Passaggio delle matrici come parametri

Importante: nella funzione va sempre specificata la seconda dimensione (statica) della matrice.

Se non e' specificata non compila.

Se non e' corrispondente a quella chiamante possono succedere "problemi" negli accessi alla matrice.

Accesso ad un elemento di una matrice con l'aritmetica dei puntatori (gli elementi sono memorizzati per riga): (si suppone che ogni elemento occupi una sola cella di memoria)

M [i] [j] ➔

*(&M[0][0]+(i*dimC)+j)

Nota: si possono individuare anche modalita` piu` sofisticate utilizzando puntatori.

Esempio di corretto utilizzo:

```
#include <stdio.h>
#define dimR 3
#define dimC 2
void leggimat(int M[][dimC], int R, int C);
void stampamat(int M[][dimC], int R, int C);

main()
{
int A[dimR][dimC];
leggimat(A, dimR, dimC);
stampamat(A, dimR, dimC);
}

void leggimat(int M[][dimC], int R, int C)
{ int i,j;

     for(i=0; i<R; i++)
          for(j=0; j<C; j++)
               scanf("%d", &M[i][j]);
}

void stampamat(int M[][dimC], int R, int C)
{ int i,j;
     printf("stampa: \n");
     for(i=0; i<R; i++)
     {    for(j=0; j<C; j++)
                printf("%d\t", M[i][j]);
          printf("\n");
     }
}
```

Esempio1:

Corretta, la funzione azzera i valori della terza riga della matrice correttamente

```
#include <stdio.h>
void fun(int  M[][]);

main(void)
{int M[10][5];
int i,j;
for(i=0;i<10;i++)
  for(j=0;j<5;j++)
    M[i][j]=i+j;
printf("stampa prima della chiamata della funzione
: \n");
for(i=0;i<10;i++)
  { for(j=0;j<5;j++)
       printf("%d\t",M[i][j]);
       printf("\n");}
fun(M);
printf("stampa dopo la chiamata della funzione :
\n");
for(i=0;i<10;i++)
  { for(j=0;j<5;j++)
       printf("%d\t",M[i][j]);
       printf("\n");}
}

void fun(int M[][5])
{int j;
 for(j=0;j<5;j++)
        M[2][j]=0;

}
```

```
stampa prima della chiamata della funzione :
0     1     2     3     4
1     2     3     4     5
2     3     4     5     6
3     4     5     6     7
4     5     6     7     8
5     6     7     8     9
6     7     8     9     10
7     8     9     10    11
8     9     10    11    12
9     10    11    12    13
stampa dopo la chiamata della funzione :
0     1     2     3     4
1     2     3     4     5
0     0     0     0     0
3     4     5     6     7
4     5     6     7     8
5     6     7     8     9
6     7     8     9     10
7     8     9     10    11
8     9     10    11    12
9     10    11    12    13
```

Esempio 2: non lo compila

```
#include <stdio.h>
void fun(int  M[][]);
main(void)
{int M[10][5];
int i,j;
for(i=0;i<10;i++)
  for(j=0;j<5;j++)
    M[i][j]=i+j;
printf("stampa prima della chiamata della funzione
: \n");
for(i=0;i<10;i++)
  { for(j=0;j<5;j++)
       printf("%d\t",M[i][j]);
       printf("\n");}
fun(M);
printf("stampa dopo la chiamata della funzione :
\n");
for(i=0;i<10;i++)
  { for(j=0;j<5;j++)
       printf("%d\t",M[i][j]);
       printf("\n");}
}

void fun(int M[][])
/* non lo compila perche' non e' specificata la
seconda dimensione
della matrice */
{int j;
 for(j=0;j<5;j++)
        M[2][j]=0;

}
```

Esempio 3: Produce un problema nell'accesso agli elementi della matrice

```
#include <stdio.h>
void fun(int M[][]);

main(void)
{int M[10][5];
int i,j;
for(i=0;i<10;i++)
  for(j=0;j<5;j++)
    M[i][j]=i+j;
printf("stampa prima della chiamata della funzione
: \n");
for(i=0;i<10;i++)
  { for(j=0;j<5;j++)
       printf("%d\t",M[i][j]);
       printf("\n");}
fun(M);
printf("stampa dopo la chiamata della funzione :
\n");
for(i=0;i<10;i++)
  { for(j=0;j<5;j++)
       printf("%d\t",M[i][j]);
       printf("\n");}
}

void fun(int M[][3])
{int j;
 for(j=0;j<3;j++)
        M[2][j]=0;

}
```

```
stampa prima della chiamata della funzione :
0     1     2     3     4
1     2     3     4     5
2     3     4     5     6
3     4     5     6     7
4     5     6     7     8
5     6     7     8     9
6     7     8     9     10
7     8     9     10    11
8     9     10    11    12
9     10    11    12    13
stampa dopo la chiamata della funzione :
0     1     2     3     4
1     0     0     0     5
2     3     4     5     6
3     4     5     6     7
4     5     6     7     8
5     6     7     8     9
6     7     8     9     10
7     8     9     10    11
8     9     10    11    12
9     10    11    12    13
```