|  |  |
| --- | --- |
| ***Instituto Tecnológico de Costa Rica***  ***Sede Regional San Carlos*** | Segunda Tarea Programada  Compiladores e Intérpretes |
| Prof. Oscar Mario Víquez Acuña | ***Análisis Contextual*** |
|  | Intérprete 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 intérprete sin declaraciones previas de variables, lo más importante en esta etapa será lo relacionado con la existencia de métodos invocados, así como a las igualdades de tipos en las expresiones que así lo requieran.

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 tipos y alcances de los identificadores de los programas del lenguaje.

La inspección de tipos y 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 por ende no se podría crear un método con el mismo nombre.

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 puesto que serán tratados como tipo neutro, 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 tipos y 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 de tipos.

Para los chequeos (visitas) de expresiones, habrá cuatro tipos de operadores de aritmética binaria de números enteros +, -, \*, / según la precedencia dada por el orden en la gramática. Con argumentos de tipo entero cada operación devuelve valores enteros.

Hay cinco operadores relacionales binarios para números enteros: <, <=, ==, >= y >. Estos operadores aceptarán solo números enteros a excepción del == que puede permitir operandos de cualquier tipo básico (Entero, String y boolean) y en todos los casos devuelven tipo boolean.

Como punto final en esta definición veremos algunas reglas contextuales del intérprete en esta fase DEBE 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. Toda variable declarada, al no tener tipo será tratada para efecto de verificación de tipos como neutra, de manera que no se conocerá su tipo en este momento y por ende si hay algún error en el código, será encontrado y mostrado en la próxima etapa.
2. Lo mismo aplica con las funciones, que al ser llamadas en el contexto de una expresión, no se podrá saber el tipo que devuelven, por lo que serán tratadas como “tipo neutro”.
3. Toda función que no retorne, no tendrá tipo (ni siquiera neutro)
4. Nombres predeclarados de funciones len, first, last, etc.
5. Ningún identificador (variable o función) debe ser declarados dos veces en el mismo ámbito.
6. Ningún identificador podrá ser utilizado en una expresión sin haber sido declarado previamente (en el caso de variables, con “let”).
7. Chequeo de tipos en las expresiones y algunos statements (como el if) considerando los tipos neutros.
8. El número de los parámetros en procedimientos/funciones deben ser igual a los declarados por dicho procedimiento/función.
9. 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)
10. Las indentaciones con elementAccess tanto de arreglos/listas como de hashLiterals, serán con valores enteros solamente (o neutros según sea el caso).
11. El if deberá contener como expresión solo aquellas que devuelvan valores booleanos (o neutros)
12. Noten que declarar una función es un expresión, por lo que es posible declarar expresiones dentro 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. El tipo de la misma será “func” o algo similar y 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 impresa y debe cumplir con el formato establecido por la carrera para las documentaciones de proyectos programados.

**Aspectos Administrativos:**

* La tarea se desarrollará en grupos de máximo dos personas.
* La fecha de entrega será el Viernes 04 de Mayo de 2018 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** | **50** |
| Control de Niveles de los identificadores | 10 |
| Control de Redeclaraciones de identificadores | 10 |
| Control de existencia de métodos | 10 |
| Control de cantidad de parámetros | 10 |
| Reporte de Errores (errores significativos) | 10 |
| **Chequeo de tipos** | **30** |
| Chequeo de tipo en expresiones expresiones | 10 |
| Chequeo de funciones sin retorno en expresiones | 10 |
| Reporte de Errores (errores significativos) | 10 |
| **Documentación** | **10** |
| Formato y Contenido | 5 |
| Ortografía y Gramática | 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.