

Algoritmos e Estruturas de Dados I (DCC003 TA1)

Lista de Exercícios 2

Professora: Camila Laranjeira
camilalaranjeira@dcc.ufmg.br

Data da entrega: 23/10/2018

- Os executáveis não devem ser enviados, apenas os arquivos de extensão ".c". Todos devem ser zipados em um arquivo chamado "lista2.zip".

1 Vetores

1.1 Escreva um programa em C para ler N inteiros, armazená-los em um vetor e imprimir as seguintes informações:

- A soma de todos os valores
- O produto de todos os valores
- A quantidade de valores pares
- A quantidade de valores positivos
- A quantidade de valores negativos
- A quantidade de zeros

Entrada: A primeira linha da entrada é um inteiro N referente à quantidade valores que serão lidos ($N \leq 100$), seguido de N linhas com os inteiros que irão compor o seu vetor.

Saída: Cada linha da saída deve conter as informações solicitadas pela questão, **na ordem que foram apresentadas**.

Exemplo:

Entrada	Saída
7	86
0	0
-56	5
127	3
2	2
0	2
20	
-7	

[salve o seu código com o nome: exercicio1-1.c]

1.2 Escreva um programa que instancia um vetor de 10 posições, lê um inteiro, coloca-o na primeira posição desse vetor e preenche cada valor subsequente com o dobro do valor anterior.

Entrada: A entrada é composta de um único valor N ($-50 < N < 50$).

Saída: A saída deve ser na forma de uma tabela com duas colunas, uma referente aos índices do seu vetor, e outra aos valores correspondentes.

Nota: Para imprimir na forma de tabela, ou seja, com espaçamento consistente entre as colunas, basta usar o caracter especial `\t` entre os valores.

```
printf("Valores\tIndices\n");
```

```
printf("%d\t%d\n");
```

Exemplo:

Entrada	Saída	
1	Indices	Valores
	0	1
	1	2
	2	4
	3	8
	4	16
	5	32
	6	64
	7	128
	8	256
	9	512

[salve o seu código com o nome: `exercicio1-2.c`]

1.3 Escreva um programa que lê uma sequência de tamanho desconhecido de valores inteiros, e armazena os pares em um vetor intitulado *par* e os ímpares em um vetor intitulado *impar* sem repetições. Para isso, crie uma função *encontra_valor* que recebe um vetor e um valor e retorna se o valor já existe no vetor. Imprima ambos os vetores na saída.

Entrada: A entrada é uma sequência de inteiros **na mesma linha** finalizada por um caracter não numérico indicando final de sequência.

Saída: A saída contém duas linhas, a primeira imprimindo o vetor *par* e a segunda imprimindo o vetor *impar*.

Exemplo:

Entrada	Saída
0 -56 127 2 0 20 -7 a	0 -56 2 20 127 -7

[salve o seu código com o nome: `exercicio1-3.c`]

1.4 Escreva um programa que lê uma sequência de inteiros de tamanho desconhecido, armazena os valores em um vetor e remove todas as instâncias de um determinado valor que será informado pelo usuário sem criar um segundo vetor.

Dica: Basta que todos os elementos a frente do valor indesejado seja movido uma posição para trás.

Entrada: A entrada começa com o valor da *flag* que indicará final de sequência (**que deve ser um inteiro**). A linha seguinte contém o valor que deve ser removido do vetor. A partir desse ponto, as linhas seguintes contém os valores que irão compor a sequência até que o valor *flag* seja digitado.

Saída: A saída é composta de uma única linha com todos os valores do vetor depois de removidas as instâncias indesejadas.

Exemplo:

Entrada	Saída
-99 0 -5 312 0 0 2 90 -7 -99	-5 312 2 90 -7

[salve o seu código com o nome: exercicio1-4.c]

2 Strings

2.1 Escreva um programa que leia duas palavras e diga qual delas vem primeiro na ordem alfabética.

Entrada: A entrada contém duas linhas, cada qual contendo uma das palavras a ser testada.

Saída: A saída deve imprimir a palavra que vem primeiro alfabeticamente.

Exemplo:

Entrada	Saída
Algoritmos Alexandre	Alexandre
Luciano Andre	Andre

[salve o seu código com o nome: exercicio2-1.c]

2.2 Faça um programa para criptografar uma mensagem. As regras de criptografia são:

- Primeiro, cada letra (maiúscula ou minúscula) deve ser deslocada três posições para a direita, de acordo com a tabela ASCII (a letra 'a' vira 'd', a letra 'y' vira o caracter '—'). Consulte a tabela ASCII em <http://www.asciitable.com/>
- Depois, cada linha deve ser invertida.
- Todos os caracteres a partir da metade (truncada) devem ser deslocados uma posição para a esquerda de acordo com a tabela ASCII ('b' se torna 'a', 'a' se torna '—'). O valor da metade truncado significa que por exemplo a metade de 5 deve ser considerado 2. Na palavra 'tesla' deve-se deslocar os caracteres 'sla'.

Enunciado inspirado na questão 1024 do URI <https://www.urionlinejudge.com.br/judge/en/problems/view/1024>

Entrada: A entrada começa com um inteiro N indicando o número de instâncias que serão criptografadas. As próximas N linhas contém as instâncias.

Saída: A saída também deve conter N linhas com as mensagens criptografadas.

Entrada	Saída
4 Texto #3 abcABC1 vxpdylY .ph vv.xwfxo.fd	3# rvzgV 1FECedc ks. \n{frzx gi.r{hyz-xx

[salve o seu código com o nome: **exercicio2-2.c**]

2.3 Sabendo que a primeira geração de pokemons (e a melhor) possui 151 monstrinhos, faça um programa que receba a lista de pokemons que um mestre possui e imprima quantos ainda faltam para que ele complete a Pokedex.

Enunciado inspirado na questão 2174 do URI <https://www.urionlinejudge.com.br/judge/en/problems/view/2174>

Entrada: A entrada começa com um inteiro N indicando quantos Pokemons o mestre já capturou. As próximas N linhas contém os nomes dos Pokemons capturados, **podendo haver repetições**.

Saída: A saída deve conter apenas um inteiro com o número de Pokemons que faltam para completar a Pokedex.

Entrada	Saída
9 Charmander Caterpie Pidgeot Rattata Rattata Zubat Zubat Zubat Zubat	146

[salve o seu código com o nome: `exercicio2-3.c`]

3 Matrizes

3.1 Escreva um programa que recebe N sequências de tamanho M de ponto flutuante e implemente as seguintes funções:

- *calcMean*: que recebe uma sequência e retorna a média dos valores da sequência.
- *calcStd*: que recebe uma sequência e retorna o desvio padrão dos seus valores.

No *main*, usando essas funções, calcule a média e a variância de cada sequência recebida na entrada.

Entrada: A primeira linha da entrada contém o inteiro N ($0 < N < 100$), referente ao número de sequências. A segunda linha traz M ($0 < M < 1000$), um inteiro que indica a quantidade de valores de cada sequência. As N linhas seguintes contém M valores cada, referentes às sequências que serão armazenadas na sua matriz.

Saída: A saída contém N linhas, e cada linha apresenta a média e o desvio padrão de cada sequência.

Exemplo:

Entrada	Saída
5	0.265 0.612
4	1.607 1.384
0.56 0.41 -0.76 0.85	-1.577 1.019
1.72 -0.49 1.8 3.40	10.219 0.432
-1.74 -1.37 -0.17 -3.03	102.527 11.704
9.56 10.73 10.44 10.15	
95.95 97.46 122.69 94.01	

[salve o seu código com o nome: `exercicio3-1.c`]

3.2 Escreva um programa que cria uma matriz quadrada $N \times N$ de valores inteiros aleatórios, e calcula a soma dos elementos de uma determinada linha ou coluna informada pelo usuário. Cada valor da matriz deve estar dentro do intervalo $\{0, 100\}$.

Enunciado inspirado na questão 1181 do URI <https://www.urionlinejudge.com.br/judge/en/problems/view/1181>

Entrada: A primeira linha contém um inteiro N referente às dimensões da matriz quadrada ($0 < N < 100$). A segunda linha pode conter o caracter "L" (maiúsculo) caso o usuário deseje a soma de uma linha, ou "C" (maiúsculo) caso deseje a soma de uma coluna. A terceira e última linha contém o índice da linha ou coluna da matriz que o usuário deseja somar ($0 < indice < N$).

Saída: A saída deve ser a impressão da matriz **de forma organizada** (use "\t" entre os números de cada linha e "\n" entre as linhas). A seguir imprima a soma da linha ou coluna de acordo com a solicitação do usuário.

Nota: Para gerar um número aleatórios podemos usar a função *rand()* da biblioteca *stdlib.h*. A linha seguinte gera valores entre 0 e 19.

```
int r = rand() % 20;
```

Entrada	Saída
4	77 31 59 72
L	89 88 21 58
0	64 3 24 8
	24 49 18 1
	239

[salve o seu código com o nome: exercicio3-2.c]

3.3 O determinante de uma matriz A , também representado por $|A|$, é dado pela subtração entre o somatório do produto dos termos da diagonal principal e do somatório do produto dos termos da diagonal secundária. Através do seu cálculo é possível saber se uma matriz possui ou não uma inversa. Se $|A| \neq 0$ a matriz A possui inversa.

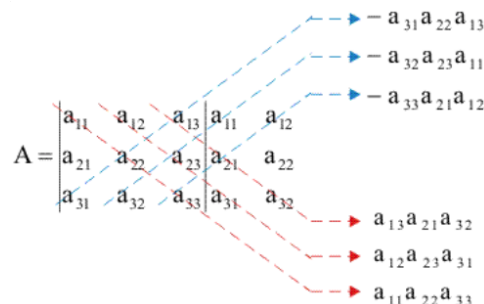
O seu código deve gerar uma matriz aleatória de ordem 3 (Tamanho $N \times N$ sendo $N = 3$) e informar se a matriz possui ou não uma inversa. Garanta que os valores aleatórios serão gerados dentro do limite $\{-10, 10\}$

Nota: O cálculo do determinante de uma matriz de ordem 3 segue os seguintes passos:

- Primeiro repetimos as duas primeiras colunas dessa matriz.

a11	a12	a13	a11	a12
a21	a22	a23	a21	a22
a31	a32	a33	a31	a32

- Depois calculamos os produtos das diagonais principais e os produtos das diagonais secundárias.



- Deve-se pegar o oposto dos produtos das diagonais secundárias e somar com os produtos das diagonais principais.

Exemplo:

Entrada	Saída
	5 0 1 -2 3 4 0 2 -1 Possui inversa

[salve o seu código com o nome: `exercicio3-3.c`]

4 Ponteiros

4.1 Crie um programa que:

- Aloque dinamicamente um array de 5 inteiros
- Peça para o usuário digitar os 5 números no espaço alocado
- Mostre na tela os 5 números
- Libera a memória alocada

[salve o seu código com o nome: `exercicio4-1.c`]

4.2 Faça um programa que leia uma quantidade desconhecida de valores inteiros, até que o usuário digite um valor negativo. Seu programa deve alocar dinamicamente a memória à medida que o usuário digitar novos valores.

Dica: Utilize a função *realloc*.

[salve o seu código com o nome: `exercicio4-2.c`]

5 Tipos definidos pelo usuário

5.1 Escreva um programa que possua a estrutura *carro*, com dois atributos: uma string *modelo* (Fusca, Gol, etc.) e *consumo*, sendo o consumo referente a quantos km esse carro faz por litro. Leia do usuário os modelos e os seus respectivos consumos e imprima qual dos carros é o mais econômico. Seu programa deve alocar dinamicamente o espaço necessário para a estrutura.

Entrada: A primeira linha da entrada é a quantidade N de instâncias que o usuário vai digitar. As $2N$ linhas seguintes contém respectivamente o modelo e o consumo de cada um dos N carros.

Saída: A saída possui uma única linha com o nome do modelo do carro mais econômico.

Exemplo:

Entrada	Saída
3 Fusca 8 Prius 18 Fiesta 13	Prius

[salve o seu código com o nome: exercicio5-1.c]

5.2 Um hospital deseja fazer um sistema de cadastro de pacientes com os seguintes dados: nome, idade, peso e altura. Crie uma estrutura chamada *paciente* para armazenar os dados de cada paciente e um vetor *lista_pacientes* para armazenar os dados informados pelo usuário. Seu programa deve alocar dinamicamente o espaço necessário para a estrutura a medida que o usuário digite novas entradas.

Entrada: A entrada é dividida em um número desconhecido de linhas, sendo cada linha referente aos dados de um único paciente. Em cada linha as informações do paciente são separadas por espaços em branco. A entrada termina quando o usuário digitar "Fim"








Saída: A saída é a impressão apenas dos nomes dos pacientes cadastrados no sistema.

Exemplo:

Entrada	Saída
Andre 35 75 1.81 Virginia 31 68 1.60 Vinicius 29 87 1.89 Lucas 23 78 1.65 Fim	Andre Virginia Vinicius Lucas

[salve o seu código com o nome: exercicio5-2.c]

5.3 Numa partitura, as notas são representadas por símbolos diferentes que indicam a sua duração. Uma música é dividida em uma sequência de compassos, e cada compasso possui um conjunto de notas. Como Pedro é um iniciante, seu professor de música lhe ensinou que a duração das notas de um compasso deve sempre somar 1. No nosso software, cada nota vai ser representada por um caracter, de acordo com a tabela a seguir:

Notes							
Identifier	W	H	Q	E	S	T	X
Duration	1	1/2	1/4	1/8	1/16	1/32	1/64

Por exemplo, Pedro compôs uma música com cinco compassos que obedecem a regra estabelecida por seu professor:

/HH/QQQQ/XXXTXTEQH/W/HH/

Sabendo disso, **escreva um programa que armazene em um vetor de estruturas as características das notas musicais** (a estrutura *nota* deve conter dois atributos: o caracter correspondente e a duração da nota). E informe quantos compassos da entrada obedecem a regra.

Entrada: Cada entrada é composta por uma sequência de caracteres identificando as notas, de modo que o limite de cada compasso é separado por uma barra.

Saída: A saída deve ser um único inteiro correspondendo à quantidade de compassos que obedecem a regra.

Exemplo:

Entrada	Saída
/HH/QQQQ/XXXTXTEQH/W/HW/	4
/W/W/SQHES/	3
/WE/TEX/THES/	0

[salve o seu código com o nome: **exercicio5-3.c**]