COMPILADORES

Prova Final – 28/03/2019 – Prof. Marcus Ramos

Questão 1 (1 ponto): Em que consiste uma gramática léxica? Como obter a mesma a partir da gramática que descreve a sintaxe livre de contexto de uma linguagem de programação?

Uma gramática léxica é uma gramática que descreve a linguagem formada pelos tokens da linguagem-fonte. Normalmente esta é uma linguagem regular. A mesma pode ser obtida reunindo-se apenas as regras que tratam especificamente da geração dos tokens, como <id> e <literal>, além de separadores e símbolos avulsos.

Questão 2 (1 ponto): Explique como o método recursivo descendente mapeia uma gramática livre de contexto em um analisador descendente. Qual o requisito que deve ser observado pela gramática para que a aplicação do método seja bem-sucedida?

O método recursivo descendente mapeia cada não-terminal da gramática num método recursivo codificado em alguma linguagem de programação (como Java por exemplo). O conjunto dos métodos opera para reconhecer o programa-fonte, da esquerda para a direita. O lado direito das regras é mapeado em combinações de comandos condicionais (para união), iterativos (para fechamento), chamada de outro método (para símbolo não-terminal) ou simplesmente avanço para o token seguinte (para símbolo terminal). Para que a aplicação do método seja bem sucedida, é importante que a gramática seja LL(1).

Questão 3 (1 ponto): Discorra sobre a natureza das mensagens de erro que são geradas pelo compilador ao longo das suas diversas fases.

Mensagens de erro podem ser emitidas durante a análise léxica (caracter inválido), análise sintática (token inválido), análise de contexto (nome já declarado, nome não declarado ou tipos incompatíveis) e também na geração de código (no caso de máquinas de registradores, se faltar registrador para armazenar resultados intermediários). Em qualquer caso, as mensagens devem ser claras e indicar com máxima precisão o ponto onde o erro ocorreu, incluindo sugestões para a correção do mesmo. Deve-se ainda evitar a ocorrência de erros em cascata.

Questão 4 (1 ponto): Justifique a necessidade de uma fase de análise de contexto nos compiladores modernos.

A necessidade decorre do fato de a gramática usada para formalizar a linguagem-fonte não conseguir capturar as dependências de contexto da mesma, ficando restrita à parte livre de contexto da linguagem. Assim, o analisador sintático aceita como válidos programas que depois precisam passar por um novo filtro (o analisador de contexto) antes de seguir para a geração de código.

Questão 5 (1 ponto): Cite três aplicações da estrutura de dados “pilha” na construção de compiladores e também na execução do código-objeto.

Na avaliação de expressões em tempo de execução (i); na estrutura de frames usada no ambiente de execução para representar o estado do programa (ii) e na estrutura da tabela de símbolos usada para armazenar os identificadores do programa-fonte durante a subfase de identificação da análise de contexto (iii).

Questão 6 (1 ponto): Nem todos os erros do programa-fonte podem ser detectados pelo compilador estaticamente. O que o compilador faz nos casos em que isto não é possível? Exemplifique.

O compilador gera código extra que permite que a verificação ocorra no momento apropriado, em tempo de execução. É o caso, por exemplo, da verificação de índices de agregados homogêneos, que devem obedecer limites superior e inferior.

Questão 7 (1 ponto): Qual o papel e a importância do uso do padrão de projeto Visitor na construção de processadores de linguagens em linguagens orientadas à objetos?

O Visitor permite a separação física entre a descrição de uma estrutura de dados e as operações que são executadas sobre a mesma. Isto garante uma maior independência entre as fases de um compilador, aumentando a coesão das mesmas e diminuindo o seu acoplamento.

Questão 8 (1 ponto): Qual a diferença entre uma variável global e uma variável local com o atributo “static”?

A única diferença refere-se ao escopo, que no caso da variável global é o programa inteiro (exceto nos blocos onde houver redeclaração) e no caso das variáveis locais é apenas o bloco onde a mesma foi declarada. Fora isso, os dois tipos de variável são alocadas estaticamente, o que significa dizer que o tempo de vida delas coincide com o tempo total de execução do programa.

Questão 9 (1 ponto): Por que, no projeto e no livro, damos preferência para a geração de código no modelo “pilha”, em detrimento do modelo “máquina de registradores”?

O modelo “pilha” produz um gerador de código mais simples ao mesmo tempo em que o resultado (o código gerado) deixa um pouco a desejar em termos de eficiência. Já o modelo “máquina de registradores” é mais complexo de ser utilizado mas produz um código mais eficiente.

Questão 10 (1 ponto): O que é e para que serve um “code template”? Exemplifique.

Um “code template” é uma forma concisa de descrever a semântica de uma estrutura sintática da linguagem-fonte (comando, declaração etc) em termos da semântica das instruções da linguagem-objeto, e também de outros “code templates”. Ele é usado para guiar a geração de código do compilador, enquanto a AST é visitada. Conforme o nó visitado, aplica-se o “code template” correspondente.