

# MC102 - Algoritmos e Programação de Computadores

## Lista de Exercícios 7

1. Suponha que criamos uma estrutura para armazenar produtos de um supermercado:

```
typedef struct Produto{
    char nome[80];
    double preco;
    int quantidade;
}Produto;
```

Implemente duas funções, uma que devolve o vetor ordenado por preços e outra que devolve o vetor ordenado pela quantidade de itens no estoque. Os protótipos são:

```
void ordenaPreco(Produto vet[], int n);
void ordenaQuant(Produto vet[], int n);
```

2. Suponha que criamos uma estrutura para armazenar Datas:

```
typedef struct Data{
    int dia;
    int mes;
    int ano;
}Data;
```

Implemente um algoritmo que receba um vetor de Datas como parâmetro e que retorne as datas em ordem cronológica. Protótipo da função é:

```
void ordena(struct Data vet[], int tam);
```

**Dica:** Ordene o vetor separadamente por cada um dos campos.

3. Suponha que criamos uma estrutura para armazenar dados de pessoas e uma outra estrutura para armazenar dados de várias pessoas como uma base de dados.

```
typedef struct Pessoa{
    int rg;
    int cpf;
    char nome[80];
}Pessoa;
```

```
typedef struct Base{
    int armazenado; //Deve sempre corresponder ao número de pessoas na base
    Pessoa pessoas[100];
}Base;
```

Crie funções para cada uma das operações abaixo:

- Cria base: esta função devolve uma Base onde o campo **armazenado** é inicializado com 0.

```
Base cria_base();
```

- Inclui Pessoa: esta função recebe como parâmetro um dado do tipo Pessoa e o inclui na base (também passada por parâmetro) caso já não exista na base uma pessoa com o mesmo RG. A função devolve 1 caso a inclusão tenha ocorrido, devolve 0 caso a Base esteja cheia e devolve -1 caso já exista uma pessoa com o RG informado.

```
int insere_base(Pessoa p, Base base);
```

- Exclui Pessoa: esta função recebe como parâmetro um dado do tipo int representando o RG de uma pessoa e o exclui da base caso esteja presente. A função devolve 1 caso a exclusão tenha ocorrido, e devolve 0 caso não exista uma pessoa com o RG informado.

```
int remove_base(int rg, Base base);
```

4. Suponha que criamos uma estrutura para armazenar dados de pessoas e um vetor para armazenar dados de várias pessoas:

```
typedef struct Pessoa{  
    int rg;  
    int cpf;  
    char nome[80];  
}Pessoa;
```

```
Pessoa cadastro[100];
```

Suponha que o vetor esteja ordenado em ordem crescente por valor de RG. Implemente uma função de busca por RG, que opera como a busca binária, e que caso exista uma pessoa no cadastro com o RG a ser buscado, devolve o índice deste no cadastro, e devolve -1 caso não exista uma pessoa com o RG a ser buscado.

5. Refaça as funções de busca sequencial e busca binária vistas em aula assumindo que o vetor possui chaves que podem aparecer repetidas. Neste caso, você deve retornar em um outro vetor todas as posições onde a chave foi encontrada.

Protótipo da função: **void busca(int vet[], int tam, int chave, int posicoes[], int \*n)**

- Você deve devolver em **posicoes[]** as posições de **vet** que possuem a **chave**, e devolver em **\*n** o número de ocorrências da chave.
  - **OBS:** Na chamada desta função, o vetor **posicoes** deve ter espaço suficiente (por exemplo, tam) para guardar todas as possíveis ocorrências da chave.

6. O que será impresso pelo programa abaixo?

```
#include <stdio.h>

struct T{
    int x;
    int y;
};
typedef struct T T;

void f1(T *a);
void f2(int *b);

int main(){
    T a, b, *c, *d;
    c = &a;
    a.x = 2;
    a.y = 4;
    b.x = 2;
    b.y = 2;
    d = c;
    f1(d);
    b = *d;
    printf("x: %d --- y: %d\n",b.x,b.y);
}

void f1(T *a){
    f2(&(a->x));
    f2(&(a->y));
}

void f2(int *b){
    *b = 2*(*b);
}
```