

**\_\_asc\_iterator.hpp**

```
1  #ifndef __ASC_ITERATOR_HPP__
2  #define __ASC_ITERATOR_HPP__
3
4  #include "franklist.h"
5
6  using namespace vhuk;
7
8  template <typename T>
9  FrankList<T>::asc_iterator::asc_iterator(const base_iterator& rhv)
10
11      : const_asc_iterator(rhv){}
12
13  template <typename T>
14  FrankList<T>::asc_iterator::asc_iterator(base_iterator&& rhv)
15
16      : const_asc_iterator(rhv){}
17
18  template <typename T>
19  typename FrankList<T>::reference FrankList<T>::asc_iterator::operator*()
20  {
21      return (const_cast<reference>((static_cast<const_asc_iterator*>(this)→
22      operator*())));
23  }
24
25  template <typename T>
26  typename FrankList<T>::pointer FrankList<T>::asc_iterator::operator→()
27  {
28      return (const_cast<pointer>((static_cast<const_asc_iterator*>(this)→operator→
29      ()))));
30  }
31
32  template <typename T>
33  const typename FrankList<T>::asc_iterator& FrankList<T>::asc_iterator::operator=
34  (const base_iterator& rhv)
35  {
36      *(static_cast<const_asc_iterator*>(this)) = rhv;
37
38      return (*this);
39  }
40
41  template <typename T>
42  const typename FrankList<T>::asc_iterator& FrankList<T>::asc_iterator::operator=
43  (base_iterator&& rhv)
44  {
45      *(static_cast<const_asc_iterator*>(this)) = rhv;
46
47      return (*this);
48  }
49
50  template <typename T>
51  FrankList<T>::asc_iterator::asc_iterator(Node* ptr)
52
53      : const_asc_iterator{ptr}{}
54
55  #endif // __ASC_ITERATOR_HPP__
```