Gli array

Docente
Mario Perna
prof.perna.mario@darzo.net

A.S.
2025/2026
Materia
Informatica

Introduzione

Negli anni precedenti abbiamo visto programmi che per memorizzare i dati necessari utilizzavano moltissime variabili.

Ad esempio, leggere 100 voti e calcolare la media di tali valori.

Proviamo a pensarci un attimo e a risolverlo…

Esempio di risoluzione

```cpp
#include <iostream>

using namespace std;

int main(){
    int n_voti = 100;
    int somma = 0;
    int voto;
    for (int i = 0; i < n_voti; i++){ // leggo 100 voti
        do{
            cout << "Inserire voto " << i << ": ";
            cin >> voto;
        }
        while (voto < 2 || voto > 10);  // se l'utente mette un voto non valido continuo a chiederlo
        somma = somma + voto;
    }
    double media = static_cast<double>(somma) / n_voti;
    cout << "La media dei voti vale: " << media << endl;
    return 0;
}
```

Troppe variabili!

Ora che abbiamo completato l'esercizio immaginiamo che dobbiamo anche dire quanti elementi sono maggiori della media appena calcolata.

Proviamo a pensarci un attimo e a risolverlo…

Purtroppo con la soluzione di prima memorizziamo nella variabile voto solo l'ultimo valore letto…

Esempio di risoluzione

```cpp
#include <iostream>
using namespace std;

int main(){
    int n_voti = 100;
    int v1;                         // DOBBIAMO DICHIARARE 100 VARIABILI!!!
    int v2;
    ...
    int v100;

    cout << "Inserisci voto 1:";    // DOBBIAMO FARLO 100 VOLTE!!!
    cin >> v1;
    cout << "Inserisci voto 2:";
    cin >> v2;
    ...
    cout << "Inserisci voto 100:";
    cin >> v100;

    int somma = v1 + ... + v100;
    double media = static_cast<double>(somma) / n_voti;

    int n_voti_maggiori_media = 0;
    if (v1 > media)                 // DOBBIAMO FARLO 100 VOLTE!!!
        n_voti_maggiori_media++;
    if (v2 > media)
        n_voti_maggiori_media++;
    ...
    if (v100 > media)
        n_voti_maggiori_media++;

    cout << "I voti maggiori della media sono: " << n_voti_maggiori_media << endl;
    return 0;
}
```

Non possiamo evitarlo???
Sì, utilizzando gli array

Gli array

Gli array sono strutture dati che permettono di memorizzare un numero definito di dati **omogenei** tra di loro. (**omogenei = dello stesso tipo**).

```
int numeri_fortunati[10];
float prezzi[5];
char lettere_ammesse[8];
```

E' una struttura dati **statica**, cioè deve essere noto a tempo di **compilazione** (prima dell'avvio del programma) quanta memoria dovere allocare.

```
const int N = 12;
int n = 9;
int v1[5]; // OK, 5 è noto a tempo di compilazione
int v2[N]; // OK, N è una costante nota a tempo di compilazione
int v3[n]; // NO!!! n è una variabile e assumerà il valore 9 a tempo di esecuzione
```

Gli array

Ed infine consente un accesso **diretto** ai dati.

```
int v1[5]; // array di 5 interi in posizioni 0,1,2,3,4

v[2] = 42;
v[4] = -2;
v[5] = 6;   // NO!!! array fuori indice
```

indice	0	1	2	3	4
valore	?	?	42	?	-2

ATTENZIONE: si parte dall'indice 0 e non 1

Gli array in generale

In generale un array viene dichiarato con la seguente sintassi:

<tipo di dato> <nome array> [<dimensione array>] tipo di dato può

essere int, float, bool, char, …

nome array deve seguire le stesse regole di convenzione dei nomi per le variabili
dimensione array deve essere maggiore di 0 e essere nota a tempo di compilazione

Alcuni modi per inizializzare gli array

Possiamo inizializzare un array come mostrato in figura nei seguenti modi:

0	1	2	3	4
8	-2	90	3	5

```
const int N = 5;
int v[N];

v[0] = 8;
v[1] = -2;
v[2] = 90;
v[3] = 3;
v[4] = 5;
```

```
const int N = 5;
int v[N] = {8, -2, 90, 3, 5};
```

Posso anche non dichiarare la dimensione in questo caso

```
int v[] = {8, -2, 90, 3, 5}
```

Gli array: navigazione

E' possibile navigare l'intero array utilizzando un ciclo a vostra scelta (for, while-do e do-while).

```
const int N = 5;
int v[N];

// Metodo 1 con while do
int i = 0;
while (i < N){
    cout << "Inserire il valore " << i << ": ";
    cin >> v[i];
    i++;
}
// Metodo 2 con do while
i = 0;
do{
    cout << "Inserire il valore " << i << ": ";
    cin >> v[i];
    i++;
}
while (i < N);
// Metodo 3 con for
for (i = 0; i < N; i++){
    cout << "Inserire il valore " << i << ": ";
    cin >> v[i];
}
```

```
Inserire il valore 0: 90
Inserire il valore 1: -2
Inserire il valore 2: 6
Inserire il valore 3: 5
Inserire il valore 4: 1
```

Gli array: esempio

Proviamo a risolvere di nuovo il seguente problema: leggere 100 voti e dire quanti voti superano la media.

```cpp
#include <iostream>
using namespace std;

int main(){
    const int N_VOTI = 100;
    int somma = 0;
    int voti[N_VOTI];
    int voto;
    for (int i = 0; i < N_VOTI; i++){ // leggo 100 voti
        do{
            cout << "Inserire voto " << i << ": ";
            cin >> voto; // l'ultimo voto letto sostiuisce gli altri
        }
        while (voto < 2 || voto > 10);  // se l'utente mette un voto non valido continuo a chiederlo
        voti[i] = voto;     // memorizzo il voto i-esimo nell'array
        somma = somma + voto;
    }
    double media = static_cast<double>(somma) / N_VOTI;
    cout << "La media dei voti vale: " << media << endl;

    int n_voti_superiori_media = 0;
    for (int i = 0; i < N_VOTI; i++){
        if (voti[i] > media)
            n_voti_superiori_media++;
    }
    cout << "Il numero di voti superiori alla media vale: " << n_voti_superiori_media << endl;
    return 0;
}
```

Molto meglio rispetto a prima!

Gli array: operazioni generali

Le operazioni effettuabili sugli array sono infinite vediamo alcune delle più importanti: le operazioni **C.R.U.D.**

Gli array: esempio pratico

Immaginiamo di dovere gestire un elenco di persone che arrivano ad una coda.
La coda può contenere al massimo 100 persone.
Ogni persona è memorizzata come una stringa contenente il suo nome.
Possiamo eseguire 4 tipi di operazioni sulla coda:

- **Inserimento** di una nuova persona in coda (C)
- **Ricerca** di una persona in coda (R)
- **Aggiornamento** dei dati di una persona in coda (U)
- **Rimozione** di una persona in coda (D)

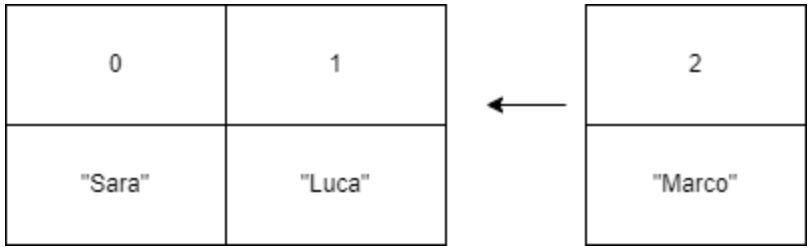
Gli array: esempio pratico

Inizio creando un modello dei miei dati:

```
const int MAX_PERSONE = 100;     // massimo numero di persone nella coda
string coda[MAX_PERSONE];        // array di 100 elementi di tipo stringa
int numero_persone = 0;          // numero di persone attuali in coda
```

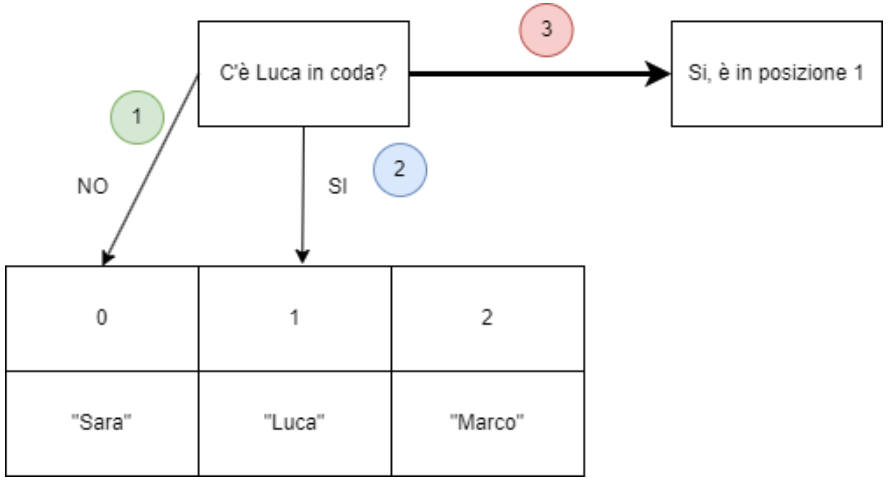
Implemento la prima operazione (inserimento):

```
string nome_persona;
cout << "Inserisci il nome della persona da inserire: ";
cin >> nome_persona;
if (numero_persone < MAX_PERSONE){
    coda[numero_persone] = nome_persona;
    numero_persone++;
}
else{
    cout << "Inserimento non eseguito, coda piena";
}
```



Gli array: esempio pratico

Implemento la seconda operazione (ricerca):




```
string nome_persona_ricerca;
cout << "Inserisci il nome della persona da cercare: ";
cin >> nome_persona_ricerca;
bool trovato = false;
int i = 0;
while (!trovato && i < numero_persone){
    if (coda[i] == nome_persona_ricerca)
        trovato = true;
    else
        i++;
}
if (trovato)
    cout << "La persona " << nome_persona_ricerca << " e' stata trovata in posizione " << i << endl;
else
    cout << "La persona " << nome_persona_ricerca << " non e' stata trovata" << endl;
```

Gli array: esempio pratico

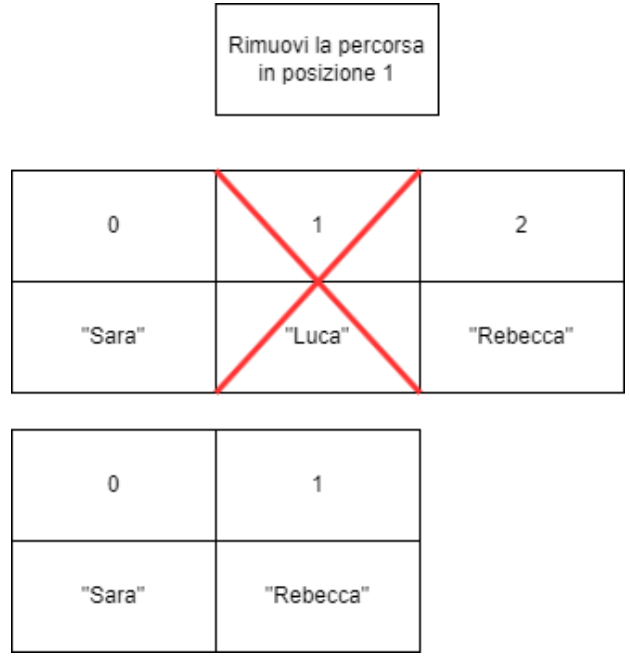
Implemento la terza operazione (modifica):

```
int posizione_aggiornamento;
cout << "Inserisci la posizione della coda che vuoi modificare: ";
cin >> posizione_aggiornamento;
if (posizione_aggiornamento >= 0 && posizione_aggiornamento < numero_persone){
    string nome_persona_aggiornamento;
    cout << "Inserisci il nome della persona aggiornato: ";
    cin >> nome_persona_aggiornamento;
    coda[posizione_aggiornamento] = nome_persona_aggiornamento;
    cout << "Persona aggiornata con successo!" << endl;
}
else
    cout << "Posizione inserita non valida!" << endl;
```

Gli array: esempio pratico

Implemento la quarta operazione (rimozione):

```
int posizione_rimozione;
cout << "Inserisci la posizione della coda da rimuovere: ";
cin >> posizione_rimozione;
if (posizione_rimozione >= 0 && posizione_rimozione < numero_persone){
    for (int i = posizione_rimozione; i < numero_persone-1; i++){
        coda[i] = coda[i+1];
    }
    numero_persone--;
    cout << "Persona rimossa con successo!" << endl;
}
else
    cout << "Posizione inserita non valida!" << endl;
```



Gli array: operazioni di ricerca min e max

Altre operazioni molto comuni sono la ricerca dell'elemento **minimo** e/o **massimo** all'interno di un array.

```
const int N = 6;
int v[N] = {5, 90,-100, 2000, 8, 49};
```

massimo

```
int max_v = v[0];

for (int i = 1; i < N; i++){
    if (v[i] > max_v)
        max_v = v[i];
}
```

minimo

```
int min_v = v[0];

for (int i = 1; i < N; i++){
    if (v[i] < min_v)
        min_v = v[i];
}

cout << "MAX: " << max_v << endl;
cout << "MIN: " << min_v << endl;
```

Informazioni sull’utilizzo dei materiali didattici

Queste slides si basano su materiali originariamente elaborati dal **Prof. Andrea Melioli**, opportunamente modificati e integrati secondo specifiche esigenze riguardanti la programmazione disciplinare.

L’autore autorizza al **prof. Mario Perna** l’utilizzo, la modifica, la pubblicazione e qualsiasi altra forma di operazione sui materiali a scopo didattico e formativo.

L’uso o la diffusione di questi materiali è **vietato** senza il preventivo contatto con il proprietario del documento, Prof. Mario Perna (prof.mario.perna@darzo.net).