UFMG – ICEX – DCC - Bacharelado em Ciência da Computação
Disciplina: Programação Modular
Prof.: Marco Túlio de Oliveira Valente - mtov@dcc.ufmg.br

**Trabalho Prático N° 1 – C++**

Data de Entrega: 26/09/2011 - Grupos de dois alunos (no máximo)
Valor: 10 pontos

**Parte I – 6 pontos**

Implemente um interpretador para a linguagem **tiny**, que possui a seguinte sintaxe:

<programa> → <lista_comandos> "endp"

<lista_comandos> → <comando> ";"
→ <comando> ";" <lista_comandos>

<comando> → "writeStr" "(" <string> ")"
→ "writeVar" "(" variavel ")"
→ "writeln"
→ "read" "(" <variavel> ")"

→ <variavel> ":=" <expressao>

→ "for" <variavel> ":= "<expressao> "to" <expressao> "do"
<lista_comandos>
"end"
→ "while" <expressao> "do"
<lista_comandos>
"end"

→ "if" <expressao> "then" <lista_comandos> "end"
→ "if" <expressao> "then" <lista_comandos> "else"
<lista_comandos> "end"

**Importante:** a implementação do interpretador deverá seguir o padrão de projeto Interpretador (ver livro GoF) e utilizar devidamente os conceitos e princípios de OO.

**Observações sobre a Linguagem tiny:**

- Variáveis possuem apenas uma letra.
- Todas variáveis são do tipo double.
- Variáveis são declaradas automaticamente com o valor zero.
- Expressões podem envolver operadores aritméticos (+,-,*,/), lógicos (and, or, not) e relacionais (>, <. >=, <=, <>, =). Existe ainda a função pré-definida sqrt.
- Expressões são sempre digitadas em notação pós-fixada.
Exemplo: (- b + sqrt (d)) / (2 * a) => 0 b – d sqrt + 2 a * /
- Programas tiny de teste serão fornecidos brevemente

**Exemplo #1**

```
writeStr("Informe o tamanho dos lados do triangulo");
writeStr("Lado 1= ");
read(a);
writeStr("Lado 2= ");
read(b);
writeStr("Lado 3= ");
read(c);
if a b =  b c = and then
   writeStr("O triangulo eh equilatero");
else
  if a b = a c = b c = or or then
     writeStr("O triangulo eh isosceles");
  else
     writeStr("O triangulo eh escaleno");
  endif
endif
endp
```

**Exemplo # 2**

```
writeStr("Entre com o numero de linhas: ");
read(n);
if n  0 < then
   writeStr("numero de linhas menor que 0");
else
  a:= 1;
  while  a  n <= do
    i:= 1;
    while i n a - <=  do
      writeStr(" ");
      i:= i 1 +;
    endw
    i:= 1;
    while i 2 a * 1 - <= do
      writeStr("*");
      i:= i 1 +;
    endw
    writeln;
    a:= a  1 +;
  endw
endif
endp
```

**Parte II – 4 pontos**

Acrescente na linguagem **tiny** chamada de procedimentos, conforme definido abaixo:

- Um programa em tiny passará a ser uma lista de procedimentos.

- Um procedimento será definido como em:

  **proc** <nome> ( <lista_parâmetros_formais> ) <lista_comandos> **endproc**

  onde <nome> é o nome do procedimento (uma string); <lista_parâmetros_formais > é uma lista de parâmetros formais, separados por vírgula; <lista_comandos> é uma lista de comandos (os mesmos já implementados no primeiro trabalho)

- Uma chamada de procedimento tem a seguinte sintaxe:

  **call** nome (<lista_parâmetros_chamada>;

  onde <nome> é o nome do procedimento a ser chamada e <lista_parâmetros_chamada> é uma lista de expressões, separadas por vírgulas

- Procedimentos podem declarar variáveis locais, definidas com a seguinte sintaxe:

  **local** <lista_de_variáveis>;

  onde <lista_de_variáveis> é uma lista de variáveis separadas por vírgulas. Existe, no máximo, um único comando **local** por procedimento, logo na primeira linha do mesmo.

- Além de variáveis locais, um programa tiny passará a permitir a declaração de variáveis globais, definidas da seguinte forma:

  **global** <lista de variáveis>;

  onde <lista_de_variáveis> é uma lista de variáveis separadas por vírgulas. Existe, no máximo, um único comando **global** por programa, logo na primeira linha do mesmo.

- O uso de uma variável, sem sua respectiva declaração como local ou global, passa a ser um erro, detectado em tempo de execução.

- A execução de um programa tiny inicia por um procedimento de nome main.

- Procedimentos podem ser recursivos.

- Parâmetros são sempre passados por valor.

**Exemplo #3**

```
global z;

proc imprime_asteriscos (n)
  local i;
  for i:= 1 to n do
    writeStr("*")
  endf
  writeln;
endproc

proc imprime_dolar (n)
  if n 0 > then
     writeStr("$");
     call imprime_dolar (n 1 -);   /* chamada recursiva */
  else
    writeln;
  endif
endproc

proc soma(x, y)
  z:= x  y +;     /* atribuição a uma variável global */
endproc

proc main()
  local x,y;
  read(x);
  call imprime_asteriscos(x);
  read(y);
  call imprime_dolar(y);
  call soma(x,y);
  call writeVar(z);
endproc
```

**Parte III – 3 pontos (extras)**

Modifique as partes I e II para aceitar expressões na forma infixada.

**Formato de Entrega:**

- Demonstração em laboratório
- Entrega de todo código fonte via Moodle