MAC122 – Princípios de Desenvolvimento de Algoritmos
Segundo Semestre de 2019 – BMAC – IMEUSP – Prof. Marcilio
EXERCÍCIO PROGRAMA II – Entregar até 13/Outubro/2019

Interpretador para o cálculo de expressões aritméticas

O programa deve ler expressões aritméticas e calcular o seu valor.
Deve funcionar como o interpretador Python no modo prompt.

```
>>> bib = 5
>>> zk=0 - 2
>>> cido = bib * (zk - 3)
>>> soma = bib+zk+cido
>>> cido
-25
>>> soma
-22
>>> cido + soma
-47
>>> k=7*(soma+zk/3*(cido + 4))
>>> k
-56.0
>>>
```

As linhas digitadas podem ser:

a) Um comando de atribuição (**<variável> = <expressão aritmética>**). Neste caso o valor da expressão deve ser calculado e atribuído à variável.

b) Uma expressão aritmética (**<expressão aritmética>**). Neste caso o valor da expressão aritmética deve ser calculado e mostrado no vídeo.

Expressões aritméticas

Vamos simplificar as expressões aritméticas. Podem conter:

a) Constantes inteiras (ex: 132, -45, 123456789, etc.)
b) Variáveis contendo apenas letras (ex: k, SoMa, contador, QuantaDor, etc.)
c) Operadores binários e parêntesis: +, -, *, /, **
d) Operadores unários: + e –
e) Parêntesis: ()
f) Atribuição: =

Outro exemplo:

```
>>> A = 3
>>> B = -4
>>> C = 1
>>> D = B ** 2 - 4 * A * C
>>> xum = (-B + D ** (1/2)) / (2*A)
>>> xds = (-B - D ** (1/2)) / (2*A)
```

```
>>> xum
1.0
>>> xds
0.3333333333333333
>>>
```

O programa deve então repetir indefinidamente:

- Colocar o prompt >>>
- Ler a expressão (string)
- Traduzir a expressão para a notação pós-fixa
- Calcular o valor da expressão usando a notação pós-fixa
- Se o último operador for a atribuição, colocar o valor calculado na tabela de variáveis. Se não, mostrar o valor calculado no vídeo

A tradução para a notação pós-fixa

Use o algoritmo de tradução para pós-fixa que utiliza uma pilha de operadores e os movimenta baseado em sua prioridade.

A linha digitada (string) pode conter ou não um ou mais espaços entre os elementos:

```
ak = bib+c*(dado+1005)
ak=  bib  +c* (dado  +1005 )
   ak  =bib + c * (dado + 1005)
```

Para facilitar a manipulação da linha para a transformação em pós-fixa, uma sugestão é varrer a string criando uma outra string contendo os elementos (tokens) separados por um branco. No exemplo acima ficaria:

```
ak = bib + c * ( dado + 1005 )
```

Assim, com um comando **`split()`** nessa string, transforma-se a mesma numa lista, cuja manipulação é mais fácil. No exemplo acima a lista ficaria:

```
['ak', '=', 'bib', '+', 'c', '*', '(', 'dado', '+', '1005', ')']
```

Outra sugestão equivalente é simplesmente varrer a expressão ignorando os espaços, separando os elementos (tokens) e colocando cada um deles num elemento da lista.

A expressão depois de traduzida pode ficar numa outra lista que será varrida posteriormente para o cálculo. No exemplo acima:

```
['ak', 'bib', 'c', 'dado', '1005', '+', '*', '+', '=']
```

Os operadores

Para a tradução, considere a prioridade usual dos operadores:

- **`-`** e **`+`** (unários), depois **`**`**, depois **`/`** e **`*`**, depois **`+`** e **`-`**. **`P`**or último a atribuição **`=`**.
- Parêntesis alteram a prioridade.
- Operadores de mesma prioridade, da esquerda para a direita.

O operador exponenciação (**) possui 2 caracteres que devem obrigatoriamente estar contíguos.
A atribuição (**=**) também é um operador. O de menor prioridade

Os operadores de soma e subtração (+ e -) podem ser unários ou binários. A decisão depende do elemento (token) anterior a ele na expressão. Se for uma variável ou um número, é binário senão é unário. Verifique se há outros casos.
Ao traduzir a expressão para a notação pós-fixa, substitua o caractere **-** pelo caractere **_** (underline) e o caractere **+** pelo caractere **#** quando forem unários.

Erros de sintaxe na expressão

A expressão a ser traduzida pode conter erros, mas não vamos considerar isso. Ou seja, não é necessário detectar erros de sintaxe. Basta verificar se os caracteres que aparecem na expressão são válidos ou não. O formato deve ser livre, isto é, podem existir brancos entre os elementos da expressão.

A tabela de variáveis

À medida que as variáveis vão sendo coletadas da expressão aritmética, são colocadas numa tabela (lista) que será consultada quando a expressão tiver sendo calculada (TabVar). Paralela a esta tabela há uma outra (TabVal) com os valores atuais da cada variável, inicialmente None. No exemplo acima:

```
TabVar: ['ak', 'bib', 'c', 'dado']
TabVal: [None, None, None, None]
```

O cálculo do valor da expressão

Use o algoritmo de cálculo do valor de uma expressão já em notação pós-fixa, que varre a lista com a expressão e usa uma pilha de operandos. Ao final do cálculo (o resultado está na base da pilha), o resultado é colocado em TabVal se foi uma atribuição ou mostrado no vídeo se não for.
O cálculo e valores devem ser feito usando o tipo float pois embora as constantes numéricas na expressão sejam sempre inteiras, durante o cálculo podem aparecer valores float por uma divisão por exemplo.

Organização do programa

Como sempre, procure estruturar o seu programa de forma modular identificando partes comuns que podem ser reutilizáveis. Por exemplo, use um ADT Pilha para implementar tanto a pilha de operadores como a pilha de operandos.

Faça pelo menos as funções:

a) TraduzPosFixa(exp) – recebe string exp contendo uma expressão aritmética com ou sem atribuição e devolve uma lista contendo essa expressão em notação pós-fixa. Devolve True se o cálculo foi feito com sucesso, ou False caso contrário.

b) CalcPosFixa(listaexp) – recebe uma lista contendo uma expressão em notação pós-fixa e calcula o seu valor. Devolve esse valor se o cálculo foi feito com sucesso ou None caso contrário.