

**LISTA 6 - REGISTROS**

**Aluno**: Vitor Bruno de Oliveira Barth

**Professor**: Ruy de Oliveira

**Disciplina**: Algoritmos II

Cuiabá

2016

1- Para controle dos veículos em uma determinada cidade (máximo 1000 veículos), a Secretaria de Transportes criou o seguinte registro: - Nome do Proprietário - Número do Chassi - Modelo do veículo - Marca - Cor - Combustível - Ano de Fabricação - Placa Construa um algoritmos para: a) listar todos os proprietários cujos carros são do ano de 1990 ou inferior, e movidos a gasolina. b) listar todas as placas e os proprietários dos veículos que comecem com a letra 'A' e terminem com '0', '2' ou '7'. DICA: faça com que a placa seja um vetor de caracteres! c) trocar o nome do proprietário do veículo informando-se o número do chassi do carro. **Algoritmo { placas }**

**Defina** MAX 500

**Declare** Cadastro veículos[MAX] (fabricação **numérico**, proprietário[30], chassi[30], cor[030], combustível[30], marca[30], modelo[30], placa[30] **literal**)

**Declare** fabricação, escolha, i, pos, bandeira **numérico**

**Declare** proprietário[30], chassi[0], cor[30], combustível[30], marca[30], modelo[30], placa[30] **literlal**

pos ← 0 escolha ← 0

**Enquanto** (escolha != 5) **faça**

**Escreva** (“ 1 – Cadastra veiculo”)

**Escreva** (“ 2 – Listar proprietários dos veículos do ano 1990 ou inferior, movidos a gasolina ”)

**Escreva** (“ 3 – Listar proprietários e placas que comecem com A e terminam com 0, 2 ou 7”)

**Escreva** (“ 4 – Trocar proprietário de um veiculo ja cadastrado”)

**Escreva** (“ 5 - Finalizar”)

**Leia** escolha

**Caso** escolha

1: **Se** (pos < MAX) e**ntão**

bandeira ← 0

**Escreva** (“Insira o nome do proprietário: “)

**Leia** proprietário

**Escreva** (“Insira o Chassi do veiculo “)

**Leia** chassi

**Escreva** (“Insira o marca do veiculo “)

**Leia** marca

**Escreva** (“Insira o modelo do veiculo “)

**Leia** modelo

**Escreva** (“Insira a cor do veiculo “)

**Leia** cor

**Escreva** (“Insira o tipo de combustível usado no veiculo “)

**Leia** combustível

**Escreva** (“Insira o ano de fabricação veiculo “)

**Leia** fabricação

**Escreva** (“Insira a placa do veiculo “)

**Leia** placa

**Se** (pos != 0) **então**

**Para** i **de** 0 **até** (pos – 1) **faça**

**Se** (strcmp(veiculos[i].chassi, chassi) == 0) **então**

Bandeira ← 1 { bandeira sinaliza se existe }

**Fim-se**

**Fim-para**

Fim-se

**Se** (bandeira = 0) { Veiculo não cadastrado, gravar informações } **então**

strcpy(veiculo[pos].chassi, chassi)

strcpy(veiculo[pos].proprietario, proprietario)

strcpy(veiculo[pos].marca, marca)

strcpy(veiculo[pos].modelo, modelo)

strcpy(veiculo[pos].combustivel, combustivel)

strcpy(veiculo[pos].cor, cor)

veiculo[pos].fabricacao = fabricacao

strcpy(veiculo[pos].placa, placa)

pos ← pos + 1

**Senão**

Escreva (“Esse veiculo ja esta cadastrado”)

**Fim-Se**

**Senão**

**Escreva** (“Limite de cadastramento de veículos foi atingido “)

**Fim-se**

2 : **Se** (pos != 0) **então** {Se a houver alguma veiculo cadastrado faça}

bandeira ← 0

**para** i **de** 0 até (pos – 1) faça

**Se** (veiculo[i].fabricação <= 1990 && veiculo[i].combustível, "gasolina" == 0) **então**

**Escreva** ("Proprietário: " veiculo[i].proprietário);

bandeira ← 1 {bandeira sinaliza se existe}

**fim**-**se**

**fim**-**para**

**Se** (bandeira == 0) **então**

**Escreva** ("Não possui nenhum veiculo cadastrado com tais requisitos")

**Então**

**Escreva** ("Não há nenhum veiculo cadastrado")

**Fim-se**

3 : **Se** (pos != 0) **então**

bandeira ← 0

**Para** i **de** 0 **até** (pos – 1) **faça**

**Se** (veiculo[i].placa[0] =='A') **então**

**Se** (veiculo[i].placa[7] =='0' ou veiculo[i].placa[7] =='2' ou veiculo[i].placa[7] =='7') então

**Escreva** (“Proprietário: “, veiculo[i].proprietário “Placa: " veiculo[i].placa)

bandeira = 1

**Fim**-**se**

**Fim**-**se**

**Fim**-**para**

**Se** (bandeira == 0) **então**

**Escreva**("Não há nenhum veiculo cadastrado com tais requisitos")

**Senão**

**Escreva**("Não há nenhum veiculo cadastrado")

**Fim**-**se**

4 :**Se**(pos != 0) **então**

bandeira = 0;

**escreva**("Insira o Chassi do veiculo desejado ");

**Leia** chassi

**para** i **de** 0 **até** (pos – 1) faça

**Se**(strcmp(veiculo[i].chassi, chassi) == 0) **então**

**Escreva** ("Insira o nome do novo proprietário ")

**Leia** (veiculo[i].proprietário)

bandeira = 1

**Fim-se**

**Fim-para**

**Se** (bandeira == 0) **Então**

**Escreva**("Veiculo não encontrado")

**Senão**

**Escreva** ("Não ha nenhum veiculo cadastrado");

**Fim-se**

**Fim-caso**

**FIM-ALGORITMO**

**2 – Faça um algoritmo que efetue reserva de passagens aéreas de uma determinada companhia. O programa deverá ler os números dos aviões e o número de lugares disponíveis em cada avião. Utilize um vetor de 4 posições, onde cada posição representa um avião, e um outro vetor também de quatro posições para armazenar os lugares disponíveis. Mostre o seguinte menu de opções: 1 – Cadastrar os números dos aviões 2 – Cadastrar o número de lugares disponíveis em cada avião 3 – Reservar passagem 4 – Consultar por avião 5 – Consultar por passageiro 6 – Finalizar**

**algoritmo** { algoritmo empresa aerea }

**defina** X = 4 { numero de avioes }

**declare** Assento[X] registro(tAssentos numérico, nPassageiro[60] **literal**)

**declare** Aviao[X] registro(numero **literal**)

**declare** i, j, vLido, contadorAvioes, contadorAssentos[4], achou **numérico**

**declare** nAviao, nPassageiro **literais**

**enquanto** vLido != 6

**imprima** 1 – Cadastrar os números dos aviões

2 – Cadastrar o número de lugares disponíveis em cada avião 60

3 – Reservar passagem

4 – Consultar por avião

5 – Consultar por passageiro

6 – Finalizar

**leia** vLido

**caso** vLido

1: **para** i **de** 0 **a** X **faça**

**leia** nAviao

achou = 0;

**se** i = 0 **então**

Aviao[0].numero ← nAviao

contadorAssentos[0] ← 0

**fim**-**se**

**para** i de 0 a X

**se** (verificaDuplicados(nAviao, Aviao[i].numero) = 1) **então**

achou ← 1

**fim**-**se**

**fim**-**para**

**se** **achou** = 1 faça

imprima Registro Duplicado

i ← i-1

**senão**

Aviao[i].numero ← nAviao

contadorAssentos[i] ← 0

**fim**-**para**

2: **para** i **de** 0 a X **faça**

**imprima** Insira o numero de assentos livres no avião Aviao[i].numero

**leia** nAviao

Assentos[i].tAssentos ← nAviao1

**fim**-**para**

3: **imprima** Insira o numero do aviao

**leia** nAviao

**para** i de 0 a X

**se** (verificaDuplicados(nAviao, Aviao[i].numero) = 1

achou = 1

nAviao ← i

**fim** **se**

**fim** para

se **achou** = 1 então

**leia** nPassageiro

**se** (contadorAssentos < tAssento[i])

Assento[i].nomePassageiro[contadorAssentos] = nPassageiro

contadorAssentos[i]++

**senão**

imprima Aviao Cheio

**fim**-**se**

4: **imprima** Insira o numero do aviao

**leia** nAviao

**para** i de 0 a X

**se** (verificaDuplicados(nAviao, Aviao[i].numero) = 1

achou = 1

nAviao ← i

**fim** **se**

**fim** **para**

**se** **achou** = 1 **então**

**para** i de 0 a contadorAssentos[nAviao]-1 **faça**

imprima i, Assento[i].nPassageiro

**fim**-**para**

**fim**-**se**

5: **imprima** insira o nome do passageiro

**leia** nPassageiro

**para** i de 0 a X

**para** j de 0 Assento[j].tAssentos faça

**se** (verificaDuplicados(nPassageiro, Assento[j].nPassageiro) = 1)

**imprima** Aviao[i].numero, nPassageiro

**fim**-**se**

**fim**-**para**

**fim**-**para**

**fim**-**enquanto**

**Fim**-**Algortimo**

**3 - Um banco esta informatizando seus controle de clientes e contas. No primeiro momento o banco deseja guardar as informações de ate 20000 clientes. Cada cliente tem os seguintes dados : Nome, idade, endereço, número de suas contas (15 no máximo) e CGC. As contas válidas tem numero diferente de 0. Cada conta possui um só cliente. As informações das contas são as seguinte : cliente, tempo em que é cliente e saldo atual. (Se existem 2000 clientes com 15 contas no máximo então devem existir 30000 contas). Escreva um algoritmo com duas funções que façam: a) retorne o número de clientes com saldo negativo em mais de uma conta b) retorne o número de cliente que abriram conta a mais de 10 anos e que tenham idade menor que 30 anos.**

**Algoritmo** { banco }

**defina** X 2000 {tottal clientes}

**declare** cliente[X] registro (nome, endereco, cgc **literais**, idade, numeroConta[16], **numericos**)

**declare** conta[X\*15] registro(cgc, tempo, saldo **numericos**)

**declare** nome, endereco, cgc **literais**

**declare** i, j, achou, vLido, contadorClientes, negativos, contadorContas, cliente, tempo, saldo, idade, nConta **numericos**

contadorClientes = 0;

contadorContas = 0;

vLido = 0;

**enquanto** (vLido != 3) {

negativos = 0;

achou = 0;

**imprima** 1 – Cadastrar cliente

2 – Cadastrar conta

3 – Numero clientes com saldo negativo

4 – Numero de clientes com conta há mais de 10 anos e menos de 30 anos d idade

**leia** vLido

**caso** vLido

1: **leia** nome

**leia** endereco

**leia** cgc

**leia** idade

**se** (nClientes < X) **então**

**para** i de 0 a contadorClientes **faça**

**se** (verificaDuplicados(cgc, cliente[i].cgc) = 1)

achou = 1

**fim-se**

**fim-para**

**se** (achou = 0)

cliente[contadorClientes].nome = nome

cliente[contadorClientes].endereco = endereco

cliente[contadorClientes].cgc = cgc

cliente[contadorClientes].idade = idade

**senao**

imprima CGC duplicado

**fim**-**se**

**senao**

imprima Numero de Clientes Excedido

**fim-se**

2: **leia** cgc

**para** i **de** 0 **a** contadorClientes **faça**

**se** (verificaDuplicados(cgc, cliente[i].cgc) = 1)

achou = 1

cliente = i

**fim**-**se**

**fim**-**para**

**se** **achou** = 1

conta[contadorContas].cgc = cgc

**leia** tempo

**leia** saldo

conta[contadorContas].tempo = tempo

conta[contadorContas].saldo = saldo

cliente.numeroContas[0]++;

cliente.numeroContas[cliente.numeroContas[0]] = contadorContas

**senao**

**imprima** cliente não encontrado

**fim-se**

3: **para** i **de** 0 **a** contadorContas **faça**

**para** i **de** 0 **a** contadorClientes **faça**

**para** j **de** 1 **a** cliente[i].numeroContas[0] **faça**

**se** conta[numeroContas[j]].saldo < 0 **faça**

negativos++

**fim**-**se**

**fim-se**

**fim-se**

**fim-para**

**imprima** negativos

4: **para** i de 0 a contadorContas faça

**para** i de 0 a contadorClientes faça

**para** j de 1 a cliente[i].numeroContas[0] faça

**se** cliente[i].idade < 30 && conta[numeroContas[j]].tempo > 10 **faça**

negativos++

**fim-se**

**fim-se**

**fim-se**

**fim-para**

**imprima** negativos

**fim-caso**

**fim-enquanto**

**fim-algoritmo**

**4- Em certo município, vários proprietários de imóveis estão em atraso com o pagamento do imposto predial. Desenvolver um algoritmo que calcule e escreva o valor da multa a ser paga por esses proprietários.**

**Algoritmo**

**defina** X 20 { numero de apartamentos }

**declare** imovel[X] **registro**(identificac literial, **imposto** numerico, mesesatraso **numérico**)

**declare** tabela[5][3] **numerico**

tabela = {{0,50,1},{51,180,2},{181,500,4},{501,1200,7},{1200,99999999,10}}

**declare** identific literal, imposto numerico, mesesatraso numerico, n, multa, vLido numerico, contador **numerico**

vLido = 0

contador = 0

**enquanto** vLido != 3

**imprima** 1 – registre imovel

2 – imprima multa imovel especifico

**achou** = 0

**leia** vLido

**caso** vLido

1: **leia** identific

**se** contador = 0 **faça**

**leia** imposto

**leia** mesesatraso

imovel[contador].identific = identific

imovel[contador].imposto = imposto

imovel[contador].mesesatraso = mesesatraso

**senão**

**para** i de 0 a contador faça

**se** verificaDuplicados(identific, imovel[i].identific) = 1

achou = 1

**fim**-se

**fim**-**para**

**se** achou = 1

**imprima** identificacao **duplicada**

**senao**

**leia** imposto

**leia** mesesatraso

imovel[contador].identific = identific

imovel[contador].imposto = imposto

imovel[contador].mesesatraso = mesesatraso

**fim**-**se**

**fim**-**se**

2: **leia** identific

**para** i **de** 0 a contador **faça**

**se** verificaDuplicados(identific, imovel[i].identific) = 1

achou = 1

n = i

**fim**-**se**

**se** achou = 0

imprima nenhum imovel encontrado

**senao**

imprima identificacao imposto mesesatraso

**para** i **de** 0 **a** 4

**se** imposto >= tabela[i][0] && imposto <= tabela[i][1]

multa = imovel[n].imposto \* (tabela[i][3]/100)\*imovel[n].mesesatraso

**fim-para**

**fim-se**

**fim** **enquanto**

**fim** **algoritmo**