PROVA DE PROGRAMAÇÃO DE COMPUTADORES I

SEGUNDA UNIDADE (PP2)

CONTEÚDO: VETORES, MATRIZES E FUNÇÕES

Sistemas de Informação Ano 2019 – UEMS

Professora: Mercedes Gonzales Márquez

POLINÔMIOS

1. Faça uma função valor_poli que calcule o valor $p(x_0)$ do polinômio $p(x)=a_0+a_1x+...+a_nx^n$ em um ponto x_0. São dados: o valor n do grau do polinômio, o valor x_o e os coeficientes reais do polinômio $a_0, a_1, ..., a_n$.

2. Faça uma função deriv_poli que determina os coeficientes reais da primeira derivada de um polinômio $p(x)=a_0+a_1x+...+a_nx^n$. São dados: o valor n do grau do polinômio e dos seus coeficientes reais $a_0, a_1, ..., a_n$.

3. Seja um polinômio $p(x)=a_0+a_1x+...+a_nx^n$ de grau n>=2. Uma possível maneira de calcular uma raiz do polinômio é pelo Método de Newton. Este método consiste em se fornecer uma aproximação inicial para a raiz, isto é, um valor que não é a raiz exata, mas é um valor próximo. Assim, se x_0 é esta aproximação inicial, p(x0) não é zero, mas espera-se que seja próximo de zero. A obtenção da raiz pelo método de Newton é feita pelo refinamento desta
solução inicial, isto é, pela tentativa de minimizar o erro cometido. Isto é feito pela expressão seguinte:

$$x_{n+1} = x_n - \frac{\mathrm{p}(x_n)}{\mathrm{p}'(x_n)}$$

n = 0, 1, 2,..., e onde p'(x) é a primeira derivada de p(x). Usualmente, repete-se este refinamento até que $|x_{n+1-}x_n| < \varepsilon$, $\varepsilon > 0$, ou até que m iterações tenham sido executadas.
Construa um programa em que receba como dados de entrada um polinômio $p(x)=a_0+a_1x+...+a_nx^n$ e uma aproximação inicial x_0 da raiz de p(x), $\varepsilon > 0$ e o número máximo de iterações, e calcule uma aproximação da raiz de p(x) pelo método de Newton. Utilize obrigatoriamente as funções valor_poli e deriv_poli, solicitadas nas questões anteriores.

MATRIZES

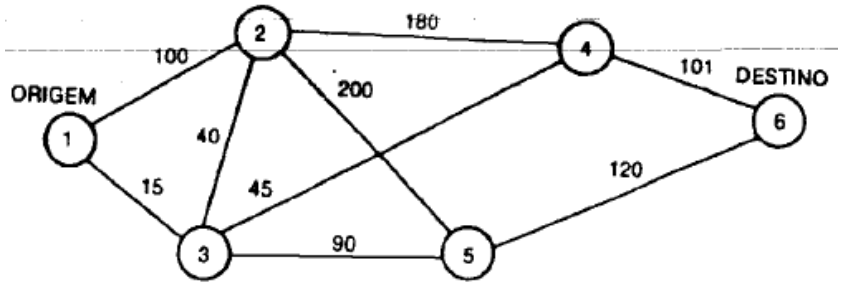
4. Escreva um programa que leia uma matriz quadrada nxn e verifique se ela e uma matriz triangular inferior, triangular superior ou diagonal.
5. Escrever um programa que calcule e imprima as n raízes do seguinte sistema particular de n equações com n incógnitas:

$$
\begin{aligned}
a_{11}x_1 &= b_1 \\
a_{21}x_1 + a_{22}x_2 &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \\
&\ldots\ldots\ldots\ldots \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots + a_{nn}x_n &= b_n
\end{aligned}
$$

Para isto, ler:

- número de equações, *n<=30;*
- a matriz triangular A dos coeficientes;
- o vetor b dos termos independentes.

6. Os nós do grafo a seguir representar cidades e os arcos, a presença de uma estrada ligando duas destas cidades. Os números ao lado dos arcos são a distância medida em quilómetros.



Pode-se representar numericamente este grafo por uma variável composta bidimensional, na qual a distância entre duas cidades i, j é indicada pelo elemento D[i.j]; sei = j ou se não houver conexão entre i e j, D[i,j] será zero. Desta forma tem-se:

	1	2	3	4	5	6
1	0	100	15	0	0	0
2	100	0	40	180	200	0
3	15	40	0	45	90	0
4	0	180	45	0	0	101
5	0	200	90	0	0	120
6	0	0	0	101	120	0

Matriz D de distância
entre as cidades

Seja proposto agora o problema de se achar o caminho mais curto entre duas cidades quaisquer. Este problema foi resolvido por Dijkstra e tem uma série de aplicações em questões de otimização. Além da matriz D das distâncias, considera-se a variável composta unidimensional DA cuja componente DA[I] representa a distância acumulada em um caminho percorrido desde a origem até a cidade I. Cada uma destas componentes será iniciada com um valor bem grande,

por exemplo 10000. Ainda serão consideradas mais duas variáveis compostas unidimensionais. A primeira designada ANT será tal que a sua componente ANT[I] indica qual é a cidade antecedente de I no caminho considerado. A outra, EXP terá componentes lógicas, todas elas inicialmente com o valor falso indicando que as cidades ainda não foram "expandidas".
Partindo de uma cidade C, inicialmente igual à origem, calcula-se a nova distância acumulada (NOVADA) de cada uma das cidades adjacentes a C ainda não expandidas. A nova distância acumulada prevalecerá sobre o valor anterior se lhe for inferior: neste caso, C será atribuído à componente ANT[I]. Quando terminar a expansão de C, registra-se que EXPA[C] é verdadeiro.
Em seguida, procura-se dentre as cidades ainda não expandidas aquela que tenha a menor distância acumulada. Esta será a nova cidade C e a sua distância acumulada é então a menor que possa ser conseguida a partir da ORIGEM.
O processo será repetido até que a cidade C seja o DESTINO ou que não se encontre nenhuma cidade ainda não expandida, cuja distância acumulada seja inferior a 10000. Neste último caso isso significa que não existe caminho ligando a ORIGEM ao DESTINO.

O algoritmo encontra-se nas páginas 125 e 126 do livro Algoritmos Estruturados de Harry Farrer. Consulte-o e faça sua implementação.

SOMATÓRIOS

7. Escreva a função pi que calcule o valor do número π através da série:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Deverá ser fornecido à função o número de termos da série para o cálculo de π.

Faça um programa que, fornecendo à função, sucessivamente, o número de termos (1,2,3,...,N), escreva uma tabela com o valor de π e o número de termos utilizados. O valor de N deve ser lido.
(Exercício resolvido na prova da Unidade I, mas agora com uma modificação).

8. Alternativamente à função pi do exercício anterior, escreva a função pi2 que calcule o valor do número π através da série:

$$S = 1 - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \cdots, \qquad \text{sendo } \pi = \sqrt[3]{32xS}$$

Deverá ser fornecido à função o número de termos da série para o cálculo de π. Faça um programa que, fornecendo às funções pi e pi2, sucessivamente, o número de termos (1,2,3,...,N), escreva uma tabela com os valores de π obtidos por ambas funções e o número de termos utilizados. O valor de N deve ser lido.

9. Faça uma função cosseno que dados *x* real e *n* natural, calcule uma aproximação para cos x através dos *n* primeiros termos da seguinte série:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^k \frac{x^{2k}}{(2k)!} + \ldots$$

(Exercício resolvido na prova da Unidade I, mas agora com uma modificação).

10. Faça uma função seno que dados *x* real e *n* natural, calcule uma aproximação para sen x através dos *n* primeiros termos da seguinte série:

$$sen\ x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^k \frac{x^{2k+1}}{(2k+1)!} + \ldots$$

(Exercício resolvido na prova da Unidade I, mas agora com uma modificação).

11. Faça um programa que usando a função cosseno e seno das questões anteriores, construa uma tabela de valores da tangente de x, para um x real e vários valores para n=1,2,...,N, onde N é lido. Do lado dos valores da tangente obtida, coloque os valores da tangente obtida pela função tan da biblioteca math.h e calcule o erro entre ambos valores.

GEOMETRIA COMPUTACIONAL

12. Fazer um programa que calcule o volume de uma esfera em função do raio R. O raio deverá variar de 0 a 20 cm de 0,5 em 0,5 cm. Use a função pi e pi2.

$$V = \frac{4}{3}\pi R^3.$$

(Exercício resolvido na prova da Unidade I, mas agora com uma modificação).

13. Para um polígono regular inscrito numa circunferência, quanto maior o número de lados do polígono, mais seu perímetro se aproxima do comprimento da circunferência. Se o número de lados for muito grande e o raio da circunferência for unitário, o semiperímetro do polígono terá um valor muito próximo de π. Fazer um programa que escreva uma tabela do semiperímetro em função do número de lados, para polígonos regulares inscritos, numa circunferência de raio unitário. O número de lados deverá variar de 5 a 100 de 5 em 5. Use a função pi e pi2 e também a função sen criada na questão anterior.
(Exercício resolvido na prova da Unidade I, mas agora com uma modificação).