Terceira Lista de Exercícios de Construção de Compiladores.
Primeiro Semestre de 2003.
Prof. José de Oliveira Guimarães.

Muitos dos exercícios abaixo só podem ser feitos examinando-se o código dos compiladores. Os exercícios restantes poderão ser colocados nas provas da disciplina.

Os exercícios abaixo referem-se ao compilador 8.

1. Explique porque várias classes herdam de Statement. A sua resposta deve justificar porque é melhor utilizar esta herança do que não utilizá-la. Em que o compilador deveria ser modificado se as classes AssignmentStatement, IfStatement, ReadStatement e WriteStatement não possuissem superclasses ? Naturalmente, classes que não herdam de ninguém herdam de Object.

2. O método abaixo pertence à classe StatementList.

```
public void genC() {
  if ( v != null ) {
     Enumeration e = v.elements();
     while ( e.hasMoreElements() )
       ((Statement ) e.nextElement()).genC();
  }
```

Explique como ele funciona. Isto é:
a) qual é o tipo de retorno declarado do método "nextElement" de Enumeration ? Este é o tipo obtido em compilação;
b) em tempo de execução, "e.nextElement()" retornará objetos de que classes ? Alguma delas é "Statement" ? Por que é feita uma conversão do tipo de "nextElement" para "Statement" ?
c) o que fazem os métodos hasMoreElements e nextElement ?

3. Que conferências semânticas são feitas no compilador 8 ?

4. Quais são os possíveis erros na análise léxica no compilador 8 ?

5. Ao inicializar a tabela de palavras-chave da linguagem, inserimos objetos da classe Integer na tabela:

```
keywordsTable = new Hashtable();
keywordsTable.put( "var", new Integer(Symbol.VAR) );
keywordsTable.put( "begin", new Integer(Symbol.BEGIN) );
...
```

Explique porque não inserimos inteiros como Symbol.VAR e Symbol.BEGIN.

6. Por que representamos a declaração de uma variável por objeto da classe Variable e uso da variável em uma expressão por um objeto de VariableExpr ?

7. Quando token é Symbol.IDENT, sabemos que o token corrente é um identificador (variável). Como sabemos qual a *string* correspondente, isto é, qual o nome da variável ?

8. Quando token é Symbol.NUMBER, como descobrimos qual é o valor do número ?

9. Por que depois de encontrado um comentário o analisador léxico chama a si mesmo recursivamente ?

10. No analisador léxico, porque utilizamos um objeto de StringBuffer e não de String para coletar os caracteres de um identificador (ou número) ?

11. Quando uma seqüência de letras é encontrada pelo analisador léxico, como descobrimos se ela é uma palavra-chave ? E como descobrimos o número correspondente a esta palavra-chave ?

12. Por que no método ReadStatement::genC utilizamos `gets` seguido de `sscanf` ao invés de `scanf` ?

13. Por que a classe Statement possui um método genC abstrato ?

Os Exercícios abaixo se referem ao compilador 9.

14. Quais são as conferências semânticas feitas neste compilador ?

15. Na declaração de uma variável, o compilador insere um objeto de Variable na tabela de símbolos. Este objeto representa a variável. Veja o código abaixo.

```
Variable v = new Variable(name);
symbolTable.put( name, v );
```

Por que não inserir simplesmente o nome da variável ? Este objeto de Variable que foi colocado na tabela é utilizado mais tarde em algum lugar ? Quando, no código acima, o objeto apontado por v é inserido em symbolTable, ele não possui tipo. Este tipo é fornecido ao objeto em algum lugar ?

15. O método Compiler::type reconhece o tipo de uma variável :

```
private Type type() {
    Type result;

    switch ( lexer.token ) {
        case Symbol.INTEGER :
          result = Type.integerType;
          break;
        case Symbol.BOOLEAN :
           ...
    }
}
```

Por que este método não cria e retorna um objeto de IntegerType, BooleanType e CharType ?

16. Explique porque é necessária uma classe Type superclasse de IntegerType, BooleanType e CharType.

17. Por que a classe CompilerError possui uma variável de instância da classe Lexer ?

18. Por que Expr e suas subclasses precisam ter um método getType ?

19. Faça o método getType de CompositeExpr.

20. Quais são os novos erros léxicos que este compilador pode sinalizar ?

21. Por que a classe Type declara as variáveis booleanType, integerType e charType como “static” ?

22. A classe UnaryExpr possui um switch com três cases no método genC. Isto indica que esta classe está desempenhando três papéis diferentes. Não seria melhor desdobrá-la em três classes diferentes ?

Os Exercícios abaixo se referem ao compilador 10.

23. A tabela de símbolos possue agora símbolos locais e globais. Diga quais símbols são locais e globais. Quando cada símbolo é inserido na tabela ? Quando é retirado ?

24. O programa deve ter um procedimento chamado “main” sem parâmetros. Como o compilador confere se o programa tem este procedimento ?

25. Teria algum problema se o compilador permitisse um nome de procedimento igual ao nome de uma função ?

26. Pode uma variável ter o mesmo nome que o procedimento/função onde ela está ?

27. Que conferências semânticas são feitas neste compilador ?

28. Por que precisamos da classe Function e FunctionCall ?

29. Por que FunctionCall deve herdar de Expr ?

30. Por que Parameter herda de Variable ? Cite um lugar onde isto é necessário.

31. Por que ProcedureCall deve herdar de Statement ?

32. Onde é usado UndefinedType ?