10ª Lista de Exercícios
Assunto

Essa lista de exercícios tem como objetivo principal desenvolver algoritmos a partir dos conteúdos abordados em sala de aula. Todos os exercícios também devem ser implementados em linguagem C. Nos programas que pedem para implementar apenas funções desenvolva também o programa principal (main) para testá-los. Não utilize variáveis globais.

1. **(Fácil)** Faça um programa que imprima, na saída padrão, com o uso de sizeof(), o tamanho - em bytes - de uma variável tipo char, uma int, uma float e uma double. Em seguida, faça o mesmo para um char *, um int *, um float * e um double *. Explique por que o tamanho dos ponteiros permanece o mesmo independente do seu tipo. Em seguida, explique por que a especificação do tipo do ponteiro, apesar disso, é necessária.

2. **(Fácil)** Caso você tenha executado o código acima em uma máquina 64-bit, compile o código para 32-bit (no gcc utilize a *flag* -m32) e observe o que acontece com o tamanho dos ponteiros. Explique por que isso ocorre.

3. **(Médio)** Desenvolva um programa em C que leia um número inteiro n e um conjunto n de valores reais, guarde-os em um vetor e então crie outros dois vetores um com os valores impares e outros com valores pares. Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser liberada.

4. **(Médio)** Desenvolva um programa em C que leia uma matriz NxN de inteiros. Em seguida, calcule a soma dos elementos de cada coluna, armazenando o resultado da soma em um vetor de N elementos. Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser liberada.

5. **(Fácil)** Desenvolva um programa em C que solicita ao usuário a quantidade de alunos de uma turma e aloca um vetor de notas (números reais). Depois de ler as notas, imprime a média aritmética. Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser liberada.

6. **(Difícil)** Desenvolva um programa em C que leia todos os elementos de uma matriz 3x3. A restrição e que se os números digitados forem pares devem ser armazenados somente em linhas pares e os ímpares, somente em linhas ímpares. Quando não houver mais espaço para armazenar um número par ou ímpar, seu programa deve realocar mais memoria e continuar a ler os próximos números. Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser liberada.

7. **(Fácil)** Desenvolva um programa em C que gere dois números positivos M e N aleatórios e gere automaticamente uma matriz MxN (M e N gerados previamente) com número aleatórios. Imprima as coordenadas (linha x coluna) do maior e do menor elemento. Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser liberada.

8. **(Médio)** Desenvolva um programa em C que gere aleatoriamente duas matrizes A e B de dimensões MxN. Os valores são gerados no intervalo $[0, 100]$. Em seguida, crie uma outra matriz com o resultado das seguintes operações:
   (a) A soma das duas matrizes.
   (b) A diferença das duas matrizes.
   (c) A transposta da matriz A.

   Obs: não deve ocorrer desperdício de memória; e após ser utilizada a memória deve ser liberada.

9. Desenvolva um programa em C que calcule a soma de duas matrizes MxN de números reais (double). A implementação deste programa deve considerar as dimensões fornecidas pelo usuário. Dica: represente a matriz através de variáveis do tipo *double ***, usando alocação dinâmica de memória.

10. **(Difícil*)** Crie um registro para os funcionários de uma empresa com as seguintes informações: número do funcionário, nome, idade, telefone, cargo e salário. O programa deve manter o cadastro de funcionários dinâmico, ou seja, cada usuário inserido deve ser alocado um espaço na memória e cada usuário deletado deve ser liberado espaço na memória.

    Crie funções que realizem as seguintes tarefas:
    (a) Inserir funcionário.
    (b) Lista todos os funcionários cadastrados.
    (c) Procurar funcionário pelo nome ou pelo número, e imprimir seus dados.
    (d) Eliminar o cadastro de um funcionário.
    (e) Editar as informações de um funcionário, dado o seu número de registro.

    *OBS: o código fica extremamente mais simples com o uso de structs.

11. **(Difícil*)**Faça um programa que gerencie o estoque de um mercado. O sistema deve ter os seguintes dados do produto: Código (inteiro e único), Nome, Preço e Quantidade.

    Crie as seguintes funções:
    (a) Inclusão de produto.
    (b) Alteração de produto. Obs: O código do produto não pode ser alterado.
    (c) Exclusão de produto.
    (d) Inclusão de pedido.
    (e) Alteração de pedido.
    (f) Exclusão de pedido.
    (g) Balanço das vendas.

Um pedido é composto por um código de produto e a quantidade. Localize este código no vetor dinâmico de produtos e, se houver quantidade suficiente para atender ao pedido integralmente, atualize o estoque e informe o usuário. Se por algum motivo não for possível atender ao pedido, mostre uma mensagem informando qual erro ocorreu.
OBS:Também é recomendado o uso de struct neste exercício.