TEC
Tecnológico
de Costa Rica

Proyecto #2

Análisis semántico y generación de código intermedio

Fredd Badilla Víquez - 2022012800

Tecnológico de Costa Rica

Compiladores e Intérpretes

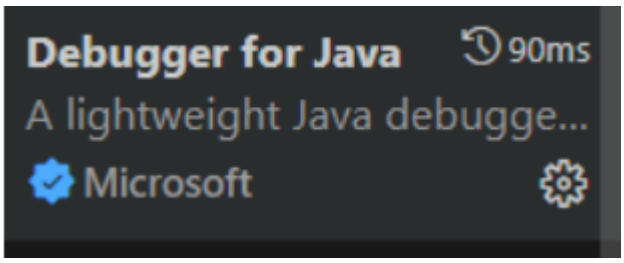
Allan Rodríguez Dávila

Semestre II, 2024

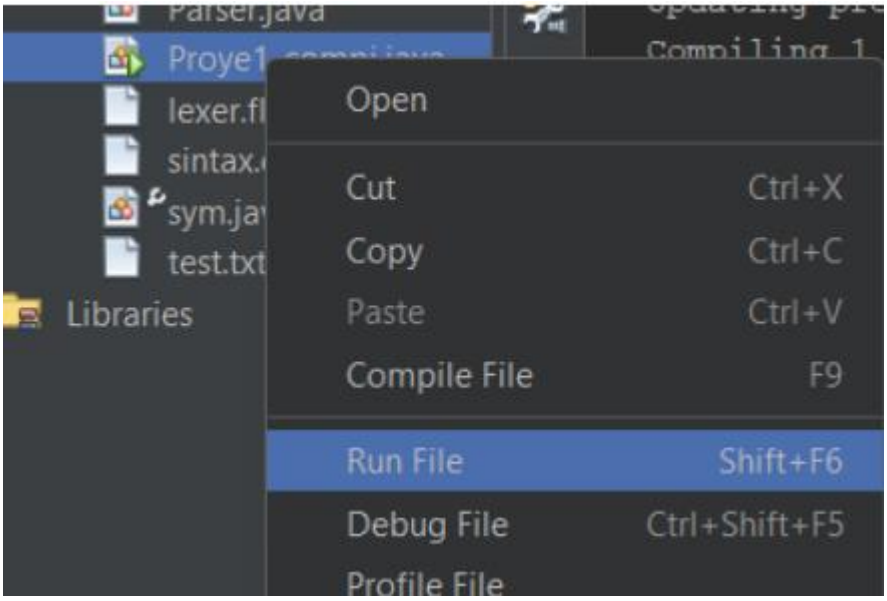
Manual de usuario

El programa es un programa de Java simple, para asegurarse de poder correrlo asegúrese de contar con el JDK de java en su equipo, puede instalarlo desde la página oficial.

Para ejecutarlo puede hacerlo desde netbeans o VisualStudioCode con la extensión para java de Microsoft que se indica a continuación.

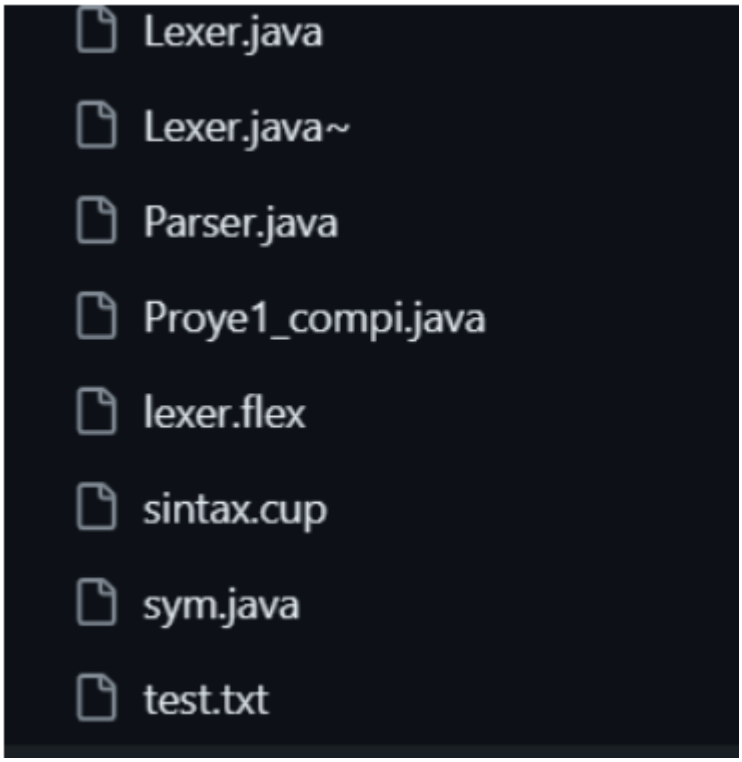


Si cuenta con la extensión y el jdk debería ser capaz de ejecutar el programa, para ello presione click derecho sobre la clase main del proyecto, en este caso se llama Proye1_Compi.java, al dar click derecho, seleccione la opción “Run File” tal como se muestra en la imagen.



Al hacer esto, el programa creará primero el archivo Lexer.java y luego los archivos Parser.java y sym.java.

En la imagen siguiente se pueden ver los archivos mencionados de primero, tercero y séptimo lugar.



Dependiendo de la máquina en la que se ejecute este proceso puede verse entorpecido si se intenta crear el Parser.java antes de que se complete la creación del Lexer.java, si esto le llega a suceder, vuelva a correr el programa una segunda vez y esto debería solucionar el problema.

Si el proceso se completa con éxito, el programa realizará en análisis del código fuente en el archivo test.txt y en el documento TOKENS.txt podrá la lista de los tokens que se encuentren en el archivo test.txt y adicionalmente se van a imprimir en pantalla las tablas de símbolos que se generen.

Pruebas de funcionalidad

Primero algunas validaciones semánticas:

Como vemos hay una variable que esta siendo declarada dos veces.

```
glob:int:hola;
glob:int:hola;
```

El error correspondiente:

```
La variable global -> hola <- ya fue declarada
BUILD SUCCESSFUL (total time: 2 seconds)
```

Como vemos las variables f11 es float y se requiere hacer una resta con un número entero,

lo cual debido al tipado fuerte no debería ser posible.

```
loc:float:f11;

loc:int:in1=f11--- -14/in1+++7-15;
```

El error correspondiente:

```
Error semantico, en esta resta los operadores deben ser del mismo tipo
BUILD SUCCESSFUL (total time: 2 seconds)
```

Se llama a una función que no existe.

```
miFunc1(miFunc(),'a'); //semantico miFunc, hola
```

El error correspondiente:

```
ERROR -> La funcion llamada no ha sido declarada
BUILD SUCCESSFUL (total time: 2 seconds)
```

En esta función su valor de retorno es entero, como vemos se esta devolviendo una variable de tipo booleano, lo cual no debería de ser posible.

```
func:int:main_
    loc:bool:bll = true;
    return:bll;
_
```

El error correspondiente:

```
ERROR RETORNO -> Tipo de retorno distinto al de la funcion
BUILD SUCCESSFUL (total time: 2 seconds)
```

Toda función debe tener un retorno al menos.

```
func:int:main_
    loc:bool:bll = true;


_
```

El error correspondiente:

```
ERROR -> Las funciones deben tener al menos un retorno
BUILD SUCCESSFUL (total time: 2 seconds)
```

Ahora seguiremos con la generación de código intermedio

```
glob:int:ss1;
glob:int:array:xd1[11]  = [1,2,2,2];

func:int:xd_

        loc:int:x22;
        param:int:xddd;
        loc:int:var = 2;
        for:i:in:range(12, 10, 3)_print(xd);_
        if(2==2)_print(2);_else_print(2);_
        while(2==2)_print(2);_
        return:5;
_
```

Salida de código intermedio.

global_data_int ss1

global_data_arr_int xd1

t1=1

t2=2

t3=2

t4=2

xd1=[t1, t2, t3, t4]

begin_func_xd:

local_data_int x22

param1 xddd

```
local_data_int var

t1=2

var = t1

t2=12

t3=10

t4=3

for_begin1:

local_data_int i

i = 12

t5 = 10

t6 = i < t5

if t6 goto forBlock1

goto for_end1

forBlock1:

t7=xd

print t7

for_end_block1:

t8 = 3
```

```
i = t8 + i

t9 = 10

t10 = i < t9

if t10 goto forBlock1

for_end1:

if_begin2:

t11=2

t12=2

t13 = t12==t11

if t13 goto ifblock2

goto else_begin2

ifblock2:

t14=2

print t14

if_end2:

else_begin2:

t15=2

print t15
```

```
else_end2:

while_begin3:

t16=2

t17=2

t18 = t17==t16

if t18 goto whileBlock3

goto end_while3

whileBlock3:

t19=2

print t19

end_block_while3:

if t18 goto whileBlock3

end_while3:

t20=5

return t20
```

Descripción del problema

Este proyecto implica desarrollar el análisis semántico y la generación de código intermedio para un lenguaje definido por la gramática del Proyecto I, junto con el Lexer y Parser asociados.

Para verificar el desarrollo del programa, se debe realizar lo siguiente con un archivo fuente:

1) Determinar si el archivo fuente puede ser generado por la gramática, considerando su sintaxis y semántica (incluyendo tipado explícito y fuerte).
2) Manejar cualquier error léxico, sintáctico o semántico encontrado utilizando la técnica de Recuperación en Modo Pánico.
3) Generar un archivo que contenga el código intermedio correspondiente al archivo fuente, manteniendo su sentido semántico.

Librerías usadas

1) import java.io.BufferedReader;
2) import java.io.File;
3) import java.io.FileNotFoundException;
4) import java.io.FileReader;
5) import java.io.FileWriter;
6) import java.io.IOException;
7) import java.io.Reader;
8) import java.nio.file.Files;
9) import java.nio.file.Path;
10) import java.nio.file.Paths;
11) import java.nio.file.StandardCopyOption;
12) import java_cup.runtime.Symbol;
13) import jflex.exceptions.SilentExit;

Análisis de resultados

Requerimientos	Porcentaje alcanzado
Realiza la comprobación semántica del código fuente.	Logrado: 100%.
Desarrollado utilizando la herramienta Cup. Realiza la comprobación semántica del código fuente.	Logrado: 100%.
Analizador Semántico	Logrado: 100%, se lograron validar todas las posibles inconsistencias encontradas.
La generación de código se realiza de manera coherente con la semántica del programa, preservando su lógica y estructura.	Logrado: 100%.
Generación de código intermedio	Logrado: 100%, se logró generar todo el código intermedio de todas las posibles producciones permitidas.

Bitácora

https://github.com/FR3DD221/Fredd-Proyecto-2Compi.git