Para cada item (a, b, c, ...) crie uma rotina de acordo com a especificação. Utilize registro ou conjunto de registros de acordo com cada problema. Para a ordenação de um conjunto de registros faça as adaptações necessárias do código DOSORT e para a busca de um registro adapte o código DOSEARCH (pg 4).

1. Defina um registro para:

    a. Tempo em horas, minutos e segundos.
    b. Uma lista telefônica.
    c. Descrição de um automóvel para uma locadora.
    d. Descrição de um livro para uma biblioteca.

2. Tendo **horas**, **minutos** e **segundos** em um registro:

    a. Ler as informações.
    b. Mostrar as informações.
    c. Transformar as informações em segundos.
    d. Mostrar as informações em segundos.

3. Tendo dois horários (**horas**, **minutos** e **segundos**) em registro:

    a. Ler as informações.
    b. Obter a diferença entre os dois horários.
    c. Mostrar a diferença entre os dois horários.

4. Para um cadastro de contas bancárias (**número da conta**, **nome do cliente** e **saldo**) com no máximo 100 clientes. Faça:

    a. Menu de opções para Cadastro, Visualização e Exclusão, Sair.
    b. Cadastrar as contas. Não pode haver contas com o mesmo número. A rotina deve aceitar um cadastro por vez.
    c. Visualizar todas as contas de um determinado cliente.
    d. Excluir a conta com menor saldo, supondo que não existem saldos iguais.

5. A prefeitura de uma cidade fez uma pesquisa entre os seus habitantes, coletando dados sobre o **salário**, **idade**, **sexo** e **número de filhos**. Crie um algoritmo que leia os dados de um número indeterminado de pessoas e, ao final, mostre:

    a. A média de idade das mulheres com salário inferior a R$ 300,00;
    b. A média de salário da população;
    c. A média do número de filhos;
    d. O maior salário;
    e. A menor idade.

6. Faça um algoritmo que controle o estoque de uma loja de brinquedos. Atualmente existem 40 itens, cada um contendo **código**, **descrição**, **preço de compra**, **preço de venda**, **quantidade em estoque** e **estoque mínimo**.

    a. Cadastre os produtos;
    b. Mostre o valor do lucro que pode ser obtido com a venda de cada produto e o percentual que esse valor representa;
    c. Mostre os produtos com quantidade em estoque abaixo do estoque mínimo permitido.

7. Considere que um médico armazena algumas informações sobre os seus 20 pacientes (**nome**, **idade**, **sexo**, **peso** e **altura**). Crie um algoritmo que leia essas informações e determine:

    a. O nome da pessoa mais pesada;
    b. Os nomes e idades das pessoas que estejam acima do seu peso ideal;
    c. Os nomes das pessoas que estejam abaixo do seu peso ideal, mostrando, ainda, o peso que essas pessoas deverão adquirir para atingirem esse peso ideal.

    Observação: utilize esses cálculos para determinar o peso ideal.
    Homens: (72.7 * altura) – 58
    Mulheres: (62.1 * altura) – 44.7

8. Faça um algoritmo que gerencie uma locadora de fitas de videocassete. Como é uma locadora nova, ela não deve ter mais de 10 clientes. Cada cliente tem os seguintes dados: **código**, **nome**, **sexo**, **data de nascimento**, **RG**, **CPF**, **endereço**, **cidade**, **estado**, **número total de fitas já locadas** e **número de fitas que estão locadas atualmente**. O código do cliente é seu número no registro da locadora. Faça o seguinte:

    a. Inclua um novo cliente. O número total de fitas locadas e o número de fitas que estão locadas atualmente devem ser zero;
    b. Mostre os clientes cadastrados;
    c. Remova um cliente, desde que ele não esteja com fitas locadas no momento;
    d. Faça a locação de novas fitas a um cliente, desde que ele não tenha nenhuma em seu poder. Deve-se entrar com o código do cliente e solicitar o número de fitas que deseja locar (nesse momento o campo “fitas locadas” atualmente deve ser atualizado);
    e. Faça a devolução de fitas. Deve-se solicitar o código do cliente e, se esse for encontrado, deve-se perguntar quantas fitas estão sendo devolvidas (o cliente não pode devolver mais fitas que o valor do campo “fitas locadas atualmente” e “total de fitas já locadas” devem estar atualizados;
    f. Mostre os clientes que estão com fitas locadas.

Declare um tipo chamado *tiporeg*, definido como um tipo de registro 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 um registro contendo três inteiros, dia, mês e ano. Declare um tipo de registro chamado TipoCadastro que contém dois campos: Um campo, Funcionario, 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 rotina, *InicializaCadastro*, que inicializa uma variável do tipo TipoCadastro. A rotina atribui a quantidade de funcionários como zero.

10. Faça um procedimento, *LeFuncionarios*, com parâmetro uma variável do tipo TipoCadastro. A rotina deve ler os dados de vários funcionários e colocar no vetor do cadastro, atualizando a quantidade de elementos não nulos. Caso o nome de um funcionário seja vazio, a rotina deve parar de ler novos funcionários. A rotina deve retornar com o cadastro atualizado. Lembre que o cadastro não suporta mais funcionários que os definidos no vetor de funcionários.

11. Faça uma rotina, chamada *ListaFuncionarios*, que imprime os dados de todos os funcionários.

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

13. Faça uma rotina, *SalarioIntervalo*, que tem como parâmetros: um parâmetro do tipo TipoCadastro e dois valores reais `v1` e `v2`, $v1 \leq v2$_ A rotina lista os funcionários com salário entre `v1` e `v2`. Depois de imprimir os funcionários, imprime a média dos salários dos funcionários listados.

14. Faça uma rotina que dado um cadastro, imprime o nome do funcionário e o imposto que é retido na fonte. Um funcionário que recebe até R$1.000,00 é isento de imposto. Para quem recebe mais que R$1.000,00 e até R$2.000,00 tem 10% do salário retido na fonte. Para quem recebe mais que R$2.000,00 e até R$3.500,00 tem 15% do salário retido na fonte. Para quem recebe mais que R$3.500,00 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. O procedimento deve retornar um registro (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, o registro deve ser retornado com nome igual a vazio.

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

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

18. Faça uma rotina que tem como parâmetros o cadastro e o RG de um funcionário. A rotina 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 rotina não modifica nem os dados do vetor nem sua quantidade.

19. Faça uma rotina, *ListaAniversarioSexo*, que tem como entrada o cadastro, três inteiros: dia, mes e ano, que correspondem a uma data e um caracter (sexo) com valor 'F' ou 'M'. A rotina deve imprimir o nome dos funcionários que nasceram nesta data e com sexo igual ao definido pelo parâmetro.

Método de ordenação DOSORT

```
//a é o vetor com os elementos, c é a quantidade de elementos
void dosort(int a[ ], int c) {
        int j, pig;
        for(pig=0;pig<c;pig++)
        {
                for(j=0;j<c;j++)
                {
                        if (a[j]>a[pig])
                        {
                                swap(&a[j],&a[pig]);
                        }
                }
        }
}
void swap(int *a, int *b) {
        int temp;
        temp=*a;
        *a=*b;
        *b=temp;
}
```

Método de pesquisa DOSEARCH

//**a** é o vetor com os elementos, **b** é o elemento procurado, **c** é a quantidade de elementos, o retorno é a posição do elemento no vetor caso exista ou -1 caso não exista.

```
int dosearch(int a[ ], int b, int c) {
        int high=c-1;
        int low=0;
        int mid=(low+high)/2;
        do
        {
                if(b>a[mid])
                {
                        low=mid+1;
                        mid=(high+low)/2;
                }
                else if (b<a[mid])
                {
                        high=mid-1;
                        mid=(high+low)/2;
                }
                else if(a[mid]==b)
                {
                        return mid;
                }
        }while(low<=high);

        return -1;
}
```