Itérateurs

16 septembre 2024 20:33

```
vector<int> v;
for( auto it = v.begin(); v != v.end(); ++v){
    int val = *it;
    //...
}
```

v.begin() retourne un objet de type

vector<int>::iterator

une autre classe servant à parcourir le vecteur.

- Les itérateurs servent à faire <u>abstraction</u> de la façon de parcourir une structure.
- Même utilisation pour vector, deque, list, set, etc.
- it peut être vu comme un "pointeur" sur un elt de la structure, qui peut
  - avancer ++
  - reculer --

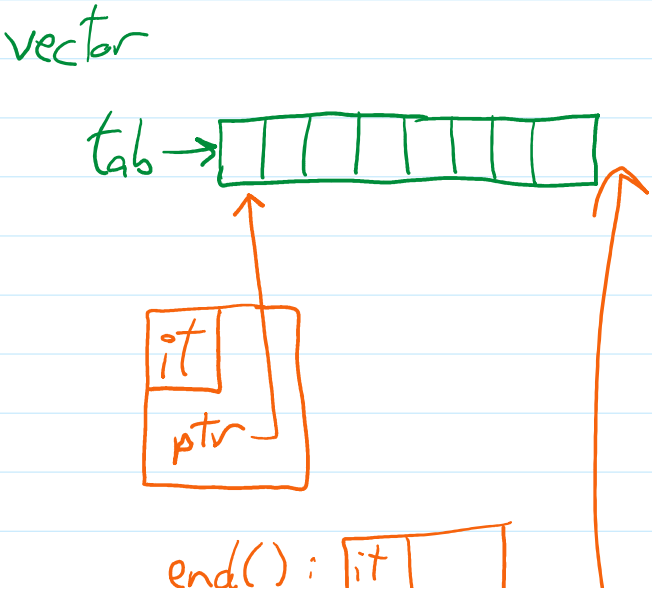
  end() un objet spécial qui "pointe" sur rien.
- Pas des pointeurs C++, mais des pointeurs sur structures

- Le for ci-haut est équivalent à :

```
vector::iterator it = v.begin();
while (it != v.end())
{
    int val = *it;
    ++it;
}
```

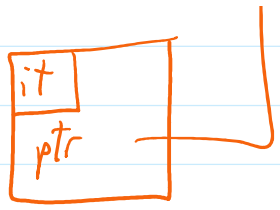
- Chaque SDD devrait avoir sa classe itérateur, qui définit begin, end, ++, --, ==, !=, *

-
```
class vector {
...
public:
...
    class iterator;   // définit vector::iterator
    iterator begin();
    iterator end();
...
};
```




```
template <typename TYPE>
class vector::iterator {
private:
    TYPE* ptr;   // pointeur vers entrée du vecteur
```

entrée du vecteur




```
public:
    TYPE& operator++();      //prefix++
    TYPE  operator++(int);   //postfix++
    //--, ==, !=
};
```

TYPE& operator++() : appelé via ++it
Incrémente l'objet it et le retourne.

TYPE operator++(int) : appelé via it++
Fait une copie de *this
Incrémente it
Retourne la copie.

```
auto x = ++it;   //x incrémenté
auto x = it++;   //x pas incrémenté
```

Pour que vector ait accès aux vars privées de iterator il faut que vector soit une

de iterator, il faut que vector soit une friend class de iterator

```
...
class iterator {
    friend class vector;
    ...
};
```

- Voir vector.h pour implémentation.