Programação: Listas Duplamente Encadeadas

Prof. André Grégio

Listas Duplamente Encadeadas

O tópico desta aula é uma estrutura de dados chamada "lista duplamente encadeada", cujos nós possuem um ponteiro para o nó anterior além do ponteiro para o próximo nó...

O nó com possibilidade de encadeamento duplo

A lista duplamente encadeada é formada por nós similares ao da lista encadeada simples, porém com um atributo adicional: um apontador para o nó anterior. Dessa forma, é possível percorrer esse tipo de lista em ambas as direções. A Figura 1 redefine a estrutura "nó" vista anteriormente para adicionar o novo apontador e simplificar o armazenamento de dados, uma vez que a chave não é realmente necessária.

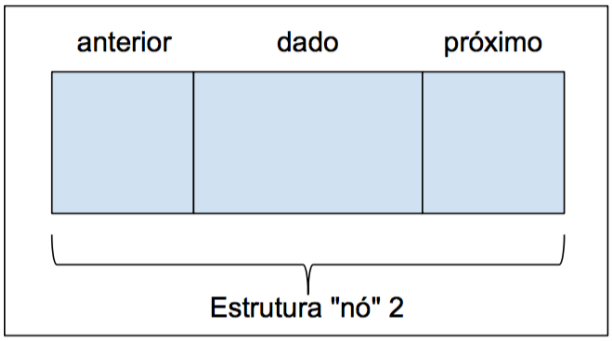

Figura 1: Estrutura "nó" redefinida com dois apontadores.

Deve-se ter cuidado ao atualizar os apontadores dos nós durante as operações com esse tipo de lista. Se o apontador "anterior" de um dado nó tem valor NULO, então este é o nó **cabeça**. Se o apontador "próximo" de um dado nó tem valor NULO, este é o nó **cauda** (que contém o último elemento). Se o nó cabeça da lista é NULO, a lista está vazia. A Figura 2 ilustra uma lista duplamente encadeada que faz uso da nova estrutura "nó".

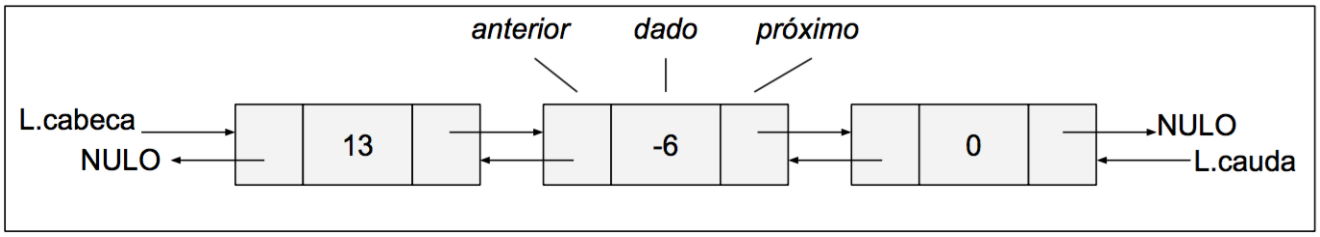

Figura 2: Exemplo de lista duplamente encadeada.

Algumas operações

A seguir, mostra-se a ideia por trás das operações INSERIR, REMOVER e IMPRIMIR sobre uma lista duplamente encadeada, seguindo os mesmos princípios estabelecidos anteriormente, isto é, dada uma lista L e um nó x:

- INSERIR(L, x)
  1. x->proximo = L->cabeca;
  2. SE L->cabeca != NULO ENTAO
  3. L->cabeca->anterior = x;
  4. L->cabeca = x;
  5. x->anterior = NULO;
- REMOVER(L, x)
  1. SE x->anterior != NULO ENTAO
  2. x->anterior->proximo = x->proximo;
  3. SENAO L->cabeca = x->proximo;
  4. SE x->proximo != NULO ENTAO
  5. x->proximo->anterior = x->anterior;
- IMPRIMIR(L)
  1. ESTRUTURA noh *aux= L->cabeca;
  2. ENQUANTO aux != NULO FAÇA
  3. IMPRIME aux->dado;
  4. aux = aux->proximo;

EXERCÍCIO 1: Defina a função INICIALIZAR para a lista duplamente encadeada.

EXERCÍCIO 2: Defina uma função na qual, dado um valor "v", vasculhe a lista em busca da existência de um nó "x" que o contém e retorne o nó em questão. Tal função deve funcionar para os dois tipos de listas com apontadores (simples e duplamente encadeada).

BÔNUS: Lista circular com apontador

Uma lista circular pode ser enxergada como um vetor no qual há estruturas de controle para todas as suas posições sejam bem aproveitadas. Por exemplo, supondo a primeira posição vazia e uma inserção após a última posição, a lista sabe que o dado a ser inserido deve ocupar a primeira posição e que, independente do índice do vetor, este dado agora ocupa a última posição da lista. Assim, uma lista circular pode ser representada na forma de um anel, conforme ilustrado na Figura 3. Deve-se levar em conta as operações que verificam se uma lista está vazia ou cheia, bem como maneiras efetivas de se manter a integridade da lista circular após inserções e remoções variadas.

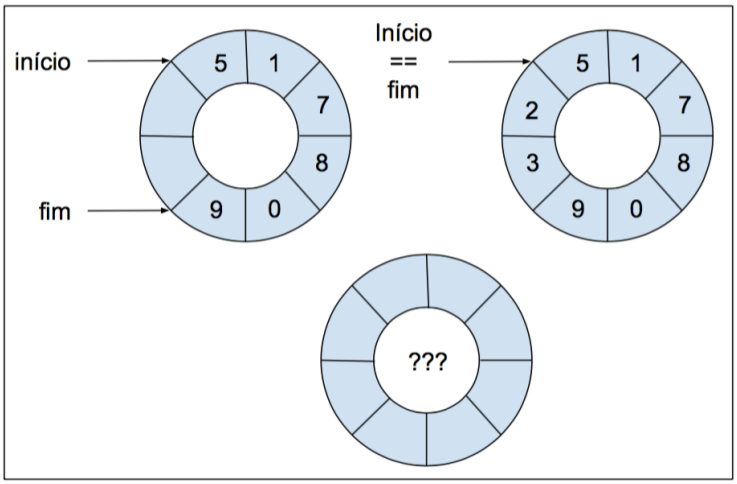

Figura 3: Exemplo de lista circular parcialmente ocupada, cheia e vazia.

Em uma lista circular com apontador, os nós inicial (cabeça) e final (cauda) são os mesmos. Para alcançar tal feito, o apontador "anterior" do primeiro nó aponta para a cauda, enquanto que o apontador "próximo" do último nó aponta para a cabeça. Um objeto que é comumente usado para auxiliar no controle dos limites (cabeça e cauda) de uma lista é a chamada *sentinela*. A sentinela é um nó com a mesma estrutura de um nó da lista, porém sem valor associado aos seus atributos. Para usar uma sentinela na simplificação de uma lista circular duplamente encadeada, seu apontador "anterior" aponta para o "próximo" do último nó de uma lista (e é apontado por este), enquanto que seu apontador "próximo" aponta para o "anterior" do primeiro elemento da lista (que também o aponta). A Figura 4 ilustra esse tipo de lista.

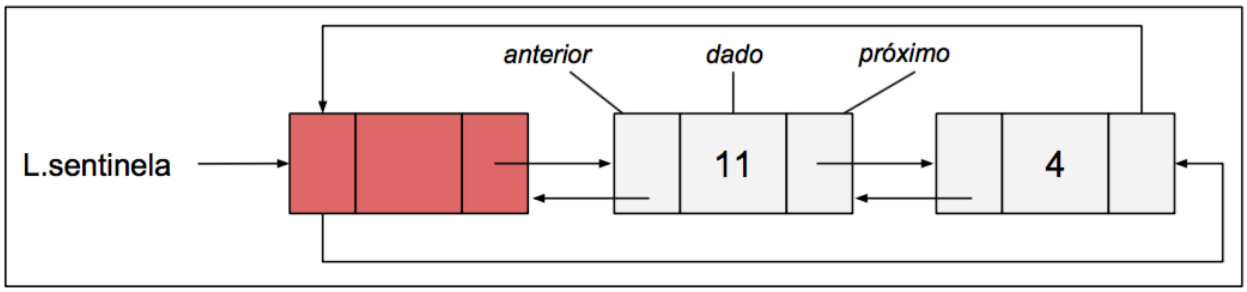

Figura 4: Exemplo de lista circular usando apontadores e uma sentinela.

EXERCÍCIO 3: Defina uma lista circular usando vetor e as funções INICIALIZAR, INSERIR, REMOVER, VAZIA e IMPRIMIR, tratando os casos especiais.

EXERCÍCIO 4: Defina as funções INICIALIZAR, INSERIR, REMOVER, VAZIA e IMPRIMIR para uma lista circular duplamente encadeada usando sentinela.

EXERCÍCIO 5: Implemente em C as listas encadeada simples, duplamente encadeada e circular (dos exercícios 4 e 5) com as operações correspondentes.