Array statici

András Horváth
horvath@di.unito.it

Gli array

- gli array sono **sequenze di variabili dello stesso tipo**
- gli elementi vengono situate consecutivamente nella memoria
- per creare (dichiarare) un array bisogna **specificare il tipo** degli elementi, **il nome** e il numero degli elementi, cioè **la dimensione**
- per esempio, con `int v[5];` si crea nella memoria un array di 5 `int`:

v[0]	v[1]	v[2]	v[3]	v[4]

- **l'indice del primo elemento è 0**
- **per accedere** ad un elemento bisogna usare il **nome** dell'array e specificare l'**indice** dell'elemento
- sono **statici** perché il tipo e la dimensione non sono modificabili durante l'esecuzione
- il **nome** dell'array scritto **senza indice è l'indirizzo del primo elemento** (`cout << v;` stampa l'indirizzo del primo elemento)

Gli array

- C e C++ **non verificano gli indici dell'array**, cioè non viene controllato se gli indici siano validi (stiano dentro la dimensione)
- per esempio:
  `double v[2]={1.2,0.8};`
  `cout << v[2];`
  è un errore perché l'indice dell'ultimo elemento è 1
- al momento della compilazione non viene segnalato nessun errore
- durante l'esecuzione:
  - può succedere che il sistema operativo ferma il programma quando il programma tenta di eseguire il commando `cout << v[2];`
  - oppure il programma esegue ma il suo comportamento è casuale (stampa ciò che si trova nella memoria dopo `v[1]`)

Esempio di dichiarazione e utilizzo di un array

```
1  // dichiarazione e utilizzo di un array
2  #include<iostream>
3  using namespace std;
4
5  int main(){
6    double n[10],m;
7    for(int i=0;i<=9;i=i+1)
8      cin >> n[i];
9    m=0;
10   for(int i=0;i<=9;i=i+1)
11     m=m+n[i];
12   cout << m/10 << endl;
13   return 0;
14 }
```

Numero di elementi

- risaputo al momento della compilazione:

```
5    double n[10]; // number of entries known at compile time, OK
```

- risaputo al momento dell'esecuzione del programma:

```
7    int k;
8    cin >> k;
9    int a[k]; // number of entries specified at run time, OK
```

- è casuale e quindi questo è UN GRAVE ERRORE:

```
11   int c;
12   int b[c]; // number of entries random at run time, NOT OK!!!!!
13   cin >> c;
```


Array come parametro, funzioni

- la funzione riceve l'indirizzo del primo elemento, "sa" dove si trova l'array, può modificarlo:

```
5  void read_array(int v[], int n){
6    for(int i=0;i<n;i++)
7      cin >> v[i];
8  }
```

- la funzione riceve l'indirizzo del primo elemento, "sa" dove si trova l'array, ma grazie alla parola chiave **const** non può modificarlo:

```
10 void print_array(const int v[], int n){
11   for(int i=0;i<n;i++)
12     cout << v[i] << "␣";
13   cout << endl;
14 }
```


Array come parametro, intero programma

```
1  // array come parametro
2  #include<iostream>
3  using namespace std;
4
5  void read_array(int v[], int n){
6    for(int i=0;i<n;i++)
7      cin >> v[i];
8  }
9
10 void print_array(const int v[], int n){
11   for(int i=0;i<n;i++)
12     cout << v[i] << "␣";
13   cout << endl;
14 }
15
16 int main(){
17   int n;
18   cin >> n;
19   int a[n];
20   read_array(a,n);
21   print_array(a,n);
22   return 0;
23 }
```


Ordinamento per selezione, idea

- procedimento:
  - si cerca l'elemento più piccolo e si scambia questo elemento con il primo elemento
  - si cerca l'elemento più piccolo escludendo la prima posizione e si scambia questo elemento con il secondo elemento
  - si cerca l'elemento più piccolo escludendo le prime due posizioni e si scambia questo elemento con il terzo elemento
  - ...
- in altre parole: si amplia la parte ordinata cercando l'elemento più piccolo della parte non ordinata e facendo un scambio
- esempio (parte in grassetto è ordinato):
  - vettore iniziale: (8,10,1,2,5)
  - dopo un scambio: (**1**,10,8,2,5)
  - dopo due scambi: (**1,2**,8,10,5)
  - dopo tre scambi: (**1,2,5**,10,8)
  - dopo quattro scambi: (**1,2,5,8,10**)

Ordinamento per selezione, codice

```
1  // ordinamento per selezione
2  #include<iostream>
3  using namespace std;
4
5  // stampa vettore di interi
6  void print(const int n, int v[]){
7    for(int i=0;i<n;i++)
8      cout << v[i] << "␣";
9    cout<<"\n";
10 }
```


Ordinamento per selezione, codice

```
12 // restituisce indice dell'elemento minimo in (v[i],v[i+1],...,v[j])
13 int minindex(const int v[], int i, int j){
14   int m=i;
15   for(int k=i+1;k<=j;k++)
16     if(v[k]<v[m])
17       m=k;
18   return m;
19 }
20
21 // scambio
22 void swap(int &a, int &b){
23   int temp=a;
24   a=b;
25   b=temp;
26 }
```


Ordinamento per selezione, codice

```
28 // ordinamento per selezione
29 void selection(int n, int v[]){
30   int i,mi;
31   for(i=0; i<n-1; i++){
32     mi = minindex(v,i,n-1);
33     swap(v[mi],v[i]);
34   }
35 }
36
37 int main(){
38   const int x=10;
39   int v[x]={10,6,8,2,4,1,7,8,2,3};
40   print(x,v);
41   selection(x,v);
42   print(x,v);
43   return 0;
44 }
```


Array multidimensionali, dichiarazione

```
5    int m1[3][5]; // number of entries known at compile time, OK
6
7    int ri,co;
8    cin >> ri;
9    cin >> co;
10   int m2[ri][co];  // number of entries specified at run time, OK
11
12   for(int i=0;i<ri;i++)
13     for(int j=0;j<co;j++)
14       cin >> m2[i][j];
15
16   for(int i=0;i<ri;i++){
17     for(int j=0;j<co;j++)
18       cout << m2[i][j] << "␣";
19     cout << endl;
20   }
21
22   int r,c;
23   int m3[r][c];  // number of entries random at run time, NOT OK!!!!!
24   cin >> r;
25   cin >> c;
```

Array multidimensionali, come parametro

possiamo omettere dalla specifica degli array multidimensionali solo la prima dimensione e le altre devono essere fornite tramite costanti:

```
5  // dimensione massima di un array durante l'esecuzione:
6  const int max_size=100;

23 // stampa una matrice a video
24 void print_matrix(const int m[][max_size], int r, int c){
25   for(int i=0;i<r;i++){
26     for(int j=0;j<c;j++)
27       cout << m[i][j]<< "␣";
28     cout << endl;
29   }
30 }
```


Altro da studiare

- operatori **++**, **--**, **+=**, **-=**, ***=** e **/=**: sezione 2 degli appunti
- programmi forniti negli appunti