Relatório de Implementação

Lista Duplamente Encadeada com Cursor

Estudante:

João Paulo Decker Oleinik (24202528)

Disciplina:

INE5609 - Estruturas de Dados

Professor:

José Eduardo De Lucca

Sobre o Relatório:

Este relatório técnico descreve a implementação da classe ListaDuplamenteEncadeada e seus métodos, conforme as especificações do trabalho. O foco é na descrição da lógica de cada operação, destacando a manipulação dos ponteiros.

Descrição dos Métodos Implementados

Propriedades de Acesso

- **`primeiro(self)`**: Retorna a referência para o primeiro nó da lista.
- **`ultimo(self)`**: Retorna a referência para o último nó da lista.
- **`tamanho(self)`**: Retorna o número de elementos na lista.
- **`cursor(self)`**: Retorna a referência para o nó onde o cursor está posicionado.

Operações de Acesso e Auxiliares

- **`vazia(self)`**: Retorna `True` se a lista não contiver nós.
- **`cheia(self)`**: Retorna `False`, pois a lista é uma estrutura de dados dinâmica em Python.
- **`posicaoDe(self, chave)`**: Percorre a lista do início ao fim e retorna a posição sequencial do primeiro nó que contém o valor da `chave`. Retorna `-1` se o nó não for encontrado.
- **`acessarAtual(self)`**: Retorna o valor do nó onde o cursor está posicionado. Se o cursor for `None`, retorna `None`.

Operações de Controle do Cursor

- **`irParaPrimeiro(self)`**: Posiciona o cursor no primeiro nó da lista.

- **`irParaUltimo(self)`**: Posiciona o cursor no último nó da lista.
- **`avancarKPosicoes(self, k)`**: Move o cursor `k` posições para frente, parando no final da lista se necessário.
- **`retrocederKPosicoes(self, k)`**: Move o cursor `k` posições para trás, parando no início da lista se necessário.

Operações de Inserção

- **`inserirComoPrimeiro(self, elemento)`**: Insere um novo nó no início da lista. O ponteiro `ant` do nó original é ajustado para apontar para o novo nó, que então se torna o novo primeiro da lista.
- **`inserirComoUltimo(self, elemento)`**: Adiciona um novo nó no final da lista. Se a lista não estiver vazia, ajusta o ponteiro `prox` do último nó para apontar para o novo nó e define o novo nó como o último. Se a lista estiver vazia, o novo nó se torna o primeiro e o último.
- **`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 incluí-lo. Se o cursor estiver no primeiro nó, o novo nó se torna o novo primeiro da lista.
- **`InserirAposAtual(self, elemento)`**: Insere um novo nó após o nó atual. O método ajusta os ponteiros do nó atual e do próximo nó para incluir o novo elemento. Se o cursor estiver no último nó, o novo nó se torna o último.
- **`inserirNaPosicao(self, k, novo)`**: Insere um novo nó na posição `k` da lista. A lógica move o cursor para a posição `k-1` e chama `InserirAntesDoAtual` para inserir o novo elemento. A função valida a posição para evitar inserções em locais inválidos.

Operações de Exclusão

- **`ExcluirPrim(self)`**: Remove o primeiro nó da lista. O ponteiro `primeiro` é reatribuído para o próximo nó.
- **`ExcluirUlt(self)`**: Remove o último nó da lista. O ponteiro `ultimo` é reatribuído para o nó anterior.
- **`ExcluirAtual(self)`**: Remove o nó onde o cursor está posicionado. A lógica lida com três casos: se a lista tem apenas um nó, se o cursor está no primeiro ou no último nó, ou se está no meio da lista. Nos casos de borda, chama as funções correspondentes de exclusão (`ExcluirPrim`, `ExcluirUlt`). Para o caso intermediário, ajusta os ponteiros dos nós anterior e próximo para ignorar o nó atual.

Operações de Busca

- **`Buscar(self, chave)`**: Percorre a lista para encontrar um nó com o valor correspondente à chave. Ao encontrar, posiciona o cursor no nó encontrado e retorna `True`. Se não encontrar, o cursor para no final da lista e a função retorna `False`.