



**TECNOLÓGICO
NACIONAL DE MÉXICO**



INSTITUTO TECNOLÓGICO DE VERACRUZ

*Guía para uso de un sistema de análisis léxico y sintáctico
propio para un lenguaje*

Nombre de la materia:

Lenguajes y Autómatas I

Alumnos:

Canseco López Danna Paola

Balderas Morales Odalis

Magaña Barocio Rodrigo

Docente:

Argüelles Lucho Primavera

Grupo:

4J1A 15:00-16:00

Índice

Índice de tablas	3
Índice de ilustraciones	3
Introducción	5
Objetivo	5
Requerimientos	5
Instalación jdk 11	5
Instalación de javacc	11
Descripción general	12
Definiciones técnicas	12
Tablas de símbolos	13
Palabras Reservadas	13
Operaciones aritméticas	14
Operaciones de comparación	14
Asignación	14
Delimitadores	15
Implementación del código	15
Inicio y Fin	15
Declaración de variables	15
Inicialización de variables	16
Impresión de datos	16
Entrada de datos	16
Uso de operaciones aritméticas	17
Uso de la condicional if	17
Uso de la condicional elseif	17
Uso del ciclo while	18
Uso del ciclo do while	18
Uso de and, or y not	19
Declaración de arreglos	19
Declarar y llamar a un método	19
Ejecución del programa	19
Tipos de errores	21
Conclusión	23
Bibliografía	24

Índice de tablas

Tabla 8-1 Palabras Reservada	14
Tabla 8-2 Operaciones Aritméticas	14
Tabla 8-3 Operaciones de comparación	14
Tabla 8-4 Asignación	14
Tabla 8-5 Delimitadores	15
Tabla 11-1 Error léxico	22
Tabla 10-2 Errores sintácticos	23

Índice de ilustraciones

Ilustración 1 Navegador de muestra	5
Ilustración 2 Búsqueda de java jdk en navegador	6
Ilustración 3 Selección de la página indicada	6
Ilustración 4 Descarga de ejecutable en Windows	6
Ilustración 5 Aceptar término de licencia Oracle	7
Ilustración 6 Inicio de sesión en Oracle para ejecutable	7
Ilustración 7 Inicio de descarga de jdk	7
Ilustración 8 Ventana inicial de instalador de jdk	8
Ilustración 9 Ubicación de instalación de jdk	8
Ilustración 10 Inicio de proceso de instalación	9
Ilustración 11 Finalización de instalación	9
Ilustración 12 Búsqueda de variables de entorno en computadora	10
Ilustración 13 Propiedades del sistema	10
Ilustración 14 Variables del sistema	10
Ilustración 15 Editar variables de entorno	11
Ilustración 16 Creación de nuevas variables	11
Ilustración 17 Comprobación de instalación de java	11
Ilustración 18 código de inicio y fin	15
Ilustración 19 Declaración de variables	16
Ilustración 20 Inicialización de variables	16
Ilustración 21 Impresión de datos	16
Ilustración 22 Entrada de datos	16
Ilustración 23 Operaciones aritméticas	17
Ilustración 24 Condicional if	17
Ilustración 25 Condicional elseif	18
Ilustración 26 Ciclo while	18
Ilustración 27 Ciclo do while	18
Ilustración 28 and, or y not	19
Ilustración 29 Declaración de arreglo	19
Ilustración 30 Declaración y uso de método	19
Ilustración 31 Ejecución de programa en terminal	19
Ilustración 32 Identificación de archivos mediante comando dir	20
Ilustración 33 Al usar el comando javacc de forma correcta se muestra la respuesta mostrada por la terminal	20

Ilustración 34 Uso de comando javac para compilar todos los archivos por eso el uso de *.java	20
Ilustración 35 Ejecución del código mediante java, regresa el texto “Fin de la revisión” una vez haya ejecutado correctamente el archivo	21
Ilustración 36 Uso de aplicación "notepad" para edición del archivo a ejecutar.....	21

Introducción

El manual actual se desarrolló con el fin de brindar asistencia en cuanto al uso y comportamiento del programa. También busca resolver cualquier duda que surja al ejecutar la fase léxica y sintáctica del proyecto deseado, es decir, el compilador. Para ejecutar el código, se requiere la instalación previa de JavaCC y Java JDK 21 para crear estas fases. Este documento está dirigido a aquellos que quieran aprender cómo usar el programa que se ha creado; el programa identificará cada par ordenado en el código fuente proporcionado.

Objetivo

Otorgar soporte al usuario acerca del uso, funcionamiento y manejo de errores de la fase léxica y sintáctica del programa.

Requerimientos

Para poder hacer uso de las fases alcanzadas del compilador es necesario tener instalado JavaCC y Java 11 en adelante. Además de contar con un dispositivo que cuente con las características básicas para hacer uso de una herramienta de desarrollo de software y del CMD. En este caso, para su desarrollo se hizo uso del SO Windows 11.

- Instalación JDK 11: <https://youtu.be/3rvdHvo9t9A?si=A4fsKSKdiBbWxUtB>
- Instalación JavaCC: <https://youtu.be/IVq3kFTkeWg?si=YGgxXMrUdRkZxfkn>
- Instalación de NetBeans (herramienta de desarrollo utilizada): https://youtu.be/KYKvMaJLSas?si=rm3gIkPJP_6MyjcZ

Instalación jdk 11

Para que la instalación sea de manera práctica pueden seguir los pasos del siguiente vídeo: <https://youtu.be/3rvdHvo9t9A?si=A4fsKSKdiBbWxUtB> .También pueden seguir los siguientes pasos que se irán explicando en este documento. Primero nos dirigimos a nuestro navegador de preferencia.

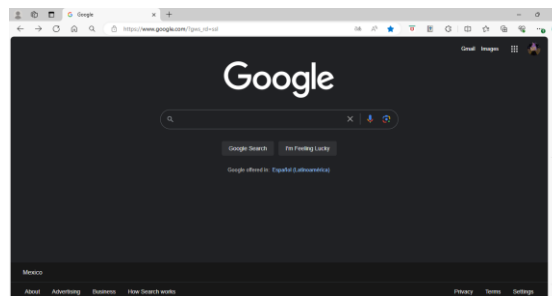


Ilustración 1 Navegador de muestra

Una vez que estemos en el navegador, en la barra de búsqueda pondremos java jdk y del daremos buscar.

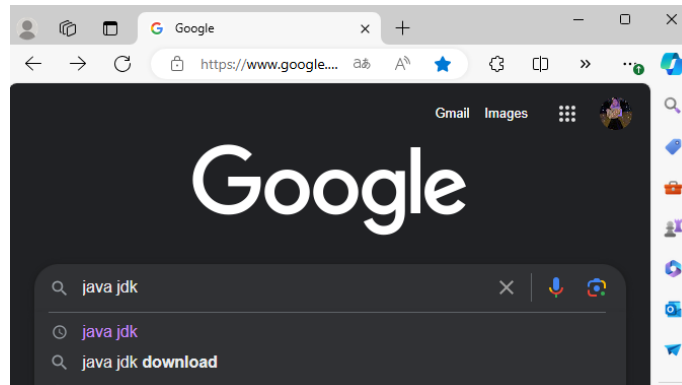


Ilustración 2 Búsqueda de java jdk en navegador

Ya que le dimos buscar, nos arrojará todas las páginas en las que podemos instalarlo, en este caso iremos a la página Java SE 11 Archive Downlads.

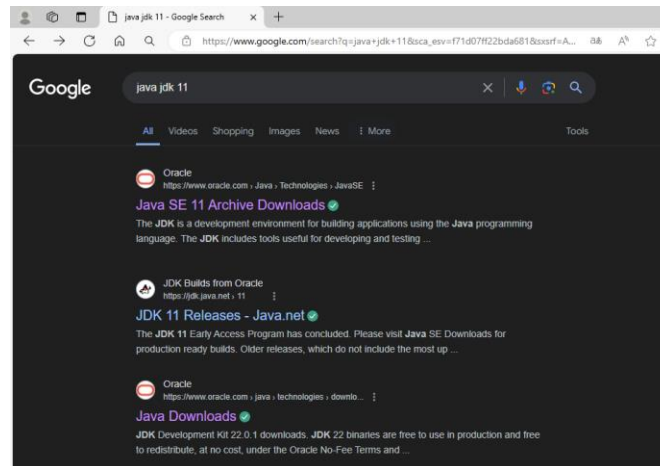


Ilustración 3 Selección de la página indicada

Seleccionamos el ejecutable de Windows para poder realizar la instalación.

Windows x64 Installer	141.43 MB	jdk-11.0.22_windows-x64_bin.exe
Windows x64 Compressed Archive	159.18 MB	jdk-11.0.22_windows-x64_bin.zip

Ilustración 4 Descarga de ejecutable en Windows

Cuando le demos a descargar Windows x64 Compressed Archive nos emerge la siguiente pantalla.

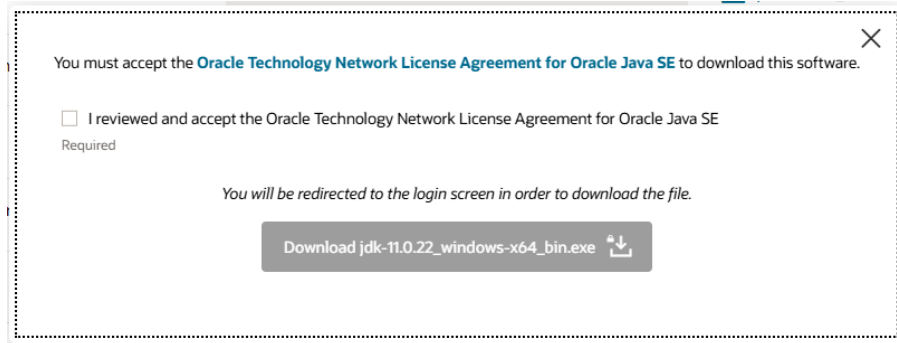


Ilustración 5 Aceptar término de licencia Oracle

Le damos en "I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE" y en el botón Download jdk-11.02. Ya que le dimos clic al botón de descargar nos abrirá la siguiente ventana.

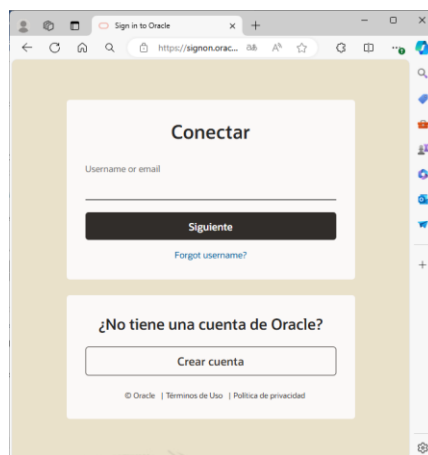


Ilustración 6 Inicio de sesión en Oracle para ejecutable

En este caso se hará un inicio de sesión para poder descargar el ejecutable del jdk y si no tienes cuenta en Oracle debes seguir el procedimiento que te indica para crear una nueva cuenta. Después de haber iniciado sesión de inmediato se descargará el JDK.

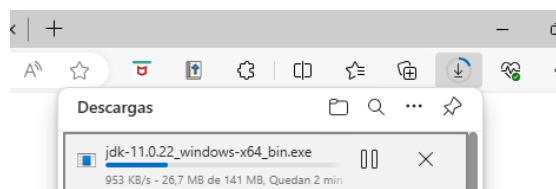


Ilustración 7 Inicio de descarga de jdk

Una vez que se haya instalado le daremos a la opción de abrir archivo para poder seguir con la instalación. Nos abrirá la siguiente ventana, en este caso seguiremos las indicaciones, por lo tanto, le daremos clic al botón “Next”.



Ilustración 8 Ventana inicial de instalador de jdk

Ya que le dimos clic a “Next”, nos aparecerá donde queremos que quede guardado el ejecutable, por omisión nos pone una dirección en automático, pero si tú lo quieres guardar en otra ruta la puedes cambiar al darle clic en “Change”. Si lo quieres dejar en la ruta que te arroja por omisión solo le daremos a “Next”.

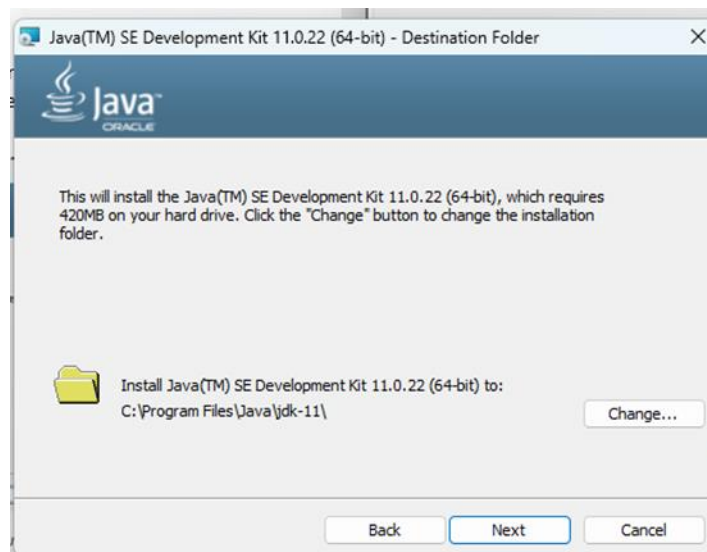


Ilustración 9 Ubicación de instalación de jdk

Al darle clic a “Next” nos mostrará otra ventana donde veremos la descarga del Java JDK.



Ilustración 10 Inicio de proceso de instalación

Cuando finaliza la descarga sin ningún problema nos aparecerá una leyenda que dirá que la descarga fue completada y si aun tienes dudas de la documentación en el botón de “Next Steps” te ayudará con un tutorial para que te puedas guiar de este, ya, si no tienes dudas solo le das clic al botón “close” y listo.



Ilustración 11 Finalización de instalación

Ahora para que nos ejecute al cien, nos iremos al buscador de la computadora y escribiremos “Variables” y seleccionamos “Editar las variables de entorno”.

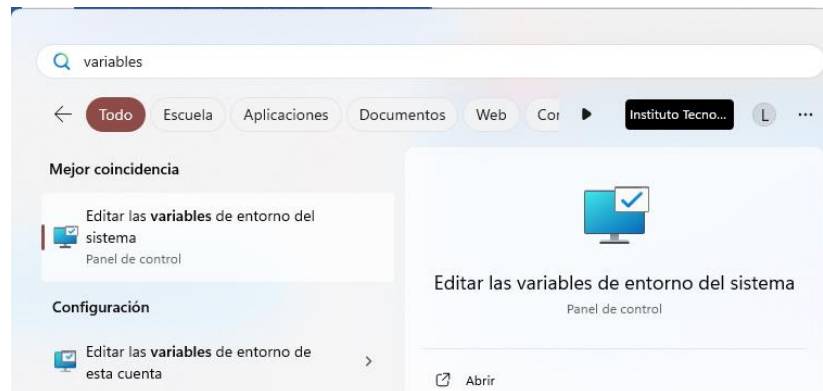


Ilustración 12 Búsqueda de variables de entorno en computadora

Da clic a “variables de entorno”.

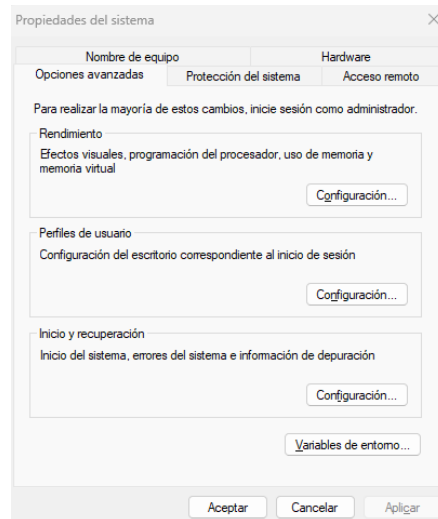


Ilustración 13 Propiedades del sistema

Seleccionamos “Path” y da clic en “Editar”.

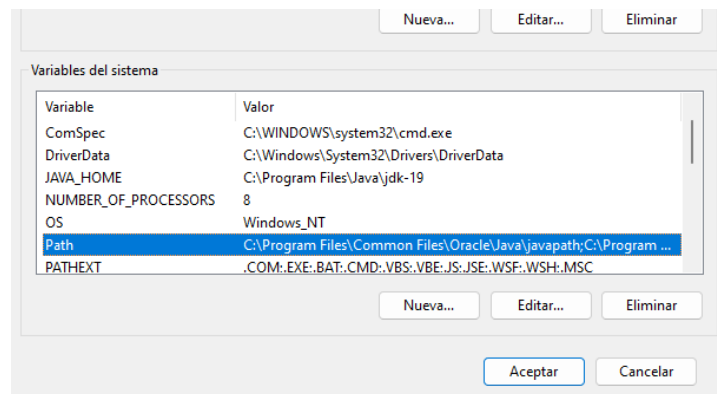


Ilustración 14 Variables del sistema

Clic en “Nuevo” e ingresa la ruta C:\Program Files\Java\jdk-11 y aplica los cambios.

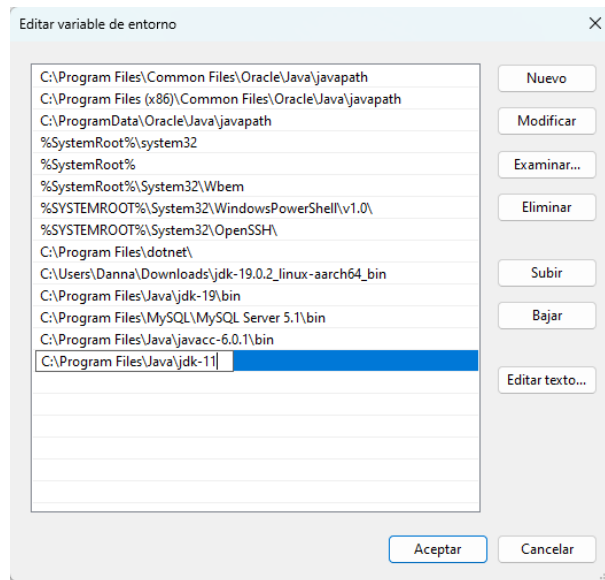


Ilustración 15 Editar variables de entorno

Ahora dentro de la ventana variables de entorno da clic en “Nueva”, asigna el nombre “JAVA_HOME” e ingresa la ruta C:\Program Files\Java\jdk-11, aplica los cambios.

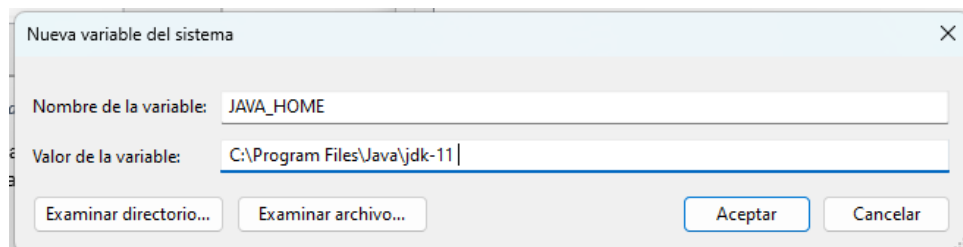


Ilustración 16 Creación de nuevas variables

Para verificar que está correctamente instalado, abriremos Windows Terminal y Ejecuta “java –version” para que nos muestre que versión de java tenemos en nuestra computadora.

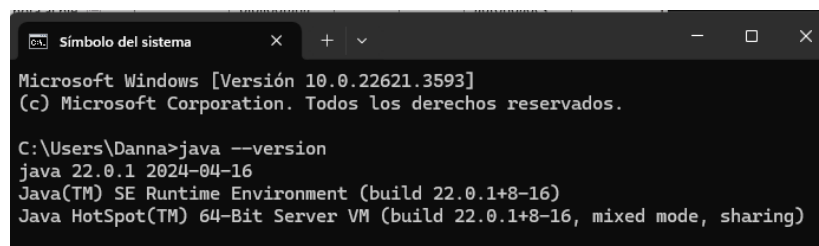


Ilustración 17 Comprobación de instalación de java

Instalación de javacc

Cómo instalar javacc: <https://www.youtube.com/watch?v=IVq3kFTkeWg&t=13s>

Descripción general

La visión del programa que se encarga de análisis de la fase léxica y sintáctica de un compilador se centra en ofrecer un analizador léxico robusto y de fácil uso que no solo identifique errores en el código fuente, sino que también promueva las mejores prácticas de codificación. El objetivo principal de la fase léxica de un compilador es transformar el código fuente en una secuencia de "tokens" o unidades léxicas significativas para el lenguaje de programación. Estos tokens representan elementos como identificadores, palabras clave, operadores, constantes y símbolos especiales.

Una vez que la fase léxica funciona adecuadamente ya es posible avanzar a la siguiente fase, la cual consiste en tomar tokens del análisis léxico como entrada y genera un árbol (o árbol de sintaxis). En esta fase, los tokens se contrastan con la gramática del código fuente, es decir, el analizador comprueba si la expresión de los tokens es sintácticamente correcta.

Definiciones técnicas

Token: Un token es la unidad básica de un lenguaje de programación. Representa una categoría de elementos léxicos reconocidos por el analizador léxico durante la fase de análisis del código fuente. Los tokens pueden ser palabras clave, identificadores, operadores, símbolos especiales, constantes, entre otros.

Palabra Reservada: Una palabra reservada es un término predefinido en un lenguaje de programación que tiene un significado especial y está reservado para su uso por el compilador o intérprete. Estas palabras suelen representar construcciones de control de flujo, tipos de datos, declaraciones o funciones esenciales del lenguaje. Ejemplos comunes de palabras reservadas en Java incluyen "if", "else", "while", "class", entre otras.

Operador Aritmético: Un operador aritmético es un símbolo matemático utilizado para realizar operaciones aritméticas en los valores numéricos. Estas operaciones pueden incluir suma (+), resta (-), multiplicación (*), división (/), entre otros. Los operadores aritméticos se utilizan para realizar cálculos matemáticos dentro de expresiones en un programa.

Operador Lógico: Un operador lógico es un símbolo utilizado para realizar operaciones lógicas entre valores booleanos (verdadero o falso). Estas operaciones incluyen la conjunción (AND), disyunción (OR), negación (NOT), entre otras. Los operadores lógicos se utilizan para evaluar condiciones y controlar el flujo de ejecución en un programa.

Comando: En el contexto de la programación, un comando es una instrucción que le indica al sistema o al programa que realice una acción específica. Los comandos pueden variar dependiendo del entorno de desarrollo, sistema operativo o lenguaje de programación utilizado. Por ejemplo, en sistemas Unix/Linux, los comandos pueden ser instrucciones para manipular archivos, ejecutar programas, administrar procesos, entre otras acciones.

Ciclo: En programación, un ciclo (también conocido como bucle) es una estructura de control que permite repetir un bloque de código múltiples veces hasta que se cumpla una condición de salida. Los ciclos son utilizados para automatizar tareas repetitivas y pueden ser de varios tipos, como el ciclo "for", el ciclo "while" y el ciclo "do-while".

Condición: Una condición es una expresión que evalúa si una afirmación es verdadera o falsa. En programación, las condiciones se utilizan en estructuras de control de flujo, como los condicionales (if-else), los bucles (while, for) y las expresiones lógicas, para controlar el comportamiento del programa basado en ciertas condiciones. Las condiciones pueden incluir comparaciones de valores, evaluaciones lógicas o cualquier expresión que devuelva un valor booleano.

Tablas de símbolos

El lenguaje que operará en el programa realizado utilizará los siguientes símbolos. Consta del uso de palabras reservadas, operaciones aritméticas, de comparación, asignación, operadores de incremento y decremento y delimitadores. En cada tabla se muestra la representación de cada sección:

Palabras Reservadas

REPRESENTACIÓN
init
end
int
float
bool
string
pi
euler
constant
void
AND
OR
NOT

print
read
If
else
elseif
while
do
define
get

Tabla 8-1 Palabras Reservada

Operaciones aritméticas

REPRESENTACIÓN
+
-
*

Tabla 8-2 Operaciones Aritméticas

Operaciones de comparación

REPRESENTACIÓN
=?
!=?
>?
<?
>=?
<=?

Tabla 8-3 Operaciones de comparación

Asignación

REPRESENTACIÓN
=

Tabla 8-4 Asignación

Delimitadores

REPRESENTACIÓN
;
{
}
(
)
[
]
:
,
.

Tabla 8-5 Delimitadores

Implementación del código

El lenguaje diseñado se implementa de la siguiente manera:

Inicio y Fin

Para declarar el inicio del código se hace uso de la palabra reservada “init” acompañada del delimitador “{”.

Para declarar el cierre del código se hace uso del delimitador “}” seguido de la palabra reservada “end”. Dentro de los delimitadores “{” y “}” va el contenido o instrucciones del código.

```
init {
```

```
} end;
```

Ilustración 18 código de inicio y fin

Declaración de variables

Para la declaración de variables basta con escribir la palabra reservada según el tipo de dato (int, float, boolean o string) seguido del nombre que se desee asignar a la variable el cual debe de iniciar con una letra minúscula.

```
int arreglo = [2];
int entradaB;
float entradaC;
bool entradaD;
String entradaE;
```

Ilustración 19 Declaración de variables

Inicialización de variables

Una variable puede declararse sin tener que asignarle un valor, por lo que solamente estaría declarada y podría recibir un valor por medio de la entrada de datos o inicializando una variable, es decir, en otra línea de código asignarle un valor determinado, por ejemplo:

```
String entradaE;
entradaE = "E";
int precio;
precio = 10;
```

Ilustración 20 Inicialización de variables

Impresión de datos

Para mostrar datos en pantalla se utiliza la palabra reservada “print” seguido de delimitadores “()” en donde se va a encontrar el mensaje a mostrar, en caso de ser una cadena de texto se hace uso de “ ” como se muestra en el ejemplo y si es una variable declarada con anterioridad simplemente se escribe dentro de los paréntesis.

```
print("Hola bienvenido al programa base de lenguaje");
print(entradaE);
print(5);
```

Ilustración 21 Impresión de datos

Entrada de datos

Para ingresar datos desde el teclado primero hay que declara una variable con su respectivo tipo de dato en la cual se almacenará el dato ingresado. Para posteriormente determinar que será una variable de entrada, como se muestra a continuación:

```
int entradaB;
read entradaB;
String entradaC;
read entradaC;
```

Ilustración 22 Entrada de datos

Uso de operaciones aritméticas

Para el uso de operaciones aritméticas (suma, resta, multiplicación y división) se declaran las variables que sean necesarias para efectuar la operación, declarando el tipo de dato con el que va a funcionar. En el ejemplo se realiza una suma de números enteros mostrando el resultado en pantalla por medio de la palabra reservada “print”.

```
int entradaA
int entradaB;
int suma;

suma = (entradaA + entradaB);
print(suma);
```

Ilustración 23 Operaciones aritméticas

Uso de la condicional if

Para el uso de la condicional se usa la palabra reservada “if” seguido de la condición que debe de seguir, ésta va dentro de los delimitadores “(“ “)” en los que se hace uso de variables y operadores de comparación, entonces se escribe la operación o mensaje que va a imprimir o realizar dentro de los delimitadores “{“ “}” en donde se hace uso del delimitador “;” como se muestra en el ejemplo.

```
int entradaB;
read entradaB;
if(entradaB <? 5) {
    print("No aplica");
}
else {
    print("Aplica para calificar");
}
```

Ilustración 24 Condicional if

Uso de la condicional elseif

A diferencia de la condicional “if”, aquí se puede añadir una nueva condición utilizando operadores aritméticos, lógicos o de comparación, después del delimitador “}” que corresponde al “if” se escribe la palabra “else if” seguido de los delimitadores “(“ “)” en los que se va a escribir la operación a realizar, entonces se coloca el delimitador “{”, en donde se escribirá el resultado de la operación o mensaje, según sea el caso, al finalizar se hace uso del delimitador “;” como se muestra en el ejemplo y finalmente se cierra el delimitador “}”.

```
int datoCliente;
read datoCliente;
int final = datoCliente-15;
int descuento = 10;

if (datoCliente >= 40) {
print(final);
}
elseif (datoCliente == 100) {
print ("Sorteo");
}
```

Ilustración 25 Condicional elseif

Uso del ciclo while

Utilizando la palabra reservada “while” que actuará como ciclo, seguido de los delimitadores “()” en los que se colocará el valor que se va a examinar para generar el bucle el cual concluye cuando el valor corresponde con la cantidad indicada. Seguido del delimitador “{” en donde irá el mensaje u operación a imprimir, la cual se imprimirá la cantidad de veces que haya sido indicada hasta llegar a 0, utilizando el delimitador “;” al finalizar la línea de código y cerrando el ciclo con el delimitador “}” como se muestra en el ejemplo.

```
int entradaB;
read entradaB;

while(entradaB<?1) {
    print("hola");
    int variable2 = 4;
}
```

Ilustración 26 Ciclo while

Uso del ciclo do while

El ciclo do while permite repetir un bloque de código varias veces, de manera iterativa, mientras se cumpla una condición específica. Antes de comenzar con la declaración del ciclo se escribe la variable y tipo de dato con el que va funcionar. Seguido de la palabra reservada “do” con el delimitador “{” donde irá la operación y/o mensaje que se ciclará cerrando con el delimitador “}” para posteriormente utilizar el ciclo while, dentro de sus delimitadores “()” va la condición que se debe de cumplir para que se realice el bucle.

```
int entradaC;
read entradaC;

do {
    print("Hola Mundo");
    entradaC = entradaC + 1;
} while (entradaC >? 4);
```

Ilustración 27 Ciclo do while

Uso de and, or y not

El uso de las operaciones lógicas se puede realizar por medio de variables, escribiendo el nombre de la operación que se desea realizar entre dos valores/entradas, por ejemplo:

```
int valor = 1 AND 2;  
bool tipo = 2 OR 2;
```

Ilustración 28 and, or y not

Declaración de arreglos

Los arreglos se declaran de la siguiente manera, primero indicando el tipo de dato que va usar seguido del nombre que recibirá el arreglo y después utilizando los delimitadores “[]” en donde irá el tamaño de la cadena.

```
int arreglo[5];
```

Ilustración 29 Declaración de arreglo

Declarar y llamar a un método

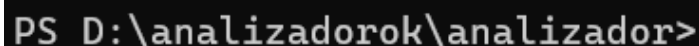
Para declarar un método se utiliza la palabra reservada “define” seguido o no de la palabra reservada void, seguido del nombre del método finalizando el nombre con paréntesis. Mientras que para llamar a un método se utiliza la palabra reservada read seguido del nombre del método sin ser seguido de paréntesis. Los métodos se colocan fuera de “init{“ y “} end;”, las cuales son las palabras reservadas que marcan el inicio y fin del código.

```
int variableA = 4;  
  
do {  
  read imprimesaludo;  
} while(5 >? variableA);  
  
} end;  
  
define void imprimeSaludo() {  
  print("Hola te saludo");  
}
```

Ilustración 30 Declaración y uso de método

Ejecución del programa

Como usuario ya se cuenta con los archivos con los que se va a ejecutar la fase sintáctica del compilador. Se selecciona una carpeta en la que se van a ubicar, y se abre desde la terminal como se muestra en la imagen:



```
PS D:\analizadorok\analizador>
```

Ilustración 31 Ejecución de programa en terminal

Utilizando el comando “dir” se mostrará el directorio y los archivos que contiene la carpeta en la que se ejecutará el programa, en este caso están el codigoFuente.txt que es en donde se encuentra el código con el lenguaje que manejará el compilador, en analizador.jj, los archivos “.jj” son archivos de especificación utilizados en la generación de analizadores léxicos y sintácticos mediante la herramienta JavaCC, y el Main.java que es donde se va a ejecutar el programa. Los archivos “.java” son fundamentales en el desarrollo de software en Java y contienen la lógica y la estructura del programa, incluyendo clases, métodos, variables, y otros elementos de programación.

```
PS D:\analizadorok\analizador> dir

Directorio: D:\analizadorok\analizador

Mode                LastWriteTime         Length Name
----                -
-a-----         22/05/2024   12:02 a. m.        18953 analizador.jj
-a-----         22/05/2024   12:02 a. m.         1058 codigoFuente.txt
-a-----         22/05/2024   12:30 a. m.         1057 Main.java
```

Ilustración 32 Identificación de archivos mediante comando dir

Para que se creen las clases necesarias para que se pueda ejecutar el programa hay que utilizar el comando “javacc” seguido del archivo “.jj” que corresponde al uso de la herramienta JavaCC.

```
PS D:\analizadorok\analizador> javacc analizador.jj
Java Compiler Compiler Version 6.0_1 (Parser Generator)
(type "javacc" with no arguments for help)
Reading from file analizador.jj . . .
Warning: Choice conflict in (...) * construct at line 235, column 2.
        Expansion nested within construct and expansion following construct
        have common prefixes, one of which is: "else"
        Consider using a lookahead of 2 or more for nested expansion.
Warning: Choice conflict in (...) * construct at line 292, column 606.
        Expansion nested within construct and expansion following construct
        have common prefixes, one of which is: "["
        Consider using a lookahead of 2 or more for nested expansion.
File "TokenMgrError.java" does not exist. Will create one.
File "ParseException.java" does not exist. Will create one.
File "Token.java" does not exist. Will create one.
File "SimpleCharStream.java" does not exist. Will create one.
Parser generated with 0 errors and 2 warnings.
```

Ilustración 33 Al usar el comando javacc de forma correcta se muestra la respuesta mostrada por la terminal

Una vez que se crean los archivos correspondientes necesarios para la ejecución del programa se requiere compilar todos los archivos “.java” que se encuentran en la carpeta para de esta manera guardar algún cambio que se haya efectuado.

```
PS D:\analizadorok\analizador> javac *.java
PS D:\analizadorok\analizador> |
```

Ilustración 34 Uso de comando javac para compilar todos los archivos por eso el uso de *.java

Para mostrar el resultado de la ejecución del programa se utiliza el comando “java” seguido del nombre del archivo, como se muestra en la ilustración.

```
PS D:\analizadorok\analizador> java Main
Fin de la revision
PS D:\analizadorok\analizador>
```

Ilustración 35 Ejecución del código mediante java, regresa el texto “Fin de la revisión” una vez haya ejecutado correctamente el archivo

Para editar el contenido del código fuente se utiliza el comando “notepad”, el cual abrirá una ventana al bloc de notas para así poder realizar cambios en él.

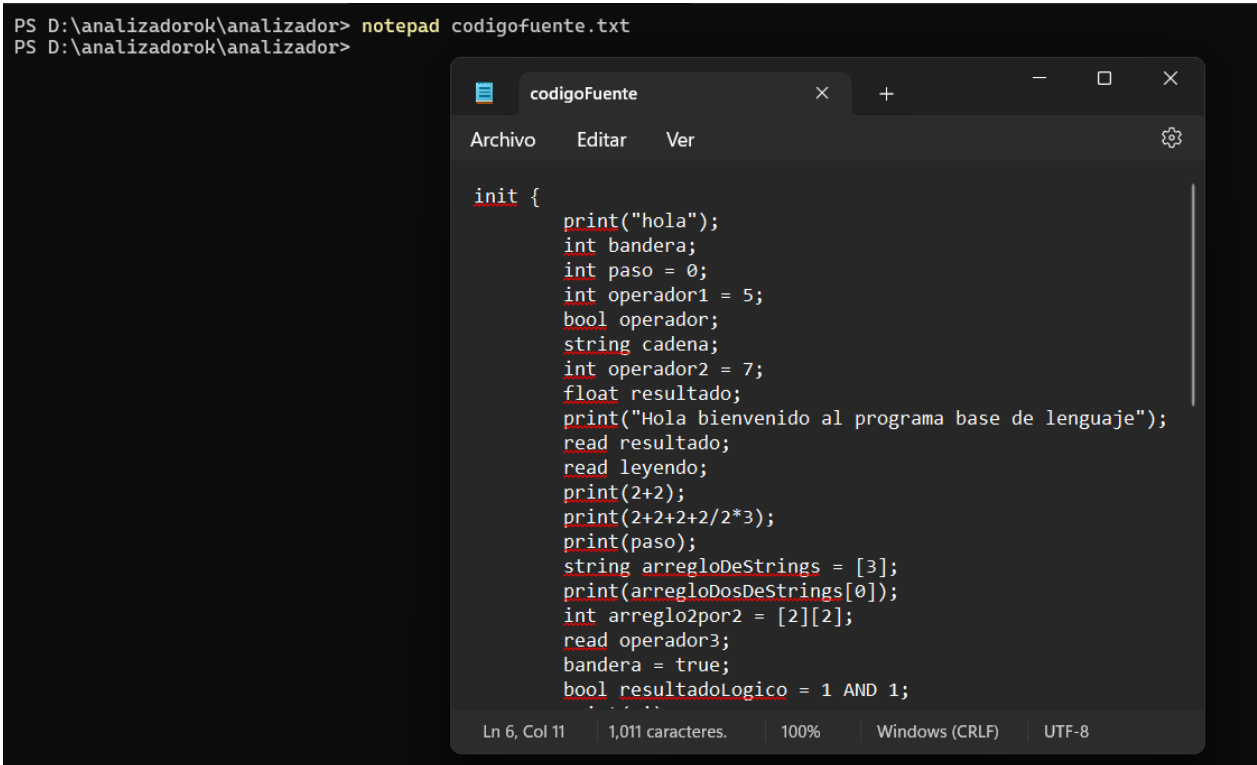


Ilustración 36 Uso de aplicación "notepad" para edición del archivo a ejecutar

Tipos de errores

Al utilizar cualquier lenguaje de programación se presentan errores, los cuales previenen los fallos para evitar que programa se comporte de manera inesperada, manejar excepciones, identificación de problemas, seguridad, entre otros factores importantes, por lo tanto, aquí se presentan los errores que se pueden presentar al utilizar el lenguaje realizado.

Errores léxicos: Ocurren cuando dentro del código fuente se encuentran símbolos o caracteres que no son reconocidos por el lenguaje, es decir, los tokens no son válidos por el compilador.

Error	Causa
Error léxico encontrado en línea "x", columna "x", se encontró "x" caracter inválido. Para corregir eliminar el carácter inválido.	Ocurre cuando se coloca un caracter no validado por el compilador, por lo que no se puede obtener el par ordenado.

Tabla 11-1 Error léxico

Errores sintácticos: Se presentan cuando la secuencia de tokens no cumple con las reglas gramaticales definidas por el lenguaje.

Error	Causa
No se encontró ";" en la línea "x", columna "x", después de "x".	Ocurre al no colocar el delimitador de finalización en la línea de código correspondiente.
No se encontró "{" en la línea "x", columna "x", después de "x".	Ocurre al no colocar la llave de apertura en el lugar correspondiente dentro del código.
No se encontró "}" en la línea "x", columna "x", después de "x".	Ocurre al no colocar la llave de cierre en el lugar correspondiente dentro del código.
No se encontró "(" en la línea "x", columna "x", después de "x".	Ocurre al no colocar paréntesis de apertura en el lugar correspondiente dentro del código.
No se encontró ")" en la línea "x", columna "x", después de "x".	Ocurre al no colocar el paréntesis de cierre en el lugar correspondiente dentro del código.
No se encontró "[" en la línea "x", columna "x", después de "x".	Ocurre al no colocar el corchete de apertura en el lugar correspondiente dentro del código.
No se encontró "]" en la línea "x", columna "x", después de "x".	Ocurre al no colocar el corchete de cierre en el lugar correspondiente dentro del código.
No se encontró delimitador de asignación =, en la línea "x", columna "x" después de "x".	Ocurre cuando no se coloca el delimitador de asignación para determinar el valor de una variable.
No se terminó la inicialización en la línea "x", columna "x", después de "x".	Ocurre cuando no se completan correctamente los valores en una operación lógica.
No se encontró un operador de comparación lógica en la línea "x", columna "x", después de "x".	Ocurre al no colocar el operador lógico entre dos valores.

No se encontró la palabra reservada void en la línea "x", columna "x", después de "x".	Ocorre al no colocar la palabra reservada void después de un define al crear un método.
No se definió el nombre del procedimiento en la línea "x", columna "x", después de "x".	Ocorre al no colocar el nombre de un método.
No se detectó el nombre de la variable en la línea "x", columna "x", después de "x".	Ocorre al no colocar un nombre de variable después de asignar un tipo de dato.
No se encontró una cadena, operación o nombre de la variable para imprimir en la línea "x", columna "x", después de "x".	Ocorre al colocar los paréntesis de la palabra reservada print vacíos.
No se encontró while en la línea "x", columna "x", después de "x".	Ocorre al no colocar la palabra reservada while al momento de realizar un ciclo do while.
Error de token	Ocorre cuando se escribe incorrectamente alguna variable, u operador.

Tabla 10-2 Errores sintácticos

Conclusión

Para resumir, la creación de las fases léxica y sintáctica de un compilador es esencial para que cualquier lenguaje de programación funcione correctamente. Estas etapas sirven como los primeros filtros y mecanismos de interpretación para garantizar que la máquina entienda correctamente el código fuente escrito por el usuario/programador.

La fase léxica permite la identificación de las estructuras básicas del lenguaje como palabras clave, identificadores, operadores y delimitadores al transformar el flujo de caracteres en una secuencia de tokens. Aquí se produce la obtención de un par ordenado. Al organizar el código en componentes manejables, esto mejora el rendimiento y facilita la detección temprana de errores léxicos.

La fase sintáctica, que se basa en la gramática del lenguaje para estructurar y validar las relaciones entre los tokens, es esencial para la creación de árboles sintácticos que reflejan la estructura jerárquica del código. Este análisis sintáctico establece una base sólida para las fases posteriores del compilador, como el análisis semántico, asegurándose de que el código cumpla con las reglas gramaticales del lenguaje.

Bibliografía

¿Qué es Java? - Explicación del lenguaje de programación Java - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/es/what-is/java/>

Robinharwood. (n.d.). Comandos de Windows. Microsoft Learn. <https://learn.microsoft.com/es-es/windows-server/administration/windows-commands/windows-commands>