ALGORITMOS II

1ª LISTA DE EXERCÍCIOS

Desenvolva todos os exercícios listados em Linguagem C.

1 Defina uma estrutura que represente bandas de música. Essa estrutura deve ter o nome da banda, que tipo de música ela toca, o número de integrantes e em que posição do ranking essa banda está dentre as suas cinco bandas favoritas. Se a banda não estiver tiver entre suas cinco bandas favoritas, utilize o número zero neste campo. Em seguida, desenvolva as funções que seguem:

a) Faça uma função que cadastre um determinado número de bandas. A quantidade de bandas a serem cadastradas deve ser fornecida pelo usuário. As bandas devem ser armazenadas em um vetor em que cada elemento é uma estrutura conforme definido anteriormente.
b) Faça uma função que faça uma busca de acordo com o ranking da banda. Essa função deve exibir todas as informações da banda de um determinado ranking.
c) Crie uma função que solicite ao usuário um tipo de música e exiba as bandas com esse tipo de música no seu ranking.
d) Crie uma função que solicite o nome de uma banda ao usuário e informe se ela está entre suas bandas favoritas ou não

2 Considere a estrutura abaixo para representar um ponto em uma grade 2D.

```
struct Ponto{
  int x;
  int y;
};
```

Implemente um programa que verifique se um determinado ponto `P` está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo `v1` e superior direito `v2`. A função deve retornar se o ponto está localizado dentro do retângulo ou não.

3 Um número complexo é dividido em duas partes na forma z=a+b.*i*, onde a e b são números reais. Defina um tipo chamado **complexo** usando ***struct*** para contemplar estas duas partes. Além disso, faça funções para ler, somar, subtrair e multiplicar números complexos.

A linguagem C possui a biblioteca `complex.h` mas <u>não</u> deve ser usada para desenvolver esse exercício.

4 Um número racional é dividido em duas partes inteiras, o numerador e o denominador. Defina um tipo chamado **racional** usando ***struct*** para contemplar estas duas partes. Além disso, faça funções para ler, somar, subtrair, multiplicar, dividir e simplificar números racionais. Por simplificar, $\frac{a}{b}$ tem seu numerador e denominador divididos pelo máximo divisor comum entre eles.

5 Suponha um cadastro de alunos onde cada aluno contém os seguintes campos: Nome, Data de Nascimento (dia, mês, ano), RG, Sexo, Endereço (Rua, Cidade, Estado, CEP), RA (Registro de Aluno) e CR (Coeficiente de Rendimento: número real no intervalo [0,1]). Faça um programa que realize o cadastro de alunos em um vetor com 100 posições. O programa deve manipular este cadastro com as seguintes opções:

1) Inserir um novo aluno no cadastro.
2) Ordenar o cadastro por nome em ordem alfabética.
3) Ordenar o cadastro por CR, maiores primeiro.
4) Ler o valor de um RA e imprimir os dados do aluno no cadastro com mesmo RA.
5) Imprimir o cadastro na ordem atual.

6 Escreva uma função que receba dois parâmetros do tipo `dma`, cada um representando uma data valida, e retorne o número de dias que decorreram entre as duas datas. ´

```
struct dma {
  int dia;
  int mes;
  int ano;
};
```

7 Considere a estrutura

```
struct vetor{
  float x;
  float y;
  float z;
};
```

que represente um vetor no R^3. Desenvolva um programa que calcule a soma de dois vetores.

8 Declare uma estrutura chamada **TipoReg** contendo os seguintes campos: Nome, RG, Salario, Idade, Sexo, DataNascimento; onde Nome e RG são *strings*, Salario é real, Idade é inteiro, Sexo é *char* e DataNascimento é uma estrutura contendo três inteiros, dia, mês e ano. Declare uma estrutura chamada **TipoCadastro** que contém dois campos: um campo funcionário, contendo um vetor com 100 posições do tipo **TipoReg** e outro campo inteiro, Quant, que indica a quantidade de funcionários no cadastro.

TODOS OS EXERCÍCIOS SEGUINTES FAZEM USO DO TIPO TipoCadastro.

9 Faça uma função, **IniciaCadastro**, que inicia uma variável do tipo **TipoCadastro**. A função atribui a quantidade de funcionários como zero.

10 Faça uma função, **LeFuncionarios**, com uma variável do tipo TipoCadastro como parâmetro de entrada. A função deve ler os dados de vários funcionários e colocar no vetor do cadastro, atualizando a quantidade de elementos não nulos. A função deve retornar com o cadastro atualizado. Lembre que o cadastro não suporta mais funcionários que os definidos no vetor de funcionários.

11 Faça uma função, chamada **ListaFuncionarios**, que imprime os dados de todos os

funcionário.

12 Faça duas funções para ordenar os funcionários no cadastro. Uma que ordena pelo nome, **OrdenaNome**, e outra que ordena pelo salário, **OrdenaSalario**.

13 Faça uma função, **SalarioIntervalo**, que tem como parâmetros: um parâmetro do tipo **TipoCadastro** e dois valores reais v_1 e v_2, $v_1 \leq v_2$. A função lista os funcionários com salário entre v_1 e v_2. Depois de imprimir os funcionários, imprime a média dos salários dos funcionários listados.

14 Faça uma função que dado um cadastro, imprime o nome do funcionário e o imposto que é retido na fonte. Um funcionário que recebe até R$1000,00 é isento do imposto. Para quem recebe mais de R$1000,00 e até R$2000,00 tem 10% do salário retido na fonte. Para quem recebe mais de R$2000,00 e até R$3500,00 tem 15% do salário retido na fonte. Para quem recebe mais de R$3500,00 tem 25% do salário retido na fonte.

15 Faça uma função, **BuscaNome**, que tem como entrada o cadastro e mais um parâmetro que é um nome de um funcionário. A função deve retornar uma estrutura (tipo **TipoReg**) contendo todas as informações do funcionário que tem o mesmo nome. Caso a função não encontre um elemento no vetor contendo o mesmo nome que o dado como parâmetro, a estrutura deve retornar com nome igual a vazio.

16 Desenvolva uma função, **AtualizaSalario**, que tem como parâmetro o cadastro de funcionários. A função deve ler do teclado o RG do funcionário a atualizar. Em seguida, a função lê o novo salário do funcionário. Por fim, a função atualiza no cadastro o salário do funcionário com o RG especificado.

17 Faça uma função, chamada **ListaMaraja**, que tem como parâmetro o cadastro e devolve uma estrutura contendo os dados de um funcionário que tem o maior salário.

18 Faça uma função, **RemoveFuncionario**, que tem parâmetros o cadastro e o RG de um funcionário. A função deve remover do cadastro o funcionário que contém o RG especificado. Lembre-se que os elementos não nulos no vetor do cadastro devem estar contíguos. Além disso, caso um elemento seja removido, a variável que indica a quantidade de elementos deve ser decrementada de uma unidade. Caso não exista nenhum elemento no vetor com o RG fornecido, a função não modifica nem os dados do vetor nem sua quantidade.

19 Faça uma função, **ListaAniversarioMes**, que tem como entrada o cadastro e um número inteiro que corresponde a um mês do ano. A função deve imprimir o nome dos funcionários que nasceram neste mês, assim como o dia do seu nascimento.

20 Faça uma função, **ListaAniversarioSexo**, que tem como entrada o cadastro, três inteiros: dia, mês e ano, que correspondem a uma data e um caractere (sexo) com valor ‘F’ ou ‘M’. A função deve imprimir o nome dos funcionários que nasceram nesta data e com sexo igual ao definido pelo parâmetro.