MESTRADO INTEGRADO EM
ENGENHARIA INFORMÁTICA E DE COMPUTADORES

EIC0028-2S – COMPILADORES

Teste - 8 de Abril de 2010 *Duração: 1 hora e 30 minutos*

Nota: Na parte I as respostas erradas têm cotação negativa. Numa pergunta com 4 alternativas, uma resposta errada tem uma cotação negativa igual a 1/3 da cotação da pergunta. Numa pergunta com 2 alternativas, a resposta errada tem uma cotação negativa igual à cotação da pergunta. As perguntas não respondidas têm cotação 0.

Nome: _____ Número: _____

PARTE I (11 valores)

1. Análise Lexical

1.1 [1val] Num compilador, a análise lexical é responsável por:

- ☒ Verificar se existem erros lexicais e caso não existam deve fornecer a sequência de *tokens* que representa o programa a compilar.
- ☐ Verificar se existem erros lexicais e sintácticos e caso não existam deve fornecer a sequência de tokens que representa o programa a compilar.
- ☐ Verificar apenas se existem erros lexicais.
- ☐ Verificar se existem erros lexicais e caso não existam deve fornecer uma árvore que representa o programa a compilar.

1.2 [1val] A implementação de um analisador lexical requer:

- ☐ A utilização de uma ferramenta que gera código para implementar *parsers*.
- ☒ A definição dos tokens e a verificação desses tokens no programa a compilar.
- ☐ A construção de uma tabela de símbolos que permita verificar se os lexemas existentes no programa a compilar estão de acordo com os símbolos nessa tabela.
- ☐ Nenhuma das outras alíneas está correcta.

2. Análise Sintáctica

2.1 [1val] Um analisador sintáctico descendente pode ser implementado considerando:

- ☒ Que cada variável da gramática origina uma função/procedimento.
- ☐ O uso de um autómato de piha.
- ☐ A utilização de autómatos finitos.
- ☐ A utilização de expressões regulares de forma a resolver possíveis conflitos.

2.2 [1val] A análise sintáctica é responsável por:

- ☐ Verificar se o programa respeita as regras lexicais e gramaticais da linguagem de programação, e caso respeite deve gerar a AST que o representa.
- ☐ Verificar se o programa se encontra correctamente definido em termos da linguagem de programação utilizada.
- ☐ Criar uma representação intermédia e anotá-la com possíveis erros.
- ☒ Verificar se o programa respeita as regras gramaticais da linguagem de programação, e caso respeite deve gerar a AST que o representa.

2.3 [1val] Indique a opção mais correcta quando estamos a comparar a análise lexical e a análise sintáctica:

- ☐ Ambas verificam erros no programa fonte e podem ser implementadas com as mesmas técnicas;

- ☐ Ambas verificam se a sequência de símbolos presente no código fonte está de acordo com a linguagem de programação utilizada e ambas podem fornecer ASTs que representam o programa;
- ☒ A análise sintáctica analisa a sequência de tokens fornecida pela análise lexical;
- ☐ Tal como a análise sintáctica, a análise lexical pode ser eficientemente implementada utilizando analisadores gramaticais;

3. Análise Semântica

3.1 [1val] Indique a opção mais correcta:

- ☐ A análise semântica deve ser realizada recorrendo a uma representação intermédia de nível alto.
- ☒ A tabela de símbolos é utilizada na análise semântica para consultar a informação sobre os identificadores existentes no programa.
- ☐ A análise semântica deve fornecer uma AST anotada com a informação sobre os tipos das variáveis.
- ☐ Alguns erros semânticos são fruto de problemas ocorridos na análise sintáctica.

3.2 [1val] Indique a opção mais correcta:

- ☒ A semântica de uma linguagem não altera as etapas de análise lexical e sintáctica de um compilador.
- ☐ Parte da análise semântica é realizada durante a análise sintáctica.
- ☐ Num compilador que utiliza uma representação intermédia baseada em ASTs, a análise semântica não deve alterar essas ASTs.
- ☐ A análise semântica fornece sempre uma representação intermédia de nível alto à próxima etapa do compilador.

4. [4val] Indique se cada uma das seguintes afirmações é verdadeira ou falsa:

V	F	
☐	☒	Uma gramática com recursividade à esquerda pode ser implementada sem modificações com um analisador sintáctico descendente desde que se utilize um valor adequado para o *lookahead*.
☒	☐	Existem gramáticas livres de contexto (*context free grammars*) que não podem ser implementadas por um analisador sintáctico descendente.
☒	☐	Existem gramáticas que podem ser implementadas por um analisador sintáctico descendente utilizando um valor de *lookahead* igual a zero.
☐	☒	Numa gramática de expressões aritméticas utiliza-se as precedências das operações para eliminar possíveis ambiguidades.
☐	☒	A separação entre análise lexical e sintáctica é apenas utilizada para explicar os conceitos já que na prática elas são implementadas em conjunto.
☐	☒	A construção da AST (*abstract syntax tree*) deve ter em conta os resultados da análise semântica.
☐	☒	As tabelas de símbolos devem reflectir a hierarquia presente nas ASTs.
☒	☐	Existem situações em se justifica a implementação de *parsers* sem utilizar ferramentas que geram o código dos mesmos.
☐	☒	A representação intermédia de nível alto deve reflectir a forma planar com que as variáveis são armazenadas em memória.
☒	☐	A representação intermédia de nível alto deve manter as estruturas de controlo existentes no programa a compilar.