

Instituto Tecnológico de Costa Rica

Compiladores e Intérpretes – IC4700

***Mini C#***

***Segunda Parte***

Mariana Artavia Venegas

Francisco Javier Ovares Rojas

Grupo 50

Oscar Mario Víquez Acuña

I Semestre – 2023

Inicialmente luego, se generó mediante la ayuda de la herramienta ANTLR los Visitors a partir del Parser generado, una vez realizado esto se creó una clase de análisis contextual llamada “CSharpContextual.cs”, la cual hereda de ParserBaseVisitor generado. Una vez realizado esto se generaron los métodos Visitor necesarios para el proyecto en la dicha clase.

Una vez generados los métodos, se empezaron a sobrescribir uno a uno, esto con el fin de que cuando se generara el árbol con el parse program, se visitara el contexto del árbol y así en cada elemento se fueran validando los tipos respectivos en cada caso de métodos, variables, comparaciones, declaraciones, constantes, boleanos entre otros.

La inserción de elementos en la tabla de símbolos ocurre en diferentes partes del código, como cuando se declara una variable, se define un método o se crea una clase. Cada vez que se encuentra una declaración de este tipo, se crea un objeto correspondiente (por ejemplo, un objeto de tipo TipoBasico (int,String,char,), Arreglo, TipoClase o Metodo) y se agrega a la tabla de símbolos utilizando el método insertar de la clase CSTablaSimbolos.

Antes de insertar un elemento, se realizan algunas validaciones importantes. Por ejemplo, se verifica si el nombre del elemento ya ha sido declarado previamente en la tabla de símbolos. Si el nombre ya existe, se muestra un mensaje de error indicando que ese nombre ya ha sido utilizado y no se puede repetir.

También se realizan validaciones específicas según el tipo de elemento que se está insertando. Por ejemplo, al insertar una variable en la tabla de símbolos, se verifica si su tipo es válido y si cumple con las reglas del lenguaje de programación. Si el tipo no es válido, se muestra un mensaje de error indicando que el tipo de variable no es permitido.

Validaciones importantes que se realizan antes de insertar un elemento en la tabla de símbolos en cada caso específico:

1. Declaración de variables (VisitVarDeclaAST):

* Se verifica si el identificador de la variable ya ha sido declarado previamente en la tabla de símbolos. Si es así, se muestra un mensaje de error indicando que el identificador ya ha sido utilizado y no se puede repetir.
* Se verifica si el tipo de la variable es válido y cumple con las reglas del lenguaje de programación. Se comprueba si es un tipo básico, un arreglo o una clase válida. Si el tipo no es válido, se muestra un mensaje de error indicando que el tipo de variable no es permitido.

1. Declaración de clases (VisitClassDeclaAST):

* Se verifica si el nombre de la clase ya ha sido declarado previamente en la tabla de símbolos. Si es así, se muestra un mensaje de error indicando que el nombre de la clase ya ha sido utilizado y no se puede repetir.
* Se verifica si el nombre de la clase es válido y cumple con las reglas del lenguaje de programación.

1. Definición de métodos (VisitMethDeclaAST):

* Se verifica si el nombre del método ya ha sido declarado previamente en la tabla de símbolos. Si es así, se muestra un mensaje de error indicando que el nombre del método ya ha sido utilizado y no se puede repetir.
* Se verifica si los parámetros del método están definidos correctamente y cumplen con las reglas del lenguaje de programación definidas en el Parser.

**Rubro de verificación de tareas realizadas**

| Rubro | Realizado % |
| --- | --- |
| Tabla de Símbolos | |
| Estructura y Funcionalidad | 100% |
| Chequeo y Alcances | |
| Control de nivel de los identificadores |  |
| Control de múltiples declaraciones de identificadores | 100% |
| Control de existencia de métodos | 100% |
| Control de cantidad de parámetros | 100% |
| Control de nombres de campos en clases | 100% |
| Reporte de Errores (errores significativos) | 100% |
| Chequeo de Tipos | |
| Chequeo de tipos de asignaciones | 100% |
| Chequeo de tipos en expresiones lógicas | 100% |
| Chequeo de tipos en métodos preestablecidos (add, del, len) |  |
| Chequeo de tipos en expresiones complejas | 100% |
| Chequeo de tipos en campos de clases | 100% |
| Chequeo de tipos en argumentos de métodos | 100% |
| Chequeo de tipos en los retornos de funciones | 100% |
| Reporte de errores (errores significativos) | 100% |
| Documentación | |
| Formato y Contenido | 100% |
| Ortografía y Gramática | 100% |

**Conclusión**

Realizar el análisis contextual en el proceso de compilación es como revisar y corregir un trabajo escolar antes de entregarlo. Durante esta etapa, nos aseguramos de que todas las palabras y oraciones estén bien escritas, de que usemos las palabras adecuadas en cada contexto y de que no hayamos repetido información innecesariamente.

Al llevar a cabo este análisis, creamos una lista donde anotamos todas las palabras y frases que usamos en nuestro programa. También verificamos que cada palabra esté escrita correctamente y que se use de la manera correcta. Además, nos aseguramos de que no hayamos utilizado la misma palabra para dos cosas diferentes, porque eso puede confundir al programa.

El haber realizado análisis contextual de este Mini-CSharp nos ayuda a revisar y corregir nuestro programa para que funcione correctamente. Aprendimos a prestar mayor atención a los detalles, a usar las palabras adecuadas en cada lugar y a evitar repeticiones innecesarias. Esto nos ayuda a crear programas más sólidos y confiables, y a entender mejor cómo funciona el proceso de compilación. Es como hacer una revisión final antes de entregar nuestro trabajo, para asegurarnos de que esté impecable.