STL中的ALGORITHM库

By Fancy

STL算法库的常用函数

- copy
- fill
- lower_bound
- make_heap
- max / min
- max_element / min_element
- merge
- next_permutation
- nth_element

STL算法库的常用函数

- partition
- random_shuffle
- reverse
- rotate
- sort
- stable_sort
- swap
- unique
- upper_bound

STL算法库的常用函数

* copy:

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

```
int myints[]={10,20,30,40,50,60,70};
std::vector<int> myvector (7);
std::copy ( myints, myints+7, myvector.begin() );
```

STL算法库的常用函数

* fill:

```
template <class ForwardIterator, class T>
  void fill (ForwardIterator first, ForwardIterator last, const T& val)
{
  while (first != last) {
    *first = val;
    ++first;
  }
}
```

```
std::vector<int> myvector (8);
// myvector: 0 0 0 0 0 0 0 0
std::fill (myvector.begin(),myvector.begin()+4,5);
// myvector: 5 5 5 5 0 0 0 0
std::fill (myvector.begin()+3,myvector.end()-2,8);
// myvector: 5 5 5 8 8 8 0 0
```

STL算法库的常用函数

* random_shuffle:

```
template <class RandomAccessIterator, class RandomNumberGenerator>
  void random_shuffle (RandomAccessIterator first, RandomAccessIterator last,
                       RandomNumberGenerator& gen)
{
  iterator_traits<RandomAccessIterator>::difference_type i, n;
  n = (last-first);
  for (i=n-1; i>0; --i) {
    swap (first[i],first[gen(i+1)]);
  }
}
```

```
int myrandom (int i) { return std::rand()%i;}

// using built-in random generator:
std::random_shuffle ( myvector.begin(), myvector.end() );
// using myrandom:
std::random_shuffle ( myvector.begin(), myvector.end(), myrandom);
```

STL算法库的常用函数

* nth_element:

```
default (1)   template <class RandomAccessIterator>
                void nth_element (RandomAccessIterator first, RandomAccessIterator nth,
                                  RandomAccessIterator last);
custom (2)    template <class RandomAccessIterator, class Compare>
                void nth_element (RandomAccessIterator first, RandomAccessIterator nth,
                                  RandomAccessIterator last, Compare comp);
```

```
bool myfunction (int i,int j) { return (i<j); }

int main () {
  std::vector<int> myvector;
  for (int i=1; i<10; i++) myvector.push_back(i);    // 1 2 3 4 5 6 7 8 9
  std::random_shuffle (myvector.begin(), myvector.end());
  std::nth_element (myvector.begin(), myvector.begin()+5, myvector.end());
  std::nth_element (myvector.begin(), myvector.begin()+5, myvector.end(), myfunction);
```

STL算法库的常用函数

* sort / stable_sort:

```
default (1)   template <class RandomAccessIterator>
                void sort (RandomAccessIterator first, RandomAccessIterator last);
custom (2)    template <class RandomAccessIterator, class Compare>
                void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);
```

```
bool myfunction (int i,int j) { return (i<j); }
struct myclass {
  bool operator() (int i,int j) { return (i<j);}
} myobject;
int main () {
  int myints[] = {32,71,12,45,26,80,53,33};
  std::vector<int> myvector (myints, myints+8);               // 32 71 12 45 26 80 53 33
  // using default comparison (operator <):
  std::sort (myvector.begin(), myvector.begin()+4);           //(12 32 45 71)26 80 53 33
  // using function as comp
  std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)
  // using object as comp
  std::sort (myvector.begin(), myvector.end(), myobject);     //(12 26 32 33 45 53 71 80)
  std::sort (myints+1, myints+5);                             //32 (12 26 45 71) 80 53 33
```

STL算法库的常用函数

* unique:

```
template <class ForwardIterator>
  ForwardIterator unique (ForwardIterator first, ForwardIterator last)
{
  if (first==last) return last;

  ForwardIterator result = first;
  while (++first != last)
  {
    if (!(*result == *first))  // or: if (!pred(*result,*first)) for version (2)
      *(++result)=*first;
  }
  return ++result;
}
```

```
bool myfunction (int i, int j) {  return (i==j); }
int main () {
  int myints[] = {10,20,20,20,30,30,20,20,10};           // 10 20 20 20 30 30 20 20 10
  std::vector<int> myvector (myints,myints+9);
  // using default comparison:
  std::vector<int>::iterator it;
  it = std::unique (myvector.begin(), myvector.end());   // 10 20 30 20 10 ?  ?  ?  ?
  myvector.resize( std::distance(myvector.begin(),it) ); // 10 20 30 20 10
```

STL算法库的常用函数

* lower_bound / upper_bound:

```
int myints[] = {10,20,30,30,20,10,10,20};
std::vector<int> v(myints,myints+8);              // 10 20 30 30 20 10 10 20
std::sort (v.begin(), v.end());                   // 10 10 10 20 20 20 30 30
std::vector<int>::iterator low,up;
low=std::lower_bound (v.begin(), v.end(), 20); //          ^
up= std::upper_bound (v.begin(), v.end(), 20); //                   ^
std::cout << "lower_bound at position " << (low- v.begin()) << '\n';
std::cout << "upper_bound at position " << (up - v.begin()) << '\n';
```

STL算法库的常用函数

* max / min / swap:

```
cout << max(1, 2) << endl;
cout << min(3, 4) << endl;
int a = 5, b = 10;
swap(a, b);
cout << a << ' ' << b << endl;
```

作业和参考

* HW8-1
* http://www.cplusplus.com/reference/algorithm/