|  |  |
| --- | --- |
| ***Instituto Tecnológico de Costa Rica***  ***Sede Regional San Carlos*** | Segunda Tarea Programada  Compiladores e Intérpretes |
| Prof. Oscar Mario Víquez Acuña | ***Análisis Contextual*** |
|  | Compilador Monkey |

**Descripción:**

La fase de análisis sintáctico ha sido concluida y ahora el problema a enfrentar es la solución del análisis contextual. Nuestro enemigo deja de ser la forma en que se escribe el programa y sus diferentes construcciones por parte del usuario (sintaxis) para convertirse en la validez con la que se usan los identificadores en el programa. Al tratarse de un lenguaje sin declaración de tipos, lo más importante en esta etapa será lo relacionado con la existencia de métodos invocados, así como el chequeo de alcances cuando así sea requerido.

En esta etapa se construirán los Visitor utilizando la estrategia creada por la herramienta ANTLR4, según lo discutido en clase, para elaborar la inspección de alcances de los identificadores de los programas del lenguaje.

La inspección de alcances debe ser auxiliada por una ***tabla de identificadores*** declarados, llevando para cada identificador la información relevante para realizar los chequeos pertinentes a el uso de dicha tabla.

Tabla se símbolos o tabla de identificadores

Es necesario recordar que estamos ante la creación de un lenguaje con estructura de bloque anidados, por lo que se debe realizar la implementación de esta tabla considerando los diferentes niveles de bloque que se pueden presentar. Así mismo, queda a criterio del programador si implementa una tabla o varias tablas.

La tabla de símbolos debe considerar los métodos pre-establecidos (len, first, last, etc) ya que estos forman parte del lenguaje y al llamar a un método, debe tratarse igual la verificación del nombre y la cantidad de parámetros, que cualquier otro método definido.

Otro dato a considerar es el contenido de los atributos de cada identificador. Puede que no sea necesario en el caso de las variables o métodos, almacenar su tipo, pero si podría ser necesario en algunos casos, guardar punteros a las declaraciones ya sea para trabajos en esta misma etapa del intérprete o más bien para la próxima.

Volviendo a la creación de los procedimientos de chequeo de alcances, la propuesta del trabajo a realizar establecerá las acciones a realizar en cada uno de los visitas, para lo cual algunos de ellos prácticamente no realizan más que visitar a sus hijos y otros de ellos si realizan chequeos tanto de alcances como los pocos de tipos que puedan requerirse.

Algunas reglas contextuales del compilador que en esta fase se DEBEN cumplir. No se incluyen todos los casos de errores contextuales en esta lista para que el alumno identifique los faltantes y sea capaz de modelar su solución:

1. Ningún identificador podrá ser utilizado en una expresión sin haber sido declarado previamente (en el caso de variables, con “let”).
2. El número de los parámetros en procedimientos/funciones deben ser igual a los declarados por dicho procedimiento/función.
3. No debe permitir retornar un valor a menos que se encuentre dentro de una función. (el retorno es con una asignación al nombre de la función)
4. Los hashContent solo deben permitir en la primer expresión, elementos de tipo entero o string.
5. En el caso de los hashLiteral, cuando se quiera acceder a un elemento a partir de una elementAccess, debe validarse que el campo al que se quiere acceder existe.
6. Noten que declarar una función es un expresión, por lo que es posible declarar funciones casi que dentro de cualquier contexto. Para limitarlo, aparte de la declaración normal de funciones, será posible declarar funciones dentro de un arreglo, sin que esto sea un error. Debe también considerarse que sea agregada a alguna tabla de símbolos considerando las restricciones de alcance que se puedan tener o no (SE DEBE DISCUTIR Y ANALIZAR ESTE ASUNTO).

Cualquier detalle de implementación o de reglas contextuales obviado en este anunciado puede ser asumido por el programador de la manera que crea conveniente previo supervisión y aprobación del profesor. Cualquier agregado o suposición hecha debe ser respectivamente documentada.

**Documentación:**

La documentación debe ser un video al igual que la etapa anterior y debe contener descripciones sobre:

* Estrategia de verificación de alcances para el lenguaje.
* Resultados obtenidos sin describir en demasía el código fuente
* Ejemplo de corrida a partir de códigos fuente creados.

El video no debe exceder los 7 minutos

**Aspectos Administrativos:**

* La tarea se desarrollará en grupos de máximo dos personas.
* La fecha de entrega será el Lunes 09 de Noviembre del 2020 antes de las **10** **PM** (**SIN EXCEPCIONES NI MODIFICACIONES A LA FECHA**).
* La entrega se realizará a través del tec-digital sin excepciones.
* Cualquier intento de plagio, copias totales o parciales de otras personas o de Internet, serán castigados con nota de 0.

**Machote de calificación de Tarea:**

|  |  |
| --- | --- |
| Rubro | % |
| **Tabla de Símbolos** | **10** |
| Estructura y Funcionalidad | 10 |
| **Chequeo de Alcances** | **60** |
| Control de Niveles de los identificadores | 10 |
| Control de Redeclaraciones de identificadores | 5 |
| Control de existencia de métodos | 10 |
| Control de cantidad de parámetros | 10 |
| Control de elementos tipo HashContent en HashLiteral | 5 |
| Control de manejo de métodos declarados en arreglos | 7.5 |
| Control de uso de funciones sin retorno en expresiones | 7.5 |
| Reporte de Errores (errores significativos) | 5 |
| **Chequeo de tipos** | **15** |
| Control de tipo en elementos HashContent (int/string) | 10 |
| Reporte de Errores (errores significativos) | 5 |
| **Documentación** | **15** |
| Formato y Contenido del video | 10 |
| No excede mucho la cantidad de tiempo indicado | 5 |
|  | **100** |

**Prueba ejemplo de Análisis Contextual:**

El estudiante será capaz de crear sus propios archivos de prueba que deberá adjuntar con la tarea en donde se contemplen todos los posibles errores contextuales que se puedan verificar.