BCC - Estruturas de Dados
Lab 04 - Listas Duplamente Encadeadas
Versão ANSI C

Prof. Dr. Paulo César Rodacki Gomes
IFC - Instituto Federal Catarinense

22 de março de 2023

1 Objetivo

O objetivo desta atividade prática em laboratório é realizar a implementação de listas duplamente encadeadas. Por motivo de simplicidade, vamos implementar listas para armazenar valores inteiros.

A atividade consiste em implementar em linguagem C as principais operações de manipulação de listas duplamente encadeadas conforme visto em sala de aula. Segue o roteiro para implementação da lista:

1. Inicialmente crie os seguintes arquivos fonte:
   - `principal.c`: função main demonstrando a lista
   - `lista_dupla.h`: declarações de estruturas, tipo e funções da lista
   - `lista_dupla.c`: implementação das funções da lista
2. Estruturas a serem criadas no arquivo `lista_dupla.h`:

```
struct NoListaDupla {
       float info;
       struct NoListaDupla *ant;
       struct NoListaDupla *prox;
};

typedef struct NoListaDupla NoListaDupla;
```

3. Funções a serem criadas no arquivo `lista_dupla.c`:

   (a) `NoListaDupla *dllCria(void)`: instancia uma nova lista vazia, e retorna a referência nula para a cabeça (referência para o primeiro nó) da lista;

   (b) `NoListaDupla *dllInsere(NoListaDupla *head, int v)`: insere um novo nó no início da lista. A função recebe o endereço da cabeça da lista (i.e. o primeiro nó) e o valor a ser inserido e deve retornar o endereço da nova cabeça;

   (c) `void dllImprime(NoListaDupla *head)`: imprime os valores armazenados na lista;

   (d) `int dllVazia(NoListaDupla *head)`: retorna 1 (True) ou 0 (False) se a lista estiver ou nao estiver vazia;

   (e) `NoListaDupla *dllBusca(NoListaDupla *head, int v)`: retorna o endereço do primeiro nó da lista que armazena o valor `v`. Se nenhum nó com esse valor for encontrado, a função deve retornar `null`;

   (f) `int dllComprimento(NoListaDupla *head)`: calcula e retorna o número atual de elementos da lista;

   (g) `NoListaDupla *dllUltimo()`: retorna o endereço do último nó da lista. Se a lista estiver vazia, a função deve retornar `null`.

   (h) `NoListaDupla *dllRetira(NoListaDupla *head, int v)`: remove da lista o primeiro nó que contiver o valor igual a v. Se nenhum nó com esse valor for encontrado, a função não retira nenhum nó da lista.

   - O valor de retorno da função é o endereço do primeiro nó da lista (lembre-se que, se o nó a ser retirado é o primeiro, a função vai retornar o endereço do "novo"primeiro nó, ou null se a lista ficar vazia;
   - não se esqueça de liberar memória ocupada pelo nó que será retirado da lista;

   (i) `void dllLibera(NoListaDupla *head)`: libera a lista, ou seja, apaga todo o conteúdo da lista, liberando a memória ocupada por cada um dos seus nós;

(j) `NoListaDupla *dllInsereFim(NoListaDupla *head, int v)`: insere um novo nó no final da lista. A função recebe o endereço da cabeça da lista (i.e. o primeiro nó) e o valor a ser inserido.

- O valor de retorno é o endereço da nova cabeça (este endereço vai mudar quando estivermos inserindo no final de uma lista vazia);
- Você pode usar a função `ultimo` implementada no exercício anterior para facilitar a inserção no final da lista.

(k) `int sllIgual(NoListaDupla *lista1, NoListaDupla *lista2)`: verifica se as duas listas passadas como parâmetros são iguais (neste caso retorna 1) ou diferentes (retorna 0). Para as duas listas serem iguais, elas devem armazenar valores iguais e na mesma ordem;

(l) `NoListaDupla sllMerge(NoListaDupla *l1, NoListaDupla *l2)`: constrói uma nova lista com a intercalação dos nós da lista atual com uma lista passada como parâmetro. Este método deve retornar a cabeça da lista resultante, conforme ilustrado na figura 1. Observação: para criação da lista resultande, você deve instanciar novos nós.

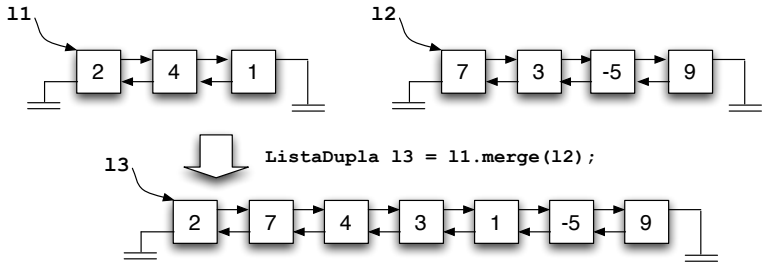

Figura 1: Exemplo de operação merge

Observação: após implementar a lista, implemente um programa principal para testar e demonstrar o funcionamento da lista implementada.