UNIVERSIDADE DE SÃO PAULO - ICMC
Departamento de Ciências de Computação e Estatística
SCC 605 - Teoria da Computação e Compiladores - 1° Sem /2010
PROFa: Sandra Aluisio
WIKI: http://wiki.icmc.usp.br/index.php/SCC-605

Primeiro Trabalho Prático (em grupo)
Entrega: 12/4/2010

Utiliza os conhecimentos de gramáticas formais, notações para gramáticas livres de contexto (BNF e EBNF) e Expressões Regulares

PARTE 1) (Valor 2.0) Definir formalmente **as extensões** do grupo à gramática Pascal Simplificado (PS) em notação **EBNF** e também os 3 elementos da gramática do PS (já estendida) formalmente. **ENTREGAR IMPRESSO.**

Lembrem que gramática é uma quádrupla ordenada composta de: G = (Vn, Vt, P, S). O conjunto P já foi dado, faltam Vn, Vt, e S.

Embora comentários não façam parte da linguagem PS e, portanto, não devem ser especificados na gramática, permitam o seu uso **nos programas** Pascal Simplificado. **Eles têm a forma: (* ..... *) e { ... }**

PARTE 2) (valor 8.0) Esta parte é formada por 3 tarefas, todas com entrega **IMPRESSA**.

2.1) Definam formalmente (quádrupla) a gramática de bc em notação BNF.

Antes do trabalho, sugiro que usem o utilitário bc.

bc é um utilitário do shell do unix. É um processador interativo (interpretador) para uma linguagem muito parecida com C, mas fornece precisão aritmética ilimitada. Vejamos suas características:

L significa uma letra minúscula (a-z),

E significa uma expressão: um operando, ou a combinação de operandos e operadores,

C um comando.

1. Comentários: incluídos em /* e */

2. Identificadores (que são operandos):

* variável simples: **L**
* variável indexada: **L [E]**
* A palavra **scale**.

Outros operandos:

* números arbitrariamente longos, com sinal e ponto decimal opcionais. Ex: +2222222222223, -22. , +.4 , 2.33333333333333333, -2.333
* **(E)**: expressão parentizada
* **sqrt (E)**: raiz quadrada
* **length (E):** números de dígitos decimais significativos
* **scale (E):** números de dígitos à direita do ponto decimal
* **L (E , E , ... , E):** chamada de função definida ou pré-definida

3. Operadores (da maior para menor prioridade):

++ --	prefix (unários)‡	Aparecem uma vez
++ --	posfix (unários) ‡	Aparecem uma vez
^	potenciação	Direita para Esquerda
* / %	% é o resto	Esquerda para Direita
+ -		Esquerda para Direita
== <= >= != < >		Aparecem uma vez
= =+ =- =* =/ =% =^	atribuição	Direita para esquerda

‡ **OBS: os operadores ++ e – (pré e pós-fixos) se aplicam SOMENTE a variáveis e não podem co-ocorrer numa expressão: ++a--, --a++,--a--, e ++a++ não são permitidos.**

4. Comandos

```
E
{ C ;... ; C }
if ( E ) C
while ( E ) C
for ( E ; E ; E ) C
break
quit
```

5. Definição de funções

define L (L , L , ..., L)
{ auto L , L, ... , L
C ; C ; ... C
return (E) }

6. Funções pré-definidas

```
s(x)  sine
c(x)  cosine
e(x)  exponential
l(x)  log
a(x)  arctangent
j(n,x)bessel function
```

Obs:
1) Os argumentos das funções são **opcionais** e passados por valor. A definição das variáveis locais (auto) é **opcional**.
2) O valor de um comando que é uma expressão é impresso, a menos que o operador principal seja a atribuição.
3) Ponto e vírgula ou \n podem separar comandos
4) A mesma letra pode ser usada por um array, função, ou variável simples simultaneamente.
5) Todas as variáveis são globais ao programa.
6) bc aceita comandos **ou** definição de funções até ser digitado o comando **quit** que termina a execução do comando **bc**.

Exemplo 1:

```
$ bc
2 + 2
4
3/4
0
scale = 2
3/4
.75
quit
$
```

Exemplo 2: Definição de função exponencial

```
scale = 20
define e(x){
    auto a, b, c, i, s
    a = 1
    b = 1
    s = 1
    for(i=1; 1==1; i++){
        a = a*x
        b = b*i
        c = a/b
        if(c == 0) return(s)
        s = s+c
    }
}
```

Exemplo 3: Chamada da função

```
for(i=1; i<=10; i++) e(i)
```

2.2) Utilizem o JFLAP para editar a gramática do bc e para rodar os 3 parsers disponíveis, com exemplos de programas bc. Imprimam tanto a gramática (item a) como as saídas dos 3 parsers (usem screemdumps), seus programas exemplos e comentários.

a) Para facilitar o uso com o JFLAP, que não reconhece **if**, **define** ou **while** como um único símbolo do VT, usem apenas a primeira letra dos terminais (por exemplo, **if** será **i**, define será d e **while** será **w**, etc.). Também, troquem os não-terminais entre parênteses angulares por letras do alfabeto latino maiúsculas (por exemplo, <comandos> por ser C, <Definição de funções> pode ser D, etc.

b) Usem todos os parsers do JFLAP: **parser força bruta, LL(1)** e o **SLR(1)** para analisarem os casos de teste.

c) Comecem com programas simples para o **teste**. Façam vários testes, mostrando exemplos de programas (sentenças) que **pertencem à linguagem** e que **não pertençam**.

d) Comentem sobre os possíveis problemas e o tempo utilizado nas 3 análises. Se houver problema de não reconhecimento do exemplo com algum dos parsers, reportem estes problemas.

2.3) Façam Expressões Regulares para denotar os elementos de VT de bc. Entreguem impressas. Os elementos do Vt compreendem:

- identificadores,
- os nomes dos símbolos simples (por exemplo, no bc temos: **+ ; { }** etc.),
- os nomes dos símbolos compostos (no bc temos**:** ++ -- == etc.),
- constantes inteiras e constantes reais como definidas na especificação
- o nome das palavras-chaves como if, while, quit, etc.