/\*

找相邻且相同的元素

bool myfunction (int i, int j) {

return (i==j);

}

vector<int>::iterator it = adjacent\_find (myvector.begin(), myvector.end());

it = adjacent\_find (++it, myvector.end(), myfunction);

二分查找

int myints[] = {1,2,3,4,5,4,3,2,1};

vector<int> v(myints,myints+9);

binary\_search (v.begin(), v.end(), 3)；

统计总共有多少个目标值

mycount = (int) count (myvector.begin(), myvector.end(), 20);（20是目标值）

比较是否相同

int myints[] = {20,40,60,80,100}; // myints: 20 40 60 80 100

vector<int> myvector (myints,myints+5); // myvector: 20 40 60 80 100

equal (myvector.begin(), myvector.end(), myints)

找第一个目标的位置，反悔指针

it = find (myvector.begin(), myvector.end(), 30);

++it;

cout << "The element following 30 is " << \*it << endl;

快速输出

void myfunction (int i) {

cout << " " << i;

}

for\_each(myvector.begin(),mevector.end(),myfunction);

归并排序

int first[] = {5,10,15,20,25};

int second[] = {50,40,30,20,10};

vector<int> v(10);

vector<int>::iterator it;

sort (first,first+5);

sort (second,second+5);

copy (first,first+5,v.begin());

copy (second,second+5,v.begin()+5);

inplace\_merge (v.begin(),v.begin()+5,v.end());

找目标元素第一次和最后一次出现的位置

int myints[] = {10,20,30,30,20,10,10,20};

vector<int> v(myints,myints+8); // 10 20 30 30 20 10 10 20

vector<int>::iterator low,up;

sort (v.begin(), v.end()); // 10 10 10 20 20 20 30 30

low=lower\_bound (v.begin(), v.end(), 20); // ^

up= upper\_bound (v.begin(), v.end(), 20); //

最大值

\*max\_element(myints,myints+7)

struct myclass {

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

} myobj;

\*min\_element(myints,myints+7,myobj)

\*max\_element(myints,myints+7,myobj)

归并排序

int first[] = {5,10,15,20,25};

int second[] = {50,40,30,20,10};

vector<int> v(10);

vector<int>::iterator it;

sort (first,first+5);

sort (second,second+5);

merge (first,first+5,second,second+5,v.begin());

部分排序

partial\_sort (myvector.begin(), myvector.begin()+5, myvector.end());

划分

bool IsOdd (int i) { return (i%2)==1; }

bound = partition (myvector.begin(), myvector.end(), IsOdd);

// print out content:

cout << "odd members:";

for (it=myvector.begin(); it!=bound; ++it)

cout << " " << \*it;

cout << "\neven members:";

for (it=bound; it!=myvector.end(); ++it)

cout << " " << \*it;

替换

replace (myvector.begin(), myvector.end(), 20, 99); 20换成99

.

int myints[] = { 10, 20, 30, 30, 20, 10, 10, 20 };

vector<int> myvector (8);

replace\_copy (myints, myints+8, myvector.begin(), 20, 99);

·

bool IsOdd (int i) { return ((i%2)==1); }

replace\_copy\_if (first.begin(), first.end(), second.begin(), IsOdd, 0);

颠倒

reverse(myvector.begin(),myvector.end());

reverse\_copy (myints, myints+9, myvector.begin());

移动

rotate(myvector.begin(),myvector.begin()+3,myvector.end());

rotate\_copy(myints,myints+3,myints+7,myvector.begin());

找差集

sort (first,first+5); // 5 10 15 20 25

sort (second,second+5); // 10 20 30 40 50

it=set\_difference (first, first+5, second, second+5, v.begin());

稳定排序

stable\_sort (myvector.begin(), myvector.end(), compare\_as\_ints);

聚合

it=set\_union (first, first+5, second, second+5, v.begin());\*/