PROGRAMMIAMO

C++ - Il contenitore vector

Il contenitore vector

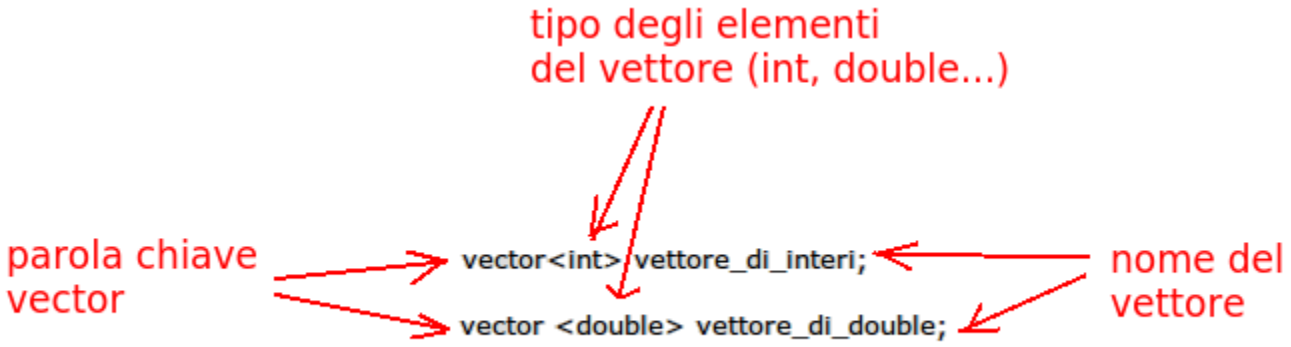
Anche la gestione dei vettori viene semplificata in C++ dall'uso di un apposito *contenitore* (cioè di una classe predefinita) di nome *vector*. Si consideri l'esempio seguente:

```
#include <vector>

int main()
{
vector<int> vettore_di_interi;
vector <double> vettore_di_double;

...
```

Occorre prestare particolare attenzione alla sintassi usata:



Per l'esattezza *vector* non è una classe ma un **template di classi**. L'argomento template è troppo vasto per essere trattato approfonditamente qui. Basti sapere che un template è un modello generale per creare tante classi diverse. Nel caso di vector le classi create differiscono per il tipo degli elementi del vettore (int, double, char...) che viene appunto indicato fra < e > dopo la parola chiave vector.

Notiamo che con questa sintassi non viene dichiarato il numero di elementi dei vettori: in questo modo i due vettori sono inizialmente vuoti, cioè non contengono elementi.

Metodi del contenitore vector

Per riempire i vettori e contestualmente creare nuovi elementi, occorre usare il metodo **push_back** come mostrato nel seguente esempio:

```
vector<double> numeri;
```

```
double valore;
int scelta=1;

while (scelta==1)
{
cout<<"Inserisci un numero: ";
cin>>valore;
numeri.push_back(valore);
cout<<"Digita 1 per proseguire e 0 per terminare l'inserimento: ";
cin>>scelta;
}
```

Si osservi che il valore da acquisire viene prima inserito nella variabile valore e quindi passato al vettore invocando il metodo push_back.

Per visualizzare il contenuto del vettore si può procedere nel seguente modo:

```
vector<double> numeri;
double valore;
int scelta=1, i;

while (scelta==1)
{
cout<<"Inserisci un numero: ";
cin>>valore;
numeri.push_back(valore);
cout<<"Digita 1 per proseguire e 0 per terminare l'inserimento: ";
cin>>scelta;
}

cout<<"Elenco dei valori inseriti:\n" ;
for (i=0; i<numeri.size();i++)
cout<<numeri[i]<<"\n";
```

Si osservi l'uso del metodo **size** per determinare il numero di elementi del vettore (analogamente a quanto visto per le stringhe). In alternativa la lettura degli elementi può essere effettuata usando il metodo **at** come nell'esempio seguente:

```
cout<<"Elenco dei valori inseriti:\n" ;
for (i=0; i<numeri.size();i++)
cout<<numeri.at(i)<<"\n";
```

L'uso del metodo *at* può essere vantaggioso in quanto gestisce anche le cosiddette eccezioni, cioè gli errori runtime che si verificano, per esempio, in caso di overflow del vettore. Usando at in caso di overflow il programma termina con un messaggio di errore (mentre con l'accesso diretto agli elementi del vettore l'overflow non viene segnalato in alcun modo).

Per una trattazione più completa ed approfondita del contenitore vector rimandiamo il lettore a cplusplus.com.

precedente - successiva

Sito realizzato in base al template offerto da

http://www.graphixmania.it

Segui @ElePrograMania