STL. Итераторы

1 Итератор

Объект, который синтаксически ведет себя как указатель (определены операторы ++, –, *, ->). Универсальный способ перебора элементов контейнеров в STL - перебор с помощью итератора. Все контейнеры имеют функции, которые возвращают итераторы на первый элемент и на элемент, следующий за последним:

Листинг 1: Example

```
vector< int > v;
vector< int >::iterator begin = v.begin();
vector< int >::iterator end = v.end();
```

Для vector и deque реализована арифметика, аналогичная арифметике указателей:

Листинг 2: Example

```
int i = *( v.begin() + 5 );
list< int > l;
list< int >::iterator it = l.begin();
for( ; it != l.end(); ++it )
    cout << *it; Получение объекта по итератору таким способом работает для любого типа
    контейнеров
list< int >::const_iterator cit = l.begin(); Такой итератор не позволяет менять
    данные, на которые он указывает
```

Итераторы позволяют задать позицию в контейнере:

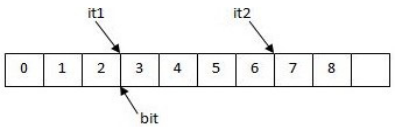

Рис. 1

Листинг 3: Example

```
Итератор
// it принадлежит контейнеру v
v.erase( it );// Удаление элемента, на который указывает итератор it
v.insert( it, 5 ); // Вставка элемента 5 на позицию, на которую указывает итератор it
v.insert( it1, 6 ); // Вставка элемента 6 между 2 и 3

*it1 == 3;
*it2 == 7;
*bit == 2; //bit - обратный итератор
// insert( it1, 6 ) и insert( bit, 6 ) дадут одинаковый результат
// Обратный итератор может быть получен следующим образом:
```

```
vector< int >::reverse_iterator ri_b = v.rbegin();
vector< int >::reverse_iterator ri_e = v.rend();
// В случае обратного итератора меняется направление прохода по контейнеру:
// оператор ++ передвигает итератор на позицию влево, а оператор -- - на позиую вправо
erase( it1, it2 ); // Два итератора задают множество элементов, которые будут удалены
insert( it, p, q ); // Итераторы p и q задают последовательность элементов
// другого контейнера, из которого идет вставка элементов
// Итераторы обязательно должны принадлежать одному контейнеру
```

Возможна инициализация нового контейнера элементами существующего. Последовательность задается с помощью двух итераторов:

Листинг 4: Example

```
vector< int > v; // Контейнер, элементами которого инициализируется новый контейнер
vector< double > w( v.begin(), v.end() );
```

2 Источники

- http://www.amse.ru/courses/cpp2/2011_02_21.html
- https://habr.com/post/122283/
- https://habr.com/post/265491/

Содержание