Ad ogni istanza `c` di una classe contenitore della STL sono associati due tipi iteratore

`C::iterator`

`C::const_iterator`

Si usa **`iterator`** quando si necessita un accesso agli elementi del contenitore come lvalue, se basta un accesso come rvalue si usa un **`const_iterator`**.

Si tratta di cosiddetti ***iteratori bidirezionali***.

vector<int>::iterator

vector<int>::const_iterator

```
template <class T>
T& vector<T>::iterator::operator*() const;

template <class T>
vector<T>::iterator  vector<T>::begin();
template <class T>
vector<T>::iterator  vector<T>::end();


template <class T>                    Text
const T& vector<T>::const_iterator::operator*() const;

template <class T>
vector<T>::const_iterator  vector<T>::begin() const;
template <class T>
vector<T>::const_iterator  vector<T>::end() const;
```

```
vector<int> v(1); const vector<int> w(2);
iterator it = w.begin();       // ILLEGALE
*(w.begin())=0;                // ILLEGALE
```

Su ogni tipo iteratore di qualche istanza di contenitore `Cont<Tipo>::iterator` sono **sempre disponibili** le seguenti funzionalita`:

```
Cont<Tipo> x;
Cont<Tipo>::iterator i;

x.begin(); // iteratore che punta al primo elemento
x.end();   // particolare iteratore che non punta ad
           // alcun elemento, e' "un puntatore
           // all'ultimo elemento + 1"
*i;        // elemento puntato da i
i++; ++i;  // puntatore all'elemento successivo. Se
           // i punta all'ultimo elemento di x
           // allora ++i == x.end()
i--; --i;  // puntatore all'elemento precedente. Se
           // i punta al primo elemento di x
           // allora i-- è indefinito (x.begin()-1)
           // (v.end())-- punta all'ultimo elemento
```

Gli iteratori per i contenitori `vector` e `deque` (**contenitori ad accesso casuale**) permettono di avanzare e di retrocedere di un numero arbitrario di elementi. Sono inoltre disponibili gli operatori di confronto per questi iteratori. Si tratta degli ***iteratori ad accesso casuale***.

```
vector<Tipo> v;                  // oppure deque<Tipo>
vector<Tipo>::iterator i,j;
int k;

i+=k;
i-=k;
j=i+k;
j=i-k;
i < j;
i <= j;
i > j;
i >= j;
```