EXERCÍCIOS DE VETORES

1. Faça um algoritmo que leia um vetor de 30 posições e informe quantos elementos são múltiplos de 2 e quantos são múltiplos de 3.

1. Construa um algoritmo que solicite 5 valores ao usuário, armazene estes em um vetor de 5 posições inteiras. Após, descubra quantos elementos são maiores do que o elemento da primeira posição.

1. Escreva um algoritmo que leia dois vetores de 10 posições e faça a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Exiba o vetor resultante.

1. Faça um algoritmo que leia dois vetores (A e B) de dez posições de números inteiros. O algoritmo deve, então, subtrair o primeiro elemento de A do último elemento de B, e acumular o valor, subtrair o segundo elemento de A do penúltimo elemento de B, acumulando o resultado, e assim por diante. Mostre o resultado do somatório acumulado.

1. Fazer um programa que leia dois vetores unidimensionais A e B, de tamanho 8, e realize a troca dos elementos destes vetores; ou seja, após a execução do programa o vetor B deverá conter os valores fornecidos para o vetor A, e vice-versa.

1. Construa um algoritmo que solicite 5 valores inteiros ao usuário e os armazene em um vetor. Após, deverá ser invertido os valores do vetor utilizando um segundo vetor.

1. Construa um algoritmo que solicite 5 valores inteiros ao usuário e os armazene em um vetor. Após, deverá ser invertido os valores do vetor sem utilizar um segundo vetor.

1. Faça um programa que carregue um vetor com 10 elementos inteiros e verifique a existência de elementos múltiplos de 5, mostrando as posições em que esses elementos aparecem.

1. Leia 30 valores e jogue os pares em um vetor e os ímpares em outro. Após a leitura calcule o somatório dos dois vetores e exiba o de maior valor.

1. Construa um algoritmo que preencha um vetor com 5 valores inteiros. Após verifique se o número 7 se encontra no vetor. Em caso positivo, exiba qual a posição em que ele foi encontrado. Se ele for encontrado mais de uma vez também quantas vezes ele foi encontrado.

1. Faça um algoritmo que preencha um vetor de 10 posições com números aleatórios (entre 10 e 30) e informe quantos números são múltiplos de 5, e em que posição se encontram.

1. Construa um algoritmo que solicite 5 valores ao usuário, armazene estes em um vetor de 5 posições inteiras. Após verifique se o número 7 se encontra no vetor. Em caso positivo, exiba qual a posição em que ele foi encontrado. Se ele for encontrado mais de uma vez também quantas vezes ele foi encontrado.

1. Faça um algoritmo que preencha um vetor de 10 posições com valores gerados (utilize o rand), e garanta que não haja números repetidos. A seguir, ordene o vetor e mostre-o na tela.

1. Diz-se que uma sequência de n elementos é balanceada, com n par, se as seguintes somas são iguais:
   * a soma do maior elemento com o menor;
   * a soma do segundo maior elemento com o segundo menor;
   * a soma do terceiro maior elemento com o terceiro menor.

ex: 2 12 3 6 16 15 é uma sequência balanceada, pois 16+2=15+3=12+6 Dados n (n par) e uma sequência de n elementos, verificar se essa sequência é balanceada. Crie um vetor de 10 elementos e informe se o mesmo é balanceado ou não.

1. Faça um programa que preencha um vetor de 30 posições com os primeiros números primos a partir de 333. Exiba o vetor resultante na tela.

1. Faça um algoritmo que preencha dois vetores de tamanho 10 (sem poder haver elementos repetidos dentro do próprio vetor). A seguir, crie um terceiro vetor que seja a união entre os dois primeiros vetores. Obs.: o vetor união é composto de todos os elementos dos dois vetores, sem repetição. Exemplo:

Vetor A

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 6 | 7 | 10 | 15 | 16 | 17 | 20 |
| 0  Vetor B | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 13 | 14 | 15 |

0 1 2 3 4 5 6 7 8 9

Vetor União

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 13 | 14 | 15 | 16 | 17 | 20 |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

17) Faça um algoritmo que preencha dois vetores de tamanho 10 (sem poder haver elementos repetidos dentro do próprio vetor). A seguir, crie um terceiro vetor que seja a intersecção entre os dois primeiros vetores. Obs.: o vetor intersecção é composto por todos elementos em comum nos dois vetores. Exemplo: Vetor A

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 6 | 7 | 10 | 15 | 16 | 17 | 20 |
| 0  Vetor B | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 13 | 14 | 15 |

0 1 2 3 4 5 6 7 8 9

Vetor Intersecção

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 6 | 7 | 15 |
| 0 | 1 | 2 | 3 | 4 |

18) Faça um algoritmo que preencha dois vetores de tamanho 10 (sem poder haver elementos repetidos dentro do próprio vetor). A seguir, crie um terceiro vetor que seja a diferença entre os dois primeiros vetores. Obs.: vetor diferença é aquele formado pelos elementos que se encontram no Vetor A, mas não no Vetor B. Exemplo:

Vetor A

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 6 | 7 | 10 | 15 | 16 | 17 | 20 |
| 0  Vetor B | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 13 | 14 | 15 |

0 1 2 3 4 5 6 7 8 9

Vetor Diferença

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 10 | 16 | 17 | 20 |
| 0 | 1 | 2 | 3 | 4 |

19) Preencha dois vetores de tamanho 5. A seguir, crie um terceiro vetor composto pela intercalação dos dois primeiros.

Vetor A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 6 | 7 |
| 0  Vetor B | 1 | 2 | 3 | 4 |
| 2 | 5 | 8 | 9 | 10 |

0 1 2 3 4

Vetor Intercalação

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 5 | 4 | 8 | 6 | 9 | 7 | 10 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

20) A pesquisa ou busca binária (em inglês binary search algorithm ou binary chop) é um algoritmo de busca em vetores que segue o paradigma de divisão e conquista. Ela parte do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor. Faça um algoritmo que preencha

um vetor de 20 posições (sem elementos repetidos e ordenado de forma crescente) utilizando rand, solicite ao usuário um valor e informe se este elemento existe ou não no vetor, utilizando a pesquisa binária.