



LISTA DE EXECÍCIOS

- A R R A Y S -

"may the force be with you"

Vetores


1. Gere um vetor com 30 valores inteiros que representam a umidade relativa do ar em cada dia deste período. Alimente o vetor com valores aleatórios entre 12 e 80. Conte e escreva os dias com maior e menor umidade relativa, a média mensal, e a quantidade de dias acima da média.
2. Faça um programa que preencha um vetor com N números aleatórios, sorteados no intervalo entre 0 e N. Imprima como resultado, todos os números sorteados e:
 - a) Os valores que coincidiram com o próprio índice do vetor.
 - b) Os valores que não foram sorteados.
 - c) Os valores sorteados repetidamente (mais de uma vez).
 - d) O menor valor da metade inicial e o maior valor da metade final.
3. Faça um programa que preencha um vetor com N números inteiros aleatórios, sorteados no intervalo entre 0 e X. O programa deve imprimir os números sorteados, e ao final o seguinte resumo:
 - a) Quantos números distintos foram sorteados.
 - b) As posições (índices) do maior e do menor valor.
 - c) O(s) número(s) mais vezes sorteado, e quantas vezes.
 - d) O(s) número(s) menos vezes sorteado, e quantas vezes.
4. Faça um programa C para ler do usuário dois valores inteiros X e Y. Alimente dois vetores de tamanhos X e Y com números inteiros aleatórios, não repetitivos, sorteados no intervalo entre 0 e X+Y (inclusive). Faça a impressão das seguintes informações:
 - a) Vetores X e Y;
 - b) Todos os números exclusivos do vetor X;
 - c) Todos os números exclusivos do vetor Y;
 - d) Os números existentes nos dois vetores;
5. Faça um programa que preencha um vetor com N números aleatórios, sorteados no intervalo entre X e Y. O programa deve informar a posição (índice) onde se encontra o menor número sorteado, seguido da posição onde se encontra o segundo menor número sorteado, e assim sucessivamente...
6. Gere um array de N números inteiros sorteados no intervalo entre X e Y (inclusive). Imprima o array sorteado e depois elimine deste array os valores repetidos, mantendo a ordem original.
A solução deve ser in place, ou seja, todas as operações devem acontecer diretamente no mesmo array de entrada (*in-place*), não sendo permitido portanto utilizar outros arrays auxiliares para ajudar na tarefa. Imprima o array resultante após as exclusões.
7. Gere dois vetores, V1 e V2, cada um com K valores aleatórios sorteados no intervalo entre 0 e X ($0 < K < X$). O usuário deve informar o valor de K e X, desde que $K < X$. O programa **NÃO** deverá aceitar números repetidos no mesmo vetor, e nem o mesmo número no mesmo índice dos dois vetores. Imprima os valores sorteados nos dois vetores, e o resultado da multiplicação dos valores dos índices de V1 e V2.

8. O algoritmo LRU (*Least Recently Used*) é uma política de gerenciamento de memória *cache* que invalida o item que foi acessado menos recentemente. Isso significa que se um item não foi acessado há muito tempo, ele terá prioridade na remoção. Considere um vetor com N elementos, sorteados no intervalo entre X e Y (inclusive). Considerando que os primeiros índices foram os acessados recentemente, exclua da lista o elemento conforme algoritmo LRU (todos os itens iguais devem ser excluídos na mesma iteração). Solução *in-place* (não deve-se usar vetor auxiliar). Execute essa operação N vezes, ou seja, até que o array esteja vazio novamente. Imprima o conteúdo da “memória cache” em cada iteração.
9. Implemente a solução do mesmo problema anterior, mas agora considerando que o algoritmo adotado seja o LFU (*Least Frequently Used*).
10. Métodos: `map`, `filter` e `reduce` são bastante comuns em linguagens de alto nível para tratamento de Arrays. Faça um programa que implemente o método...
MAP => Dado um vetor V1, com N números inteiros sorteados entre 0 e 1023, crie um novo vetor V2, preenchendo-o com os valores do primeiro array convertidos para a base binária. Imprima V1 e V2.
11. Métodos: `map`, `filter` e `reduce` são bastante comuns em linguagens de alto nível para tratamento de Arrays. Faça um programa que implemente o método...
FILTER => Dado um vetor V1, com N números inteiros sorteados entre 0 e 99, e um outro valor inteiro X, crie um novo vetor V2, preenchendo-o apenas com valores do primeiro vetor que são múltiplos de X. Um mesmo valor não deve ser filtrado mais de 1 vez. Imprima V1 e V2.
12. Métodos: `map`, `filter` e `reduce` são bastante comuns em linguagens de alto nível para tratamento de Arrays. Faça um programa que implemente o método...
REDUCE => Dado um vetor V, com N números inteiros sorteados aleatoriamente (entre 0 e 9), imprima a maior sequência crescente do vetor (em caso de empate, imprima a primeira sequência).
13. Gere um vetor de tamanho aleatório (no máximo 30) e preencha-o com valores também aleatórios entre 0 e 10. Sabe-se que o ponto de equilíbrio de um vetor é o índice no qual, partindo deste índice o lado esquerdo possui exatamente o mesmo valor de soma do que o lado direito. Gere e imprima inúmeros vetores até encontrar algum que possua ponto de equilíbrio, e qual é este ponto.
14. Faça um programa que sorteie aleatoriamente N números, no intervalo entre X e Y (inclusive). Após o sorteio, o programa deve imprimir os números sorteados originalmente. Após isso, e a cada iteração, o programa deve imprimir novamente a relação, mas agora trocando as posições do MAIOR elemento pelo MENOR elemento. Uma vez trocadas as posições, estes mesmos número não poderão ser mais trocados. Repita essa operação até quando for possível...

Exemplo de Execução:

Para: N = 8; X = 1; Y = 10;

Original:	4	-	6	-	8	-	9	-	3	-	2	-	7	-	8
1ª Iteração:	4	-	6	-	8	-	2	-	3	-	9	-	7	-	8
2ª Iteração:	4	-	6	-	3	-	2	-	8	-	9	-	7	-	8
3ª Iteração:	8	-	6	-	3	-	2	-	8	-	9	-	7	-	4
4ª Iteração:	8	-	7	-	3	-	2	-	8	-	9	-	6	-	4

15. A bandeira da Itália possui três cores, nesta sequência: 

Faça um programa que sorteia aleatoriamente um vetor de tamanho N com valores que representam as 03 cores da bandeira italiana. Seu programa deve gerar como resultado a quantidade de bandeiras que podem ser formadas a partir desse Array. Todas operações devem ser *in-place* (Não é permitido usar arrays auxiliares).

- 16.** Declare 03 arrays contendo 10 valores aleatórios. Faça a ordenação dos 03 arrays (utilizando o algoritmo de ordenação preferido). Imprima os 03 arrays ordenados e, após isso, faça a operação “merge”, ou seja, mescle os valores dos 03 arrays em um único array, mantendo os valores ordenados.
Mas atenção... não deve-se realizar uma quarta ordenação para obter o resultado final.
- 17.** Faça um programa que gere um vetor com K números aleatórios entre 0 e N. Após isso, peça um número X ao usuário, e consulte pelo valor de PISO (maior número menor ou igual a X) e valor de TETO (menor número maior ou igual a X). Faça várias buscas, sempre utilizando o algoritmo de **Pesquisa Binária**.
- 18.** Faça um programa que alimente um vetor com N números aleatórios sorteados no intervalo entre X e Y. Faça a ordenação deste vetor, mas considerando a frequência de cada número sorteado.
P.Ex.: 5 - 5 - 5 - 5 - 8 - 8 - 8 - 3 - 3 - 6 - 6 - 9
- 19.** Gere um array com N valores inteiros sorteados entre 0 e 99. Após isso, peça ao usuário um valor X e verifique se há no array um par de valores (V1 e V2), tal que:

$$V1 - V2 == X; \quad \text{ou} \quad V1 * V2 == X;$$

$$V1 + V2 == X; \quad \text{ou} \quad V1 / V2 == X;$$
Imprima todos os pares (V1 e V2) que satisfaça alguma dessas condições.
PS.: Pare, pense e analise qual seria a solução mais eficiente para este problema? Imagine as possíveis soluções considerando um valor muito alto para N.
- 20.** Faça um programa que alimenta um array de N inteiros sorteados entre 1 e 10, e onde cada valor representa o número de chocolates em um pacote. Considerando que existam M alunos ($M < N$) na turma, a tarefa do seu programa é informar quais pacotes de chocolate devem ser distribuídos de forma que: (i) cada aluno deve receber apenas um pacote; (ii) A diferença entre a quantidade de chocolates que cada aluno irá receber deve ser a mínima possível. Em caso de empate (como o prof é gente boa) a distribuição deve ser dos pacotes que contiverem mais chocolates.
P.Ex.: Pacotes == [2, 5, 1, 4, 9, 6, 9], sendo M = 3. Resultado Esperado: Distribuir pacotes contendo 4, 5, 6.

Strings

- 21.** Faça um programa em C que leia um valor inteiro correspondente ao seu número de matrícula no curso e, como saída, gera e imprime a string formada pelos dígitos deste número. *P.Ex.: 2024 == “2024”.*
- 22.** Faça um programa que leia, em formato de string, um valor numérico representado na base binária. O programa deve validar se o valor informado pelo usuário realmente é um número binário. Em caso positivo, o programa deve informar o valor correspondente na base decimal.
- 23.** Números romanos são representados por letras, sendo:
- | | | |
|-----------|------------|-------------|
| 'I' == 1 | 'L' == 50 | 'M' == 1000 |
| 'V' == 5 | 'C' == 100 | |
| 'X' == 10 | 'D' == 500 | |
- Faça um programa que leia uma string em formato de número romano, verifique se é válida (não possui caracteres inválidos) e, estando válida, converta para o número decimal respectivo. Repita a operação até ser informada a string “exit” para encerrar a execução do programa.
P.Ex.: “MCMXCIX” => 1999 “MMM DLXVIII” => 3568 “XXZVII” => Inválido.
- 24.** Escreva um programa em C para ler uma frase. A seguir, imprima a frase com: (a) todas as letras maiúsculas, (b) todas as letras minúsculas, (c) início de cada palavra em maiúsculo. Em todos os casos, retirar os espaços em branco em excesso, que são desnecessários. Algoritmo deve ser **in-place**.

25. Programe o sorteio de um “Amigo Oculto” com N participantes. Leia o nome de cada amigo e imprima o resultado do sorteio. Lembre que um amigo não pode tirar ele mesmo (e cuidado com o último).
26. Faça um programa que verifique se um endereço de e-mail é válido ou não. Para um e-mail ser válido, deve possuir um único símbolo '@' (mas que não seja o primeiro), pelo menos um símbolo ponto '.' após o símbolo '@' (mas que não seja o último), e não possuir espaços em branco nem outros caracteres especiais. Faça várias validações na mesma execução do programa.
27. Desenvolva um programa em C que faça a geração de senhas fortes. Uma senha forte é uma string contendo entre 8 e 16 caracteres, com obrigatoriamente: 1 caractere numérico, 1 caractere maiusculo, 1 minúsculo e 1 caractere especial. Imprima 10 senhas fortes geradas aleatoriamente.
28. Faça um programa que leia, em formato de string, dois números inteiros excepcionalmente grandes e imprima o valor da soma destes números.
- P.ex.: "25996478547851225"
 + "1452565475541"

 25997931113326766
29. Faça um programa que leia uma string A e outra string B. O programa deve informar a seguinte informação, conforme o caso: "**String A está contida em B**" ou "**String B está contida em A**" ou "**As strings são incompatíveis**".
30. Faça um algoritmo de "Pattern Search". Dada uma string "text" e outra string "pattern", imprima a quantidade de vezes que a sub-string "pattern" pode ser encontrada dentro da string "text".
- P.Ex.: Text = "AABACAABAABAABBB" Pattern = "AABA" ==> 04 Ocorrências partindo de: [0, 5, 8, 11]
31. O código de barras é essencial na automatização do processo de vendas. Esse código possui um formato conhecido por EAN-13, contendo 13 dígitos numéricos, sendo que os 3 primeiros identificam o País de origem (Brasil == 789), os 9 dígitos seguintes identificam a empresa fabricante e o produto em si, e o último - 13º dígito - é um validador, chamado de “Dígito Verificador - DV” que é calculado com base na sequência anterior, e que serve para dar confiabilidade ao processo de leitura do código de barras. O cálculo do DV no EAN-13 é o seguinte: Considerando do 1º ao 12º dígito, somar todas as posições pares e multiplicar a soma pelo valor 3. Ao resultado do passo anterior, somar todas as posições ímpares da sequência. O DV deverá ser o valor que tornará toda a soma alcançada em um múltiplo de 10. Faça um programa que valide vários Códigos de Barras lidos pelo usuário.
32. Semelhante ao problema anterior, implemente o algoritmo para checar a validade de um número de CPF (pergunte ao professor o melhor método de cálculo).
33. Semelhante ao problema anterior, implemente o algoritmo para checar a validade de um número de cartão de crédito (pergunte ao professor o método de cálculo).
34. Faça um programa que calcule quantos segundos já se passaram no dia de hoje.
- Obs.: Para obter uma *string* com o horário aproximado do sistema utilize a constante `__TIME__`
- `printf("Horário: %s\n", __TIME__);`
35. Faça um programa que leia a data de nascimento do usuário no seguinte formato *string* DD/MM/AAAA. O seu programa deve validar a entrada, tanto no formato indicado, quanto na validade dos valores informados. Após isto, leia novamente outra string DD/MM/AAAA correspondente ao dia atual e calcule a idade completa do usuário (anos, meses e dias).
36. Escreva um programa em C para ler uma frase qualquer e contar o número de palavras existentes na frase. Considere uma palavra como sendo um conjunto maior que dois caracteres separados por um ou mais espaços em branco.

37. Um algoritmo básico de compactação de arquivos consiste em: ler o texto, indexar um código-valor para cada palavra do texto, e após isso, representar o texto através destes códigos. As palavras repetidas, portanto, podem ser substituídas por um único código, conseguindo assim diminuir consideravelmente o tamanho final do arquivo. Implemente um protótipo deste algoritmo de compactação.

38. Faça um programa que leia uma string S, e criptografe-a com o seguinte algoritmo: sorteie um vetor de $\text{strlen}(S)$ números, com valores aleatórios e não-repetitivos entre 0 e $\text{strlen}(s)$. Após o sorteio, embaralhe as letras da mensagem original, de acordo com as posições sorteadas no vetor. Atenção, todos os caracteres têm que ter sua posição inicial alterada!

Exemplo de Execução:

String:	AULA DE ED
Vetor Sorteado:	1369847502
Texto Cifrado:	EADUD LE A

39. Considerando o problema anterior, faça um programa que leia uma String S criptografada, e um vetor de $\text{strlen}(s)$ números inteiros. Valide a consistência do vetor (não deve ter números repetidos e devem estar no intervalo adequado). Se tudo estiver em conformidade, informe a mensagem decifrada...

Exemplo de Execução:

Texto Cifrado:	EADUD LE A
Vetor (Chave):	1369847502
Texto Decifrado:	AULA DE ED

40. Em uma eleição para representante da turma, existem K candidatos. Faça com que seu programa leia o valor de K e o nome de cada um dos candidatos.

Após isso, programe uma espécie de urna eletrônica, onde o voto do candidato é representado pelo seu número de sequência de cadastro na urna (p.ex.: 1 - 2 - 3 - ... - K).

O valor 0 corresponde a um voto em BRANCO, e um valor superior a K representa um voto NULO.

A votação deve ser encerrada assim que for informado um voto com valor negativo.

Ao final da votação, seu programa deve gerar o seguinte relatório:

- a) A quantidade total de votos;
- b) O nome e a quantidade de votos para cada candidato;
- c) A quantidade de votos nulos;
- d) A quantidade de votos em branco;
- e) O nome do vencedor das eleições;
- f) O percentual do vencedor em relação ao número de votos válidos;

Matrizes

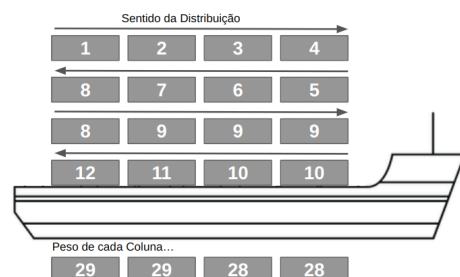
41. Gere aleatoriamente uma matriz $N \times N$ de inteiros sorteados no intervalo entre X e Y (inclusive).

Imprima-a em formato de tabela (linhas e colunas) e informe a localização do **maior** e do **menor** valor da matriz.

42. Gere aleatoriamente uma matriz $N \times N$ de inteiros sorteados no intervalo entre X e Y (inclusive).

Imprima-a em formato de tabela, e os valores que compõem a coluna que resulta na **maior** soma, e os valores da linha que resulta na **menor** soma.

- 43.** Gere aleatoriamente uma matriz $N \times N$ de inteiros sorteados no intervalo entre 0 e N^2 (inclusive). Todo número sorteado, obrigatoriamente deve ser exclusivo em sua linha e em sua coluna. Imprima a matriz em formato de tabela (linhas/colunas).
- 44.** Gere aleatoriamente uma matriz $N \times N$ de inteiros sorteados no intervalo entre X e Y (inclusive). Peça ao usuário o índice de duas colunas, e faça a troca das respectivas colunas. Faça o mesmo com duas linhas escolhidas pelo usuário. Imprima a matriz após realizadas as trocas.
- 45.** Faça um programa que leia o Nome e a Idade de N pessoas. Imprima a relação de nomes ordenada pelas idades de forma decrescente.
- 46.** Gere uma cartela de bingo 5×5 , com números aleatórios entre 1 e 75. (Em uma cartela de bingo, não há números repetidos e os números são apresentados em ordem crescente). Imprima a cartela e faça com que o programa sorteie as bolas, uma a uma, e realize a “marcação” da cartela em tempo real.
- 47.** Crie um programa que leia o nome de N alunos. Após isto, para cada aluno cadastrado, imprima o nome e solicite ao usuário os valores das notas da prova 1, prova 2 e prova 3. Ao final, o programa deverá gerar um relatório em formato de lista, contendo NOME, NOTA 1, NOTA 2, NOTA 3, MÉDIA, e se está APROVADO ou REPROVADO.
- 48.** Gere uma matriz 10×100 , com números aleatórios entre 1 e 4, que se refere às respostas de 10 questões de múltipla escolha, referentes a 100 alunos. Leia (a partir do usuário) um vetor de 10 posições contendo o gabarito de respostas corretas. O seu programa deverá comparar as respostas de cada candidato com o gabarito, e emitir a nota final de cada aluno. O programa também deverá informar: qual foi a média das notas, a maior nota, quantos alunos atingiram a maior nota, a menor nota, e quantos alunos atingiram a menor nota.
- 49.** Considere $N \times N$ valores aleatórios sorteados no intervalo entre 1 e 20, que representam o peso de contêineres para transporte, distribua-os no navio cargueiro de modo que o peso das cargas fique relativamente equilibrado, conforme ilustração a seguir. Imprima também o valor do peso total de cada coluna de contêineres embarcados.



- 50.** Leia uma matriz $N \times 3$, onde cada linha representa uma data de nascimento, e as colunas representam, respectivamente: dia, mês e ano. Após ler todos os valores, faça a impressão das datas ordenadas, em formato: DD/MM/AAAA.