Insper

Lógica da Computação - 2020/1

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

Raul Ikeda - rauligs@insper.edu.br

Entrega: 01/Jun/2020 às 13h30

Nome:

Objetivos

1. Implementar Declarações de Funções
2. Escopo de variáveis

Exemplo de sintaxe:

```
int soma(int x, int y) {
  int a;
  a = x + y;
  print(a);
  return(a);
}

int main() {
  int a;
  int b;
  a = 3;
  b = soma(a, 4);
  print(a);
  print(b);
}
```

Parte I: Funções

Como exemplificado acima, o C permite declarar funções simples e com declaração de tipo. Todas as funções a princípio possuem visibilidade pública e podem ser usadas em outras Funções. Não será possível declarar uma função dentro de uma função.

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 mais 2 nós da AST:
  - *FuncDec*: possui 2 filhos: *VarDec* e *Statements*. Os argumentos da declaração devem ser incorporados ao *VarDec*, incluindo o próprio nome da função e seu tipo correspondente. O *Evaluate()* apenas cria uma variável na *SymbolTable* atual, 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 identificador ou expressão - são os argumentos da chamada. O *Evaluate()* vai realizar o verdadeiro *Evaluate()* da *FuncDec*, recuperando o nó de declaração na *SymbolTable*, atribuindo os valores dos argumentos de entrada e executando o bloco (segundo 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).
- Ao criar a raiz da sua AST (um nó do tipo *Statements*), adicione como último filho, uma chamada de função *FuncCall* para a função *main()*. Lembrando que é responsabilidade do Sintático verificar se ela existe ou não.

Parte II: Escopo de variáveis

A ideia principal da implementação de escopo é criar uma nova *SymbolTable* para cada chamada de função. Contudo, Uma chamada de função deve buscar o ponteiro do nó na *SymbolTable* principal, criada no início da execução.

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.
  - Passar argumentos de tipos diferentes.
  - Chamar uma função que não existe.
  - Usar uma variável fora de escopo.
- Sem erros:
  - Fazer uma função recursiva.

1. Sugira testes adicionais

Para ganhar A+

- Prazo: 02/Dez/2019.
- 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.