



Standard Template Library

ЛЕКЦИЯ №9

STL

1. Входит в поставку стандартных C++ компиляторов
2. Содержит контейнеры, структуры данных, итераторы и алгоритмы
3. Спецификация находится тут:
<http://www.cplusplus.com/reference>

Контейнеры

Контейнер — это объект, который может содержать в себе другие объекты. Существует несколько разных типов контейнеров. Например, класс `vector` определяет динамический массив, `deque` создает двунаправленную очередь, а `list` представляет связный список. Эти контейнеры называются **последовательными контейнерами** (sequence containers), потому что в терминологии STL последовательность — это линейный список.

STL также определяет **ассоциативные контейнеры** (associative containers), которые обеспечивают эффективное извлечение значений на основе ключей. Таким образом, ассоциативные контейнеры хранят пары “ключ/значение”. Примером может служить `map`. Этот контейнер хранит пары “ключ/значение”, в которых каждый ключ является уникальным. Это облегчает извлечение значения по заданному ключу.

Итераторы

Предоставляет способ последовательного доступа ко всем элементам контейнера, не раскрывая его внутреннего представления.

Зачем

- Составной объект, скажем список, должен предоставлять способ доступа к своим элементам, не раскрывая их внутреннюю структуру.
- Иногда требуется обходить список по-разному, в зависимости от решаемой задачи.
- Нужно, чтобы в один и тот же момент было определено несколько активных обходов списка.

Идея

Основная его идея в том, чтобы за доступ к элементам и способ обхода отвечал не сам список, а отдельный объект - итератор. В классе `Iterator` определен интерфейс для доступа к элементам списка. Объект этого класса отслеживает текущий элемент, то есть он располагает информацией, какие элементы уже посещались.

Требования к последовательному контейнеру

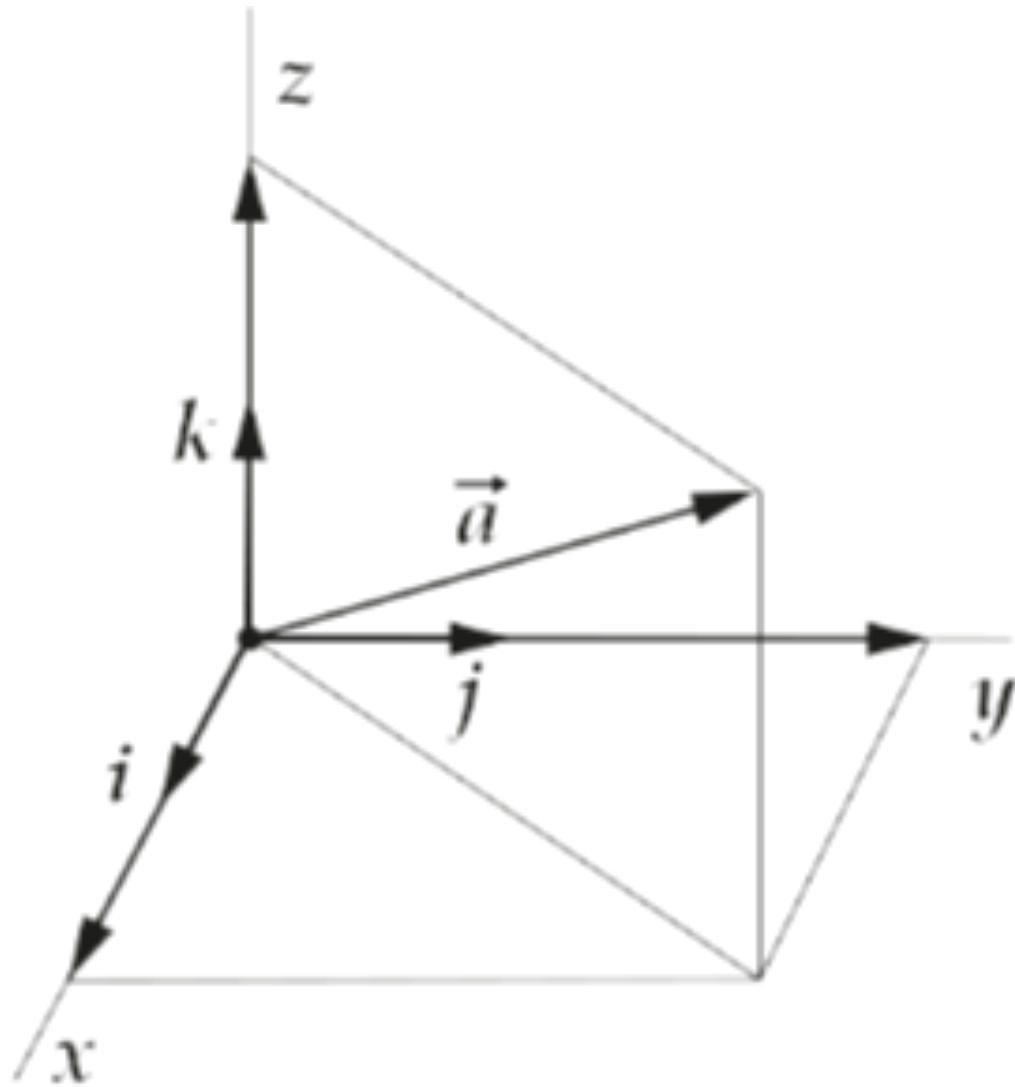
iterator begin()	Возвращает итератор, указывающий на первый элемент контейнера.
const_iterator begin() const	Возвращает константный итератор, указывающий на первый элемент контейнера.
iterator end()	Возвращает итератор, указывающий на позицию, следующую за последним элементом контейнера.
const_iterator end() const	Возвращает константный итератор, указывающий на позицию, следующую за последним элементом контейнера.
bool empty() const	Возвращает true, если контейнер пуст.
size_type size() const	Возвращает количество элементов, в текущий момент хранящихся в контейнере.
void swap (ContainerType c)	Обменивает между собой содержимое двух контейнеров.

<code>void clear()</code>	Удаляет все элементы из контейнера.
<code>iterator erase(iterator i)</code>	Удаляет элемент, на который указывает <code>i</code> . Возвращает итератор, указывающий на элемент, находящийся после удаленного.
<code>iterator erase(iterator start, iterator end)</code>	Удаляет элементы в диапазоне, указанном <code>start</code> и <code>end</code> . Возвращает итератор, указывающий на элемент, находящийся после последнего удаленного.
<code>iterator insert(iterator i, const T &val)</code>	Вставляет <code>val</code> непосредственно перед элементом, специфицированным <code>i</code> . Возвращает итератор, указывающий на вставленный элемент.
<code>void insert(iterator i, size_type num, const T &val)</code>	Вставляет <code>num</code> копий <code>val</code> непосредственно перед элементом, специфицированным <code>i</code> .
<code>template <class InIter> void insert(iterator i, InIter start, InIter end)</code>	Вставляет последовательность, определенную <code>start</code> и <code>end</code> , непосредственно перед элементом, специфицированным <code>i</code> .

Требования к последовательному контейнеру

Последовательные контейнеры в STL

1. `std::array`
2. `std::vector`
3. `std::deque`
4. `std::stack`
5. `std::queue`
6. `std::priority_queue`
7. `std::forward_list`
8. `std::list`



`std::vector` `vector.cpp`

1. аналог динамическому массиву Си
2. эмулирует расширяемость
3. добавление в начало неэффективно
4. данные лежат в непрерывной области памяти (в куче)
5. итераторы произвольного доступа
6. инвалидация итераторов почти всегда

Простейший итератор для RangeFor iterator.cpp

```
// for работает с итератором как с указателем!  
1.class IntIterator{  
2. int operator*( ) ;  
3. int operator->( ) ;  
4. bool operator!=(IntIterator const& other) const;  
5. IntIterator & operator++( ) ;  
6. }
```

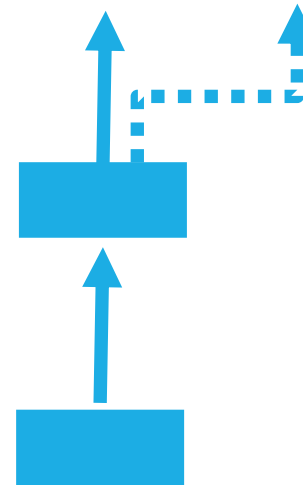
Итератор

Контейнер может иметь произвольную структуру и различные методы доступа:



Итератор указывает на элемент контейнера и знает как перейти к следующему элементу

Программе работает только с итератором и его интерфейсом (++)



iterator_traits

http://www.cplusplus.com/reference/iterator/iterator_traits/

Атрибут	Тип
difference_type	Тип для хранения значения результата вычитания двух итераторов (ptrdiff_t)
value_type	Тип, на который указывает итератор (обычно T – параметр шаблона)
pointer	Тип указателя на элемент контейнера (обычно T*)
reference	Тип ссылки на элемент шаблона (обычно T&)
iterator_category	Категория итератора: input iterator tag output iterator tag forward iterator tag bidirectional iterator tag random access iterator tag

сложный пример с итератором unique_iterator.cpp

```
class ListIterator{
private:
    List&    list;
    size_t   index;
    friend class List;
public:
    using difference_type = int ;
    using value_type = List::value_type;
    using reference = List::value_type& ;
    using pointer = List::value_type*;
    using iterator_category = std::forward_iterator_tag;

    ListIterator(List &l,int i) : list(l), index(i){}

    ListIterator& operator++(){
        ++index;
        return *this;
    }

    reference operator*(){
        return list[index];
    }

    pointer operator->(){
        return &list[index];
    }

    bool operator!=(const ListIterator& other){
        if(index!=other.index) return true;
        if(&list!=&(other.list)) return true;
        return false;
    }
};
```



Предопределенные итераторы
<http://www.cplusplus.com/reference/iterator/>

reverse_iterator

move_iterator

back_insert_iterator

front_insert_iterator

insert_iterator

istream_iterator

ostream_iterator

istreambuf_iterator

ostreambuf_iterator

std::back_insert_iterator back_insert.cpp

```
std::vector<int> foo;
std::vector<int> bar;
for (int i = 1; i <= 5; i++) {
    bar.push_back(i * 10);
}

std::back_insert_iterator< std::vector<int> > back_it(foo);
copy(bar.begin(), bar.end(), back_it);
```

Как устроен back_insert_iterator?

back_insert_iterator.cpp

```
template <class Container>
    class back_insert_iterator
    {
protected:
    Container* container;

public:
    typedef Container container_type;
    explicit back_insert_iterator (Container& x) : container(&x) {}

    // копирование значения
    back_insert_iterator<Container>&
    operator= (const typename Container::value_type& value)
    { container->push_back(value); return *this; }

    // перемещение значения
    back_insert_iterator<Container>&
    operator= (typename Container::value_type&& value)
    { container->push_back(std::move(value)); return *this; }

    // стандартный набор операторов
    back_insert_iterator<Container>& operator* ()
    { return *this; }
    back_insert_iterator<Container>& operator++ ()
    { return *this; }
    back_insert_iterator<Container> operator++ (int)
    { return *this; }
    };
```

istream_iterator,
insert_iterator,
ostream_iterator

//iostream.cpp

```
1. std::istream_iterator<double> eos; // end-of-stream
   iterator
2. std::istream_iterator<double> iit(std::cin); // stdin
   iterator
3. if (iit != eos) value1 = *iit;

4. std::vector<double> vec;
5. std::insert_iterator<std::vector<double>>
   insert_it(vec, vec.begin());
6. std::copy(iit, eos, insert_it);

7. std::ostream_iterator<double> out(std::cout, " ");
8. std::copy(vec.begin(), vec.end(), out);
```


Итераторы - ИТОГО

1. Итераторы – хранят ссылку на контейнер и даже на определенный элемент в контейнере
2. Итераторы – знают о структуре контейнера
3. Итераторы – предоставляют однотипный интерфейс по доступу к любому контейнеру
4. Итераторы – могут не только получать данные из контейнера, но могут записывать данные в контейнер (зависит от структуры контейнера)
5. Итераторы не могут менять размер контейнера



Спасибо!

ВСЕ ИДЕМ НА ПЕРЕРЫВ