Gli array: concetti avanzati

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

A.S.
2025/2026
Materia
Informatica

Dimensione fisica

Molto spesso non si è a conoscenza di quanti elementi dovrà contenere un array fino a quando il programma non viene eseguito. Un modo per cercare di risolvere questo problema è quello di stabilire un «numero massimo» di valori che prevediamo di memorizzare.

Ad esempio analizziamo il seguente problema:
Memorizzare i voti degli studenti fino a quando l'utente non digita un numero negativo.

In questo caso possiamo stabilire un numero massimo di elementi pari, ad esempio, a 500 che chiameremo DIMENSIONE_FISICA.

```
const int DIM_FISICA = 500;

int main(){

    int vettore[DIM_FISICA];
}
```

Dimensione logica

Per memorizzare quanti elementi verranno effettivamente memorizzati utilizziamo una variabile d'appoggio di tipo intero che conta quanti voti sono stati inseriti dall'utente.

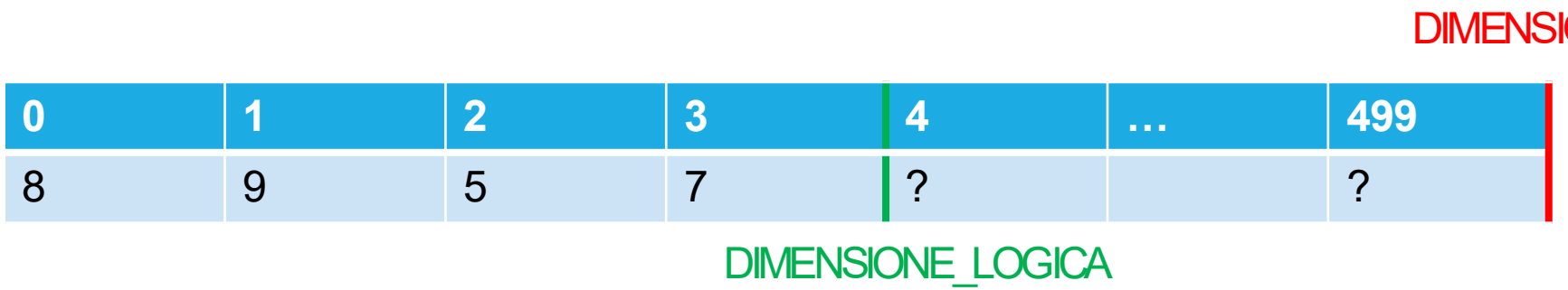
Possiamo definire questa variabile come la dimensione_logica del nostro array.

Dove 0 <= dimensione_logica <= dimensione_fisica

```
const int DIM_FISICA = 500;

int main(){

    int vettore[DIM_FISICA];
    int dimensione_logica = 0;
    int voto;
    do{
        cout << "Inserire il voto " << dimensione_logica << ": ";
        cin >> voto;
        if (voto >= 2 && voto <= 10){
            vettore[dimensione_logica] = voto;
            dimensione_logica++;
        }
    }while (voto >= 0);
}
```



Dimensione fisica vs logica

Riassumendo se:

1. Se conosci la dimensione PRIMA dell'esecuzione del programma → DIMENSIONE_FISICA impostata al valore prescelto

Esempio: *leggere in input 50 prezzi e mostrare i prezzi superiori alla media* (DIMENSIONE_FISICA = 50)

2. Se non conosci la dimensione PRIMA dell'esecuzione del programma → DIMENSIONE_LOGICA e DIMENSIONE_FISICA impostata ad un valore «tetto» massimo non superabile

Esempio: *leggere in input le temperature delle varie giornate fino a quando l'utente non inserisce il carattere «F»*. (DIMENSIONE_FISICA = ? Imposto una numerosità massima e poi utilizzo la DIMENSIONE_LOGICA)

Ordinamento degli array

Un'operazione molto comune sugli array è quella di ordinarli secondo qualche criterio.
In queste slide vedremo come ordinare gli elementi in ordine crescente/decrescente mediante l'algoritmo di ordinamento **bubblesort**.

0	1	2	3	4	5	6
8	9	5	7	2	4	4

array disordinato

0	1	2	3	4	5	6
2	4	4	5	7	8	9

array ordinato in ordine crescente

0	1	2	3	4	5	6
9	8	7	5	4	4	2

array ordinato in ordine decrescente

Bubblesort: versione «forza bruta»

I passaggi dell'algoritmo sono i seguenti:

1. Inizia confrontando i primi due elementi. Se sono nell'ordine sbagliato, scambiali.
2. Continua questo processo per tutti gli elementi che si spostano da sinistra a destra. Dopo il primo passaggio, l'elemento più grande sarà alla fine.
3. Nel passaggio successivo, salta l'ultimo elemento poiché è già ordinato e ripeti i passaggi precedenti. Il secondo elemento più grande si sposterà nella penultima posizione.
4. Ripeti i passaggi finché l'intero array non sarà ordinato.

```
// ***Algoritmo Bubble Sort ordine crescente***
// Ad ogni iterazione metto nella
// posizione corretta un elemento
for (int i = 0; i< DIM_FISICA-1; i++){
    // Per ogni elemento tranne l'ultimo
    for (int j = 0; j < DIM_FISICA-1; j++){
        // Se il suo successivo è minore
        if (vettore[j] > vettore[j+1]){
            // allora scambio i due elementi
            int box = vettore[j];
            vettore[j] = vettore[j+1];
            vettore[j+1] = box;
        }
    }
}
```

Bubblesort: versione «raffinata»

Possiamo migliorare il nostro algoritmo in termini di efficienza e di numeri di scambi effettuati. Ad esempio se nel for più interno non eseguiamo neanche uno scambio allora abbiamo ordinato l'array e non ha senso continuare ad iterare…

Meditate bene su quanto appena detto

```
// ***Algoritmo Bubble Sort ordine crescente***
bool ordinato = false;
// Finchè l'array non è ordinato
while (!ordinato){
    // A questo giro per ora l'array è ordinato
    ordinato = true;
    // Per ogni elemento tranne l'ultimo
    for (int i = 0; i < DIM_FISICA-1; i++){
        // Se il suo successivo è minore
        if (vettore[i] > vettore[i+1]){
            // allora scambio i due elementi
            int box = vettore[i];
            vettore[i] = vettore[i+1];
            vettore[i+1] = box;
            // e segnalo che non è ordinato
            ordinato = false;
        }
    }
}
```

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).