Sample of STL

STL 范例(一)

容器部分

Vector

constructors

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;
int main ()
{
    string str[]={"Alex","John","Robert"};
    // empty vector object
    vector<int> v1;
    // creates vector with 10 empty elements
    vector<int> v2(10);
    // creates vector with 10 elements,
    // and assign value 0 for each
    vector<int> v3(10,0);
    // creates vector and assigns
    // values from string array
    vector<string> v4(str+0,str+3);
    vector<string>::iterator sIt = v4.begin();
    while ( sIt != v4.end() )
        cout << *sIt++ << " ";
    cout << endl;
    // copy constructor
    vector<string> v5(v4);
    for ( int i=0; i<3; i++ )
        cout << v5[i] << " ";
    cout << endl;
    return 0;
}
OUTPUT:
//  Alex John Robert
//  Alex John Robert
```

assign

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;
```

```
int main ()
{
    int ary[]={1,2,3,4,5};
    vector<int> v;
    // assign to the "v" the contains of "ary"
    v.assign(ary,ary+5);
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    // replace v for 3 copies of 100
    v.assign(3,100);
    copy(v.begin(),v.end(),
            ostream_iterator(cout," "));
    cout << endl;
    return 0;
}
OUTPUT:
// 1 2 3 4 5
// 100 100 100
```

at

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> v(3,0);
    v[0] = 100;
    v.at(1) = 200;
    for ( int i=0; i<3; i++ )
        cout << v.at(i) << " ";
    cout << endl;
    return 0;
}
OUTPUT:
// 100 200 0
```

back

```
#include <iostream>
#include <vector>
```

```
#include <string>
#include <iterator>
using namespace std;
template<class T, class D>
class Member
{
    public:
        Member(T t, D d) : name(t), sal(d) {}
        void print();
    private:
        T name;
        D sal;
};
template<class T, class D>
void Member::print()
{
    cout << name << "  " << sal << endl;
}
//=====================================
int main ()
{
    typedef Member<string,double> M;
    vector<M> v;
    v.push_back(M("Robert",60000));
    v.push_back(M("Linda",75000));
    vector<M>::iterator It = v.begin();
    cout << "Entire vector:" << endl;
    while ( It != v.end() )
        (It++)->print();
    cout << endl;
    cout << "Return from back()" << endl;
    v.back().print();
    return 0;
}
```

OUTPUT:

```
// Entire vector:
// Robert  60000
// Linda  75000
//
// Return from back()
// Linda  75000
```

begin

```
#include <iostream>
#include <vector>
#include <iterator>
#include <numeric>
using namespace std;

int main ()
{
    vector<int> v(5);
    iota(v.begin(),v.end(),1);
    vector<int>::iterator It = v.begin();
    while ( It != v.end() )
        cout << *It++ << " ";
    cout << endl;
    // third element of the vector
    It = v.begin()+2;
    cout << *It << endl;
    return 0;
}
OUTPUT:
// 1 2 3 4 5
// 3
```

capacity

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> v(10);
    cout << "Size     of v = "
         << v.size() << endl;
    cout << "Capacity of v = "
         << v.capacity() << endl;
    v.resize(100);
    cout << "After resizing:" << endl;
    cout << "Size     of v = "
         << v.size() << endl;
    cout << "Capacity of v = "
         << v.capacity() << endl;
    return 0;
```

```
}
OUTPUT:
// Size of v = 10
// Capacity of v = 10
// After resizing:
// Size of v = 100
// Capacity of v = 100

clear
```

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//==============================
int main ()
{
    vector<int> v(10);
    Print<int> print;
    fill(v.begin(),v.end(),5);
    cout << "Vector v : ";
    for_each(v.begin(),v.end(),print);
    cout << endl;
    cout << "Size of v = " << v.size()
         << endl;
    cout << "v.clear" << endl;
    v.clear();
    cout << "Vector v : ";
    for_each(v.begin(),v.end(),print);
    cout << endl;
    cout << "Size of v = " << v.size()
         << endl;
    cout << "Vector v is ";
    v.empty() ? cout << "" : cout << "not ";
```

```
    cout << "empty" << endl;

    return 0;
}
// Vector v : 5 5 5 5 5 5 5 5 5 5
// Size of v = 10
// v.clear
// Vector v :
// Size of v = 0
// Vector v is empty
```

empty

```
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
    vector<int> v;
    cout << "Vector is ";
    v.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;
    v.push_back(100);
    cout << "Vector is ";
    v.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;
    return 0;
}
// Vector is empty
// Vector is not empty
```

end

```
#include <iostream>
#include <vector>
#include <iterator>
#include <numeric>
using namespace std;
int main ()
{
    vector<int> v(5);
```

```
    iota(v.begin(),v.end(),1);
    vector<int>::iterator It = v.begin();
    while ( It != v.end() )
        cout << *It++ << " ";
    cout << endl;
    // last element of the vector
    It = v.end()-1;
    cout << *It << endl;
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5
// 5
```

erase

```
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
using namespace std;

int main ()
{
    vector<int> v(10);
    vector<int>::iterator It;
    for ( int i=0; i<10; i++ )
        v[i] = i+1;
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    It = v.begin()+2;
    // remove third element
    v.erase(It);
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    It = v.begin();
    // remove 2 elements from beginning fo v
    v.erase(It,It+2);
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
```

```
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5 6 7 8 9 10
// 1 2 4 5 6 7 8 9 10
// 4 5 6 7 8 9 10
```

front

```
#include <iostream>
#include <vector>
#include <string>
#include <iterator>
using namespace std;
template<class T, class D>
class Member
{
    public:
        Member(T t, D d) : name(t), sal(d) {}
        void print();
    private:
        T name;
        D sal;
};

template<class T, class D>
void Member::print()
{
    cout << name << "  " << sal << endl;
}
//=====================================
int main ()
{
    typedef Member<string,double> M;
    vector<M> v;
    v.push_back(M("Linda",75000));
    v.push_back(M("Robert",60000));
    vector<M>::iterator It = v.begin();
    cout << "Entire vector:" << endl;
    while ( It != v.end() )
        (It++)->print();
    cout << endl;
    cout << "Return from front()" << endl;
    v.front().print();
```

```
    return 0;
}
OUTPUT:
// Entire vector:
// Linda  75000
// Robert  60000
//
// Return from front()
// Linda  75000
insert
```

```
#include <iostream>
#include <vector>
#include <iterator>
#include <algorithm>
using namespace std;
template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
             cout << t << " ";
        }
};
//============================
int main ()
{
    int ary[5];
    fill(ary,ary+5,1);
    vector<int> v;
    vector<int>::iterator It;
    Print<int> print;
    copy(ary,ary+5,
            back_inserter(v));
    cout << "vector v                : ";
    for_each(v.begin(),v.end(),print);
    cout << endl;
    It = v.begin();
    // insert value "5" at the position "It"
    cout << "v.insert(It,5)          : ";
    v.insert(It,5);
    for_each(v.begin(),v.end(),print);
```

```
    cout << endl;

    // insert range ary+2 - ary+5 at the position "It"
    It = v.begin()+5;
    cout << "v.insert(It,ary+2,ary+5 : ";
    v.insert(It,ary+2,ary+5);
    for_each(v.begin(),v.end(),print);
    cout << endl;
    // insert 2 value of "20" at the position "It"
    It = v.end()-2;
    cout << "v.insert(It,2,20)       : ";
    v.insert(It,2,20);
    for_each(v.begin(),v.end(),print);
    cout << endl;
    return 0;
}
OUTPUT:
// vector v                 : 1 1 1 1 1
// v.insert(It,5)           : 5 1 1 1 1 1
// v.insert(It,ary+2,ary+5 : 5 1 1 1 1 1 1 1 1
// v.insert(It,2,20)        : 5 1 1 1 1 1 1 20 20 1 1
```

max_size

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> v(10);
    cout << "Size of v     = "
         << v.size() << endl;
    cout << "Max_size of v = "
         << v.max_size() << endl;
    return 0;
}
OUTPUT:
// Size of v     = 10
// Max_size of v = 1073741823
```

pop_back

```
#include <iostream>
```

```
#include <vector>
#include <algorithm>
using namespace std;
template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//============================
int main ()
{
    vector<int> v;
    Print<int> print;
    for ( int i=0; i<5; i++ )
        v.push_back(i+1);
    while ( !v.empty() )
    {
        for_each(v.begin(),v.end(),print);
        cout << endl;
        v.pop_back();
    }
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5
// 1 2 3 4
// 1 2 3
// 1 2
// 1
```

push_back

```
#include <iostream>
#include <vector>
#include <string>
#include <iterator>
using namespace std;
template <class T>
class Name
```

```
{
    public:
        Name(T t) : name(t) {}
        void print()
        {
            cout << name << " ";
        }
    private:
        T name;
};
//============================
int main ()
{
    typedef Name<string> N;
    typedef vector<N> V;
    V v;
    N n1("Robert");
    N n2("Alex");
    v.push_back(n1);
    v.push_back(n2);
    // unnamed object of the type Name
    v.push_back(N("Linda"));
    V::iterator It = v.begin();
    while ( It != v.end() )
        (It++)->print();
    cout << endl;
    return 0;
}
```

OUTPUT:

```
// Robert Alex Linda
```

rbegin and rend

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include <algorithm>
#include <iterator>
using namespace std;

class ID
{
```

```
    friend bool operator < ( const ID&, const ID& );
        public:
        ID(string name,int score) : name(name), score(score) {}
        void display ()
        {
            cout.setf(ios::left);
            cout << setw(3) << score << name << endl;
        }
    private:
        string name; int score;
};
//-------------------------------------------------------
// comperation function for sorting
bool operator < ( const ID& a, const ID& b )
{
    return a.score < b.score;
}
//-------------------------------------------------------
typedef vector<ID> Vector; // new name for existing datatype

int main ()
{
    Vector v;
    Vector::iterator Iter;
    v.push_back(ID("Smith A",96));
    v.push_back(ID("Amstrong B.",91));
    v.push_back(ID("Watson D.",82));

    for ( Iter = v.begin(); Iter != v.end(); Iter++ )
        Iter->display();
    sort(v.begin(),v.end()); // sort algorithm
    cout << endl << "Sorted by Score" << endl;
    cout << "===============" << endl;
    for ( Iter = v.begin(); Iter != v.end(); Iter++ )
        Iter->display();

    cout << endl << "Reverse output" << endl;
    cout << "===============" << endl;

    Vector::reverse_iterator r = v.rbegin();
    while ( r != v.rend() )
        cout << r->display();
    cout << endl;
    return 0;
```

```
}
OUTPUT:
// 96 Smith A.
// 91 Amstrong B.
// 82 Watson D.
//
// Sorted by Score
// ===============
// 82 Watson D.
// 91 Amstrong B.
// 96 Smith A.
//
// Reverse output
// ===============
// 96 Smith A.
// 91 Amstrong B.
// 82 Watson D.
```

reserve

```
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
    vector<int> v(5,0); // 5 elements, each - value 0
    /*-------------------------------------------*/
    cout << "Size of v  = " << v.size() << endl;
    cout << "Capacity v = " << v.capacity() << endl;
    cout << "Value of each element is - ";
    for ( int i = 0; i < v.size(); i++ )
        cout << v[i] << "  ";
    cout << endl;
    v[0] = 5;       // new value for first element
    v[1] = 8;
    v.push_back(3); // creates new (6th) element of vector,
    v.push_back(7); // automatically increases size
    cout << endl;   // capacity of vector v
    cout << "Size of v  = " << v.size() << endl;
    cout << "Capacity v = " << v.capacity() << endl;
    cout << "Value of each element is - ";
    for ( int i = 0; i < v.size(); i++ )
```

```
        cout << v[i] << "  ";
    cout << endl << endl;

    v.reserve(100); // increase capacity to 100
    cout << "Size of v1_int  = " << v.size() << endl;
    cout << "Capacity v1_int = " << v.capacity() << endl;
    int size = sizeof(v); // how big is vector itself
    cout << "sizeof v   = " << size << endl;

    return 0;
}
OUTPUT:
// Size of v  = 5
// Capacity v = 5
// Value of each element is - 0  0  0  0  0
//
// Size of v  = 7
// Capacity v = 10
// Value of each element is - 5  8  0  0  0  3  7
//
// Size of v  = 7
// Capacity v = 100
// sizeof v   = 12
```

resize

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;

int main ()
{
    vector<int> v(5);
    for ( int i=0; i<5; i++ )
        v[i] = i*2;
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    v.resize(7,100);
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
```

```
    cout << endl;

    v.resize(4);
    copy(v.begin(),v.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    return 0;
}
OUTPUT:
// 0 2 4 6 8
// 0 2 4 6 8 100 100
// 0 2 4 6
```

size

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;
template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//============================
int main ()
{
    vector<char> v(5);
    Print<char> print;
    cout << "Size of v = " << v.size() << endl;
    fill(v.begin(),v.end(),'*');
    for_each(v.begin(),v.end(),print);
    cout << endl;
    for ( int i=0; i < v.size(); i++ )
        cout << v[i] << " ";
    cout << endl;
    for ( int i=0; i<5; i++ )
    {
```

```
        cout << "Size of v = ";
        for_each(v.begin(),v.end(),print);
        cout << endl;
        v.pop_back();
    }
    return 0;
}
```

OUTPUT:

```
// Size of v = 5
// * * * * *
// * * * * *
// Size of v = * * * * *
// Size of v = * * * *
// Size of v = * * *
// Size of v = * *
// Size of v = *
```

swap

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//============================
int main ()
{
    int ary[] = {1,2,3,4,5,6,7,8,9,10};
    Print print;
    vector<int> v1(ary,ary+7);
    vector<int> v2(ary+7,ary+10);
    cout << "Vector v1 : ";
    for_each(v1.begin(),v1.end(),print);
    cout << endl;
```

```
    cout << "Size of v1 = " << v1.size()
         << endl << endl;

    cout << "Vector v2 : ";
    for_each(v2.begin(),v2.end(),print);
    cout << endl;
    cout << "Size of v2 = " << v2.size()
         << endl << endl;
    v1.swap(v2);
    cout << "After swapping:" << endl;
    cout << "Vector v1 : ";
    for_each(v1.begin(),v1.end(),print);
    cout << endl;
    cout << "Size of v1 = " << v1.size()
         << endl << endl;

    cout << "Vector v2 : ";
    for_each(v2.begin(),v2.end(),print);
    cout << endl;
    cout << "Size of v2 = " << v2.size()
         << endl << endl;
    return 0;
}
```

OUTPUT:

```
// Vector v1 : 1 2 3 4 5 6 7
// Size of v1 = 7
//
// Vector v2 : 8 9 10
// Size of v2 = 3
//
// After swapping:
// Vector v1 : 8 9 10
// Size of v1 = 3
//
// Vector v2 : 1 2 3 4 5 6 7
// Size of v2 = 7
```

Deque

constructors

```
#include <iostream>
#include <deque>
#include <string>
#include <algorithm>
using namespace std;
int main ()
{
    string str[]={"Alex","John","Robert"};

    // empty deque object
    deque<int> d1;

    // creates deque with 10 empty elements
    deque<int> d2(10);

    // creates deque with 10 elements,
    // and assign value 0 for each
    deque<int> d3(10,0);

    // creates deque and assigns
    // values from string array
    deque<string> d4(str+0,str+3);
    deque<string>::iterator sIt = d4.begin();
    while ( sIt != d4.end() )
        cout << *sIt++ << " ";
    cout << endl;
    // copy constructor
    deque<string> d5(d4);
    for ( int i=0; i<3; i++ )
        cout << d5[i] << " ";
    cout << endl;
    return 0;
}
OUTPUT:
//  Alex John Robert
//  Alex John Robert
```

assign

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <iterator>
using namespace std;

int main ()
{
    int ary[]={1,2,3,4,5};
    deque<int> d;
    // assign to the "d" the contains of "ary"
    d.assign(ary,ary+5);
    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    // replace d for 3 copies of 100
    d.assign(3,100);
    copy(d.begin(),d.end(),
            ostream_iterator(cout," "));
    cout << endl;

    return 0;
}
OUTPUT:
// 1 2 3 4 5
// 100 100 100
```

at

```
#include <iostream>
#include <deque>
using namespace std;

int main ()
{
    deque<int> d(3,0);
    d[0] = 100;
    d.at(1) = 200;
    for ( int i=0; i<3; i++ )
```

```
        cout << d.at(i) << " ";
    cout << endl;
    return 0;
}
```

OUTPUT:

```
// 100 200 0
```

back

```
#include <iostream>
#include <deque>
#include <string>
#include <iterator>
using namespace std;
template<class T, class D>
class Member
{
    public:
        Member(T t, D d) : name(t), sal(d) {}
        void print();
    private:
        T name;
        D sal;
};

template<class T, class D>
void Member::print()
{
    cout << name << "  " << sal << endl;
}
//=====================================
int main ()
{
    typedef Member<string,double> M;
    deque<M> d;
    d.push_back(M("Robert",60000));
    d.push_back(M("Linda",75000));
    deque<M>::iterator It = d.begin();
    cout << "Entire deque:" << endl;
    while ( It != d.end() )
        (It++)->print();
    cout << endl;
    cout << "Return from back()" << endl;
```

```
    d.back().print();
    return 0;
}
OUTPUT:
// Entire deque:
// Robert  60000
// Linda  75000
//
// Return from back()
// Linda  75000
```

begin

```
#include <iostream>
#include <deque>
#include <iterator>
#include <numeric>
using namespace std;

int main ()
{
    deque<int> d(5);
    iota(d.begin(),d.end(),1);
    deque<int>::iterator It = d.begin();
    while ( It != d.end() )
        cout << *It++ << " ";
    cout << endl;
    // third element of the deque
    It = d.begin()+2;
    cout << *It << endl;
    return 0;
}
OUTPUT:
// 1 2 3 4 5
// 3
```

clear

```
#include <iostream>
#include <deque>
#include <algorithm>
using namespace std;
```

```
template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//==============================
int main ()
{
    deque<int> d(10);
    Print<int> print;
    fill(d.begin(),d.end(),5);
    cout << "Deque d : ";
    for_each(d.begin(),d.end(),print);
    cout << endl;
    cout << "Size of d = " << d.size()
         << endl;
    cout << "d.clear" << endl;
    d.clear();
    cout << "Deque d : ";
    for_each(d.begin(),d.end(),print);
    cout << endl;
    cout << "Size of d = " << d.size()
         << endl;
    cout << "Deque d is ";
    d.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;

    return 0;
}
// Deque d : 5 5 5 5 5 5 5 5 5 5
// Size of d = 10
// d.clear
// Deque d :
// Size of d = 0
// Deque d is empty
```

empty

```
#include <iostream>
```

```
#include <deque>
using namespace std;

int main ()
{
    deque<int> d;
    cout << "Deque is ";
    d.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;
    d.push_back(100);
    cout << "Deque is ";
    d.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;
    return 0;
}
// Deque is empty
// Deque is not empty
```

end

```
#include <iostream>
#include <deque>
#include <iterator>
#include <numeric>
using namespace std;
int main ()
{
    deque<int> d(5);
    iota(d.begin(),d.end(),1);
    deque<int>::iterator It = d.begin();
    while ( It != d.end() )
        cout << *It++ << " ";
    cout << endl;
    // last element of the deque
    It = d.end()-1;
    cout << *It << endl;
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5
// 5
```

erase

```
#include <iostream>
#include <deque>
#include <iterator>
#include <algorithm>
using namespace std;

int main ()
{
    deque<int> d(10);
    deque<int>::iterator It;
    for ( int i=0; i<10; i++ )
        d[i] = i+1;
    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    It = d.begin()+2;
    // remove third element
    d.erase(It);
    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    It = d.begin();
    // remove 2 elements from beginning fo d
    d.erase(It,It+2);
    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5 6 7 8 9 10
// 1 2 4 5 6 7 8 9 10
// 4 5 6 7 8 9 10
```

front

```
#include <iostream>
#include <deque>
#include <string>
#include <iterator>
using namespace std;
template<class T, class D>
```

```
class Member
{
    public:
        Member(T t, D d) : name(t), sal(d) {}
        void print();
    private:
        T name;
        D sal;
};

template<class T, class D>
void Member::print()
{
    cout << name << "  " << sal << endl;
}
//=====================================
int main ()
{
    typedef Member<string,double> M;
    deque<M> d;
    d.push_back(M("Linda",75000));
    d.push_back(M("Robert",60000));

    deque<M>::iterator It = d.begin();
    cout << "Entire deque:" << endl;
    while ( It != d.end() )
        (It++)->print();
    cout << endl;
    cout << "Return from front()" << endl;
    d.front().print();
    return 0;
}
```

OUTPUT:

```
// Entire deque:
// Linda  75000
// Robert  60000
//
// Return from front()
// Linda  75000
```

insert

```
#include <iostream>
#include <deque>
#include <iterator>
#include <algorithm>
using namespace std;

template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
             cout << t << " ";
        }
};
//==============================
int main ()
{
    int ary[5];
    fill(ary,ary+5,1);

    deque<int> d;
    deque<int>::iterator It;
    Print<int> print;
    copy(ary,ary+5,
            back_inserter(d));
    cout << "deque d                : ";
    for_each(d.begin(),d.end(),print);
    cout << endl;
    It = d.begin();
    // insert value "5" at the position "It"
    cout << "d.insert(It,5)         : ";
    d.insert(It,5);
    for_each(d.begin(),d.end(),print);
    cout << endl;
    // insert range ary+2 - ary+5 at the position "It"
    It = d.begin()+5;
    cout << "d.insert(It,ary+2,ary+5 : ";
    d.insert(It,ary+2,ary+5);
    for_each(d.begin(),d.end(),print);
    cout << endl;
    // insert 2 value of "20" at the position "It"
    It = d.end()-2;
    cout << "d.insert(It,2,20)      : ";
```

```
    d.insert(It,2,20);
    for_each(d.begin(),d.end(),print);
    cout << endl;

    return 0;
}
OUTPUT:
// deque d                  : 1 1 1 1 1
// d.insert(It,5)           : 5 1 1 1 1 1
// d.insert(It,ary+2,ary+5 : 5 1 1 1 1 1 1 1 1
// d.insert(It,2,20)        : 5 1 1 1 1 1 1 20 20 1 1
```

max_size

```
#include <iostream>
#include <deque>
using namespace std;
int main ()
{
    deque<int> d(10);
    cout << "Size of d     = "
         << d.size() << endl;
    cout << "Max_size of d = "
         << d.max_size() << endl;

    return 0;
}
OUTPUT:
// Size of d     = 10
// Max_size of d = 1073741823
```

pop_back

```
#include <iostream>
#include <deque>
#include <algorithm>
using namespace std;

template <class T>
class Print
{
    public:
```

```
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//============================
int main ()
{
    deque<int> d;
    Print<int> print;
    for ( int i=0; i<5; i++ )
        d.push_back(i+1);
    while ( !d.empty() )
    {
        for_each(d.begin(),d.end(),print);
        cout << endl;
        d.pop_back();
    }
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5
// 1 2 3 4
// 1 2 3
// 1 2
// 1
```

pop_front

```
#include <iostream>
#include <deque>
#include <algorithm>
using namespace std;
template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//============================
```

```
int main ()
{
    deque<int> d;
    Print<int> print;
    for ( int i=0; i<5; i++ )
        d.push_back(i+1);
    while ( !d.empty() )
    {
        for_each(d.begin(),d.end(),print);
        cout << endl;
        d.pop_front();
    }
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5
// 2 3 4 5
// 3 4 5
// 4 5
// 5
```

push_back

```
#include <iostream>
#include <deque>
#include <string>
#include <iterator>
using namespace std;

template <class T>
class Name
{
    public:
        Name(T t) : name(t) {}
        void print()
        {
            cout << name << " ";
        }
    private:
        T name;
};
//============================
int main ()
```

```
{
    typedef Name<string> N;
    typedef deque<N> D;
    D d;
    N n1("Robert");
    N n2("Alex");
    d.push_back(n1);
    d.push_back(n2);
    // unnamed object of the type Name
    d.push_back(N("Linda"));
    D::iterator It = d.begin();
    while ( It != d.end() )
        (It++)->print();
    cout << endl;
    return 0;
}
```

OUTPUT:

```
// Robert Alex Linda
```

push_front

```
#include <iostream>
#include <deque>
#include <string>
#include <iterator>
using namespace std;
template <class T>
class Name
{
    public:
        Name(T t) : name(t) {}
        void print()
        {
            cout << name << " ";
        }
    private:
        T name;
};
//============================
int main ()
{
    typedef Name<string> N;
    typedef deque<N> D;
```

```
    D d;
    N n1("Robert");
    N n2("Alex");
    d.push_front(n1);
    d.push_front(n2);

    // unnamed object of the type Name
    d.push_front(N("Linda"));
    D::iterator It = d.begin();
    while ( It != d.end() )
        (It++)->print();
    cout << endl;
    return 0;
}
```

OUTPUT:

```
// Linda Alex Robert
```

rbegin and rend

```
#include <iostream>
#include <iomanip>
#include <deque>
#include <string>
#include <algorithm>
#include <iterator>
using namespace std;
class ID
{
    friend bool operator < ( const ID&, const ID& );
        public:
        ID(string name,int score) : name(name), score(score) {}
        void display ()
        {
            cout.setf(ios::left);
            cout << setw(3) << score << name << endl;
        }
    private:
        string name; int score;
};
//-------------------------------------------------
// comperation function for sorting
bool operator < ( const ID& a, const ID& b )
{
```

```
    return a.score < b.score;
}
//-----------------------------------------------------
typedef deque<ID> Deque; // new name for existing datatype

int main ()
{
    Deque d;
    Deque::iterator Iter;
    d.push_back(ID("Smith A",96));
    d.push_back(ID("Amstrong B.",91));
    d.push_back(ID("Watson D.",82));
    for ( Iter = d.begin(); Iter != d.end(); Iter++ )
        Iter->display();
    sort(d.begin(),d.end()); // sort algorithm
    cout << endl << "Sorted by Score" << endl;
    cout << "===============" << endl;
    for ( Iter = d.begin(); Iter != d.end(); Iter++ )
        Iter->display();

    cout << endl << "Reverse output" << endl;
    cout << "===============" << endl;

    Deque::reverse_iterator r = d.rbegin();
    while ( r != d.rend() )
        cout << r->display();
    cout << endl;
    return 0;
}
```

OUTPUT:

```
// 96 Smith A.
// 91 Amstrong B.
// 82 Watson D.
//
// Sorted by Score
// ===============
// 82 Watson D.
// 91 Amstrong B.
// 96 Smith A.
//
// Reverse output
// ===============
// 96 Smith A.
// 91 Amstrong B.
```

```
// 82 Watson D.
```

resize

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <iterator>
using namespace std;

int main ()
{
    deque<int> d(5);
    for ( int i=0; i<5; i++ )
        d[i] = i*2;

    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    d.resize(7,100);
    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
    d.resize(4);
    copy(d.begin(),d.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    return 0;
}
OUTPUT:
// 0 2 4 6 8
// 0 2 4 6 8 100 100
// 0 2 4 6
```

size

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <iterator>
```

```
using namespace std;

template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//============================
int main ()
{
    deque<char> d(5);
    Print<char> print;
    cout << "Size of d = " << d.size() << endl;
    fill(d.begin(),d.end(),'*');
    for_each(d.begin(),d.end(),print);
    cout << endl;
    for ( int i=0; i < d.size(); i++ )
        cout << d[i] << " ";
    cout << endl;

    for ( int i=0; i<5; i++ )
    {
        cout << "Size of d = ";
        for_each(d.begin(),d.end(),print);
        cout << endl;
        d.pop_back();
    }

    return 0;
}
```

OUTPUT:

```
// Size of d = 5
// * * * * *
// * * * * *
// Size of d = * * * * *
// Size of d = * * * *
// Size of d = * * *
// Size of d = * *
// Size of d = *
```

swap

```
#include <iostream>
#include <deque>
#include <algorithm>
using namespace std;

template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//==============================
int main ()
{
    int ary[] = {1,2,3,4,5,6,7,8,9,10};
    Print print;
    deque<int> d1(ary,ary+7);
    deque<int> d2(ary+7,ary+10);
    cout << "Deque d1 : ";
    for_each(d1.begin(),d1.end(),print);
    cout << endl;
    cout << "Size of d1 = " << d1.size()
         << endl << endl;
    cout << "Deque d2 : ";
    for_each(d2.begin(),d2.end(),print);
    cout << endl;
    cout << "Size of d2 = " << d2.size()
         << endl << endl;
    d1.swap(d2);
    cout << "After swapping:" << endl;
    cout << "Deque d1 : ";
    for_each(d1.begin(),d1.end(),print);
    cout << endl;
    cout << "Size of d1 = " << d1.size()
         << endl << endl;

    cout << "Deque d2 : ";
    for_each(d2.begin(),d2.end(),print);
```

```
    cout << endl;
    cout << "Size of d2 = " << d2.size()
         << endl << endl;
    return 0;
}
```

OUTPUT:

```
// Deque d1 : 1 2 3 4 5 6 7
// Size of d1 = 7
//
// Deque d2 : 8 9 10
// Size of d2 = 3
//
// After swapping:
// Deque d1 : 8 9 10
// Size of d1 = 3
//
// Deque d2 : 1 2 3 4 5 6 7
// Size of d2 = 7
```

list

assign

```
// assign a sequence to the list
#include <iostream>
#include <list>
#include <algorithm>
#include <iterator>
using namespace std;

int main ()
{
```

```
    int ary[]={1,2,3,4,5};
    list<int> l;

    // assign to l the contains of ary
    l.assign(ary,ary+5);

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    // replace l for 3 copies of 100
    l.assign(3,100);

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;


    return 0;
}
OUTPUT:
// 1 2 3 4 5
// 100 100 100
```

back

```
// returns the last element
#include <iostream>
#include <list>
#include <algorithm>
#include <string>
#include <iterator>
using namespace std;

template<class T, class D>
class Member
{
    public:
        Member(T t, D d) : name(t), sal(d) {}
        void print();
    private:
        T name;
```

```
        D sal;
};
template<class T, class D>
void Member<T,D>::print()
{
    cout << name << "  " << sal << endl;
}
//-------------------------------------
int main ()
{
    typedef Member<string,double> M;
    list<M> l;

    l.push_back(M("Robert",60000));
    l.push_back(M("Linda",75000));

    list<M>::iterator It = l.begin();
    cout << "Entire list:" << endl;

    while ( It != l.end() )
        (It++)->print();
    cout << endl;

    cout << "Return from back()" << endl;
    l.back().print();
    return 0;
}
```

OUTPUT:

```
// Entire list:
// Robert  60000
// Linda  75000
//
// Return from back()
// Linda  75000
```

begin

```
// returns an iterator to the beginning
#include <iostream>
#include <list>
#include <algorithm>
#include <iterator>
```

```
#include <numeric>
using namespace std;

int main ()
{
    list<int> l(5);
    iota(l.begin(),l.end(),1);

    list<int>::iterator It = l.begin();
    while ( It != l.end() )
        cout << *It++ << " ";
    cout << endl;

    // third element of the list
    It = l.begin()+2;
    cout << *It << endl;

    return 0;
}
OUTPUT:
// 1 2 3 4 5
// 3
```

clear

```
// removes all elements
#include <iostream>
#include <list>
using namespace std;

int main ()
{
    list<int> l(5,10);
    cout << "Size of list = "
         << l.size() << endl;

    l.clear();
    cout << "After l.clear() size of list = "
         << l.size() << endl;

    return 0;
}
```

```
OUTPUT:
// Size of list = 5
// After l.clear() size of list = 0
```

empty

```
// true if the list is empty
#include <iostream>
#include <list>
using namespace std;

int main ()
{
    list<int> l;

    cout << "List is ";
    l.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;

    l.push_back(100);

    cout << "List is ";
    l.empty() ? cout << "" : cout << "not ";
    cout << "empty" << endl;

    return 0;
}
OUTPUT:
// List is empty
// List is not empty
```

end

```
// returns an iterator to the end
#include <iostream>
#include <list>
#include <numeric>
using namespace std;

int main ()
```

```
{
    list<int> li(10);
    iota(li.begin(),li.end(),1);

    list<int>::iterator It = li.begin();
    while ( It != li.end() )
        cout << *(It++) << " ";
    cout << endl;

    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5 6 7 8 9 10
```

erase

```
// erase an elemen
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
using namespace std;

template <class T>
void print (list<T>& l)
{
    list<int>::iterator It = l.begin();
    while ( It != l.end() )
    {
        cout << *(It++) << " ";
    }
    cout << endl;
}
//====================
int main ()
{
    list<int> li(10);
    iota(li.begin(),li.end(),1);

    print(li);

    list<int>::iterator It;
```

```
    It = find(li.begin(),li.end(),6);

    // erase at the pos It
    li.erase(It);
    print(li);

    It = find(li.begin(),li.end(),4);

    // erase from beginning to the pos It
    li.erase(li.begin(),It);
    print(li);

    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5 6 7 8 9 10
// 1 2 3 4 5 7 8 9 10
// 4 5 7 8 9 10
```

front

```
// returns the first element
#include <iostream>
#include <list>

int main ()
{
    int ary[] = {1,2,3,4,5};
    list li;

    for ( int i=0; i<5; i++ )
    {
        li.push_front(ary[i]);
        cout << "front() : "
             << li.front() << endl;
    }

    return 0;
}
```

OUTPUT:

```
// front() : 1
// front() : 2
```

```
// front() : 3
// front() : 4
// front() : 5
```

insert

```
// insert elements into the list
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
using namespace std;

template <class T>
void print (list<T>& l)
{
    list<int>::iterator It = l.begin();
    while ( It != l.end() )
    {
        cout << *(It++) << " ";
    }
    cout << endl;
}
//====================================
int main ()
{
    list<int> li1(10,0);
    list<int> li2(5);
    list<int>::iterator It;
    iota(li2.begin(),li2.end(),1);

    cout << "li1 : ";
    print(li1);
    cout << "li2 : ";
    print(li2);

    It = li1.begin();
    // value of 20 at the pos It
    li1.insert(++It,20);
    cout << "li1 : ";
    print(li1);
```

```
    // two value of 25 at the beginning
    li1.insert(li1.begin(),2,25);
    cout << "li1 : ";
    print(li1);

    // contents of li2 at the end of li1
    li1.insert(li1.end(),li2.begin(),li2.end());
    cout << "li1 : ";
    print(li1);

    return 0;
}
OUTPUT:
// li1 : 0 0 0 0 0 0 0 0 0 0
// li2 : 1 2 3 4 5
// li1 : 0 20 0 0 0 0 0 0 0 0 0
// li1 : 25 25 0 20 0 0 0 0 0 0 0 0 0
// li1 : 25 25 0 20 0 0 0 0 0 0 0 0 0 1 2 3 4 5
```

max_size

```
// returns the maximum number of elements the list can hold
#include <iostream>
#include <list>

int main ()
{
    list li(10);

    cout << "size() of li = "
     << li.size() << endl;
    cout << "max_size     = "
     << li.max_size() << endl;
    return 0;
}
OUTPUT:
// size() of li = 10
// max_size     = 4294967295
```

merge

```
// merge two lists
#include <iostream>
#include <list>
#include <algorithm>
#include <iterator>
using namespace std;

int main ()
{
    int ary[] = {2,5,9,7,2,7,6,5};
    list<int> list1(ary,ary+4);
    list<int> list2(ary+4,ary+8);

    cout << "list1 : ";
    copy(list1.begin(),list1.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    cout << "list2 : ";
    copy(list2.begin(),list2.end(),
            ostream_iterator<int>(cout," "));
    cout << endl << endl;

    // you have to sort data before megring it
    list1.sort();
    list2.sort();
    list1.merge(list2);

    cout << "After \"list1.merge(list2)\" :" << endl;
    cout << "list1 : ";
    copy(list1.begin(),list1.end(),
            ostream_iterator(cout," "));
    cout << endl;

    cout << "size of list2 = " << list2.size()
         << endl;
    cout << "list2 is " << (list2.empty() ? "" : "not ")
         << "empty" << endl;

    return 0;
}
OUTPUT:
// list1 : 2 5 9 7
```

```
// list2 : 2 7 6 5
//
// After "list1.merge(list2)" :
// list1 : 2 2 5 5 6 7 7 9
// size of list2 = 0
// list2 is empty
```

pop_back

```
// removes the last element
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
using namespace std;

int main ()
{
    list<int> l(5);
    iota(l.begin(),l.end(),1);

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    while ( !l.empty() )
    {
        l.pop_back();

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
        cout << endl;
    }
    return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5
// 1 2 3 4
// 1 2 3
// 1 2
// 1
```

pop_front

```
// removes the first element
#include <iostream>
#include <list>
#include <algorithm>

int main ()
{
    list<int> l(5,0);
    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    cout << "Size of list = "
         << l.size() << endl;

    int size = l.size();

    while ( !l.empty() )
    {
        l.pop_front();
        cout << "Size of list = "
             << l.size() << endl;
    }

    return 0;
}
OUTPUT:
// 0 0 0 0 0
// Size of list = 5
// Size of list = 4
// Size of list = 3
// Size of list = 2
// Size of list = 1
// Size of list = 0
```

push_back

```
// add an element to the end of the list
```

```
#include <iostream>
#include <list>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Name
{
    public:
        Name(T f, T l) : first(f), last(l) {}
        void print()
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
};

//========================================
int main ()
{
    typedef Name<string> N;
    typedef list<N> L;
    L l;
    L::iterator It;

    N n1(string("Albert"),string("Johnson"));
    N n2("Lana","Vinokur");

    l.push_back(n1);
    l.push_back(n2);

    // unnamed object
    l.push_back(N("Linda","Bain"));

    It = l.begin();
    while ( It != l.end() )
        (It++)->print();
    cout << endl;

    return 0;
```

```
}
OUTPUT:
// Albert          Johnson
// Lana            Vinokur
// Linda           Bain
```

push_front

```
// add an element to the front of the list
#include <iostream>
#include <list>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Name
{
    public:
        Name(T f, T l) : first(f), last(l) {}
        void print()
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
};

//========================================
int main ()
{
    typedef Name<string> N;
    typedef list<N> L;
    L l;
    L::iterator It;

    N n1(string("Albert"),string("Johnson"));
    N n2("Lana","Vinokur");

    l.push_front(n1);
```

```
    l.push_front(n2);

    // unnamed object
    l.push_front(N("Linda","Bain"));

    It = l.begin();
    while ( It != l.end() )
        (It++)->print();
    cout << endl;

    return 0;
}
```

OUTPUT:

```
// Linda          Bain
// Lana           Vinokur
// Albert         Johnson
```

rbegin

```
// returns a reverse iterator to the beginning of the list
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
#include <iterator>
using namespace std;

int main ()
{
    list<int> l(10);
    iota(l.begin(),l.end(),1);

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    list<int>::reverse_iterator It = l.rbegin();
    while ( It != l.rend() )
    cout << *(It++) << " ";
    cout << endl;

    return 0;
```

```
}
OUTPUT:
// 1 2 3 4 5 6 7 8 9 10
// 10 9 8 7 6 5 4 3 2 1
```

remove

```
// removes elements from the list
#include <iostream>
#include <list>
#include <algorithm>
#include <string>
using namespace std;

template <class T, class D>
class Salary
{
    public:
        Salary(T t) : id(t) {}
        Salary(T t,D d) : id(t), sal(d) {}
        void print ()
        { cout << id << "  " << sal << endl; }
    private:
        T id;
        D sal;
    friend bool operator ==
        (const Salary& s1,const Salary& s2)
    { return s1.id == s2.id; }
};
//=======================================
int main ()
{
    typedef Salary<string,double> S;
    typedef list<S> L;

    L l;
    l.push_back(S("012345",70000.0));
    l.push_back(S("012346",60000.0));
    l.push_back(S("012347",72000.0));

    L::iterator It = l.begin();
    while ( It != l.end() )
```

```
        (It++)->print();
    cout << endl;

    S s("012345");
    l.remove(s);

    It = l.begin();
    while ( It != l.end() )
        (It++)->print();
    cout << endl;

    return 0;
}
OUTPUT:
// 012345  70000
// 012346  60000
// 012347  72000
//
// 012346  60000
// 012347  72000
```

remove_if

```
// removes elements conditionally
#include <iostream>
#include <list>
#include <algorithm>
OUTPUT:
```

rend

```
// returns a reverse iterator to the start of the list
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
#include <iterator>
using namespace std;

int main ()
```

```
{
    list<int> l(10);
    iota(l.begin(),l.end(),1);

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    list<int>::reverse_iterator It = l.rbegin();
    while ( It != l.rend() )
    cout << *(It++) << " ";
    cout << endl;

    return 0;
}
OUTPUT:
// 1 2 3 4 5 6 7 8 9 10
// 10 9 8 7 6 5 4 3 2 1
```

resize

```
// change the size of the list
#include <iostream>
#include <list>

int main ()
{
    list<int> l(10);

    cout << "Size of list l = "
         << l.size();

    l.resize(100);
    cout << "After l.resize(100)" << endl;
    cout << "Size of list l = "
         << l.size();

    l.resize(5);
    cout << "After l.resize(5)" << endl;
    cout << "Size of list l = "
         << l.size();
```

```
    return 0;
}
OUTPUT:
// Size of list l = 10After l.resize(100)
// Size of list l = 100After l.resize(5)
// Size of list l = 5
```

reverse

```
// reverse the list
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
using namespace std;

int main ()
{
    list<int> l(10);
    iota(l.begin(),l.end(),1);

    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    l.reverse();
    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    return 0;
}
OUTPUT:
// 1 2 3 4 5 6 7 8 9 10
// 10 9 8 7 6 5 4 3 2 1
```

size

```
// the number the elements in the list
#include <iostream>
```

```
#include <list>
#include <algorithm>

int main ()
{
    list<int> l(5,0);
    copy(l.begin(),l.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;

    cout << "Size of list = "
         << l.size() << endl;

    int size = l.size();

    for ( int i=0; i<size; i++ )
    // or while ( !l.empty() ) - safer
    {
        l.pop_front();
        cout << "Size of list = "
             << l.size() << endl;
    }

    return 0;
}
OUTPUT:
// 0 0 0 0 0
// Size of list = 5
// Size of list = 4
// Size of list = 3
// Size of list = 2
// Size of list = 1
// Size of list = 0
```

sort 1.

```
// sorts the list
#include <iostream>
#include <list>
#include <algorithm>
#include <functional>
```

```
using namespace std;

template <class T>
class Print
{
    public:
        void operator () (T& t)

            cout << t << " ";
        }
};
//-----------------------------
int main ()
{
    int ary[] = {3,2,5,7,3,6,7,2,4,5};
    list<int> li(ary,ary+10);
    Print<int> print;

    cout << "Before sorting\nli : ";
    for_each(li.begin(),li.end(),print);
    cout << endl << endl;

    li.sort(greater<int>());

    cout << "After li.sort(greater())\nli : ";
    for_each(li.begin(),li.end(),print);
    cout << endl << endl;

    li.sort(less<int>());

    cout << "After li.sort(less())\nli : ";
    for_each(li.begin(),li.end(),print);
    cout << endl;

    return 0;
}
OUTPUT:
// Before sorting
// li : 3 2 5 7 3 6 7 2 4 5
//
// After li.sort(greater<int>())
// li : 7 7 6 5 5 4 3 3 2 2
//
// After li.sort(less<int>())
```

```
// li : 2 2 3 3 4 5 5 6 7 7
```

sort 2.

```
// sorts with user datatype
#include <iostream>
#include <iomanip>
#include <list>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T f, T l) :
            first_n(f), last_n(l) {}
        void print();
    private:
        string last_n, first_n;
    // for sort() list member function
    friend bool operator < (Member& m1,
            Member& m2)
    { return m1.last_n < m2.last_n; }
};
//-------------------------------------
template <class T>
void Member<T>::print()
{
    cout.setf(ios::left);
    cout << setw(15) << last_n.c_str()
         << first_n << endl;
}

typedef Member<string> M;
//=====================================
int main ()
{
    list<M> li;
    li.push_back(M("Linda","Smith"));
    li.push_back(M("Frost","Robert"));
    li.push_back(M("Alex","Amstrong"));
```

```
    cout << "Before sorting by last name:\n"
     << "============================"
     << endl;

    list<M>::iterator It = li.begin();
    while ( It != li.end() )
    {
        (It++)->print();
    }
    cout << endl;

    li.sort();

    cout << "After sorting by last name:\n"
     << "============================"
     << endl;

    It = li.begin();
    while ( It != li.end() )
    {
        (It++)->print();
    }

        return 0;
}
```

OUTPUT:

```
// Before sorting by last name:
// ============================
// Smith          Linda
// Robert         Frost
// Amstrong       Alex
//
// After sorting by last name:
// ============================
// Amstrong       Alex
// Robert         Frost
// Smith          Linda
```

splice

```
// merge two lists
```

```
#include <iostream>
#include <list>
#include <algorithm>
#include <iterator>
using namespace std;

template <class T>
class Print
{
    public:
        void operator () (T& t)
        {
            cout << t << " ";
        }
};
//====================================
int main ()
{
    ist<int> li1, li2, li3, li4;
    Print print;

    for ( int i=0; i<5; i++ )
    {
        li1.push_back(i);
        li2.push_back(i+5);
        li3.push_back(i+10);
        li4.push_back(i+15);
    }

    cout << "li1 : ";
    for_each(li1.begin(),li1.end(),print);
    cout << endl;

    cout << "li2 : ";
    for_each(li2.begin(),li2.end(),print);
    cout << endl;

    cout << "li3 : ";
    for_each(li3.begin(),li3.end(),print);
    cout << endl;

    cout << "li4 : ";
    for_each(li4.begin(),li4.end(),print);
    cout << endl << endl;
```

```
    li1.splice(li1.end(),li2);

    cout << "li1 : ";
    for_each(li1.begin(),li1.end(),print);
    cout << endl << endl;

    li1.splice(li1.end(),li3,li3.begin(),li3.end());

    cout << "li1 : ";
    for_each(li1.begin(),li1.end(),print);
    cout << endl << endl;

    list<int>::iterator It;
    It = find(li4.begin(),li4.end(),18);

    li1.splice(li1.begin(),li4,It);
    cout << "li1 : ";
    for_each(li1.begin(),li1.end(),print);
    cout << endl;

    cout << "li4 : ";
    for_each(li4.begin(),li4.end(),print);
    cout << endl;

    return 0;
}
```

OUTPUT:

```
// li1 : 0 1 2 3 4
// li2 : 5 6 7 8 9
// li3 : 10 11 12 13 14
// li4 : 15 16 17 18 19
//
// li1 : 0 1 2 3 4 5 6 7 8 9
//
// li1 : 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
//
// li1 : 18 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
// li4 : 15 16 17 19
```

swap

```
// exchange two lists
#include <iostream>
#include <list>
#include <algorithm>
#include <numeric>
using namespace std;

void print (list<int>& l)
{
    list<int>::iterator It = l.begin();
    while ( It != l.end() )
    {
        cout << *(It++) << " ";
    }
    cout << endl;
}
//================================
int main ()
{
    list<int> li1(5), li2(5);
    iota(li1.begin(),li1.end(),1);
    iota(li2.begin(),li2.end(),5);

    cout << "li1 : ";
    print(li1);
    cout << "li2 : ";
    print(li2);

    li1.swap(li2);

    cout << endl <<"After swapping:" << endl;
    cout << "li1 : ";
    print(li1);
    cout << "li2 : ";
    print(li2);

    return 0;
}
OUTPUT:
// li1 : 1 2 3 4 5
// li2 : 5 6 7 8 9
//
// After swapping:
// li1 : 5 6 7 8 9
```

```
// li2 : 1 2 3 4 5
```

unique

```
// removes duplicate elements
#include <iostream>
#include <list>
#include <algorithm>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T f, T l) :
            first_n(f), last_n(l) {}
        void print();
    private:
        string last_n, first_n;
    // for sort function
    friend bool operator < (Member& m1,
            Member& m2)
    { return m1.last_n < m2.last_n; }

    // for merge and unique functions
    friend bool operator == (Member& m1,
            Member& m2)
    { return m1.last_n == m2.last_n; }
};
//-------------------------------------
template <class T>
void Member<T>::print()
{
    cout.setf(ios::left);
    cout << setw(15) << last_n.c_str()
         << first_n << endl;
}

typedef Member<string> M;
//=====================================
```

```
int main ()
{
    list<M> li1;
    li1.push_back(M("Linda","Smith"));
    li1.push_back(M("Robert","Frost"));
    li1.push_back(M("Alex","Amstrong"));

    list li2;
    li2.push_back(M("Linda","Smith"));
    li2.push_back(M("John","Wood"));
    li2.push_back(M("Alex","Amstrong"));

    li1.sort();
    li2.sort();
    li1.merge(li2);

    cout << "li1 after sorting and mergin"
         << endl;

    list<M>::iterator It = li1.begin();
    while ( It != li1.end() )
    {
        (It++)->print();
    }
    cout << endl;

    li1.unique();

    cout << "After li1.unique()" << endl;

    It = li1.begin();
    while ( It != li1.end() )
    {
        (It++)->print();
    }
    cout << endl;

    return 0;
}
```

OUTPUT:

```
// li1 after sorting and mergin
// Amstrong       Alex
// Amstrong       Alex
// Frost          Robert
```

```
// Smith          Linda
// Smith          Linda
// Wood           John
//
// After li1.unique()
// Amstrong       Alex
// Frost          Robert
// Smith          Linda
// Wood           John
```

set

constructors

```
default preicate is less
#include <iostream>
#include <set>

int main ()
{
    int ary[] = { 5,3,7,5,2,3,7,5,5,4 };
    set<int> s1;
    set<int, greater<int> > s2;

    for ( int i=0; i<sizeof(ary)/sizeof(int); i++ )
    {
        s1.insert(ary[i]);
        s2.insert(ary[i]);
    }

    set<int>::iterator It = s1.begin();
    cout << "s1 : ";
    while ( It != s1.end() )
        cout << *(It++) << " ";
    cout << endl;
```

```
    It = s2.begin();
    cout << "s2 : ";
    while ( It != s2.end() )
        cout << *(It++) << " ";
    cout << endl;

    // second form of constructor
    set<int> s3(ary,ary+3);
    It = s3.begin();
    cout << "s3 : ";
    while ( It != s3.end() )
        cout << *(It++) << " ";
    cout << endl;

    // copy constructor (predicate of s1 is important)
    set<int, less > s4(s1);
    It = s4.begin();
    cout << "s4 : ";
    while ( It != s4.end() )
    cout << *(It++) << " ";
    cout << endl;

    return 0;
}
OUTPUT:
// s1 : 2 3 4 5 7
// s2 : 7 5 4 3 2
// s3 : 3 5 7
// s4 : 2 3 4 5 7
```

begin

---

returns an iterator to the first element

```
#include <iostream>
#include <set>
using namespace std;

int main ()
{
     int ary[] = {1,2,3,2,4,5,7,2,6,8};
     set<int> s(ary,ary+10);

     copy(s.begin(),s.end(),
```

```
            ostream_iterator<int>(cout," "));

    return 0;
}
OUTPUT:
// 1 2 3 4 5 6 7 8
```

clear

```
removes all elements
#include <iostream>
#include <set>
using namespace std;

void print (set<int, less<int> >& s)
{
    set<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     s.clear();
     cout << "Size of set s = " << s.size() << endl;
     print(s);

     return 0;
}
OUTPUT:
// 1 2 3 4 5 6 8
// Size of set s = 0
```

count

```
returns the number of elements
#include <iostream>
#include <set>
using namespace std;

void print (set<int, less<int> >& s)
{
    set<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     cout << "count of '2' (0 or 1) is ";
     int n = s.count(2);

     cout << n << endl;

     return 0;
}
OUTPUT:
// 1 2 3 4 5 6 8
// count of '2' (0 or 1) is 1
```

empty

```
true if the set is empty
#include <iostream>
#include <set>
using namespace std;

void print (set<int, less<int> >& s)
{
    set<int, less<int> >::iterator It;
```

```
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     cout << "set is " << ((s.empty()) ? "" : "not ")
          << "empty" << endl;

     s.clear();

     cout << "set is " << ((s.empty()) ? "" : "not ")
          << "empty" << endl;

     return 0;
}
OUTPUT:
// 1 2 3 4 5 6 8
// set is not empty
// set is empty

end
```

```
returns an iterator to the last element
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
```

```
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
    typedef set<M, less<M> > S;
    M m("Frost","Robert");
    S s;

    s.insert(m);
    s.insert(M("Smith","John"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();

    return 0;
}
OUTPUT:
// Bill           Amstrong
// Linda          Bain
// Robert         Frost
// John           Smith
```

equal_ranges

```
returns iterators to the first and last elements that
match a certain key
#include <iostream>
#include <set>
using namespace std;

int main ()
{
    set<int> c;

    c.insert(1);
    c.insert(2);
    c.insert(4);
    c.insert(10);
    c.insert(11);

    cout << "lower_bound(3): "
        << *c.lower_bound(3) << endl;
    cout << "upper_bound(3): "
        << *c.upper_bound(3) << endl;
    cout << "equal_range(3): "
        << *c.equal_range(3).first << " "
        << *c.equal_range(3).second << endl;
    cout << endl;

    cout << "lower_bound(5): "
        << *c.lower_bound(5) << endl;
    cout << "upper_bound(5): "
        << *c.upper_bound(5) << endl;
    cout << "equal_range(5): "
        << *c.equal_range(5).first << " "
        << *c.equal_range(5).second << endl;
cin.get();
}
OUTPUT:
// lower_bound(3): 4
// upper_bound(3): 4
// equal_range(3): 4 4
//
// lower_bound(5): 10
// upper_bound(5): 10
// equal_range(5): 10 10
```

erase

```
removes elements
#include <iostream>
#include <set>
using namespace std;

void print (set<int, less<int> >& s)
{
    set<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//----------------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     // erase '2'
     s.erase(2);
     print(s);

     set<int, less<int> >::iterator It;

     It = s.find(5);

     // erase '5'
     s.erase(It);
     print(s);


     It = s.find(4);
     // erase from It to the end of set
     s.erase(It,s.end());
     print(s);

     return 0;
```

```
}
OUTPUT:
// 1 2 3 4 5 6 8
// 1 3 4 5 6 8
// 1 3 4 6 8
// 1 3
```

find

```
finds a given element
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
```

```
    typedef set<M, less<M> > S;
    M m("Frost","Robert");
    S s;

    s.insert(m);
    s.insert(M("Smith","John"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();

    It = s.find(m);
    if ( It == s.end() )
        cout << "element not found" << endl;
    else
    {
        cout << "element is found : ";
        (*It).print();
    }

    return 0;
}
OUTPUT:
// Bill           Amstrong
// Linda          Bain
// Robert         Frost
// John           Smith
// element is found : Robert         Frost
```

insert

inserts elements into the set

```
#include <iostream>
#include <set>
using namespace std;

void print (set<int, less<int> >& s)
{
    set<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
```

```
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(10);
     print(s);

     s.insert(ary,ary+5);
     print(s);

     set<int, less<int> >::iterator It = s.begin();
     s.insert(It,20);
     print(s);

     return 0;
}
OUTPUT:
// 10
// 1 2 3 10
// 1 2 3 10 20
```

lower_bound

```
returns an iterator to the first element greater
than a certain value
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l) : last(l), first("") {} // for upper_bound
                                            // and lower_bound
```

```
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
    typedef set<M, less<M> > S;
    S s;

    s.insert(M("Smith","John"));
    s.insert(M("Shevchenko","Taras"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));
    s.insert(M("Pushkin","Alexander"));
    s.insert(M("Pasternak","Biris"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();
    cout << endl;

    M m1("P");
    M m2("Pzz");

    S::iterator low = s.lower_bound(m1);
    S::iterator upp = s.upper_bound(m2);
```

```
    It = low;

    while ( It != upp )
        (It++)->print();

    return 0;
}
OUTPUT:
// Bill            Amstrong
// Linda           Bain
// Biris           Pasternak
// Alexander       Pushkin
// Taras           Shevchenko
// John            Smith
//
// Biris           Pasternak
// Alexander       Pushkin
```

key_comp

```
returns the function that compares keys
#include <iostream>
#include <set>
using namespace std ;

template
void truefalse(T t)
{
    cout << (t?"True":"False") << endl;
}

int main ()
{
    set<int> s;

    cout << "s.key_comp()(1,2) returned ";
    truefalse(s.key_comp()(1,2));  // True

    cout << "s.key_comp()(2,1) returned ";
    truefalse(s.key_comp()(2,1));  // False

    cout << "s.key_comp()(1,1) returned ";
```

```
    truefalse(s.key_comp()(1,1));  // False

    return 0;
}
```

OUTPUT:

```
// s.key_comp()(1,2) returned True
// s.key_comp()(2,1) returned False
// s.key_comp()(1,1) returned False
```

max_size

```
the maximum number of elements that the set can hold
#include <iostream>
#include <set>
#include <algorithm>
using namespace std;

void print (set<int, less<int> >& s)
{
    copy(s.begin(),s.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     cout << "size of set 's' = "
          << s.size() << endl;
     cout << "max_size of 's' = "
          << s.max_size() << endl;

     return 0;
}
```

OUTPUT:

```
// 1 2 3 4 5 6 8
// size of set 's' = 7
// max_size of 's' = 4294967295
```

rbegin

```
returns a reverse iterator to the end of the set
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
    typedef set<M, less<M> > S;
    M m("Frost","Robert");
```

```
    S s;

    s.insert(m);
    s.insert(M("Smith","John"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();

    cout << endl;

    S::reverse_iterator rI = s.rbegin();
    while ( rI != s.rend() )
        (rI++)->print();

    return 0;
}
OUTPUT:
// Bill           Amstrong
// Linda          Bain
// Robert         Frost
// John           Smith
//
// John           Smith
// Robert         Frost
// Linda          Bain
// Bill           Amstrong
```

rend

---

returns a reverse iterator to the beginning of the set

```
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
```

```
    public:
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
    typedef set<M, less<M> > S;
    M m("Frost","Robert");
    S s;

    s.insert(m);
    s.insert(M("Smith","John"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();

    cout << endl;

    S::reverse_iterator rI = s.rbegin();
    while ( rI != s.rend() )
        (rI++)->print();

    return 0;
```

```
}
OUTPUT:
// Bill            Amstrong
// Linda           Bain
// Robert          Frost
// John            Smith
//
// John            Smith
// Robert          Frost
// Linda           Bain
// Bill            Amstrong
```

size

```
the number of elements in the set
#include <iostream>
#include <set>
using namespace std;

void print (set<int, less<int> >& s)
{
    set<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     set<int, less<int> > s;

     s.insert(ary,ary+10);
     cout << "Size of set s = " << s.size() << endl;
     print(s);

     s.clear();
     cout << "Size of set s = " << s.size() << endl;

     return 0;
}
OUTPUT:
// Size of set s = 7
```

```
// 1 2 3 4 5 6 8
// Size of set s = 0
```

swap

```
exchanges two sets
#include <iostream>
#include <set>
#include <algorithm>
using namespace std;

void print (set<int, less<int> >& s)
{
    copy(s.begin(),s.end(),
            ostream_iterator<int>(cout," "));
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary1[] = {1,2,3,2,3,4,8,2,5,6};
     int ary2[] = {5,0,9,2,3,4,8,2,5,6};
     set<int, less<int> > s1, s2;

     s1.insert(ary1,ary1+10);
     cout << "s1 : ";
     print(s1);

     cout << "s2 : ";
     s2.insert(ary2,ary2+10);
     print(s2);

     if ( s1 != s2 )
         s1.swap(s2);

     cout << "s1 : ";
     print(s1);

     cout << "s2 : ";
     print(s2);

     return 0;
```

```
}
OUTPUT:
// s1 : 1 2 3 4 5 6 8
// s2 : 0 2 3 4 5 6 8 9
// s1 : 0 2 3 4 5 6 8 9
// s2 : 1 2 3 4 5 6 8
```

upper_bound

```
returns an iterator to the first element greater than
a certain value
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l) : last(l), first("") {} // for upper_bound
                                            // and lower_bound
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
```

```
//=============================================
int main ()
{
    typedef Member<string> M;
    typedef set<M, less<M> > S;
    S s;

    s.insert(M("Smith","John"));
    s.insert(M("Shevchenko","Taras"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));
    s.insert(M("Pushkin","Alexander"));
    s.insert(M("Pasternak","Boris"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();
    cout << endl;

    M m1("P");
    M m2("Pzz");

    S::iterator low = s.lower_bound(m1);
    S::iterator upp = s.upper_bound(m2);

    It = low;

    while ( It != upp )
        (It++)->print();

    return 0;
}
```

OUTPUT:

```
// Bill          Amstrong
// Linda         Bain
// Biris         Pasternak
// Alexander     Pushkin
// Taras         Shevchenko
// John          Smith
//
// Boris         Pasternak
// Alexander     Pushkin
```

value_comp

```
returns the function that compares values
#include <iostream>
#include <set>
using namespace std ;

template
void truefalse(T t)
{
    cout << (t?"True":"False") << endl;
}

int main ()
{
    set<int> s;

    cout << "s.value_comp()(1,2) returned ";
    truefalse(s.value_comp()(1,2));  // True

    cout << "s.value_comp()(2,1) returned ";
    truefalse(s.value_comp()(2,1));  // False

    cout << "s.value_comp()(1,1) returned ";
    truefalse(s.value_comp()(1,1));  // False

    return 0;
}
OUTPUT:
// s.value_comp()(1,2) returned True
// s.value_comp()(2,1) returned False
// s.value_comp()(1,1) returned False
```

multiset

constructors

```
#include <iostream>
#include <set>
using namespace std;

int main ()
{
     int ary[] = {1,2,3,2,5,4,2,1,4,5};

     multiset<int, less<int> > ms1;

     multiset<int, greater<int> > ms2(ary,ary+10);
     multiset<int>::iterator It;

     cout << "ms2 : ";
     for ( It = ms2.begin(); It != ms2.end(); It++ )
         cout << *It << " ";
     cout << endl;

     // copy constructor
     multiset<int, greater<int> > ms3(ms2);

     cout << "ms3 : ";
     for ( It = ms3.begin(); It != ms3.end(); It++ )
         cout << *It << " ";
     cout << endl;

     It = ms2.begin();

     while ( It != ms2.end() )
         ms1.insert(*It++);

     cout << "ms1 : ";
     for ( It = ms1.begin(); It != ms1.end(); It++ )
         cout << *It << " ";
     cout << endl;

     return 0;
}
```

OUTPUT:

```
// ms2 : 5 5 4 4 3 2 2 2 1 1
// ms3 : 5 5 4 4 3 2 2 2 1 1
// ms1 : 1 1 2 2 2 3 4 4 5 5
```

begin

```
returns an iterator to the first element
#include <iostream>
#include <set>
#include <algorithm>
using namespace std;

int main ()
{
     int ary[] = {1,2,3,2,4,5,7,2,6,8};
     multiset<int> s(ary,ary+10);

     copy(s.begin(),s.end(),
             ostream_iterator<int>(cout," "));

     return 0;
}
OUTPUT:
// 1 2 2 2 3 4 5 6 7 8
```

clear

```
removes all elements
#include <iostream>
#include <set>
using namespace std;

void print (multiset<int, less<int> >& s)
{
    multiset<int>::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
```

```
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     multiset<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     s.clear();
     cout << "Size of multiset s = " << s.size() << endl;
     print(s);

     return 0;
}
```

OUTPUT:

```
// 1 2 2 2 3 3 4 5 6 8
// Size of multiset s = 0
```

count

```
returns the number of elements
#include <iostream>
#include <set>
using namespace std;

void print (multiset<int, less<int> >& s)
{
    multiset<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     multiset<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     cout << "count of '2' is ";
```

```
    int n = s.count(2);

    cout << n << endl;

    return 0;
}
```

OUTPUT:

```
// 1 2 2 2 3 3 4 5 6 8
// count of '2' is 3
```

empty

```
true if the multiset is empty
#include <iostream>
#include <set>
using namespace std;

void print (multiset<int, less<int> >& s)
{
    multiset<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
    int ary[] = {1,2,3,2,3,4,8,2,5,6};
    multiset<int, less<int> > s;

    s.insert(ary,ary+10);
    print(s);

    cout << "multiset is " << ((s.empty()) ? "" : "not ")
         << "empty" << endl;

    s.clear();

    cout << "multiset is " << ((s.empty()) ? "" : "not ")
         << "empty" << endl;

    return 0;
```

```
}
OUTPUT:
// 1 2 2 2 3 3 4 5 6 8
// multiset is not empty
// multiset is empty


end
```

```
returns an iterator to the last element
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
```

```
    typedef multiset<M, less<M> > S;
    M m("Frost","Robert");
    S s;

    s.insert(m);
    s.insert(M("Smith","John"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();

    return 0;
}
OUTPUT:
// Bill          Amstrong
// Linda         Bain
// Robert        Frost
// John          Smith
```

equal_ranges

returns iterators to the first and last elements that match a certain key

```
#include <iostream>
#include <set>
```

OUTPUT:

erase

removes elements

```
#include <iostream>
#include <set>
using namespace std;

void print (multiset<int, less<int> >& s)
{
    multiset<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
```

```
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     multiset<int, less<int> > s;

     s.insert(ary,ary+10);
     print(s);

     // erase all '2'
     s.erase(2);
     print(s);

     multiset<int, less<int> >::iterator It;

     It = s.find(5);

     // erase '5'
     s.erase(It);
     print(s);


     It = s.find(4);
     // erase from It to the end of multiset
     s.erase(It,s.end());
     print(s);

     return 0;
}
OUTPUT:
// 1 2 2 2 3 3 4 5 6 8
// 1 3 3 4 5 6 8
// 1 3 3 4 6 8
// 1 3 3
```

find

---

finds a given element

```
#include <iostream>
```

```
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_str()
                 << last << endl;
        }
    private:
        T first, last;
    // const !!!
    friend bool operator < (const Member& m1, const Member& m2)
    {
        return (m1.last < m2.last) ? true : false;
    }
    friend bool operator == (const Member& m1, const Member& m2)
    {
        return (m1.last == m2.last) ? true : false;
    }
};
//=============================================
int main ()
{
    typedef Member<string> M;
    typedef multiset<M, less<M> > S;
    M m("Frost","Robert");
    S s;

    s.insert(m);
    s.insert(M("Smith","John"));
    s.insert(M("Amstrong","Bill"));
    s.insert(M("Bain","Linda"));

    S::iterator It = s.begin();
    while ( It != s.end() )
        (It++)->print();
```

```
    It = s.find(m);
    if ( It == s.end() )
        cout << "element not found" << endl;
    else
    {
        cout << "element is found : ";
        (*It).print();
    }

    return 0;
}
```

OUTPUT:

```
// Bill            Amstrong
// Linda           Bain
// Robert          Frost
// John            Smith
// element is found : Robert         Frost
```

insert

```
inserts elements into the multiset
#include <iostream>
#include <set>
using namespace std;

void print (multiset<int, less<int> >& s)
{
    multiset<int, less<int> >::iterator It;
    for ( It = s.begin(); It != s.end(); It++ )
        cout << *It << " ";
    cout << endl;
}
//-----------------------------------------
int main ()
{
     int ary[] = {1,2,3,2,3,4,8,2,5,6};
     multiset > s;

     s.insert(10);
     print(s);
```

```
    s.insert(ary,ary+5);
    print(s);

    multiset<int, less<int> >::iterator It = s.begin();
    s.insert(It,20);
    print(s);

    return 0;
}
OUTPUT:
// 10
// 1 2 2 3 3 10
// 1 2 2 3 3 10 20
```

lower_bound

```
returns an iterator to the first element greater
than a certain value
#include <iostream>
#include <set>
#include <iomanip>
#include <string>
using namespace std;

template <class T>
class Member
{
    public:
        Member(T l) : last(l), first("") {} // for upper_bound
                                            // and lower_bound
        Member(T l, T f) : last(l), first(f) {}
        void print() const // const !!!
        {
            cout.setf(ios::left);
            cout << setw(15) << first.c_
```

map

constructors

```
#include <iostream>
#include <map>
using namespace std;

int main ()
{
     typedef map<int, char, less<int> > M;
     M m1;

     m1.insert(M::value_type(2,'B'));
     m1.insert(M::value_type(3,'C'));
     m1.insert(M::value_type(1,'A'));

     M::iterator It = m1.begin();
     cout << endl << "m1:" << endl;
     while ( It != m1.end() )
     {
         cout << (*It).first << " - "
              << (*It).second
              << endl;
         It++;
     }

     // copy constructor
     M m2(m1);

     It = m2.begin();
     cout << endl << "m2:" << endl;
     while ( It != m2.end() )
     {
         cout << (*It).first << " - "
              << (*It).second
              << endl;
         It++;
     }

     M m3(m2.begin(),m2.end());
```

```
    It = m3.begin();
    cout << endl << "m3:" << endl;
    while ( It != m3.end() )
    {
        cout << (*It).first << " - "
             << (*It).second
             << endl;
        It++;
    }

    return 0;
}
OUTPUT:
// m1:
// 1 - A
// 2 - B
// 3 - C
//
// m2:
// 1 - A
// 2 - B
// 3 - C
//
// m3:
// 1 - A
// 2 - B
// 3 - C
```

begin

```
returns an iterator to the first element
#include <iostream>
#include <map>
using namespace std;

int main ()
{
    typedef map<int, char, greater<int> > M;
    typedef M::value_type v_t;
    M m;
```

```
    m.insert(v_t(2,'B'));
    m.insert(v_t(3,'C'));
    m.insert(v_t(1,'A'));

    M::iterator It = m.begin();
    cout << "m:" << endl;
    while ( It != m.end() )
    {
        cout << (*It).first << " - "
             << (*It).second
             << endl;
        It++;
    }

    return 0;
}
```

OUTPUT:

```
// m:
// 3 - C
// 2 - B
// 1 - A
```

clear

---

removes all elements

```
#include <iostream>
#include <map>
```

OUTPUT:

count

---

returns the number of elements

```
#include <iostream>
#include <map>
#include <list>
#include <numeric>
using namespace std;

int main ()
{
```

```
    list<int> L1(3), L2(3);
    iota(L1.begin(),L1.end(),1);
    iota(L2.begin(),L2.end(),4);

    typedef map<int, list<int> > M;
    M m;
    m.insert(M::value_type(1,L1));
    m.insert(M::value_type(2,L2));

    M::iterator It;
    list<int>::iterator Li;
    for ( It = m.begin(); It != m.end(); It++ )
    {
        cout << "map " << (*It).first << ": ";
        for ( Li = It->second.begin();
                Li != It->second.end(); Li++ )
            cout << *Li << " ";
        cout << endl;
    }

    int n = m.count(2);
    cout << "count of element with key '2' (0 or 1) is "
         << n << endl;

    return 0;
}
```

OUTPUT:

```
// map 1: 1 2 3
// map 2: 4 5 6
// count of element with key '2' (0 or 1) is 1
```

empty

```
true if the map is empty
#include <iostream>
#include <map>
using namespace std;

int main ()
{
    typedef map<int,int> M;
    M m;
```

```
    m[1] = 100;
    m[3] = 200;
    m[5] = 300;

    cout << "values of map 'm': ";
    M::iterator It = m.begin();
    while ( It != m.end() )
    {
        cout << (*It).second << " ";
        It++;
    }
    cout << endl;
    cout << "size of map = " << m.size()
         << endl;
    cout << "map 'm' is " << (m.empty() ?
            "" : "not ") << "empty" << endl << endl;

    m.erase(m.begin(),m.end());
    cout << "After m.erase(m.begin(),m.end())"
         << endl;

    cout << "size of map = " << m.size()
         << endl;
    cout << "map 'm' is " << (m.empty() ?
            "" : "not ") << "empty" << endl;

    return 0;
}
OUTPUT:
// values of map 'm': 100 200 300
// size of map = 3
// map 'm' is not empty
//
// After m.erase(m.begin(),m.end())
// size of map = 0
// map 'm' is empty
```

end

---

returns an iterator to the last element

```
#include <iostream>
```

```
#include <map>
```

OUTPUT:

equal_ranges

```
returns iterators to the first and last elements that
match a certain key
#include <iostream>
#include <map>
```

OUTPUT:

erase

```
removes elements
#include <iostream>
#include <map>
#include <string>
using namespace std;

typedef map<string, int, less<string> > M;

void print (M& m)
{
    M::iterator It = m.begin();
    cout << "map :" << endl;
    while ( It != m.end() )
    {
        cout << (*It).first << " - ";
        cout << (*It).second << endl;
        It++;
    }
}
//-----------------------------------
int main ()
{
     typedef M::value_type v_t;

     M m;
     m.insert(v_t("AAA",1));
     m.insert(v_t("BBB",2));
```

```
	m.insert(v_t("CCC",3));

	m["DDD"] = 4;
	m["EEE"] = 5;

	print(m);

	// remove element with key 'BBB'
	m.erase("BBB");
	print(m);

	M::iterator It;
	It = m.find("DDD");
	// remove element pointed by It
	m.erase(It);
	print(m);

	It = m.find("CCC");
	// remove the range of elements
	m.erase(m.begin(),++It);
	print(m);

	return 0;
}
```

OUTPUT:

```
// map :
// AAA - 1
// BBB - 2
// CCC - 3
// DDD - 4
// EEE - 5
// map :
// AAA - 1
// CCC - 3
// DDD - 4
// EEE - 5
// map :
// AAA - 1
// CCC - 3
// EEE - 5
// map :
// EEE - 5
```

find

```
finds a given element
#include <iostream>
#include <map>
using namespace std;

int main ()
{
     typedef map<int,char> M;
     char ch = 'A';
     M m;

     for ( int i=0; i<5; i++ )
         m[i] = ch++;

     M::iterator It = m.begin();
     cout << "map m:" << endl;

     while ( It != m.end() )
     {
         cout << (*It).first << " - "
              << (*It).second << endl;
         It++;
     }

     It = m.find(4);
     if ( It != m.end() )
         cout << "element key '4' has value "
              << (*It).second << endl;
     else
         cout << "element key '4' not found"
              << endl;
     return 0;
}
OUTPUT:
// map m:
// 0 - A
// 1 - B
// 2 - C
// 3 - D
// 4 - E
// element key '4' has value E
```

insert

```
inserts elements into the map
#include <iostream>
#include <map>
#include <string>
using namespace std;

int main ()
{
     typedef map<int, char, less<char> > M;
     typedef M::value_type v_t;

     M m1, m2;

     char ch = 'A';
     for ( int i=0; i<3; i++ )
     {
         m1[i+1] = ch+i;
         m2[i+4] = ch+i+3;
     }

     cout << "m1 :" << endl;
     M::iterator It = m1.begin();
     while ( It != m1.end() )
     {
         cout << (*It).first << " - "
              << (*It).second << endl;
         It++;
     }

     cout << "m2 :" << endl;
     It = m2.begin();
     while ( It != m2.end() )
     {
         cout << (*It).first << " - "
              << (*It).second << endl;
         It++;
     }

     // insert new element
```

```
    m1.insert(v_t(5,'E'));

    It = m2.find(6);
    // insert element pointed by It
    m1.insert(*It);

    cout << "m1 :" << endl;
    It = m1.begin();
    while ( It != m1.end() )
    {
        cout << (*It).first << " - "
             << (*It).second << endl;
        It++;
    }

    // insert the range of elements
    m1.insert(m2.begin(),m2.end());

    cout << "m1 :" << endl;
    It = m1.begin();
    while ( It != m1.end() )
    {
        cout << (*It).first << " - "
             << (*It).second << endl;
        It++;
    }

    return 0;
}
OUTPUT:
// m1 :
// 1 - A
// 2 - B
// 3 - C
// m2 :
// 4 - D
// 5 - E
// 6 - F
// m1 :
// 1 - A
// 2 - B
// 3 - C
// 5 - E
// 6 - F
```

```
// m1 :
// 1 - A
// 2 - B
// 3 - C
// 4 - D
// 5 - E
// 6 - F
```

lower_bound

```
returns an iterator to the first element greater
than a certain value
#include <iostream>
#include <map>
#include <ctime>
using namespace std;

unsigned long int gener_rand()
{
    unsigned long int random =
        (unsigned long int)
        (10000.0 * rand() /
        (RAND_MAX + 1.0 )) % 10;
    return random;
}
//---------------------------------
int main ()
{
    unsigned long int ary[100];
    typedef map<int, unsigned long int> M;
    M m;

    // initialize all values to 0
    for ( int i=0; i<10; i++ )
        m[i] = 0;

    srand(time(0));

    // initialize ary[] with random values
    for ( int i=0; i<100; i++ )
        ary[i] = gener_rand();
```

```
    for ( int i=0; i<100; i++ )
    {
        if ( i % 10 == 0 && i != 0 )
            cout << endl;
        cout << ary[i] << " ";
        // generate freaquances
        m[ary[i]] += 1;
    }
    cout << endl << endl;

    M::iterator It = m.begin();
    while ( It != m.end() )
    {
        cout << "number " << (*It).first
             << " occurred " << (*It).second
             << " time(s)" << endl;
        It++;
    }
    cout << endl;

    m[12] = 123;
    m[15] = 234;
    m[18] = 345;

    It = m.lower_bound(11);
    cout << "lower_bound(11) = "
         << (*It).first << endl;

    It = m.upper_bound(11);
    cout << "upper_bound(11) = "
         << (*It).first << endl;

    return 0;
}
```

OUTPUT:

```
// 9 7 9 6 9 6 2 0 8 9
// 6 6 6 8 0 9 5 6 5 7
// 2 1 0 3 2 3 4 4 2 2
// 2 1 9 1 8 9 8 0 0 6
// 9 6 3 6 5 3 5 0 0 0
// 8 2 2 8 6 4 2 0 9 4
// 3 1 6 2 3 4 2 1 5 2
// 8 5 8 1 1 3 5 7 4 6
// 7 2 9 0 1 5 4 4 6 4
```

```
// 9 9 5 5 2 8 0 4 0 6
//
// number 0 occurred 12 time(s)
// number 1 occurred 8 time(s)
// number 2 occurred 14 time(s)
// number 3 occurred 7 time(s)
// number 4 occurred 10 time(s)
// number 5 occurred 10 time(s)
// number 6 occurred 14 time(s)
// number 7 occurred 4 time(s)
// number 8 occurred 9 time(s)
// number 9 occurred 12 time(s)
//
// lower_bound(11) = 12
// upper_bound(11) = 12
```

key_comp

---

returns the function that compares keys

```
#include <iostream>
#include <map>
```

OUTPUT:

max_size

---

the maximum number of elements that the map can hold

```
#include <iostream>
#include <map>
```

OUTPUT:

rbegin

---

returns a reverse iterator to the end of the map

```
#include <iostream>
#include <map>
#include <iomanip>
#include <string>
using namespace std;
```

```
template<class T>
class ID
{
    public:
        ID(T t, T n) : id(t), name(n) {}
        void print ()
        {
            cout.setf(ios::left);
            cout << setw(15) << name.c_str()
                 << id << endl;
            cout.unsetf(ios::left);
        }
    private:
        T id, name;
};
//=====================================
int main ()
{
    typedef ID<string> Id;
    typedef map<int, Id> M;
    typedef M::value_type v_t;

    M m;
    m.insert(v_t(1,Id("000123","Shevchenko")));
    m.insert(v_t(2,Id("000124","Pushkin")));
    m.insert(v_t(3,Id("000125","Shakespeare")));
    // same key
    m.insert(v_t(3,Id("000126","Smith")));

    M::reverse_iterator It = m.rbegin();
    while ( It != m.rend() )
    {
        cout.setf(ios::left);
        cout << setw(3) << (*It).first;
        It->second.print();
        It++;
    }

    return 0;
}
```

OUTPUT:

```
// 3  Shakespeare    000125
// 2  Pushkin        000124
```

```
// 1  Shevchenko      000123


rend
```

```
returns a reverse iterator to the beginning of the map
#include <iostream>
#include <map>
#include <iomanip>
#include <string>
using namespace std;

template<class T>
class ID
{
    public:
        ID(T t, T n) : id(t), name(n) {}
        void print ()
        {
            cout.setf(ios::left);
            cout << setw(15) << name.c_str()
                 << id << endl;
            cout.unsetf(ios::left);
        }
    private:
        T id, name;
};
//=====================================
int main ()
{
    typedef ID<string> Id;
    typedef map<int, Id> M;
    typedef M::value_type v_t;

    M m;
    m.insert(v_t(1,Id("000123","Shevchenko")));
    m.insert(v_t(2,Id("000124","Pushkin")));
    m.insert(v_t(3,Id("000125","Shakespeare")));
    // same key
    m.insert(v_t(3,Id("000126","Smith")));

    M::reverse_iterator It = m.rbegin();
    while ( It != m.rend() )
```

```
    {
        cout.setf(ios::left);
        cout << setw(3) << (*It).first;
        It->second.print();
        It++;
    }

    return 0;
}
OUTPUT:
// 3  Shakespeare    000125
// 2  Pushkin        000124
// 1  Shevchenko
```

multimap

constructors

```
#include <iostream>
#include <map>
using namespace std;

int main ()
{
     typedef multimap<int, char, less<int> > M;
     M m1;

     m1.insert(M::value_type(2,'B'));
     m1.insert(M::value_type(3,'C'));
     m1.insert(M::value_type(1,'A'));
     m1.insert(M::value_type(1,'a'));

     M::iterator It = m1.begin();
     cout << endl << "m1:" << endl;
     while ( It != m1.end() )
     {
```

```
        cout << (*It).first << " - "
             << (*It).second
             << endl;
        It++;
    }

    // copy constructor
    M m2(m1);

    It = m2.begin();
    cout << endl << "m2:" << endl;
    while ( It != m2.end() )
    {
        cout << (*It).first << " - "
             << (*It).second
             << endl;
        It++;
    }

    M m3(m2.begin(),m2.end());

    It = m3.begin();
    cout << endl << "m3:" << endl;
    while ( It != m3.end() )
    {
        cout << (*It).first << " - "
             << (*It).second
             << endl;
        It++;
    }

    return 0;
}
OUTPUT:
// m1:
// 1 - A
// 1 - a
// 2 - B
// 3 - C
//
// m2:
// 1 - A
// 1 - a
// 2 - B
```

```
// 3 - C
//
// m3:
// 1 - A
// 1 - a
// 2 - B
// 3 - C
```

begin

```
returns an iterator to the first element
#include <iostream>
#include <map>
#include <string>
using namespace std;

int main ()
{
     typedef multimap<string,int> M;
     typedef M::value_type v_t;
     M m;

     m.insert(v_t("first",100));
     m.insert(v_t("second",200));
     m.insert(v_t("third",300));
     m.insert(v_t("second",400));
     m.insert(v_t("third",500));

     M::iterator It = m.begin();
     cout << "m:" << endl;
     while ( It != m.end() )
     {
         cout << (*It).first << " - "
              << (*It).second
              << endl;
         It++;
     }

     return 0;
}
OUTPUT:
// m:
```

```
// first - 100
// second - 200
// second - 400
// third - 300
// third - 500
```

clear

```
removes all elements
#include <iostream>
#include <map>
```

OUTPUT:

count

```
returns the number of elements
#include <iostream>
#include <map>
#include <string>
#include <fstream>
using namespace std;

int main ()
{
     typedef multimap<char,string> M1;
     typedef M1::value_type v_t1;
     M1 m1;
     typedef multimap<string,char,less<string> > M2;
     typedef M2::value_type v_t2;
     M2 m2;

     string word;
     int counter = 0;

     ifstream In("/usr/share/dict/words");
     if ( In.good() )
     {
         while(1)
         {
             getline(In,word);
```

```
            char ch = word.at(0);
            // file is sorted
            if ( ch != 'A' && ch != 'a' )
                break;
            else
            {
                // for conting of words
                m1.insert(v_t1(ch,word));
                // for upper-lower bound
                m2.insert(v_t2(word,ch));
            }
            counter++;
        }
        In.close();
    }

    cout << "System Dictionary consists " << counter
         << " words,\nwith first letter 'a' or 'A'"
         << endl;
    cout << m1.count('A') << " words start with 'A'"
         << endl;
    cout << m1.count('a') << " words start with 'a'"
         << endl;

    M2::iterator low = m2.lower_bound("Aba");
    M2::iterator upp = m2.upper_bound("Abe");
    cout << "Range of the words from 'Aba' to 'Abe':"
         << endl;
    while ( low != upp )
    {
        cout << (*low).first << endl;
        low++;
    }
    return 0;
}
```

OUTPUT:

```
// System Dictionary consists 3577 words,
// with first letter 'a' or 'A'
// 491 words start with 'A'
// 3086 words start with 'a'
// Range of the words from 'Aba' to 'Abe':
// Ababa
// Abba
// Abbott
```

```
// Abby
// Abe
```

empty

```
true if the maltimap is empty
#include <iostream>
#include <map>
using namespace std;

int main ()
{
    typedef multimap<int,int> M;
    typedef M::value_type v_t;
    M m;

    m.insert(v_t(1,100));
    m.insert(v_t(1,200));
    m.insert(v_t(2,300));
    m.insert(v_t(3,400));

    cout << "values of multimap 'm': ";
    M::iterator It = m.begin();
    while ( It != m.end() )
    {
        cout << (*It).second << " ";
        It++;
    }
    cout << endl;
    cout << "size of multimap = " << m.size()
         << endl;
    cout << "multimap 'm' is " << (m.empty() ?
            "" : "not ") << "empty" << endl << endl;

    m.erase(m.begin(),m.end());
    cout << "After m.erase(m.begin(),m.end())"
         << endl;

    cout << "size of multimap = " << m.size()
         << endl;
    cout << "multimap 'm' is " << (m.empty() ?
            "" : "not ") << "empty" << endl;
```

```
    return 0;
}
OUTPUT:
// values of multimap 'm': 100 200 300 400
// size of multimap = 4
// multimap 'm' is not empty
//
// After m.erase(m.begin(),m.end())
// size of multimap = 0
// multimap 'm' is empty
```

end

```
returns an iterator to the last element
#include <iostream>
#include <map>
#include <string>
using namespace std;

int main ()
{
     typedef multimap<string,int> M;
     typedef M::value_type v_t;
     M m;

     m.insert(v_t("first",100));
     m.insert(v_t("second",200));
     m.insert(v_t("third",300));
     m.insert(v_t("second",400));
     m.insert(v_t("third",500));

     M::iterator It = m.begin();
     cout << "m:" << endl;
     while ( It != m.end() )
     {
         cout << (*It).first << " - "
              << (*It).second
              << endl;
         It++;
     }
```

```
    return 0;
}
OUTPUT:
// m:
// first - 100
// second - 200
// second - 400
// third - 300
// third - 500
```

equal_ranges

```
returns iterators to the first and last elements that
match a certain key
#include <iostream>
#include <map>

OUTPUT:
```

erase

```
removes elements
#include <iostream>
#include <map>
#include <string>
using namespace std;

typedef multimap<string, int, less<string> > M;

void print (M& m)
{
    M::iterator It = m.begin();
    cout << "multimap :" << endl;
    while ( It != m.end() )
    {
        cout << (*It).first << " - ";
        cout << (*It).second << endl;
        It++;
    }
}
//-----------------------------------
```

```
int main ()
{
      typedef M::value_type v_t;

      M m;
      m.insert(v_t("AAA",1));
      m.insert(v_t("BBB",2));
      m.insert(v_t("CCC",3));
      m.insert(v_t("EEE",4));
      m.insert(v_t("CCC",5));
      m.insert(v_t("DDD",6));

      print(m);

      // remove element with key 'BBB'
      m.erase("BBB");
      print(m);

      M::iterator It;
      It = m.find("DDD");
      // remove element pointed by It
      m.erase(It);
      print(m);

      It = m.find("CCC");
      // remove the range of elements
      m.erase(m.begin(),It);
      print(m);

      return 0;
}
```

OUTPUT:

```
// multimap :
// AAA - 1
// BBB - 2
// CCC - 3
// CCC - 5
// DDD - 6
// EEE - 4
// multimap :
// AAA - 1
// CCC - 3
// CCC - 5
// DDD - 6
```

```
// EEE - 4
// multimap :
// AAA - 1
// CCC - 3
// CCC - 5
// EEE - 4
// multimap :
// CCC - 3
// CCC - 5
// EEE - 4
```

find

```
finds a given element
#include <iostream>
#include <map>
using namespace std;

int main ()
{
     typedef multimap<int,char> M;
     typedef M::value_type v_t;
     char ch = 'A';
     M m;

     for ( int i=0; i<5; i++ )
         m.insert(v_t(i,ch++));
     m.insert(v_t(4,'F'));

     M::iterator It = m.begin();
     cout << "multimap m:" << endl;

     while ( It != m.end() )
     {
         cout << (*It).first << " - "
              << (*It).second << endl;
         It++;
     }

     It = m.find(4);
     if ( It != m.end() )
         cout << "element key '4' has value "
```

```
                << (*It).second << endl;
     else
         cout << "element key '4' not found"
              << endl;

     M::iterator upp = m.upper_bound(4);
     cout << "all elements with key '4'" << endl;
     while ( It != upp )
     {
         cout << (*It).first << " - "
              << (*It).second << endl;
         It++;
     }
     return 0;
}
```

OUTPUT:

```
// multimap m:
// 0 - A
// 1 - B
// 2 - C
// 3 - D
// 4 - E
// 4 - F
// element key '4' has value E
// all elements with key '4'
// 4 - E
// 4 - F
```

insert

---

inserts elements into the maltimap

```
#include <iostream>
#include <map>
#include <string>
using namespace std;

int main ()
{
     typedef multimap<int, char, less<char> > M;
     typedef M::value_type v_t;

     M m1, m2;
```

```
char ch = 'A';
for ( int i=0; i<3; i++ )
{
    m1.insert(v_t(i+1,ch+i));
    m2.insert(v_t(i+4,ch+i+3));
}

cout << "m1 :" << endl;
M::iterator It = m1.begin();
while ( It != m1.end() )
{
    cout << (*It).first << " - "
         << (*It).second << endl;
    It++;
}

cout << "m2 :" << endl;
It = m2.begin();
while ( It != m2.end() )
{
    cout << (*It).first << " - "
         << (*It).second << endl;
    It++;
}

// insert new element
m1.insert(v_t(5,'E'));

It = m2.find(6);
// insert element pointed by It
m1.insert(*It);

cout << "m1 :" << endl;
It = m1.begin();
while ( It != m1.end() )
{
    cout << (*It).first << " - "
         << (*It).second << endl;
    It++;
}

// insert the range of elements
m1.insert(m2.begin(),m2.end());
```

```
    cout << "m1 :" << endl;
    It = m1.begin();
    while ( It != m1.end() )
    {
        cout << (*It).first << " - "
             << (*It).second << endl;
        It++;
    }

    return 0;
}
```

OUTPUT:

```
// m1 :
// 1 - A
// 2 - B
// 3 - C
// m2 :
// 4 - D
// 5 - E
// 6 - F
// m1 :
// 1 - A
// 2 - B
// 3 - C
// 5 - E
// 6 - F
// m1 :
// 1 - A
// 2 - B
// 3 - C
// 4 - D
// 5 - E
// 5 - E
// 6 - F
// 6 - F
```

lower_bound

```
returns an iterator to the first element greater
than a certain value
#include <iostream>
```

```
#include <map>
#include <string>
#include <fstream>
using namespace std;

int main ()
{
     typedef multimap<char,string> M1;
     typedef M1::value_type v_t1;
     M1 m1;
     typedef multimap<string,char,less<string> > M2;
     typedef M2::value_type v_t2;
     M2 m2;

     string word;
     int counter = 0;

     ifstream In("/usr/share/dict/words");
     if ( In.good() )
     {
         while(1)
         {
             getline(In,word);
             char ch = word.at(0);
             // file is sorted
             if ( ch != 'A' && ch != 'a' )
                 break;
             else
             {
                 // for counting of words
                 m1.insert(v_t1(ch,word));
                 // for upper-lower bound
                 m2.insert(v_t2(word,ch));
             }
             counter++;
         }
         In.close();
     }

     cout << "System Dictionary consists " << counter
          << " words,\nwith first letter 'a' or 'A'"
          << endl;
     cout << m1.count('A') << " words start with 'A'"
          << endl;
```

```
    cout << m1.count('a') << " words start with 'a'"
         << endl;

    M2::iterator low = m2.lower_bound("Aba");
    M2::iterator upp = m2.upper_bound("Abe");
    cout << "Range of the words from 'Aba' to 'Abe':"
         << endl;
    while ( low != upp )
    {
        cout << (*low).first << endl;
        low++;
    }
    return 0;
}
```

OUTPUT:

```
// System Dictionary consists 3577 words,
// with first letter 'a' or 'A'
// 491 words start with 'A'
// 3086 words start with 'a'
// Range of the words from 'Aba' to 'Abe':
// Ababa
// Abba
// Abbott
// Abby
// Abe
```

key_comp

returns the function that compares keys

```
#include <iostream>
#include <map>
```

OUTPUT:

max_size

the maximum number of elements that the maltimap can hold

```
#include <iostream>
#include <map>
using namespace std;
```

```
int main ()
{
     typedef multimap<int, char, greater<int> > M;
     typedef M::value_type v_t;
     M m;

     m.insert(v_t(2,'B'));
     m.insert(v_t(3,'C'));
     m.insert(v_t(1,'A'));

     M::iterator It = m.begin();
     cout << "m:" << endl;
     while ( It != m.end() )
     {
         cout << (*It).first << " - "
              << (*It).second
              << endl;
         It++;
     }
         cout << "size of multimap 'm' "
                 << m.size() << endl;
         cout << "max_size of 'm'       "
                 << m.max_size() << endl;

     return 0;
}
```

OUTPUT:

```
// m:
// 3 - C
// 2 - B
// 1 - A
// size of multimap 'm' 3
// max_size of 'm'      4294967295
```

rbegin

returns a reverse iterator to the end of the maltimap

```
#include <iostream>
#include <map>
#include <iomanip>
#include <string>
using namespace std;
```

```
template<class T>
class ID
{
    public:
        ID(T t, T n) : id(t), name(n) {}
        void print ()
        {
            cout.setf(ios::left);
            cout << setw(15) << name.c_str()
                 << id << endl;
            cout.unsetf(ios::left);
        }
    private:
        T id, name;
};
//=====================================
int main ()
{
    typedef ID<string> Id;
    typedef multimap<int, Id> M;
    typedef M::value_type v_t;

    M m;
    m.insert(v_t(1,Id("000123","Shevchenko")));
    m.insert(v_t(2,Id("000124","Pushkin")));
    m.insert(v_t(3,Id("000125","Shakespeare")));
    // same key
    m.insert(v_t(3,Id("000126","Smith")));

    M::reverse_iterator It = m.rbegin();
    while ( It != m.rend() )
    {
        cout.setf(ios::left);
        cout << setw(3) << (*It).first;
        It->second.print();
        It++;
    }

    return 0;
}
```

OUTPUT:

```
// 3  Smith          000126
// 3  Shakespeare    000125
```

```
// 2   Pushkin         000124
// 1   Shevchenko      000123
```

rend

---

returns a reverse iterator to the beginning of the maltimap

```
#include <iostream>
#include <map>
#include <iomanip>
#include <string>
using namespace std;

template<class T>
class ID
{
    public:
        ID(T t, T n) : id(t), name(n) {}
        void print ()
        {
            cout.setf(ios::left);
            cout << setw(15) << name.c_str()
                 << id << endl;
            cout.unsetf(ios::left);
        }
    private:
        T id, name;
};
//=====================================
int main ()
{
    typedef ID<string> Id;
    typedef multimap<int, Id> M;
    typedef M::value_type v_t;

    M m;
    m.insert(v_t(1,Id("000123","Shevchenko")));
    m.insert(v_t(2,Id("000124","Pushkin")));
    m.insert(v_t(3,Id("000125","Shakespeare")));
    // same key
    m.insert(v_t(3,Id("000126","Smith")));

    M::reverse_iterator It = m.rbegin();
```

```
    while ( It != m.rend() )
    {
        cout.setf(ios::left);
        cout << setw(3) << (*It).first;
        It->second.print();
        It++;
    }

    return 0;
}
```

OUTPUT:

```
// 3  Smith          000126
// 3  Shakespeare    000125
// 2  Pushkin        000124
// 1  Shevchenko     000123
```

size

```
the number of elements in the maltimap
#include <iostream>
#include <map>
#include <iomanip>
#include <string>
using namespace std;

template<class T>
class ID
{
    public:
        ID(T t, T n) : id(t), name(n) {}
        void print ()
        {
            cout.setf(ios::left);
            cout << setw(15) << name.c_str()
                 << id << endl;
            cout.unsetf(ios::left);
        }
    private:
        T id, name;
};
//=====================================
int main ()
```

```
{
    typedef ID<string> Id;
    typedef multimap<int, Id> M;
    typedef M::value_type v_t;

    M m;
    m.insert(v_t(1,Id("000123","Shevchenko")));
    m.insert(v_t(2,Id("000124","Pushkin")));
    m.insert(v_t(3,Id("000125","Shakespeare")));
    // same key
    m.insert(v_t(3,Id("000126","Smith")));

    cout << "size of multimap 'm' = "
         << m.size() << endl;

    M::iterator It = m.begin();
    while ( It != m.end() )
    {
        cout.setf(ios::left);
        cout << setw(3) << (*It).first;
        It->second.print();
        It++;
    }

    return 0;
}
OUTPUT:
// size of multimap 'm' = 4
// 1  Shevchenko     000123
// 2  Pushkin        000124
// 3  Shakespeare    000125
// 3  Smith          000126
```

swap

```
exchanges two maltimaps
#include <iostream>
#include <map>
#include <list>
#include <numeric>
#include <algorithm>
using namespace std;
```

```
typedef multimap<int, list<int> > M;

void print (M m)
{
    M::iterator It = m.begin();
    list<int>::iterator Li;
    while ( It != m.end() )
    {
        cout << "key : " << (*It).first
             << "; value : ";
        for ( Li = It->second.begin();
                Li != It->second.end(); Li++ )
            cout << *Li << " ";
        It++;
    }
    cout << endl;
}
//-----------------------------------------
int main ()
{
    list<int> L1, L2;
    L1.push_back(1);
    L1.push_back(2);
    L1.push_back(3);
    copy(L1.begin(),L1.end(),
            back_inserter(L2));
    M m1, m2;

    m1.insert(M::value_type(1,L1));
    m2.insert(M::value_type(2,L2));

    cout << "multimap m1:" << endl;
    print(m1);
    cout << "multimap m2:" << endl;
    print(m2);

    if ( m1 == m2 )
        cout << "multimaps m1 and m2 are equal"
             << endl;
    else
    {
        cout << endl << "After m1.swap(m2)"
             << endl;
```

```
        m1.swap(m2);
        cout << "multimap m1:" << endl;
        print(m1);
        cout << "multimap m2:" << endl;
        print(m2);
    }

    return 0;
}
OUTPUT:
// multimap m1:
// key : 1; value : 1 2 3
// multimap m2:
// key : 2; value : 1 2 3
//
// After m1.swap(m2)
// multimap m1:
// key : 2; value : 1 2 3
// multimap m2:
// key : 1; value : 1 2 3
```

upper_bound

```
returns an iterator to the first element greater than
a certain value
#include <iostream>
#include <map>
#include <string>
#include <fstream>
using namespace std;

int main ()
{
     typedef multimap<char,string> M1;
     typedef M1::value_type v_t1;
     M1 m1;
     typedef multimap<string,char,less<string> > M2;
     typedef M2::value_type v_t2;
     M2 m2;

     string word;
     int counter = 0;
```

```
    ifstream In("/usr/share/dict/words");
    if ( In.good() )
    {
        while(1)
        {
            getline(In,word);
            char ch = word.at(0);
            // file is sorted
            if ( ch != 'A' && ch != 'a' )
                break;
            else
            {
                // for conting of words
                m1.insert(v_t1(ch,word));
                // for upper-lower bound
                m2.insert(v_t2(word,ch));
            }
            counter++;
        }
        In.close();
    }

    cout << "System Dictionary consists " << counter
         << " words,\nwith first letter 'a' or 'A'"
         << endl;
    cout << m1.count('A') << " words start with 'A'"
         << endl;
    cout << m1.count('a') << " words start with 'a'"
         << endl;

    M2::iterator low = m2.lower_bound("Aba");
    M2::iterator upp = m2.upper_bound("Abe");
    cout << "Range of the words from 'Aba' to 'Abe':"
         << endl;
    while ( low != upp )
    {
        cout << (*low).first << endl;
        low++;
    }
    return 0;
}
```

OUTPUT:

```
// System Dictionary consists 3577 words,
```

```
// with first letter 'a' or 'A'
// 491 words start with 'A'
// 3086 words start with 'a'
// Range of the words from 'Aba' to 'Abe':
// Ababa
// Abba
// Abbott
// Abby
// Abe
```

value_comp

```
returns the function that compares values
#include <iostream>
#include <map>
OUTPUT:
```

//容器适配器

stack

all stack functions

```
// The C++ Stack is a container adapter that gives the
// programmer the functionality of a stack -- specifically,
// a FILO (first-in, last-out) data structure.
// push, pop, size, top, empty
#include <iostream>
#include <stack>
#include <vector>
#include <algorithm>
#include <numeric>
using namespace std;

int main ()
{
    vector<int> v1(5), v2(5), v3(5);
```

```
    iota(v1.begin(), v1.end(), 0);
    iota(v2.begin(), v2.end(), 5);
    iota(v3.begin(), v3.end(), 10);

    stack<vector<int> > s;
    s.push(v1);
    s.push(v2);
    s.push(v3);

    cout << "size of stack 's' = "
         << s.size() << endl;

    if ( v3 != v2 )
        s.pop();

    cout << "size of stack 's' = "
         << s.size() << endl;

    vector<int> top = s.top();
    cout << "Contents of v2 : ";

    copy(top.begin(),top.end(),
            ostream_iterator(cout," "));
    cout << endl;

    while ( !s.empty() )
        s.pop();

    cout << "Stack 's' is " << (s.empty() ? ""
            : "not ") << "empty" << endl;

    return 0;
}
OUTPUT:
// size of stack 's' = 3
// size of stack 's' = 2
// Contents of v2 : 5 6 7 8 9
// Stack 's' is empty
```

Queue

all queue functions

```
// Queue is a container adapter that gives
// the programmer a FIFO (first-in, first-out)
// data structure.
// push, pop, size, front, back, empty
#include <iostream>
#include <queue>
#include <string>
using namespace std;

int main ()
{
    string s1("C++");
    string s2("is");
    string s3("powerfull");
    string s4("language");

    queue que;
    que.push(s1);
    que.push(s2);
    que.push(s3);
    que.push(s4);

    cout << "size of queue 'que' = "
         << que.size() << endl;

    string temp = que.back();
    cout << temp << endl;

    while ( !que.empty() )
    {
        temp = que.front();
        cout << temp << " ";
        que.pop();
    }
    cout << endl;
    return 0;
}
OUTPUT:
// size of queue 'que' = 4
// language
// C++ is powerfull language
```

priority_queue

all priority_queue functions

```
// Priority Queues are like queues, but the
// elements inside the data structure are
// ordered by some predicate.
#include <iostream>
#include <queue>
#include <vector>
#include <string>
using namespace std;

int main ()
{
    priority_queue<int, vector<int>, less<int> > ipq;
    ipq.push(100);
    ipq.push(200);
    ipq.push(300);

    cout << "size of priority_queue ipq = "
         << ipq.size() << endl;

    cout << "ipq <int,vector<int>, less<int> > = ";
    while ( !ipq.empty() )
    {
        cout << ipq.top() << " ";
        ipq.pop();
    }
    cout << endl << endl;

    cout << "priority_queue<string,vector<string> > spq;"
         << endl;

    priority_queue<string,vector<string> > spq;
    for ( int i=1; i<10; i++ )
        spq.push(string(i,'*'));

    while ( !spq.empty() )
    {
        cout << spq.top() << endl;
```

```
        spq.pop();
    }

    return 0;
}
```

OUTPUT:

```
// size of priority_queue ipq = 3
// ipq <sring,vector<string> > = 300 200 100
//
// priority_queue<string,vector<string> > spq;
// *********
// ********
// *******
// ******
// *****
// ****
// ***
// **
// *
```