STL

Тема 1. Итераторы

Итераторы (iterator)

- Указателеобразные объекты
- Связь между алгоритмами и контейнерами
- Категории
  - Выходные
  - Выходные
  - Однонаправленные
  - Двунаправленные
  - Произвольного доступа
- Диапазон итераторов [first,last)
  - Корректный диапазон

Входной итератор (InputIterator)

```
template <typename InputIterator, typename T>
InputIterator find(     InputIterator first,
                        InputIterator last,
                        const T& value)
{
    while (first 1= last && *first != value)
            ++first;
    return first;
}
```

Входной итератор (InputIterator)

Требования:

- operator !=
- ++iterator и iterator++
- value = *iterator
- operator ==
- O(1)

Выходной итератор (OutputIterator)

```
template <typename InputIterator, typename OutputIterator>
OutputIterator copy(        InputIterator first,
                            InputIterator last,
                            OutputIterator result
                          )
{
    while (first != last)
    {
            *result = *first;
            ++first;
            ++result;
     }
    return result;
}
```

Выходной итератор (OutputIterator)

Требования:

- *iterator = value
- ++iterator и iterator++
- O(1)

Однонаправленные итераторы (Forward Iterator)

- Входной итератор
- Выходной итератор
- Сохранение для последующего использования

```
template <typename ForwardIterator, typename T>
void replace(ForwardIterator first, ForwardIterator last, const T& x, const T& y)
{
        while (first != last)
        {
                if (*first == x)
                        *first = y;
                ++first;
        }
}
```

Двунаправленные итераторы

- Однонаправленный
- operator--

```
int a[10] = {12, 3, 25, 7, 11, 213, 7, 123, 29, -3l};

reverse(&a[0],&a[10]);

for(int i = 0; i < 10; ++i)
{
        cout << a[i] << endl;
}

list<int> list1(&a[0], &a[10]);
reverse(list1.begin(),list1.end());
```

Итераторы с произвольным доступом

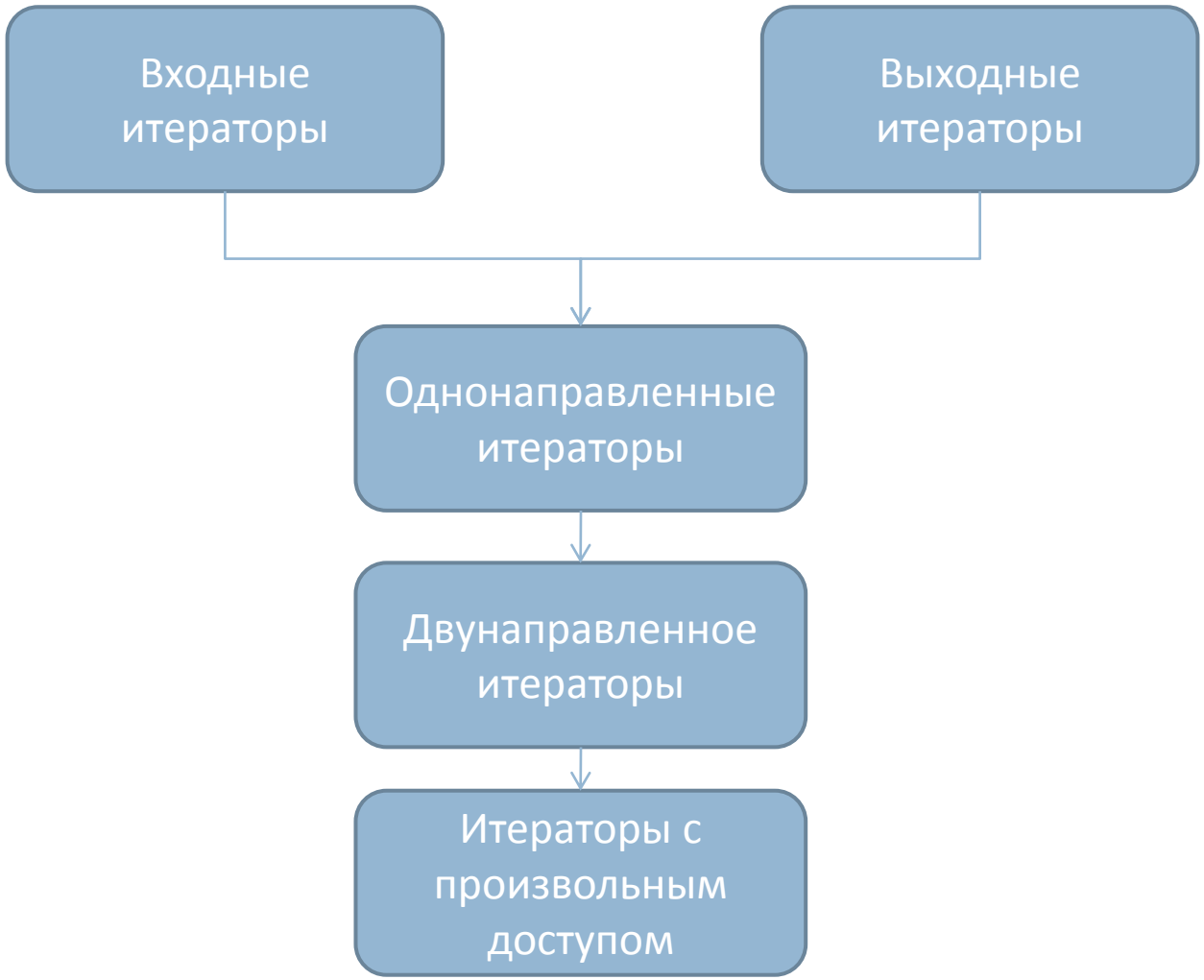
```
vector<int> v;
// …. Заполнение вектора
bool b = binary_search(v.begin(),v.end(), 6);
```

Итераторы с произвольным доступом

- Двунаправленный итератор
- Достижение любой позиции за O(1)

  Пусть r и s – итераторы с произвольным доступом, n – целое число , тогда:
  - r+n, n+r, r-n
  - r[n]=*(r+n)
  - r+=n, r-=n
  - r-s ->int
  - r<s, r>s, r<=s, r>=s -> bool

Иерархия итераторов STL



Итераторы

- Описание контейнеров включает описание предоставляемых ими итераторов
- Описание обобщенных алгоритмов включает описание категорий итераторов с которыми они работают

Вывод:

Интерфейсы контейнеров и алгоритмов STL спроектированы так, чтобы поддерживать эффективные комбинации и препятствовать неэффективным

Итератор вставки

- back_insert_iterator<Container> (push_back)
- front_insert_iterator<Container> (push_front)
- insert_iterator<Container> (insert)

```
vector<int> vec;
list<int> l(200,1);

copy(l.begin(),l.end(),back_insert_iterator<vector<int>>(vec));
```

Потоковые итераторы

- istream_iterator
  - Ввод
  - Входной, но не выходной итератор
- ostream_iterator
  - Вывод
  - Выходной, но не входной итератор

```
vector<int> v(10,1);
list<int> l;

merge(   v.begin(), v.end(),
          istream_iterator<int>(cin),istream_iterator<int>(),
         back_insert_iterator<list<int>>(l));

merge(   v.begin(), v.end(),
         l.begin(), l.end(),
         ostream_iterator<int>(cout," "));
```

iterator / const_iterator

```
const_iterator i = …..
*i=10;  // !!Ошибка
```

```
const vector<int> v(100,0);

//vector<int>::iterator i = v.begin();  // !!Ошибка
vector<int>::const_iterator i = v.begin();
```

Итератор

Контейнер	Итератор	Тип
T a[n]	T*	Изм. Произв доступ
T a[n]	const T*	Конст. ,произв доступ
vector<T>	vector<T>::iterator	Изм. Произв доступ
vector<T>	vector<T>::const_iterator	Конст. ,произв доступ
deque<T>	deque<T>::iterator	Изм. Произв доступ
deque<T>	deque<T>::const_iterator	Конст. ,произв доступ
list<T>	list<T>::iterator	Изм., двунаправленный
list<T>	list<T>::const_iterator	Конст., двунаправленный

Итераторы

Контейнер	Итератор	Тип
set<t>	set<t>::iterator	Конст., двунапр.
set<t>	set<t>::const_iterator	Конст., двунапр.
multiset<T>	multiset<T>::iterator	Конст., двунапр.
multiset<T>	multiset<T>::const_iterator	Конст., двунапр.
map<Key,T>	map<Key,T>::iterator	Изм., двунаправленный
map<Key,T>	map<Key,T>::const_iterator	Конст., двунапр.
multimap<Key,T>	multimap<Key,T>::iterator	Изм., двунаправленный
multimap<Key,T>	multimap<Key,T>::const_iterator	Конст., двунапр.

Задание

- Заполнить vector<int> из файла, найти сколько раз в нем встречается значение 1
- Скопировать vector<int> в list<int>, исключая значения равные 1
- Заполнить list<int> из файла, скопировать в vector<int> в обратном порядке
- Заполнить вектор из list<int>, считанного из файла и данных введенных в консоль
- Написать обобщенную функцию, вывода list<T> в файл