Libreria STL
Liste, Iteratori, Vettori

Corso di Programmazione ad Oggetti
Corso di Laurea in Informatica
Università della Calabria
https://www.mat.unical.it/informatica/ProgrammaAdOggetti

STL `list`

Occorre: **`#include <list>`** (e **`using namespace std`**)

È una classe *template*:

- **`list<int> integer_list;`**
- **`list<string> string_list;`**
- **`list<Contatto> contatti_list;`**

Implementazione versatile ed ottimizzata

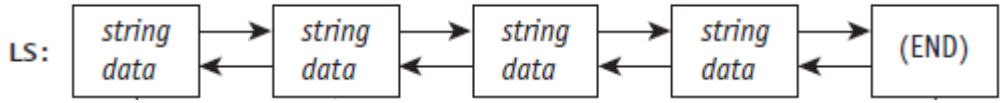
STL `list`

Implementata come **doppiamente concatenata**

- Ogni nodo è collegato al precedente e al successivo
- Iterazione bidirezionale

Esempio:

```
list<string> LS;
```



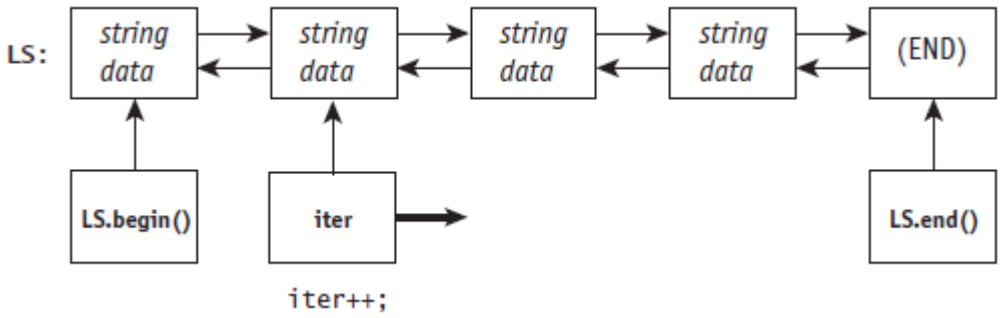
Iteratori e scansione di una lista

Meccanismo per la scansione di una `list`: ***Iteratori bidirezionali***

```
for(list<string>::iterator iter=LS.begin();iter!=LS.end();iter++){
      cout<<(*iter)<<endl;

}
```

`const_iterator` per impedire la modifica

```
for(list<string>::const_iterator iter=LS.begin();iter!=LS.end();iter++){
      cout<<(*iter)<<endl;

}
```



Principali metodi in STL `list`

Consultare la documentazione: http://www.cplusplus.com/reference/list/list/

Metodo	Descrizione
begin	Restituisce un iteratore che «punta» al primo nodo
end	Restituisce un iteratore che «punta» ad un nodo *FITTIZIO* finale
size	Restituisce la dimensione
empty	Restituisce *true* se la lista è vuota
front	Restituisce il primo elemento
back	Restituisce l'ultimo elemento
push_back	Inserisce un elemento alla fine
push_front	Inserisce un elemento all'inizio
insert	Permette di inserire un elemento in una posizione arbitraria
pop_back	Rimuove l'ultimo elemento
pop_front	Rimuove il primo elemento
erase	Dato un iteratore che «punta» ad un nodo (o un range) elimina l'elemento corrispondente (o gli elementi corrispondenti)
remove	Dato un elemento, lo cerca e rimuove tutte le sue occorrenze (Usa l'operatore ==)
clear	Rimuove tutti gli elementi
unique	Rimuove gli elementi duplicati *CONSECUTIVI* (Usa l'operatore ==)
sort	Ordina la lista (Usa l'operatore <)

STL `vector`

Occorre: **`#include <vector>`** (e **`using namespace std`**)

È una classe *template*:

- **`vector<int> integer_vector;`**
- **`vector<string> string_vector;`**
- **`vector<Contatto> contatti_vector;`**

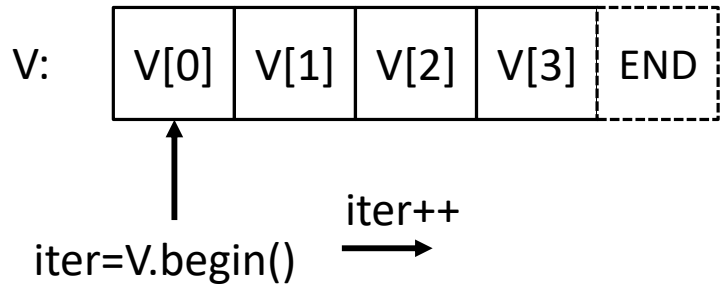
Locazioni di memoria contigue

- Accesso casuale (diretto, «random»)
- Gestione automatica della memoria dinamica allocata
  - Quando non c'è sufficiente memoria a disposizione, viene allocata un'area di memoria contigua più grande, ricopiati gli elementi originali e deallocata la vecchia memoria

Iteratori e scansione di un vettore

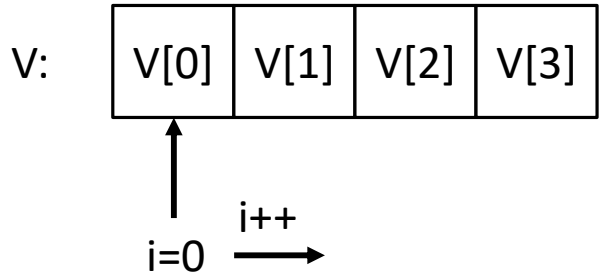
Meccanismo per la scansione di un vettore: ***Iteratori ad accesso diretto***

```
vector<string> V;

for(vector<string>::iterator iter=V.begin();
    iter!=V.end();iter++){
       cout<<(*iter)<<endl;

}
```



Oppure semplicemente:

```
for(unsigned int i=0;i<V.size();i++){
      cout<<V[i]<<endl;

}
```



Principali metodi in STL `vector`

Consultare la documentazione: http://www.cplusplus.com/reference/vector/vector/

Metodo	Descrizione
begin	Restituisce un iteratore che «punta» al primo elemento
end	Restituisce un iteratore che «punta» ad un elemento *FITTIZIO* finale
operator[]	Restituisce l'elemento presente in posizione i-esima. ***Attenzione**: non controlla se la posizione è valida.*
at	Restituisce l'elemento presente in posizione i-esima. ***Attenzione**: controlla se la posizione è valida e segnala un «errore».*
size	Restituisce la dimensione
capacity	Restituisce la capacità
empty	Restituisce *true* se il vettore è vuoto
front	Restituisce il primo elemento
back	Restituisce l'ultimo elemento
push_back	Inserisce un elemento alla fine
pop_back	Rimuove l'ultimo elemento
insert	Permette di inserire un elemento in una posizione arbitraria
erase	Dato un iteratore che «punta» ad un elemento (o un range) elimina l'elemento corrispondente (o gli elementi corrispondenti)
resize	Dato un intero *n*, ridimensiona il vettore in modo che contenga *n* elementi.
clear	Rimuove tutti gli elementi

Vettori VS Liste

Da preferire i vettori quando:

- È necessario avere accesso diretto agli elementi
- È sufficiente inserire ed eliminare elementi alla fine del vettore

Da preferire le liste quando:

- Non è necessario avere accesso diretto agli elementi
- Non è sufficiente inserire ed eliminare elementi alla fine del vettore
- Non è noto a priori quanti elementi dovranno essere inseriti