STL : <algorithm>

**find**

template <class InputIterator, class T>

InputIterator find (InputIterator first, InputIterator last, const T& val);

Find value in range

Returns an iterator to the first element in the range [first,last) that compares equal to *val*. If no such element is found, the function returns *last*.

**Example**

|  |
| --- |
| *#include <iostream>*  *#include <algorithm>*  *#include <vector>*  *int* main ()  {  *int* myints[] = { 10, 20, 30 ,40 };  *int* \* p;  p = find (myints,myints+4,30); *// pointer to array element:*  ++p;  cout << "The element following 30 is " << \*p << '\n';  vector<*int*> myvector (myints,myints+4);  vector<*int*>::iterator it;  it = find (myvector.begin(), myvector.end(), 30); *// iterator to vector element:*  ++it;  cout << "The element following 30 is " << \*it << '\n';  *return* 0;  } |

Output:

|  |
| --- |
| The element following 30 is 40  The element following 30 is 40 |

**Complexity**

Up to linear in the [distance](http://www.cplusplus.com/distance) between *first* and *last*: Compares elements until a match is found.

**count**

template <class InputIterator, class T>

Count appearances of value in range

**Return value**

The number of elements in the range [first,last) that compare equal to *val*.

**Example**

|  |  |
| --- | --- |
|  | *#include <iostream>*  *#include <algorithm>*  *#include <vector>*  *int* main ()  {  *int* myints[] = {10,20,30,30,20,10,10,20};*//counting elements*  *int* mycount = count (myints, myints+8, 10);  cout << "10 appears " << mycount << " times.\n";  *// counting elements in container:*  vector<*int*> myvector (myints, myints+8);  mycount = count (myvector.begin(), myvector.end(), 20);  cout << "20 appears " << mycount << " times.\n";  *return* 0;  } |

Output:

|  |
| --- |
| 10 appears 3 times.  20 appears 3 times. |

**Complexity**

Linear in the [distance](http://www.cplusplus.com/distance) between *first* and *last*:

**Swap**

template <class T> void swap (T& a, T& b);

Exchange values of two objects

Notice how this function involves a copy construction and two assignment operations, which may not be the most efficient way of swapping the contents of classes that store large quantities of data, since each of these operations generally operate in linear time on their size.

**Example**

|  |  |
| --- | --- |
|  | *#include <iostream>*  *#include <algorithm>*  *#include <vector>*  *int* main ()  {  *int* x=10, y=20;  swap(x,y);  vector<*int*> foo (4,x), bar (6,y);  vector<*int*>::iterator it  swap(foo,bar);  cout << "foo contains:";  *for* (it=foo.begin(); it!=foo.end(); ++it)  cout << ' ' << \*it;  std::cout << '\n';  *return* 0;  } |

Output:

|  |
| --- |
| foo contains: 10 10 10 10 10 10 |

**Complexity**

**Non-array:** Constant: Performs exactly one construction and two assignments (although notice that each of these operations works on its own complexity).  
**Array:** Linear in *N*: performs a swap operation per element.