Procedura Trasposizione

- Scopo

  Trasposizione della matrice

- Specifiche

  void trasposizione (int **Matrice, int **Trasposta, int Mc)

- Descrizione

  a) Background del problema
  La partizione di un Array consiste nel controllare i valori contenuti nelle celle speculari e nel caso in cui siano superiori o inferiori al discriminante x verrano scambiati o meno seguendo il concetto che nella parte sinista <=x e nella destra >x.
  Indicando con Matrice=$(m_0,..., m_{N-1})(m_0,..., m_{N-1})$ la matrice

  b) Descrizione del algoritmo
  L' algoritmo adoperato traspone la matrice dalla sorgente alla trasposta ne consegue il seguente codice in Pascal-LIKE

```
for i:=1 to Mc do
     for j:=1 to Mc do
          Trasposta(j)(i):=Matrice(i)(j)
     end for
end for
```

- Riferimenti bibliografici

  A. Murli, G. Laccetti, et al., Laboratorio di Programmazione I Liguori 2003

- Lista dei parametri

int Matrice[][]	:	Matrice sorgente in Input.
int Trasposta[][]	:	Matrice sorgente in Input.
int Mc	:	Lunghezza della matrice. Ricevuta in Input non va ad essere modificato.
int i	:	Indice.
int j	:	Cella speculare a quella puntata.

- Indcatore d' errore

  Nessuno

- Procedure ausiliarie

  Nessuno

- Raccomandazioni sull'uso

  Nessuno

- Complessità Computazionale

  a)Complessità di tempo
  $T(n)=T(n_2)$

b)Complessità di spazio

$T(n)=S(n^2)$

- Esempio d'uso

Esempio di programma chiamante

```
#include <stdio.h>
#include <stdlib.h>

   void trasposizione (int **Matrice, int **Trasposta, int Mc);

main ()
{
//Dichiarazione
int   **Matrice,
      **Trasposta;
int   i,
      j,
      Mc;  //Colonne

//Inizializzazione righe e colonne della Matrice
printf  ("\nInserisci il numero righe e colonne della Matrice (Quadrata): ");
scanf   ("%d",&Mc);

//allocazione della matrice originale
Matrice = (int **)malloc(Mc * sizeof(int *));
for (i=0; i<Mc; i++)
{
    Matrice[i] = (int *)malloc(Mc * sizeof(int));
}
//allocazione della matrice trasposta
Trasposta = (int **)malloc(Mc * sizeof(int *));
for (i=0; i<Mc; i++)
{
    Trasposta[i] = (int *)malloc(Mc * sizeof(int));
}


//Riempi Matrice
for (i=0; i<Mc; i++){
    for (j=0; j<Mc; j++){
        printf  ("\nInserisci il valore della cella[%d][%d]: ", i, j);
        scanf   ("%d",&Matrice[i][j]);
    }
}

 //Stampo la matrice del
 printf("\n Matrice caricata:\n");
 for (i=0; i<Mc; i++){
      for (j=0; j<Mc; j++){
           printf("%4d", Matrice[i][j]);
      }
      printf("\n");
 }
 _getch();

 //Chiamata delle function
 trasposizione ( Matrice, Trasposta, Mc);

 //Stampo del risultato
 printf("\n Matrice risultante:\n");
 for (i=0; i<Mc; i++){
      for (j=0; j<Mc; j++){
```

```
            printf("%4d", Trasposta[i][j]);
        }
        printf("\n");
    }
    _getch();
    //Dealloco la matrice
    for (i=0; i<Mc; i++)
       free(Matrice[i]);
    free(Matrice);

    //Dealloco la trasposta
    for (i=0; i<Mc; i++)
              free(Trasposta[i]);
              free(Trasposta);
    }
```

- Esempio di esecuzione

```
Inserisci il numero righe e colonne della Matrice (Quadrata): 3

Inserisci il valore della cella[0][0]: 1

Inserisci il valore della cella[0][1]: 2

Inserisci il valore della cella[0][2]: 3

Inserisci il valore della cella[1][0]: 4

Inserisci il valore della cella[1][1]: 5

Inserisci il valore della cella[1][2]: 6

Inserisci il valore della cella[2][0]: 7

Inserisci il valore della cella[2][1]: 8

Inserisci il valore della cella[2][2]: 9

 Matrice caricata:
   1   2   3
   4   5   6
   7   8   9
```