Tópico 11: arranjos multidimensionais

Auto Avaliação 11.1 - Soluções

Utilizando como referência o material da **Apresentação 11.1**, desenvolva os seguintes programas.

1. Ler uma matriz 4 x 3 de inteiros. Calcula e apresenta os somatórios dos valores da primeira linha da matriz e da última coluna. Apresenta a matriz.

Programa C:

```
//Inserir cabecalho …
#include <stdio.h>
#include <stdlib.h>
#define MAXLINHAS 4
#define MAXCOLUNAS 3
int main ( )
{
  int mat_int [MAXLINHAS] [MAXCOLUNAS];
  int i , j , soma;
  //leitura da matriz
  for (i=0;i<MAXLINHAS; i++)
     for (j=0;j <MAXCOLUNAS;j++)
        {
           printf("\nmat_int [%d , %d] = " , i, j);
           scanf("%d", &mat_int [i] [j]);
        }
  //somatorio dos elementos da primeira linha
  soma = 0;
  for (i=0; i < MAXCOLUNAS; i++)
        soma = soma + mat_int [0] [i];
  printf ("\nSomatorio dos elementos da primeira linha %d \n", soma);
 //somatorio dos elementos da última coluna
  soma = 0;
  for (i=0; i < MAXLINHAS; i++)
        soma = soma + mat_int [i] [MAXCOLUNAS - 1];
  printf ("\nSomatorio dos elementos da ultima coluna %d \n", soma);
  //impressao da matriz em formato matricial
  printf("\nMatriz em formato matricial\n" );
  for (i=0; i < MAXLINHAS; i++)
  {
      printf("\n");
```

```
        for (j = 0;j < MAXCOLUNAS;j++)
              printf("%d  ", mat_int [i] [j]);
        printf("\n");
      }
   system("PAUSE");
   return(0);
  } //fim de main
```

2. Fazer um programa C que localiza o elemento **Minimax** de uma matriz quadrada 4 x 4 de inteiros, i.e., o menor elemento da linha que contem o maior elemento da matriz.

Programa C:

```
//Inserir cabecalho …
#include <stdio.h>
#include <stdlib.h>
#define TAM 4
int main (  )
{
  int mat [TAM] [TAM];
  int l,c;   // l=linha , c=coluna
  int maior, menor, linmax, colmax;
  // Leitura da Matriz
  for (l=0; l < TAM; l++)
      for (c=0; c < TAM; c++)
         {
            printf("\nmat [%d , %d] = " , l, c);
            scanf("%d", &mat [l] [c]);
         }
 // Escrita da Matriz em formato matricial
  printf ("\tMatriz em formato matricial\n" );
  printf ("\t==========================\n");
  for (l=0; l < TAM; l++)
  {
       printf("\n");
       for (c = 0; c < TAM; c++)
            printf("%d  ", mat [l] [c]);
       printf("\n");
  }
  // Localizacao do maior elemento
  maior = mat [1] [1];
  linmax = 1;
  colmax = 1;
  for (l = 0; l < TAM; l++)
    for (c = 0; c < TAM; c++)
          if ( maior < mat [l] [c] )
```

```
            {
                maior = mat [l] [c];
                linmax = l;
                colmax = c;
              }
    // Localização do Minimax
    menor = maior;
    for (c = 0; c < TAM; c++)
          if ( mat [linmax] [c] < menor )
          {
               menor = mat [linmax] [c];
               colmax = c;
           }
    // Impressao do Minimax
    printf ("\n\nElemento Minimax = %d", mat [linmax] [colmax]);
    printf ("\n\nLinha: %d; Coluna: %d", linmax, colmax);
    system("PAUSE");
    return(0);
  } //fim de main
```

3. Fazer um programa em C, com L linhas, L<=20, que a partir de 2 vetores A e B gere uma matriz AB tal que a 1ª. coluna de AB seja formado pelos valores de A e a segunda coluna de AB seja formada pelos elementos de B. O vetor A é gerado randomicamente com valores entre 0 e 12. O vetor B é formado pelos fatoriais dos valores de A, respectivamente. Mostrar o vetor gerado AB.

Programa C:

```
//Inserir cabecalho …
#include <stdio.h>
#include <stdlib.h>
#define TAM 20
int main (  )
{
  int l,c,n,j;   // l=linha , c=coluna , n= numero de elementos para os vetores
  float ab[TAM][2];  //
  int a[TAM];
  float b[TAM];
   float fat;
   n=1;
  while ( (n<=1 ) || (n>20))
  {
    printf("Digite o numero de elementos da matriz A (máximo 20, mínimo 1)\n");
    scanf("%d",&n);
  }
   for (l=0; l<n; l++) // geração dos elementos de A
        a[l]= rand()% 13; // A terá valores no intervalo [0,12]
```

```
    // geração dos elementos de B
    for (l=0; l<n; l++) // para cada valor de A
    {
       fat=1;    // calcular o fatorial de a[l]
       for (j=1; j<= a[l]; j++)
             fat=fat * j;
       b[l] = fat;   // e armazenar o fatorial em B[l]
     }
    printf("\n\n");

    // Gera e Mostra a matriz ab
    printf("AB [ 1 ]      AB [ 2 ] \n");
    for (l=0; l<n; l++)
      {
          ab [l] [0] = a[l];
          ab [l] [1] = b[l];
          printf("  %6.0f     %9.0f \n\n", ab [l] [0], ab [l] [1]);
       }
 system("pause");
 return 0;
} //fim de main
```

4. Uma matriz quadrada é dita triangular se os elementos situados acima de sua diagonal principal são todos nulos. Escreva um programa C que receba uma matriz quadrada com ordem de no máximo 30, através de uma variável e determine se ela é triangular ou não.

Programa C:

```
//Incluir cabecalho …
#include <stdio.h>
#include <stdlib.h>
#define TAM 30
int main (  )
{
  int triang,l,c,n;   // l=linha , c=coluna    n= ordem da matriz
  int matriz[TAM][TAM]; //
  n=1;
  while ( (n<=1 ) || (n>30))
  {
    printf(" Digite a ordem da matriz (máximo 30, mínimo 2)\n");
    scanf("%d",&n);
   }
  printf(" Digite os elementos da matriz, linha por coluna\n");
  for (l = 0; l < n; l++)
     for (c=0; c < n; c++)
     {
```

```
            printf("\n Valor[%d] [%d] = ",l,c);
            scanf("%d",&matriz [l] [c]);
        }
    printf("          A matriz lida \n\n");
    for (l = 0; l < n; l++)
    {
         for (c=0; c < n; c++)
               printf("%d ",matriz [l] [c]);
         printf("\n");
    }
    // Determinação se é triangular: basta percorrer os elementos acima da
    // diagonal principal e verificar se existe 1 valor diferente de zero.
    // Restringir a pesquisa acima da diagonal principal, isto é, para cada linha l,
    // inicia-se na coluna l+1
    triang = 0;
    for (l=0; l<n; l++)
        for ( c=l+1; c< n; c++)
              if (matriz [l] [c]  != 0)
                  triang = 1;
    if (triang)
        printf("\n A matriz dada não eh triangular \n");
    else
        printf(" \n A matriz dada  eh triangular\n");
    system("pause");
    return 0;
} //fim de main
```

5. Uma matriz ***esparsa*** é uma matriz que tem aproximadamente 2/3 de seus elementos iguais a zero. Fazer um programa que lê (linha a linha) uma matriz ***esparsa*** **matesp [10] [10]**, contendo valores inteiros, e forma uma matriz ***condensada*** **matcon**, de apenas três colunas, contendo os elementos não nulos de **matesp**, de forma que:

    a. A **primeira coluna** contenha um valor não nulo de **matesp**;

    b. A **segunda coluna** contenha a linha de **matesp** onde foi encontrado o valor armazenado na coluna **1** e;

    c. A **terceira coluna** contenha a coluna de **matesp** onde foi encontrado o valor armazenado na coluna **1**.

- Imprimir as duas matrizes, APÓS o preenchimento da matriz ***condensada***.

- Como determinar o número de linhas de **matcon**?

    - dim*dim/3 linhas (+- 1/3 de dim*dim) ou simplesmente fazer o número de linhas = dim. Neste caso, a matriz não seria tão esparsa assim ;-)

Programa C:

```
//Incluir cabecalho …
#include <stdio.h>
#include <stdlib.h>
#define TAM 10
int main (  )
{
  int matesp [TAM] [TAM];
  int matcon [TAM] [3];
  int lin, col; //indices da matriz esparsa
  int lincon; //índice da linha da matriz condensada
// Leitura da Matriz Esparsa
. . .
// Impressão da matriz esparsa lida
. . .
// Preenchimento e impressao da matriz condensada
  printf ("\n\n\tMatriz Condensada");
  printf ("\n\t================\n\n");
  lincon = 0; // inicializa linha da matriz condensada
  for (lin = 0; lin < TAM; lin++)
    for (col = 0; col < TAM; col++)
        if (matesp [lin] [col] != 0)
        {
              matcon [lincon] [0] = matesp [lin] [col]; //copia o valor da esparsa
              matcon [lincon] [1] = lin; //copia a linha onde estava na esparsa
              matcon [lincon] [2] = col; // copia a coluna onde estava na esparsa
              printf ("%2d\t%2d\t%d\n",
              matcon[lincon][0], matcon[lincon][1],
              matcon[lincon][2]); // imprime linha da condensada
              lincon++; // posiciona nova linha da matriz condensada
         }
} // Fim de main
```