Lista de exercícios
LP1-ADS

Prof. Luciano Bernardes de Paula
Profa. Talita de Paula Cypriano de Souza

(Lista adaptada do livro "Treinamento em linguagem C" – Victorine Viviane Mizrahi, ec. Pearson).

Resolva os exercícios abaixo, utilizando funções.

VETORES

1 – Escreva um programa que receba dez valores diferentes do tipo float e os apresente na tela, primeiro na ordem em que foram inseridos e depois na ordem contrária de inserção. Utilize vetor.

2 – Escreva um programa que receba 7 valores e apresenta a média, o maior valor inserido e o menor valor inserido. O seu programa deve possuir somente um laço *for*.

3 - Escreva um programa que receba dez valores diferentes do tipo inteiro, em qualquer ordem, e os armazene em um vetor. Depois, todos os valores são apresentados na tela. Após isso, o usuário entra com um valor 0 ou 1. Se o usuário digitar 0, o programa apresenta o maior elemento encontrado no vetor. Caso o usuário escolha 1, o programa apresenta o menor elemento encontrado no vetor.

4 – Faça um programa que receba e armazene em um vetor dezesseis valores do tipo **float**. Depois, o programa deve trocar os oito primeiros valores pelos oito últimos e vice-e-versa. Escreva ao final o vetor resultante.

5 – Escreva um programa que receba dez valores diferentes do tipo inteiro, em qualquer ordem, e os armazene em um vetor. Depois o usuário entra com um valor 0 ou 1. Se o usuário digitar 0, o programa ordena o vetor na ordem crescente e o apresenta na tela. Caso o usuário escolha 1, o programa ordena o vetor na ordem decrescente e apresenta na tela.

6 - Faça um programa que leia 10 valores inteiros armazenando-os em um vetor chamado teste1. Utilizando outro vetor (teste2) de 10 posições, preencha-o utilizando a seguinte regra: se o valor do índice de teste2 for par, o valor do elemento deve ser igual ao elemento equivalente de teste1 multiplicado por 5; se for ímpar, deverá ser somado com 5. Ao final, mostrar o conteúdo dos dois vetores.

7 - Faça um programa que leia um vetor de inteiros e "remova" os elementos contendo o valor 0 (transfira-os para o final do vetor). Mostre o vetor resultante na tela.
Ex.: O vetor 0 1 3 -1 0 0 5 ficaria: 1 3 -1 5 0 0 0.

8 - Escreva um programa que defina um valor qualquer e o usuário deve descobri-lo. Se o usuário entrar com um valor maior ou menor que o valor procurado, o programa deve

informá-lo, indicando se o valor procurado é maior ou menor que o valor inserido. O usuário deve ter no máximo 10 tentativas e, caso ele repita um dos números já inseridos, o programa deve avisar o usuário que aquele número já foi inserido e permitir a entrada de um novo valor sem contar essa tentativa.

9 – Faça uma nova versão do programa do exercício anterior, mas agora use a função *rand()* para que o valor a ser descoberto seja alterado a cada execução do programa.

Dicas:
- para usar a função *rand( )*,é preciso incluir a biblioteca ***stdlib.h***.

- para usar a função *rand( )* use: **variável = rand();**

- para que sejam gerados números entre 0 e n, é possível calcular o resultado do resto da divisão do resultado da função *rand()* por n+1. Exemplo:

num = rand() % 101; // gera valores entre 0 e 100

- toda vez que a função rand() é executada, ela gera um número "aleatório" de acordo com um valor chamado semente (*seed*), que é próprio da função. Para cada programa, a semente será diferente. Para inicializar a semente para cada execução de programa, utilize a função *srand( )*. Essa função recebe um valor que será usado como semente. Uma forma de gerar sementes diferentes para cada execução é passar o horário do computador para que seja usado como semente. Cada vez que for executado, a probabilidade de ser utilizada uma semente diferente é grande. Para passar o horário do computador como semente, utilize:

srand(time(NULL)): // use esse comando antes de usar a função rand( ) no seu programa

Para utilizar a função ***time()***, é preciso incluir a biblioteca ***time.h***.

10 – Faça um programa que receba do usuário 7 caracteres, armazene-os em um vetor. Após isso o vetor é colocado em ordem alfabética e apresentado na tela.

11 – Faça um programa que leia um vetor de 10 caracteres, e diga quantas consoantes foram lidas. Imprima as consoantes.

12 – Faça um programa que use um vetor de 10 posições de caracteres e preencha-os com entradas feitas pelo usuário. Após isso, o usuário deve entrar com um novo caracter e o programa deve dizer se esse caracter existe no vetor e, se existir, qual sua posição nele. Após 3 consultas do usuário, o programa deve terminar.

13 – Repita o exercício anterior, mas o vetor deve ser preenchido de forma aleatória.

MATRIZES

1 – Escreva um programa que receba nove valores e os apresente na tela. Use uma matriz 3 x 3.

2 - Faça um programa que utilize uma matriz TAM x TAM e preencha-a com zeros e em seguida preencha as duas diagonais com o valor 1. Após isso, imprima a matriz resultante. Utilize TAM como uma constante e teste o programa com TAM = 10, 20, 25 e 30.

3 - Faça um programa que utilize uma matriz TAM x TAM e preencha-a com zeros e em seguida preencha somente as linhas e colunas que estiverem na "borda" com o valor 1. Após isso, imprima a matriz resultante. Utilize TAM como uma constante e teste o programa com TAM = 10, 20, 25 e 30.

4 - Faça um programa que utilize uma matriz TAM x TAM e preencha-a valores aleatórios entre 0 e 500. Imprima a matriz resultante e em seguida são apresentadas duas opções para o usuário: a primeira opção faz com que seja apresentado o menor valor presente na matriz e a segunda opção o maior. Utilize TAM como uma constante e teste o programa com TAM = 10, 20, 25 e 30.
Dicas:
- para gerar números aleatórios, use a função *rand( )*. Para usá-la é preciso incluir a biblioteca ***stdlib.h***.
- para usar a função *rand( )* use:

variável = rand();

- para que sejam gerados números entre 0 e n, é possível calcular o resultado do resto da divisão do resultado da função *rand()* por n+1. Exemplo:
a = rand() % 101; // gera valores entre 0 e 100
- toda vez que a função rand() é executada, ela gera um número "aleatório" de acordo com um valor chamado semente, que é próprio da função. Para cada programa, a semente será uma. Para inicializar a semente para cada execução de programa, utilize a função *srand( )*. Essa função recebe um valor que será usado como semente. Uma forma de gerar sementes diferentes para cada execução é passar o horário do computador para que seja usado como semente. Cada vez que for executado, a probabilidade de ser utilizada uma semente diferente é grande. Para passar o horário do computador como semente, utilize:

srand(time(NULL)): // use esse comando antes de usar a função rand( ) no seu programa

Para utilizar a função ***time()***, é preciso incluir a biblioteca ***time.h***.

5 - Faça um programa que utilize uma matriz TAM x TAM e preencha-a valores aleatórios entre 0 e 500. Em seguida o programa deve ordenar a matriz em ordem crescente (da esquerda para direita, de cima para baixo) e apresentá-la na tela. Utilize TAM como uma constante e teste o programa com TAM = 10, 20, 25 e 30.