Universidade Tecnológica Federal do Paraná – Campus Apucarana
Estruturas de Dados (ED62A)
"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 dos mesmos.

Exercício 1. O que é uma Pilha?

Exercício 2. Para que serve uma Pilha? Em que tipos de situações uma pilha pode ser utilizada?

Exercício 3. O que é uma Fila?

Exercício 4. Para que serve uma Fila? Em que tipos de situações uma fila pode ser utilizada?

Exercício 5. Qual a diferença entre uma Fila e Pilha?

Exercício 6. O que significa alocação estática de memória para um conjunto de elementos?

Exercício 7. O que significa alocação dinâmica de memória para um conjunto de elementos?

Exercício 8. 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 9. Identifique e liste aplicações que possam explorar o uso de Pilhas.

Exercício 10. Identifique e liste aplicações que possam explorar o uso de Filas.

Exercício 11. Mostre como implementar uma fila usando duas pilhas.

Exercício 12. Mostre como implementar uma pilha usando duas filas.

Exercício 13. Desenvolva um algoritmo para transferir todos os elementos de uma Pilha P1 para uma Pilha P2. Considere que ambas as pilhas P1 e P2 já existem, ou seja, não precisam ser inicializadas.

Exercício 14. Desenvolva um algoritmo para testar se uma pilha P1 tem mais elementos do que uma Pilha P2. Considere que P1 e P2 já existem e são passadas como parâmetro. Considere também que as Pilhas P1 e P2 possuem elementos inteiros, sem contador interno. Você pode criar pilhas auxiliares, se necessário. Você deve preservar os dados de P1 e P2. Ou seja, ao final da execução dessa operação, P1 e P2 precisam conter exatamente os mesmos elementos que continham no início da operação, e na mesma sequência.

Exercício 15. Desenvolva um algoritmo para verificar se uma Pilha P possui algum elemento igual a um valor X. P e X são passados como parâmetros. Considere que a pilha P possui elementos do tipo inteiro.

Exercício 16. Desenvolva um algoritmo para testar se duas pilhas P1 e P2 são iguais. Duas pilhas são iguais se possuem os mesmos elementos, exatamente na mesma ordem. para propor essa operação, você pode utilizar pilhas auxiliares, se preciso. Considere que P1 e P2 já estão inicializadas e são passadas como parâmetro.

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

Tabela 1: Operações básicas com uma pilha dinâmica

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

Exercício 17. 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:

a) Defina o(s) tipo(s) abstrato(s) de dados em C necessário(s) para a definição de uma Pilha que atue no sistema;

b) 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 (deposito);
   – 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 18. Desenvolva um algoritmo para trocar elementos de duas Filas F1 e F2. Ao final da operação, a fila F2 deve conter os elementos que estavam em F1 antes do início da operação, na mesma ordem, e a fila F1 deve conter os elementos que estavam em F2 antes do início da operação, também na mesma ordem. Considere que F1 e F2 já existem e são passadas como parâmetros.

Exercício 19. 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 20. 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);
```

Exercício 21. Considerando listas encadeadas com valores inteiros, desenvolva uma função que retorne o número de nós da lista que possuem um campo info (chave) com valores valores do que x. Essa função deve obedecer o seguinte protótipo:

```
int maiores(Lista* lista, int x);
```

Exercício 22. 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 23. 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 24. 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 25. 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);
```