6-5-2024

Universidad de Guadalajara CUCEI

Practica: Analizador Semántico

Materia: Seminario de solución de problemas de traductores de lenguaje II

Nombre: Nudelstejer Gómez Iván

Código: 218130122

Profesor: Michel Emanuel López Franco

El objetivo de la práctica es implementar un analizador semántico básico para un lenguaje de programación determinado. El análisis semántico es una etapa crucial en el proceso de compilación que verifica la coherencia y la corrección de los aspectos semánticos del código fuente, más allá de su sintaxis.

1. Completa los tres métodos agrega de la tabla de símbolos:

- La tabla de símbolos es una estructura de datos importante utilizada durante el análisis semántico para almacenar información sobre los símbolos encontrados en el código fuente, como variables, funciones, etc.

- Los tres métodos que se deben completar son:

  - `agrega(DefVar *defVar)`: Agrega una definición de variable a la tabla de símbolos.

  - `agrega(DefFunc *defFunc)`: Agrega una definición de función a la tabla de símbolos.

  - `agrega(Parametro *parametros)`: Agrega parámetros de una función a la tabla de símbolos.

2. Define los métodos de validación para las clases que heredan de las expresiones:

- Las clases que heredan de las expresiones representan diferentes constructos del lenguaje de programación, como expresiones aritméticas, asignaciones, etc.

- Se espera que se definan métodos de validación semántica en estas clases para verificar la corrección de los aspectos semánticos asociados con cada tipo de expresión. Por ejemplo, la validación de tipos, el alcance de las variables, etc.

Explicacion:

1. Clase `Nodo`*:

```cpp
class Nodo {
public:
```

```
    char tipoDato;
    static TablaSimbolos *tablaSimbolos;
    static string ambito;

    virtual void validaTipos() {
        tipoDato = 'v';
        if (sig != NULL) sig->validaTipos();
    }
};
```

- Hemos agregado los atributos `tipoDato`, `tablaSimbolos`, y `ambito` a la clase `Nodo`, que son necesarios para el análisis semántico.

- También hemos agregado el método `validaTipos()`, que es virtual y puede ser implementado por las clases derivadas para realizar la validación de tipos.

2. Clase `Semantico`:

```cpp
class Semantico {
public:
    TablaSimbolos *tablaSimbolos;

    Semantico() {
        Nodo::tablaSimbolos = tablaSimbolos = new TablaSimbolos(&listaErrores);
    }

    void analiza(Nodo *arbol) {
        this->arbol = arbol;
```

```
        arbol->validaTipos();
        tablaSimbolos->muestra();
        muestraErrores();
    }
};
```

- La clase `Semantico` se encarga del análisis semántico del árbol de sintaxis abstracta.

- En el constructor, se inicializa la tabla de símbolos y se asigna a `Nodo::tablaSimbolos`.

- El método `analiza(Nodo *arbol)` toma un árbol de sintaxis abstracta como entrada y realiza el análisis semántico llamando al método `validaTipos()` del nodo raíz y mostrando la tabla de símbolos y errores.

3. Método `dimeTipo` en la clase `Tipo`:

```cpp
char dimeTipo() {
    if (simbolo.compare("int") == 0) return 'i';
    if (simbolo.compare("float") == 0) return 'f';
    if (simbolo.compare("string") == 0) return 's';
    if (simbolo.compare("void") == 0) return 'v';
}
```

- Este método devuelve el tipo de dato asociado con un símbolo específico.

4. **Variables globales en el archivo `principal.cpp`**:

```cpp
TablaSimbolos *Nodo::tablaSimbolos = NULL;
string Nodo::ambito = "";
```

- Se inicializan las variables globales `Nodo::tablaSimbolos` y `Nodo::ambito`.

En resumen, este código proporciona la estructura básica y funcionalidades necesarias para realizar un análisis semántico simple, incluyendo el seguimiento de los tipos de datos, la tabla de símbolos y la detección de errores semánticos.

En general, un analizador semántico es una parte crucial de un compilador o intérprete que verifica si el código fuente cumple con las reglas semánticas del lenguaje de programación en cuestión. Mientras que el analizador léxico y el analizador sintáctico se encargan de la estructura y la gramática del código, el analizador semántico se ocupa de aspectos más profundos, como la declaración y uso de variables, tipos de datos, control de flujo, entre otros.

En que consiste la practica:

1. Clase `Nodo` y sus atributos y método: La clase `Nodo` es una clase base que representa los nodos del árbol de sintaxis abstracta (AST). Se le han agregado los atributos `tipoDato`, `tablaSimbolos`, y `ambito`, los cuales son importantes para el análisis semántico. El método `validaTipos()` es una función virtual que se puede implementar en las clases derivadas para realizar la validación de tipos.

2. Clase `Semantico`: Esta clase se encarga de realizar el análisis semántico del árbol de sintaxis abstracta. En su método `analiza`, recorre el árbol de sintaxis y llama al método `validaTipos` de cada nodo para realizar la validación semántica.

3. Clase `Tipo`: Esta clase proporciona funcionalidad relacionada con los tipos de datos en el lenguaje. El método `dimeTipo()` devuelve el tipo de datos asociado con un símbolo específico.

4. Tabla de símbolos: La tabla de símbolos es una estructura de datos que almacena información sobre los símbolos encontrados en el código fuente, como variables, funciones, etc. En la práctica, se han proporcionado métodos para agregar elementos a la tabla de símbolos y realizar consultas sobre ella.

5. Implementación de métodos faltantes en la tabla de símbolos: Se han proporcionado esqueletos de métodos que necesitan ser implementados en la tabla de símbolos para agregar definiciones de variables, funciones y parámetros.

6. Implementación de métodos de validación en clases derivadas: Se espera que implementes métodos de validación semántica en las clases derivadas, como `DefVar`, para validar la semántica de las definiciones de variables.

la práctica consiste en implementar un analizador semántico básico que pueda verificar la corrección semántica del código fuente de un lenguaje de programación dado. Esto implica validar tipos, verificar el alcance de las variables, detectar errores semánticos y mantener una tabla de símbolos actualizada con la información necesaria sobre los símbolos encontrados en el código.

Captura del compilador:

```
Tabla de Símbolos:
Símbolo: x, Tipo: int, Ámbito: global
S mbolo: y, Tipo: float, Ámbito: global
Símbolo: foo, Tipo: funcion, Ámbito: global
Símbolo: bar, Tipo: funcion, Ámbito: global
La variable 'x' no está definida en la función 'foo'
```

Raíz de un árbol del programa que se genera: