MAC122 – Princípios de Desenvolvimento de Algoritmos
Segundo Semestre de 2013 – BMAC – IMEUSP – Prof. Marcilio
EXERCÍCIO PROGRAMA I – Entregar 22/Set/2013

Compilador e interpretador de expressões aritméticas

Dada uma sequencia de comandos de atribuição, fazer um programa que calcule e mostre o valor das variáveis que aparecem na expressão após o cálculo.
Os comandos são de dois tipos:
<var>=<número> ou
<var>=<expressão aritmética>
As expressões aritméticas possuem operadores (**^**, *****, **/**, **+**, **-**, **=**), variáveis e parêntesis. Não possuem números.
O formato da expressão é livre, ou seja, pode ter brancos entre os operandos e operadores.
Pode limitar o seu tamanho em 40 caracteres para caber numa linha digitada.
Os nomes de variáveis começam com uma letra e os demais caracteres são letras (maiúsculas ou minúsculas) ou algarismos com tamanho máximo de 20 caracteres. Exemplos:

```
A1
MinhaVariavel
TotalGeral
Unitario2
```

O símbolo **^** significa exponenciação.
O cálculo é feito com valores não inteiros, ou seja, como se as variáveis fossem do tipo **double**.
Se for usada uma variável dentro da expressão ainda indefinida, o programa deve solicitar a entrada de seu valor.

Exemplo de dados:

```
CTE = 1.00
IPI =   0.15
ICMS =  0.12
CustoFOB = 100.00
Fator = 2.4849
Local = Fator * CustoFOB
Final = Local/(CTE - IPI - ICMS)
ValorIPI = Final * IPI
ValorICMS = Final * ICMS
ValorPC = Final * PC
```

Para esta entrada de dados o programa deveria imprimir:

```
Entre com o valor de PC:0.0925

Valores Finais das Variaveis:

CTE = 1.00
IPI = 0.15
```

```
ICMS = 0.12
CustoFOB = 100.00
Fator = 2.4849
Local = 248.49
Final = 340.40
ValorIPI = 51.06
ValorICMS = 40.85
ValorPC = 31.49
PC = 0.0925
```

Comandos do tipo **<var>=<número>**
Pode usar a função **atof** (string para double) para pegar o número. Para decidir se é um número ou não basta comparar o primeiro caractere após a atribuição (**=**).

Operador **^** (exponenciação)
Como em C não há esta operação usar a função **pow** para o cálculo.

Erros de sintaxe na expressão
A expressão a ser traduzida pode conter erros, mas não vamos considerar isso. Basta verificar se os caracteres são válidos ou não. Opcionalmente, o formato pode ser livre, isto é, podem existir brancos entre os elementos da expressão.

Comandos do tipo **<var>=<expressão aritmética>**
Para se calcular o valor da expressão é necessário antes traduzi-la para a notação pós-fixa. O algoritmo que faz isso usa uma pilha de operadores e a prioridade entre eles. Para calcular a expressão a partir da notação pós-fixa, deve usar uma pilha de operandos.
Considere a prioridade usual:

- **^**, **/** e *****, **+** e **–**, **=**
- Realizar primeiro as operações entre parêntesis
- Operações de mesma prioridade, da esquerda para a direita.

Temos então que ter uma tabela com os nomes das variáveis **char TabNomeVar[30][20]** (máximo de 30 variáveis com tamanho máximo de 20 caracteres), e uma tabela com os valores das variáveis **double TabValorVar[30]**. As duas tabelas são paralelas.

A estrutura do programa fica:

```
Iniciar tabelas;
while (1) {
  Leia uma linha L;
  Se L = <var>=<número>
     Guardar <var> em TabNomeVar e <número> em TabValorVar;
  Se L = <var>=<expressão aritmética>
     Guardar <var> em TabNomeVar;
     Traduzir <var>=<expressão aritmética> para pós-fixa;
     Calcular o valor de <expressão aritmética>;
        Se houver alguma variável não definida na
        <expressão aritmética> pedir a entrada desta variável;
```

```
      Guardar valor calculado em TabValorVar;
  Se L = "" imprimir as variáveis e seus valores;
}
```

Faça pelo menos 2 funções:

1) Traduz o comando normal em uma expressão pós-fixa

`int traduz_pos(char exp[], char exp_pos[])`

Recebe a expressão em **`exp[]`** e devolve a expressão traduzida em **`exp_pos[]`**. Devolver **`0`** se a tradução foi bem e **`-1`** se algo errado foi encontrado durante a tradução (por exemplo, algum caractere inválido). Note que os nomes das variáveis podem ser trocados pelo seu índice em **`TabNomeVar[]`** ou **`TabValorVar[]`**. Assim, cada elemento da expressão em **`exp_pos[]`** tem apenas um caracter.

No exemplo acima temos as seguintes variáveis:

Índice	Variável
0	**CTE**
1	**IPI**
2	**ICMS**
3	**CustoFOB**
4	**Fator**
5	**Local**
6	**Final**
7	**ValorIPI**
8	**ValorICMS**
9	**ValorPC**
10	**PC**

As expressões:

```
Local = Fator * CustoFOB
Final = Local/(CTE – IPI – ICMS)
```

Ficam em notação pós-fixa:

```
5 4 3 * =
6 5 0 1 – 2 - / =
```

2) Calcula o valor da expressão:

`double calcula_valor(char exp_pos[], double TabValorVar[])`

Recebe a expressão em notação pós-fixa em **`exp_pos[]`** e a tabela com os valores das variáveis e devolve valor da expressão calculada. Se durante o cálculo, for usada alguma variável com o valor ainda indefinido, a função deve solicitar a entrada o valor desta variável. Quando for uma operação de divisão, verificar se o divisor é zero. Neste caso, imprimir mensagem de erro e retornar.