Matrici e Array Multidimensionali

Informatica A AA 2023 / 2024

Giacomo Boracchi

11 Ottobre 2023

giacomo.boracchi@polimi.it

Array Multidimensionali

È possibile definire array con più di una dimensione

- Avremo un **insieme** di variabili **omogenee** ed **indicizzate**

Sintassi dichiarazione di una matrice (array 2D) mediante il costruttore array

```
tipo nomeArray[dim1][dim2];
```

- `tipo` la keyword di un tipo (built in o user-defined)
- **`nomeArray`** è il nome della variabile
- **`dim1`** e **`dim2`** sono **numeri** che stabiliscono il valore massimo del primo e del secondo indice rispettivamente

Array a più dimensioni

Gli array a 1D realizzano i vettori, quelli a 2D realizzano le matrici,

Dichiarazione:

```
int A[20][30];
```

A è una matrice di 20 × 30 elementi interi (600 variabili distinte)

```
float F[20][20][30];
```

F è una matrice 3D di 20×20×30 variabili di tipo float (12.000!)

Oppure a quattro dimensioni, ecc…

Esempio Acquisizione di una Matrice

```
int M[10][10];


for(i = 0; i < r; i++)
      for(j = 0; j < c ; j++)
      {
            printf("Inserire el. pos. [%d][%d]", i+1, j+1);
            scanf("%d", &M[i][j]);
      }
```

Stampa di una Matrice

```
int M[10][10];


for(i = 0; i < r; i++)
      {
            for(j = 0; j < c ; j++)
                  printf("%5d" , M[i][j]);
            printf("\n");
      }
```

Excursus Matematico – Le Matrici Quadrate

Una matrice si dice **quadrata** se ha lo stesso numero di righe e colonne.

Una matrice quadrata ha una **diagonale principale**, quella formata da tutti gli elementi in posizione (i, i), con indici uguali. La somma di questi elementi è chiamata **traccia** della matrice. L'operazione di **trasposizione** trasforma una matrice quadrata A nella matrice A^t ottenuta scambiando ogni $A(i, j)$ con $A(j, i)$, in altre parole ribaltando la matrice intorno alla sua diagonale principale.

Una matrice tale che $A(i, j) = A(j, i)$ per ogni i, j è una **matrice simmetrica**. In altre parole, A è simmetrica se $A = A^t$. Se tutti gli elementi che non stanno nella **diagonale** principale sono nulli, la matrice è detta diagonale.

TODO: Esercizio su Matrici

Si scriva un programma che prende in ingresso una matrice quadrata e controlla che sia simmetrica.

Una matrice è simmetrica se per ogni elemento vale la seguente proprietà: L'elemento alla riga i, colonna j coincide con l'elemento alla riga j colonna i

Es di matrice simmetrica

1	12	1
12	0	3
1	3	23

```
/* Frammento di programma per verificare se una matrice è simmetrica */

int MatQuadra[N][N], j, i=0;
int sim = 1;            /* sim inizialmente vale 1 */

   /* ... inserimento dati nella matrice ... codice omesso ... */

while ( i < N ) {
    j = 0;
    while ( j < N ) {
        if ( MatQuadra[i][j]  !=  MatQuadra[j][i] )
            sim = 0;  /*Alla fine sim varrà 0 se e solo se non è simmetrica*/
        ++j;
    }
    ++i;
}
```

è corretto?

si può migliorare?

È corretto, ma…

Confronta MatQuadra[i][j] con MatQuadra[j][i] e, poi, anche MatQuadra[j][i] con MatQuadra[i][j]

Confronta MatQuadra[i][i] con se stesso

Se trova una dissimmetria, continua con la verifica, ma è inutile

Esercizio: se ne scriva una versione “ottimizzata” che risolva i tre problemi

```
int MatQuadra[N] [N], j, i=0;
int sim = 1;            /* sim inizialmente vale 1 */

   /* ... inserimento dati nella matrice ... codice omesso ... */

while ( i < N && sim ) {
    j = 0;
    while ( j < i && sim )   {
        if ( MatQuadra[i] [j]  !=  MatQuadra[j] [i] )
            sim = 0;  /*Alla fine sim varrà 0 se e solo se non è simmetrica*/
        ++j;
    }
    ++i;
}
```

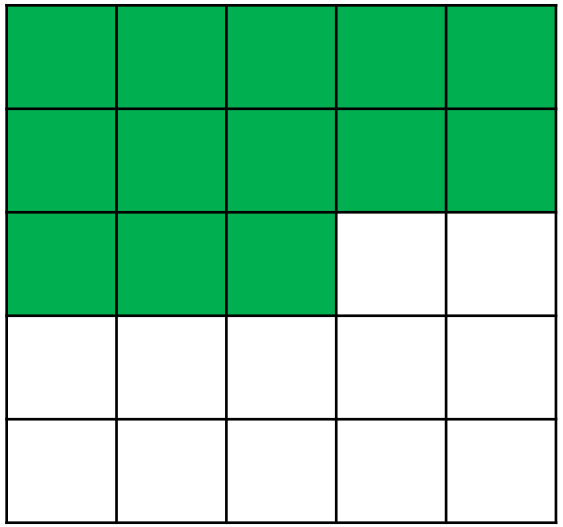
ORA È OK

Esercizio

Scrivere un programma che esegue un inserimento controllato di una matrice ed in particolare controlla che il valore corrente non sia già stato inserito dall'utente in precedenza

Hint: Si consideri come viene riempita la matrice. Tipicamente l'inserimento avviene per righe, quindi occorre controllare interamente le righe precedenti e la riga corrente fino alla colonna specificata.

Hint: Se si controlla al di fuori degli elementi popolati o delle dimensioni effettive della matrice, il risultato potrebbe essere errato.



```c
#include <stdio.h>
#define N 2
int main(){
    int i,j,k,t,cont,A[N][N],howMany=0,ok=1;
    for(i=0;i<N;i++) {
        for(j=0;j<N;j++) {
            do {
                ok=1;
                printf("Inserire un valore\n"); scanf("%d", &A[i][j]);

                // controlla le righe precedenti
                for (k=0; k<i && ok; k++)
                    for (t=0; t<N && ok; t++)
                        if (A[i][j] == A[k][t]) {
                            ok=0;
                            printf("duplicato\n");}
                // controlla la riga corrente
                for (t = 0; t < j && ok; t++)
                    if(A[i][j] == A[i][t]){
                        ok=0;
                        printf("duplicato nella stessa riga\n");
                    }
            }while (ok==0);
        }
    // stampa (vedi slide seguente)
    return 0;
}
```

```
// stampa della matrice
    printf("\n");
    for(i=0;i<N;i++) {
        for(j=0;j<N;j++)
            printf("%5d", A[i][j]);
        printf("\n");
    }
```

Esercizio

Scrivere un programma che chiede all’utente di inserire una matrice 20x30, poi (dopo aver terminato la fase di inserimento) copia gli elementi dispari in una seconda matrice 20x30 senza lasciare buchi, se non in fondo.

Gli elementi in fondo (i "buchi") siano messi a zero.

TODO: Esercizio su Matrici

Scrivere un programma che richiede l'inserimento di una matrice di interi **`M`** e di un intero **`n`**.

Il programma conta il numero di occorrenze di **`n`** in ogni riga di **`M`**.

Il programma stampa con un istogramma (verticale) il numero di occorrenze di **`n`** per ogni riga di **`M`**.

Array Multidimensionali

Osservazioni

Array di Array

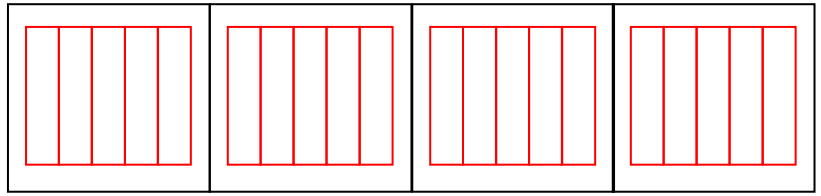
Gli elementi degli array possono essere di qualsiasi tipo (predefinito, definito dall'utente, semplice o strutturato)

⇒ **è possibile costruire "array di array"**

Esempio:

```
typedef   int Vettore[5];
```

```
typedef   Vettore Matrice4Per5[4];
```



```
Matrice4Per5 matrice1;
```

Array di Array

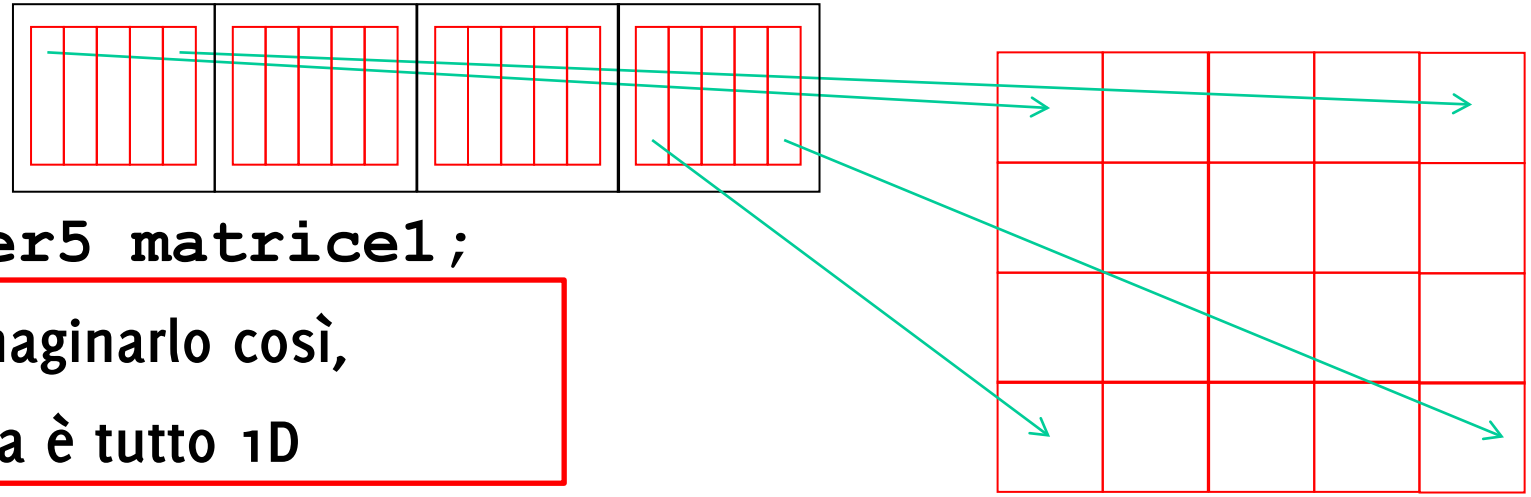
Gli elementi degli array possono essere di qualsiasi tipo (predefinito, definito dall'utente, semplice o strutturato)

⇒ **è possibile costruire "array di array"**

Esempio:

```
typedef   int Vettore[5];
```

```
typedef   Vettore Matrice4Per5[4];
```



```
Matrice4Per5 matrice1;
```

Possiamo immaginarlo così,

ma in memoria è tutto 1D

Mappa di Memorizzazione di un array 2D

Cioè la rappresentazione in memoria di un array a 2 dimensioni:

- array memorizzato riga per riga, per indice di riga crescente e, all'interno di ogni riga, per indice di colonna crescente.

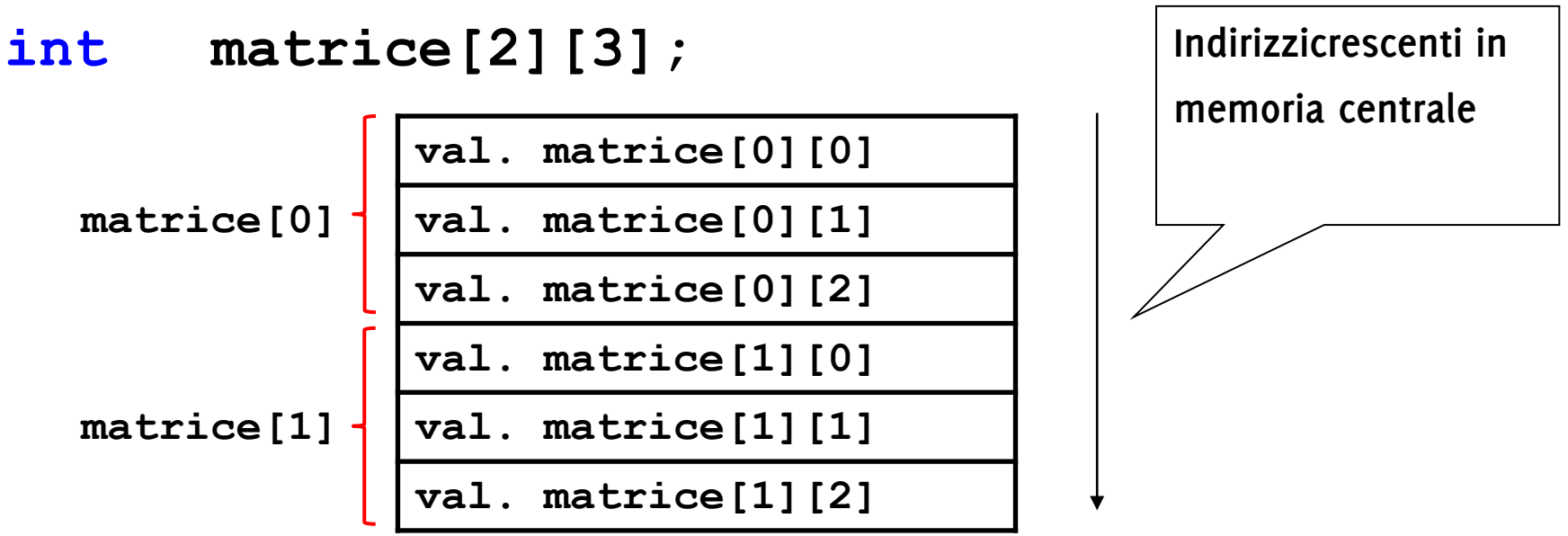
```
int   matrice[2][3];
```

`matrice[0][0]`	`matrice[0][1]`	`matrice[0][2]`
`matrice[1][0]`	`matrice[1][1]`	`matrice[1][2]`

Mappa di Memorizzazione di un array 2D

Cioè la rappresentazione in memoria di un array a 2 dimensioni:

- array memorizzato riga per riga, per indice di riga crescente e, all'interno di ogni riga, per indice di colonna crescente.



Mappa di Memorizzazione di un array 2D

Data la seguente dichiarazione di un vettore

```
int  vett [Dim1];
```

Vale la seguente uguaglianza (attenzione, si tratta indirizzo + intero!)

```
&vett[i] == &vett[0] + i
```

definiscono entrambe l'indirizzo dell'elemento **i**-simo

Per le matrici invece, data

```
int  matrice[Dim1][Dim2];
```

Vale la seguente uguaglianza

```
&matrice[i][j]== &matrice[0][0] + i*Dim2 + j
```

definiscono entrambe l'indirizzo dell'elemento alla riga **i** e colonna **j** in **matrice**

N.B: l'indice parte da **0**, per la prima riga vale come per i vettori

```
&matrice[0][j] == &matrice[0][0] +j
```

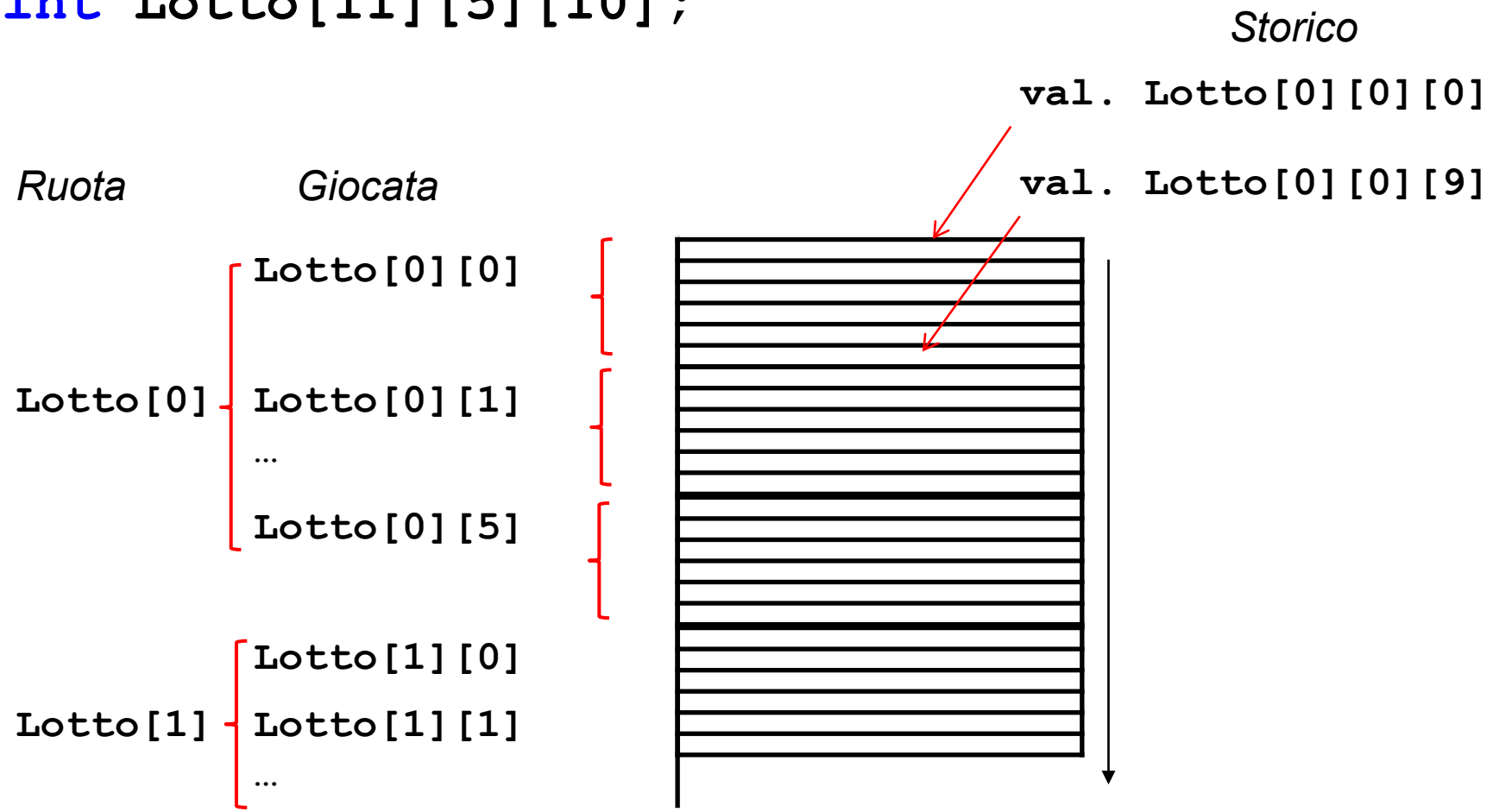
Mappa di Memorizzazione di un array 3D

Definire un tipo di dato atto a contenere i numeri estratti nelle ultime 10 giocate su tutte le 11 ruote (vengono estratti 5 numeri per giocata)

Mappa di Memorizzazione di un array 3D

Definire un tipo di dato atto a contenere i numeri estratti nelle ultime 10 giocate su tutte le 11 ruote (vengono estratti 5 numeri per giocata)

```
typedef int Lotto[11][5][10];
```



Matrici: Array di Array

dichiarazione più sintetica e per la variabile **M**

```
int M[4][5];
```

Quindi per il tipo **Matrice4Per5**

```
typedef int Matrice4Per5[4][5];
Matrice4Per5 M;
```

assegnamento a un elemento della matrice: `M[1][3]=7;`

Esempio: matrice tridimensionale:

```
int matrice3D[10][20][30];
```

per accedere agli elementi di **matrice3D**

```
matrice3D [2][8][15]
```

Conversioni

Compatibilità e Conversioni

Criterio adottato da C per espressioni e assegnamenti:

- se costanti e variabili dello **stesso tipo**, applica **operazione associata a quel tipo**
- se di **tipi diversi** applica, se possibile **regole di conversione**

In questo corso tratteremo solo regole di **conversione implicita**, non quelle di conversione esplicita (casting).

- La conversione viene eseguita dal compilatore.
- Ci sono casi in cui non è possibile eseguire conversioni implicite, ad esempio quando provo ad assegnare una variabile strutturata ad una non strutturata o variabili strutturate differenti.

Espressioni Aritmetiche tra Elementi Eterogenei

Ogni espressione aritmetica è caratterizzata da:

- **valore del risultato**
- **tipo del risultato**

Il **tipo** degli **operandi determina l'operazione** eseguita:

- se x ed y sono dello stesso tipo, nessun problema
- se x ed y sono di tipo diverso,
  - valuto se è possibile la conversione implicita
  - se non è possibile la conversione implicita l'operazione non è eseguibile

La fattibilità delle operazioni è valutata a compile-time: **tipizzazione forte** del C

Espressioni Aritmetiche: Conversione Implicita

Espressioni del tipo

`x op y`

con **`x`** e **`y`** di tipi diversi

in C i tipi sono ordinati in base alla precisione

```
char < short int < int < long int < float < double < long double
```

Regola Generale

1. I valori del tipo meno preciso sono implicitamente convertiti a valore del tipo più preciso (**promozione**).
2. L'operazione è definita dal tipo più preciso.
3. Il risultato è un valore del tipo più preciso.

Esempio

Esempio:

```
short x; int y,z;
```

Cosa avviene quando valuto l'espressione **`x + y ?`**

Il valore di **`x`** viene convertito **temporaneamente** in **`int`**

viene applicata operazione di somma tra **`int`**

Il risultato è **`int`**

NB la variabile **`x`** continua a restare **`short`**. La **promozione riguarda solamente il valore letto, non viene quindi assegnato il valore "promosso" alla variabile**

NB. conversioni, terreno scivoloso ... attenzione a quanto accade con gli **`unsigned`** ($\Rightarrow$ evitarli)

Assegnamenti: Conversione Implicita

Lo stesso ordinamento in base alla precisione

`char < short int < int < long int < float < double < long double`

viene utilizzato per assegnamenti tra variabili eterogenee

Esempi

- Siano **`double d; int i;`**
- **`d = i;`** valore di **`i`** convertito a **`double`** e assegnato a **`d`** $\Rightarrow$ non c'è perdita di informazione
- **`i = d;`** **`d`** convertito a **`int`** (troncandolo), valore intero assegnato a **`i`** $\Rightarrow$ perdita informazione

Esempio

```
// conversione da gradi Fahrenheit a Celsius
#include<stdio.h>
void main()
{
  int  Ftemp;
  float Ctemp;
  printf("Inserire gradi    Fahrenheit\n");
  scanf("%d", &Ftemp);

  Ctemp = (5.0 / 9.0 ) * (Ftemp - 32);

  printf("Celsius %2.2f" , Ctemp);
}
```

Cosa succede? `Ftemp` è `int` `Ctemp` è `float`

Cosa stampano le seguenti istruzioni se **`Ftemp = 50`**?

1. **`Ctemp = (5.0 / 9.0) * (Ftemp - 32);`**
2. **`Ctemp = (5 / 9) * (Ftemp - 32);`**
3. **`Ctemp = (5.0 / 9) * (Ftemp - 32);`**
4. **`Ctemp = 1.0 * (5 / 9) * (Ftemp - 32);`**
5. Se avessi dichiarato in **`int Ctemp`** (con **`%d`** in **`printf`**)?

Cosa succede? `Ftemp` è `int` `Ctemp` è `float`

Cosa stampano le seguenti istruzioni se **`Ftemp = 50`**?

1. `Ctemp = (5.0 / 9.0) * (Ftemp - 32);`
2. `Ctemp = (5 / 9) * (Ftemp - 32);`
3. `Ctemp = (5.0 / 9) * (Ftemp - 32);`
4. `Ctemp = 1.0 * (5 / 9) * (Ftemp - 32);`
5. Se avessi dichiarato in **`int Ctemp`** (con **`%d`** in **`printf`**)?

1) **`Celsius = 10.0`** (**`5.0`** e **`9.0`** sono **`float`** ⇒ **`/`** è divisione tra **`float`**, **`Ftemp`** variabile **`int`** e **`32`** e costante **`int`** ⇒ risultato sottrazione **`int`** ma la moltiplicazione causa conversione implicita (promozione) di (**`Ftemp - 32`**) a **`float`**)
2) **`Celsius = 0`** (**`5`** e **`9`** sono **`int`** ⇒ **`/`** diventa divisione tra **`int`**)
3) **`Celsius = 10.0`** (**`5.0`** è **`float`**, **`9`** è promosso a **`float`**).
4) **`Celsius = 0`** (**`1.0`** viene moltiplicato per **`0`**)
5) **`Celsius = 10`** (risultato **`float`** assegnato alla **`Ctemp`**, che è **`int`** ⇒ possibile perdita di informazione)

Qualche esercizio

Esercizio

Le seguenti dichiarazioni definiscono tipi di dati relativi alla categoria degli impiegati di un'azienda (gli impiegati possono essere di prima, seconda, …, quinta categoria), agli uffici occupati da tali impiegati, all'edificio che ospita tali uffici (l'edificio è diviso in 20 piani ognuno con 40 uffici).

Esercizio

```
/* definizioni dei tipi */
typedef struct {char nome[20], cognome[20];
      int cat; // contiene valori tra 1 e 5
      int stipendio;
} Impiegato;

typedef struct { int superficie; /*in m^2*/
      char esp[20];
      Impiegato occupante;
} Ufficio;

/* definizioni delle variabili */
Ufficio torre[20][40]; /* rappresenta un edificio di 20 piani con
40 uffici per piano */
```

Esercizio

Si scriva un frammento di codice, che includa eventualmente anche le dichiarazioni di ulteriori variabili, che, per tutte e sole le persone che occupano **un ufficio** (tra quelli memorizzati nella variabile torre) **orientato a sud oppure a sudEst e** avente una **superficie compresa tra 20 e 30 metri quadri**, stampi il cognome, lo stipendio e la categoria.

Soluzione punto 1

```
int p, u; /* indice di piano nell'edificio e di ufficio nel piano
*/
for (p=0; p<20; p++)
   for (u=0; u<40; u++)
      if ((strcmp(torre[p][u].esp, "sudEst") == 0 ||
      (strcmp(torre[p][u].esp, "sud") == 0)
      && (torre[p][u].superficie >= 20
      && torre[p][u].superficie <= 30))
      {
      printf("\n il Signor %s è impiegato di categoria %d",
      torre[p][u].occupante.cognome, torre[p][u].occupante.cat);
       printf (" e ha uno stipendio pari a %d euro \n",
      torre[p][u].occupante.stipendio);
   }
```

Soluzione punto 1

```
int p, u; /* indice di piano nell'edificio e di ufficio nel piano
*/
for (p=0; p<20; p++)
   for (u=0; u<40; u++)
      if ((strcmp(torre[p][u].esp, "sudEst") == 0 ||
      (strcmp(torre[p][u].esp, "sud") == 0)
      && (torre[p][u].superficie >= 20
      && torre[p][u].superficie <= 30))
      {
      printf("\n il Signor %s è impiegato di categoria %d",
      torre[p][u].occupante.cognome, torre[p][u].occupante.cat);
       printf (" e ha uno stipendio pari a %d euro \n",
      torre[p][u].occupante.stipendio);
   }
```

Scorro l'array torre come una normale matrice

Esercizio

Si scriva un frammento di codice, che includa eventualmente anche le dichiarazioni di ulteriori variabili, che, per tutte e sole le persone che occupano **un ufficio** (tra quelli memorizzati nella variabile torre) **orientato a sud oppure a sudEst e** avente una **superficie compresa tra 20 e 30 metri quadri**, stampi il cognome, lo stipendio e la categoria.

[aggiunto] Visualizzi a schermo i numeri dei piani che non hanno neanche un ufficio esposto a nord.

Soluzione punto 2

```
int uffNord; /* uffNord fa da flag*/
for (p=0; p<20; p++)
{
  uffNord = 0;//flag = 0 in ogni piano
  for (u=0; u<40 && uffNord == 0; u++)
     if(strcmp(torre[p][u].esp, "nord«) == 0
       uffNord = 1;
- /* se qui vale ancora 0 vuol dire che non ci sono uffici a
nord*/
  if (uffNord == 0);
     printf("il piano %d  non ha edifici esposti a nord",
p);
}
```

Soluzione punto 2

```
int uffNord; /* uffNord fa da flag*/
for (p=0; p<20; p++)
{
  uffNord = 0;//flag = 0 in ogni piano
   for (u=0; u<40 && uffNord == 0; u++)
      if(strcmp(torre[p][u].esp, "nord«) == 0
        uffNord = 1;
- /* se qui vale ancora 0 vuol dire che non ci sono uffici a
nord*/
   if (uffNord == 0);
      printf("il piano %d  non ha edifici esposti a nord",
p);
}
```

Sto usando una variabile di flag per vedere se non ci sono uffici che affacciano a nord

Esercizio

Si scriva un frammento di codice, che includa eventualmente anche le dichiarazioni di ulteriori variabili, che, per tutte e sole le persone che occupano **un ufficio** (tra quelli memorizzati nella variabile torre) **orientato a sud oppure a sudEst e** avente una **superficie compresa tra 20 e 30 metri quadri**, stampi il cognome, lo stipendio e la categoria.

[aggiunto] Visualizzi a schermo i numeri dei piani che non hanno neanche un ufficio esposto a nord

[aggiunto] Dire in che piano ed in che ufficio si trova Boracchi

[aggiunto] Copiare in un array tutti gli uffici occupati da dipendenti di categoria 5

[aggiunto] Copiare in un array tutti i dipendenti di categoria 5

Esercizio

[aggiunto] Come modificare le strutture dati (ed i codici) precedenti per rappresentare un edificio avente un diverso numero (max 40) di uffici per piano?

Esempio Matrici

Si scriva un programma C che acquisisce una matrice di interi di nome matr e di dimensione NxN (con N definito come costante) e due interi X e Y da standard input.

Se X e Y non sono indici ammissibili della matrice, il programma termina.

In caso contrario, il programma stampa la stringa “successo!” se l’elemento matr[X][Y] è uguale a X*Y.

Se quest’ultima condizione non è verificata, il programma considera gli elementi della riga X in matr e controlla se tra questi quelli maggiori di matr[X][Y] sono di più di quelli minori di matr[X][Y]. Se questo è il caso, il programma stampa il valore contenuto in matr[X][Y].

Ad esempio, se la riga X di matr è costituita dagli elementi {12, 7, 15, 5, 3} e l’elemento che stiamo considerando è quello di posizione 1 (il valore 7), possiamo concludere che due elementi della riga sono minori di tale numero e altri due sono maggiori di esso. Di conseguenza, la condizione non è verificata e quindi il programma non stampa nulla.