

Fondamenti di Informatica T-1

modulo 2

Laboratorio 05: ***ancora array***
(matrici e ordinamento)

Contenuti

- Matrici
- Algoritmi di ordinamento

Esempio 1: Matrici e rappresentazione

Stampa di una matrice rettangolare con cambio di dimensioni

- Vogliamo stampare una matrice 4 x 6 come se fosse una matrice 3 x 8
 - Nota: le matrici hanno lo stesso numero di elementi

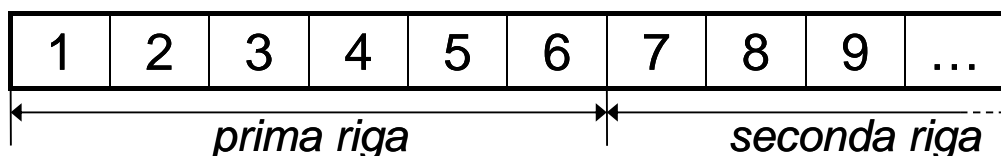
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24

3

Esempio 1 - Considerazioni

- Possibili soluzioni
 - Matrice come ADT
 - Effettuo esplicitamente la copia della matrice di partenza (4x6) in quella di arrivo (3x8)
 - Stampo la seconda matrice
 - Matrice “a basso livello”
 - La matrice è in realtà rappresentata come un vettore, in cui il numero di colonne permette di identificare quando “si passa” da una riga all'altra

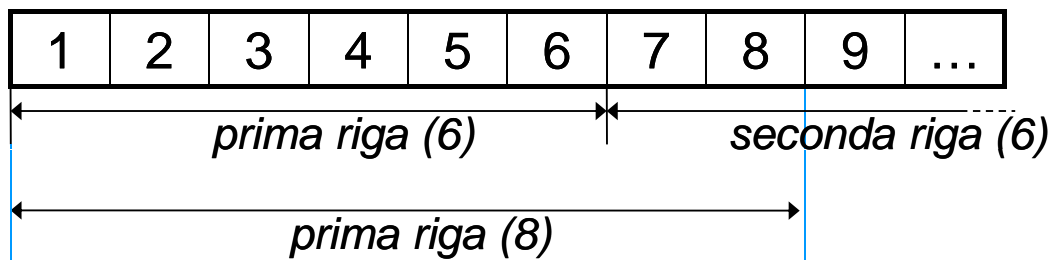


4

Esempio 1 - Approccio scelto

Seguendo il secondo approccio, nessuno ci vieta di considerare un diverso valore per le colonne

- Siamo però costretti a effettuare **esplicitamente il calcolo sul puntatore**



5

Esempio 1 - Soluzione (1)

Matrice come
vettore!

```
void customPrint(int* matrice, int righe, int colonne)
{
    int i,j;
    for(i = 0; i < righe; i++)
    {
        for(j = 0; j < colonne; j++)
            printf("%d\t",matrice[i * colonne + j]);
        printf("\n");
    }
}
```

6

Esempio 1 - Soluzione (2)

```
#define RIGHE 4
#define COLONNE 6
#define RIGHE2 3
#define COLONNE2 8
int main ()
{
    int matrice[RIGHE][COLONNE];
    int i,j, value;
    value = 0;
    for(i = 0; i < RIGHE; i++)
    {
        for(j = 0; j < COLONNE; j++)
            matrice[i][j] = value++;
    }
    customPrint(&matrice[0][0], RIGHE, COLONNE);
    customPrint(&matrice[0][0], RIGHE2, COLONNE2);
}
```

7

Esercizio 1: Quadrato Magico

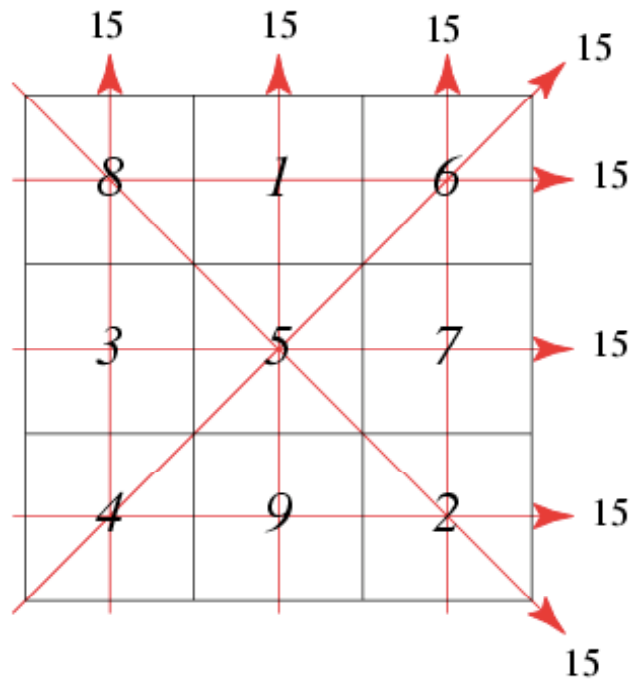
Magic Square

- Realizzare una funzione che, presa in input una matrice quadrata, determini **se è un quadrato magico**
- Un quadrato magico è una matrice NxN
 - I cui elementi sono TUTTI i numeri interi da 1 a N^2
 - Le somme degli elementi per tutte le righe, tutte le colonne e le diagonali sono uguali
 - Tale somma è detta “magic constant”

8

Esercizio 1 - Esempio

Esempio di quadrato magico



9

Esercizio 1 - Schema

Linee guida

- Ragionare sempre a livelli di astrazione e decomporre la funzione in sotto-funzioni
- **Quattro sotto-funzioni**
 1. Verifica che la matrice sia ben formata
 2. Verifica che la somma di ogni riga sia equivalente (e in caso affermativo, restituisce tale valore)
 3. Come 2, ma sulle colonne
 - NOTA: il calcolo è molto simile a quello del punto 2
 4. Come 2, ma sulle diagonali
- Una funzione che, data in input la matrice e la sua dimensione, invoca opportunamente le sotto-funzioni e restituisce dei codici differenziati
 - Nel caso in cui la matrice sia un quadrato magico, restituisce anche la "magic constant"

10

Esercizio 2: Algoritmi di Ordinamento

“Astrazione” degli algoritmi di ordinamento

- Implementare i diversi algoritmi di ordinamento, facendo in modo di **astrarre completamente dal tipo** degli elementi del vettore
- Fare anche in modo che vengano stampate delle **statistiche sul numero di confronti e di scambi** effettuati
- Validare la soluzione su un vettore di interi, un vettore di caratteri, un vettore di stringhe

11

Esercizio 2 - Considerazioni

- Quali sono le istruzioni utilizzate in fase di ordinamento che dipendono dal TIPO dell'elemento?
 - Confronto tra due elementi
 - Assegnamento di un elemento a un altro elemento
 - Swap?
 - dipende dal tipo a causa degli assegnamenti effettuati
 - quindi ci riconduciamo al caso precedente

12

Esercizio 2 - Idea di base

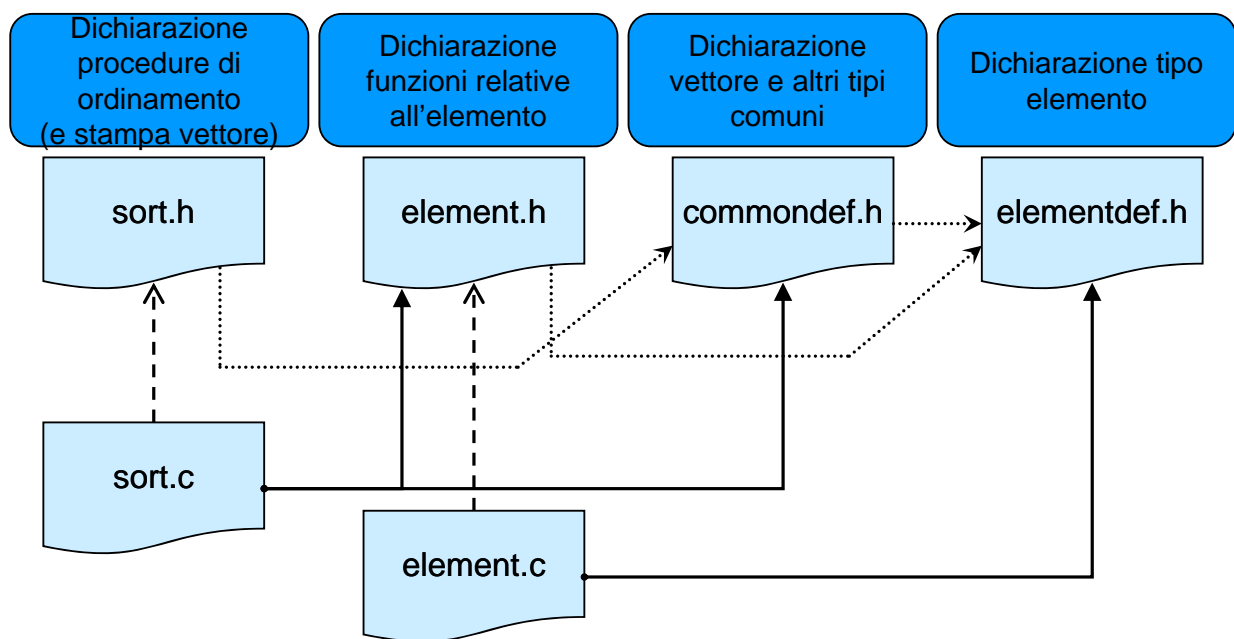
- Quindi dobbiamo sostituire
 - Confronti
 - Assegnamenti
- ...con delle funzioni capaci di eseguire il confronto e l'assegnamento

```
int compare(Element e1, Element e2);
```

```
void assign(Element *lvalue, Element rvalue);
```

13

Esercizio 2 - Moduli



.....> usa ---> implementa —> include

14

Esercizio 2 - Elemento

- **elementdef.h**
 - Contiene la dichiarazione `typedef ... Element;`
- **element.h**
 - Contiene le dichiarazioni delle funzioni per manipolare un elemento
- **Quindi se cambio tipo devo aggiornare unicamente**
 - **elementdef.h**
 - **element.c** (l'header rimane uguale, cambia l'implementazione in base al tipo)

15

Esercizio 2 - Manipolazione degli elementi

Contenuto di `element.h`

- **`int compare(Element e1, Element e2);`**
 - Restituisce un numero negativo se `e1 < e2`, 0 se `e1 == e2`, un numero positivo se `e1 > e2`
- **`void swap(Element *e1, Element *e2);`**
 - Scambio elementi (utilizzando `assign!!!`)
- **`void assign(Element *lvalue, Element rvalue);`**
 - Assegna il contenuto di `rvalue` a `lvalue`
- **`void printElement(Element e);`**
 - Stampa l'elemento a video
- **`void printStatistics();`**
 - Stampa le statistiche relative a confronti e scambi
 - Suggerimento: utilizzare due variabili contatore globali

16