Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Compiladores e Intérpretes

Proyecto 1

Profesor:

Ignacio Trejos Zelaya

Estudiante:

Luis Enrique Loria Cordero - 2015041974

I Semestre, 2020.

Esquema para el manejo del texto fuente

En el apartado del manejo del texto fuente, no fue necesario realizar cambio alguno.

Modificaciones

Dentro de las nueva modificaciones se agregaron nuevos tokens “and, elsif, exit, next, nil, private, rec, repeat, return,to, until”. Para la creación de identificadores propios, se creó una clase que genera identificadores cuando estos son omitidos

Nuevas rutinas

Estas fueron las nuevas rutinas que se agregaron al compilador
"repeat" "while" Expression "do" Command "end"
"repeat" "until" Expression "do" Command "end"
"repeat" "do" Command "while" Expression "end"
"repeat" "do" Command "until" Expression "end"
"repeat" "var" Identifier "in" Expression "to" Expression "do" Command "end"
"repeat" "loop" [Identifier] "do" Command "end"
"exit" [Identifier]
"next" [Identifier]
"Return"

Proc-Func ::=
"proc" Identifier "(" Formal-Parameter-Sequence ")" "~" Command "end"
| "func" Identifier "(" Formal-Parameter-Sequence ")" ":" Type-denoter "~" Expression

Proc-Funcs ::= Proc-Func ("and" Proc-Func)+

compound-Declaration ::= single-Declaration
|"rev" Proc-Funcs "end"
|"private" Declaration "in" Declaration "end"

Extensión para ver los árboles en el ide

Para que el ide pueda poder imprimir los nuevos cambios, era necesario que los nuevos métodos visit de los nuevos ast fuera implementados en la clase de treevisitor.

Extension para generar el XML

Para generar el archivo XML se creó un visitor concreto que implementa la interfaz visitor. Esto lo que hace es un método personalizado para objeto que extiende visitor. Se crea un string por cada árbol de sintaxis abstracta que se visite.

Casos de prueba

Objetivo de este caso de prueba es mostrar el funcionamiento correcto del comando let
Prueba positiva:
Este es el código del caso de prueba positivo

```
enrique@pop-os:~/Documents/COMPI/Proyecto 1/ejemplos$ cat ejemplo1.tri
let
 var i: Integer;
 var j: Integer;
 var k: Integer;
 var num: Integer;

 proc test (var q: Integer) ~ i:= 1 end

in

 getint (var num); ! Number of iterations.
 i:= 1;
 j:= 2;
 repeat loop enrique do j:= k end ;
 return

 end ! Comentario
```

Resultado esperado:

Se espera que el código compile sin error alguno.

Resultado observado:
Efectivamente, el código compila sin ningún error.

Prueba Negativa
Este es el código del caso de prueba negativa, no posee el end en el comando let

```
enrique@pop-os:~/Documents/COMPI/Proyecto 1/ejemplos$ cat ejemplo1.tri
let
 var i: Integer;
 var j: Integer;
 var k: Integer;
 var num: Integer;

 proc test (var q: Integer) ~ i:= 1 end

in

 getint (var num); ! Number of iterations.
 i:= 1;
 j:= 2;
 repeat loop enrique do j:= k end ;
 return
```

Resultado esperado:
Se espera que el código compile con error debido a la falta del end en el comando let;

Resultado observado:
Efectivamente, el código compila con error, este muestra un mensaje de error donde solicita la palabra end al final del código.

Objetivo de este caso de prueba es mostrar el funcionamiento correcto de la declaración proc

Prueba positiva:
Este código muestra una prueba positiva para la declaración de proc

```
enrique@pop-os:~/Documents/COMPI/Proyecto 1/ejemplos$ cat ejemplo1.tri
let
 var i: Integer;
 var j: Integer;
 var k: Integer;
 var num: Integer;

 proc test (var q: Integer) ~ i:= 1 end

in

 getint (var num); ! Number of iterations.
 i:= 1;
 j:= 2;
 repeat loop enrique do j:= k end ;
 return

 end ! Comentario
```

Resultado esperado:
Se espera que el código compile sin error alguno.

Resultado observado:
Efectivamente, el código compila sin ningún error.

Prueba Negativa
En esta prueba, la definición de proc requiere un identificador antes de los parámetros

```
enrique@pop-os:~/Documents/COMPI/Proyecto 1/ejemplos$ cat ejemplo1.tri
let
 var i: Integer;
 var j: Integer;
 var k: Integer;
 var num: Integer;

 proc  (var q: Integer) ~ i:= 1 end

in

 getint (var num); ! Number of iterations.
 i:= 1;
 j:= 2;
 repeat loop enrique do j:= k end ;
 return

 end ! Comentario
```

Resultado esperado:
Se espera que el código compile con errores.

Resultado observado:
Efectivamente, el código compila con errores, mostrando la carencia del identificador

Objetivo de este caso de prueba es mostrar el funcionamiento correcto del comando repeat while

Prueba positiva

Este caso muestra el correcto funcionamiento del comando repeat while

```
enrique@pop-os:~/Documents/COMPI/Proyecto 1/ejemplos$ cat ejemplo1.tri
let
 var i: Integer;
 var j: Integer;
 var k: Integer;
 var num: Integer;

 proc test (var q: Integer) ~ i:= 1 end

in

repeat while (k>9) do j:=2 end

 getint (var num); ! Number of iterations.
 i:= 1;
 j:= 2;
 repeat loop enrique do j:= k end ;
 return

 end ! Comentario
```

Resultado esperado:
Se espera que el código compile sin errores.

Resultado observado:
Efectivamente, el código compila sin errores.

Prueba negativa:
Esta prueba muestra un error en el comando repeat while

```
enrique@pop-os:~/Documents/COMPI/Proyecto 1/ejemplos$ cat ejemplo1.tri
let
 var i: Integer;
 var j: Integer;
 var k: Integer;
 var num: Integer;

 proc test (var q: Integer) ~ i:= 1 end

in

repeat while (k>9)  j:=2 end

 getint (var num); ! Number of iterations.
 i:= 1;
 j:= 2;
 repeat loop enrique do j:= k end ;
 return

 end ! Comentario
```

Resultado esperado:
Se espera que el código compile con errores.

Resultado observado:
Efectivamente, el código compila con errores, mostrando la carencia de la palabra reservada *do*

Discusion y analisis de resultados

Basándose en las pruebas descritas, y en otras realizadas mientras de modificar el compilador, este cumple con los requerimientos de los cambios solicitados al compilador. Estos resultados reflejan que los cambios fueron exitosos.

Reflexion sobre la experiencia de modificar código de otros

Cambiar el código de otras personas resulta relativamente complicado, puesto que se tiene que pensar que estaba estaba haciendo el otro programador en otro momento. Si bien siempre hay discrepancias en las implementaciones de otros, es importante que se sigan estándares para que futuras personas puedan modificar el código correctamente sin contratiempos algunos.

Como se compila

El proyecto inicial estaba creado en el entorno de desarrollo *netbeans*, pero para los cambios realizados, se utilizó *intellij*, pero la importación de nuevo a netbeans debería de ser si problemas.
También como sistema operativo se utilizo Linux, para la compilación es necesario posicionarse en la carpeta de los fuentes del código (src) para compilar el proyecto

Como se ejecuta

Para ejecutar el proyecto desde la línea de comandos, se debe hacer el llamado a *compiler* desde la carpeta src y pasar como parámetro el código fuente a compilar.

Archivos con el texto fuente

Los archivos fuentes se encuentran en el siguiente repositorio public de GitHub:

https://github.com/Lenrique/proyecto-1-compi

Ejecutable

Se adjunta el ejecutable del IDE