

```
#include <stdio.h>
#include <ctype.h>
#define MAXPORDA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPORDA]; /* valore di confronto delle frequenze delle parole */
    int i, len, length = 0;
    int f1, f2, f3;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non sono stati inseriti i parametri richiesti nel comando del file\n");
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(freq, MAXPORDA, fp) != NULL)
```

## Programmazione in C

### Unità Matrici - Vettori di stringhe

2

## Matrici – Vettori di stringhe

- » Matrici
- » Definizione di matrici in C
- » Operazioni elementari sulle matrici
- » Vettori di stringhe
- » Esercizi proposti
- » Sommario

### Riferimenti al materiale

#### » Testi

- Kernighan & Ritchie: capitoli 1 e 5
- Cabodi, Quer, Sonza Reorda: capitolo 5
- Dietel & Dietel: capitolo 6

#### » Dispense

- Scheda: "Matrici e Vettori di stringhe in C"

3

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int lung(MAXPARIOLA); /* valore di controllo della frequenza delle lunghezze delle parole */
    int i, indice, lunghezza;
    int RIS = 1;

    if(argc != 2)
    {
        printf("Errore: \"%s\" non è un paragrafo valido il numero del file (*.c)\n", argv[0]);
        exit(1);
    }

    i = fopen(argv[1], "r");
    if(i == NULL)
    {
        printf("Errore: \"%s\" non è un paragrafo valido il numero del file (*.c)\n", argv[1]);
        exit(1);
    }

    while(fgets(lunga, MAXSIGA, i) != NULL)
    {
        lung = strlen(lunga);
        if(lung > lunghezza)
            lunghezza = lung;
    }

    if(lunga == lunghezza)
        RIS = 0;
    else
        RIS = 1;

    printf("%d\n", RIS);
}
```

## Matrici – Vettori di stringhe

## Matrici

5

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAXPARIOLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int lung(MAXPARIOLA); /* valore di controllo della frequenza delle lunghezze delle parole */
    int i, indice, lunghezza;
    int RIS = 1;

    if(argc != 2)
    {
        printf("Errore: \"%s\" non è un paragrafo valido il numero del file (*.c)\n", argv[0]);
        exit(1);
    }

    i = fopen(argv[1], "r");
    if(i == NULL)
    {
        printf("Errore: \"%s\" non è un paragrafo valido il numero del file (*.c)\n", argv[1]);
        exit(1);
    }

    while(fgets(lunga, MAXSIGA, i) != NULL)
    {
        lung = strlen(lunga);
        if(lung > lunghezza)
            lunghezza = lung;
    }

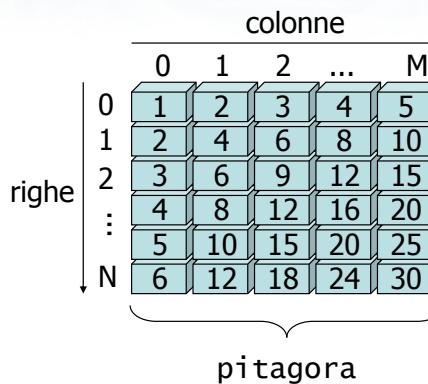
    if(lunga == lunghezza)
        RIS = 0;
    else
        RIS = 1;

    printf("%d\n", RIS);
}
```

## Matrici

## Matrici bidimensionali

### Matrice bidimensionale



8

- » Matrici bidimensionali
- » Matrici come vettori di vettori
- » Matrici pluridimensionali

### Il concetto di matrice

- » La **matrice (array)** è un'estensione logica del concetto di vettore
  - **Vettore = Sequenza uni-dimensionale di valori**
    - Tutti dello stesso tipo
    - Identificati da un indice intero
    - Dimensione fissa
  - **Matrice = Schiera bi- (o n-) dimensionale di valori**
    - Tutti dello stesso tipo
    - Identificati da 2 (o n) indici interi
    - Dimensioni fisse

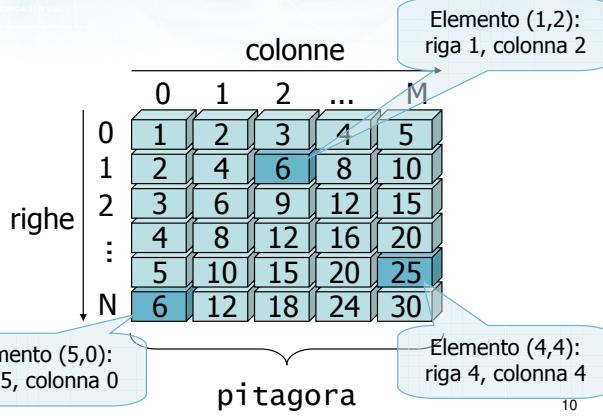
7

### Caratteristiche

- » Una matrice bi-dimensionale è caratterizzata da
  - Nome : **pitagora**
  - Numero di righe : **N**
  - Numero di colonne : **M**
  - Tipo degli elementi : **int**
- » Le righe sono numerate da 0 ad **N-1**
- » Le colonne sono numerate da 0 ad **M-1**
- » In totale ci sono **N×M** elementi
- » In generale **M≠N**; per matrici quadrate, **M=N**

9

## Identificazione degli elementi



## Matrici

## Matrici come vettori di vettori

13

## Lavorare con le matrici

- ▶ Ogni operazione su una matrice deve essere svolta lavorando singolarmente su ciascuno degli elementi
- ▶ Ciò solitamente significa dover ricorrere a due cicli annidati

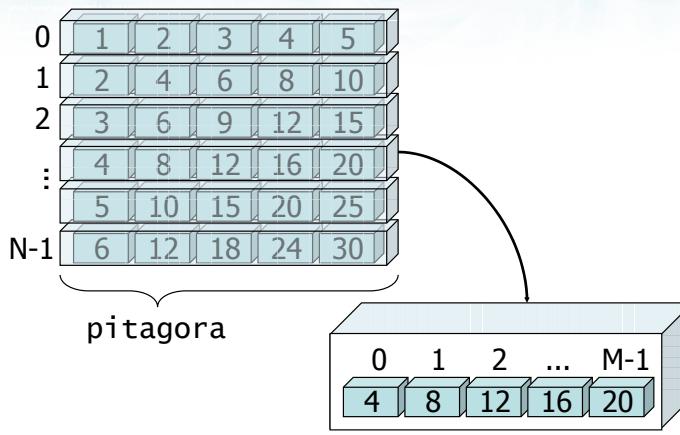
```
for(i=0; i<N; i++) /* righe */
{
    for(j=0; j<M; j++) /* colonne */
    {
        somma = somma + matrice[i][j];
    }
}
```

11

## Vettori di vettori

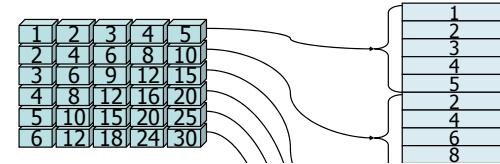
- ▶ Un altro modo di vedere le matrici è concepirle come vettori di vettori:
- Un vettore di N oggetti (righe)
- Ciascun oggetto (riga) è composto da M elementi (colonne)
- ▶ Questa prende il nome di **rappresentazione "per righe"** di una matrice

## Esempio



## Codifica

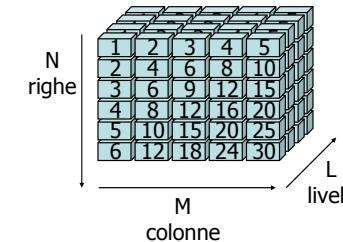
- ▶ Nella realtà, poiché la memoria di un calcolatore è uni-dimensionale, le matrici vengono effettivamente memorizzate "per righe"



15

## Matrici con più dimensioni

- Il concetto di matrice può essere generalizzato anche a più di 2 dimensioni
  - Non vi sono, a priori, limiti sul numero di dimensioni



$N \times M \times L$  elementi:

$$\begin{aligned} \text{elemento}(i, j, k) \\ 0 \leq i \leq N-1 \\ 0 \leq j \leq M-1 \\ 0 \leq k \leq L-1 \end{aligned}$$

17

## Matrici

### Matrici pluridimensionali

## Caratteristiche

- Anche le matrici a più dimensioni condividono i vincoli di base dell'intera famiglia degli array:
  - Tipo di elementi uniforme
  - Dimensioni fissate a priori
  - Indici interi a partire da 0
- In pratica è molto raro utilizzare più di 3 dimensioni

18

## Definizione di matrici in C

- Sintassi della definizione
- Operazioni di accesso

### Matrici – Vettori di stringhe

## Definizione di matrici in C

2

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di costante
                           contiene la lunghezza massima */
    char sig(MAXSIGA);
    int i, j, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("Errore: '%s' non puo' aprire il file %s\n", argv[1]);
        exit(1);
    }

    /* legge ogni riga del file */
    while(fgets(sig, MAXSIGA, f)!=NULL)
    {
        /* controlla se la parola è una parola */
        if(isalpha(sig[0]))
        {
            /* legge ogni parola */
            for(i=0; i<lung; i++)
            {
                /* legge ogni carattere */
                if(sig[i]=='.')
                    break;
                else if(sig[i]==' ' || sig[i]==','
                        || sig[i]==';' || sig[i]==':'
                        || sig[i]==',' || sig[i]==';'
                        || sig[i]==':'))
                {
                    /* salta il carattere */
                    i++;
                    continue;
                }
                /* inserisce il carattere nella parola */
                lunghezza++;
            }
            /* salta il carattere */
            i++;
        }
        /* salta il carattere */
        i++;
    }
}
```

### Definizione di matrici in C

## Sintassi della definizione

4

(Argomenti)

Definizione di matrici in C

## Definizione di matrici in C

**int mat[10][5] ;**



## Definizione di matrici in C

**int mat[10][5] ;**



- **int**
- **float**
- **char**
- In futuro vedremo: **struct**

5

## Definizione di matrici in C

**int mat[10][5] ;**



- Stesse regole che valgono per i nomi di variabili e vettori
- I nomi delle matrici devono essere diversi dai nomi di altre variabili o vettori

6

## Definizione di matrici in C

```
int mat[10][5] ;
```

Tipo di dato base

Nome della matrice

Numero di righe

Numero di colonne

- Interi positivi
- Costanti note a tempo di compilazione
  - const oppure #define
- Uguali o diverse

7

## Esempi

```
► int pitagora[10][10] ;
```

0	1	2	3	4	5	6	7	8	9	10
1	2	4	6	8	10	12	14	16	18	20
2	3	6	9	12	15	18	21	24	27	30
3	4	8	12	16	20	24	28	32	36	40
4	5	10	15	20	25	30	35	40	45	50
5	6	12	18	24	30	36	42	48	54	60
6	7	14	21	28	35	42	49	56	63	70
7	8	16	24	32	40	48	56	64	72	80
8	9	18	27	36	45	54	63	72	81	90
9	10	20	30	40	50	60	70	80	90	100

8

## Esempi

```
► int pitagora[10][10] ;
► char tris[3][3] ;
```

0	1	2
.	.	.
.	.	.

0	1	2
.	X	.
.	.	.

O	X	.
.	X	.
.	.	.

O	X	.
.	X	.
.	.	.

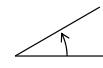
9

## Esempi

```
► int pitagora[10][10] ;
► char tris[3][3] ;
► float rot[2][2] ;
```

0	1
0.707	0.707

-0.707	0.707
0.707	0.707



0.500	0.866
-0.866	0.500



cos β	sin β
-sin β	cos β

0.000	1.000
-1.000	0.000



10

## Matrici a più dimensioni

```
int mat[10][5] ;
```

Più dimensioni

Numero di elementi per ciascuna dimensione

```
float dati[10][5][6] ;
```

11

## Errore frequente

► Dichiarare una matrice usando variabili anziché costanti per le dimensioni

```
int N = 10 ;
int mat[N][N] ;
```

```
int mat[10][10] ;
```

```
const int N = 10 ;
int mat[N][N] ;
```



## Errore frequente

- Dichiarare una matrice usando il nome degli indici

```
int i, j;
int mat[i][j];
```

```
. . .
for(i=0; i<10; i++)
for(j=0; j<10; j++)
scanf("%d",&mat[i][j]);
```

```
const int N = 10 ;
int mat[N][N] ;
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main()
{
    int i,MAXPAROLA; /* veloce di parola */
    char str[MAXPAROLA]; /* stringa che contiene la parola */
    char riga[MAXIGRA];
    int nlinee, lunghezza;
    FILE *f;

    f=fopen("file.txt", "r");
    fgets(riga, MAXIGRA, f);
    lunghezza=strlen(riga);

    if(lunga(>0))
    {
        i=0;
        while(str[i]!='\0')
        {
            if(isalpha(str[i]))
            {
                str[i]=toupper(str[i]);
            }
            else
            {
                str[i]=str[i];
            }
            i++;
        }
    }
    else
    {
        printf("ERRORE: impossibile aprire il file %s", arg1);
        exit(1);
    }

    while(fgets(riga, MAXIGRA, f)>NULL)
    {
        i=0;
        while(str[i]!='\0')
        {
            if(isalpha(str[i]))
            {
                str[i]=toupper(str[i]);
            }
            else
            {
                str[i]=str[i];
            }
            i++;
        }
    }
}
```

## Definizione di matrici in C



## Operazioni di accesso

```
nomematrice[ valoreindice1 ][ valoreindice2 ]
```

Costante, variabile o  
espressione aritmetica  
con valore intero

## Sintassi

```
nomematrice[ valoreindice1 ][ valoreindice2 ]
```

Valore intero compreso  
tra 0 e numero di righe -1

Costante, variabile o  
espressione aritmetica  
con valore intero

Valore intero compreso  
tra 0 e numero di  
colonne -1

## Errore frequente

- Dichiarare una matrice usando il simbolo di "virgola"

```
const int N = 10 ;
const int M = 20 ;
```

```
int mat[N,M] ;
```

```
int mat[N][M] ;
```

## Accesso ai valori di una matrice

- Ciascun elemento di una matrice è una variabile del tipo base
- Per accedere a tale elemento si usa l'operatore di **indicizzazione**: [ ]
- Vi sono tanti indici quante sono le dimensioni della matrice
  - Ogni indice è racchiuso da una coppia di parentesi quadre

## Sintassi

## Esempi

pitagora[1][2]					
0	1	2	3	4	5
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15
3	4	8	12	16	20
4	5	10	15	20	25
5	6	12	18	24	30

```
int pitagora[6][5] ;
```

19

## Vincoli (1/2)

- In una matrice NxMxKx..., il valore dell'indice deve essere compreso tra 0 e N-1/M-1/K-1/....
  - La responsabilità è del programmatore
- Se qualche indice non è un numero intero, viene automaticamente troncato

```
pitagora[i][j] = (i+1)*(j+1) ;  
x = pitagora[1][2] ;
```

20

## Vincoli (2/2)

- Una variabile di tipo matrice può essere utilizzata solamente mediante l'operatore di indicizzazione
  - Occorre agire individualmente sui singoli elementi
  - Non è possibile agire sull'intera matrice in una sola istruzione

```
pitagora[i][j] = (i+1)*(j+1) ;  
x = pitagora[1][2] ;
```

21

## Uso di una cella di un vettore

- L'elemento di una matrice è utilizzabile come una qualsiasi variabile:
  - Utilizzabile all'interno di un'espressione
    - tot = tot + mat[i][j] ;
  - Utilizzabile in istruzioni di assegnazione
    - mat[0][0] = 0 ;
  - Utilizzabile per stampare il valore
    - printf("%d\n", mat[z][k]) ;
  - Utilizzabile per leggere un valore
    - scanf("%d\n", &mat[k][z]) ;

22