PORTUGOL | DanteMarinho

**// Ficha 1 | Questão 03. Transformar temperatura Celsius em Fahrenheit**

inicio

real celsius, fah

escrever "Insira uma temperatura em graus Celsius: "

ler celsius

fah <- 1.8 \* celsius + 32

escrever "Temperatura em Fahrenheit: ", fah

fim

Questão 8. Escreva um algoritmo que simule um pequeno computador de bordo automóvel, que calcule o consumo médio de um automóvel a partir do número de quilómetros efectuados e da quantidade de combustível gasto.

Se o consumo médio for superior a 5 l/100 Km, deve imprimir a mensagem “consumo elevado”;

Se o consumo médio for inferior a 4,5 l/100Km deve imprimir a mensagem “consumo baixo”;

Noutras situações imprimir “consumo normal”.

//calcula o consumo médio de um automóvel a partir do número de quilómetros efectuados e da quantidade de combustível gasto

INICIO // Exercicio 08

real km, litros, media

Ler km, litros

media <- litros \* 100 / km // calculo da média de consumo

SE media > 5 ENTAO

ESCREVER "consumo elevado!!!\n", "média = ", media, " litros/100"

SENAO

SE media < 4.5 ENTAO

ESCREVER "consumo baixo\n", "média = ", media, " litros/100"

SENAO

ESCREVER "consumo normal\n", "média = ", media, " litros/100"

FIM SE

FIM SE

FIM

// **FICHA 02 | 03 - Ano bissexto**

Ajuda no link: https://pt.wikipedia.org/wiki/Ano\_bissexto

inicio

inteiro ano

// gera número aleatório entre 1900 e 2012

ano <- 1900 + aleatorio ( ) \* ( 2012 - 1900 )

escrever "Ano " , ano

escrever "\n\n"

se ( ( ano % 4 = 0 e ano % 100 =/= 0 ) ou ano % 400 = 0 ) entao

escrever ano , " é ano bissexto."

senao

escrever ano , " não é ano bissexto."

fim se

fim

**FICHA 02 | JOGO ADIVINHA (Portugol)**

inicio

inteiro chute , num\_aleatorio , tentativas

num\_aleatorio <- aleatorio ( ) \* 10

tentativas <- 0

texto opcao

repete// opcao de repetir o jogo

repete// jogo a rodar

escrever "Escolha um número: "

ler chute

se chute < num\_aleatorio entao

escrever "Maior!\n\n"

tentativas <- tentativas + 1

senao

se chute > num\_aleatorio entao

escrever "Menor!\n\n"

tentativas <- tentativas + 1

fim se

fim se

ate chute = num\_aleatorio ou tentativas >= 10

se chute = num\_aleatorio entao

escrever "Acertou!!!\nO número é: " , num\_aleatorio , "\n"

escrever "Foram " , tentativas + 1 , " tentativas.\n\n"

senao

escrever "Game Over! Acabaram as " , tentativas , " tentativas!"

fim se

escrever "Novo jogo(S/N)?"

ler opcao

ate opcao = "N" ou opcao = "n" ou opcao = "NAO" ou opcao = "nao"

escrever "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

escrever "\* FIM DO JOGO \*\n"

escrever "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

fim

// FICHA 02 - Questão 04 | Obtem o fatorial de um número

// By Dante Marinho

inicio

inteiro cont , num , fatorial

fatorial <- 1

ler num

para cont de 2 ate num// posso iniciar por 1 ou por 2

fatorial <- fatorial \* cont

proximo

escrever "O fatorial de " , num , " é " , fatorial

fim

**// FICHA 02 - Questão 05 | Programa que verifica se um número é primo ou não**

inicio

inteiro num , cont , conta\_zero

conta\_zero <- 0

// Poderia ser definida uma variável (logico primo) por exemplo,

// inicializar como verdadeiro e atribuir como falso se alguma divisão der resto 0

escrever "Insira um número para saber se é primo: "

ler num

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

para cont de 2 ate ( num - 1 )

se num % cont = 0 entao

conta\_zero <- conta\_zero + 1

fim se

proximo

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

se conta\_zero > 0 entao

escrever "Não primo!"

senao

escrever "PRIMOOOOOOOOOOO!"

fim se

fim

**// FICHA 02 - Questão 06 | Sequencia de Fibonacci**

inicio

inteiro num\_termos , fibonacci , aux1 , aux2 , temp , cont

aux1 <- 0

aux2 <- 1

fibonacci <- 0

escrever "Número de termos a imprimir: "

ler num\_termos

se num\_termos >= 1 entao

escrever "0 , "

fim se

se num\_termos >= 2 entao

escrever "1"

fim se

para cont de 3 ate num\_termos

escrever " , " , ( aux1 + aux2 )

temp <- aux2

aux2 <- ( aux1 + aux2 )

aux1 <- temp

proximo

fim

**// FICHA 02 - Questão 07 | Saber se é número perfeito**

inicio

inteiro num , soma , cont

soma <- 0

escrever "Número: "

ler num

para cont de 1 ate num / 2

se num % cont = 0 entao

soma <- soma + cont

fim se

proximo

se soma = num entao

escrever "Soma = " , soma , " -> Número perfeito!"

senao

escrever "Soma = " , soma , " -> Não é número perfeito!"

fim se

fim

**// FICHA 03 - Questão 01 | indicar se um dado número é defetivo(deficiente)**

**// Um número defetivo (ou deficiente) é aquele em que**

**// a soma dos seus divisores próprios é menor do que esse número.**

**inicio**

**inteiro num, i, soma**

**texto opcao**

**soma <- 0**

**repete**

**escrever "\nInsira um número: "**

**ler num**

**para i de 1 ate num / 2**

**se num % i = 0 entao**

**soma <- soma + i**

**fim se**

**proximo**

**se soma < num entao**

**escrever "O número ", num, " é defetivo. Soma dos divisores = ", soma**

**senao**

**escrever "O número ", num, " não é defetivo. Soma dos divisores = ", soma**

**fim se**

**soma <- 0 // zerando(reinicializando) a variável soma**

**escrever "\nDeseja verificar outro número? (S/N) "**

**ler opcao**

**ate opcao = "n" ou opcao = "N"**

**fim**

**// FICHA 03 - Questão 04 | Somatório entre dois pares**

// Recebe uma lista de 10 números e soma os números que estão entre dois pares(inclusive)

inicio

inteiro i , soma , pares , n

soma <- 0

pares <- 0

escrever " Insira dez numeros \n"

para i de 0 ate 9

ler n

se pares < 2 entao

se n % 2 = 0 entao

soma <- soma + n

pares <- pares + 1

senao

se pares >= 1 entao

soma <- soma + n

fim se

fim se

fim se

proximo

se pares < 2 entao

escrever "-1"

senao

escrever "A soma é: " , soma , "."

fim se

fim

**// FICHA 03 - Questão 05 | Converter anos terrestres**

**inicio**

inteiro anos\_terra

real conversao , anos\_planeta , dias\_terra

texto nome\_planeta

escrever "Anos terrestres: "

ler anos\_terra

dias\_terra <- 365 \* anos\_terra

escrever "Planeta: "

ler anos\_planeta

escolhe anos\_planeta

caso 1: conversao <- dias\_terra / 88

nome\_planeta <- "Mercúrio"

caso 2: conversao <- dias\_terra / 225

nome\_planeta <- "Vénus"

caso 3: conversao <- dias\_terra / 365

nome\_planeta <- "Terra"

caso 4: conversao <- dias\_terra / 687

nome\_planeta <- "Marte"

caso 5: conversao <- dias\_terra / 4330

nome\_planeta <- "Júpiter"

caso 6: conversao <- dias\_terra / 10752

nome\_planeta <- "Saturno"

fim escolhe

escrever "Anos em " , nome\_planeta , ": " , conversao , " anos"

**fim**

inicio

**// FICHA 04 - Questão 01 | Recebe array com 10 números e no final indica quantos estão acima da média.**

// Variváveis

inteiro i, contador

// num[10] array com 10 números

// contador irá contar os números acima da média

real media, soma, num[10]

para i de 0 ate 9

escrever "Indique o ", (i + 1), "º número : "

ler num[i]

soma <- soma + num[i]

proximo

media <- soma / 10

para i de 0 ate 9

se num[i] > media entao

contador <- contador + 1

fim se

proximo

escrever "Média = ", media, "\nQuantos números acima da média? :: ", contador

fim

**// FICHA 04 - Questão 02 | Pontuação(0 a 20) de 10 participantes num concurso de programação.**

// Gera outro array apenas com as posições positivas

inicio

inteiro i , i\_novo , pontuacao [ 10 ] , positivos [ 10 ]

i\_novo <- 0

para i de 0 ate 9

escrever "Pontua"

ler pontuacao [ i ]

proximo

// construir novo array com as pontuacoes positivas

para i de 0 ate 9

se pontuacao [ i ] >= 10 entao

positivos [ i\_novo ] <- pontuacao [ i ]

i\_novo <- i\_novo + 1

fim se

proximo

escrever "Pontuações positivas: "

se i\_novo = 0 entao

escrever "...não tem participantes com pontuação positiva. "

fim se

para i de 0 ate i\_novo - 1

escrever positivos [ i ] , " "

proximo

fim

**// FICHA 04 - Questão 03 | Nomes e pontuação(0 a 20) de 10 participantes num concurso de programação.**

// Gera outro array apenas com as posições positivas(nome e pontuação)

inicio

inteiro i , i\_novo , pontuacao [ 10 ] , positivo [ 10 ]

i\_novo <- 0

texto nome [ 10 ] , nome\_positivo [ 10 ]

para i de 0 ate 9

escrever "Participante: "

ler nome [ i ]

escrever "Pontuação: "

ler pontuacao [ i ]

proximo

// construir novo array com as pontuacoes positivas

para i de 0 ate 9

se pontuacao [ i ] >= 10 entao

positivo [ i\_novo ] <- pontuacao [ i ]

nome\_positivo [ i\_novo ] <- nome [ i ]

i\_novo <- i\_novo + 1

fim se

proximo

escrever "Pontuações positivas: "

se i\_novo = 0 entao

escrever "...não tem participantes com pontuação positiva. "

fim se

escrever "\n"

para i de 0 ate i\_novo - 1

escrever "Nome: " , nome\_positivo [ i ] , "\t ... Nota: " , positivo [ i ] , "\n "

proximo

fim

**// FICHA 04 - Questão 04 | Faturação mensal, menor, maior e média da faturação**

// Dante Marinho

inicio

texto mes [ 12 ] <- {"Jan" , "Fev" , "Mar" , "Abr" , "Mai" , "Jun" , "Jul" , "Ago" , "Set" , "Out" , "Nov" , "Dez"}

inteiro faturacao [ 12 ] , menor , maior , posicao\_menor , posicao\_maior , i

real soma <- 0 , media

para i de 0 ate 11

escrever "Faturação de " , mes [ i ] , ":"

ler faturacao [ i ]

soma <- soma + faturacao [ i ]

proximo

// atribuição de menor e maior faturação para o 1º mês

menor <- faturacao [ 0 ]

maior <- faturacao [ 0 ]

// Calcular a média

media <- soma / 12

// Verificao o mes de menor e maior faturacao

**// (continuação 04)**

para i de 1 ate 11

se faturacao [ i ] < menor entao

menor <- faturacao [ i ]

posicao\_menor <- i

senao

se faturacao [ i ] > maior entao

maior <- faturacao [ i ]

posicao\_maior <- i

fim se

fim se

proximo

escrever "\nMês de menor faturação: " , mes [ posicao\_menor ] , " " , menor

escrever "\nMês de maior faturação: " , mes [ posicao\_maior ] , " " , maior

escrever "\nMédia da faturação: " , media

fim

**// FICHA 04 - Questão 05 | Indicar se valor existe no array**

// Recebe um array de 10 números inteiros

inicio

inteiro num [ 10 ] , i , procura , posicao

logico existe <- falso

para i de 0 ate 9

ler num [ i ]

proximo

escrever "Número a procurar: "

ler procura

para i de 0 ate 9

se procura = num [ i ] entao

existe <- verdadeiro

posicao <- i + 1// soma 1 para acertar o valor da posição

fim se

proximo

se existe = verdadeiro entao

escrever "Existe o número " , procura , " na posição " , posicao

senao

escrever "Não existe o número " , procura , " na lista."

fim se

fim

**// FICHA 04 - Questão 06 | Array 10 posições ordenado ASC, pesuisar por um valor e inserir**

// OBS: Programa dá erro se pesquisar por um número maior que não tenha na lista

inicio

inteiro num [ 11 ] , valor , i , pos

para i de 0 ate 9// irá ler 10 posições

ler num [ i ]

proximo

ler valor// valor de pesquisa

i <- 0

enquanto num [ i ] < valor faz// enquanto o valor do array for menor que o valor pesquisado

i <- i + 1// avança uma casa

fim enquanto

pos <- i// salva a posição onde deve inserir o novo número

i <- 0// zerar o valor de i

para i de 9 ate pos passo - 1// "andar para trás" vai avançar os números do array, após o valor encontrado

num [ i + 1 ] <- num [ i ]

proximo

num [ pos ] <- valor// insere o valor do novo elemento

i <- 0// zerar o valor de i

para i de 0 ate 10

escrever num [ i ] , " "

proximo

fim

**// FICHA 04 - Questão 07 | Registar pluviosidade de cada mês ao longo de um ano**

inicio

inteiro valor [ 12 ] , i , j , temp

texto mes [ 12 ] <- {"Jan" , "Fev" , "Mar" , "Abr" , "Mai" , "Jun" , "Jul" , "Ago" , "Set" , "Out" , "Nov" , "Dez" , }

texto temp\_mes

para i de 0 ate 11

ler valor [ i ]

proximo

para i de 0 ate 10

para j de i + 1 ate 11

se valor [ i ] < valor [ j ] entao

temp <- valor [ i ]

valor [ i ] <- valor [ j ]

valor [ j ] <- temp

// troca nome do mês

temp\_mes <- mes [ i ]

mes [ i ] <- mes [ j ]

mes [ j ] <- temp\_mes

fim se

proximo

proximo

// Apresentar a nova array

para i de 0 ate 11

escrever "\n" , mes [ i ] , "\t" , valor [ i ]

proximo

fim

**// FICHA 04 - Questão 08 | Estacionamento de 10 lugares**

// Versão 17:07 25/10/2015

// Programa que verifica lugares vagos numa fila de estacionamento e ocupa-os.

inicio

inteiro estacionamento [ 10 ] <- {0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0}// 0 = lugar vago

inteiro i// indice

inteiro cont <- 0// contar número de lugares cheios

inteiro pesquisa\_lugar// lugar a procurar se este está vago

enquanto cont <= 9 faz

repete// ira repetir até que introduza um valor entre 1 e 10

escrever "\n\nIndique um lugar: (de 1 a 10) "

ler pesquisa\_lugar

ate pesquisa\_lugar > 0 e pesquisa\_lugar <= 10

// Verificar se está ocupado ou livre. Se estivar livre, assina o lugar com o valor "1"

se estacionamento [ pesquisa\_lugar - 1 ] = 1 entao

escrever "OCUPADO\n"

senao

escrever "LIVRE\n"// diz que está livre, ocupa este lugar e soma 1 ao contador

estacionamento [ pesquisa\_lugar - 1 ] <- 1

cont <- cont + 1

fim se

// Apresentar os lugares

escrever "Lugares do estacionamento: \n\n"

escrever " E S T A C I O N A M E N T O\n"

escrever "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

para i de 0 ate 9

escrever estacionamento [ i ] , " | "

proximo

escrever "\n  |   |   |   |   |   |   |   |   |   |"

fim enquanto

escrever "\n\n##########################################\n"

escrever "### Estacionamento totalmente ocupado! ###"

escrever "\n##########################################\n"

fim

**// FICHA 04 - Questão 09 | Produção mensal da fábrica**

inicio

inteiro producao [ 12 ] , acumulado [ 12 ] , i

real metade

texto mes [ 12 ] <- {"Jan" , "Fev" , "Mar" , "Abr" , "Mai" , "Jun" , "Jul" , "Ago" , "Set" , "Out" , "Nov" , "Dez"}

para i de 0 ate 11

escrever "Produção de " , mes [ i ] , ": "

ler producao [ i ]

se i = 0 entao//se p ,es ek jameorp

acumulado [ i ] <- producao [ i ]// acumulado é a própria producao

senao

acumulado [ i ] <- acumulado [ i - 1 ] + producao [ i ]

fim se

proximo

// resposta (a)

escrever "\nMÊS\tVALOR ACUMULADO \n"

para i de 0 ate 11

escrever mes [ i ] , "\t" , acumulado [ i ] , "\n"

proximo

// ------------------------------------

// resposta (b)

escrever "A produção total anual foi de " , acumulado [ 11 ] , " toneladas."

//-------------------------------------

// resposta (c)

metade <- acumulado [ 11 ] / 2

i <- 0// voltar a por o valor de i = zero

enquanto acumulado [ i ] < metade faz

i <- i + 1

fim enquanto

escrever "\nMetade da producao em " , mes [ i ] , " com o valor de " , acumulado [ i ] , " toneladas."

fim

**// FICHA 04 - Questão 10 | Producao de Parafusos**

// Revisão em 13:29 25/10/2015

// Um parafuso é uma linha, numa tabela de duas colunas (coluna peso e comprimento)

// -> Ler n

// -> Para os n parafusos

// ler peso

// ler comprimeno

// -> Clacular nº parafusos com defeito

// -> Calcular media de peso e do comprimento

inicio

// variavais

inteiro n

inteiro produtos [ 20 ] [ 2 ]// array com 20 linhas e duas colunas (peso e comprimento)

inteiro i , j , peso\_ref , comprimento\_ref

real peso\_med <- 0 , comprimento\_med <- 0

inteiro conta\_defeito <- 0

// Ler n parafusos (linha e coluna)

escrever "Quantos parafusos? (até 20 parafusos)"

ler n

// Referência de peso e comprimento

escrever "Peso referência: "

ler peso\_ref

escrever "Comprimento referência: "

ler comprimento\_ref

para i de 0 ate n - 1

para j de 0 ate 1

// Exibir para o usuário o nome da coluna correto

// A coluna varia quando j = 0 ou j = 1

// 0 para peso e

// 1 para comprimento

se j = 0 entao// (peso)

escrever "Peso do parafuso " , i + 1 , ": "

ler produtos [ i ] [ j ]// ou simplesmente produtos[i][0]

peso\_med <- peso\_med + produtos [ i ] [ j ]

senao// quando j = 1 (comprimento)

escrever "Comprimento do parafuso " , i + 1 , ": "

ler produtos [ i ] [ j ]// ou simplesmente produtos[i][1]

comprimento\_med <- comprimento\_med + produtos [ i ] [ j ]

fim se

proximo

// Calcular defeitos, o mesmo contador para peso e comprimento

// j = 0 => peso, j = 1 => comprimento

se produtos [ i ] [ 0 ] > peso\_ref \* 1.1 ou produtos [ i ] [ 0 ] < peso\_ref \* 0.9 ou produtos [ i ] [ 1 ] > comprimento\_ref \* 1.1 ou produtos [ i ] [ 1 ] < comprimento\_ref \* 0.9 entao

conta\_defeito <- conta\_defeito + 1

fim se

proximo

// Médias de peso e comprimento

peso\_med <- peso\_med / n

comprimento\_med <- comprimento\_med / n

escrever "\nNúmero de parafuros dentro do padrão: " , n - conta\_defeito

escrever "\nNúmero de parafuros com defeito: " , conta\_defeito

escrever "\nMédia de peso: " , peso\_med

escrever "\nMédia de comprimento: " , comprimento\_med

fim

**// FICHA 04 - Questão 11 | Quantidade de vezes de um elementos num array**

// Versao 17:56 25/10/2015

// Verificar quantas vezes ocorre um número pesquisado no array

inicio

inteiro lista [ 20 ] , n\_elementos , pesquisa , i , contador

escrever "Quantos números na lista? (até 20 elementos) "

ler n\_elementos

escrever "\nInsira os números na lista... "

para i de 0 ate ( n\_elementos - 1 )

ler lista [ i ]

proximo

escrever "Número a pesquisar repetições? "

ler pesquisa

para i de 0 ate ( n\_elementos - 1 )

se lista [ i ] = pesquisa entao

contador <- contador + 1

fim se

proximo

escrever "\n\nO número " , pesquisa , " ocorre " , contador , " vezes."

fim

**// FICHA 04 - Questão 12 | Gerar nova lista sem valores duplicados**

// Neste, o programa gera a primeira lista e depois gera a nova lista sem repetições

// 19:01 25/10/2015

inicio

inteiro lista [ 10 ] , nova\_lista [ 10 ] , i , aux

//-----

escrever "Insira 10 valores inteiros: "

para i de 0 ate 9

ler lista [ i ]

proximo

// Novo array

para i de 0 ate 9

se i = 0 entao

nova\_lista [ aux ] <- lista [ i ]

aux <- aux + 1

senao

se i > 0 e lista [ i ] =/= lista [ i - 1 ] entao

nova\_lista [ aux ] <- lista [ i ]

aux <- aux + 1

fim se

fim se

proximo

escrever "Array 1\n"

// Apresentar a 1ª lista

para i de 0 ate 9

escrever lista [ i ] , " "

proximo

escrever "\n\nArray 2\n"

// Apresentar a nova lista

para i de 0 ate aux - 1

escrever nova\_lista [ i ] , " "

proximo

fim

// 13 - Juntar dois arrays - merge

**// FICHA 04 - Questão 13 | Funciona exatamente para o tipo de exemplo do exercício**

// Versão 22:06 25/10/2015

inicio

inteiro lista1 [ 3 ] , lista2 [ 3 ] , merge [ 6 ] , i , j , k

k <- 0

// Receber dados na lista 1

escrever "Lista 1: \n"

para i de 0 ate 2

ler lista1 [ i ]

proximo

// Receber dados na lista 2

escrever "Lista 1: \n"

para i de 0 ate 2

ler lista2 [ i ]

proximo

// Construir nova lista mergida

i <- 0

j <- 0

enquanto k <= 5 faz

//----- CASO DE PARAGEM

// se lista 1 terminou entao entrega o restante da lista 2

se i > 2 entao

merge [ k ] <- lista2 [ j ]

k <- k + 1

j <- j + 1

senao

// se lista 2 terminou entao entrega o restante da lista 1

se j > 2 entao

merge [ k ] <- lista1 [ i ]

k <- k + 1

i <- i + 1

//----- ----- -----

senao

// merger valores das duas listas

se lista1 [ i ] < lista2 [ j ] entao

merge [ k ] <- lista1 [ i ]

k <- k + 1

i <- i + 1

senao

merge [ k ] <- lista2 [ j ]

k <- k + 1

j <- j + 1

fim se

fim se

fim se

fim enquanto

para i de 0 ate 5

escrever merge [ i ] , " "

proximo

fim

**// FICHA 04 - Questão 14 | Dois arrays, determinar elementos iguais**

// Versão 22:55 25/10/2015

inicio

inteiro lista\_a [ 8 ] , lista\_b [ 8 ] , i , aux , cont

// Constroi lista 1

escrever "Array A com 8 elementos: "

para i de 0 ate 7

ler lista\_a [ i ]

proximo

// Constroi lista 2

escrever "Array B com 8 elementos: "

para i de 0 ate 7

ler lista\_b [ i ]

proximo

para i de 0 ate 7

para aux de 0 ate 7

se lista\_a [ i ] = lista\_b [ aux ] entao

cont <- cont + 1

fim se

proximo

proximo

escrever "\nResultado: " , cont , " elementos do array A pertencem ao array B."

fim

C# SHARP

**// FICHA 04 – Questão 04 | Faturação do mês**

. Elabore um algoritmo que permita ler a faturação mensal de um empresa ao longo dos 12 meses do ano, e no final determine: a) o mês de maior facturação b) o mês de menor facturação c) a média mensal de facturação

static void Main(string[] args)

{

string[] mes = new string[12] {"Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"};

int [] faturacao = new int[12];

int maior = 0, menor = 0;

for (int i = 0; i < 12; i++)

{

Console.SetCursorPosition(10, 10);

Console.Write("Faturação de {0}: ", mes[i]);

Console.SetCursorPosition(33, 10);

faturacao[i] = Convert.ToInt16(Console.ReadLine());

}

maior = faturacao.Max();

menor = faturacao.Min();

int pos = Array.IndexOf(faturacao, maior);

Console.WriteLine("Mês de maior faturação: {0}", mes[pos]);

pos = Array.IndexOf(faturacao, menor);

Console.WriteLine("Mês de menor faturação: {0}", mes[pos]);

Console.ReadLine();

}

C# SHARP

**// FICHA 04 – Questão 05 | Pesquisa no array**

Elabore um algoritmo que leia um array de 10 números inteiros. Em seguida, dado um determinado valo de pesquisa, o algoritmo deve indicar se esse valor existe no array, e em caso afirmativo em que posição.

static void Main(string[] args)

{

int[] numeros = new int[10];

for (int i = 0; i < 10; i++)

{

Console.SetCursorPosition(10, 10);

Console.Write("Número:");

Console.SetCursorPosition(23, 10);

numeros[i] = Convert.ToInt16(Console.ReadLine());

} /// for

Console.SetCursorPosition(10, 12);

Console.WriteLine("valor a Pesquisar:");

int pesquisa = Convert.ToInt16(Console.ReadLine());

int pos = Array.IndexOf(numeros, pesquisa);

if (pos == -1)

{

Console.WriteLine("O número não existe");

}

else

{

Console.WriteLine("O valor de pesquisa existe na posição {0}", pos + 1);

}

Console.ReadLine();

}

C# SHARP

**// FICHA 04 – Questão 12 | Imprimir array sem duplicado**

Dado um array de N elementos , ordenados, o seu algoritmos deve gerar um outro array sem valores duplicados.

static void Main(string[] args)

{

// FICHA 4 - EX 12 - DADO UM ARRAY DE

//N ELEMENTOS, DEVEMOS GERAR OUTRO SEM DUPLICADOS

//------------------------------------------------------------------------------------

Console.Title = "Array sem duplicados";

Console.Write("Número de elementos do array:");

int N = Convert.ToInt16(Console.ReadLine());

int[] numeros = new int[N]; // array para leitura dos n elementos

int[] backup = new int[N]; // array com os numeros sem duplicações

for (int i = 0; i < N; i++) // Ler N elementos do Array6

{

Console.Write("Elementos {0} do Array :", i+1);

numeros[i] = Convert.ToInt16(Console.ReadLine());

}

Array.Sort(numeros); // ordena o array de leitura

// Preenche o array backup - sem duplicados

backup[0] = numeros[0];

int j = 1;

for (int i = 1; i < N; i++)

{

if (numeros[i] != numeros[i-1])

{

backup[j] = numeros[i];

j++;

}

}

// Imprime o array backup SEM duplicados

Console.WriteLine("====================");

Console.WriteLine("Array sem duplicados");

Console.WriteLine("====================");

for (int i = 0; i < j; i++)

{

Console.Write("{0} \t ", backup[i]);

}

Console.ReadLine();

}

C# SHARP

**// FICHA 04 – Questão 15 | MENU**

. Elabore um programa que permita gerir uma fila de espera com capacidade máxima para 20 lugares. Quando o programa se inicia, todos os lugares da fila devem estar livres. Layout da fila de espera: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 O seu programa deve conter um menu com as seguintes opções:

**MENU**

1 – Tirar Ticket

2 - Atendimento

3 - Estado da fila de espera

0 - Sair

• Tirar Ticket: deve ocupar o primeiro lugar que estiver livre na fila de espera, indicando na consola a posição ocupada. Se todos os lugares estiverem ocupados deverá surgir a mensagem de “Fila completa”.

• Atendimento: É sempre atendido o elemento que está na primeira posição da fila de espera, num dado momento. Neste caso, todos os restantes elementos da fila de espera devem avançar uma posição à frente.

• Estado da fila de espera: esta opção deve indicar, na consola, o número de lugares ocupados e o número de lugares livres na fila de espera.

static void Main(string[] args)

{

int[] fila = new int[10] {0,0,0,0,0,0,0,0,0,0};

int livres = 0, ocupados = 0;

// EXEMPLO de um programa que se inicia com um MENU

char op = ' ';

do

{

Console.Clear();

Console.WriteLine("MENU");

Console.WriteLine("1 - Tirar Ticket");

Console.WriteLine("2 - Atendimento");

Console.WriteLine("3 - Estado da Fila");

Console.WriteLine("0 - Terminar");

Console.Write(" Opcao:");

op = Convert.ToChar(Console.ReadLine());

switch (op)

{

case '1': int i = 0; // Procurar a primeira posição vazia

while ((fila[i] != 0) && (i<9))

{

i++;

}

if ((i < 10) && (fila[i]==0)) // se indice é <10 => tem lugares vazios

{

fila[i] = 1;

Console.SetCursorPosition(15, 15);

Console.WriteLine("Ticket Nº {0}", i + 1);

Console.ReadLine();

}

else // senão FILA está cheia

{

Console.SetCursorPosition(15, 15);

Console.WriteLine("Fila Está cheia!!");

Console.ReadLine();

}

break;

case '2': int j = 1;

while ((fila[j] ==1) && (j<9))

{

fila[j - 1] = fila[j];

j++;

}

Console.SetCursorPosition(15, 15);

Console.WriteLine("Atendido um Ticket" );

fila[j-1] = 0;

Console.ReadLine();

break;

case '3': livres = 0;

ocupados = 0;

for (int k=0; k<10;k++)

{

if (fila[k] == 0)

livres++;

else

ocupados++;

}

Console.SetCursorPosition(15, 15);

Console.WriteLine("Lugares Livres: {0}", livres);

Console.SetCursorPosition(15, 17);

Console.WriteLine("Lugares Ocupados: {0}", ocupados);

Console.ReadLine();

break;

default: Console.WriteLine("Opção Inválida!");

break;

}

}

while (op != '0');

}