Aula Prática: Ponteiros e Alocação Dinâmica

Objetivo: *Capacitar ao aluno trabalhar com variáveis do tipo ponteiro, bem como, alocar e liberar memória durante a execução do programa.*

Exercícios:

1. O método *Selection Sort* é um método de ordenação bastante conhecido. Tal método consiste em percorrer o vetor a ser ordenado a fim de localizar o menor elemento (ou maior, se ordenação decrescente), colocando-o em sua posição correta, ou seja, trocando-o com o primeiro (ou último) elemento da parte do vetor que ainda não está ordenada. Então, esse processo se repete para o restante do vetor que ainda não foi ordenado, até que todos os elementos estejam em suas posições corretas. Por exemplo, no início (primeira iteração) de uma ordenação ascendente, nenhum elemento do vetor ainda foi ordenado. Portanto, o menor elemento será trocado com aquele que está na primeira posição do vetor, a qual passa a ser desconsiderada no restante do processo. Na segunda iteração, a posição alterada é a 1 e assim por diante até todo o vetor estar ordenado. Percebam que, a cada iteração, o número de posições do vetor a serem percorridas na busca é decrementada. Maiores detalhes sobre esse método podem ser encontrados na vídeo-aula do Prof. André Backes (link: https://programacaodescomplicada.wordpress.com/2014/05/16/ed1-aula-50-ordenacao-selectionsort/)

   Faça um programa que utilize um vetor de elementos do tipo **aluno**, o qual deve ser alocado dinamicamente após o usuário informar a quantidade de alunos de uma turma. A estrutura aluno deve possuir os campos: nome (*string*), média (*double*) e número de faltas (*int*). Além disso, o programa também deve permitir que o usuário preencha os registros de todos os alunos; ordene-os em ordem crescente pela média (usando o *selection sort*); e imprima o vetor ordenado. Após a impressão, o espaço alocado pelo vetor deve ser liberado. Vale destacar que todos as entradas devem ser consistidas para garantir que seus valores são válidos (ex: não permitir quantidade negativa).

OBS: a liberação pedida é desnecessária, uma vez que ocorre no final do programa, toda a área de memória destinada ao programa é liberada.

2. Faça um programa que mantenha uma tabela (na memória) para cadastro de bebidas que utiliza a seguinte estrutura:

Bebida:

Nome	Volume (ml)	Preço
char[20]	int	float

A tabela é representada por um vetor de ponteiros para a estrutura, capaz de comportar no máximo 20 bebidas. Todos os elementos desse vetor devem ser devidamente inicializados com NULL. Após a criação e inicialização da tabela, o programa deve apresentar as opções descritas abaixo, e solicitar que o usuário informe a operação desejada. Esse processo deve ser feito repetidamente, até que ele solicite a saída:

[1] Inserir registro

[2] Apagar último registro

[3] Imprimir tabela

[4] Sair

OBS: o programa também deve prover a consistência dos dados de entrada e da situação do vetor (ex: não pode apagar de um vetor vazio ou inserir em um vetor cheio), bem como realizar a alocação e liberação dos registros de modo a garantir o uso otimizado da memória.