**MAC122 – Princípios de Desenvolvimento de Algoritmos**

**Segundo Semestre de 2021 – BMAC – IMEUSP – Prof. Marcilio**

**Exercício Programa 1 – Entregar até 03/Outubro/2021**

**Interpretador de Expressões Numéricas com Inteiros e Frações Ordinárias**

O programa deve ler expressões numéricas e calcular o seu valor.

Deve funcionar como o interpretador Python no modo prompt.

Como vamos usar apenas valores inteiros, o valor da expressão deve ser dado como uma Fração Ordinária, como no exemplo abaixo.

**>>> 2 + 3 \* 5**

**17**

**>>> (2 + 3) / (5 + 4 \* 2)**

**5 / 13**

**>>> (2 + 1) \*\* 2 / 5 + 2**

**19 / 5**

**>>> (2 + 1) \*\* 2 / 5 - 2**

**-1 / 5**

**>>>**

As linhas digitadas contém números (**int**) e os caracteres **(, ), +, -, \*, /, \*\***.

O expoente deve ser sempre um inteiro.

**Não** considere os operadores **+** e **-** unários.

Pode supor que **sempre** a expressão está sem erros de sintaxe.

O programa deve então repetir até que seja digitado **“fim”**:

* 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
* Mostrar o resultado no vídeo
* Voltar ao início

**A tradução para a notação pós-fixa**

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

Para facilitar a manipulação da linha antes de transformá-la em pós fixa, use expressões regulares (abaixo) para separar os elementos da expressão. Exemplo:

**import re**

**# ...**

**t = "(12 + 531 \* (42 \*\* 5 - 21) + 1)"**

**r = re.findall(r"(\b\w\*[\.]?\w+\b|[\(\)\+\\*\-\/])", t)**

**print(r)**

Será impresso:

**['(', '12', '+', '531', '\*', '(', '42', '\*', '\*', '5', '-', '21', ')', '+', '1', ')']**

Cuidado especial deve ser tomado com o operador \*\*. Nesse caso, tem que verificar o próximo elemento da lista já “splitada” e verificar se é \* ou \*\*.

**Os operadores**

Para a tradução, considere a prioridade usual dos operadores:

* **\*\*,** depois **/** e **\*,** depois **+** e **–**.
* Parêntesis alteram a prioridade.
* Operadores de mesma prioridade, da esquerda para a direita.

**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).

**Organização do programa**

Como sempre, procure estruturar o seu programa de forma modular identificando partes comuns que podem ser reutilizáveis. Use um ADT **Pilha** (classe **Pilha)** para implementar tanto a pilha de operadores (pilha de strings), como a pilha de operandos (pilha de Frações).

Além da classe **Pilha**, faça pelo menos as funções:

1. **TraduzPosFixa(exp)** – recebe string **exp** contendo uma expressão aritmética e devolve uma lista contendo essa expressão em notação pós-fixa. Exemplo:

**TraduzPosFixa("(2 +52)\* (3 - 51/31) ")**

devolve: **["2", "52", "+", "3", "51", "31", "/", "-", "\*"]**

1. **CalcPosFixa(listaexp)** – recebe uma lista contendo uma expressão em notação pós-fixa, calcula e devolve o seu valor. Se houver algum problema no cálculo devolve **None**. Exemplo:

**CalcPosFixa(["2", "52", "+", "3", "51", "31", "/", "-", "\*"])**

devolve: **2268 / 31**

Obs: Não esqueça de identificar o seu programa. Exemplo: No início do programa:

**# Exercício Programa I – MAC 122 – PDA**

**# João Antonio dos Santos**

**# NUSP: 14253646**