Aula 06: 21/AGO/2018

Aula passada

Sequências de parênteses, colchetes e chaves bem-formadas e pilhas.

Hoje

Notação polonesa e mais pilhas pilhas

Sobre notação polonesa podemos primeiro calcular o valor de uma expressão aritmética em notação polonesa reversas (= **notação posfixa**) e em seguida fazer o programa que transforma de infixa para posfixa.

Outro problema bacana é o de *transformar a representação de um valor da notação decimal para a binária*. O uso da pilha é meio sem graça, já que só empilhamos todods os dígitos e depois desempilhamos todos esses dígitos. O problema é bacana para depois falarmos de recursão. Podemo fazer uma conversão de uma representação para outra através de uma recursão de calda (=*tail recursion*).

Precedência

Qual é o valor de

```
2 + 3 * 4 = 14
2 * 3 + 4 = 10
2 - 3 + 4 = 3
2 + 3 - 4 = 1
```

Cada operador tem a sua precedência. A única coisa que muda a precedência é a presença de parênteses

```
(2 + 3) * 4 = 20
2 * (3 + 4) = 14
2 - (3 + 4) = -5
2 + (3 - 4) = 1
```

A expressão `A + B * C + D` pode ser reescrita como `((A + (B * C)) + D)`.

Notação infixa

Usualmente os operadores são escritos *entre* os operandos

```
(A + B) * D + E / (F + A * D) + C
   1    1   2        3   2    4
(1 + 2) * 3 + 4 / (5 + 6 * 7) + 8 = 17.085106382978722
```

Essa é a chamada **notação infixa**.

Notação polonesa

Na **notação polonesa** (reversa) ou **posfixa** os operadores são escritos depois dos operandos

```
  A B + D * E F A D * + / + C +
      1   1         2 3   2   4
  1 2 + 3 * 4 5 6 7 * + / + 8 +
```

infixa	posfixa
2+3*4	2 3 4 * +
(2+3)*4	2 3 + 4 *
2*(3+4)/5-6	2 3 4 + * 5 / 6 -
2-1	2 1 -
10 + 3 * 5 / (16 - 4)	10 3 5 * 16 4 - / +

Uma expressão é composta de **itens léxicos** (=*tokens*).

Itens lexicais são palavras simples ou grupos de palavras no léxico de uma língua

Problema: valor de expressão posfixa

Versão vaga: Escreva um programa (= função `main()`) que lê uma expressão numérica pósfixa que contém apenas os operadores binários `+`, `-`, `*` e `/` e os operandos são valores da classe `int` ou `float` e calcula o valor da expressão.

Exemplos:

```
Calculadora de expressões numéricas posfixas
expr >>>  7 8 + 3 2 + /
3
expr >>> 4 5 6 * +
34
expr >>>  7 8 + 3 2 + /
3
expr >>> 4 5 6 * +
34
expr >>> 1 2 + 3 * 4 5 6 7 * + / + 8 +
17.0851
expr >>> 10 3 5 * 16 4 - / +
11.25
expr >>> 1 2 3 + -
-4
expr >>> 1 2 3 - +
0
expr >>> 4 7 9 - +
2
expr >>> 1 2 3 + 4 5 * 2 / +
15
expr >>> 22 33 44 * *
31944
```

Depois é melhor supor que há um espaço separando os itens léxicos.

Solução

Arquivo `calculadora.py`

```
from pilha import Pilha

# constantes
PROMPT = "expr >>> "
QUIT   = ""
ABRE   = "("
FECHA  = ")"
ADD    = "+"
SUB    = "-"
MUL    = "*"
DIV    = "/"
OPERADORES = "+-*/"
```

```
def main():
    '''
    Calcula o valor de uma expressão númerica
    posfixa que contém apenas os operadores
    binários +, -, * e /.

    Pré-condição: o programa supõe que os operadores
      e operandos estão separados por pelo menos um
      espaço.
    '''
    print("Calculadora de expressões numéricas posfixas")
    print("(Tecle ENTER para encerrar o programa.)"

    expressao = input(PROMPT)
    while expressao != QUIT:
        # posfixa = infixa_para_posfixa(expressao)
        posfixa = expressao.split() # supõe espaço entre itens lexicos
        valor   = valor_expressao(posfixa)
        print("%g" %valor)
        expressao = input(PROMPT)
```

Escreva uma função `valor_expressao()` que respeita a seguinte especificação:

```
def valor_expressao(posfixa):
    ''' (list) -> float

    Recebe um lista de tokens representando um expressão
    numérica em notação posfixa e retorna o valor da expressao.

    Pre-condiçao: a funçao supõe que a expressao está correta.
    '''
    pilha = Pilha()
    for item in posfixa:
        if item in OPERADORES:
            valor_2 = pilha.desempilha()
            valor_1 = pilha.desempilha()
            if item == ADD:
                valor = valor_1 + valor_2
            elif item == SUB:
                valor = valor_1 - valor_2
            elif item == MUL:
                valor = valor_1 * valor_2
            elif item == DIV:
                valor = valor_1 / valor_2
        else:
            # é um numero
            valor = float(item)

        pilha.empilhe(valor)

    resultado = pilha.desempilha()
    return resultado
```

Problema: infixa para posfixa

Escreva uma função `infixa_para_posfixa()` que respeita a seguinte especificação:

```
def infixa_para_posfixa(infixa):
    '''(str) -> list

    Recebe um string infixa com uma expressão numérica
    em notação infixa e cria e retorna uma lista com
    com os tokens da expressão representando a correspondente
    expressão em notação posfixa.

    Pré-condição: a função supões que os itens da expressão
    estão separado por pelo menos um espaço.
    '''
    posfixa = []
    tokens = infixa.split()

    pilha = Pilha()
    for item in tokens:
        #-----------------------------------
        if item == ABRE:
            pilha.empilhe(ABRE)

        #-----------------------------------
        elif item == FECHA:
            item_topo = pilha.desempilha()
            while item_topo != ABRE:
                posfixa.append(item_topo)
                item_topo = pilha.desempilha()

        #-----------------------------------
        elif item in [ADD,SUB]:
            while not pilha.vazia() and \
                pilha.topo() in OPERADORES:
                item_topo = pilha.desempilha()
                posfixa.append(item_topo)
            pilha.empilhe(item)

        #-----------------------------------
        elif item in [MUL,DIV]:
            while not pilha.vazia() and \
                  pilha.topo() in [MUL,DIV]:
                item_topo = pilha.desempilha()
                posfixa.append(item_topo)
            pilha.empilhe(item)

        #-----------------------------------
        else: # encontramos um número
           posfixa.append(item)
```

```
    while not pilha.vazia():
        item_topo = pilha.desempilhe()
        posfixa.append(item_topo)

    return posfixa
```

Resumo

Com esta aula teremos visto dois ou três problemas envolvendo pilhas:

- sequências bem-formada
- calculadora de expressões posfixas
- conversão de expressões infixas para posfixas

Junto com pilha vem o conceito de objetos.