Programmazione Procedurale

Tipi Strutturati
Array Multidimensionali

versione 3.0

1

Sommario

- Introduzione
- Dichiarazione di Array Bidimensionali
  - ⇨Sintassi e Semantica
  - ⇨Interpretazione degli Array Bidimensionali
- Utilizzo degli Array Bidimensionali
  - ⇨Scansione degli Array Bidimensionali
- Tecniche Algoritmiche

2

Array Bidimensionali

- Finora
  - ⇨array monodimensionali
- In realtà
  - ⇨è possibile costruire array con un numero arbitrario di dimensioni
  - ⇨particolarmente interessanti sono gli array con due dimensioni
  - ⇨consentono di rappresentare strutture bidimensionali (es: matrici, scacchiere ecc.)

3

Dich. di Array Bidimensionale

- Sintassi

```
<tipo> <nome> [<dim1>][<dim2>];
```

- Dove
  - ⇨<tipo> è uno dei tipi di dato del linguaggio
  - ⇨<nome> è un identificatore
  - ⇨<dim1> e <dim2> sono valori costanti interi positivi (costante numerica o simbolica o espressione costante)

4

Dich. di Array Bidimensionale

- Semantica
  - ⇨la dichiarazione ha l'effetto di dichiarare <dim1>*<dim2> variabili del tipo specificato
  - ⇨più una ulteriore variabile corrispondente all'array nel suo complesso
  - ⇨nomi delle variabili costituiti di tre parti:
    - ⇨l'identificatore dell'array
    - ⇨una coppia di indici interi, il primo da 0 a <dim1>-1 ed il secondo da <dim2>-1

5

Dich. di Array Bidimensionale

- Esempio

```
const int N = 2;
const int M = 3;
int matrice[N][M];
```

Ha l'effetto di dichiarare 7 variabili, di cui 6 di tipo int:
matrice[0][0]
matrice[0][1]
matrice[0][2]
matrice[1][0]
matrice[1][1]
matrice[1][2]

#99	...	xxx
#100	matrice	#101
#101	matrice[0][0]	xxx
#102	matrice[0][1]	xxx
#103	matrice[0][2]	xxx
#104	matrice[1][0]	xxx
#105	matrice[1][1]	xxx
#106	matrice[1][2]	xxx
#107	...	xxx

6

Dich. di Array Bidimensionale

- Esempio

```
char scacchiera[3][3];
```

Ha l'effetto di dichiarare 10 variabili, di cui 9 di tipo char:
scacchiera[0][0]
scacchiera[0][1]
scacchiera[0][2]
scacchiera[1][0]
...
scacchiera[2][1]
scacchiera[2][2]

#99	...	xxx
#100	scacchiera	#101
#101	scacchiera[0][0]	xxx
#102	scacchiera[0][1]	xxx
#103	scacchiera[0][2]	xxx
#104	scacchiera[1][0]	xxx
#105	scacchiera[1][1]	xxx
#106	scacchiera[1][2]	xxx
#107	scacchiera[2][0]	xxx
#108	scacchiera[2][1]	xxx
#109	scacchiera[2][2]	xxx

7

Dich. di Array Bidimensionale

- Inizializzazione
  - ⇨anche per gli array bidimensionali è possibile inizializzare le componenti alla dichiarazione
  - ⇨inizializzazione "riga" per "riga"
- Esempio

```
int matrice[2][3] = { {1, -7, 6},
                      {12, 45, 6} }
```

valori della prima "riga" (primo indice = 0)

valori della seconda "riga" (primo indice = 1)

8

Dich. di Array Multidimensionale

- Array Multidimensionali
  - ⇨è possibile dichiarare array con un numero qualsiasi di dimensioni
  - ⇨raramente però ne servono più di due
- Esempio

```
int cubo[3][3][3];
```

ha l'effetto di dichiarare
3x3x3 = 27 variabili
di tipo intero

9

Interpretazione degli Array Bidim.

- Interpretazione degli array bidimensionali
  - ⇨la programmazione sugli array bidimensionali è meno semplice rispetto a quelli monodim.
  - ⇨è utile darne un'interpretazione che aiuti a pensare all'array in termini più comodi
- Due possibili interpretazioni
  - ⇨interpretazione su due dimensioni
  - ⇨interpretazione come array di array

10

Interpretazione degli Array Bidim.

- I interpretazione: due dimensioni
  - ⇨può essere utile, durante la scrittura del programma, pensare ad un array bidim. come se fosse realmente su due dimensioni
  - ⇨ovvero organizzato per righe e colonne
  - ⇨in questo caso il primo indice può essere pensato come l'indice della riga
  - ⇨il secondo indice come indice della colonna
  - ⇨si tratta però di un'astrazione

11

Interpretazione degli Array Bidim.

- Esempio

```
char scacchiera[3][3];
```

	colonna 0	colonna 1	colonna2
riga 0	scacchiera[0][0]	scacchiera[0][1]	scacchiera[0][2]
riga 1	scacchiera[1][0]	scacchiera[1][1]	scacchiera[1][2]
riga 2	scacchiera[2][0]	scacchiera[2][1]	scacchiera[2][2]

ATTENZIONE: si tratta di un'astrazione
per il programmatore, in quanto le variabili sono organizzate diversamente nella memoria

12

Interpretazione degli Array Bidim.

- II interpretazione: array di array
  - ⇨alternativamente, è possibile pensare ad un array bidimensionale come ad un array monodimensionale
  - ⇨le cui componenti non sono variabili semplici, ma array monodimensionali
  - ⇨si tratterebbe di conseguenza di un “array di array”

13

Interpretazione degli Array Bidim.

- Esempio

```
char scacchiera[3][3];
```

un array “esterno” monodimensionale di 3 componenti, ciascuna delle quali è a sua volta un array “interno” monodimensionale

scacchiera[0]	scacchiera[0][0]
	scacchiera[0][1]
	scacchiera[0][2]
scacchiera[1]	scacchiera[1][0]
	scacchiera[1][1]
	scacchiera[1][2]
scacchiera[2]	scacchiera[2][0]
	scacchiera[2][1]
	scacchiera[2][2]

14

Utilizzo degli Array Bidimensionali

- Nel caso di array bidimensionali
  - ⇨è necessario utilizzare due indici
- Sintassi
  - ⇨`<nomeArray>[<espr1>][<espr2>]`
- Semantica analoga alla precedente
  - ⇨i risultati del calcolo delle due espressioni vengono utilizzati come indici per identificare la componente dell'array

15

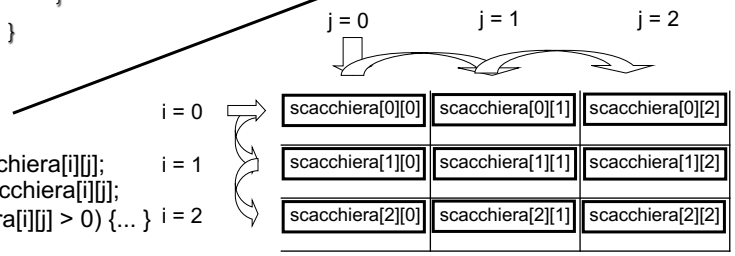
Scansione degli Array Bidimensionali

- Tipicamente
  - ⇨vengono utilizzati due cicli nidificati
  - ⇨uno esterno per il primo indice, uno interno per il secondo indice
- Intuizione
  - ⇨il ciclo esterno serve a scandire le "righe" e il ciclo interno serve a scandire le "colonne" di una riga

16

Scansione degli Array Bidimensionali

```
char scacchiera[N][M];
int i, j;
for (i = 0; i < N; i++) {
   for (j = 0; j < M; j++) {
      <operazione su scacchiera[i][j]>
   }
}
```



esempi:
cin >> scacchiera[i][j];
cout << scacchiera[i][j];
if (scacchiera[i][j] > 0) {... }

17

Esempio: Somma di Matrici

```
const int N=2; const int M=3;

void leggi (float matrice[N][M]);
void stampa (float matrice[N][M]);
void somma(float matrice1[N][M], float matrice2[N][M],
           float matSomma[N][M]);

int main() {
    float matrice1[N][M], matrice2[N][M];
    float matrice3[N][M];
    leggi(matrice1);
    stampa(matrice1);
    leggi(matrice2);
    stampa(matrice2);
    somma(matrice1, matrice2, matrice3);
    stampa(matrice3);
    return 0;
}
```

18

Esempio: Somma di Matrici

```
void leggi (float matrice[N][M]){
    cout << "Immetti gli elementi \n";
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            cin >> matrice[i][j];
        }
    }
    return;
}
```

due cicli nidificati per la lettura

alla I esecuzione: i=0, j=0 - cin>> matrice[0][0]
alla II esecuzione: i=0, j=1 - cin>> matrice[0][1]
alla III esecuzione: i=0, j=2 - cin>> matrice[0][2]
alla IV esecuzione: i=1, j=0- cin>> matrice[1][0]
...

19

Esempio: Somma di Matrici

```
void stampa (float matrice[N][M]){
    cout << "Valori della matrice:\n";
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            cout << matrice[i][j] << " ";
        }
        cout << endl;
    }
    return;
}
```

al termine di ciascuna riga si va a capo

20

Esempio: Somma di Matrici

```
void somma(float mat1[N][M], float mat2[N][M],
           float matSomma[N][M]){
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            matSomma[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
    return;
}
```

la scansione delle tre matrici procede in parallelo

21

Utilizzo degli Array Bidimensionali

- Uso dei cicli
  - ⇨normalmente due cicli nidificati
- Attenzione
  - ⇨è uno dei pochi casi in cui è opportuno usarli senza pregiudicare la leggibilità del codice
  - ⇨altro caso: ciclo di convalida dei dati
  - ⇨in tutti gli altri casi: difetto di progetto dell'algoritmo (sarebbe necessario un ulteriore raffinamento del codice)

22

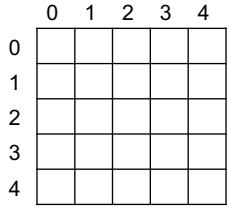
Utilizzo degli Array Bidimensionali

- Nota
  - ⇨la somma tra matrici viene calcolata da una procedura
  - ⇨non sarebbe stato più appropriato utilizzare una funzione, che riceve due matrici e restituisce come risultato la mat. somma ?
- Regola
  - ⇨le funzioni NON possono restituire risultati che sono array

23

Tecniche Algoritmiche

- Generazione della matrice diagonale
  - ⇨solo gli elementi sulla diagonale sono non nulli
  - ⇨elemento sulla diagonale: i == j



24

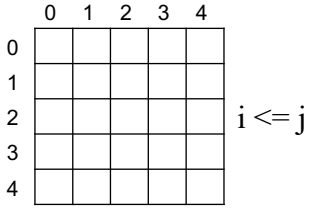
Tecniche Algoritmiche

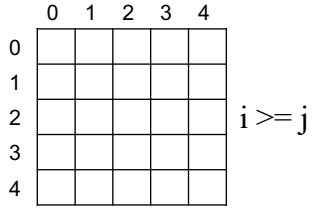
```
void generaMatDiagonale(int matrice[N][N], int valore) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (i == j) {
                matrice[i][j] = valore;
            } else {
                matrice[i][j] = 0;
            }
        }
    }
    return;
}
```

25

Tecniche Algoritmiche

- Generazione della matrice triangolare inf.
  - ⇨solo gli elementi sulla diagonale e quelli sotto sono non nulli
  - ⇨elemento sopra o sotto la diagonale:





26

Tecniche Algoritmiche

```
void generaMatTriangolare(int matrice[N][N], int valore) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (i >= j) {
                matrice[i][j] = valore;
            } else {
                matrice[i][j] = 0;
            }
        }
    }
    return;
}
```

27

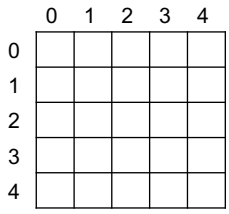
Tecniche Algoritmiche

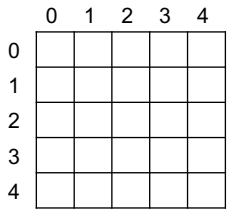
- Una annotazione importante
  - ⇨la matrice nel suo complesso è una struttura bidimensionale
  - ⇨per le elaborazioni servono due cicli
- Ma
  - ⇨le righe, le colonne e la diagonale sono strutture monodimensionali
  - ⇨per le elaborazioni basta un unico ciclo

28

Tecniche Algoritmiche

- Infatti
  - ⇨per gli el. di una riga l'indice di riga è fissato
  - ⇨per gli el. di una colonna l'indice di colonna è fissato





29

Tecniche Algoritmiche

```
int sommaRiga(int matrice[N][N], int riga) {
    int somma = 0;
    for (int j = 0; j < N; j++) {
        somma += matrice[riga][j];
    }
    return somma;
}

int sommaColonna(int matrice[N][N], int colonna) {
    int somma = 0;
    for (int i = 0; i < N; i++) {
        somma += matrice[i][colonna];
    }
    return somma;
}
```

30

Tecniche Algoritmiche

- La diagonale
  - ⇨un elemento per ciascuna riga
  - ⇨l'indice di colonna è uguale all'indice di riga

```
int sommaDiagonale(int matrice[N][N]) {
    int somma = 0;
    for (int i = 0; i < N; i++) {
        somma += matrice[i][i];
    }
    return somma;
}
```

31

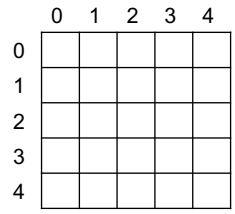
Tecniche Algoritmiche

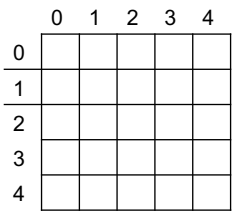
- Attenzione
  - ⇨in questo esempio: somma
  - ⇨ma allo stesso modo si programmano le altre tecniche algoritmiche notevoli
  - ⇨es: conteggio
  - ⇨es: massimo (della matrice, di una riga, di una colonna, della diagonale)
  - ⇨es: verifica di condizioni (sulla matrice, su una riga, su una colonna, sulla diagonale

32

Tecniche Algoritmiche

- Analisi del contorno
  - ⇨data una cella x, y, bisogna contare gli 1 nel contorno
  - ⇨righe coinvolte: da x-1 a x+1
  - ⇨col. coinvolte: da y-1 a y+1
  - ⇨es: x=2, y=1
  - ⇨attenzione agli sconfinamenti
  - ⇨es: x=1, y=0





33

Tecniche Algoritmiche

```
int contaVicini (int scacchiera[N][M], int x, int y) {
    int conta = 0;
    for (int i = x - 1; i <= x + 1; i++) {
        for (int j = y - 1; j <= y + 1; j++) {
            if (i >= 0 && i < N &&
                j >= 0 && j < M) {
                 conta += scacchiera[i][j];
            }
        }
    }
    return conta - scacchiera[x][y];
}
```

34

Riassumendo

- Array bidimensionali
  - ⇨Interpretazione
- Dichiarazione
- Utilizzo
  - ⇨Scansione
- Tecniche Algoritmiche

35

Termini della Licenza

36