Lista de Exercícios

Alocação Dinâmica

1. Implemente as funções com o seguintes cabeçalhos:
    a. `double* somaVet(double* vetA, double* vetB, int dim)`, que recebe dois vetores do espaço de dimensão `dim` e retorna um vetor da mesma dimensão com a soma, componente a componente, dos vetores;
    b. `double* prodVetEscal(double* vetA, double a, int dim)`, que recebe um vetor do espaço de dimensão `dim` e um escalar (um valor numérico), e retorna um vetor da mesma dimensão com o produto de cada elemento do vetor pelo escalar;
    c. `void imprimeVet(double* vetA, int dim)`, que mprimir todos os componentes de um vetor de espaço de dimensão dim, um do lado do outro, separados por virgula e delimitado por parênteses.
2. Implemente um programa em C que gere dois vetores alocados dinamicamente, no espaço de dimensão 5, de forma aleatória e:
    a. Retorne o vetor soma dos dois vetores. Imprima o vetor resultante na tela;
    b. Multiplique o vetor resultante da operação anterior por um valor fornecido pelo usuário e imprima o vetor resultante na tela.
3. Para implementar uma aplicação que simule uma conversa com o computador vamos implementar as seguintes funções:
    a. `char* geraPalavra(int tam)` que gera uma string com uma "palavra" gerada de forma aleatória misturando letras e alternando consoantes e vogais. As palavras devem ter no mínimo dois caracteres e no máximo `tam` caracteres, todos minúsculos.
    b. `char* geraFrase(int tam)` que gera uma string com uma "frase" gerada de forma aleatória, misturando palavras . As palavras podem ser geradas com a função implementada no exercício anterior. A frase deve ter exatamente `tam` caracteres. O primeiro caractere da frase deve ser maiúsculo.
    c. Implemente uma aplicação em C que converse com o usuário (;-)
4. Crie uma função, para gerar uma imagem, que recebe dois inteiros contendo o número de linhas e colunas da mesma. A função retorna um ponteiro de ponteiros para inteiros (`int** geraImagem(int lin, int col)`). A matriz gerada representa uma imagem, contendo `lin x col` pixels, e em cada posição da mesma temos a intensidade da cor do pixel correspondente, preenchidos de forma aleatória, com valores entre 0 e 255.
5. Crie uma função para liberar o espaço de memória ocupado por uma imagem matriz de inteiros. A função recebe um ponteiro de ponteiros para inteiros, contendo valores entre 0 e 255, dois inteiros contendo o número de linhas e colunas do array (void liberaImagem(int** img, int lin, int col). Implemente uma aplicação que:
    a. Gere uma imagem de 640 x 480 pixels preenchidos de forma aleatória e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo).
    b. Libere a memória alocada para a imagem;

6. Crie uma função que recebe um ponteiro de ponteiros para inteiros, contendo valores entre 0 e 255, dois inteiros contendo o número de linhas e colunas do array, e retorna um ponteiro para inteiros (`int* histograma(int **img, int lin, int col)`). O array representa uma imagem, contendo `lin x col` pixels, e em cada posição da mesma temos a intensidade da cor do pixel correspondente. A função retorna o endereço de uma array de inteiros, alocado dinamicamente, contendo o histograma da imagem. Isto é, o array conte quantas vezes aparece na matriz cada uma dos 256 possíveis valores. Este exercício já foi feito anteriormente sem alocação dinâmica de memória. Implemente uma aplicação que:
    a. Gere uma imagem de 640 x 480 pixels preenchidos de forma aleatória.
    b. Determine o histograma da imagem e imprima os 10 primeiros valores, os 10 últimos e 10 valores centrais do mesmo.
    c. Modifique a imagem para que a mesma contenha um fundo branco (cor 255) com um retângulo de 300 x 200 pixels no meio de cor 125 e, dentro dele, outro retângulo de 100 por 100 de cor preta (0).
    d. Determine o histograma da imagem e imprima os 10 primeiros valores, os 10 últimos e 10 valores centrais do mesmo.
    e. Libere a memória alocada para a imagem;
7. Crie uma função que recebe um ponteiro de ponteiros para inteiros, contendo valores entre 0 e 255, dois inteiros contendo o número de linhas e colunas do array, e retorna um ponteiro de ponteiros para inteiros (`int** filtroMedia(int **img, int lin, int col)`). O array representa uma imagem, contendo `lin x col` pixels, e em cada posição da mesma temos a intensidade da cor do pixel correspondente. A função retorna o endereço de uma nova imagem, alocado dinamicamente, gerada a partir da aplicação de um filtro de média na imagem original. Isto é, cada pixel, nas coordenadas `(i, j)` da nova imagem, se constrói como a média das intensidades das cores nos pontos `(i, j)`, `(i+1, j)`, `(i-1, j)`, `(i, j+1)`, e `(i, j-1)` da imagem original. Para os pixels nas bordas mantemos a cor original. Implemente uma aplicação que:
    a. Gere uma imagem de 640 x 480 pixels preenchidos de forma aleatória e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo).
    b. Aplique o filtro de média na imagem gerada e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo) da nova imagem filtrada.
    c. Libere a memória alocada para a imagem;
8. Uma matriz bidimensional pode ser substituída por um array unidimensional. Com esta finalidade precisa ser definida uma estratégia para mapear as coordenadas da matriz em índices do array. Uma forma de se fazer isto é: Dada uma matriz de tamanho `lin x col`, podemos ordenar os elementos da matriz linha por linha no array de forma que o elemento na posição `(i, j)` da matriz será mapeado para as posição de índice `(col*i + j)` do array. Utilizando como base este mapeamento:
    a. Implemente uma nova versão da função do exercício 4 (`int* geraImagem(int lin, int col)`) para que ela agora retorne uma imagem armazenada um array;

b. Crie uma função que retorna a cor de um pixel específico de uma imagem armazenada como um array (`int getColor(int* img, int i, int j)`);
c. Implemente uma nova versão da função do exercício 6 (`int* histograma(int *img, int lin, int col)`).
d. Implemente uma nova versão da função do exercício 7 (`int* filtroMedia(int *img, int lin, int col)`).
e. Implemente uma aplicação que:
    i. Gere uma imagem de 640 x 480 pixels preenchidos de forma aleatória e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo).
    ii. Determine o histograma da imagem e imprima os 10 primeiros valores, os 10 últimos e 10 valores centrais do mesmo.
    iii. Aplique o filtro de média na imagem gerada e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo) da nova imagem filtrada.
    iv. Libere a memória alocada para a imagem;

9. Uma imagem colorida pode ser representada de diversas formas. Uma delas envolve armazenar a intensidade de cada um dos canais de cores vermelho, verde e azul (Red, Green and Blue ou RGB). Cada um destes canais pode conter um valor de intensidade entre 0 e 255. A seguinte estrutura pode ser utilizada para representar a cor de um pixel:

```
struct pixel{
    unsigned char r;
    unsigned char g;
    unsigned char b;
};

typedef struct pixel Pixel;
```

Implemente:

a. uma nova versão da função do exercício 4 (`Pixel* geraImagem(int lin, int col)`) para que ela agora retorne uma imagem armazenada um array de pixels;
b. Crie uma função que retorna a cor de um pixel específico de uma imagem armazenada como um array (`Pixel getColor(Pixel* img, int i, int j)`);
c. Implemente uma nova versão da função do exercício 6 (`int* histograma(Pixel *img, int lin, int col, int chanel)`). Esta nova versão retorna o histograma de uma canal de cor específico.

d. Implemente uma nova versão da função do exercício 7 (`Pixel* filtroMedia(Pixel *img, int lin, int col)`).
e. Implemente uma aplicação que:
    i. Gere uma imagem de 640 x 480 pixels preenchidos de forma aleatória e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo).
    ii. Determine o histograma de cada um dos três canais da imagem e imprima os 10 primeiros valores, os 10 últimos e 10 valores centrais de cada um.
    iii. Aplique o filtro de média na imagem gerada e mostre os 25 pixels do canto superior esquerdo (a submatriz de 5x5 do campo superior esquerdo) da nova imagem filtrada.
    iv. Libere a memória alocada para a imagem;