UNIVERSIDAD DEL BÍO-BÍO

Tarea 2
Fundamentos Ciencias de la Computación

Profesor: Luis Gajardo.
Alumnos:
Sebastián Cádiz
Pablo Melgarejo
Javier Toro
Asignatura: Fundamentos Ciencias de la Computación.
Fecha de entrega: 27 de junio, 2025.

Descripción del programa

Para este programa, como fue indicado se utiliza Sablecc como framework, el cual entrega una estructura para elaborar compiladores e interpretadores.

Respecto a aquello, se le debe entregar una gramática para definir la sintaxis del lenguaje que se busca procesar. Esta gramática se encuentra especificada en el archivo **.grammar** donde se establecen las reglas utilizadas como la base para que Sablecc elabore las clases necesarias para su utilización y manejo.

En un comienzo se lee un archivo de texto, es este caso llamado **programa.txt**, del cual mediante la utilización de la clase ***Lexer() y TokenIndex()*** se "tokeniza" cada uno de los carácteres presentes en el archivo, los cuales pueden ser operadores, identificadores, etc. A partir de esto, mediante la clase ***Parser()*** se crea el AST con los tokens obtenidos, enfocándose en la jerarquía semántica del código entregado.

También se encuentra la clase **Interpreter()**, la cual extiende a **DepthFirstAdapter()**, lo cual permite recorrer el AST de manera sistemática. Este enfoque de recorrido **depth-first** facilita el análisis y evaluación de cada nodo del árbol en un orden jerárquico descendente.

Donde además de las funciones heredadas, la clase contiene métodos propios que permiten identificar la naturaleza de cada nodo del árbol. Según el tipo de nodo por ejemplo, una operación aritmética, una asignación o una entrada/salida se determina el valor a retornar o la acción a ejecutar.

Durante este proceso, se hace uso de una estructura de tipo *HashMap* para almacenar y recuperar variables declaradas por el usuario, permitiendo así manejar el estado del programa durante su ejecución.

Instrucciones de compilación y ejecución

Es importante verificar que su equipo tenga instalado Java y su versión, puede verificar desde la consola de comandos ingresando:

java -version

También verificar que tiene Sablecc instalado, de preferencia en el disco **C:**.

Luego de descargar el archivo **.zip** que contiene la tarea, descomprimirlo en la carpeta donde tenga Sablecc. Por ejemplo:

C:\sablecc-3.7

Al realizar lo anterior, dirigirse al directorio donde se encuentra descomprimido el proyecto, y desde ahí abrir el terminal, también puede directamente abrir el terminal y escribir:

cd “ruta del archivo”

Tras aquello, ejecutar el siguiente comando:

sablecc lenguaje.grammar

De surgir errores verificar que la ruta se encuentra correctamente especificada. Continuando, en la línea de comandos ejecutar los siguientes comandos:

javac work\Interpreter.java

javac work\Main.java

Para ejecutar el programa, en la línea de comandos ejecutar:

java work.Main

También, puede seguir las instrucciones especificadas en el archivo REAMDE.md.

Ejemplo de uso

Ejemplo 1:

Acá se utiliza como archivo base desde donde comienza el parseo un código ubicado en un archivo del proyecto, lo parsea, creando el AST para luego utilizar el interpretador para evaluar y ejecutar el código.

A continuación, una muestra del código:

```
main() {
int side1;
int side2;
int side3;
/*ingresar lados del triangulo*/
println("ingrese el lado 1: ");
input(side1);
println("ingrese el lado 2: ");
input(side2);
println("ingrese el lado 3: ");
input(side3);

if(side1==side2 && side2==side3){
/* si todos los lados son iguales*/
println("tringualo equilatero");
}
if(side1==side2 || side1==side3 || side2==side3){
/* si hay dos lados iguales*/
println("tringualo isosceles");
}
if(side1!=side2 && side1!=side3 && side2!=side3){
/*si no hay lados iguales*/
println("triangulo escaleno");
}
}
```

A medida que interpreta el código, permite que el usuario ingrese datos para verificar con el código a interpretar, el cual en este caso identifica que tipo de triángulo es a partir de la longitud de sus lados.

```
PS C:\SableCC\work2_fdm> java work.Main
¡Parseo exitoso!
ingrese el lado 1:
2
ingrese el lado 2:
3
ingrese el lado 3:
3
tringualo isosceles
```

Ejemplo 2:

En este caso, el programa interpreta un código que solicita al usuario ingresar un número de filas y, a partir de este valor, imprime una pirámide de asteriscos alineada al centro. El código demuestra el uso de estructuras de control como bucles while, así como operaciones aritméticas básicas.

A continuación, una muestra del código:

```
main() {
int f;
int i = 1;
println("Ingrese el número de filas");
input(f);
while(i <= f) {
int espacio = f - i;
int j = 0;
/*Imprime los espacios antes de los asteriscos*/
while(j < espacio) {
print(" ");
j++;
}
int k = 0;
/*Imprime los asteriscos de la fila actual*/
while(k < (2 * i - 1)) {
print("*");
k++;
}
println("");
i++;
}
}
```

Al ejecutar el programa e ingresar, por ejemplo, el valor 5, se obtiene el siguiente resultado:

```
¡Parseo exitoso!
Ingrese el número de filas
5
    *
   ***
  *****
 *******
*********
```