Il linguaggio C

Complementi sui puntatori

I puntatori a funzione

- Consideriamo la funzione

```
int somma (int x, int y) {
    return x + y;
}
```

  - se proviamo ad eseguire

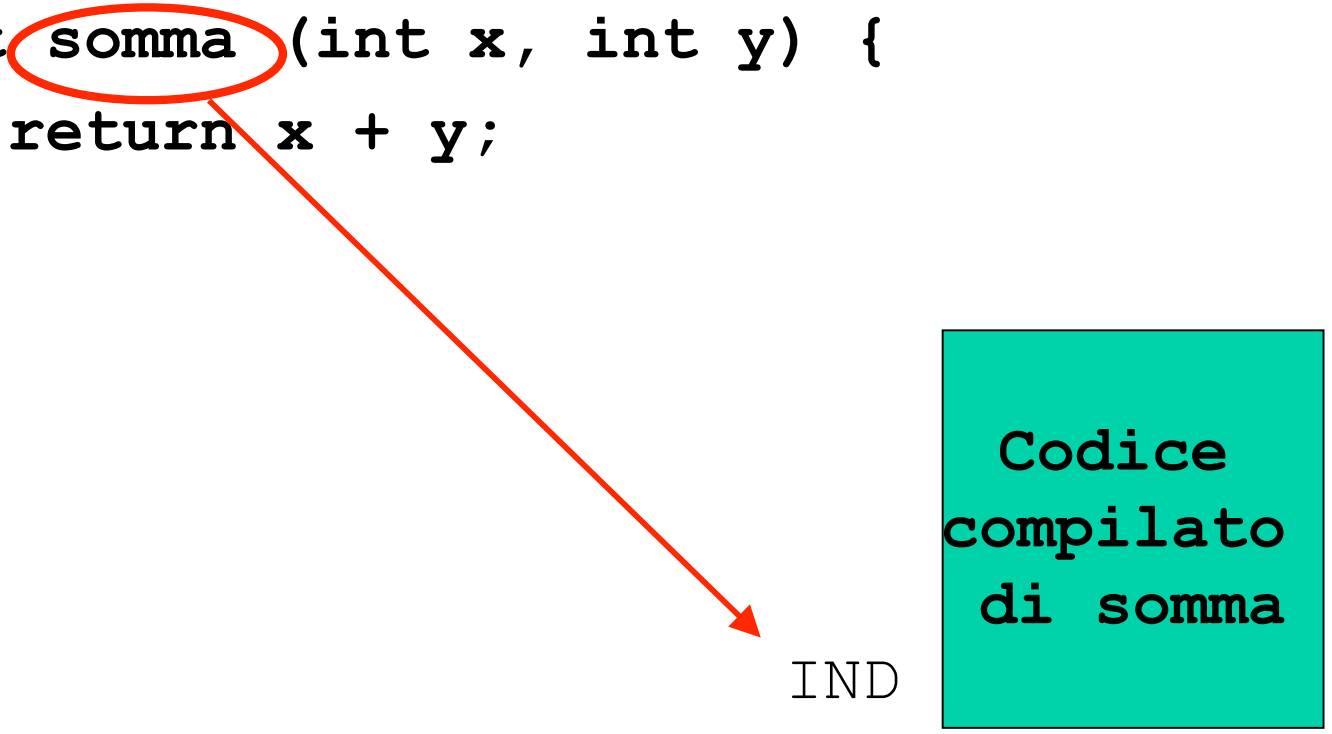
```
printf("%p", somma);
```

    otteniamo un valore esadecimale che rappresenta un indirizzo legale del nostro programma
  - ??????????????????????????

I puntatori a funzione (2)

- Consideriamo la funzione

```
int somma (int x, int y) {
    return x + y;
}
```



somma è un puntatore costante con valore pari a **IND**

I puntatori a funzione (3)

- Consideriamo la funzione

```
int somma (int x, int y) {
    return x + y;
}

/* variabile di tipo funzione
  (int,int)->int */
int (*fun) (int,int);
int a;

fun = somma;
a = fun(3,5);
```

I puntatori a funzione (4)

- Consideriamo la funzione

```
int somma (int x, int y) {
    return x + y;
}

/* variabile di tipo funzione
  (int,int)->int */
int (*fun) (int,int);
int a;

fun = somma;
a = fun(3,5);
```

Ma a che serve ???????????

I puntatori a funzione (5)

- Serve a definire funzioni che prendono come argomenti altre funzioni (*di ordine superiore*)

```
void map (int (*fun) (int),
          int x[ ], int l) {
   for(i = 0; i < l; i++)
     x[i] = fun(x[i]);
}
```

è un iteratore che applica la funzione **fun** a tutti gli elementi dell'array **x**

I puntatori a funzione (6)

- Esempio di uso della map

```
int piu_uno (int x) {
   return x+1;}
int quad (int x) {
   return x*x;}
…
int a[3] = {3,4,5};
map(piu_uno,a,3); /* somma uno a
  tutti gli elementi */
map(quad,a,2); /* eleva al quadrato
  i primi due elementi */
```

Argomenti della linea di comando

- Gli argomenti della linea di comando sono accessibili all'interno della funzione main
  - il SO li inserisce sullo stack prima di attivare il processo
  - il formato in cui sono resi disponibili è fisso

```
int main (int argc, char * argv[ ]) {
    …
}
```

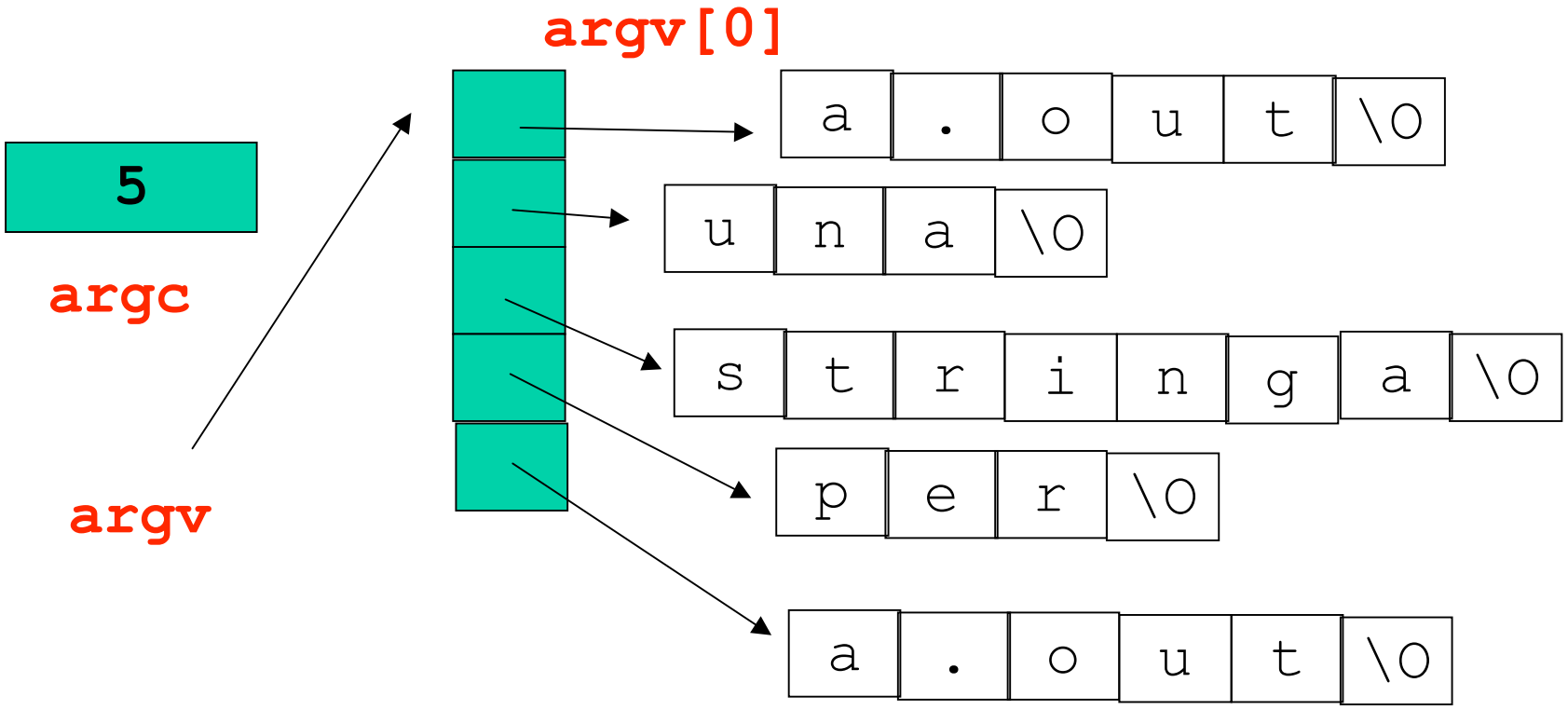
Numero di argomenti nella linea di comando

Array di puntatori agli argomenti (ciascuno è una stringa, tipo char*)

Argomenti della linea di comando (2)

- Esempio

```
%> a.out una stringa per a.out
```



Argomenti della linea di comando (3)

- Esempio : Schema di programma che stampa gli argomenti sulla linea di comando

```
int main (int argc, char * argv[ ]) {
 …
 for(i = 0; i < argc; i++)
   printf("arg %d: %s", i, argv[i]);
 …
}
```

Array multidimensionali come array di puntatori

- Vogliamo permettere la definizione di funzioni su array multidimensionali che non dipendono dal valore delle dimensioni successive alla prima

```
int sum_mat (int ** MAT,
             int n, int m) {
   ...

}
```

Nome della matrice

Numero di righe, colonne

Array multidimensionali come array di puntatori (2)

- Vogliamo accedere agli elementi dell'array usando la solita notazione [-][-]

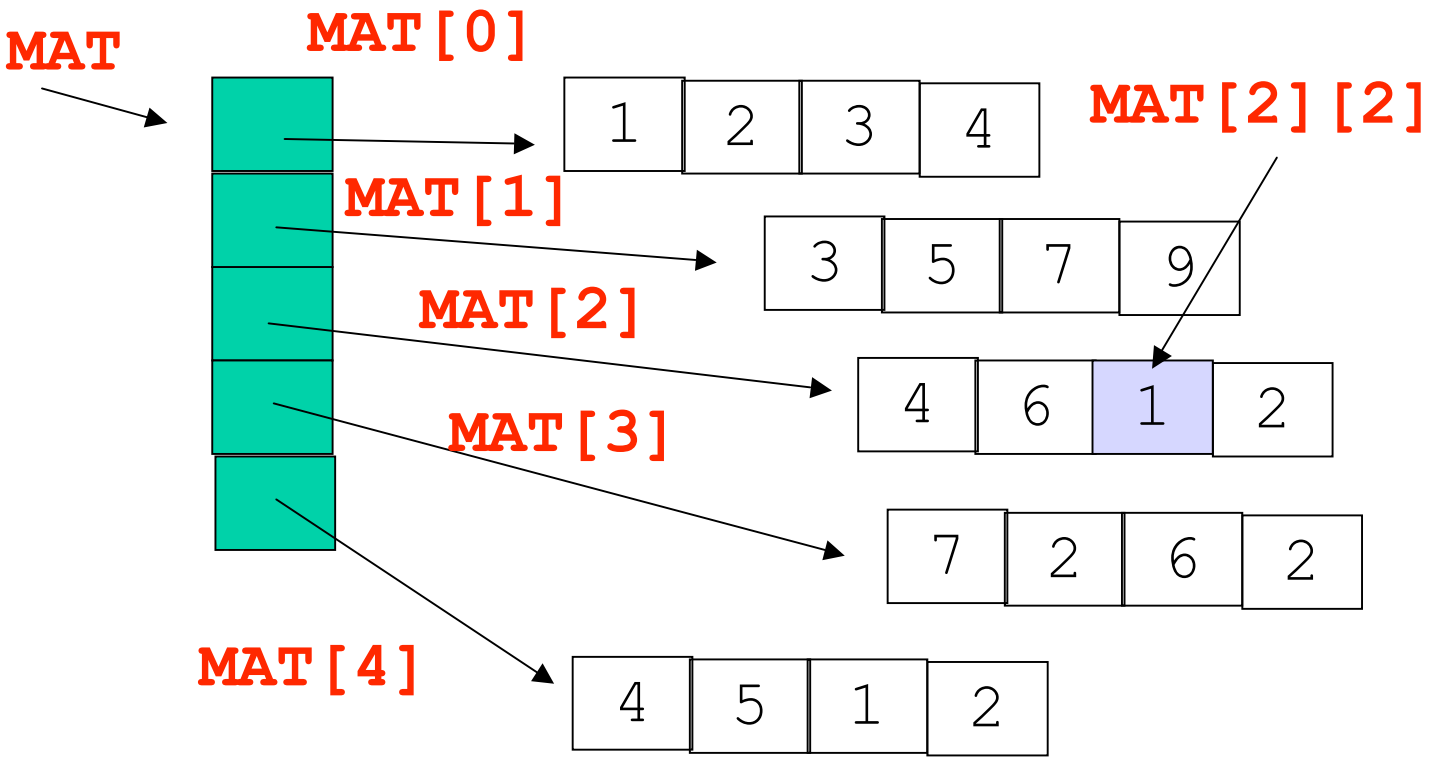
```
int sum_mat (int ** MAT,
             int n, int m) {
    …
    MAT[i][j] = …;

    *(MAT + i*m + j)
}
```

Il compilatore dovrebbe conoscere il legame fra questi due oggetti

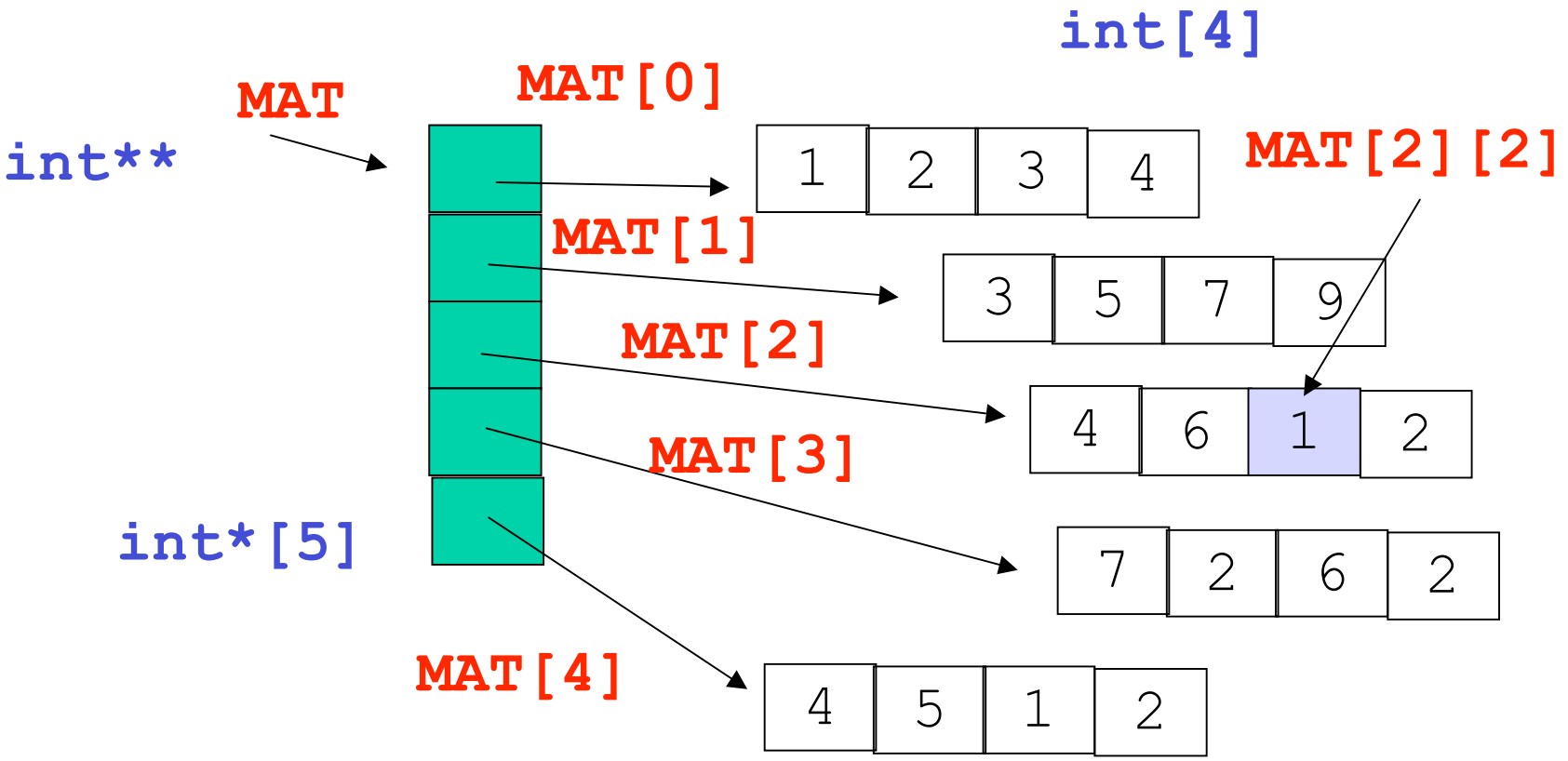
Array multidimensionali come array di puntatori (3)

- Una alternativa: abbandonare l'allocazione contigua per righe



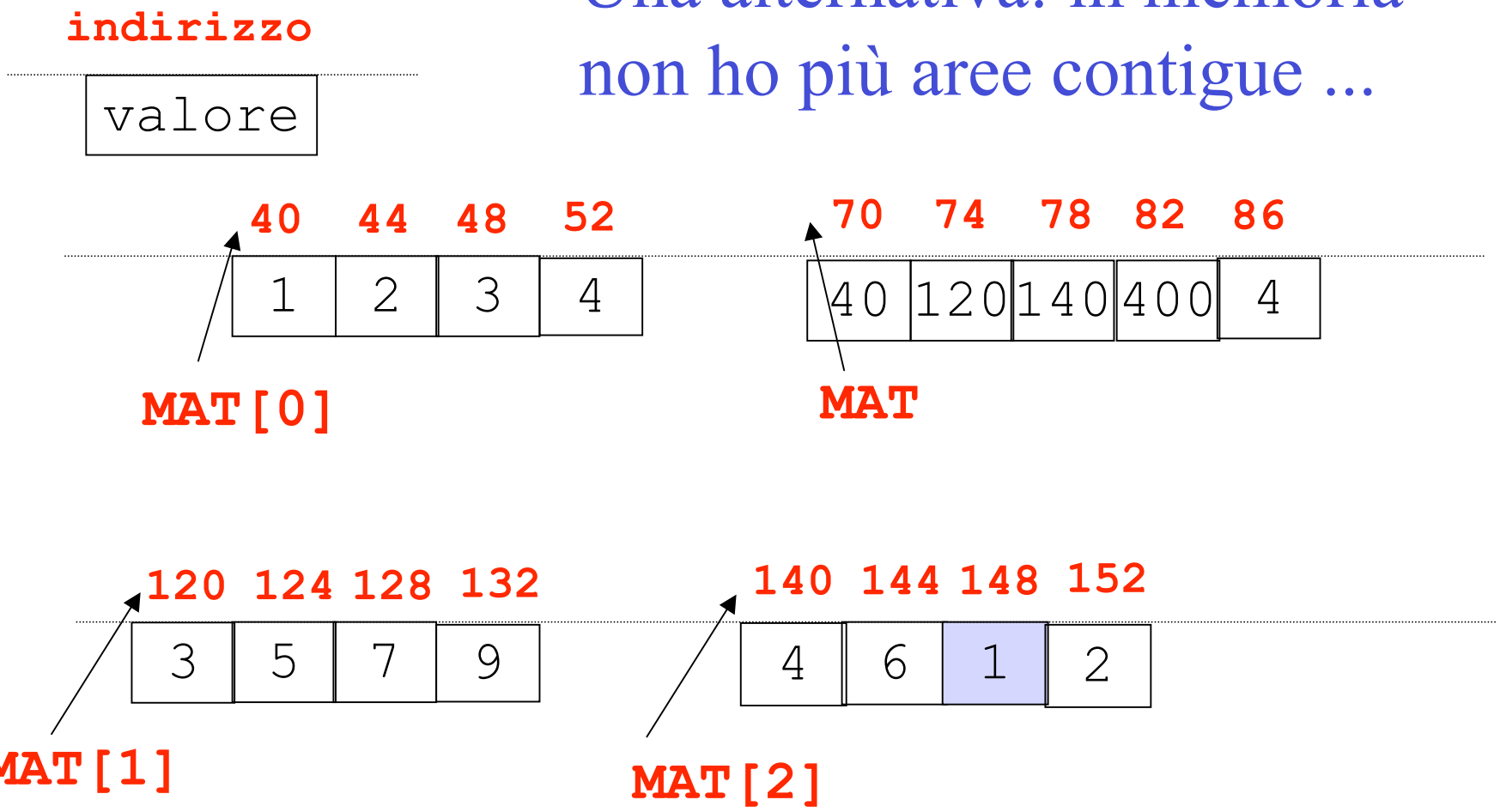
Array multidimensionali come array di puntatori (4)

- Una alternativa: vediamo i tipi ...



Array multidimensionali come array di puntatori (5)

- Una alternativa: in memoria non ho più aree contigue ...



Array multidimensionali come array di puntatori (6)

- Perché funziona?

```
int sum_mat (int ** MAT,
             int n, int m)  {
             …
      MAT[i][j] = …;

      *(*(MAT + i) + j)

}
```

Questo legame non interessa più!

Array multidimensionali come array di puntatori (7)

- Funzione di allocazione

```
int ** mat_new(unsigned int m,
               unsigned int n) {
   int i, ** a, errore = FALSE;
   a = malloc(m*sizeof(int*));
   if (a == NULL) return NULL;
   for(i = 0; (i < m) && (!errore); i++) {
     a[i] = malloc(n*sizeof(int));
     if (a[i] == NULL) errore = TRUE;
   }
   if (errore) /* gestione errori */
   else return a;
}
```

Array multidimensionali come array di puntatori (8)

- Funzione di deallocazione

```
void mat_free(int ** a, unsigned int m) {
  int i;
  /* dealloco tutte le righe */
  for(i = 0; i < m; i++)
    free(a[i]);
  /* dealloco l'array dei puntatori alle
     righe */
  free(a);
}
```