Programmazione in C

Unità
Matrici - Vettori di stringhe

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"

Matrici – Vettori di stringhe

Matrici

Matrici

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

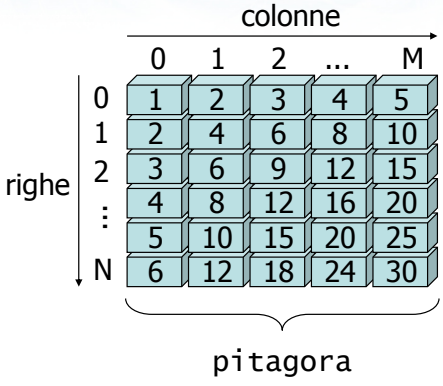
Matrici

Matrici bidimensionali

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

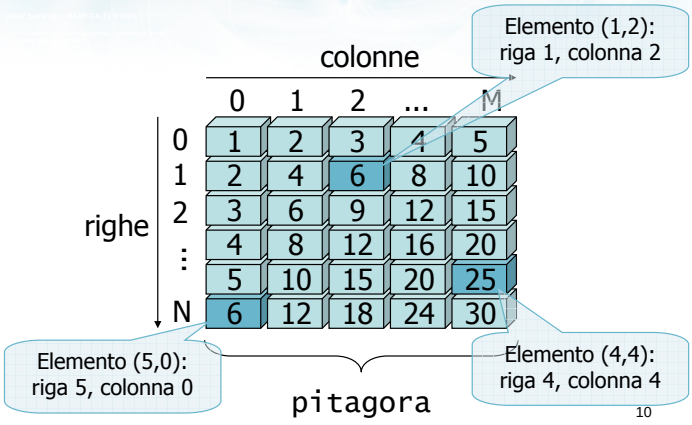
Matrice bidimensionale



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

Identificazione degli elementi



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] ;
    }
}
```

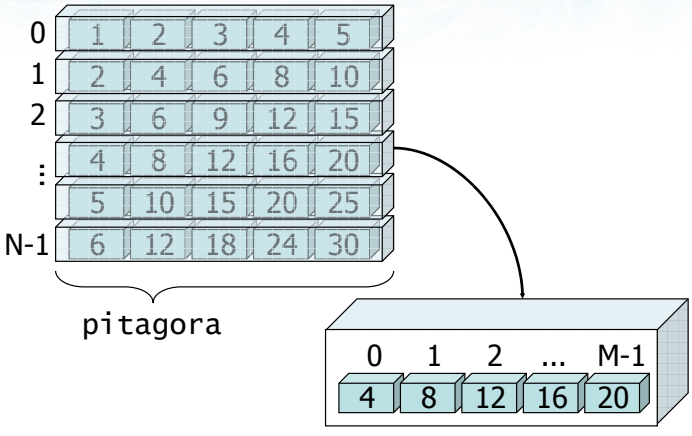

Matrici

Matrici come vettori di vettori

Vettori di vettori

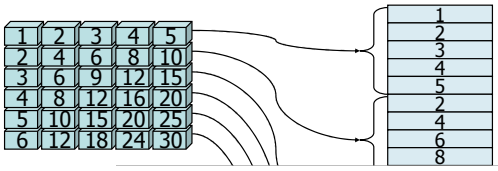
- Un altro modo di vedere le matrici è concepirle come vettori di vettori:
  - Un vettore di N oggetti (righe)
    - Ciascun oggetto (riga) è composto da M elementi (colonne)
- Questa prende il nome di **rappresentazione "per righe"** di una matrice

Esempio



Codifica

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

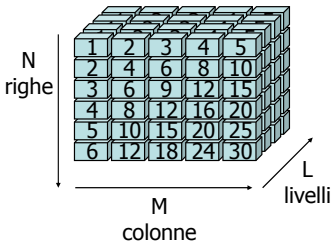


Matrici

Matrici pluridimensionali

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×M×L elementi:

elemento(i, j, k)
$0 \le i \le N-1$
$0 \le j \le M-1$
$0 \le k \le L-1$

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

Matrici – Vettori di stringhe

Definizione di matrici in C

Definizione di matrici in C

- Sintassi della definizione
- Operazioni di accesso

Definizione di matrici in C

Sintassi della definizione

Definizione di matrici in C

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

Tipo di dato base | Nome della matrice | Numero di righe | Numero di colonne

Definizione di matrici in C

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

Tipo di dato base | Nome della matrice | Numero di righe | Numero di colonne

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

Definizione di matrici in C

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

Tipo di dato base | Nome della matrice | Numero di righe | Numero di colonne

- 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

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

Esempi

- `int pitagora[10][10] ;`

	0	1	2	3	4	5	6	7	8	9
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

Esempi

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

	0	1	2
0	.	.	.
1	.	.	.
2	.	.	.

	0	1	2
0	.	.	.
1	.	X	.
2	.	.	.

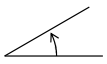
	0	1	2
0	O	.	.
1	.	X	.
2	.	.	.

	0	1	2
0	O	X	.
1	.	X	.
2	.	.	.

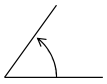
Esempi

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

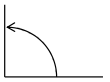
	0	1
0	0.707	0.707
1	-0.707	0.707



	0	1
0	0.500	0.866
1	-0.866	0.500



	0	1
0	0.000	1.000
1	-1.000	0.000



	0	1
0	cos β	sin β
1	-sin β	cos β

Matrici a più dimensioni

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

- Più dimensioni
- Numero di elementi per ciascuna dimensione

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


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] ;
```

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] ;
```


Definizione di matrici in C

Operazioni di accesso

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

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

Costante, variabile o espressione aritmetica con valore intero

Sintassi

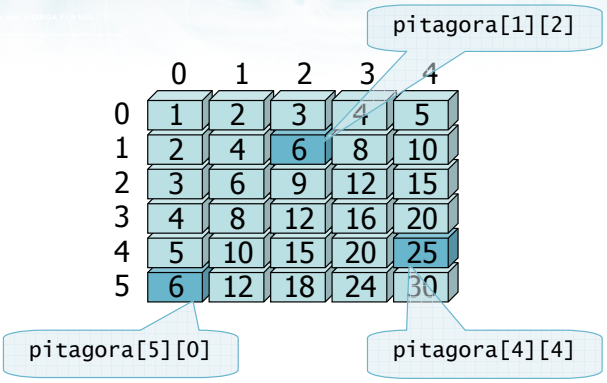
```
nomematrice[ valoreindice1 ][ valoreindice2 ]
```

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

Esempi




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


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] ;
```


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] ;
```


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]) ;`

Matrici – Vettori di stringhe

Operazioni elementari sulle matrici

- Definizioni
- Stampa di una matrice
- Lettura di una matrice
- Copia di una matrice
- Somme di riga o di colonna
- Ricerca di un elemento
- Ricerca del massimo o del minimo

Operazioni elementari sulle matrici

Definizioni

matrici.c

```
const int N = 10 ;
const int M = 5 ; /* dimensioni massime */

float mat[N][M] ; /* matrice 10x5 di reali */
float mat2[N][M] ; /* uguali dimensioni */

float sr[N] ; /* somma per righe */
float sc[M] ; /* somma per colonne */

int i, j ; /* indici dei cicli */
```


matrici.c

```
int trovato ; /* flag */
int riga, col ; /* risultati ricerca */

float dato ; /* elemento da ricercare */

float somma, sommar, sommac ;
    /* per calcolo di somme */

float maxr, maxc ; /* massimi */
```


Operazioni elementari sulle matrici

Stampa di una matrice

Stampa di matrici

- Occorre stampare un elemento per volta, all'interno di cicli `for`
- Sono necessari due cicli annidati
  - Il ciclo esterno per scandire le righe (da 0 a N-1)
  - Il ciclo interno per scandire ciascuna colonna (da 0 a M-1) della riga data
- Si può stampare "per righe" (caso normale) o "per colonne" (trasposta)

Stampa per righe matrice di reali

matrici.c

```
printf("Matrice: %d x %d\n", N, M);

for(i=0; i<N; i++)
{
        Stampa la riga i-esima
}
```


Stampa per righe matrice di reali

matrici.c

```
printf("Matrice: %d x %d\n", N, M);

for(i=0; i<N; i++)
{
    /* Stampa la riga i-esima */
    for(j=0; j<M; j++)
    {
        printf("%f ", mat[i][j]) ;
    }
    printf("\n");
}
```


Esempio

Prompt dei comandi

```
Matrice: 10 righe, 5 colonne
1.000000 0.500000 0.333333 0.250000 0.200000
2.000000 1.000000 0.666667 0.500000 0.400000
3.000000 1.500000 1.000000 0.750000 0.600000
4.000000 2.000000 1.333333 1.000000 0.800000
5.000000 2.500000 1.666667 1.250000 1.000000
6.000000 3.000000 2.000000 1.500000 1.200000
7.000000 3.500000 2.333333 1.750000 1.400000
8.000000 4.000000 2.666667 2.000000 1.600000
9.000000 4.500000 3.000000 2.250000 1.800000
10.000000 5.000000 3.333333 2.500000 2.000000
```


Stampa per colonne matrice di reali

matrici.c

```
printf("Matrice: %d x %d\n", N, M);

for(j=0; j<M; j++)
{
    for(i=0; i<N; i++)
    {
        printf("%f ", mat[i][j]) ;
    }
    printf("\n");
}
```


Esempio

Prompt dei comandi

```
Matrice: 10 righe, 5 colonne
1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00
0.50 1.00 1.50 2.00 2.50 3.00 3.50 4.00 4.50 5.00
0.33 0.67 1.00 1.33 1.67 2.00 2.33 2.67 3.00 3.33
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50
0.20 0.40 0.60 0.80 1.00 1.20 1.40 1.60 1.80 2.00
```

Operazioni elementari sulle matrici

Lettura di una matrice

Lettura di matrici

- Occorre leggere un elemento per volta
- Si procede per righe (o per colonne)
- Si utilizzano solitamente due cicli annidati

Lettura per righe matrice di reali

matrici.c

```
printf("Immetti matrice %d x %d\n",
        N, M) ;

for(i=0; i<N; i++)
{
    printf("Riga %d:\n", i+1) ;
    for(j=0; j<M; j++)
    {
        printf("Elemento (%d,%d): ",
               i+1, j+1) ;
        scanf("%f", &mat[i][j]) ;
    }
}
```


Esempio

```
Prompt dei comandi
Immetti una matrice 10 x 5
Riga 1:
Elemento (1,1): 3.2
Elemento (1,2): 1
Elemento (1,3): -12.4
Elemento (1,4): 2.112
Elemento (1,5): 23
Riga 2:
Elemento (2,1): 23.1
Elemento (2,2): 2.11
Elemento (2,3): .22
Elemento (2,4): 3.14
Elemento (2,5): 2.71
```


Operazioni elementari sulle matrici

Copia di una matrice

Copia di matrici (1/2)

- L'operazione di copia di una matrice "sorgente" in una "destinazione" richiede che ciascun elemento venga copiato individualmente
- La matrice destinazione deve avere dimensioni uguali o superiori a quelle della sorgente
- L'operazione di copia avviene ovviamente a livello del singolo elemento

Copia di matrici (2/2)

matrici.c

```
for(i=0; i<N; i++)
    for(j=0; j<M; j++)
        mat2[i][j] = mat[i][j] ;
```


Operazioni elementari sulle matrici

Somme di riga o di colonna

Sommatorie in matrici

- Il calcolo di totali sui dati contenuti in una matrice può corrispondere a tre diverse operazioni:
  - Somma degli elementi di ciascuna riga (totali di riga)
  - Somma degli elementi di ciascuna colonna (totali di colonna)
  - Somma di tutti gli elementi della matrice

Esempio

`float mat[N][M] ;`

1.00	0.50	0.33	0.25	0.20
2.00	1.00	0.67	0.50	0.40
3.00	1.50	1.00	0.75	0.60
4.00	2.00	1.33	1.00	0.80
5.00	2.50	1.67	1.25	1.00
6.00	3.00	2.00	1.50	1.20
7.00	3.50	2.33	1.75	1.40
8.00	4.00	2.67	2.00	1.60
9.00	4.50	3.00	2.25	1.80
10.00	5.00	3.33	2.50	2.00

Esempio

`float mat[N][M] ;`

`float sr[N] ;`

1.00	0.50	0.33	0.25	0.20	2.28
2.00	1.00	0.67	0.50	0.40	4.56
3.00	1.50	1.00	0.75	0.60	6.85
4.00	2.00	1.33	1.00	0.80	9.13
5.00	2.50	1.67	1.25	1.00	11.41
6.00	3.00	2.00	1.50	1.20	13.70
7.00	3.50	2.33	1.75	1.40	15.98
8.00	4.00	2.67	2.00	1.60	18.26
9.00	4.50	3.00	2.25	1.80	20.55
10.00	5.00	3.33	2.50	2.00	22.83
55.00	27.50	18.33	13.75	11.00	

`float sc[M] ;`

Somma per righe

matrici.c

```
for(i=0 ; i<N ; i++)
{
    somma = 0.0 ;
    for(j=0; j<M; j++)
        somma = somma + mat[i][j] ;
    sr[i] = somma ;
}

for(i=0; i<N; i++)
    printf("Somma riga %d = %f\n",
           i+1, sr[i]) ;
```

Somma per colonne

matrici.c

```
for(j=0 ; j<M ; j++)
{
    somma = 0.0 ;
    for(i=0; i<N; i++)
        somma = somma + mat[i][j] ;
    sc[j] = somma ;
}

for(j=0; j<M; j++)
    printf("Somma colonna %d = %f\n",
           j+1, sc[j]) ;
```


Somma complessiva

matrici.c

```
somma = 0.0 ;
for(i=0 ; i<N ; i++)
{
    for(j=0; j<M; j++)
        somma = somma + mat[i][j] ;
}

printf("Somma complessiva = %f\n",
       somma) ;
```


Operazioni elementari sulle matrici

Ricerca di un elemento

Ricerca di elementi

- Dato un valore dato, ricercare se esso esiste (almeno una volta) nella matrice
- In caso affermativo, specificare la riga e la colonna
- Si utilizzano i soliti due cicli annidati

Ricerca elemento (1/2)

matrici.c

```
printf("Dato da ricercare: ");
scanf("%f", &dato) ;

trovato = 0 ;
riga = -1 ;
col  = -1 ;

for(i=0; i<N && trovato==0; i++)
    for(j=0; j<M && trovato==0; j++)
        if( mat[i][j]==dato )
        {
            trovato=1 ;
            riga = i ;
            col = j ;
        }
```


Ricerca elemento (2/2)

matrici.c

```
if(trovato==1)
  printf("Dato %f presente: (%d,%d)\n",
    dato, riga, col) ;
else
  printf("Dato %f non presente\n",
      dato) ;
```

Operazioni elementari sulle matrici

Ricerca del massimo o del minimo

Massimi e minimi

- Per quanto riguarda il calcolo di massimi e minimi si possono distinguere i 3 scenari:
  - Massimo/minimo degli elementi di ciascuna riga
  - Massimo/minimo degli elementi di ciascuna colonna
  - Massimo/minimo tra tutti gli elementi della matrice

Massimo per righe


```
for(i=0; i<N; i++)
{
    col = 0 ;
    maxr = mat[i][0] ;
    for(j=1; j<M; j++)
        if( mat[i][j] > maxr )
        {
            maxr = mat[i][j] ;
            col = j ;
        }

    printf("Max di riga %d vale %f
       e si trova nella colonna %d\n",
        i+1, maxr, col+1) ;
}
```


Massimo per colonne


```
for(j=0; j<M; j++)
{
    riga = 0 ;
    maxc = mat[0][j] ;
    for(i=1; i<N; i++)
        if( mat[i][j] > maxc )
        {
            maxc = mat[i][j] ;
            riga = i ;
        }

    printf("Max di colonna %d vale %f
       e si trova nella riga %d\n",
        j+1, maxc, riga+1) ;
}
```


Osservazioni

- Le operazioni qui citate sono gli elementi fondamentali dell'elaborazione di matrici
- Nei problemi concreti si osserveranno delle combinazioni di tali operazioni
- Occorre analizzare il problema, scomporlo nei suoi sottoproblemi e combinare opportunamente le varie tecniche

Esercizio "Max Sum Abs"

- Data una matrice NxN, determinare la riga in cui la somma dei valori assoluti degli elementi sia massima

$$r = \max_i \left(\sum_j \left| M_{ij} \right| \right)$$

Soluzione 1

- Inizializza `max`
- Per ogni riga `i`:
  - Calcola la somma `sommar` dei valori assoluti di tale riga
  - Confronta `sommar` con il `max` corrente, ed eventualmente aggiorna il `max`
- Stampa `max`

Soluzione 1

maxsumabs1.c

```
max = -1.0 ;

for(i=0; i<N; i++)
{
    sommar = 0.0 ;
    for(j=0; j<M; j++)
    {
        sommar = sommar +
                 fabs(mat[i][j]) ;
    }

    if(sommar>max)
        max = sommar ;
}
printf("R = %f\n", max) ;
```


Soluzione 2

- Calcola un vettore di appoggio, di N elementi, contenente le sommatorie per ogni riga
- Trova il max all'interno di questo vettore di appoggio
- Soluzione più lunga dal punto di vista del codice, ma più semplice da concepire e realizzare

Soluzione 2 (1/2)

maxsumabs2.c

```
for(i=0; i<N; i++)
{
   sommar = 0.0 ;
   for(j=0; j<M; j++)
   {
       sommar = sommar +
                fabs(mat[i][j]) ;
   }
   sr[i] = sommar ;
    }
```


Soluzione 2 (2/2)

maxsumabs2.c

```
max = -1.0 ;

for(i=0; i<N; i++)
    if(sr[i]>max)
        max = sr[i] ;

printf("R = %f\n", max) ;
```

Matrici – Vettori di stringhe

Vettori di stringhe

Vettori di stringhe

- Matrici di caratteri
- Vettori di stringhe
- I/O di vettori di stringhe

Vettori di stringhe

Matrici di caratteri

Matrici di caratteri

- Nel definire una matrice, è ovviamente possibile usare il tipo base **`char`**
- Permette di memorizzare una tabella NxM di caratteri ASCII
- Ogni posizione `[i][j]` deve contenere **un carattere**
  - Non può essere vuota
  - Non può contenere più di un carattere

```
char tris[3][3] ;
```

	0	1	2
0	O	X	.
1	.	X	.
2	.	.	.

Esercizio "Verifica Sudoku"

- Si realizzi un programma in C che verifichi la corretta soluzione della griglia di un "Sudoku"
- Il programma acquisisce da tastiera la griglia 9x9, in cui ciascun elemento è un carattere tra 1 e 9
- Il programma deve verificare se il Sudoku è stato correttamente risolto

Esempio

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Analisi

- Tutti i valori devono essere singoli caratteri tra '1' e '9'
- Su ciascuna delle 9 righe, non devono esserci valori ripetuti
- Su ciascuna delle 9 colonne, non devono esserci valori ripetuti
- In ciascuno dei 9 blocchi 3x3, non devono esserci valori ripetuti

Soluzione (1/10)

sudoku.c

```
const int N = 9 ;

char sudoku[N][N] ;

int i, j, k ; /* indici dei cicli */
int r1, c1, r2, c2 ;
char ch ;
int err, good ; /* flag */

printf("Verifica Sudoku\n") ;
```


Soluzione (2/10)

sudoku.c

```
for(i=0; i<N; i++)
{
    printf("Riga %d:\n", i+1) ;
    for(j=0; j<N; j++)
    {
        Acquisisci un carattere
        ch tra '1' e '9'
        sudoku[i][j] = ch ;
    }
}
```


Soluzione (2/10)

sudoku.c

```
do {
    printf("  Colonna %d: ", j+1) ;
    ch = getchar() ;
    if( ch<'1' || ch>'9' )
        printf("Errata - ripeti\n") ;

    /* elimina fino fine linea */
    while( getchar()!='\n')
        /*nop*/ ;

} while( ch<'1' || ch>'9') ;
```


Soluzione (3/10)

sudoku.c

```
/* Stampa il tabellone */
for(i=0; i<9; i++)
{
    for(j=0; j<9; j++)
    {
        printf("%c ", sudoku[i][j]) ;

        if(j==2 || j==5)
            printf("  ") ;
    }
    printf("\n");

    if(i==2 || i==5)
        printf("\n") ;
}
```


Soluzione (4/10)

sudoku.c

```
good = 1 ; /* flag generale */

/* Verifica le righe */
for(i=0; i<N; i++)
{
    printf("Riga %d: ", i+1) ;
    err = 0 ;

    /* ricerca duplicati su col. j,k */
    for(j=0; j<N; j++)
        for(k=j+1; k<N; k++)
            if(sudoku[i][j]==
                        sudoku[i][k])
                err = 1 ;
```

Soluzione (5/10)

sudoku.c

```
    if(err==0)
        printf("OK\n");
    else
    {
        printf("Errore!\n");
        good = 0 ;
    }
}
```


Soluzione (6/10)

sudoku.c

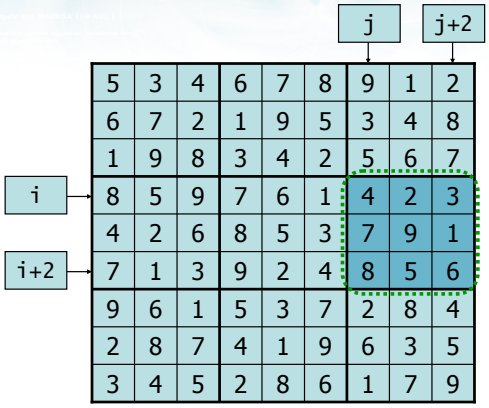
```
for(i=0; i<N; i++) /* Colonne */
{   printf("Colonna %d: ", i+1) ;
    err = 0 ;
    /* ricerca dupl. su righe j,k */
    for(j=0; j<N; j++)
      for(k=j+1; k<N; k++)
        if(sudoku[j][i]==sudoku[k][i])
          err = 1 ;

    if(err==0) printf("OK\n");
    else
    {   printf("Errore!\n");
        good = 0 ;
    }
}
```

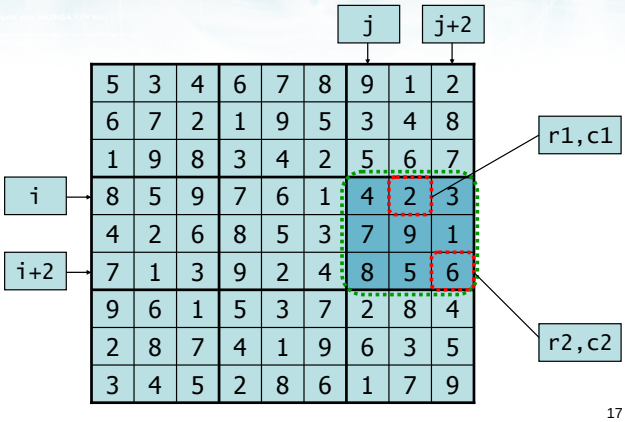

Ricerca per blocchi

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Ricerca per blocchi



Ricerca per blocchi



Soluzione (7/10)

sudoku.c

```
for(i=0; i<N; i=i+3)
{
  for(j=0; j<N; j=j+3)
  {
    printf("Blocco (%d,%d)-(%d,%d): ",
         i+1, j+1, i+3, j+3) ;

    /* ricerca duplicati nel blocco */
    /* Confronta [r1][c1]
                 con [r2][c2] */

    err = 0 ;
```

Soluzione (8/10)

sudoku.c

```
for(r1=i; r1<i+3; r1++)
for(c1=j; c1<j+3; c1++)
{   /* elemento [r1][c1]... */
    for(r2=i; r2<i+3; r2++)
    for(c2=j; c2<j+3; c2++)
    {
        /* ..rispetto a [r2][c2] */
        if( ((r1!=r2)||(c1!=c2)) &&
            sudoku[r1][c1]==
                    sudoku[r2][c2] )
        {
            err = 1 ;
        }
    } /*r2,c2*/
} /*r1,c1*/
```


Soluzione (9/10)

sudoku.c

```
        if(err==0)
            printf("OK\n");
        else
        {
            printf("Errore!\n");
            good = 0 ;
        }
    } /*j*/
} /*i*/
```


Soluzione (10/10)

sudoku.c

```
if(good==1)
    printf("Sudoku valido!\n");
else
    printf("Sudoku errato...\n");
```


Vettori di stringhe

Vettori di stringhe

Vettori di stringhe

- Una matrice di caratteri può anche essere vista come:
  - Un vettore di vettori di caratteri, cioè
  - Un vettore di stringhe
- Si tratta di un metodo diverso di interpretare la stessa struttura dati

Esempio

```
char tris[3][3] ;
```

	0	1	2
0	O	X	.
1	.	X	.
2	.	.	.

```
char nomi[5][10] ;
```

	0	1	2	3	4	5	6	7	8	9
0	F	u	l	v	i	o	\0	x	!	w
1	A	n	t	o	n	i	o	\0	.	z
2	C	r	i	s	t	i	n	a	\0	u
3	E	l	e	n	a	\0	5	g	r	d
4	D	a	v	i	d	e	\0	$	2	e

Caratteristiche

5 stringhe di lunghezza variabile

Lunghezza max 9 caratteri

Terminatore nullo in ogni stringa (riga)

```
char nomi[5][10] ;
```

	0	1	2	3	4	5	6	7	8	9
0	F	u	l	v	i	o	\0	x	!	w
1	A	n	t	o	n	i	o	\0	.	z
2	C	r	i	s	t	i	n	a	\0	u
3	E	l	e	n	a	\0	5	g	r	d
4	D	a	v	i	d	e	\0	$	2	e

Sintassi

- Definizione

```
char vett[MAX][LUN] ;
```

Nome del vettore di stringhe

Numero di stringhe

Lunghezzza massima di ciascuna stringa (compreso terminatore nullo)

Sintassi

- Definizione

```
char vett[MAX][LUN] ;
```

- Accesso al singolo carattere

```
vett[i][j]
```

Carattere (j+1)-esimo della stringa (i+1)-esima, da usarsi per l'elaborazione carattere-per-carattere

Sintassi

- Definizione

```
char vett[MAX][LUN] ;
```

- Accesso

Stringa (i+1)-esima, da usarsi con le funzioni di libreria

- Accesso all'intera stringa

```
vett[i]
```


Esempio 1

- Dato un vettore di stringhe, determinare quante volte è presente la lettera 'A' (maiuscola o minuscola)

```
cont = 0 ;
for(i=0; i<MAX; i++)
{
   for(j=0; vett[i][j]!=0; j++)
   {
      if(toupper(vett[i][j])=='A')
         cont++ ;
   }
}
```


Esempio 2

- Dato un vettore di stringhe, determinare se esistono due stringhe identiche

```
uguali = 0 ;
for(i=0; i<MAX; i++)
{
   for(k=i+1; k<MAX; k++)
   {
      if(strcmp(vett[i], vett[k])==0)
         uguali=1 ;
   }
}
```

Occupazione variabile

- In un vettore di stringhe, ogni riga (ogni stringa) è intrinsecamente un vettore di caratteri ad occupazione variabile
  - Terminatore nullo per indicare la lunghezza effettiva
- Il numero di stringhe effettivamente memorizzato potrebbe non riempire l'intero vettore
  - Variabile intera che indica l'effettiva occupazione del vettore

Esempio

```
const int MAX = 5 ;
const int LUN = 9 ;

char nomi[MAX][LUN+1] ;
int N ;
```

N=3

	0	1	2	3	4	5	6	7	8	9
0	F	u	l	v	i	o	\0	x	!	w
1	A	n	t	o	n	i	o	\0	.	z
2	C	r	i	s	t	i	n	a	\0	u
3	e	4	1	)	a	\0	5	g	r	d
4	1	%	<	d	d	e	g	$	2	e

Errore frequente

- Confondere una stringa (vettore di caratteri) con un vettore di stringhe (matrice di caratteri)

```
char s[LUN+1] ;
```

- s[i] è un singolo carattere
- s è l'intera stringa

```
char v[MAX][LUN+1] ;
```

- v[i][j] è il singolo carattere
- v[i] è un'intera stringa
- v è l'intera matrice

Vettori di stringhe

I/O di vettori di stringhe

Stampa (1/2)

- La stampa del contenuto di un vettore di stringhe si ottiene semplicemente stampando ciascuno degli elementi
  - Si può utilizzare `puts` oppure `printf`

Stampa (2/2)

vettstr.c

```
for(i=0; i<N; i++)
{
    puts(vett[i]) ;
}
```

Lettura

- Acquisire da tastiera un vettore di stringhe
- Un ciclo per ciascuna delle stringhe da leggere
  - Lunghezza nota a priori
  - Lunghezza determinata dalla lettura di un certo dato (es.: "FINE")
- Acquisizione, nel ciclo, di ciascuna delle stringhe
  - Utilizzo della funzione `gets`
  - Eventualmente, lettura in una stringa d'appoggio per la verifica di correttezza, prima di ricopiare nel vettore destinazione

Dimensione nota a priori (1/2)

vettstr.c

```
char vett[MAX][LUN+1] ;
char s[250] ;
. . .

do {
    printf("Quante stringhe? ") ;
    gets(s) ;
    N = atoi(s) ;
    if(N<1 || N>MAX)
        printf("Valore errato: deve
          essere tra 1 e %d\n", MAX) ;
} while(N<1 || N>MAX) ;
```


Dimensione nota a priori (2/2)

vettstr.c

```
for(i=0; i<N; i++)
{   printf("Stringa %d: ", i+1) ;
    gets(s) ;
    if (strlen(s)==0)
    {   printf("Vuota - ripeti\n");
        i-- ;
    }
    else if(strlen(s)>LUN)
    {   printf("Troppo lunga -
                max %d chr\n", LUN) ;
        i-- ;
    }
    else
        strcpy(vett[i], s) ;
}
```


Lettura terminata da "FINE"

vettstr.c

```
N = 0 ;
end = 0 ;
do {
    printf("Stringa %d: ", N+1) ;
    gets(s) ;
    if (strlen(s)==0)
        printf("Vuota - ripeti\n");
    else if(strlen(s)>LUN)
        printf("Troppo lunga\n") ;
    else if(strcmp(s, "FINE")==0)
        end = 1 ;
    else
    {   strcpy(vett[N], s) ;
        N++ ;
    }
} while(end==0) ;
```

Matrici – Vettori di stringhe

Esercizi proposti

Esercizi proposti

- Esercizio "Statistiche testo"
- Esercizio "Magazzino"

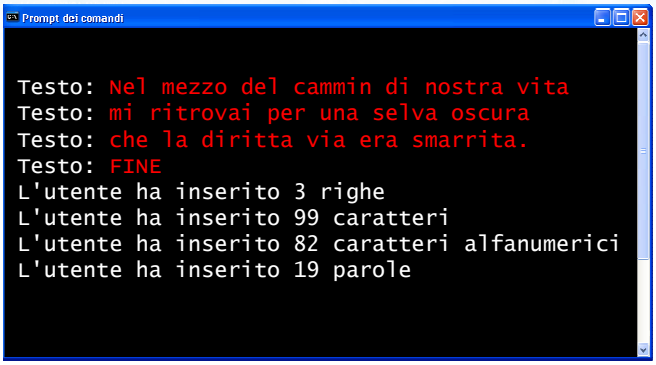
Esercizi proposti

Esercizio "Statistiche testo"

Esercizio "Statistiche testo"

- Un utente inserisce una serie di frasi da tastiera, su più righe
- L'inserimento termina quando l'utente inserisce la parola FINE su una riga da sola
- Il programma deve determinare:
  - 1) Quante righe sono state inserite dall'utente
  - 2) Quanti caratteri sono stati inseriti
  - 3) Quanti caratteri alfanumerici sono stati inseriti
  - 4) Quante parole sono state inserite

Analisi



Soluzione (1/5)


```
const int MAXRIGHE = 2000 ;
const int LUN = 80 ;

char testo[MAXRIGHE][LUN] ;
int Nrighe ; /* righe inserite */
char riga[LUN*10] ;
int i, j ;
int caratteri, caralfa, parole ;
```

Soluzione (2/5)

statistichetesto.c

```
Nrighe = 0 ;
do {
   printf("Testo: ");
   gets(riga) ;

   if( strcmp(riga, "FINE")!=0 )
   {
      /*copia riga in testo[Nrighe] ;*/
      strcpy( testo[Nrighe] , riga ) ;

      Nrighe++ ;
   }
} while( strcmp(riga, "FINE")!=0 ) ;
```


Soluzione (3/5)

statistichetesto.c

```
printf("L'utente ha inserito"
       " %d righe\n", Nrighe);

caratteri = 0 ;
for(i=0; i<Nrighe; i++)
    caratteri = caratteri +
                strlen(testo[i]) ;

printf("L'utente ha inserito"
       " %d caratteri\n", caratteri) ;
```


Soluzione (4/5)

statistichetesto.c

```
caralfa = 0 ;
for(i=0; i<Nrighe; i++)
{
    for(j=0; testo[i][j]!=0; j++)
    {
        if( isalnum(testo[i][j] ) )
            caralfa++ ;
    }
}

printf("L'utente ha inserito "
       "%d caratteri alfanumerici\n",
       caralfa) ;
```


Soluzione (5/5)

statistichetesto.c

```
parole = 0 ;
for(j=0; j<Nrighe; j++)
{
   for(i=0; testo[j][i]!=0; i++)
   {
      if( isalpha(testo[j][i]) &&
      (i==0 || !isalpha(testo[j][i-1])))
      {
         parole ++ ;
      }
   }
}

printf("L'utente ha inserito "
       "%d parole\n", parole) ;
```


Esercizi proposti

Esercizio "Magazzino"

Esercizio "Magazzino" (1/2)

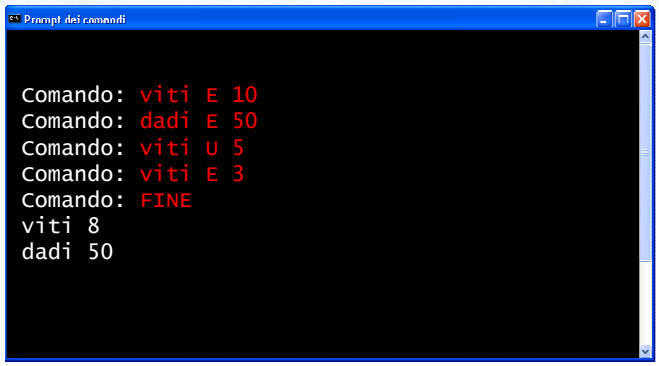
- Un'azienda deve tenere traccia dei beni presenti in un magazzino
- L'utente inserisce da tastiera dei "comandi" nel seguente formato:
  - `bene EU quantità`

  dove:
  - `bene` è il nome di un bene
  - `EU` è la lettera `'E'` per entrata, `'U'` per uscita
  - `quantità` è la quantità di bene entrata o uscita

Esercizio "Magazzino" (2/2)

- L'utente termina il caricamento inserendo un comando pari a "FINE". In tal caso il programma deve stampare le quantità di beni presenti a magazzino

Analisi



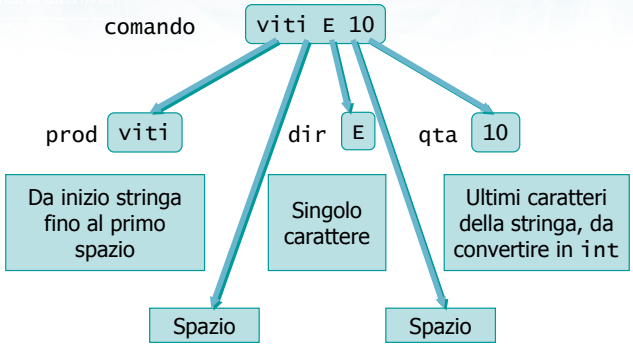
Problemi

- Estrazione dei 3 parametri dal comando immesso dall'utente
  - Nome prodotto: stringa
  - Direzione: carattere
  - Quantità: intero

`viti E 10`

- Memorizzazione di nomi di prodotti e quantità relative
  - Vettori paralleli
- Aggiornamento delle giacenze

Stringa di comando



Analisi del comando (1/2)

magazzino.c

```
i = 0 ;
while(comando[i]!=' ')
{
    prod[i] = comando[i] ;
    i++ ;
}
prod[i] = 0 ;

/* salta lo spazio */
i++ ;

dir = comando[i] ; /* 'E' o 'U' */
```


Analisi del comando (2/2)

magazzino.c

```
/* salta lo spazio */
i++ ;

/* da qui a fine: la quantità */
j = 0 ;
while(comando[i]!=0)
{
    temp[j] = comando[i] ;
    i++ ;
    j++ ;
}
temp[j] = 0 ;
qta = atoi(temp) ;
```

Osservazione

- L'analisi di una stringa di comando composta da più campi è sempre un'operazione complessa
- Nel caso in cui si dovessero gestire delle condizioni anomale (es. più spazi consecutivi) o di errore (es. manca la quantità), il codice diverrebbe estremamente articolato
- Vedremo più avanti la funzione `sscanf` che può aiutare in questi casi

Rappresentazione del magazzino

- Occorre memorizzare, per ciascun prodotto
  - Il nome
  - La quantità corrente
- Si possono usare due vettori "paralleli"

```
char prodotti[MAX][LUN+1] ;
int quantita[MAX] ;

int N ; /* occupazione effettiva
           vettori prodotti e quantita */
```


Inserimento di un prodotto

- Determinare se il prodotto è già in magazzino
  - Ricerca del nome del prodotto nel vettore `prodotto`
- Se c'è già, incrementa la quantità
- Se non c'è ancora, aggiungi una riga ai vettori

Inserimento di un prodotto


```
/* trova la posizione del prodotto */
trovato = -1 ;
for(i=0; i<N; i++)
    if(strcmp(prodotti[i], prod)==0)
        trovato = i ;

if( trovato != -1 ) /* esiste già */
    quantita[trovato] =
        quantita[trovato] + qta ;

else /* prodotto nuovo */
{   strcpy(prodotti[N], prod) ;
    quantita[N] = qta ;
    N++ ;
}
```


Eliminazione di un prodotto

- Determinare se il prodotto è già in magazzino
  - Ricerca del nome del prodotto nel vettore `prodotto`
- Se c'è già, decrementa la quantità
- Se non c'è ancora, errore

Eliminazione di un prodotto


```
/* trova la posizione del prodotto */
trovato = -1 ;
for(i=0; i<N; i++)
    if(strcmp(prodotti[i], prod)==0)
        trovato = i ;

if( trovato == -1 )
    printf("Prodotto %s non "
     "trovato in magazzino\n", prod);
else
    quantita[trovato] =
        quantita[trovato] - qta ;
```

Matrici – Vettori di stringhe

Sommario

Argomenti trattati

- Matrici bi-dimensionali e pluri-dimensionali
- Matrici di numeri interi e reali
  - Definizione
  - Operazioni frequenti
- Matrici di caratteri
- Vettori di stringhe
  - Caso particolare di matrici di caratteri
  - Operazioni frequenti

Tecniche di programmazione

- Usare matrici per memorizzare schiere di dati numerici
- Usare vettori di stringhe per memorizzare stringhe di testo di lunghezza variabile
- Compiere operazioni di ricerca nei vettori di stringhe

Materiale aggiuntivo

- Sul CD-ROM
  - Testi e soluzioni degli esercizi trattati nei lucidi
  - Scheda sintetica
  - Esercizi risolti
  - Esercizi proposti
- Esercizi proposti da altri libri di testo