Práctica 12

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Compiladores

PRÁCTICA	TEMA	DURACIÓN
12	Analizador Semántico - Scope	3 horas

1. Datos de los estudiantes

- Grupo: 1
- Git Hub: `https://github.com/CrazyDani17/Practica12-Compiladores`
- Integrantes:
  - Guillermo Aleman
  - Marvik Del Carpio
  - Daniel Mendiguri
  - Daniela Vilchez

2. Preguntas

1. Defina una clase o estructura para representar la tabla de símbolos. Recuerde que en la tabla de símbolos usted debe almacenar:

   - Identificadores
   - Funciones o métodos
   - Objetos

   Normalmente un dato de la tabla de símbolos deber ıa contener:

   - Token y lexema.
   - Posición del token.
   - En que función fue creado.
   - Información adicional que usted considere importante.

   Además, esta tabla de simbolos debería tener métodos/funciones para buscar, insertar y eliminar un elemento.

2. Defina una función que permita recorrer un arbol sintáctico desde la raíz a sus hijos. Mientras que se recorre el arbol, deberá buscar/insertar/elimnar cada identificador encontrado en la tabla de símbolos según corresponda y verificar el scope de estos identificadores. Si detecta un error referente al scope, debera mostrar mensajes de error detallando el tipo de error.

Solución:

```
import ll1
class simbolo:
  def __init__(self,t,lex,tp,pos,lin,scope):
    self.token=t
    self.lexema=lex
    self.tipo=tp
    self.posicion=pos
    self.linea=lin
    self.scope=scope

def imprimir_tds(tabla):
  for i in tabla:
    print(i.token,i.lexema,i.tipo,i.posicion,i.linea,i.scope)

n=0
es_yoyo=False
tabla_de_simbolos=[]
ll1.arbol.imprimirArbol(ll1.arbolito)
funcion_actual=None
funcion_actual_aux=None

def comprobar_duplicado(valor,linea):
  for x in reversed(tabla_de_simbolos):
    if x.lexema==valor and (x.tipo=="mision" or x.tipo == "variable"):
      print("Error en lnea "+ str(linea) + ": " + x.lexema + " ya fue declarado")
      break

def comprobar_existencia(valor,tipo,linea):
  encontrado=False
  for x in reversed(tabla_de_simbolos):
    if x.lexema==valor:
      encontrado=True
      break
  if encontrado==False:
    print("Error en lnea "+ str(linea) + ": Llamaste a la " + tipo + " " + valor + ", pero no
        fue declarada")


def verificar_declaracion_variable(arbol):
  comprobar_duplicado(arbol.hijos[1].token.value,arbol.hijos[1].token.lineno)
  tabla_de_simbolos.append(simbolo(arbol.hijos[1].token,arbol.hijos[1].token.value,"variable",
  arbol.hijos[1].token.lexpos,arbol.hijos[1].token.lineno,funcion_actual))

def insertar_parametros(arbol):
  if arbol.elemento=="Y":
    tabla_de_simbolos.append(simbolo(arbol.hijos[1].token,arbol.hijos[1].token.value,
    "variable",arbol.hijos[1].token.lexpos,arbol.hijos[1].token.lineno,funcion_actual))
  for x in arbol.hijos:
    insertar_parametros(x)

def verificar_mision(arbol):
  global funcion_actual
  global funcion_actual_aux
```

```
  comprobar_duplicado(arbol.hijos[1].token.value,arbol.hijos[1].token.lineno)
  tabla_de_simbolos.append(simbolo(arbol.hijos[1].token,arbol.hijos[1].token.value,"mision",
  arbol.hijos[1].token.lexpos,arbol.hijos[1].token.lineno,"global"))
  funcion_actual=arbol.hijos[1].token.value
  funcion_actual_aux=funcion_actual
  if arbol.hijos[3].hijos[0]!="'":
    insertar_parametros(arbol.hijos[3])

def verificar_asignacion_de_variable(arbol):
  comprobar_existencia(arbol.hijos[0].token.value,"variable",arbol.hijos[0].token.lineno)
  tabla_de_simbolos.append(simbolo(arbol.hijos[0].token,arbol.hijos[0].token.value,"asignacion",
  arbol.hijos[0].token.lexpos,arbol.hijos[0].token.lineno,funcion_actual))

def verificar_llamada_de_variable(arbol):
  comprobar_existencia(arbol.hijos[0].token.value,"variable",arbol.hijos[0].token.lineno)

def verificar_llamada_de_mision(arbol):
  comprobar_existencia(arbol.hijos[0].token.value,"mision",arbol.hijos[0].token.lineno)

def eliminar_scope(scope):
  imprimir_tds(tabla_de_simbolos)
  print("")
  for item in reversed(tabla_de_simbolos):
    if item.scope == scope:
      tabla_de_simbolos.pop(tabla_de_simbolos.index(item))
  imprimir_tds(tabla_de_simbolos)
  print("")

def verificar_scope_llama(arbol):
  for subarbol in arbol.hijos:
    if subarbol.elemento=="FuncionPrincipal":
      verificar_scope(subarbol)

def verificar_scope(arbol):
  global funcion_actual
  global n
  global funcion_actual_aux
  global es_yoyo

  if arbol.elemento == "FuncionPrincipal":
    funcion_actual="llama"
    funcion_actual_aux="llama"

  if arbol.elemento == "yoyo":
    es_yoyo=True

  if len(arbol.hijos)>0 and arbol.hijos[0].elemento=="mision":
    n=0
    verificar_mision(arbol)

  if arbol.elemento == "DeclaracionVariable":
    verificar_declaracion_variable(arbol)

  if (arbol.elemento=="SentenciasYoyo" or arbol.elemento == "Sentencias") and
      arbol.hijos[0].elemento=="identificador"and arbol.hijos[1].hijos[0].elemento=="igual":
    verificar_asignacion_de_variable(arbol)
```

```
  if (arbol.elemento=="SentenciasYoyo" or arbol.elemento == "Sentencias" or arbol.elemento
      =="Tx") and arbol.hijos[0].elemento=="identificador"and
      arbol.hijos[1].hijos[0].elemento=="pizquierdo":
    verificar_llamada_de_mision(arbol)

  if arbol.elemento == "Tx" and arbol.hijos[0].elemento == "identificador" and
      arbol.hijos[1].hijos[0].elemento == "')'":
    verificar_llamada_de_variable(arbol)

  if arbol.elemento == "Sentencias" and arbol.hijos[0].elemento=="lizquierdo":
    n=n+1
    funcion_actual= funcion_actual_aux + str(n)

  if arbol.elemento == "lderecho" and n==0:
    if es_yoyo==True:
      es_yoyo=False
    else:
      eliminar_scope(funcion_actual)

  if arbol.elemento == "lderecho" and n>0:
    if es_yoyo==True:
      es_yoyo=False
    else:
      n=n-1
      eliminar_scope(funcion_actual)
      if n!=0:
        funcion_actual= funcion_actual_aux + str(n)
      else:
        funcion_actual= funcion_actual_aux

  for subarbol in arbol.hijos:
    if subarbol.elemento!="FuncionPrincipal":
      verificar_scope(subarbol)

verificar_scope(ll1.arbolito)
print("Scope Llama")
verificar_scope_llama(ll1.arbolito)
print("Final")
imprimir_tds(tabla_de_simbolos)
```

Código de ejemplo

```
mision saludar(numero x, numero y){
  numero y=3
  numero x=2
  {
    numero z=1
    y=1+x
  }
  devuelve z
}
llama(){
  despedir()
  saludar()
  holis()
  numero x=1
  x=1
  yoyo(x<=1){
    x=2
  }
  {
    x=2
  }
  mostrar (x)
}
mision despedir(){
  numero y=2
  mostrar("Adios")
}
```

Evidencia de ejecución