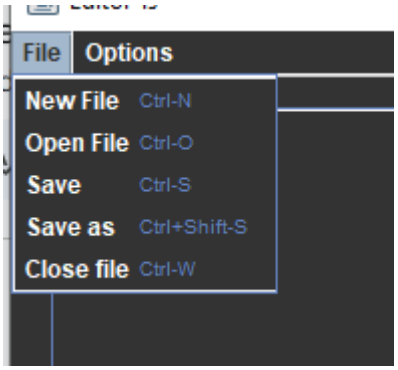
Para ejecutar la puede hacerse por medio de consola o haciendo doble clic en el jar.

Para realizar la ejecución por medio de consola se ingresa a la carpeta donde esta el archivo jar y se ejecuta el siguiente comando:
“java -jar TypeSecure-1.0-SNAPSHOT-shaded.jar”

Al ejecutar este comando se mostrará la interfaz gráfica de la aplicación.

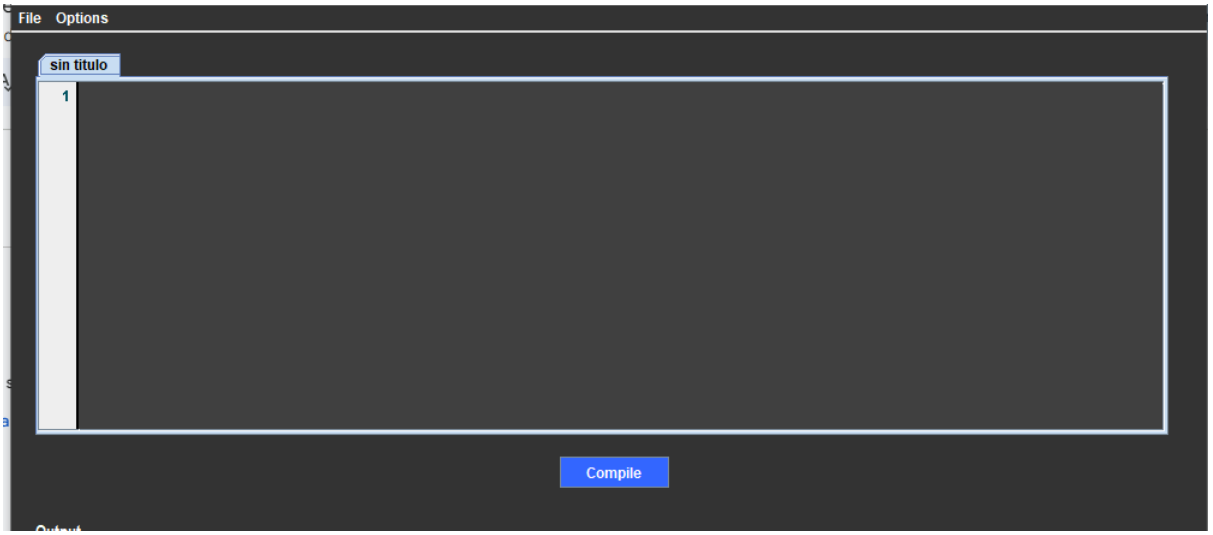


Las opciones disponibles son las siguientes.



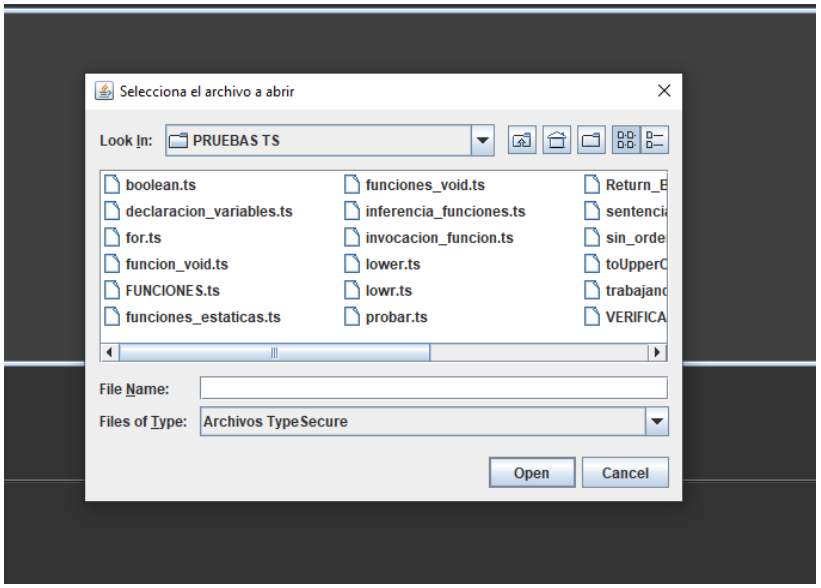
New file

Esta opción abre una nueva pestaña para ingresar nuevo código a compilar.

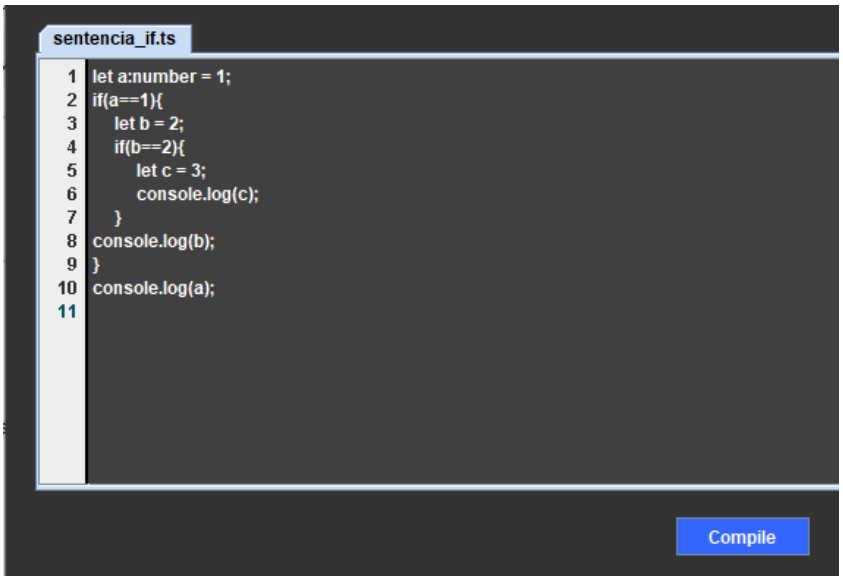


Open file

Esta opción muestra una pestaña para cargar un archivo guardado desde el dispositivo.

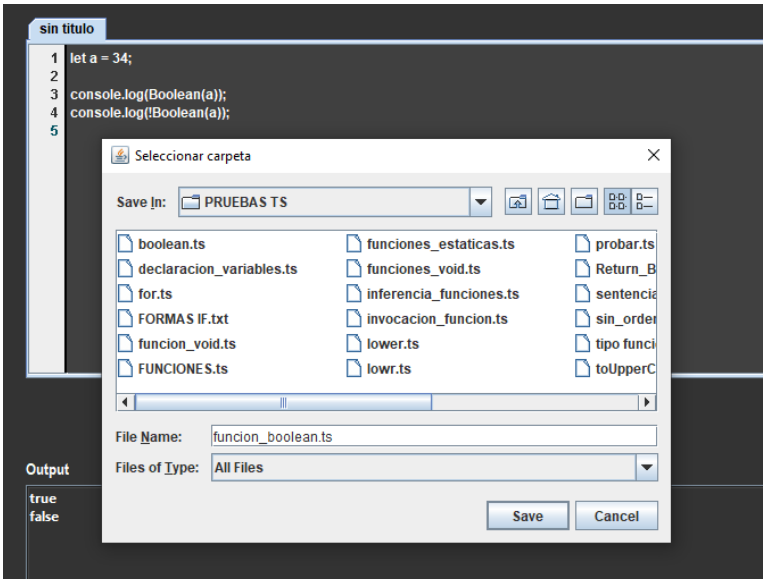


Al seleccionar un archivo el contenido se muestra en una pestaña del editor.



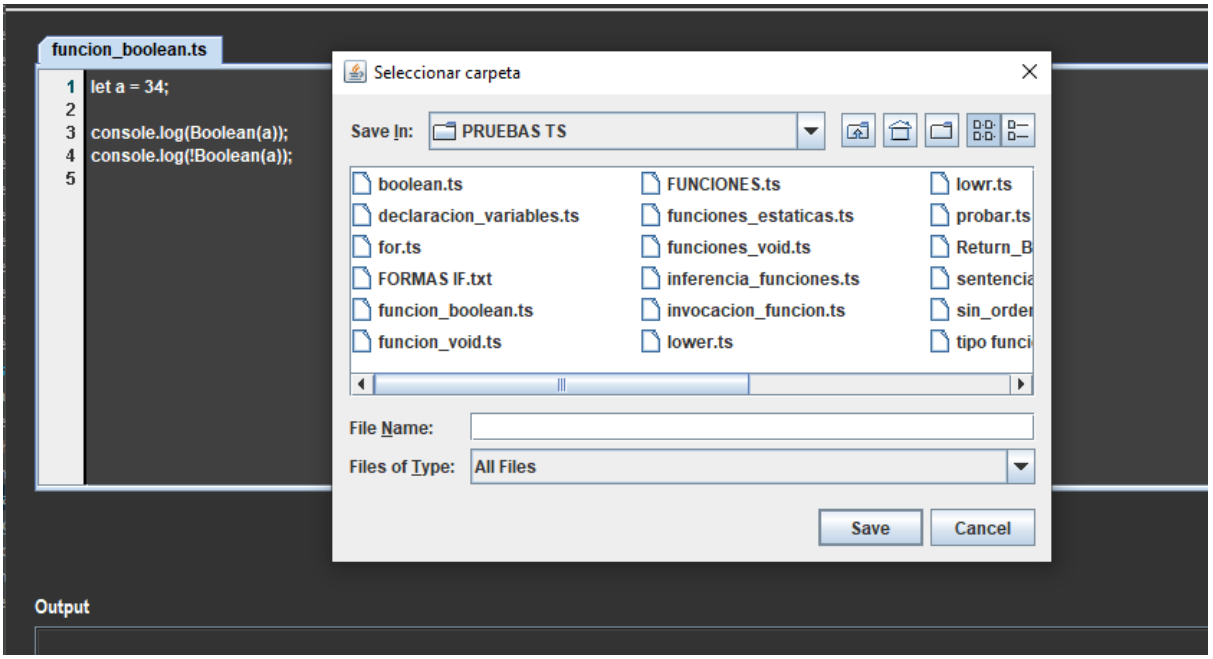
Save

Esta opción guarda el archivo, si el archivo ya existe guarda los cambios, si no existe, crea un nuevo archivo en el dispositivo.



Save as

Esta opción guarda un nuevo archivo en el dispositivo, exista o no el archivo en el que se está trabajando.



Close file

Esta opción cierra la pestaña que está seleccionada.

Compile

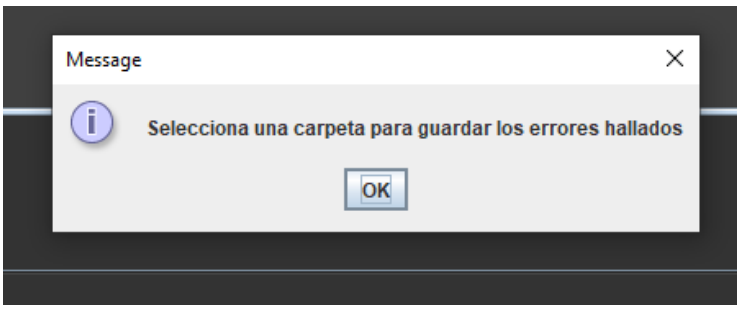
Al dar click al botón “Compilar” se realiza la compilación del código que se ingresó al área de texto.

Clear output

Al dar click al botón “Clear output” se borra la salida de código que se tenga en la consola.

Errores

Cuando hay errores léxicos, sintácticos o semánticos se genera un archivo en formato html. En ese caso se muestra una ventana indicando que se seleccione una carpeta.



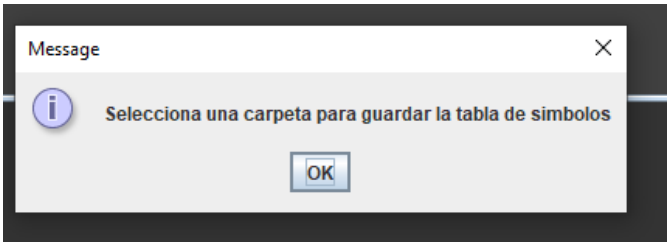
Al seleccionar la carpeta el archivo html generado contendrá una descripción de los errores.

Errores hallados

Tipo de Error	Lexema	Linea	Columna	Descripcion
Lexico	?	3	21	No se encuentra definido en la gramatica
Sintactico	)	3	22	No se esperaba este token
Sintactico	)	3	22	No se esperaba este token

Tabla de símbolos

Cuando en el código se ejecuta la instrucción “getSymbolTable()” se genera un archivo html describiendo la tabla de símbolos en ese momento.



Se debe seleccionar una carpeta y dentro de ella estará el documento html que contiene una descripción de la tabla de símbolos.

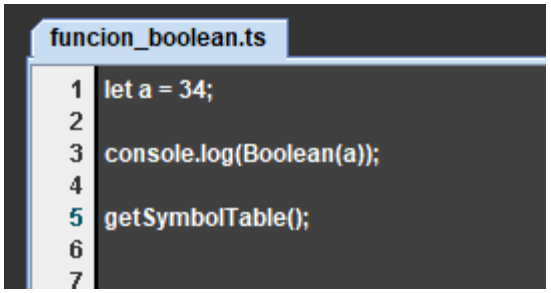


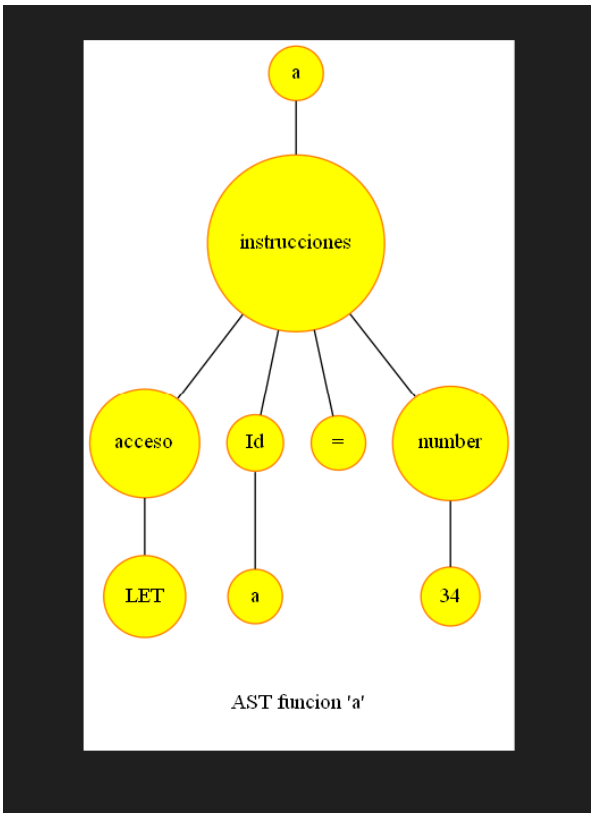
Tabla de simbolos

Funcion	Id	Tipo	Valor	Linea de declaracion	Columna de declaracion
NO	a	NUMBER	34.0	1	1

Gráfica ast (Abstract Symbol Tree)

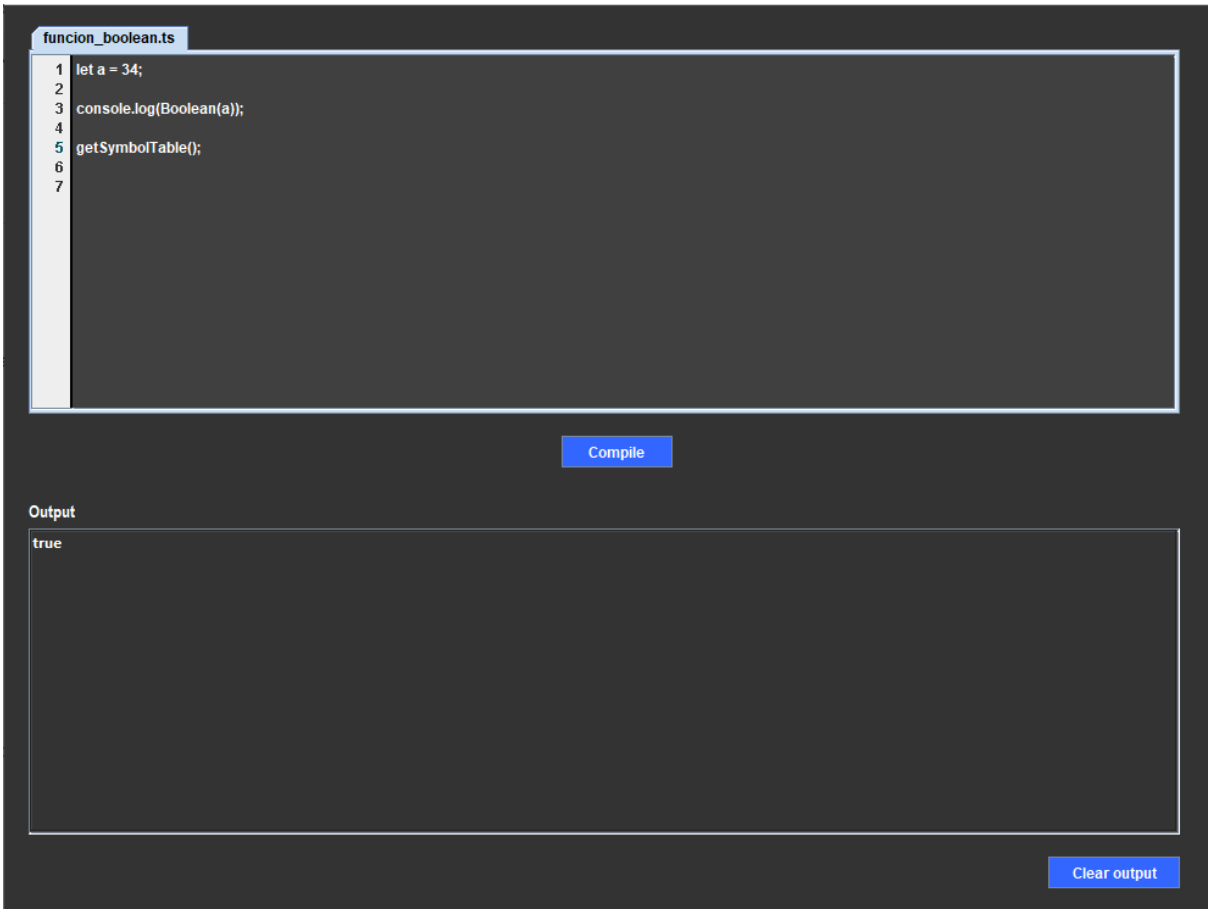
Cuando en el código se ejecuta la instrucción “graficarAst(name)” se genera una imagen en formato jpg donde se muestra el ast de la función pasada por nombre a la función “graficarAst”.

Se graficara el ast de la función de la función tenga el nombre que se pasa por parámetro en la función printAst.



Output

En este apartado se muestran las instrucciones que se ejecutan con la instrucción “console.log”.



Lenguaje TypeSecure

Tipos de Datos

Tipo de dato	Palabra reservada	Descripción
Number	number	Valores de coma flotante de 64 bits de precisión “double”.
BigInt	bigint	Valores enteros de 64 bits
String	string	Representa una secuencia de caracteres unicode, entre comillas dobles o simples.
Boolean	boolean	Representa valores lógicos (true o false)
Void	void	Se utiliza en tipos de devolución de funciones, para todas aquellas que no devuelven ningún valor.
Undefined	undefined	Denota el valor dado a todas las variables no inicializadas.

Operaciones aritméticas

Operador	Descripción	Ejemplo
+	Suma, retorna la suma de los operandos	a + 1
-	Resta, retorna la diferencia de los operandos	b - 3
*	Multiplicación, retorna el producto de los operandos	a * 123
/	División, ejecuta la división y devuelve el cociente	a / 100
%	Módulo, ejecuta la división y devuelve el resto.	a % b
++	Incrementa el valor de la variable en 1	a++
--	Decrementa el valor de la variable en 1	b--
()	Paréntesis	a * (b + 1)

Operadores Relacionales

Operador	Descripción
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual a
!=	Distinto de ó no igual que

Operadores Lógicos

Operador	Descripción	Ejemplo
&&	El operador AND devuelve verdadero si todas las expresiones evaluadas son verdaderas.	(a > 10 && b > 10) es ***false***
||	El operador OR devuelve verdadero si alguna de las expresiones evaluadas es verdadera.	(a > 10 || b > 10) es ***true***
!	El operador NOT devuelve el inverso de la expresión evaluada	!(a > 10) es ***true***

Operador de concatenación (+):

El operador de concatenación es aplicable cuando se quiere unir dos strings. Pero también es aplicable cuando al menos uno de los operadores es de tipo string.

Función Number
Valores de otros tipos pueden ser convertidos al tipo *number* usando la función *Number()*, esta función se puede resumir como sigue:

- Datos de tipo *number*, son retornados tal cual.
- Datos de tipo *bigint*, son convertidos a datos de tipo *number*.
- Para las expresiones u operaciones de tipo *boolean*: *true* es convertido a 1; *false* a 0.
- *Strings* se convierten analizandolas y verificando que contengan un número literal (number literal). Aspectos a tomar en cuenta para esto:
    - Espacios en blanco al inicio y al final son ignorados.
    - Los símbolos + y - están permitidos al comienzo de la cadena para indicar el signo. Sin embargo, el signo sólo puede aparecer una vez y no debe ir seguido de un espacio en blanco.
    - Cadenas vacías y espacios en blanco son convertidos a cero.

Función BigInt
Valores de otros tipos pueden ser convertidos al tipo *bigint* usando la función *BigInt()*, que puede resumirse como sigue:

- Valores u operaciones de tipo *bigint*, son retornados tal cuál.
- Valores u operaciones booleanas con valor *true* son convertidas a 1n y con valor *false* a 0n.

- Valores de tipo *number* son convertidos a *bigint*, y cuando estos tengan un valor decimal se debe de lanzar un error indicando que no es posible la conversión por "pérdida de precisión", además de indicar la fila y columna de la instrucción.
- *Strings* se convierten verificando que contengan un *bigint literal*, es decir:
    - Los espacios al inicio y al final son ignorados.
    - Espacios en blanco al inicio y al final son ignorados.
    - Los símbolos + y - están permitidos al comienzo de la cadena para indicar el signo. Sin embargo, el signo sólo puede aparecer una vez y no debe ir seguido de un espacio en blanco.
    - Cadenas vacías y espacios en blanco son convertidos a cero.

Función Boolean

Valores de otros tipos pueden convertirse a tipo *boolean*. La función *Boolean()* se puede resumir de la siguiente manera:

- Valores u operaciones booleanas son retornadas tal cual.
- *undefined* o variables que no tengan un valor definido son retornadas como *false*.
- Los valores 0 y -0 son convertidos a *false*, otros valores de tipo *number* toman el valor de *true*.
- Los valores 0n y -0n son convertidos a *false*, otros valores de tipo *bigint* toman en valor de true.
- Cadenas vacías son convertidas en *false* y el resto de cadenas son convertidas en *true*.

Función String

La función *String()*, se puede resumir de la siguiente manera:

- Las cadenas son regresadas tal cual.
- Variables sin valor o el valor *undefined*, es transformada a "undefined".
- Los valores *true* y *false* son convertidos a *"true"* y *"false"*.
- Los valores de tipo *number*, son convertidos a cadenas y el mismo caso para las variables o valores de tipo *bigint*.

Métodos y propiedades del tipo de dato *string*

length: *number*

Indica el número de unidades de código UTF-16 en la cadena (longitud de la cadena).

charAt(index: number): string

Devuelve el carácter en la posición especificada en la cadena, si *index* es menor que cero o más grande que la longitud de la cadena, se deberá lanzar un error de ejecución: "índice fuera de los límites", indicando el número de fila y columna donde surge el error.

toLowerCase(): *string*, toUpperCase(): *string*

Devuelve la cadena en minúsculas o en mayúsculas, respectivamente.

concat(str: string): string

Combina el texto de dos cadenas y devuelve una nueva cadena.

Instrucción console.log

Su función es imprimir en la consola el valor o valores u operaciones separados por coma que se le pasan como argumentos.

Instrucción if, else if, else

Una instrucción if puede incluir una o más expresiones que devuelven un valor booleano, si el valor booleano es verdadero, se ejecutan el conjunto de sentencias en el cuerpo de la instrucción if.

Instrucción for Loop
Se utiliza para ejecutar un bloque de código un número determinado de veces, que se especifica mediante una condición.

Instrucción While
El ciclo while ejecuta un conjunto de instrucciones cada vez que la condición especificada se evalúa como verdadera. En otras palabras, el ciclo evalúa la condición antes de que se ejecute el bloque de código.

Instrucción Do While
El ciclo Do While es similar al ciclo While, excepto que la condición se da al final del ciclo. El bucle Do While ejecuta el bloque de código al menos una vez antes de verificar la condición especificada. Para el resto de las iteraciones, ejecuta el bloque de código sólo si se cumple la condición especificada.

Instrucción break
La instrucción break se usa para terminar o tomar el control de un ciclo. El uso de break hace que la ejecución del programa salga del ciclo en el cual se encuentra, es decir que esta instrucción es sólo permitida dentro de los loops o bucles for, while y do-while.

Instrucción continue
La instrucción ***continue*** salta las instrucciones subsiguientes en la iteración o ciclo en el que se encuentra y lleva el control de la ejecución al comienzo del ciclo. A diferencia de la sentencia ***break***, ***continue*** no sale del ciclo, sino que termina la interacción actual y comienza la iteración subsiguiente. Es importante aclarar que esta sentencia es permitida únicamente dentro de los loops for, while y do-while.

Propiedades matemáticas estáticas presentes en TypeSecure

Math.E: *number*
Devuelve el número Euler que es la base de los logaritmos naturales, aproximadamente 2.718

Math.PI: *number*
Devuelve el número PI, aproximadamente 3.141592

Math.SQRT2: *number*
Raíz cuadrada del número 2, aproximadamente 1.414

Funciones estáticas presentes en TypeSecure

Math.abs(value: number): number
Valor absoluto del valor pasado como argumento.

Math.ceil(value: number): number
Devuelve el entero más pequeño mayor o igual que *value*.

Math.cos(value: number): number
Devuelve el coseno de value en radianes.

Math.sin(value: number): number
Devuelve el seno de value en radianes.

Math.tan(value: number): number
Devuelve la tangente de value en radianes.

Math.exp(value: number): number
Devuelve "*e* elevado a *value"*, donde *value* es el argumento y *e* es el número de Euler (2,718…, la base del logaritmo natural).

Math.floor(value: number): *number*
Devuelve el entero más grande menor o igual que *value*.

Math.pow(base: number, potencia: number): number
El método estático Math.pow() devuelve el valor de la *base* elevada a la *potencia*.

Math.sqrt(value: number): *number*
Devuelve la raíz cuadrada de *value*.

Math.random(): *number*
El método estático *Math.random()* devuelve un número *pseudoaleatorio* de punto flotante que es mayor o igual que 0 y menor que 1.

Comentarios
Existen comentarios de línea y comentarios multilínea:

```
// comentarios de línea

/*
   Comentarios multilínea
*/

/* otro comentario multilínea */
```

Función printAst(function_name: string) : *void*
La función *printAst()* recibirá el nombre de una función e imprimirá o generará una imagen en formato png o similar, con el ast del cuerpo de dicha función.

Función getSymbolTable(): *void*
La función *getSymbolTable()* generará una archivo en formato html con la tabla de símbolos en el contexto en donde se invoque la función, se deberá incluir el nombre de cada variable o función a la cual la tabla de símbolos tiene acceso, así como su valor actual o tipo de retorno y la línea y columna donde fue declarada la variable o función.