

Universidade Tecnológica Federal do Paraná – Campus Apucarana
Estruturas de Dados (ED62A) – 1º semestre
“Exercícios de Revisão para P1”
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- A resolução da lista não vale nota. Ela serve como preparação para a primeira prova da disciplina;
- Todos os exercícios que aparecem na lista podem aparecer também na prova;
- Na resolução dos exercícios, crie um arquivo para exercício. Por exemplo: quando resolver o exercício 3, crie um arquivo chamado ex03.c, isso facilita a organização;
- Elabore códigos legíveis com variáveis que possuam nomes significativos, isso ajuda no entendimento do código, e na correção da prova.

Exercício 1. Considerando listas de valores inteiros, implemente uma função que receba como parâmetro uma lista encadeada e um valor inteiro **n**, e divida essa lista em duas partes, de tal forma que a segunda lista comece no primeiro nó logo após a primeira ocorrência de **n** na lista original. Essa função deve obedecer o seguinte protótipo:

```
Lista* separa(Lista* lista, int n);
```

A função deve retornar um ponteiro para a segunda sub-divisão da lista original, enquanto **lista** deve continuar apontando para o primeiro elemento da primeira sub-divisão da lista.

Exercício 2. Implemente uma função que receba duas listas ordenadas e gere a união das duas. A função deve retornar a lista resultante da união, obedecendo ao protótipo:

```
Lista* uniao(Lista* lista1, Lista* lista2);
```

Exercício 3. Implemente uma função que receba um vetor de valores inteiros com **n** elementos e construa uma lista encadeada armazenando os elementos do vetor nos nós da lista. Se o vetor tiver zero elementos, a função deve ter como valor de retorno uma lista vazia. O protótipo da função é dado por:

```
Lista* constroi(int n, int* v);
```

Exercício 4. Considere uma lista singularmente encadeada que armazena os seguintes dados de alunos de uma disciplina:

- número de matrícula;
- nome;
- média na disciplina.

Sendo assim:

- a) Defina os tipos abstratos de dados para representar a estrutura de lista encadeada que manipula tais dados;
- b) Implemente uma função que insira, em ordem crescente de número de matrícula, os dados de um novo aluno na lista. Essa função deve obedecer ao seguinte protótipo:

```
bool insere(Lista* lista, Aluno info);
```

Função	Descrição
<code>void iniciaPilha(Pilha *pilha);</code>	Inicia uma nova pilha dinâmica;
<code>int tamanho(Pilha *pilha);</code>	Consulta o número de elementos de uma pilha;
<code>Caixa* topo(Pilha *pilha);</code>	Consulta o elemento no topo de uma pilha;
<code>Caixa* pop(Pilha *pilha);</code>	Retira um elemento do topo de uma pilha;
<code>void push(Pilha *pilha, Caixa *caixa);</code>	Insere um elemento no topo de uma pilha.
<code>void destroi(Pilha *pilha);</code>	Esvazia e libera a memória alocada para uma pilha.

Tabela 1: Operações básicas com uma pilha dinâmica

```
typedef struct {
    float peso;
    char nome[100];
} Caixa;
```

Exercício 5. Um depósito guarda caixas de madeira em pilhas numeradas que são distribuídas por um grande galpão. Para gerenciar as informações sobre o armazenamento, o sistema de administração do depósito utiliza o tipo abstrato de dados *Pilha*, que implementa as funções descritas na tabela a seguir:

O tipo de dados *Pilha* foi declarado para armazenar ponteiros para variáveis *Caixa*, cujo campo *peso* armazena o peso da caixa e o campo *objeto* contém o nome do objeto guardado na caixa, conforme descrito no código abaixo:

O sistema mantém um vetor de ponteiros para o tipo de dados *Pilha*, em que cada elemento aponta para uma pilha de caixas no galpão. Sendo assim, responda às seguintes questões:

- (0,5 pontos)** Defina o(s) tipo(s) abstrato(s) de dados em C necessário(s) para a definição de uma *Pilha* que atue no sistema;
- (1,5 pontos)** Escreva uma função em C para empilhar uma nova caixa escolhendo a primeira pilha - na ordem do vetor -, de tal forma que a caixa não seja colocada sobre uma mais leve e que cada pilha tenha no máximo 8 caixas. Os parâmetros da função são:
 - o vetor de pilhas (depósito);
 - o tamanho do vetor (*n*); e
 - o ponteiro para a nova caixa a ser empilhada.

O retorno da função deve ser o índice que indica a pilha onde a caixa foi empilhada, ou -1 se não foi possível empilhar a caixa obedecendo aos critérios desejados.

Exercício 6. Considere *F* uma fila não vazia e *P* uma pilha vazia. Usando apenas uma única variável temporária *x*, e quatro operações:

- `x = pop(P)`
- `push(x)`
- `x = dequeue(F)`
- `enqueue(F)`

escreva uma função/procedimento em C para reverter a ordem dos elementos em *F*.

Exercício 7. Se você tem que escolher uma representação por lista encadeada ou uma representação usando posições contínuas de memória para um vetor, quais informações são necessárias para você selecionar uma representação apropriada? Como esses fatores influenciam a escolha da representação?

Exercício 8. Considere a existência de um tipo abstrato `FilaDinamica` de números de ponto flutuante. Sem conhecer a representação interna desse tipo, e usando as funções declarados em sala de aula (e contidas no arquivo `FilaDinamica.h`), implemente uma função que receba três filas: `fRes`, `f1`, `f2`, e transfira alternadamente os elementos das filas `f1` e `f2`, para a fila resultante `fRes` (começando pela fila `f1`). Note que, ao final dessa função, as filas `f1` e `f2` vão estar vazias e a fila `fRes` conterá todos os valores que estavam originalmente nas outras filas. Essa função deve obedecer o seguinte protótipo:

```
void combinaFilas (Fila* fRes, Fila* f1, Fila* f2);
```