UNIVERSIDADE FEDERAL DE SANTA CATARINA
INE5609 - Estruturas de Dados
Jonathan Moraes Pereira (23205205)

TRABALHINHO 1 - LISTA DUPLAMENTE ENCADEADA

1. **acessarAtual(self)**: Retorna o valor do nó atual, onde se cursor não for None, retorna o dado armazenado no nó atual;
2. **inserirAntesDoAtual(self, elemento)**: Insere um novo nó antes do nó atual, criando um novo nó e ajustando os ponteiros do nó anterior e do nó atual para incluir o novo nó, se o nó anterior for None, define o novo nó como o primeiro da lista;
3. **inserirAposAtual(self, elemento)**: Insere um novo nó após o nó atual, ajustando os ponteiros do nó atual e do próximo nó para incluir o novo nó, se o proximo nó for None, define o novo no como o último da lista;
4. **inserirComoUltimo(self, elemento)**: Insere um novo nó no final da lista, se a lista não estiver vazia, ajusta o ponteiro do último nó para apontar para o novo nó; se a lista estiver vazia, o novo nó se torna o primeiro;
5. **inserirComoPrimeiro(self, elemento)**: Insere um novo nó no início da lista, similar a inserirComoUltimo, mas o novo nó se torna o primeiro da lista;
6. **inserirNaPosicao(self, k, novo)**: Insere um novo nó na posição k, nesse caso se k for 1, chama inserirComoPrimeiro, se k for o tamanho +1, chama inserirComoUltimo. Para outras posições, percorre a lista até a posição desejada, ajustando os ponteiros para inserir o novo nó;
7. **excluirAtual(self)**: Remove o nó atual da lista, nessa função se a lista tiver apenas um nó, chama reset() para resetar a lista. Se o nó atual for o primeiro ou o último, chama as funções correspondentes. Nos nós intermediários, ajusta os ponteiros dos nós anterior e próximo para remover o nó atual;
8. **excluirPrim(self)**: Remove o primeiro nó da lista, ajustando o ponteiro do primeiro nó para o próximo nó, se o nó atual for o primeiro, avança o cursor;
9. **excluirUlt(self)**: Remove o último nó da lista, ajusta o ponteiro do último nó para o anterior, se o atual for o último, retrocede o cursor;
10. **excluirElem(self, chave)**: Remove o primeiro nó que contém a chave especificada, vai percorrer a lista para encontrar o nó correspondente e chama excluirAtual() para removê-lo;

11. **excluirDaPos(self, posicao)**: Remove o nó na posição especificada, se a posição passada for válida, avança o cursor para essa posição e chama excluirAtual().

12. **buscar(self, chave)**: Busca um nó com o valor correspondente à chave, faz isso percorrendo a lista e retorna o valor se encontrado;

13. **avancarKPosicoes(self, k)**: Avança o cursor k posições para frente na lista, percorrendo k nós a partir do cursor atual e parando se chegar ao final da lista;

14. **retrocederKPosicoes(self, k)**: Retrocede o cursor k posições na lista, semelhante a avancarKPosicoes mas move o cursor para trás;

15. **irParaPrimeiro(self) e irParaUltimo(self)**: Move o cursor para o primeiro ou último nó da lista, respectivamente, Definindo o cursor como primeiro ou ultimo;

16. **vazia(self)**: Retorna True se o tamanho for 0, caso contrário False;

17. **posicao(self, chave)**: Percorre a lista até encontrar o nó correspondente, retornando sua posição.

18. **reset(self)**: Define todos os ponteiros como None e o tamanho como 0.

19. **listar(self)**: Percorre a lista e imprime os valores de cada nó, usando acessarAtual() e verificando os nós anterior e próximo.