Array bidimensionali

Contenuti rieditati delle slide della Prof. L. Caponetti

Struttura array

- L’array è una struttura dati omogenea, cioè costituita da elementi dello stesso tipo detto tipo **base**
- Il tipo delle componenti di un array può essere **semplice**, int float, char, oppure **strutturato**
- Se ogni componente è un array allora si ha una **matrice** o **array bidimensionale**

Array bidimensionale

- Un array bidimensionale può essere considerato un **array di array monodimensionale**, cioè ogni componente dell'array è esso stesso un array
- L'accesso a ogni componente di un array bidimensionale si ha tramite una **coppia di indici** (i, j).
- Il primo indice si riferisce alla riga ed il secondo alla colonna

Array bidimensionali

- Matice rettangolare di 4 righe e 3 colonne

(0,0)	**(0,1)**	**(0,2)**
(1,0)	**(1,1)**	**(1,2)**
(2,0)	**(2,1)**	**(2,2)**
(3,0)	**(3,1)**	**(3,2)**

Rappresentazione sequenziale

- Sia **a** una matrice di **n** righe ed **m** colonne
- Supponiamo che ogni elemento di a occupi **L** byte

(0,0)	**(0,1)**	**(0,m-1)**
(1,0)	**(1,1)**	**(1,m-1)**
…	**…**	**…**
(n-1,0)	**(n-1,1)**	**(n-1,m-1)**

Rappresentazione sequenziale

- Come per gli array monodimensionale, gli elementi della matrice a sono rappresentati in memoria centrale in maniera sequenziale, a partire dall'indirizzo iniziale o base
- L'indirizzo base è uguale all'indirizzo dell'elemento **a[0][0]**, o **&a[0][0]**. Per fare riferimento a tale indirizzo si può utilizzare semplicemente l'identificatore della matrice

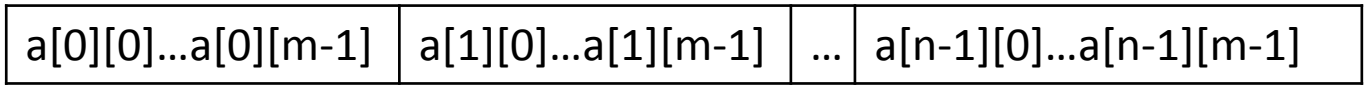
Rappresentazione sequenziale

- Gli elementi di una matrice possono essere memorizzati in sequenza una riga dopo l'altra o una colonna dopo l'altra

- Il linguaggio C utilizza a **memorizzazione per righe**

Numero di byte occupati

- La matrice **a** di **n** righe ed **m** colonne occupa **n*m*L** locazioni di memoria (byte)



riga 0 riga 1 riga n-1

Memorizzazione per righe

- L'indirizzo del generico elemento a[i][j] è dato da:

  ind(a[i][j]) = ind(a) + i*m*L + j*L

Indirizzo base

numero di byte occupati dalle i righe prima dell'elemento a[i][j]

numero di byte occupati dagli j elementi della riga i prima dell'elemento a[i][j]

Memorizzazione per righe

ind(a[i][j]) = ind(a) + i*m*L + j*L

- Nel calcolo dell'indirizzo di a[i][j] interviene solo m, numero di colonne della matrice, e non il numero di righe
- L'indirizzo dell'array a può essere indicato con &a[0][0] oppure solo con **a**, nome dell'array

Sintassi

- Dichiarazione in C di una variabile array

  <tipo-base>

  <identificatore-variabile>

  [<costante-righe>][<costante-colonne>]

Sintassi

- Dichiarazione in C di una variabile array

  typedef <tipo-base>
  <identificatore-tipo>
  [<costante-righe>][<costante-colonne>]

Esempio

```
int a[10][20]
```

Oppure

```
#define NRIGHE 10
#define NCOLONNE20
typedef int matrice [NRIGHE][NCOLONNE]
matrice a;
```

- Mediante typedef si assegna il nome matrice al tipo costruito come un array bidimensionale di 10 righe e 20 colonne

#define

```
#define NRIGHE 10
```

- #define è una direttiva per il preprocessore. Ogni occorrenza di NRIGHE nel programma viene sostituita con 10
- Lo svantaggio è che il preprocessore non effettua alcuna analisi sintattica

Scansione degli elementi di una matrice

- Sia **a** una matrice di **n** righe ed **m** colonne
- Indichiamo con **i,j** gli indici per accedere agli elementi di a, dove i varia da 0 a (n-1) e j varia da 0 a (m-1)
- La scansione degli elementi di a può essere effettuata per righe o per colonne, in relazione al problema da risolvere

Scansione per righe

- La scansione per righe richiede che si acceda ad a una riga dopo l'altra a partire dalla riga iniziale
- Tale scansione può essere effettuata mediante 2 cicli innestati

Mentre i varia da 0 a n-1 righe

Mentre j varia da 0 a m-1 colonne

Elabora l'elemento a[i][j]

Scansione per colonne

- La scansione per colonne richiede che si acceda ad a una colonna dopo l'altra a partire dalla colonna iniziale
- Tale scansione può essere effettuata mediante 2 cicli innestati

Mentre i varia da 0 a n-1 colonne

Mentre j varia da 0 a m-1 righee

Elabora l'elemento a[i][j]

Codice

```
#define NRIGHE 10
#define NCOLONNE 20
typedef int matrice[NRIGHE][NCOLONNE]
```

Codice

- Funzione Leggi matrice

```
void leggi (matrice a, int n, int m)
{int i=0, j;
while (i<n)
        {j=0;
          while (j<m)
                    {printf("a[%d,%d] = ", i,j);
                     scanf("%d", &a[i][j]);
                     j++}
       i++;}
}
```

Codifica

- Funzione visualizza matrice

```
void visualizza (int a[][NCOLONNE], int n, int m)
{int i=0, j;
while (i<n)
        {j=0;
         while (j<m)
                {printf("%10d", a[i][j]);
                printf("<n");
                j++;}
       i++;}
}
```

Codice

- Funzione trasponi matrice

```
void trasponi (int a[][NCOLONNE], int n, int m)
    {int i=0, j, w;
    while (i<n)
            {j=i+1;
             while (j<m)
                        {w=a[i][j];
                        a[i][j]=a[j][i];
                        a[j][i]=w;
                        j++;}
            i++;}
    }
```

-