UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
I.C.M.C.

SCC-206 INTRODUÇÃO À COMPILAÇÃO
Prof. Thiago A. S. Pardo

LISTA DE EXERCÍCIOS

Atenção: há mais exercícios nos livros recomendados da disciplina

1) O que é um compilador?

2) Qual a diferença entre um interpretador e um compilador?

3) Liste as principais fases de um compilador e descreva brevemente suas funcionalidades.

4) Qual a diferença entre as notações BNF e EBNF?

5) Liste as características que diferenciam as gramáticas da hierarquia de Chomsky. Qual o tipo de linguagem das linguagens de programação? Justifique e exemplifique.

6) Quais são as classes de tokens que um analisador léxico reconhece? Exemplifique.

7) Para que serve a tabela de palavras reservadas em um compilador?

8) Qual a função das rotinas de tratamento de erros em um compilador? Que tipos de erro podem ser detectados durante a análise léxica? Exemplifique.

9) Construa um autômato finito para o reconhecimento de números reais.

10) Implemente uma sub-rotina correspondente ao autômato anterior.

11) Sejam as gramáticas:

G1 = (Vn, Vt, P, S), em que P consiste das regras

S → A | B | λ
A → A+B | A-B | 1 | 2 | 3 | λ
B → A | C
C → (A)

G2 = (Vn, Vt, P, S), em que P consiste das regras

S → λ | abA | abB | abC
A → aSaa | b
B → bSbb | c
C → cScc | d

G3 = (Vn, Vt, P, S), em que P consiste das regras

S → bAb
A → CB | a
B → Aa
C → c | λ

(a) Descreva as linguagens de G1, G2 e G3, dando alguns exemplos de cadeias possíveis para cada linguagem.

(b) Calcule os conjuntos primeiro e seguidor para todo não terminal.

12) Simule a saída da análise léxica para o programa abaixo:

```
program nome1;
{exemplo 1}
var a, a1, b: integer;
begin
read(a, b);
a1:= a + b;
while a1>a do
begin
write(a1);
a1:= a1-1;
end;
if a<> b then write(a);
end.
```

13) As gramáticas abaixo são LL(1)? Transforme as que não são.

(a) S→ABc
A→a|λ
B→b|λ

(b) S→Ab
A→a|B|λ
B→b|λ

(c) S→ABBA
A→a|λ
B→b|λ

(d) S→aSe|B
B→bBe|C
C→cBe|d

14) Construa os grafos sintáticos para os trechos da gramática abaixo, em número reduzido quando adequado.

<corpo> ::= <dc> begin <comandos> end
<comandos> ::= <cmd> ; <comandos> | λ
<cmd> ::= read (<variaveis>) |
write (<variaveis>) |
while <condicao> do <cmd> |
if <condicao> then <cmd> <pfalsa> |
ident := <expressao> |
ident <lista_arg> |
begin <comandos> end

15) Para os grafos anteriores, produza os procedimentos recursivos da análise sintática descendente preditiva recursiva com o tratamento adequado de erros.

16) Qual a diferença entre a análise sintática descendente preditiva recursiva e a não recursiva? Por que são chamadas 'preditivas'?

17) Considere a gramática abaixo:

E → E + T | T
T → T * F | F
F → (E) | id

(a) Verifique se é LL(1). Se não for, transforme-a.

(b) Faça a análise sintática descendente preditiva não recursiva para a cadeia id*id+id

18) Altere a gramática da LALG para que ela reconheça o comando "case".

19) Quais as vantagens da separação da análise léxica da sintática?

20) Explique brevemente o funcionamento da análise 'empilha e reduz'.

21) Observe a gramática abaixo:

S --> (L) | a
L --> L , S | S

(i) que linguagem essa gramática gera?
(ii) construa uma derivação mais à direita para (a, (a, a)) e mostre o *handle* de cada forma sentencial à direita
(iii) mostre os passos de um analisador sintático empilha e reduz correspondentes à derivação mais à direita de (a)

22) A tabela abaixo mostra as relações de precedência de operadores para a gramática anterior.

	a	**(**	**)**	**,**	**$**
a			>	>	>
(	<	<	=	<	
)			>	>	>
,	<	<	>	>	
$	<	<			

Usando essas relações, mostre os passos da análise sintática das sentenças abaixo:

(a, a)
(a, (a, a))
(a, ((a, a), (a,a)))

Qual a diferença entre a análise sintática descendente e ascendente em termos de eficiência e de linguagens que cobrem?

23) Quais as vantagens em se usar analisadores sintáticos LR?

24) Fale brevemente sobre as características das três técnicas para construir tabelas sintáticas: SLR, LALR e LR canônico.

25) Construa uma tabela sintática SLR para a gramática abaixo

E --> E sub R | E sup E | { E } | c
R --> E sup E | E

26) Discorra sobre as funções da análise semântica e as estruturas de dados que utiliza.

27) Supondo que o programa abaixo está sendo compilado, monte e manipule passo a passo a tabela de símbolos até o encerramento da compilação do programa.

```
program p;
var a: real;
var b: integer;
procedure nomep(x: real);
var a, c: integer;
begin
read(c, a);
if a<x+c then
begin
a:= c+x;
write(a);
end
else c:= a+x;
end;
begin {programa principal}
read(b);
nomep(b);
end.
```

28) Diga o que são descritores e como eles são usados na tabela de símbolos. Dê exemplos de descritores para os elementos do programa do exercício anterior.

29) Considerando a análise semântica, escreve a gramática de atributos completa (com tratamento de erros e manipulação da tabela de símbolos) para o trecho abaixo de gramática.

<corpo> ::= <dc> begin <comandos> end
<comandos> ::= <cmd> ; <comandos> | λ
<cmd> ::= read (<variaveis>) |
write (<variaveis>) |
while <condicao> do <cmd> |
if <condicao> then <cmd> <pfalsa> |
ident := <expressao> |
ident <lista_arg> |
begin <comandos> end

30) O que são atributos sintetizados e herdados? Dê exemplos de trechos de gramáticas de atributos (podem ser hipotéticas) em que eles ocorrem.

31) O que é uma gramática S-atribuída?

32) Em termos de implementação, como atributos herdados e sintetizados podem ser implementados em um compilador dirigido pela sintaxe?

33) Fale sobre as formas de se computar os atributos de uma gramática de atributos de forma consistente. Dê exemplos.

34) Por que a gramática de atributos é um dos formalismos mais usados? Seu uso é restrito à análise semântica?

35) Qual a vantagem em se considerar uma máquina hipotética para a geração de códigos?

36) Considerando que 'D' é uma pilha de dados e 's' é um ponteiro para as posições desta pilha, mostre a interpretação correspondente para as instruções da máquina hipotética abaixo:

CRCT k
SOMA
DIVI
CONJ
CPMA
ARMZ n
DSVI p
LEIT
IMPR
INPP
DESM m

Explique e dê exemplos de como esses códigos são gerados a partir dos grafos sintáticos da LALG.

37) Utilizando o conjunto de instruções da máquina hipotética, gere o código correspondente para os programas abaixo:

(a)
```
program exemplo1;
var a, b: integer;
begin
read(a,b);
write(a,b);
end.
```

(b)
```
program exemplo2;
var a: real;
var b: integer;
procedure nomep(x: real);
var a, c: integer;
begin
read(c, a);
if a<x+c then
begin
a:= c+x;
write(a);
end
else c:= a+x;
end;
begin {programa principal}
read(b);
nomep(b);
end.
```

(c)
```
program exemplo3;
var a, b: integer;
var c: real;
begin
read(a,b);
c:=5;
while a<b do
begin
a:=a+1;
c:=c*a;
write(c);
end.
```

38) Discorra sobre os tipos principais de ambientes de execução e sobre para quais situações são mais adequados.

39) Nos ambientes de execução, quais as diferenças entre vinculação de controle e vinculação de acesso?

40) Fale das principais técnicas de geração de código. Quais as diferenças entre as técnicas para geração de código intermediário e geração de código objeto?
41) Como a gramática de atributos pode ser usada para gerar código?

42) Por que o termo “otimização de código” é um termo enganoso?

43) Cite algumas das principais fontes de otimização de código.

44) Quais as diferenças entre o P-código e o código de 3 endereços? Dê exemplos.

45) Dada a gramática de atributos abaixo, faça a geração de código para a expressão (x=x+3)+4

Regra Gramatical	Regras Semânticas
$exp_1 \rightarrow$ **id** = exp_2	exp_1 .*name* = exp_2 .*name* exp_1.*tacode* = exp_2 .*tacode* ++ **id**.*strval* || "=" || exp_2 .*name*
$exp \rightarrow aexp$	*exp* .*name* = *aexp* .*name* *exp* .*tacode* = *aexp* .*tacode*
$aexp_1 \rightarrow aexp_2$ + *fator*	$aexp_1$.*name* = *newtemp*( ) $aexp_1$.*tacode* = $aexp_2$.*tacode* ++ *fator* .*tacode* ++ $aexp_1$.*name* || "=" || $aexp_2$.*name* || "+" || *fator* .*name*
aexp → *fator*	*aexp* .*name* = *fator* .*name* *aexp* .*tacode* = *fator* .*tacode*
fator → (*exp*)	*fator* .*name* = *exp* .*name* *fator* .*tacode* = *exp* .*tacode*
fator → **num**	*fator* .*name* = **num**.*strval* *fator* .*tacode* = ""
fator → **id**	*fator* .*name* = **id**.*strval* *fator* .*tacode* = ""

46) Considerando a linguagem de uma calculadora simples (em que são possíveis expressões com números inteiros e reais e os operadores tradicionais de soma, subtração, multiplicação e divisão, além do sinal de igual), escreva uma gramática para essa linguagem e construa um compilador completo para ela, incluindo todas as etapas e as decisões de projeto pertinentes.