Insper

Lógica da Computação - 2020/1

Roteiro 9 - Simple Calculator **v2.4**

Raul Ikeda - rauligs@insper.edu.br

Entrega: 10/Jun/2020 às 13h30

Nome:

Objetivos

1. Implementar Declarações de Funções
2. Escopo de variáveis

Exemplo de sintaxe:

```php
<?php

function soma($x, $y) {

  function echoes($b) {
    echo $b;
  }

  $a = $x + $y;
  echoes($a);
  return $a;
}

$a = 3;
$b = soma($a, 4);
echo $b;
echoes($a);
$c = soma($b, $a); /* ERROR: Cannot redeclare function echoes */

?>
```

Parte I: Funções

Como exemplificado acima, o PHP permite declarar funções simples e sem declaração de tipo explicitamente. As funções a princípio possuem visibilidade pública a medida em que são declaradas e podem ser usadas em outras funções. É possível declarar uma função dentro de uma função. Não existe *overload* ou *override* de função, o que significa que declarar duas vezes uma função irá disparar um erro.

Tarefas

- Criar uma *branch* a partir da versão v2.3.X.
- Atualizar o EBNF e o DS no GitHub.
- Modificar o *Tokenizer* para realizar a tokenização correta.
- Criar um atributo estático do tipo dicionário na *SymbolTable* para acomodar as funções.

- Criar também os métodos estáticos getter e setter.
- Criar mais 2 nós da AST:
  - *FuncDec*: possui n+1 filhos: n nós do tipo Identifier que representam os argumentos e um nó que recebe o retorno do parseCommand. O *Evaluate()* apenas cria uma variável na *SymbolTable* estática, sendo o nome da variável o nome da função, o valor apontando para o próprio nó *FuncDec* e o tipo será *FUNCTION*.
  - *FuncCall*: possui n filhos do tipo RelExpression - são os argumentos da chamada. O *Evaluate()* vai realizar o verdadeiro *Evaluate()* da *FuncDec*, recuperando o nó de declaração na *SymbolTable* estática, atribuindo os valores dos argumentos de entrada e executando os commandos (último filho).
- Modificar o analisador sintático para interpretar o bloco de declaração corretamente. Colocar a análise da chamada da função dentro da função que analisa atribuições (se você atualizou o DS sabe o porquê disso). Não esquecer da chamada de procedimento (sem retorno).

Parte II: Escopo de variáveis

A ideia principal da implementação de escopo é criar uma nova *SymbolTable* para cada chamada de função. As variáveis declaradas localmente terão escopo reduzido apenas à *SymbolTable* atual. Lembrando que a chama de funções usarão os métodos estáticos.

Tarefas:

- Cada vez que houver uma Chamada de Função, criar uma nova *SymbolTable* e realizar a troca das tabelas na etapa semântica.
- Tomar cuidado com as mensagens de erro.

Base de Testes:

Além do código exemplo acima que deve funcionar conforme o esperado, teste os seguintes elementos:

- Com erros:
  - Chamar a função com número incorreto de argumentos.
  - Declarar uma função duas vezes.
  - Chamar uma função que não existe.
  - Usar uma variável fora de escopo.
- Sem erros:
  - Fazer uma função recursiva.
- Sugira testes adicionais

Para atingir A+

- Prazo: 17/Jun/2020.
- Você deve implementar a geração de código para funções (v3.1).
  - O seu compilador deve realizar todas as etapas de compilação.
  - Ele deve ser hábil suficiente para fazer chamadas recursivas.
  - Gerar um relatório explicando como foi feita a etapa.