Lista 2 – vetores (extraclasse entregar)

Assunto:

Vetores numéricos.

Lembretes:

Vetores precisam ter tamanho definido quando são declarados.

Cuidado para não ultrapassar o tamanho do vetor, ou seja, percorrer índices (ler) ou armazenar valores além do tamanho definido para o vetor.

Indispensável:

Listar todos os exercícios como opções de um menu. Usar as funções gerar e mostrar vetor e criar funções no mesmo arquivo que está a main() para "chamar" as funções que representam cada um dos exercícios.

1) Gerar aleatoriamente 10 elementos para um vetor. Mostrar esses elementos e informar se cada um deles é par ou ímpar. Fazer a média dos pares e somar os ímpares.

2) Gerar aleatoriamente 10 elementos para um vetor menores que 50. Criar outros dois vetores, um deles armazenando os valores divisíveis por 2, por 3 ou por 5 e em outro os outros valores. Fazer a média dos valores divisíveis por 2, por 3 ou por 5. Mostrar os dois vetores criados e a média obtida.

3) Gerar um vetor com 100 elementos entre 0 e 400, inclusive, mostrar o vetor gerado. Em seguida percorrer o vetor e contar quantos elementos estão entre 00 e 100, quantos estão entre 101 e 200, quantos estão entre 201 e 300 e quantos estão entre 301 e 400. A quantidade será armazenada em outro vetor com tamanho 4, para cada uma das respectivas quantidades. Por exemplo, VetQuantidades[0] conterá a quantidade de valores entre 10 e 100 no vetor de origem.
Obs.: é **indispensável** que os valores sejam contados à medida que o vetor é percorrido, assim: VetQuantidade[0] = VetQuantidade[0] + 1.

4) Gerar um vetor de números aleatórios com o tamanho e a faixa de valores informado pelo usuário. Mostrar o vetor gerado, contar a quantidade de números múltiplos de 5 e informar.
Exemplo:
Tamanho do vetor: 20
Faixa

Limite inferior 30

Limite superior 50

Gerar um vetor de 20 elementos aleatórios entre 30 e 50

5) Gerar um vetor aleatório com 10 elementos entre 0 e 100. Verificar se cada um dos elementos armazenados no vetor é primo. Mostrar da seguinte forma:
Vet[0] = 10 ➔ não é primo;
Vet[1] = 7 ➔ é primo;
Vet[2] = 30 ➔ não é primo;
Utilizar, **obrigatoriamente**, uma função para verificar se o número é primo. Essa função pode retornar 0 para indicar que o número e primo e 1 para indicar que não é primo. Tratar esse retorno na função chamadora.

6) O que faz o algoritmo a seguir:

```
declare vetA[30], vetB[30], i, j como inteiro

repetir i = 0, até i < 30, passo 1
     leia vetA[i]
fim-repetir

j=0;
repetir i = 0, até i < 30, passo 1
     se ( vetA[i] % 2 == 0 ) then
          vetB[j] ← vetA[i]
          j++;
     fim-se
fim-repetir

repetir i = 0, até i < j, passo 1
     escreva vetB[j]
fim-repetir
```

Implemente uma solução na linguagem C para esse algoritmo. A entrada de dados deve ser gerada por um função e a apresentação do conteúdo do vetor também.