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

**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, tanto lo relacionado con la existencia de declaraciones previas, como a la igualdad de tipos en las expresiones.

Durante esta fase se debe construir una serie de procedimientos que se encarguen de verificar los tipos y alcances de los identificadores en el programa. Esto procedimientos se realizarán implementando una clase que implemente la interfaz Visitor utilizada en la solución de la primera etapa, según lo discutido en clase.

La inspección de tipos y alcances debe ser auxiliada por una ***tabla de identificadores*** declarados, llevando para cada identificador mínimo su token, su nivel de declaración y su tipo más cualquier otro dato/elemento que consideren. Dicha tabla será una de las estructuras principales para el desarrollo de esta fase.

Tabla se símbolos o tabla de identificadores

Es necesario recordar que estamos ante la creación de un lenguaje con estructura de bloque planos, por lo que se debe realizar la implementación de esta tabla considerando los diferentes niveles de bloque que se pueden presentar.

Es diseño del programador la estrategia a utilizar para referencias cada tipo posible y la cantidad de tablas a crear para los diferentes tipos de identificadores definidos en el lenguaje

La escogencia e implementación de las estructuras de datos es crucial en el desempeño del una herramienta de compilación. En este caso para la creación de la tabla es preciso escoger **la mejor** alternativa de estructura. Como recomendación es mejor utilizar herramientas implementadas en el lenguaje de programación con el objetivo de aprovechar la existencia y eficiencia de las mismas. Se debe documentar y justificar la escogencia de dicha estructura.

La tabla de símbolos debe considerar algunos métodos pre-establecidos (ord, chr y add, del, len) y constantes, como “null”, etc.

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 cinco tipos de operadores de aritmética binaria de números enteros +, -, \*, / y %. Los operadores de suma y resta tienen precedencia más baja que los de multiplicación, división y módulo; además el – puede ser utilizado como operador unario. Con argumentos de tipo entero cada operación devuelve valores enteros.

Hay seis operadores relacionales binarios para números enteros: <, <=, ==, !=, >= y >. Los tipos se manejan como se expresó anteriormente. Estos operadores deben evaluarse para permitir diversos tipos en el == y ¡= y números en los demás. Deben retornar tipo “bool”. Se permite además en la gramática el uso de operadores and (&&) y or (||) que aceptan operadores de tipo “bool” y devuelven tipo “bool”.

Como punto final en esta definición veremos algunas reglas contextuales que nuestro compilador 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. Los arreglos y las clases son llamados tipos referencia, por lo que se deben manejar dentro de la tabla de símbolos con un tipo. Además los arreglos solamente pueden ser de enteros o caracteres; **no de clases**.
2. Se deben manejar las constantes **solamente de tipo** enteras, carácter y String.
3. Para efectos de chequeo de tipos, no será posible asignar arreglos de forma directa (por ejemplos “var\_arreglo1 = var\_arreglo2”).
4. Nombres predeclarados (para efectos del ambiente estándar en la tabla de símbolos): 1) null; valor nulo de una clase o arreglo variable. 2) chr(i); convierte el entero “i” a carácter. 3) ord(ch); convierte el carácter ch en entrero. 4) len(a); retorna el número de elementos de un arreglo/lista 5) add(e); agrega un elemento a una lista 6) del(i) elimina el elemento del index i de la lista, si existe.
5. Ningún identificador debe ser declarados dos veces en el mismo ámbito.
6. Ningún identificador podrá ser utilizado sin haber sido declarado previamente.
7. Chequeo de tipos en las expresiones y algunos statements.
8. El número y tipo 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. La expresión de retorno de una función debe tener el mismo tipo que fue declarado en la función.

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

**Documentación (no mayor a 5 hojas):**

La documentación debe ser impresa y debe incluir las siguientes partes:

* Portada formal.
* Soluciones e implementación. Debe incluir cualquier agregado o cambio en las reglas contextuales definidas en este enunciado.
* Resultados obtenidos.
* Concusiones del trabajo.
* Bibliografía.

**Aspectos Administrativos:**

* La tarea se desarrollará en grupos de máximo dos personas.
* La fecha de entrega será el Jueves 11 de Mayo de 2023 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** | **30** |
| Control de Niveles de los identificadores | 5 |
| Control de Redeclaraciones de identificadores | 5 |
| Control de existencia de métodos | 5 |
| Control de cantidad de parámetros | 6 |
| Control de nombres de campos en clases | 6 |
| Reporte de Errores (errores significativos) | 3 |
| **Chequeo de tipos** | **50** |
| Chequeo de tipos asignaciones | 4 |
| Chequeo de tipo en expresiones lógicas | 7 |
| Chequeo de tipos en métodos preestablecidos (ord, chr,  len) | 7 |
| Chequeo de tipos en Expresiones complejas | 7 |
| Chequeo de tipos en campos de clases | 7 |
| Chequeo de tipos en argumentos de métodos | 7 |
| Chequeo de tipos en los retornos de funciones | 5 |
| Reporte de Errores (errores significativos) | 6 |
| **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.