Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Disciplina de Fundamentos de Programação
Professora: Mariza Miola Dosciatti
Curso de Engenharia de Computação

Lista 3 - Estruturas Homogêneas

Vetores numéricos e de caracteres, funções, arquivos de cabeçalho

Exercícios para sala de aula

Observações:

- ✓ Vetores precisam ter tamanho definido quando são declarados.
- ✓ Cuidado para não ultrapassar o tamanho de um vetor, ou seja, percorrer índices (ler) ou armazenar valores além do tamanho definido para o vetor.
- ✓ Para gerar números aleatório utilizar a função **rand()** que está na biblioteca **stdlib.h** e necessário incluir a biblioteca **time.h** para usar **time(NULL)**. Inicialmente declarar **srand(time(NULL));** para que seja gerado um início (semente) aleatória para a função **rand()**; depois utilizar **rand(),** que pode ser **rand()** / dividido por alguma constante ou variável ou **rand()** % resto de alguma constante ou variável para obter números em uma determinada faixa.
- ✓ A constante **RAND_MAX** tem valor 32767. As funções **srand()** e **rand()** e a constante **RAND_MAX** requerem a biblioteca **stdlib.h** e a função **time()** requer a biblioteca **time.h**.

1) Criar uma função para ordenar os elementos de um vetor de tamanho *n*, sendo *n* um número positivo. Insira essa função na biblioteca "vetores.h" (criada no Exercício 2 - Lista 1) que já contém as funções para gerarVetorInteiro() e mostrarVetorInteiro(), e que podem ser reutilizadas neste exercício. Usar essas funções para gerar a saída mostrada no exemplo a seguir.

Exemplo:

```
==== VETOR ORIGINAL ====
  6    4    1    0    1   10    1    1    5    0

==== VETOR ORDENADO ====
  0    0    1    1    1    1    4    5    6   10

Deseja repetir o programa (S ou N)? n
```

2) Criar uma função para gerar valores aleatórios com ambos os limites especificados com o seguinte protótipo:

```
void gerarVetorInteiroComFaixa(int vet[], int tam, int limInf, int limSup);
```

Sendo que `limInf` e `limSup` aceitam quaisquer valores positivos.

Dica: Use `vetor[i] = (rand() % ((limSup - limInf) + 1)) + limInf;`

Usando a função gerarVetorInteiroComFaixa(), gerar aleatoriamente um vetor com 20 elementos entre 5 e 20. Em seguida ordenar o vetor. Percorrer o vetor ordenado e mostrar os divisores e a quantidade de divisores de cada um dos valores armazenados. Caso existam elementos repetidos, considerar apenas um deles.

Atenção: Neste exercício, além de utilizar a função gerarVetorInteiroComFaixa(), a função mostrarVetorInteiro() e a função ordenarVetorInteiro(), utilizar também a função mostrarDivisores() e a função qtdeDivisores(), ambas criadas em exercícios anteriores e armazenadas em uma biblioteca chamada "divisores.h".

A seguir um exemplo da execução do programa, utilizá-lo como modelo de saída.

Exemplo:

```
==== VETOR ORIGINAL ====
 14   11    5   16    7   12   12   17   20    9   14   15   19   18   14   14
  5    8    9   10

==== VETOR ORDENADO ====
  5    5    7    8    9    9   10   11   12   12   14   14   14   14   15   16
 17   18   19   20

==== DIVISORES ====
 5 =>  1   5  -  2 divisores
 7 =>  1   7  -  2 divisores
 8 =>  1   2   4   8  -  4 divisores
 9 =>  1   3   9  -  3 divisores
10 =>  1   2   5  10  -  4 divisores
11 =>  1  11  -  2 divisores
12 =>  1   2   3   4   6  12  -  6 divisores
14 =>  1   2   7  14  -  4 divisores
15 =>  1   3   5  15  -  4 divisores
16 =>  1   2   4   8  16  -  5 divisores
17 =>  1  17  -  2 divisores
18 =>  1   2   3   6   9  18  -  6 divisores
19 =>  1  19  -  2 divisores
20 =>  1   2   4   5  10  20  -  6 divisores

Deseja repetir o programa (S ou N)? n
```

3) Gerar um vetor com 20 elementos aleatórios entre 5 e 20. Em seguida ordenar o vetor. Percorrer o vetor ordenado e mostrar os pares e a soma dos pares de cada um dos valores armazenados. Caso existam elementos repetidos, considerar apenas um deles.

Neste exercício, além de utilizar a função gerarVetorInteiroComFaixa(), a função mostrarVetorInteiro() e a função ordenarVetorInteiro(), utilizar também uma função mostrarPares() e uma função somarPares(), ambas devem ser criadas em uma biblioteca "pares.h".

O exemplo a seguir mostra a execução do programa, utilizá-lo como modelo de saída.

Exemplo:

```
==== VETOR ORIGINAL ====
 10   20    8    5   19   10   20    7    6   18    6   20   16    7   15   16
 16   10    9   20

==== VETOR ORDENADO ====
  5    6    6    7    7    8    9   10   10   10   15   16   16   16   18   19
 20   20   20   20

==== PARES ====
 5 =>  2   4  Soma =  6
 6 =>  2   4   6  Soma = 12
 7 =>  2   4   6  Soma = 12
 8 =>  2   4   6   8  Soma = 20
 9 =>  2   4   6   8  Soma = 20
10 =>  2   4   6   8  10  Soma = 30
15 =>  2   4   6   8  10  12  14  Soma = 56
16 =>  2   4   6   8  10  12  14  16  Soma = 72
18 =>  2   4   6   8  10  12  14  16  18  Soma = 90
19 =>  2   4   6   8  10  12  14  16  18  Soma = 90
20 =>  2   4   6   8  10  12  14  16  18  20  Soma = 110

Deseja repetir o programa (S ou N)? n
```

4) Na biblioteca "vetores.h" criar função:

a) Com o protótipo `void gerarVetorFloat0a1(float vetor[], int tam)` que gera vetor com valores float aleatórios entre 0 e 1, para isso use:

```
vetor[i] =  (float)(rand()) / RAND_MAX;
```

b) Com o protótipo `void gerarVetorFloat0a100(float vetor[], int tam)` que gera vetor com valores float aleatórios entre 0 e 100, para isso use:

```
vetor[i] = ((float)(rand()) / RAND_MAX)*100;
```

c) Com o protótipo `void mostrarVetorFloat(float vetor[], int tam)` que mostra os elementos de um vetor float.

d) Com o protótipo `float somarVetorFloat(float vetor[], int tam)` que soma os elementos de um vetor float.

Usando essas funções:

a) Gerar um vetor do tipo float com 10 elementos aleatórios entre 0 e 1, exibir o vetor e mostrar a soma de todos os elementos do vetor.

b) Gerar um vetor do tipo float com 50 elementos aleatórios entre 0 e 100 e exibir o vetor. Criar outros dois vetores numéricos do tipo float, em um deles armazenar os valores menores que 50 e em outro os maiores que 50. Mostrar os vetores gerados.

Exemplo:

```
==== VETOR 1 ====
 0.5     0.1     0.3     0.2     0.8     0.5     0.8     0.0     0.3     0.6

Soma: 4.08


==== VETOR 2 ====
50.1    10.2    25.9    19.8    78.0    45.3    83.0     3.7    30.8    61.6
44.0    87.3    91.1    13.8    88.7    49.4    18.6    84.6    32.3     3.4
44.9    14.1    99.9    87.4    65.4    36.0    81.1    12.4    57.8    34.9
18.5     9.3     4.1    92.8    33.4    64.3    45.1    37.3    41.1    85.2
45.5    60.2    59.9    87.8    97.0    19.5    65.8    43.2    38.3    92.4


==== VETOR 3 ====
10.2    25.9    19.8    45.3     3.7    30.8    44.0    13.8    49.4    18.6
32.3     3.4    44.9    14.1    36.0    12.4    34.9    18.5     9.3     4.1
33.4    45.1    37.3    41.1    45.5    19.5    43.2    38.3


==== VETOR 4 ====
50.1    78.0    83.0    61.6    87.3    91.1    88.7    84.6    99.9    87.4
65.4    81.1    57.8    92.8    64.3    85.2    60.2    59.9    87.8    97.0
65.8    92.4

Deseja repetir o programa (S ou N)?
```

5) Na biblioteca "vetores.h" criar função:

a) Com o protótipo `void gerarVetorCharMinuscula(char vetor[], int tam)` para gerar um vetor randômico de caracteres alfabéticos minúsculos. A instrução para gerar randomicamente caracteres alfabéticos minúsculos é:

```
vetor[i] = rand() % 26 + 97;
```

Explicando: A função rand() % 26 gera um número aleatório entre 0 e 25, que somado com 97 equivale a um valor entre 97 a 122, que se refere aos caracteres alfabéticos minúsculos da tabela ASCII.

b) Com o protótipo `void gerarVetorCharMaiuscula(char vetor[], int tam)` para gerar um vetor randômico de caracteres alfabéticos maiúsculos. A instrução para gerar randomicamente caracteres alfabéticos maiúsculos é:

```
vetor[i] = rand() % 26 + 65;
```

Explicando: A função rand() % 26 gera um número aleatório entre 0 e 25, que somado com 65 equivale a um valor entre 65 a 90, que se refere aos caracteres alfabéticos maiúsculos da tabela ASCII.

c) Com o protótipo `void mostrarVetorChar(char vetor[], int tam, int n)` para mostrar o vetor gerado em colunas com *n* caracteres por linha, separados por um espaço.

Usando essas funções:

a) Gerar randomicamente um vetor com 100 caracteres alfabéticos minúsculos.

b) Gerar randomicamente um vetor com 200 caracteres alfabéticos maiúsculos.

c) Mostrar ambos os vetores gerados em colunas com 10 caracteres por linha, sendo cada caractere separado por um espaço.

Exemplo:

```
Quantos caracteres por linha deseja mostrar? 10

=== VETOR DE MINUSCULAS ===
h x x d r k v l i x
u c y x g o q q j r
o v b b p t h l t j
o j d d e w j f x o
d d t x k s b h o i
d w n x f f x r n n
q o k p c v t s u g
g m i v w t k t o j
d l q p i q m f t b
i q u s v i b q j b

=== VETOR DE MAIUSCULAS ===
H X X D R K V L I X
U C Y X G O Q Q J R
O V B B P T H L T J
O J D D E W J F X O
D D T X K S B H O I
D W N X F F X R N N
Q O K P C V T S U G
G M I V W T K T O J
D L Q P I Q M F T B
I Q U S V I B Q J B
T W J N J Y H Q U N
U H W Q F E G F A V
J B S R N U O U C A
R X K T K Q G P U W
G H J A V M E J S V
F M W C P J U K U F
F Y C G L A P F H O
Q J D S U U A C Y B
G N Q U U C J T N F
F U K Q X K B N G L

Deseja repetir o programa (S ou N)?
```

6) Na biblioteca "vetores.h" criar uma função com o protótipo `void gerarVetorCharAlfanumerico(char vetor[], int tam)` que gera um vetor randômico de caracteres alfanuméricos (números, letras e símbolos especiais, exceto caracteres de controle). A instrução para gerar randomicamente caracteres alfanuméricos é:

```
vetor[i] = rand() % 223 + 33;
```

Explicando: A função rand() % 223 gera um número aleatório entre 0 e 222, que somado com 33 equivale a um valor entre 33 a 255, que equivale aos caracteres alfanuméricos da tabela ASCII.

Usando essa função:

a) Gerar randomicamente um vetor com 200 caracteres alfanuméricos (números, letras e símbolos especiais, exceto caracteres de controle).

b) Mostrar o vetor gerado em colunas com 12 caracteres por linha separados por um espaço.

c) Percorrer o vetor e contar quantos caracteres são alfabéticos maiúsculos, quantos são alfabéticos minúsculos e quantos são números. Armazenar essas quantidades em um vetor. Esse vetor possui tamanho 3 e cada índice armazena a quantidade de um desses tipos. É indispensável ir armazenando as quantidades

à medida que o vetor é percorrido `(vet[0]=vet[0]+1)`, portanto, é necessário inicializar com zero o vetor das quantidades antes de utilizá-lo.

d) Mostrar o vetor com as quantidades.

Exemplo:

```
Quantos caracteres por linha deseja mostrar? 12

=== VETOR ALFANUMERICO ===
= ! # T ╣ a 2 ? ▓ D Ø 0
f ( ▒ ç I ó   ~ ╢ C 3 $
h ã ₧ ╝   , ì : M M ú û f
E X @ } # 5 ┌ í f ─ s >
Å ^ ┌ ┴ _ ø ~ Þ h º 8 ¨
Z å T $ k ö ' ÿ a 9 A z
ä ■ 3 } = à └ J 6 s á <
á ╬ º F î ╢ U á X z G &
µ ¡ ┘ ø Ë ä í ¤ Ï õ ╝ δ
╝ ª á ₧ δ 8 . Ø § ë » ╠
ª K ; Î é $ Ø , { ×  V T
% È c = & . ¤ ò ¦ M ñ ¡
┌ = * H δ · ┘ P ─ ¡ ² 0
ó Þ ø k í T Ø c K ▓ V X
R ¼ ê ù ] L ų 0 Ü , k ö
├ ã ' á K ·   » [ f ╚ ò
ò $ 1 ò 2 < ¥ ┬

Caracteres alfabeticos maiusculos: 30
Caracteres alfabeticos minusculos: 19
Caracteres numericos: 10


Deseja repetir o programa (S ou N)? n
```

7) Na biblioteca "vetores.h" criar uma função com o protótipo `void gerarVetorPositivoNegativo(int vetor[], int tam, int limNegativo, int limPositivo)` para gerar um vetor com valores positivos e negativos em um intervalo.

A instrução para gerar valores inteiros positivos e negativos em um intervalo é:

```
vetor[i] = rand() % (limPositivo + limNegativo + 1) - limNegativo;
```

Usando essa função:

Gerar um vetor A de inteiros com 100 elementos, com valores entre -50 e +50. Em seguida, armazenar no vetor B somente os valores positivos do vetor A.

Exemplo:

```
==== VETOR A ====
-15  -22   50    9    3   40  -27   35   19  -42   48   38    9  -21  -43  -24
 -2   14  -13  -44  -39   16  -33   -9  -31  -48  -26  -48   46   40   -8   18
-11   33  -34   36  -35    1   50  -50   38  -24   48   -1   -8  -48   -4  -23
 -7   -8   39   47   34   -5  -20  -23  -48   -3  -34   -8  -15   17    9  -47
 39   17   35   23   28  -49    0  -25   48   13   39  -49   -8   -6  -14  -50
 -5  -29  -23  -30   20   34  -39  -18   31  -28  -12  -38  -32  -20   32  -24
-17   -8   42   26

==== VETOR B ====
50   9   3  40  35  19  48  38   9  14  16  46  40  18  33  36   1  50  38  48
39  47  34  17   9  39  17  35  23  28  48  13  39  20  34  31  32  42  26

Deseja repetir o programa (S ou N)?
```

8) Gerar um vetor A de inteiros com 100 elementos aleatórios entre 0 e 10. Criar um vetor B float que conterá os valores do vetor A divididos pelo maior valor contido no vetor A. Mostrar os dois vetores.

Exemplo:

```
==== VETOR A ====
  6    2    8    9    7    4    6    6   10    6    3    1    5    0    5    6
  8    0    5    3    5    4    4    9    9    3    9    1    1    3    6    6
  8    0   10    8    7    8   10    9    6    4    7    7    9    0    0    7
  8   10    0    7    3    5    4    9    0    7    8    7    2    5    1    0
  0    9    8    4    8    6    6    0    4    8    6    9    6    7    2    1
  9    2    8    2    6    1    9    4    8    4    4    4    1    1    6    6
  0    6    3    2
Maior valor: 10

==== VETOR B ====
0.6  0.2  0.8  0.9  0.7  0.4  0.6  0.6  1.0  0.6  0.3  0.1  0.5  0.0  0.5  0.6
0.8  0.0  0.5  0.3  0.5  0.4  0.4  0.9  0.9  0.3  0.9  0.1  0.1  0.3  0.6  0.6
0.8  0.0  1.0  0.8  0.7  0.8  1.0  0.9  0.6  0.4  0.7  0.7  0.9  0.0  0.0  0.7
0.8  1.0  0.0  0.7  0.3  0.5  0.4  0.9  0.0  0.7  0.8  0.7  0.2  0.5  0.1  0.0
0.0  0.9  0.8  0.4  0.8  0.6  0.6  0.0  0.4  0.8  0.6  0.9  0.6  0.7  0.2  0.1
0.9  0.2  0.8  0.2  0.6  0.1  0.9  0.4  0.8  0.4  0.4  0.4  0.1  0.1  0.6  0.6
0.0  0.6  0.3  0.2

Deseja repetir o programa (S ou N)?
```

9) Uma locadora de vídeos armazena em um vetor A de 300 posições a quantidade de filmes retirados por seus clientes durante o ano. A locadora está fazendo uma promoção e para cada 10 filmes retirados, o cliente tem direito a uma locação grátis. Faça um programa que crie um vetor B contendo a quantidade de locações gratuitas a que cada cliente tem direito. Declare ambos os vetores com valores inteiros.

Exemplo:

```
==== VETOR A ====
 30   38   48    3   89   86    6   91    0   39   68   10   87   35   83   88
 62   85    7   90    4   43   38   16   35   45   44    5   23   44   77   78
 38   74   79   30   13   10   22   65    4   36   65   54   56   11    5   92
 87   13   59   15   87   42   23   91   66   17   54   13   77    7   91   97
  0    6   24   11   95   86    3   69   74   14   44   61   26   52    2   14
 24   85   58   59   44   57   51   77   46   10   36   91   35   45   76   44
  4   73  100   15   75   86   55   77   99   55   16   84   78   88   20   81
 94   92   79   23    4   28   20   18   50   86   53   45   29   76   88   23
 31   66   22   66   53   18    7   19    9    7   53   40   91   26   33   17
 12   97   59   79   86   87   81   93   46   26   16   22    5    8   37   84
 74   74   35   37   79   96    4   30   46   93   50    1   52   36    5   54
  5   74   13   90   21   12   60   90   17   44   18    2   88   14   64   90
  6   71   72   38   13   64   48   15   86   43   51   61   49   20   80   33
 97   38   38   55   92   32    1   15   69   63   42   35   15   86   64   65
 28   72   13   46   56   73   12   68   77   30   78   77   15   28   18   58
 60    3   45  100   23   64   95   79    0   76   62   16   75   31   46   90
 63   16   21   89   18   65   81    9   46   82   58   74   34   72   23   87
 25   66   58   82   46   29    9   88   24    1   46   26   99   51   92   92
 15   57   50   58    3   29   43   51   46   50   87   41

==== VETOR B ====
 3   3   4   0   8   8   0   9   0   3   6   1   8   3   8   8   6   8   0   9
 0   4   3   1   3   4   4   0   2   4   7   7   3   7   7   3   1   1   2   6
 0   3   6   5   5   1   0   9   8   1   5   1   8   4   2   9   6   1   5   1
 7   0   9   9   0   0   2   1   9   8   0   6   7   1   4   6   2   5   0   1
 2   8   5   5   4   5   5   7   4   1   3   9   3   4   7   4   0   7  10   1
 7   8   5   7   9   5   1   8   7   8   2   8   9   9   7   2   0   2   2   1
 5   8   5   4   2   7   8   2   3   6   2   6   5   1   0   1   0   0   5   4
 9   2   3   1   1   9   5   7   8   8   8   9   4   2   1   2   0   0   3   8
 7   7   3   3   7   9   0   3   4   9   5   0   5   3   0   5   0   7   1   9
 2   1   6   9   1   4   1   0   8   1   6   9   0   7   7   3   1   6   4   1
 8   4   5   6   4   2   8   3   9   3   3   5   9   3   0   1   6   6   4   3
 1   8   6   6   2   7   1   4   5   7   1   6   7   3   7   7   1   2   1   5
 6   0   4  10   2   6   9   7   0   7   6   1   7   3   4   9   6   1   2   8
 1   6   8   0   4   8   5   7   3   7   2   8   2   6   5   8   4   2   0   8
 2   0   4   2   9   5   9   9   1   5   5   5   0   2   4   5   4   5   8   4

Deseja repetir o programa (S ou N)?
```