Construção de Compiladores
Daniel Lucrédio
Especificação e Critérios de notas do Trabalho 4
(Última revisão: dez/2021)

O trabalho 4 (T4) da disciplina consiste em implementar uma segunda parte de um analisador semântico para a linguagem LA (Linguagem Algorítmica) desenvolvida pelo prof. Jander, no âmbito do DC/UFSCar.

Analisador semântico

O analisador semântico deve detectar, além daqueles do T3, outros 5 tipos de erros:

1. Identificador (variável, constante, procedimento, função, tipo) já declarado anteriormente no escopo, **mas agora envolvendo também ponteiros, registros, funções**
    a. O mesmo identificador não pode ser usado novamente no mesmo escopo mesmo que para categorias diferentes
2. Identificador (variável, constante, procedimento, função) não declarado, **mas agora envolvendo também ponteiros, registros, funções**
3. Incompatibilidade entre argumentos e parâmetros formais (número, ordem e tipo) na chamada de um procedimento ou uma função
    a. A quantidade e tipo dos argumentos deve ser exata
        ```
        endereço → ponteiro
        real → real
        inteiro → inteiro
        literal → literal
        logico → logico
        registro → registro (com mesmo nome de tipo)
        ```
4. Atribuição não compatível com o tipo declarado, **agora envolvendo ponteiros e registros**
    a. Atribuições possíveis
        ```
        ponteiro ← endereço
        (real | inteiro) ← (real | inteiro)
        literal ← literal
        logico ← logico
        registro ← registro (com mesmo nome de tipo)
        ```
    b. As mesmas restrições são válidas para expressões, por exemplo, ao tentar combinar um literal com um logico (como em literal + logico) deve dar tipo_indefinido e inviabilizar a atribuição
5. Uso do comando 'retorne' em um escopo não permitido

Ao encontrar um erro, o analisador **NÃO DEVERÁ** interromper sua execução. Ele deverá continuar reportando erros até o final do arquivo. Exemplo:

Entrada:
```
{ leitura de nome e idade com escrita de mensagem usando estes dados }

algoritmo
	declare
		nome: literal
	declare
		idade: inteir

	{ leitura de nome e idade do teclado }
	leia(nome)
	leia(idades)
```

```
        { saída da mensagem na tela }
        escreva(nome, " tem ", idade, " anos.")

fim_algoritmo
```

Saída produzida:

```
Linha 7: tipo inteir nao declarado
Linha 11: identificador idades nao declarado
Fim da compilacao
```

Na dúvida, utilize os casos de teste como guia. Caso uma verificação particular não esteja presente nos casos de teste, não será preciso implementar.

O analisador deve poder ser executado em linha de comando (windows, mac ou linux), com DOIS ARGUMENTOS OBRIGATORIAMENTE:

Argumento 1: arquivo de entrada (caminho completo)
Argumento 2: arquivo de saída (caminho completo)

Exemplo de como seu analisador deve rodar:

```
c:\java -jar c:\compilador\meu-compilador.jar c:\casos-de-teste\arquivo1.txt
c:\temp\saida.txt
```

Como resultado, seu compilador deve ler a entrada de `c:\casos-de-teste\arquivo1.txt` e salvar a saída no arquivo `c:\temp\saida.txt`

NÃO SERÃO ACEITOS programas que imprimem a saída no terminal. É obrigatório salvar no arquivo.

Critérios de avaliação do Trabalho 4 e descontos nas notas

O trabalho 4 deve ser desenvolvido em **grupos de até 3 estudantes** (máximo). A nota do trabalho 4 será composta de 3 parcelas, cada uma valendo de 0 a 10, e com os pesos assim distribuídos:

DE - Documentação externa: 10%
DI - Documentação interna: 10%
CT4 - Casos de teste com erros semânticos: 80%

DE - Documentação externa: deve ser fornecido um arquivo de ajuda, para possibilitar que qualquer pessoa consiga compilar e executar seu trabalho. Deve incluir programas que precisam ser instalados, suas respectivas versões, configurações necessárias, e os passos de execução.

Exemplos de erros comuns na documentação externa e que causarão desconto na nota:
- Só diz como executar, mas não como compilar o programa, ou vice-versa
- Nada foi dito sobre como compilar/interpretar o programa e executá-lo
- Ausência da documentação externa

DI - Documentação interna: o código-fonte (gramática + demais arquivos) devem ser documentados a ponto de possibilitar seu entendimento por parte de outros programadores olhando seu código. Insira comentários explicativos em todos os pontos relevantes do seu código. Nomes de variáveis e funções também fazem parte da documentação interna.

Exemplos de erros comuns na documentação interna e que causarão desconto na nota:
- Pouco documentado (sem explicação do propósito das regras léxicas, sintáticas, semânticas e geração de código, entrada, saída, descrição das variáveis, etc.)
- Descuido nos comentários ou nomes de funções/variáveis pouco indicativos
- Ausência de comentários sobre o processo de compilação
- Nenhuma linha de documentação relevante

CT4 - Casos de teste com erros semânticos: o compilador deve apontar corretamente todos os erros semânticos, da forma exata como nos casos de teste. São **9 casos de teste** nesta categoria.