Dipartimento di Informatica e Sistemistica
Antonio Ruberti

“Sapienza” Università di Roma

Vettori
Algoritmi elementari di ordinamento

Corso di Fondamenti di Informatica

Laurea in Ingegneria Informatica

(Canale di Ingegneria delle Reti e dei Sistemi Informatici - Polo di Rieti)

Anno Accademico 2007/2008

Prof. Paolo Romano

Si ringrazia il Prof. Alberto Finzi per aver reso
disponibile il materiale didattico sul quale si basano queste slides

Tipi di dato semplici e strutturati

i tipi di dato visti finora erano tutti semplici:
`int, char, float`

Gli array sono uno dei tipi di dato strutturati sono composti da *elementi omogenei* (tutti dello stesso tipo)

- ogni elemento è identificato all'interno dell'array da un *numero d'ordine* detto *indice* dell'elemento
- il numero di elementi del vettore è detto *lunghezza* (o*dimensione*) del vettore

Array monodimensionale (vettore)

Dichiarazione variabili tipo vettore:

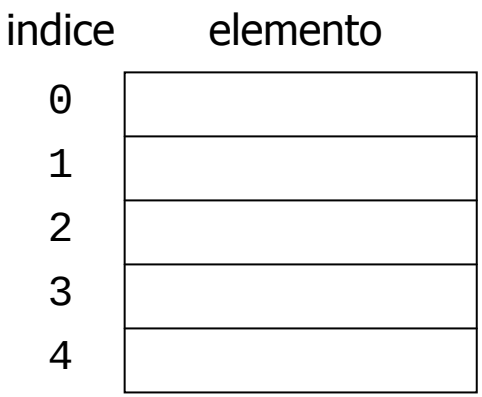
tipo-elementi nome-array[lunghezza];

Esempio: `int v[5];`

Alloca un vettore di 5 elementi: ovvero 5 locazioni di memoria consecutive, ciascuna contenente un intero. La **lunghezza** del vettore è 5.

La *lunghezza di un vettore deve essere costante* (nota a tempo di compilazione). In C l'indice degli elementi} va sempre da 0 a *lunghezza*-1.

Array monodimensionale (vettore)



`v[i]` denota l'elemento del vettore `v` di indice `i`

Ogni elemento del vettore è una variabile.

Esempio:

```
int vet[6], a;

vet[0] = 15;
a = vet[0];
vet[1] = vet[0] + a;
```

Esempio:

```
...
a = 2;
vet[a] = 12;
vet[a+1] = 23;
```

L'indice del vettore deve essere un intero.

Manipolazione vettori

- avviene solitamente attraverso cicli `for`
- l'indice del ciclo varia in genere da `0` a *lunghezza-1*.
- si può definire la lunghezza come una costante
  Es.: `#define LUNG 6`

Esempio: Lettura e stampa di un vettore.

```
#include <stdio.h>
#define LUNG 5

int main () {
int v[LUNG]; /*vettore di LUNG elementi, indicizzati da 0 a LUNG-1 */
int i;
  for (i = 0; i < LUNG; i++) {
    printf("Inserisci l'elemento di indice %d: ", i);
    scanf("%d", &v[i]);
  }
  printf("Indice Elemento\n");

  for (i = 0; i < LUNG; i++) {
    printf("%6d %8d\n", i, v[i]);
  }
  return 0;
}
```

Inizializzazione vettori

Gli elementi del vettore possono essere inizializzati con *valori costanti* (valutabili a compile-time) contestualmente alla dichiarazione del vettore .

Esempio: `int n[4] = {11, 22, 33, 44};`

- l'inizializzazione deve essere contestuale alla dichiarazione

```
int n[4];
n = {11, 22, 33, 44};   Errore!
```

- se ci sono meno inizializzatori di elementi, quelli rimanenti vengono posti a 0

```
int n[10] = {3};
float af[5] = {0.0};
int x[5] = {};   Errore!
```

Inizializzazione vettori

se ci sono più inizializzatori di elementi, si ottiene un errore di sintassi

```
int v[2] = {1, 2, 3};
```
errore!

se si mette una lista di inizializzatori, si può evitare di specificare la lunghezza (viene presa la lunghezza della lista)

`int n[] = {1, 2, 3};` equivale a `int n[3] = {1, 2, 3};`

In C l'unica operazione possibile sugli array è l'accesso agli elementi. Non si possono effettuare direttamente delle assegnazioni tra vettori.

```
int a[3] = {11, 22, 33};
int b[3];
b = a;
```
errore!

Esempi

Esempio: Calcolare la somma degli elementi di un vettore.

```
int a[10], i, somma = 0;
...
for (i = 0; i < 10; i++)
  somma += a[i];
printf("%d", somma);
```

Esempio: Calcolare il massimo di un vettore di 10 elementi interi.

```
int a[10], i, max;
...
max = a[0];
for (i = 1; i < 10; i++)
  if (a[i] > max) max = a[i];
printf("%d", max);
```

Esempio1

Esempio: Leggere 20 reali e stampare i valori inferiori al 50% della media. 2 aspetti da considerare

- le elaborazioni sui 20 elementi sono molto simili
- prima di poter iniziare a stampare i risultati bisogna avere letto tutti e 20 i valori

```
#include <stdio.h>
#define DIM 4
int main() {
  double ris[DIM];
  double media;
  int i;
  printf("Inserire i %d risultati dell'esperimento:\n", DIM);
  for (i = 0; i < DIM; i++) {
     printf("Inserire risultato n. %d: ", i);
     scanf("%lg", &ris[i]);
   }
  /* calcolo della media */
  media = 0;
  for (i = 0; i < DIM; i++)
    media = media + ris[i];
  media = media/DIM;
  printf("Valore medio: %lg\n", media);
```

Esempio1 (continua)

```
/* stampa dei valori minori di media*0.5 */
  printf("Stampa dei valori minori di media*0.5:\n");
  for (i = 0; i < DIM; i++)
    if (ris[i] < media * 0.5)
      printf("Risultato n. %d: %lg\n", i, ris[i]);
  return 0;

}
```

Esempio: Leggere una sequenza di caratteri terminata da `'\n'` e stampare le frequenze delle cifre da 0 a 9.

```
#include <stdio.h>
#define DIM  10
int main(){
  int i;
  int a[DIM] = {0};
  char ch = ' ';
```

Esempio2 (continua)

```
 while (ch != '\n') {
    switch(ch) {
    case '0': a[0]++; break;
    case '1': a[1]++; break;
    case '2': a[2]++; break;
    case '3': a[3]++; break;
    case '4': a[4]++; break;
    case '5': a[5]++; break;
    case '6': a[6]++; break;
    case '7': a[7]++; break;
    case '8': a[8]++; break;
    case '9': a[9]++; break;
    }
    scanf("%c",&ch);
  }
  printf("Le frequenze sono:\n");
  for (i = 0; i < DIM; i++)
    printf("Freq. di %d: %d\n", i, a[i]);
  return 0;
}
```

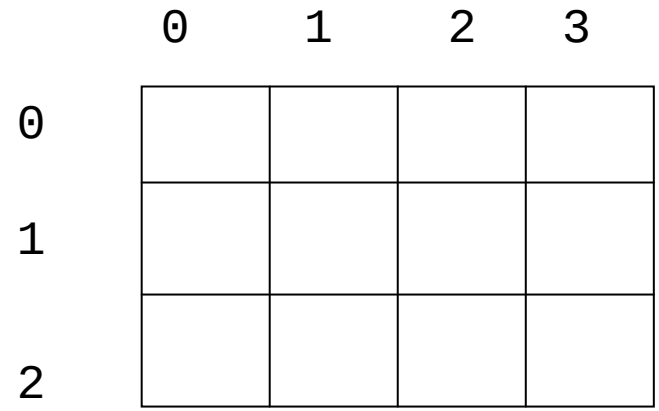
Array multidimensionali

Dichiarazione di array multidimensionali

tipo-elementi *nome-array* [*lung-1*][*lung-2*]…[*lung-n*]

Esempio: `int mat[3][4];` array bidimensionale di 3 righe per 4 colonne (ovvero matrice 3x4)

Per ogni dimensione `i` l'indice va da 0 a *lung-i* -1.



Esempio: `int m[11][3][4];`

Accesso agli elementi di una matrice

Esempio:

```
int i, mat[3][4];
…
i = mat[0][0]; /*elemento di riga 0 e colonna 0 (primo elemento)*/
mat[2][3] = 28; /*elemento di riga 2 e colonna 3 (ultimo    elemento)*/
mat[2][1] = mat[0][0] * mat[1][3];
```

Come per i vettori, l'unica operazione possibile sulle matrici è l'accesso agli elementi tramite l'operatore `[]`.

Esempio: Programma che legge due matrici MxN (ad esempio 4x3) e stampa la matrice somma:

```
for (i = 0; i < M; i++)
  for (j = 0; j < N; j++)
    c[i][j] = a[i][j] + b[i][j];
```

Esempio

Esempio: lettura e stampa di una matrice.

```
#include <stdio.h>
#define RIG 2
#define COL 3
int main() {
  int mat[RIG][COL];
  int i, j;
  /* lettura matrice */  printf("Lettura matrice %d x %d;\n", RIG, COL);
  for (i = 0; i < RIG; i++)
    for (j = 0; j < COL; j++)
      scanf("%d", &mat[i][j]);
  /* stampa matrice */
  printf("La matrice e':\n");
  for (i = 0; i < RIG; i++) {
    for (j = 0; j < COL; j++)
      printf("%6d ", mat[i][j]);
    printf("\n");
  }
  return 0;
}
```

Inizializzazione di matrici

Esempio:

```
int mat[2][3] = {{1,2,3}, {4,5,6}};
int mat[2][3] = {1,2,3,4,5,6};
```

1	2	3
4	5	6

Esempio:

```
int mat[2][3] = {{1,2,3}};
int mat[2][3] = {1,2,3};
```

1	2	3
0	0	0

Esempio:

```
int mat[2][3] = {{1}, {2,3}};
```

1	0	0
2	3	0

Esempio

Programma che legge una matrice A (MxP) ed una matrice B (PxN) e calcola e stampa la matrice C prodotto delle due matrici.

La matrice C è di dimensione MxN. Il generico elemento C_{ij} di C è dato da:

$$C_{ij} = \sum_{k=0}^{P-1} A_{ik} B_{kj}$$

```
#include <stdio.h>
#define M 3
#define P 4
#define N 2
int main() {
  int a[M][P];  int b[P][N];  int c[M][N];  int i, j, k;
  /* lettura matrici */
  printf("Lettura prima matrice %dX%d;\n", M, P);
  for (i = 0; i < M; i++)
    for (j = 0; j < P; j++)
      scanf("%d", &a[i][j]);
 printf("Lettura seconda matrice %dX%d;\n", P, N);
  for (i = 0; i < P; i++)
    for (j = 0; j < N; j++)
      scanf("%d", &b[i][j]);
```

Esempio (continua)

```
 /* calcolo prodotto */
for (i = 0; i < M; i++)
    for (j = 0; j < N; j++) {
      c[i][j] = 0;
      for (k = 0; k < P; k++)
        c[i][j] = c[i][j] + a[i][k] * b[k][j];
    }
  /* stampa risultato */
  printf("La matrice prodotto e':\n");
  for (i = 0; i < M; i++) {
    for (j = 0; j < N; j++)
      printf("%d ", c[i][j]);
    printf("\n");
  }
  return 0;
}
```

Esercizio

Programma che legge una matrice A (M x N) e:

- stampa l'elemento massimo con i suoi indici di riga e di colonna;
- costruisce il vettore degli elementi massimi di ogni riga, e lo stampa;
- costruisce il vettore degli elementi massimi di ogni colonna, e lo stampa;
- verifica se la matrice è diagonale (in questo caso M=N) (una matrice si dice diagonale se $A_{ij}=0$ quando $i \neq j$);
- verifica se la matrice è simmetrica (una matrice si dice simmetrica se $A_{ij}=A_{ji}$ per ogni coppia di indici i e j);
- calcola la matrice T (NxM) trasposta di A, e la stampa (il generico elemento T_{ij} della trasposta T di A è pari ad A_{ji}).

Esercizio

Esempio: Programma che legge una matrice A (M x N) e verifica se tutte le somme degli elementi di ogni riga coincidono.

ALGORITMO
prima Ã somma degli elementi della prima riga di A
FOR ogni riga i di A, finchè le somme delle righe sono tutte uguali
DO somma Ã somma degli elementi della riga i di A
IF somma ≠ prima
le somme delle righe sono diverse
stampa se le somme delle righe sono diverse o meno

```
#define M 3
#define N 2
int main(){
  int mat[M][N]; int i, j, prima = 0, somma;
  int diversa = 0; /* booleana: indica riga somma diversa*/
  printf("Inserire gli elementi matrice %dx%d\n", M, N);
  for (i = 0; i < M; i++)
    for (j = 0; j < N; j++)
      scanf("%d", &mat[i][j]);
```

Esercizio

```
for (j = 0; j < N; j++)
    prima += mat[0][j];
  for (i = 1; (i < M && !diversa); i++) {
    somma = 0;
    for (j = 0; j < N; j++)
      somma += mat[i][j];
    if (somma != prima)
      diversa = 1;
  }
  printf("Le somme degli elementi delle righe sono ");
  if (diversa)
    printf("diverse tra loro.\n");
  else
    printf("tutte uguali tra loro.\n");
  return 0;
}
```

Ordinamento per selezione

```
void scambia(int *vet,int i,int j){
  int tmp;
  tmp = vet[i];
  vet[i] = vet[j];
  vet[j] = tmp;
}
void selectMax(int *vet, int dim){
  int i,max,imax=0;
  max = vet[0];
  for(i=0;i<dim; i++)
     if(vet[i]>max) {
        max = vet[i]; imax = i;
     }
  scambia(vet,dim-1,imax);
}
void sort(int *vet, int dim){
  int i;
  for(i=0;i<dim; i++)
     selectMax(vet,dim-i);
}
```

scambia

selezione del max

ordina

Ordinamento a bolle

```
void bubble(int *vet, int n){
  int i,j;
  for ( i=0;i< n-1; i++)
   for (j=n-1;j>i; j--)
    if ( vet[j-1] > vet[j] )
     scambia(vet,j-1,j);
}
```

bubble sort

```
void bubble(int *vet, int n){
  int i,j,ordinato=1;
  for ( i=0;i< n-1 && ordinato; i++){
   ordinato=1;
   for (j=n-1;j>i; j--)
    if ( vet[j-1] > vet[j] ){
     scambia(vet,j-1,j);
     ordinato = 0;
     }
   }
}
```

bubble sort ottimizzato