INF101 – Introdução à Programação II

Roteiro de Prática: 18 de março de 2021

Introdução

O objetivo desta prática é exercitar o uso da estrutura de dados *pilha* que, como visto em aula teórica, é uma lista que obedece a regra LIFO para todas as inserções e remoções de elementos. Para ilustrar o uso, vamos considerar o problema clássico da ciência da computação denominado *balanceamento de parênteses* em uma expressão aritmética, por exemplo. Considere que a expressão seja armazenada em um *string*. A solução do problema baseia-se no seguinte:

1. Crie uma pilha vazia (lista vazia).
2. Varra o *string* de entrada caractere por caractere. Ao encontrar um abre-parêntese (parêntese esquerdo), empilhe-o.
3. Se o caractere inspecionado no momento for um fecha-parêntese (parêntese direito), desempilhe o correspondente abre-parêntese do topo da pilha. Se a pilha estiver vazia antes de desempilhar, termine a varredura relatando (retornando) desbalanceamento.
4. Para qualquer outro caractere que não seja abre ou fecha-parêntese (parêntese direito), ignore-o. Aqui poderemos fazer isso, porque não queremos analisar a expressão de entrada mais profundamente, como por exemplo, reconhecer operandos e operadores.
5. Ao terminar a varredura da expressão, a pilha deve estar vazia. Neste caso, relate sucesso para o balanceamento de parênteses. Caso contrário (pilha não vazia no final), relate desbalanceamento.

Para cumprir o objetivo, siga as instruções abaixo.

Instruções

1. Usando o IDLE, crie o arquivo-fonte vazio `p07.py`. De pronto, digite os comentários obrigatórios no cabeçalho do arquivo contendo: o nome do programador, o nº de matrícula, a data de criação do arquivo e um breve texto dizendo o que o programa faz.
2. Salve o arquivo-fonte de tempos em tempos para o caso em que ocorra falha de energia elétrica ou qualquer outro incidente com seu computador e, assim, você não perca todo o trabalho.
3. Estruture seu programa em duas funções: `main()` e `analise_parenteses(expressao)`.
4. A função `main` deve ler as expressões de entrada a serem analisadas. Para cada expressão lida, chame a função `analise_parenteses(expr)` em que `expr` é a última expressão lida. Para terminar a entrada de expressões, o usuário deve entrar com um *string* vazio.
5. A função `analise_parenteses(expressao)` segue o algoritmo esboçado acima na Introdução. Considere que a função vai ser booleana: retorna verdadeiro (`True`), se o balanceamento de parênteses de `expressao` estiver ok. Caso contrário, ela retorna falso (`False`).
6. Não se esqueça de chamar a função `main()` no final do arquivo-fonte para desencadear todo o processo.
7. Teste seu programa com várias expressões de entrada. Veja a o Exemplo de Teste do Programa abaixo. O que está enfatizado em fundo amarelo são as entradas digitadas pelo usuário.
8. Se seu programa entrar em *laço infinito* ou travar por alguma razão, digite `CTRL-C` na janela do *Python Shell* para interromper a execução do programa.

☞ Não se esqueça de preencher o cabeçalho do código fonte com seus dados, a data de hoje e uma breve descrição do programa.

Após certificar-se de que seu programa esteja correto, envie o arquivo com o código fonte (`p07.py`) através do sistema de entrega do LBI.

Exemplo de Teste do Programa

```
Digite uma expressão com parênteses (ENTER para terminar): (a + b)
(a + b) está OK
Digite uma expressão com parênteses (ENTER para terminar): (A + (B - C))
(A + (B - C)) está OK
Digite uma expressão com parênteses (ENTER para terminar): (x + y)*(x - 2)(
(x + y)*(x - 2)( está ERRADO
Digite uma expressão com parênteses (ENTER para terminar): (x + y)*(x - 2)/((w -
z)**(A + B))
(x + y)*(x - 2)/((w - z)**(A + B)) está OK
Digite uma expressão com parênteses (ENTER para terminar): (area - 4.5)
(area - 4.5) está OK
Digite uma expressão com parênteses (ENTER para terminar): (area - 4.5))
(area - 4.5)) está ERRADO
Digite uma expressão com parênteses (ENTER para terminar): ((area + 10)
((area + 10) está ERRADO
Digite uma expressão com parênteses (ENTER para terminar):
```