Руководство по стандартной библиотеке шаблонов (STL)

Примеры программ с шаблонами

Эти примеры демонстрируют использование нового продукта *STL<ToolKit>* от компании *ObjectSpace*. *STL<ToolKit>* - это самый простой способ использования *STL*, который работает на большинстве комбинаций платформ/компиляторов, включая *cfront*, *Borland*, *Visual C++*, *Set C++*, *ObjectCenter* и последние компиляторы от *Sun&HP*.

accum1.cpp

```
#include <ospace/stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v (5);
  for (int i = 0; i < v.size (); i++)
    v[i] = i + 1;
  int sum = accumulate (v.begin (), v.end (), 0);
  cout << "sum = " << sum << endl;
  return 0;
}
```

accum2.cpp

```
#include <stl.h>
#include <iostream.h>

int mult (int initial_, int element_)
{
  return initial_ * element_;
}

int main ()
{
  vector <int> v (5);
  for (int i = 0; i < v.size (); i++)
    v[i] = i + 1;
  int prod = accumulate (v.begin (), v.end (), 1, mult);
  cout << "prod = " << prod << endl;
  return 0;
}
```

search2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_equal (const char* a_, const char* b_)
```

```
{
  return ::strcmp (a_, b_) == 0 ? 1 : 0;
}

char* grades[] = { "A", "B", "C", "D", "F" };
char* letters[] = { "Q", "E", "D" };

int main ()
{
  const unsigned gradeCount = sizeof (grades) / sizeof (grades[0]);
  const unsigned letterCount = sizeof (letters) / sizeof (letters[0]);
  ostream_iterator <char*> iter (cout, " ");
  cout << "grades: ";
  copy (grades, grades + gradeCount, iter);
  cout << "\nletters:";
  copy (letters, letters + letterCount, iter);
  cout << endl;

  char** location =
    search (grades, grades + gradeCount,
            letters, letters + letterCount,
            str_equal);

  if (location == grades + gradeCount)
    cout << "letters not found in grades" << endl;
  else
    cout << "letters found in grades at offset: " << location - grades << endl;

  copy (grades + 1, grades + 1 + letterCount, letters);

  cout << "grades: ";
  copy (grades, grades + gradeCount, iter);
  cout << "\nletters:";
  copy (letters, letters + letterCount, iter);
  cout << endl;

  location = search (grades, grades + gradeCount,
                     letters, letters + letterCount,
                     str_equal);

  if (location == grades + gradeCount)
    cout << "letters not found in grades" << endl;
  else
    cout
      << "letters found in grades at offset: " << location - grades << endl;
  return 0;
}
```

incl2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool compare_strings (const char* s1_, const char* s2_)
{
  return ::strcmp (s1_, s2_) < 0 ? 1 : 0;
}

char* names[] = {  "Todd", "Mike", "Graham", "Jack", "Brett"};

int main ()
{
  const unsigned nameSize = sizeof (names)/sizeof (names[0]);
  vector <char*> v1(nameSize);
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = names[i];
  }
  vector <char*> v2 (2);
  v2[0] = "foo";
  v2[1] = "bar";
  sort (v1.begin (), v1.end (), compare_strings);
  sort (v2.begin (), v2.end (), compare_strings);

  bool inc = includes (v1.begin (), v1.end (),
                       v2.begin (), v2.end (),
                       compare_strings);
  if (inc)
    cout << "v1 includes v2" << endl;
  else
    cout << "v1 does not include v2" << endl;
  v2[0] = "Brett";
  v2[1] = "Todd";
  inc = includes (v1.begin (), v1.end (),
                  v2.begin (), v2.end (),
                  compare_strings);
  if (inc)
    cout << "v1 includes v2" << endl;
  else
    cout << "v1 does not include v2" << endl;
  return 0;
}
```

search1.cpp

```
  #include <stl.h>
  #include <iostream.h>

  int main ()
  {
    typedef vector <int> IntVec;
    IntVec v1 (10);
    iota (v1.begin (), v1.end (), 0);
    IntVec v2 (3);
    iota (v2.begin (), v2.end (), 50);
    ostream_iterator <int> iter (cout, " ");

    cout << "v1: ";
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
    cout << "v2: ";
    copy (v2.begin (), v2.end (), iter);
    cout << endl;

    IntVec::iterator location;
    location = search (v1.begin (), v1.end (), v2.begin (), v2.end ());

    if (location == v1.end ())
      cout << "v2 not contained in v1" << endl;
    else
      cout << "Found v2 in v1 at offset: " << location - v1.begin () << endl;

    iota (v2.begin (), v2.end (), 4);
    cout << "v1: ";
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
    cout << "v2: ";
    copy (v2.begin (), v2.end (), iter);
    cout << endl;

    location = search (v1.begin (), v1.end (), v2.begin (), v2.end ());

    if (location == v1.end ())
      cout << "v2 not contained in v1" << endl;
    else
      cout << "Found v2 in v1 at offset: " << location - v1.begin () << endl;

    return 0;
  }
```

istmit2.cpp

```
#include <iostream.h>
#include <fstream.h>
#include <stl.h>

typedef vector<char> Line;

void printLine (const Line* line_)
{
  vector<char>::const_iterator i;
  for (i = line_->begin (); i != line_->end (); i++)
   cout << *i;
  cout << endl;
}

int main ()
{
  Line buffer;
  vector<Line*> lines;
  ifstream s ("data.txt");
  s.unsetf (ios::skipws); // Disable white-space skipping.
  istream_iterator<char, ptrdiff_t> it1 (s); // Position at start of file.
  istream_iterator<char, ptrdiff_t> it2; // Serves as "past-the-end" marker.
  copy (it1, it2, back_inserter (buffer));
  Line::iterator i = buffer.begin ();
  Line::iterator p;
  while (i != buffer.end ())
  {
    p = find (i, buffer.end (), '\n');
    lines.push_back (new Line (i, p));
    i = ++p;
  }
  sort (lines.begin (), lines.end (), less_p<Line*> ());
  cout << "Read " << lines.size () << " lines" << endl;
  vector<Line*>::iterator j;
  for (j = lines.begin (); j != lines.end (); j++)
    printLine (*j);
  release (lines.begin (), lines.end ()); // Release memory.
  return 0;
}
```

alloc1.cpp

```
#include <stl.h>
```

```
#include <ospace/stl/examples/myaloc.h>


int main ()
{
  {
    cout << "vectors:" << endl;
    os_my_allocator<int> alloc;
    vector<int> v3 (alloc);
    v3.push_back (42);
    vector<int> v4 (alloc);
    v4.push_back (42);
  }

  {
    cout << "bit_vectors:" << endl;
    os_my_allocator<unsigned int> alloc;
    bit_vector v1 (alloc);
    v1.push_back (1);
  }

  {
    cout << "deques:" << endl;
    os_my_allocator<int> alloc;
    deque<int> d (alloc);
    d.push_back (42);
  }

  {
    cout << "lists:" << endl;
    os_my_allocator<os_list_node<int> > alloc;
    list<int> l (alloc);
    l.push_back (42);
  }

  {
    cout << "sets:" << endl;
    os_my_allocator<os_value_node<int> > alloc;
    set<int, less<int> > s (alloc);
    s.insert (42);
  }

  {
    cout << "maps" << endl;
    os_my_allocator<os_value_node<os_pair<const int, float> > > alloc;
    map<int, float, less<int> > m (alloc);
    m[4] = 2.0;
  }

  return 0;
}
```

release2.cpp

```
#include <stl.h>
#include <iostream.h>

class X
{
  public:
    X (int i_) : i (i_) {}
    ~X () { cout << "Delete X(" << i << ")" << endl; }
    int i;
};

ostream& operator << (ostream& stream_, const X& x_)
{
  return stream_ << "X(" << x_.i << ")";
}

int main ()
{
  vector<X*> v;
  v.push_back (new X (2));
  v.push_back (new X (1));
  v.push_back (new X (4));
  vector<X*>::iterator i;
  cout << "Initial contents:" << endl;
  for (i = v.begin (); i != v.end (); i++)
    cout << "  " << *(*i) << endl;
  release (v.begin ()); // Delete the first heap-based object.
  v.erase (v.begin ()); // Erase the first element.
  cout << "Remaining contents:" << endl;
  for (i = v.begin (); i != v.end (); i++)
    cout << "  " << *(*i) << endl;
  release (v.begin (), v.end ()); // Delete remaining heap-based objects.
  v.erase (v.begin (), v.end ()); // Erase remaining elements.
  return 0;
}
```

map1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  typedef map<char, int, less<char> > maptype;
  maptype m;
  // Store mappings between roman numerals and decimals.
  m['l'] = 50;
  m['x'] = 20; // Deliberate mistake.
  m['v'] = 5;
  m['i'] = 1;
  cout << "m['x'] = " << m['x'] << endl;
  m['x'] = 10; // Correct mistake.
  cout << "m['x'] = " << m['x'] << endl;
  cout << "m['z'] = " << m['z'] << endl; // Note default value is added.
  cout << "m.count ('z') = " << m.count ('z') << endl;
  pair<maptype::iterator, bool> p;
  p = m.insert (pair<const char, int> ('c', 100));
  if (p.second)
    cout << "First insertion successful" << endl;
  p = m.insert (pair<const char, int> ('c', 100));
  if (p.second)
    cout << "Second insertion successful" << endl;
  else
    cout << "Existing pair " << (*(p.first)).first
         << " -> " << (*(p.first)).second << endl;
  return 0;
}
```

mismtch2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_equal (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) == 0 ? 1 : 0;
}

const unsigned size = 5;
char* n1[size] = { "Brett", "Graham", "Jack", "Mike", "Todd" };

int main ()
{
  char* n2[size];
  copy (n1, n1 + 5, n2);
  pair <char**, char**> result;
  result = mismatch (n1, n1+ size, n2, str_equal);
  if (result.first == n1 + size && result.second == n2 + size)
    cout << "n1 and n2 are the same" << endl;
  else
    cout << "mismatch at index: " << (result.first - n1) << endl;
  n2[2] = "QED";
  result = mismatch (n1, n1 + size, n2, str_equal);
  if (result.first == n2 + size && result.second == n2 + size)
    cout << "n1 and n2 are the same" << endl;
  else
    cout << "mismatch at index: " << (result.first - n1) << endl;
  return 0;
}
```

mismtch1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  typedef vector<int> IntVec;
  IntVec v1 (10);
  IntVec v2 (v1.size ());
  iota (v1.begin (), v1.end (), 0);
  iota (v2.begin (), v2.end (), 0);
  pair <IntVec::iterator, IntVec::iterator> result;
  result = mismatch (v1.begin (), v1.end (), v2.begin ());
  if (result.first == v1.end () && result.second == v2.end ())
    cout << "v1 and v2 are the same" << endl;
  else
    cout << "mismatch at index: " << (result.first - v1.begin ()) << endl;
  v2[v2.size()/2] = 42;
  result = mismatch (v1.begin (), v1.end (), v2.begin ());
  if (result.first == v1.end () && result.second == v2.end ())
    cout << "v1 and v2 are the same" << endl;
  else
    cout << "mismatch at index: " << (result.first - v1.begin ()) << endl;
  return 0;
}
```

mmap2.cpp

```
#include <iostream.h>
#include <stl.h>

typedef multimap<int, char, less<int> > mmap;

typedef pair<const int, char> pair_type;

pair_type p1 (3, 'c');
pair_type p2 (6, 'f');
pair_type p3 (1, 'a');
pair_type p4 (2, 'b');
pair_type p5 (3, 'x');
pair_type p6 (6, 'f');

pair_type array [] =
  {
    p1,
    p2,
    p3,
    p4,
    p5,
    p6
  };

int main ()
{
  mmap m (array, array + 7);
  mmap::iterator i;
  // Return location of first element that is not less than 3
  i = m.lower_bound (3);
  cout << "lower bound:" << endl;
  cout << (*i).first << " -> " << (*i).second << endl;
  // Return location of first element that is greater than 3
  i = m.upper_bound (3);
  cout << "upper bound:" << endl;
  cout << (*i).first << " -> " << (*i).second << endl;
  return 0;
}
```

adjfind2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

typedef vector <char*> CStrVector;

int equal_length (const char* v1_, const char* v2_)
{
  return ::strlen (v1_) == ::strlen(v2_);
}

char* names[] = { "Brett", "Graham", "Jack", "Mike", "Todd" };

int main ()
{
  const int nameCount = sizeof (names)/sizeof(names[0]);
  CStrVector v (nameCount);
  for (int i = 0; i < nameCount; i++)
    v[i] = names[i];
  CStrVector::iterator location;
  location = adjacent_find (v.begin (), v.end (), equal_length);
  if (location != v.end ())
    cout
      << "Found two adjacent strings of equal length: "
      << *location
      << " -and- "
      << *(location + 1)
      << endl;
  else
    cout << "Didn't find two adjacent strings of equal length.";
  return 0;
}
```

list3.cpp

```
#include <iostream.h>
#include <stl.h>

char array [] = { 'x', 'l', 'x', 't', 's', 's' };

int main ()
{
  list<char> str (array, array + 6);
  list<char>::iterator i;
  cout << "original: ";
  for (i = str.begin (); i != str.end (); i++)
    cout << *i;
  cout << endl;
  cout << "reversed: ";
  str.reverse ();
  for (i = str.begin (); i != str.end (); i++)
    cout << *i;
  cout << endl;
  cout << "removed: ";
  str.remove ('x');
  for (i = str.begin (); i != str.end (); i++)
    cout << *i;
  cout << endl;
  cout << "uniqued: ";
  str.unique ();
  for (i = str.begin (); i != str.end (); i++)
    cout << *i;
  cout << endl;
  cout << "sorted: ";
  str.sort ();
  for (i = str.begin (); i != str.end (); i++)
    cout << *i;
  cout << endl;
  return 0;
}
```

parsrtc2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}
```

```
char* names[] = { "aa", "ff", "dd", "ee", "cc", "bb" };

int main ()
{
  const unsigned nameSize = sizeof (names) / sizeof (names[0]);
  vector <char*> v1 (nameSize);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = names[i];
  ostream_iterator<char*> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  vector <char*> result (5);
  partial_sort_copy (v1.begin (),
                     v1.end (),
                     result.begin (),
                     result.end (),
                     str_compare);
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

vec6.cpp

```
#include <iostream.h>
#include <stl.h>

int array [] = { 1, 4, 9, 16, 25, 36 };

int main ()
{
  vector<int> v (array, array + 6);
  for (int i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  cout << endl;
  v.erase (v.begin ()); // Erase first element.
  for (i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  cout << endl;
  v.erase (v.end () - 1); // Erase last element.
  for (i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  cout << endl;
  v.erase (v.begin () + 1, v.end () - 1); // Erase all but first and last.
  for (i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  cout << endl;
  v.erase (); // Erase all.
  return 0;
}
```

inrprod2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

int add (int a_, int b_)
{
  return a_ + b_;
}

int mult (int a_, int b_)
{
  return a_ * b_;
}

int main ()
{
  vector <int> v1 (3);
  vector <int> v2 (v1.size ());
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = i + 1;
    v2[i] = v1.size () - i;
  }
  ostream_iterator<int> iter (cout, " ");
  cout << "Inner product (product of sums):\n\t";
  copy (v1.begin (), v1.end (), iter);
  cout << "\n\t";
  copy (v2.begin (), v2.end (), iter);
  int result =
    inner_product (v1.begin (), v1.end (),
                   v2.begin (),
                   1,
                   mult, add);
  cout << "\nis: " << result << endl;
  return 0;
}
```

mmap1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  typedef multimap<char, int, less<char> > mmap;
  mmap m;
  cout << "count ('X') = " << m.count ('X') << endl;
  m.insert (pair<const char, int> ('X', 10)); // Standard way.
  cout << "count ('X') = " << m.count ('X') << endl;
  m.insert ('X', 20); // Non-standard, but very convenient!
  cout << "count ('X') = " << m.count ('X') << endl;
  m.insert ('Y', 32);
  mmap::iterator i = m.find ('X'); // Find first match.
  while (i != m.end ()) // Loop until end is reached.
  {
    cout << (*i).first << " -> " << (*i).second << endl;
    i++;
  }
  int count = m.erase ('X');
  cout << "Erased " << count << " items" << endl;
  return 0;
}
```

adjfind0.cpp

```
#include <stl.h>
```

```
#include <iostream.h>

int numbers1 [5] = { 1, 2, 4, 8, 16 };
int numbers2 [5] = { 5, 3, 2, 1, 1 };

int main ()
{
  int* location = adjacent_find (numbers1, numbers1 + 5);

  if (location != numbers1 + 5)
    cout
      << "Found adjacent pair of: "
      << *location
      << " at offset "
      << (location - numbers1)
      << endl;
  else
    cout << "No adjacent pairs" << endl;
  location = adjacent_find (numbers2, numbers2 + 5);
  if (location != numbers2 + 5)
    cout
      << "Found adjacent pair of: "
      << *location
      << " at offset "
      << (location - numbers2)
      << endl;
  else
    cout << "No adjacent pairs" << endl;
  return 0;
}
```

parsrt2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* names[] = { "aa", "ff", "dd", "ee", "cc", "bb" };

int main ()
{
  const unsigned nameSize = sizeof (names) / sizeof (names[0]);
  vector <char*> v1 (nameSize);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = names[i];
  ostream_iterator<char*> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  partial_sort (v1.begin (),
                v1.begin () + nameSize / 2,
                v1.end (),
                str_compare);
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

mset5.cpp

```
#include <iostream.h>
#include <stl.h>

bool less_than (int a_, int b_)
{
  return a_ < b_;
}

bool greater_than (int a_, int b_)
{
  return a_ > b_;
}

int array [] = { 3, 6, 1, 9 };

int main ()
{
  typedef pointer_to_binary_function<int, int, bool> fn_type;
  typedef multiset<int, fn_type> mset;
  fn_type f (less_than);
  mset s1 (array, array + 4, f);
  mset::const_iterator i = s1.begin ();
  cout << "Using less_than: " << endl;
  while (i != s1.end ())
    cout << *i++ << endl;
  fn_type g (greater_than);
  mset s2 (array, array + 4, g);
  i = s2.begin ();
  cout << "Using greater_than: " << endl;
  while (i != s2.end ())
    cout << *i++ << endl;
  return 0;
}
```

mset1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  typedef multiset<int, less<int> > mset;
  mset s;
  cout << "count (42) = " << s.count (42) << endl;
  s.insert (42);
  cout << "count (42) = " << s.count (42) << endl;
  s.insert (42);
  cout << "count (42) = " << s.count (42) << endl;
  set<int, less<int> >::iterator i = s.find (40);
  if (i == s.end ())
    cout << "40 Not found" << endl;
  else
    cout << "Found " << *i << endl;
  i = s.find (42);
  if (i == s.end ())
    cout << "Not found" << endl;
  else
    cout << "Found " << *i << endl;
  int count = s.erase (42);
  cout << "Erased " << count << " instances" << endl;
  return 0;
}
```

vec2.cpp

```
#include <iostream.h>
#include <stl.h>

void print (vector<double>& vector_)
{
  for (int i = 0; i < vector_.size (); i++)
    cout << vector_[i] << " ";
  cout << endl;
}

int main ()
{
  vector<double> v1; // Empty vector of doubles.
  v1.push_back (32.1);
  v1.push_back (40.5);
  vector<double> v2; // Another empty vector of doubles.
  v2.push_back (3.56);
  cout << "v1 = ";
  print (v1);
  cout << "v2 = ";
  print (v2);
  v1.swap (v2); // Swap the vector's contents.
  cout << "v1 = ";
  print (v1);
  cout << "v2 = ";
  print (v2);
  v2 = v1; // Assign one vector to another.
  cout << "v2 = ";
  print (v2);
  return 0;
}
```

uniqcpy2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_equal (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) == 0 ? 1 : 0;
}

char* labels[] = { "Q","Q","W","W","E","E","R","T","T","Y","Y" };

int main ()
{
  const unsigned count = sizeof (labels) / sizeof (labels[0]);
  ostream_iterator <char*> iter (cout);
  copy (labels, labels + count, iter);
  cout << endl;
  char* uCopy[count];
  fill (uCopy, uCopy + count, "");
  unique_copy (labels, labels + count, uCopy, str_equal);
  copy (labels, labels + count, iter);
  cout << endl;
  copy (uCopy, uCopy + count, iter);
  cout << endl;
  return 0;
}
```

mismtch0.cpp

```
#include <stl.h>
#include <iostream.h>

int n1[5] = { 1, 2, 3, 4, 5 };
int n2[5] = { 1, 2, 3, 4, 5 };
int n3[5] = { 1, 2, 3, 2, 1 };

int main ()
{
  pair <int*, int*> result;
  result = mismatch (n1, n1 + 5, n2);
  if (result.first == (n1 + 5) && result.second == (n2 + 5))
    cout << "n1 and n2 are the same" << endl;
  else
    cout << "Mismatch at offset: " << (result.first - n1) << endl;
  result = mismatch (n1, n1 + 5, n3);
  if (result.first == (n1 + 5) && result.second == (n3 + 5))
    cout << "n1 and n3 are the same" << endl;
  else
    cout << "Mismatch at offset: " << (result.first - n1) << endl;
  return 0;
}
```

rndshuf2.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>

class MyRandomGenerator
{
  public:
    unsigned long operator () (unsigned long n_);
};

unsigned long
MyRandomGenerator::operator () (unsigned long n_)
{
  return rand () % n_;
}

int main ()
{
  vector <int> v1(10);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  MyRandomGenerator r;
  for (int i = 0; i < 3; i++)
  {
    random_shuffle (v1.begin (), v1.end (), r);
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
  }
  return 0;
}
```

merge2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (5);
  vector <int> v2 (v1.size ());
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = 10 - i;
    v2[i] =  7 - i;
  }
  vector <int> result (v1.size () + v2.size ());
  merge (v1.begin (), v1.end (),
         v2.begin (), v2.end (),
         result.begin (),
         greater<int>() );
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  copy (result.begin (), result.end (), iter);
  cout << endl;
  return 0;
}
```

adjfind1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  typedef vector<int> IntVector;
  IntVector v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = i;
  IntVector::iterator location;
  location = adjacent_find (v.begin (), v.end ());
  if (location != v.end ())
    cout << "Found adjacent pair of: " << *location << endl;
  else
    cout << "No adjacent pairs" << endl;
  v[6] = 7;
  location = adjacent_find (v.begin (), v.end ());
  if (location != v.end ())
    cout << "Found adjacent pair of: " << *location << endl;
  else
    cout << "No adjacent pairs" << endl;
  return 0;
}
```

vec7.cpp

```
#include <iostream.h>
#include <stl.h>

int array1 [] = { 1, 4, 25 };
int array2 [] = { 9, 16 };

int main ()
{
  vector<int> v (array1, array1 + 3);
  v.insert (v.begin (), 0); // Insert before first element.
  v.insert (v.end (), 36); // Insert after last element.
  for (int i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  cout << endl;
  // Insert contents of array2 before fourth element.
  v.insert (v.begin () + 3, array2, array2 + 2);
  for (i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  cout << endl;
  return 0;
}
```

bcompos1.cpp

```
#include <iostream.h>
#include <stl.h>

struct odd : public unary_function<int, bool>
{
  odd () {}
  bool operator () (int n_) const { return (n_ % 2) == 1; }
};

struct positive : public unary_function<int, bool>
{
  positive () {}
  bool operator () (int n_) const { return n_ >= 0; }
};

int array [6] = { -2, -1, 0, 1, 2, 3 };

int main ()
{
  binary_compose<logical_and<bool>, odd, positive>
    b (logical_and<bool> (), odd (), positive ());
  int* p = find_if (array, array + 6, b);
  if (p != array + 6)
    cout << *p << " is odd and positive" << endl;

  return 0;
}
```

setsymd2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>
```

```
char* word1 = "ABCDEFGHIJKLMNO";
char* word2 = "LMNOPQRSTUVWXYZ";

int main ()
{
  ostream_iterator <char> iter (cout, " ");
  cout << "word1: ";
  copy (word1, word1 + ::strlen (word1), iter);
  cout << "\nword2: ";
  copy (word2, word2 + ::strlen (word2), iter);
  cout << endl;
  set_symmetric_difference (word1, word1 + ::strlen (word1),
                            word2, word2 + ::strlen (word2),
                            iter,
                            less<char>());
  cout << endl;
  return 0;
}
```

search0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[6] = { 1, 1, 2, 3, 5, 8 };
int v2[6] = { 0, 1, 2, 3, 4, 5 };
int v3[2] = { 3, 4 };

int main ()
{
  int* location;
  location = search (v1, v1 + 6, v3, v3 + 2);
  if (location == v1 + 6)
    cout << "v3 not contained in v1" << endl;
  else
    cout << "Found v3 in v1 at offset: " << location - v1 << endl;
  location = search (v2, v2 + 6, v3, v3 + 2);
  if (location == v2 + 6)
    cout << "v3 not contained in v2" << endl;
  else
    cout << "Found v3 in v2 at offset: " << location - v2 << endl;
  return 0;
}
```

eqlrnge1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  typedef vector <int> IntVec;
  IntVec v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = i / 3;
  ostream_iterator<int> iter (cout, " ");
  cout << "Within the collection:\n\t";
  copy (v.begin (), v.end (), iter);
  pair <IntVec::iterator, IntVec::iterator> range;
  range = equal_range (v.begin (), v.end (), 2);
  cout
    << "\n2 can be inserted from before index "
    << (range.first - v.begin ())
    << " to before index "
    << (range.second - v.begin ())
    << endl;
  return 0;
}
```

rotcopy1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  vector <int> v2 (v1.size ());
  for (int i = 0; i < v1.size (); i++)
  {
    rotate_copy (v1.begin (),
                 v1.begin () + i,
                 v1.end (),
                 v2.begin ());
    ostream_iterator <int> iter (cout, " ");
    copy (v2.begin (), v2.end (), iter);
    cout << endl;
  }
  cout << endl;
  return 0;
}
```

eqlrnge2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

char chars[] = "aabbccddggghhklllmqqqqssyyzz";

int main ()
{
  const unsigned count = sizeof (chars) - 1;
  ostream_iterator<char> iter (cout);
  cout << "Within the collection:\n\t";
  copy (chars, chars + count, iter);
  pair <char*, char*> range;
  range = equal_range (chars, chars + count, 'q', less<char>());
  cout
    << "\nq can be inserted from before index "
    << (range.first - chars)
    << " to before index "
    << (range.second - chars)
    << endl;
  return 0;
}
```

release1.cpp

```
#include <stl.h>
#include <iostream.h>

class X
{
  public:
    X (int i_) : i (i_) {}
    ~X () { cout << "Delete X(" << i << ")" << endl; }
    int i;
};

ostream& operator << (ostream& stream_, const X& x_)
{
  return stream_ << "X(" << x_.i << ")";
}

int main ()
{
  vector<X*> v;
  v.push_back (new X (2));
  v.push_back (new X (1));
  v.push_back (new X (4));
  vector<X*>::iterator i;
  for (i = v.begin (); i != v.end (); i++)
    cout << *(*i) << endl;
  release (v.begin (), v.end ()); // Delete heap-based objects.
  return 0;
}
```

incl1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector<int> v1(10);
  vector<int> v2(3);
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = i;
  }
  if (includes (v1.begin (), v1.end (), v2.begin (), v2.end ()))
    cout << "v1 includes v2" << endl;
  else
    cout << "v1 does not include v2" << endl;
  for (i = 0; i < v2.size (); i++)
    v2[i] = i + 3;
  if (includes (v1.begin (), v1.end (), v2.begin (), v2.end ()))
    cout << "v1 includes v2" << endl;
  else
    cout << "v1 does not include v2" << endl;
  return 0;
}
```

setintr2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

char* word1 = "ABCDEFGHIJKLMNO";
char* word2 = "LMNOPQRSTUVWXYZ";

int main ()
{
  ostream_iterator <char> iter (cout, " ");
  cout << "word1: ";
  copy (word1, word1 + ::strlen (word1), iter);
  cout << "\nword2: ";
  copy (word2, word2 + ::strlen (word2), iter);
  cout << endl;
  set_intersection (word1, word1 + ::strlen (word1),
                    word2, word2 + ::strlen (word2),
                    iter,
                    less<char>());
  cout << endl;
  return 0;
}
```

inrprod1.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

int main ()
{
  vector <int> v1 (3);
  vector <int> v2 (v1.size ());
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = i + 1;
    v2[i] = v1.size () - i;
  }
  ostream_iterator<int> iter (cout, " ");
  cout << "Inner product (sum of products) of:\n\t";
  copy (v1.begin (), v1.end (), iter);
  cout << "\n\t";
  copy (v2.begin (), v2.end (), iter);
  int result = inner_product (v1.begin (), v1.end (), v2.begin (), 0);
  cout << "\nis: " << result << endl;
  return 0;
}
```

merge1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (5);
  vector <int> v2 (v1.size ());
  iota (v1.begin (), v1.end (), 0);
  iota (v2.begin (), v2.end (), 3);
  vector <int> result (v1.size () + v2.size ());
  merge (v1.begin (), v1.end (), v2.begin (), v2.end (), result.begin ());
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  copy (result.begin (), result.end (), iter);
  cout << endl;
  return 0;
}
```

bcompos2.cpp

```
#include <iostream.h>
#include <stl.h>

struct odd : public unary_function<int, bool>
{
  odd () {}
  bool operator () (int n_) const { return (n_ % 2) == 1; }
};

struct positive : public unary_function<int, bool>
{
  positive () {}
  bool operator () (int n_) const { return n_ >= 0; }
};

int array [6] = { -2, -1 , 0, 1, 2, 3 };

int main ()
{
  int* p = find_if (array, array + 6,
    compose2 (logical_and<bool> (), odd (), positive ()));
  if (p != array + 6)
    cout << *p << " is odd and positive" << endl;
  return 0;
}
```

error3.cpp

```
#include <stl.h>

// Compile this code without defining OS_USE_EXCEPTIONS.

void my_handler (int code_, const char* str_)
{
  cout << "Caught " << str_ << " [code " << code_ << "]" << endl;
}

int main ()
{
  os_handler_function_t old_h = os_set_error_handler (my_handler);
  vector<int> v;
  v.pop_back (); // Generates an empty object error.
  cout << "returned from pop_back()" << endl;
  os_set_error_handler (old_h);
  v.pop_back (); // Generates an empty object error.
  cout << "successful termination" << endl;
  return 0;
}
```

incl0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers1[5] = { 1, 2, 3, 4, 5 };
int numbers2[5] = { 1, 2, 4, 8, 16 };
int numbers3[2] = { 4, 8 };

int main ()
{
  if (includes (numbers1, numbers1 + 5, numbers3, numbers3 + 2))
    cout << "numbers1 includes numbers3" << endl;
  else
    cout << "numbers1 does not include numbers3" << endl;
  if (includes (numbers2, numbers2 + 5, numbers3, numbers3 + 2))
    cout << "numbers2 includes numbers3" << endl;
  else
    cout << "numbers2 does not include numbers3" << endl;
  return 0;
}
```

setdiff2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

char* word1 = "ABCDEFGHIJKLMNO";
char* word2 = "LMNOPQRSTUVWXYZ";

int main ()
{
  ostream_iterator <char> iter (cout, " ");
  cout << "word1: ";
  copy (word1, word1 + ::strlen (word1), iter);
  cout << "\nword2: ";
  copy (word2, word2 + ::strlen (word2), iter);
  cout << endl;
  set_difference (word1, word1 + ::strlen (word1),
                  word2, word2 + ::strlen (word2),
                  iter,
                  less<char>());
  cout << endl;
  return 0;
}
```

setunon2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

char* word1 = "ABCDEFGHIJKLMNO";
char* word2 = "LMNOPQRSTUVWXYZ";
```

```
int main ()
{
  ostream_iterator <char> iter (cout, " ");
  cout << "word1: ";
  copy (word1, word1 + ::strlen (word1), iter);
  cout << "\nword2: ";
  copy (word2, word2 + ::strlen (word2), iter);
  cout << endl;
  set_union (word1, word1 + ::strlen (word1),
             word2, word2 + ::strlen (word2),
             iter,
             less<char>());
  cout << endl;
  return 0;
}
```

unique2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_equal (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) == 0 ? 1 : 0;
}

char* labels[] = { "Q","Q","W","W","E","E","R","T","T","Y","Y" };

int main ()
{
  const unsigned count = sizeof (labels) / sizeof (labels[0]);
  ostream_iterator <char*> iter (cout);
  copy (labels, labels + count, iter);
  cout << endl;
  unique (labels, labels + count, str_equal);
  copy (labels, labels + count, iter);
  cout << endl;
  return 0;
}
```

parsrtc1.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = rand () % 10;
  vector <int> result (5);
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  partial_sort_copy (v1.begin (),
                     v1.end (),
                     result.begin (),
                     result.end ());
  copy (result.begin (), result.end (), iter);
  cout << endl;
  return 0;
}
```

equal1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = i;
  vector <int> v2 (10);
  if (equal (v1.begin (), v1.end (), v2.begin ()))
    cout << "v1 is equal to v2" << endl;
  else
    cout << "v1 is not equal to v2" << endl;
  copy (v1.begin (), v1.end (), v2.begin ());
  if (equal (v1.begin (), v1.end (), v2.begin ()))
    cout << "v1 is equal to v2" << endl;
  else
    cout << "v1 is not equal to v2" << endl;
  return 0;
}
```

equal0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers1[5] = { 1, 2, 3, 4, 5 };
int numbers2[5] = { 1, 2, 4, 8, 16 };
int numbers3[2] = { 1, 2 };

int main ()
{
  if (equal (numbers1, numbers1 + 5, numbers2))
    cout << "numbers1 is equal to numbers2" << endl;
  else
    cout << "numbers1 is not equal to numbers2" << endl;
  if (equal (numbers3, numbers3 + 2, numbers1))
    cout << "numbers3 is equal to numbers1" << endl;
  else
    cout << "numbers3 is not equal to numbers1" << endl;
  return 0;
}
```

genern2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <stdlib.h>

class Fibonacci
{
  public:
    Fibonacci () : v1 (0), v2 (1) {}
    int operator () ();
  private:
    int v1;
    int v2;
};

int
Fibonacci::operator () ()
{
  int r = v1 + v2;
  v1 = v2;
  v2 = r;
  return v1;
}

int main ()
{
  vector <int> v1 (10);
  Fibonacci generator;
  generate_n (v1.begin (), v1.size (), generator);
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

gener2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <stdlib.h>

class Fibonacci
{
  public:
    Fibonacci () : v1 (0), v2 (1) {}
    int operator () ();
  private:
    int v1;
    int v2;
};

int
Fibonacci::operator () ()
{
  int r = v1 + v2;
  v1 = v2;
  v2 = r;
  return v1;
}

int main ()
{
  vector <int> v1 (10);
  Fibonacci generator;
  generate (v1.begin (), v1.end (), generator);
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

repcpif1.cpp

```
#include <stl.h>
#include <iostream.h>

bool odd (int a_)
{
  return a_ % 2;
}

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = i % 5;
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  vector <int> v2 (v1.size ());
  replace_copy_if (v1.begin (), v1.end (), v2.begin (), odd, 42);
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

setsymd.cpp

deque1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  deque<int> d;
  d.push_back (4); // Add after end.
  d.push_back (9);
  d.push_back (16);
  d.push_front (1); // Insert at beginning.
  for (int i = 0; i < d.size (); i++)
    cout << "d[" << i << "] = " << d[i] << endl;
  cout << endl;
  d.pop_front (); // Erase first element.
  d[2] = 25; // Replace last element.
  for (i = 0; i < d.size (); i++)
    cout << "d[" << i << "] = " << d[i] << endl;
  return 0;
}
```

findif1.cpp

```
#include <stl.h>
#include <iostream.h>

bool div_3 (int a_)
{
  return a_ % 3 ? 0 : 1;
}

int main ()
{
  typedef vector <int> IntVec;
  IntVec v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = (i + 1) * (i + 1);
  IntVec::iterator iter;
  iter = find_if (v.begin (), v.end (), div_3);
  if (iter != v.end ())
    cout
      << "Value "
      << *iter
      << " at offset "
      << (iter - v.begin ())
      << " is divisible by 3"
      << endl;
  return 0;
}
```

ucompos1.cpp

```
#include <iostream.h>
#include <math.h>
#include <stl.h>

struct square_root : public unary_function<double, double>
{
  square_root () {}
  double operator () (double x_) const { return sqrt (x_); }
};

int input [3] = { -1, -4, -16 };

int main ()
{
  int output [3];
  transform (input, input + 3, output,
    unary_compose<square_root, negate<int> > (square_root (), negate<int> ()));
  for (int i = 0; i < 3; i++)
    cout << output[i] << endl;
  return 0;
}
```

rawiter.cpp

```
#include <iostream.h>
#include <stl.h>

class X
{
  public:
    X (int i_ = 0) : i (i_) {}
    operator int () const { return i; }

  private:
    int i;
};
```

```
int main ()
{
  os_heap_allocator<X> a;
  // Allocate (but do not construct) storage for 5 elements.
  os_heap_allocator<X>::pointer p = a.allocate (5);
  raw_storage_iterator<X*, X> r (p);
  for (int i = 0; i < 5; i++)
    *r++ = X (i);
  for (i = 0; i < 5; i++)
    cout << *p++ << endl;
  return 0;
}
```

set2.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  set<int, less<int> > s;
  pair<set<int, less<int> >::const_iterator, bool> p;
  p = s.insert (42);
  if (p.second)
   cout << "Inserted new element " << *(p.first) << endl;
  else
   cout << "Existing element = " << *(p.first) << endl;
  p = s.insert (42);
  if (p.second)
   cout << "Inserted new element " << *(p.first) << endl;
  else
   cout << "Existing element = " << *(p.first) << endl;
  return 0;
}
```

mset3.cpp

```
#include <iostream.h>
#include <stl.h>

int array [] = { 3, 6, 1, 2, 3, 2, 6, 7, 9 };

int main ()
{
  multiset<int, less<int> > s (array, array + 9);
  multiset<int, less<int> >::iterator i;
  // Return location of first element that is not less than 3
  i = s.lower_bound (3);
  cout << "lower bound = " << *i << endl;
  // Return location of first element that is greater than 3
  i = s.upper_bound (3);
  cout << "upper bound = " << *i << endl;
  return 0;
}
```

binsrch2.cpp

```
#include <stl.h>
```

```
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* labels[] = { "aa", "dd", "ff", "jj", "ss", "zz" };

int main ()
{
  const unsigned count = sizeof (labels) / sizeof (labels[0]);
  if (binary_search (labels, labels + count, "ff", str_compare))
    cout << "ff is in labels." << endl;
  else
    cout << "ff is not in labels." << endl;
  return 0;
}
```

nthelem2.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = rand () % 10;
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  nth_element (v1.begin (),
               v1.begin () + v1.size () / 2,
               v1.end (),
               greater<int>());
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

setintr1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  iota (v1.begin (), v1.end (), 0);
  vector <int> v2 (10);
  iota (v2.begin(), v2.end (), 7);
  ostream_iterator <int> iter (cout, " ");
  cout << "v1: ";
  copy (v1.begin (), v1.end (), iter);
  cout << "\nv2: ";
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  set_intersection (v1.begin (), v1.end (), v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

setdiff1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  iota (v1.begin (), v1.end (), 0);
  vector <int> v2 (10);
  iota (v2.begin(), v2.end (), 7);
  ostream_iterator <int> iter (cout, " ");
  cout << "v1: ";
  copy (v1.begin (), v1.end (), iter);
  cout << "\nv2: ";
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  set_difference (v1.begin (), v1.end (), v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

adjdiff2.cpp

```
#include <stl.h>
#include <iostream.h>

int mult (int a_, int b_)
{
  return a_ * b_;
}

int main ()
{
  vector <int> v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = i + 1;
  vector <int> rslt (v.size ());
  adjacent_difference (v.begin (), v.end (), rslt.begin (), mult);
  ostream_iterator<int> iter (cout, " ");
  copy (v.begin (), v.end (), iter);
  cout << endl;
  copy (rslt.begin (), rslt.end (), iter);
  cout << endl;
  return 0;
}
```

rotate1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
```

```
{
  vector <int> v1 (10);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  for (int i = 0; i < v1.size (); i++)
  {
    rotate (v1.begin (), v1.begin () + i, v1.end ());
    ostream_iterator <int> iter (cout, " ");
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
  }
  cout << endl;
  return 0;
}
```

setunon1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  iota (v1.begin (), v1.end (), 0);
  vector <int> v2 (10);
  iota (v2.begin(), v2.end (), 7);
  ostream_iterator <int> iter (cout, " ");
  cout << "v1: ";
  copy (v1.begin (), v1.end (), iter);
  cout << "\nv2: ";
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  set_union (v1.begin (), v1.end (), v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

insert1.cpp

```
#include <iostream.h>
#include <stl.h>

char* array1 [] = { "laurie", "jennifer", "leisa" };
char* array2 [] = { "amanda", "saskia", "carrie" };

int main ()
{
  deque<char*> names (array1, array1 + 3);
  deque<char*>::iterator i = names.begin () + 2;
  copy (array2, array2 + 3, insert_iterator<deque <char*> > (names, i));
  deque<char*>::iterator j;
  for (j = names.begin (); j != names.end (); j++)
    cout << *j << endl;
  return 0;
}
```

ucompos2.cpp

```
#include <iostream.h>
#include <math.h>
#include <stl.h>

struct square_root : public unary_function<double, double>
{
  square_root () {}
  double operator () (double x_) const { return sqrt (x_); }
};

int input [3] = { -1, -4, -16 };

int main ()
{
  int output [3];
  transform (input, input + 3, output,
    compose1 (square_root (), negate<int> ()));
  for (int i = 0; i < 3; i++)
   cout << output[i] << endl;
  return 0;
}
```

parsrt1.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = rand () % 10;
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  partial_sort (v1.begin (),
                v1.begin () + v1.size () / 2,
                v1.end ());
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

equal2.cpp

```
#include <stl.h>
#include <iostream.h>

bool values_squared (int a_, int b_)
{
  return (a_ * a_ == b_);
}

int main ()
{
  vector <int> v1 (10);
  vector <int> v2 (10);
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = i;
    v2[i] = i * i;
  }
  if (equal (v1.begin (), v1.end (), v2.begin (), values_squared))
    cout << "v2[i] == v1[i] * v1[i]" << endl;
  else
    cout << "v2[i] != v1[i] * v1[i]" << endl;
  return 0;
}
```

inplmrg2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1(10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = (v1.size () - i - 1) % 5;
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  inplace_merge (v1.begin (), v1.begin () + 5,
                 v1.end (),
                 greater<int>());
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

nthelem1.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = rand () % 10;
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  nth_element (v1.begin (),
               v1.begin () + v1.size () / 2,
               v1.end ());
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

vec4.cpp

```
#include <iostream.h>
#include <stl.h>
```

```
int main ()
{
  vector<int> v (4);
  v[0] = 1;
  v[1] = 4;
  v[2] = 9;
  v[3] = 16;
  cout << "front = " << v.front () << endl;
  cout << "back = " << v.back () << ", size = " << v.size () << endl;
  v.push_back (25);
  cout << "back = " << v.back () << ", size = " << v.size () << endl;
  v.pop_back ();
  cout << "back = " << v.back () << ", size = " << v.size () << endl;
  return 0;
}
```

lwrbnd2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool char_str_less (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* str [] = { "a", "a", "b", "b", "q", "w", "z" };

int main ()
{
  const unsigned strCt = sizeof (str)/sizeof (str[0]);
  cout
    << "d can be inserted at index: "
    << (lower_bound (str,  str + strCt, "d", char_str_less) - str)
    << endl;
  return 0;
}
```

pheap2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector<int> v;

  v.push_back (1);
  v.push_back (20);
  v.push_back (4);
  make_heap (v.begin (), v.end (), greater<int> ());

  v.push_back (7);
  push_heap (v.begin (), v.end (), greater<int> ());

  sort_heap (v.begin (), v.end (), greater<int> ());
  ostream_iterator<int> iter (cout, " ");
  copy (v.begin (), v.end (), iter);
  cout << endl;
  return 0;
}
```

insert2.cpp

```
#include <iostream.h>
#include <stl.h>

char* array1 [] = { "laurie", "jennifer", "leisa" };
char* array2 [] = { "amanda", "saskia", "carrie" };

int main ()
{
  deque<char*> names (array1, array1 + 3);
  deque<char*>::iterator i = names.begin () + 2;
  copy (array2, array2 + 3, inserter (names, i));
  deque<char*>::iterator j;
  for (j = names.begin (); j != names.end (); j++)
    cout << *j << endl;
  return 0;
}
```

uprbnd2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool char_str_less (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* str [] = { "a", "a", "b", "b", "q", "w", "z" };

int main ()
{
  const unsigned strCt = sizeof (str)/sizeof (str[0]);
  cout
    << "d can be inserted at index: "
    << upper_bound (str,  str + strCt, "d", char_str_less) - str
    << endl;
  return 0;
}
```

vec3.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<char> v1; // Empty vector of characters.
  v1.push_back ('h');
  v1.push_back ('i');
  cout << "v1 = " << v1[0] << v1[1] << endl;
  vector<char> v2 (v1);
  v2[1] = 'o'; // Replace second character.
  cout << "v2 = " << v2[0] << v2[1] << endl;
  cout << "(v1 == v2) = " << (v1 == v2) << endl;
  cout << "(v1 < v2) = " << (v1 < v2) << endl;
  return 0;
}
```

iter4.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<int> v; // Empty vector of integers.
  v.push_back (1);
  v.push_back (2);
  v.push_back (3);
  // Position immediately after last item.
  vector<int>::iterator i = v.end ();
  // Move back one and then access.
  cout << "last element is " << *--i << endl;
  i -= 2; // Jump back two items.
  cout << "first element is " << *i << endl;
  return 0;
}
```

setdiff0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[3] = { 13, 18, 23 };
int v2[4] = { 10, 13, 17, 23 };
int result[4] = { 0, 0, 0, 0 };

int main ()
{
  set_difference (v1, v1 + 3, v2, v2 + 4, result);
  for (int i = 0; i < 4; i++)
    cout << result[i] << ' ';
  cout << endl;
  set_difference (v2, v2 + 4, v1, v1 + 2, result);
  for (i = 0; i < 4; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

lexcmp2.cpp

```
#include <stl.h>
#include <iostream.h>

const unsigned size = 6;
char n1[size] = "shoe";
char n2[size] = "shine";
```

```
int main ()
{
  bool before =
    lexicographical_compare (n1, n1 + size,
                             n2, n2 + size,
                             greater<char>());
  if (before)
    cout << n1 << " is after " << n2 << endl;
  else
    cout << n2 << " is after " << n1 << endl;
  return 0;
}
```

adjdiff1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = i * i;
  vector <int> result (v.size ());
  adjacent_difference (v.begin (), v.end (), result.begin ());
  ostream_iterator<int> iter (cout, " ");
  copy (v.begin (), v.end (), iter);
  cout << endl;
  copy (result.begin (), result.end (), iter);
  cout << endl;
  return 0;
}
```

stblptn1.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = rand () % 20;
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  stable_partition (v1.begin (), v1.end (), bind2nd(less<int>(), 11));
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

ptition1.cpp

```
#include <stl.h>
#include <stdlib.h>
#include <iostream.h>
```

```
int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = rand () % 20;
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  partition (v1.begin (), v1.end (), bind2nd(less<int>(), 11));
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

vec1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<int> v1; // Empty vector of integers.
  cout << "empty = " << v1.empty () << endl;
  cout << "size = " << v1.size () << endl;
  cout << "max_size = " << v1.max_size () << endl;
  v1.push_back (42); // Add an integer to the vector.
  cout << "size = " << v1.size () << endl;
  cout << "v1[0] = " << v1[0] << endl;
  return 0;
}
```

sort2.cpp

```
#include <stl.h>
#include <iostream.h>

int array[] = { 1, 50, -10, 11, 42, 19 };

int main ()
{
  int count = sizeof (array) / sizeof (array[0]);
  ostream_iterator <int> iter (cout, " ");
  cout << "before: ";
  copy (array, array + count, iter);
  cout << "\nafter: ";
  sort (array, array + count, greater<int>());
  copy (array, array + count, iter);
  cout << endl;
  return 0;
}
```

copy4.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
```

```
{
  typedef vector<int> IVec;
  vector<int> v1 (10);
  for (int loc = 0; loc < v1.size (); loc++)
    v1[loc] = loc;
  vector<int> v2;
  insert_iterator<IVec> i (v2, v2.begin ());
  copy (v1.begin (), v1.end (), i);
  ostream_iterator<int> outIter (cout, " ");
  copy (v2.begin (), v2.end (), outIter);
  cout << endl;
  return 0;
}
```

prevprm2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (3);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  for (int i = 0; i < 9; i++)
  {
    prev_permutation (v1.begin (), v1.end (), greater<int>());
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
  }
  return 0;
}
```

trnsfrm2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

char map_char (char a_, int b_)
{
  return char(a_ + b_);
}

int trans[] = {-4, 4, -6, -6, -10, 0, 10, -6, 6, 0, -1, -77};
char n[] = "Larry Mullen";

int main ()
{
  const unsigned count = ::strlen (n);
  ostream_iterator <char> iter (cout);
  transform (n, n + count, trans, iter, map_char);
  cout << endl;
  return 0;
}
```

iter1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<const char*> v; // Vector of character strings.
  v.push_back ((char*) "zippy"); // First element.
  v.push_back ((char*) "motorboy"); // Second element.
  vector<const char*>::iterator i = v.begin (); // Position at end.
  for (i = v.begin (); i != v.end (); i++)
    cout << *i << endl; // Display item.
  return 0;
}
```

nextprm2.cpp

maxelem2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* names[] = { "Brett", "Graham", "Jack", "Mike", "Todd" };

int main ()
{
  const unsigned namesCt = sizeof (names)/sizeof (names[0]);
  cout << *max_element (names, names + namesCt, str_compare) << endl;
  return 0;
}
```

minelem2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* names[] = { "Brett", "Graham", "Jack", "Mike", "Todd" };

int main ()
{
  const unsigned namesCt = sizeof (names)/sizeof (names[0]);
  cout << *min_element (names, names + namesCt, str_compare) << endl;
  return 0;
}
```

partsum2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (5);
  iota (v1.begin (), v1.end (), 1);
  vector <int> v2 (v1.size());
  partial_sum (v1.begin (), v1.end (), v2.begin (), times<int>());
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

istmit1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  char buffer [100];
  int i = 0;
  cin.unsetf (ios::skipws); // Disable white-space skipping.
  cout << "Please enter a string: ";
  istream_iterator<char, ptrdiff_t> s (cin);
  while (*s != '\n')
    buffer[i++] = *s++;
  buffer[i] = '\0'; // Null terminate buffer.
  cout << "read " << buffer << endl;
  return 0;
}
```

findif0.cpp

```
#include <stl.h>
#include <iostream.h>

bool odd (int a_)
{
  return a_ % 2;
}

int numbers[6] = { 2, 4, 8, 15, 32, 64 };

int main ()
{
  int* location = find_if (numbers, numbers + 6, odd);
  if (location != numbers + 6)
    cout
      << "Value "
      << *location
      << " at offset "
      << (location - numbers)
      << " is odd"
      << endl;
  return 0;
}
```

pheap1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector<int> v;

  v.push_back (1);
  v.push_back (20);
  v.push_back (4);
  make_heap (v.begin (), v.end ());

  v.push_back (7);
  push_heap (v.begin (), v.end ());

  sort_heap (v.begin (), v.end ());
  ostream_iterator<int> iter (cout, " ");
  copy (v.begin (), v.end (), iter);
  cout << endl;

  return 0;
}
```

stblsrt2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool string_less(const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

char* letters[6] = {"bb", "aa", "ll", "dd", "qq", "cc" };

int main ()
{
  stable_sort (letters, letters + 6, string_less);
  for (int i = 0; i < 6; i++)
    cout << letters[i] << ' ';
  cout << endl;
  return 0;
}
```

nextprm1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (3);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  for (int i = 0; i < 9; i++)
  {
    next_permutation (v1.begin (), v1.end ());
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
  }
  return 0;
}
```

prevprm1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (3);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator<int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  for (int i = 0; i < 9; i++)
  {
    prev_permutation (v1.begin (), v1.end ());
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
  }
  return 0;
}
```

rndshuf1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1(10);
  iota (v1.begin (), v1.end (), 0);
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  for (int i = 0; i < 3; i++)
  {
    random_shuffle (v1.begin (), v1.end ());
    copy (v1.begin (), v1.end (), iter);
    cout << endl;
  }
  return 0;
}
```

ptrbinf1.cpp

```
#include <iostream.h>
#include <stl.h>
```

```
int sum (int x_, int y_)
{
  return x_ + y_;
}

int input1 [4] = { 7, 2, 3, 5 };
int input2 [4] = { 1, 5, 5, 8 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output,
    pointer_to_binary_function<int, int, int> (sum));
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

iter2.cpp

```
#include <iostream.h>
#include <stl.h>

void print (const vector<const char*>& v_)
{
  vector<const char*>::const_iterator i;
  for (i = v_.begin (); i != v_.end (); i++)
    cout << *i << endl;
}

int main ()
{
  vector<const char*> v; // Vector of character strings.
  v.push_back ((char*) "zippy");
  v.push_back ((char*) "motorboy");
  print (v);
  return 0;
}
```

partsum1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  iota (v1.begin (), v1.end (), 0);
  vector <int> v2 (v1.size());
  partial_sum (v1.begin (), v1.end (), v2.begin ());
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

replif1.cpp

```
#include <stl.h>
#include <iostream.h>

bool odd (int a_)
{
  return a_ % 2;
}

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = i % 5;
    cout << v1[i] << ' ';
  }
  cout << endl;
  replace_if (v1.begin (), v1.end (), odd, 42);
  for (i = 0; i < v1.size (); i++)
    cout << v1[i] << ' ';
  cout << endl;
  return 0;
}
```

mset4.cpp

```
#include <iostream.h>
#include <stl.h>

int array [] = { 3, 6, 1, 2, 3, 2, 6, 7, 9 };

int main ()
{
  typedef multiset<int, less<int> > mset;
  mset s (array, array + 9);
  pair<mset::const_iterator, mset::const_iterator> p = s.equal_range (3);
  cout << "lower bound = " << *(p.first) << endl;
  cout << "upper bound = " << *(p.second) << endl;
  return 0;
}
```

iter3.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<const char*> v; // Vector of character strings.
  v.push_back ((char*) "zippy"); // First element.
  v.push_back ((char*) "motorboy"); // Second element.
  vector<const char*>::reverse_iterator i;
  for (i = v.rbegin (); i != v.rend (); i++)
    cout << *i << endl; // Display item.
  return 0;
}
```

list2.cpp

```
#include <iostream.h>
#include <stl.h>

int array1 [] = { 1, 16 };
int array2 [] = { 4, 9 };

int main ()
{
  list<int> l1 (array1, array1 + 2);
  list<int> l2 (array2, array2 + 2);
  list<int>::iterator i = l1.begin ();
  i++;
  l1.splice (i, l2, l2.begin (), l2.end ());
  i = l1.begin ();
  while (i != l1.end ())
    cout << *i++ << endl;
  return 0;
}
```

set1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  set<int, less<int> > s;
  cout << "count (42) = " << s.count (42) << endl;
  s.insert (42);
  cout << "count (42) = " << s.count (42) << endl;
  s.insert (42);
  cout << "count (42) = " << s.count (42) << endl;
  int count = s.erase (42);
  cout << count << " elements erased" << endl;
  return 0;
}
```

list1.cpp

```
#include <iostream.h>
#include <stl.h>

int array1 [] = { 9, 16, 36 };
int array2 [] = { 1, 4 };

int main ()
{
  list<int> l1 (array1, array1 + 3);
  list<int> l2 (array2, array2 + 2);
  list<int>::iterator i1 = l1.begin ();
  l1.splice (i1, l2);
  list<int>::iterator i2 = l1.begin ();
  while (i2 != l1.end ())
    cout << *i2++ << endl;
  return 0;
}
```

alg5.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  list<int> years;
  years.push_back (1962);
  years.push_back (1992);
  years.push_back (2001);
  years.push_back (1999);
  sort (years.begin (), years.end ()); // Causes linker error.
  list<int>::iterator i;
  for (i = years.begin (); i != years.end (); i++)
    cout << *i << endl;
  return 0;
}
```

eqlrnge0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[10] = { 0, 0, 1, 1, 2, 2, 2, 2, 3, 3 };

int main ()
{
  pair <int*, int*> range;
  range = equal_range (numbers, numbers + 10, 2);
  cout
    << "2 can be inserted from before index "
    << (range.first - numbers)
    << " to before index "
    << (range.second - numbers)
    << endl;
  return 0;
}
```

advance.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  typedef vector <int> IntVector;
  IntVector v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = i;
  IntVector::iterator location = v.begin ();
  cout << "At Beginning: " << *location << endl;
  advance (location, 5);
  cout << "At Beginning + 5: " << *location << endl;
  return 0;
}
```

replace1.cpp

```
#include <stl.h>
```

```
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = i % 5;
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  replace (v1.begin (), v1.end (), 2, 42);
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

alg3.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<int> i;
  i.push_back (1);
  i.push_back (4);
  i.push_back (2);
  i.push_back (8);
  i.push_back (2);
  i.push_back (2);
  int n = 0; // Must be initialized, as count increments n.
  count (i.begin (), i.end (), 2, n);
  cout << "Count of 2s = " << n << endl;
  return 0;
}
```

func2.cpp

```
#include <iostream.h>
#include <stl.h>

bool bigger_than (int x_, int y_)
{
  return x_ > y_;
}

int main ()
{
  vector<int>v;
  v.push_back (4);
  v.push_back (1);
  v.push_back (5);
  sort (v.begin (), v.end (), bigger_than);
  vector<int>::iterator i;
  for (i = v.begin (); i != v.end (); i++)
    cout << *i << endl;
  return 0;
}
```

unegate1.cpp

```
#include <iostream.h>
#include <stl.h>

struct odd : public unary_function<int, bool>
{
  odd () {}
  bool operator () (int n_) const { return (n_ % 2) == 1; }
};

int array [3] = { 1, 2, 3 };

int main ()
{
  int* p = find_if (array, array + 3, unary_negate<odd> (odd ()));
  if (p != array + 3)
    cout << *p << endl;
  return 0;
}
```

alg4.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<int> years;
  years.push_back (1962);
  years.push_back (1992);
  years.push_back (2001);
  years.push_back (1999);
  sort (years.begin (), years.end ());
  vector<int>::iterator i;
  for (i = years.begin (); i != years.end (); i++)
    cout << *i << endl;
  return 0;
}
```

countif1.cpp

```
#include <stl.h>
#include <iostream.h>

int odd (int a_)
{
  return a_ % 2;
}

int main ()
{
  vector <int> numbers(100);
  for (int i = 0; i < 100; i++)
    numbers[i] = i % 3;
  int elements = 0;
  count_if (numbers.begin (), numbers.end (), odd, elements);
  cout << "Found " << elements << " odd elements." << endl;
  return 0;
}
```

lwrbnd1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (20);
  for (int i = 0; i < v1.size (); i++)
  {
    v1[i] = i/4;
    cout << v1[i] << ' ';
  }
  int* location =  lower_bound (v1.begin (), v1.end (), 3);
  cout
    << "\n3 can be inserted at index: "
    << (location - v1.begin ())
    << endl;
  return 0;
}
```

lexcmp1.cpp

```
#include <stl.h>
#include <iostream.h>

const unsigned size = 6;
char n1[size] = "shoe";
char n2[size] = "shine";

int main ()
{
  bool before = lexicographical_compare (n1, n1 + size, n2, n2 + size);
  if (before)
    cout << n1 << " is before " << n2 << endl;
  else
    cout << n2 << " is before " << n1 << endl;
  return 0;
}
```

copyb.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = i;
  vector <int> v2(v1.size ());
  copy_backward (v1.begin (), v1.end (), v2.end ());
  ostream_iterator<int> iter (cout, " ");
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

ptrbinf2.cpp

```
#include <iostream.h>
#include <stl.h>

int sum (int x_, int y_)
{
  return x_ + y_;
}

int input1 [4] = { 7, 2, 3, 5 };
int input2 [4] = { 1, 5, 5, 8 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, ptr_fun (sum));
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

copyb0.cpp

```
#include <stl.h>
#include <iostream.h>

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

int main ()
{
  int result[5];
  copy_backward (numbers, numbers + 5, result + 5);
  for (int i = 0; i < 5; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  for (i = 0; i < 5; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

binsert1.cpp

```
#include <iostream.h>
#include <stl.h>

char* array [] = { "laurie", "jennifer", "leisa" };

int main ()
{
  vector<char*> names;
  copy (array, array + 3, back_insert_iterator<vector <char*> > (names));
  vector<char*>::iterator i;
  for (i = names.begin (); i != names.end (); i++)
    cout << *i << endl;
  return 0;
}
```

unegate2.cpp

```
#include <iostream.h>
#include <stl.h>

struct odd : public unary_function<int, bool>
{
  odd () {}
  bool operator () (int n_) const { return (n_ % 2) == 1; }
};

int array [3] = { 1, 2, 3 };

int main ()
{
  int* p = find_if (array, array + 3, not1 (odd ()));
  if (p != array + 3)
    cout << *p << endl;
  return 0;
}
```

revcopy1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 3, 4, 5 };

int main ()
{
  int result[6];
  reverse_copy (numbers, numbers + 6, result);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  for (i = 0; i < 6; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

finsert1.cpp

```
#include <iostream.h>
#include <stl.h>

char* array [] = { "laurie", "jennifer", "leisa" };

int main ()
{
  deque<char*> names;
  copy (array, array + 3, front_insert_iterator<deque <char*> > (names));
  deque<char*>::iterator i;
  for (i = names.begin (); i != names.end (); i++)
    cout << *i << endl;
  return 0;
}
```

remcpif1.cpp

```
#include <stl.h>
#include <iostream.h>

bool odd (int a_)
{
  return a_ % 2;
}

int numbers[6] = { 1, 2, 3, 1, 2, 3 };
int result[6] = { 0, 0, 0, 0, 0, 0 };

int main ()
{
  remove_copy_if (numbers, numbers + 6, result, odd);
  for (int i = 0; i < 6; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

inplmrg1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 1, 10, 42, 3, 16, 32 };

int main ()
{
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  inplace_merge (numbers, numbers + 3, numbers + 6);
  for (i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

list4.cpp

```
#include <iostream.h>
#include <stl.h>

int array1 [] = { 1, 3, 6, 7 };
int array2 [] = { 2, 4 };

int main ()
{
  list<int> l1 (array1, array1 + 4);
  list<int> l2 (array2, array2 + 2);
  l1.merge (l2);
  for (list<int>::iterator i = l1.begin (); i != l1.end (); i++)
    cout << *i;
  cout << endl;
  return 0;
}
```

revbit1.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  list<int> v (array, array + 4);
  reverse_bidirectional_iterator<list<int>::iterator, int,
    list<int>::reference, list<int>::difference_type> r (v.end ());
  while (r != v.begin ())
    cout << *r++ << endl;
  return 0;
}
```

copy3.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = i;
  vector <int> v2 (10);
  copy (v1.begin (), v1.end (), v2.begin ());
  ostream_iterator<int> iter (cout, " ");
  copy (v2.begin (), v2.end (), iter);
  cout << endl;
  return 0;
}
```

merge0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers1[5] = { 1, 6, 13, 25, 101 };
int numbers2[5] = {-5, 26, 36, 46, 99 };

int main ()
{
  int result[10];
  merge (numbers1, numbers1 + 5, numbers2, numbers2 + 5, result);
  for (int i = 0; i < 10; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

reviter1.cpp

```
#include <iostream.h>
#include <stl.h>

int array [] = { 1, 5, 2, 3 };
int main ()
{
  vector<int> v (array, array + 4);
  stl_reverse_iterator<vector<int>::iterator, int,
    vector<int>::reference, vector<int>::difference_type> r (v.end ());
  while (r != v.begin ())
    cout << *r++ << endl;
  return 0;
}
```

find1.cpp

```
#include <stl.h>
#include <iostream.h>

int years[] = { 1942, 1952, 1962, 1972, 1982, 1992 };

int main ()
{
  const unsigned yearCount = sizeof (years) / sizeof (years[0]);
  int* location = find (years, years + yearCount, 1972);
  cout << "Found 1972 at offset " << (location - years) << endl;
  return 0;
}
```

trnsfrm1.cpp

```
#include <stl.h>
#include <iostream.h>

int negate_int (int a_)
{
  return -a_;
}

int numbers[6] = { -5, -1, 0, 1, 6, 11 };

int main ()
{
  int result[6];
  transform (numbers, numbers + 6, result, negate_int);
  for (int i = 0; i < 6; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

binsert2.cpp

```
#include <iostream.h>
#include <stl.h>

char* array [] = { "laurie", "jennifer", "leisa" };

int main ()
{
  vector<char*> names;
  copy (array, array + 3, back_inserter (names));
  vector<char*>::iterator i;
  for (i = names.begin (); i != names.end (); i++)
    cout << *i << endl;
  return 0;
}
```

setsymd0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[3] = { 13, 18, 23 };
int v2[4] = { 10, 13, 17, 23 };
int result[4] = { 0, 0, 0, 0 };

int main ()
{
  set_symmetric_difference (v1, v1 + 3, v2, v2 + 4, result);
  for (int i = 0; i < 4; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

finsert2.cpp

```
#include <iostream.h>
#include <stl.h>

char* array [] = { "laurie", "jennifer", "leisa" };

int main ()
{
  deque<char*> names;
  copy (array, array + 3, front_inserter (names));
  deque<char*>::iterator i;
  for (i = names.begin (); i != names.end (); i++)
    cout << *i << endl;
  return 0;
}
```

mset2.cpp

```
#include <iostream.h>
#include <stl.h>

char* names [] = { "dave", "alf", "chas", "bob", "ed", "chas" };

int main ()
{
  typedef multiset<char*, less_s> mset;
  mset s;
  s.insert (names, names + 6);
  for (mset::iterator i = s.begin (); i != s.end (); i++)
    cout << *i << endl;
  return 0;
}
```

ostmit.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  char* string = "hello";
  ostream_iterator<char> it1 (cout);
  copy (string, string + 5, it1);
  cout << endl;
  ostream_iterator<int> it2 (cout);
  copy (array, array + 4, it2);
  cout << endl;
  return 0;
}
```

ptrunf1.cpp

```
#include <iostream.h>
#include <stl.h>

bool even (int n_)
{
  return (n_ % 2) == 0;
}

int array [3] = { 1, 2, 3 };

int main ()
{
  int* p = find_if (array, array + 3,
    pointer_to_unary_function<int, bool> (even));
  if (p != array + 3)
    cout << *p << " is even" << endl;
  return 0;
}
```

func1.cpp

```
#include <iostream.h>
#include <stl.h>

bool bigger (int i_)
{
  return i_ > 3;
}

int main ()
{
  vector<int>v;
  v.push_back (4);
  v.push_back (1);
  v.push_back (5);
  int n = 0;
  count_if (v.begin (), v.end (), bigger, n);
  cout << "Number greater than 3 = " << n << endl;
  return 0;
}
```

stblptn0.cpp

```
#include <stl.h>
#include <iostream.h>

bool less_10 (int a_)
{
  return a_ < 10 ? 1 : 0;
}

int numbers[6] = { 10, 5, 11, 20, 6, -2 };

int main ()
{
  stable_partition (numbers, numbers + 6, less_10);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

setunon0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[3] = { 13, 18, 23 };
int v2[4] = { 10, 13, 17, 23 };
int result[7] = { 0, 0, 0, 0, 0, 0, 0 };

int main ()
{
  set_union (v1, v1 + 3, v2, v2 + 4, result);
  for (int i = 0; i < 7; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

mkheap1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 5, 10, 4, 13, 11, 19 };

int main ()
{
  make_heap (numbers, numbers + 6, greater<int> ());
  for (int i = 6; i >= 1; i--)
  {
    cout << numbers[0] << endl;
    pop_heap (numbers, numbers + i, greater<int> ());
  }
  return 0;
}
```

setintr0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[3] = { 13, 18, 23 };
int v2[4] = { 10, 13, 17, 23 };
int result[4] = { 0, 0, 0, 0 };

int main ()
{
  set_intersection (v1, v1 + 3, v2, v2 + 4, result);
  for (int i = 0; i < 4; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

logicand.cpp

```
#include <iostream.h>
#include <stl.h>

bool input1 [4] = { 1, 1, 0, 1 };
bool input2 [4] = { 0, 1, 0, 0 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, logical_and<bool> ());
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

logicor.cpp

```
#include <iostream.h>
#include <stl.h>

bool input1 [4] = { 1, 1, 0, 1 };
bool input2 [4] = { 0, 1, 0, 0 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, logical_or<bool> ());
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

nequal.cpp

```
#include <iostream.h>
#include <stl.h>

int input1 [4] = { 1, 7, 2, 2 };
int input2 [4] = { 1, 6, 2, 3 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, not_equal_to<int> ());
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

ptition0.cpp

```
#include <stl.h>
#include <iostream.h>

int less_10 (int a_)
{
  return a_ < 10 ? 1 : 0;
}

int numbers[6] = { 6, 12, 3, 10, 1, 20 };

int main ()
{
  partition (numbers, numbers + 6, less_10);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

inrprod0.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

int vector1[5] = { 1, 2, 3, 4, 5 };
int vector2[5] = { 1, 2, 3, 4, 5 };

int main ()
{
  int result;
  result = inner_product (vector1, vector1 + 5, vector2, 0);
  cout << "Inner product = " << result << endl;
  return 0;
}
```

func3.cpp

```
#include <iostream.h>
#include <stl.h>
```

```
int main ()
{
  vector<int>v;
  v.push_back (4);
  v.push_back (1);
  v.push_back (5);
  sort (v.begin (), v.end (), greater<int> ());
  vector<int>::iterator i;
  for (i = v.begin (); i != v.end (); i++)
    cout << *i << endl;
  return 0;
}
```

modulus.cpp

```
#include <iostream.h>
#include <stl.h>

int input1 [4] = { 6, 8, 10, 2 };
int input2 [4] = { 4, 2, 11, 3 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, modulus<int> ());
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

uprbnd1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  int array[20];
  for (int i = 0; i < 20; i++)
  {
    array[i] = i/4;
    cout << array[i] << ' ';
  }
  cout
    << "\n3 can be inserted at index: "
    << upper_bound (array, array + 20, 3) - array
    << endl;
  return 0;
}
```

equalto.cpp

```
#include <iostream.h>
#include <stl.h>

int input1 [4] = { 1, 7, 2, 2 };
int input2 [4] = { 1, 6, 2, 3 };
int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, equal_to<int> ());
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

count1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> numbers(100);
  for (int i = 0; i < 100; i++)
    numbers[i] = i % 3;
  int elements = 0;
  count (numbers.begin (), numbers.end (), 2, elements);
  cout << "Found " << elements << " 2's." << endl;
  return 0;
}
```

uniqcpy1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[8] = { 0, 1, 1, 2, 2, 2, 3, 4 };
int result[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };

int main ()
{
  unique_copy (numbers, numbers + 8, result);
  for (int i = 0; i < 8; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

minus.cpp

```
#include <iostream.h>
#include <stl.h>

int input1 [4] = { 1, 5, 7, 8 };
int input2 [4] = { 1, 4, 8, 3 };

int main ()
{
  int output [4];
  transform (input1, input1 + 4, input2, output, minus<int> ());
  for (int i = 0; i < 4; i++)
    cout << output[i] << endl;
  return 0;
}
```

replcpy1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 0, 1, 2 };
int result[6] = { 0, 0, 0, 0, 0, 0 };

int main ()
{
  replace_copy (numbers, numbers + 6, result, 2, 42);
  for (int i = 0; i < 6; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

swprnge1.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

int main ()
{
  char* word1 = "World";
  char* word2 = "Hello";
  cout << word1 << " " << word2 << endl;
  swap_ranges (word1, word1 + ::strlen (word1), word2);
  cout << word1 << " " << word2 << endl;
  return 0;
}
```

vec8.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  vector<int> v;
  cout << "capacity = " << v.capacity () << endl;
  v.push_back (42);
  cout << "capacity = " << v.capacity () << endl;
  v.reserve (5000);
  cout << "capacity = " << v.capacity () << endl;
  return 0;
}
```

plus.cpp

```
#include <iostream.h>
#include <stl.h>
```

```
int input1 [4] = { 1, 6, 11, 8 };
int input2 [4] = { 1, 5, 2, 3 };

int main ()
{
  int total =
    inner_product (input1, input1 + 4, input2, 0, plus<int> (), times <int> ());
  cout << "total = " << total << endl;
  return 0;
}
```

remcopy1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 1, 2, 3, 1, 2, 3 };
int result[6] = { 0, 0, 0, 0, 0, 0 };

int main ()
{
  remove_copy (numbers, numbers + 6, result, 2);
  for (int i = 0; i < 6; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

error2.cpp

```
#include <stl.h>

// Compile this code with the symbol OS_USE_EXCEPTIONS defined.

int main ()
{
  vector<int> v;
  try
  {
    v.pop_back (); // Generates an exception.
  }
  catch (const char* str)
  {
    cout << "Caught exception " << str << endl;
  }
  return 0;
}
```

iterswp1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v1 (6);
  iota (v1.begin (), v1.end (), 0);
  iter_swap (v1.begin (), v1.begin () + 3);
  ostream_iterator <int> iter (cout, " ");
  copy (v1.begin (), v1.end (), iter);
  cout << endl;
  return 0;
}
```

remif1.cpp

```
#include <stl.h>
#include <iostream.h>

bool odd (int a_)
{
  return a_ % 2;
}

int numbers[6] = { 0, 0, 1, 1, 2, 2 };

int main ()
{
  remove_if (numbers, numbers + 6, odd);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

foreach1.cpp

```
#include <stl.h>
#include <iostream.h>

void print_sqr (int a_)
{
  cout << a_ * a_ << " ";
}

int main ()
{
  vector <int> v1 (10);
  for (int i = 0; i < v1.size (); i++)
    v1[i] = i;
  for_each (v1.begin (), v1.end (), print_sqr);
  cout << endl;
  return 0;
}
```

parsrtc0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 5, 2, 4, 3, 1, 6 };

int main ()
{
  int result[3];
  partial_sort_copy (numbers, numbers + 6, result, result + 3);
  for (int i = 0; i < 3; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

pqueue2.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  priority_queue<deque<char*>, greater_s> q;
  q.push ((char*) "cat");
  q.push ((char*) "dog");
  q.push ((char*) "ape");
  while (!q.empty ())
  {
    cout << q.top () << endl;
    q.pop ();
  }
  return 0;
}
```

binsrch1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  int vector[100];
  for (int i = 0; i < 100; i++)
    vector[i] = i;
  if (binary_search (vector, vector + 100, 42))
    cout << "found 42" << endl;
  else
    cout << "did not find 42" << endl;
  return 0;
}
```

ptrunf2.cpp

```
#include <iostream.h>
#include <stl.h>

bool even (int n_)
{
  return (n_ % 2) == 0;
}

int array [3] = { 1, 2, 3 };

int main ()
{
  int* p = find_if (array, array + 3, ptr_fun (even));
  if (p != array + 3)
```

```
    cout << *p << " is even" << endl;
  return 0;
}
```

rotcopy0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 3, 4, 5 };

int main ()
{
  int result[6];
  rotate_copy (numbers, numbers + 3, numbers + 6, result);
  for (int i = 0; i < 6; i++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

mkheap0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 5, 10, 4, 13, 11, 19 };

int main ()
{
  make_heap (numbers, numbers + 6);
  for (int i = 6; i >= 1; i--)
  {
    cout << numbers[0] << endl;
    pop_heap (numbers, numbers + i);
  }
  return 0;
}
```

copy1.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

char string[23] = "A string to be copied.";

int main ()
{
  char result[23];
  copy (string, string + 23, result);
  cout << " Src: " << string << "\nDest: " << result << endl;
  return 0;
}
```

find0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[10] = { 0, 1, 4, 9, 16, 25, 36, 49, 64 };

int main ()
{
  int* location;
  location = find (numbers, numbers + 10, 25);
  cout << "Found 25 at offset " << (location - numbers) << endl;
  return 0;
}
```

partsum0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 1, 2, 3, 4, 5, 6 };

int main ()
{
  int result[6];
  partial_sum (numbers, numbers + 6, result);
  for (int i = 0; i < 6; i ++)
    cout << result[i] << ' ';
  cout << endl;
  return 0;
}
```

bvec1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  bit_vector b (3);
  for (int i = 0; i < b.size (); i++)
    cout << b[i];
  cout << endl;
  b[0] = b[2] = 1;
  for (i = 0; i < b.size (); i++)
    cout << b[i];
  cout << endl;
  return 0;
}
```

bind2nd1.cpp

```
#include <iostream.h>
#include <stl.h>

int array [3] = { 1, 2, 3 };

int main ()
{
  replace_if (array, array + 3,
    binder2nd<greater<int> > (greater<int> (), 2), 4);
  for (int i = 0; i < 3; i++)
    cout << array[i] << endl;
  return 0;
}
```

bind1st1.cpp

```
#include <iostream.h>
#include <stl.h>

int array [3] = { 1, 2, 3 };

int main ()
{
  int* p = remove_if (array, array + 3,
    binder1st<less<int> > (less<int> (), 2));
  for (int* i = array; i != p; i++)
    cout << *i << endl;
  return 0;
}
```

reviter2.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  vector<int> v (array, array + 4);
  vector<int>::reverse_iterator r;
  for (r = v.rbegin (); r != v.rend (); r++)
    cout << *r << endl;
  return 0;
}
```

copy2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v (10);
  for (int i = 0; i < v.size (); i++)
    v[i] = i;
  ostream_iterator<int> iter (cout, " ");
  copy (v.begin (), v.end (), iter);
  cout << endl;
  return 0;
}
```

max2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

int main ()
{
  cout << max ("shoe", "shine", str_compare) << endl;
  return 0;
}
```

min2.cpp

```
#include <stl.h>
#include <iostream.h>
#include <string.h>

bool str_compare (const char* a_, const char* b_)
{
  return ::strcmp (a_, b_) < 0 ? 1 : 0;
}

int main ()
{
  cout << min ("shoe", "shine", str_compare) << endl;
  return 0;
}
```

parsrt0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 5, 2, 4, 3, 1, 6 };

int main ()
{
  partial_sort (numbers, numbers + 3, numbers + 6);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

partsrt0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 5, 2, 4, 3, 1, 6 };

int main ()
{
  partial_sort (numbers, numbers + 3, numbers + 6);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

bnegate1.cpp

```
#include <iostream.h>
#include <stl.h>

int array [4] = { 4, 9, 7, 1 };

int main ()
{
  sort (array, array + 4, binary_negate<greater<int> > (greater<int> ()));
  for (int i = 0; i < 4; i++)
    cout << array[i] << endl;
  return 0;
}
```

nthelem0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 5, 2, 4, 1, 0, 3 };

int main ()
{
  nth_element (numbers, numbers + 3, numbers + 6);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

revbit2.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  list<int> v (array, array + 4);
  list<int>::reverse_iterator r;
  for (r = v.rbegin (); r != v.rend (); r++)
    cout << *r << endl;
  return 0;
}
```

count0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[10] = { 1, 2, 4, 1, 2, 4, 1, 2, 4, 1 };

int main ()
{
  int result = 0;
  count (numbers, numbers + 10, 1, result);
  cout << "Found " << result << " 1's." << endl;
  return 0;
}
```

negate.cpp

```
#include <iostream.h>
#include <stl.h>

int input [3] = { 1, 2, 3 };

int main ()
{
  int output[3];
  transform (input, input + 3, output, negate<int> ());
  for (int i = 0; i < 3; i++)
    cout << output[i] << endl;
  return 0;
}
```

pqueue1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  priority_queue<deque<int>, less<int> > q;
  q.push (42);
  q.push (101);
  q.push (69);
  while (!q.empty ())
  {
    cout << q.top () << endl;
    q.pop ();
  }
  return 0;
}
```

genern1.cpp

```
#include <stl.h>
#include <iostream.h>
#include <stdlib.h>

int main ()
{
  vector <int> v1 (10);
  generate_n (v1.begin (), v1.size (), rand);
  for (int i = 0; i < 10; i++)
    cout << v1[i] << ' ';
  cout << endl;
  return 0;
}
```

rotate0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 3, 4, 5 };

int main ()
{
  rotate (numbers, numbers + 3, numbers + 6);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

foreach0.cpp

```
#include <stl.h>
#include <iostream.h>

void print (int a_)
{
  cout << a_ << ' ';
}

int numbers[10] = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 };

int main ()
{
  for_each (numbers, numbers + 10, print);
  cout << endl;
  return 0;
}
```

alg2.cpp

```
#include <iostream.h>
#include <stl.h>

int i [] = { 1, 4, 2, 8, 2, 2 };

int main ()
{
  int n = 0; // Must be initialized, as count increments n.
  count (i, i + 6, 2, n);
  cout << "Count of 2s = " << n << endl;
  return 0;
}
```

gener1.cpp

```
#include <stl.h>
#include <iostream.h>
#include <stdlib.h>

int main ()
{
  int numbers[10];
  generate (numbers, numbers + 10, rand);
  for (int i = 0; i < 10; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

replace0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 0, 1, 2 };

int main ()
{
  replace (numbers, numbers + 6, 2, 42);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

rndshuf0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 1, 2, 3, 4, 5, 6 };

int main ()
{
  random_shuffle (numbers, numbers + 6);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

bind1st2.cpp

```
#include <iostream.h>
#include <stl.h>

int array [3] = { 1, 2, 3 };

int main ()
{
  int* p = remove_if (array, array + 3, bind1st(less<int> (), 2));
  for (int* i = array; i != p; i++)
```

```
    cout << *i << endl;
  return 0;
}
```

unique1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[8] = { 0, 1, 1, 2, 2, 2, 3, 4 };

int main ()
{
  unique (numbers, numbers + 8);
  for (int i = 0; i < 8; i ++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

bind2nd2.cpp

```
#include <iostream.h>
#include <stl.h>

int array [3] = { 1, 2, 3 };

int main ()
{
  replace_if (array, array + 3, bind2nd (greater<int> (), 2), 4);
  for (int i = 0; i < 3; i++)
    cout << array[i] << endl;
  return 0;
}
```

vec5.cpp

```
#include <iostream.h>
#include <stl.h>

int array [] = { 1, 4, 9, 16 };

int main ()
{
  vector<int> v (array, array + 4);
  for (int i = 0; i < v.size (); i++)
    cout << "v[" << i << "] = " << v[i] << endl;
  return 0;
}
```

iterswp0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 3, 4, 5 };
int main ()
{
  iter_swap (numbers, numbers + 3);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

remove1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 1, 2, 3, 1, 2, 3 };

int main ()
{
  remove (numbers, numbers + 6, 1);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

stblsrt1.cpp

```
#include <stl.h>
#include <iostream.h>

int array[6] = { 1, 50, -10, 11, 42, 19 };

int main ()
{
  stable_sort (array, array + 6);
  for (int i = 0; i < 6; i++)
    cout << array[i] << ' ';
  cout << endl;
  return 0;
}
```

reverse1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 0, 1, 2, 3, 4, 5 };

int main ()
{
  reverse (numbers, numbers + 6);
  for (int i = 0; i < 6; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

logicnot.cpp

```
#include <iostream.h>
#include <stl.h>

bool input [7] = { 1, 0, 0, 1, 1, 1, 1 };

int main ()
{
  int n = 0;
  count_if (input, input + 7, logical_not<bool> (), n);
  cout << "count = " << n << endl;
  return 0;
}
```

sort1.cpp

bnegate2.cpp

```
#include <iostream.h>
#include <stl.h>

int array [4] = { 4, 9, 7, 1 };

int main ()
{
  sort (array, array + 4, not2 (greater<int> ()));
  for (int i = 0; i < 4; i++)
    cout << array[i] << endl;
  return 0;
}
```

queue1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  queue<list<int> > q;
  q.push (42);
  q.push (101);
  q.push (69);
  while (!q.empty ())
  {
    cout << q.front () << endl;
    q.pop ();
  }
  return 0;
}
```

stack1.cpp

```
#include <iostream.h>
#include <stl.h>
```

```
int main ()
{
  stack<deque<int> > s;
  s.push (42);
  s.push (101);
  s.push (69);
  while (!s.empty ())
  {
    cout << s.top () << endl;
    s.pop ();
  }
  return 0;
}
```

greateq.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  sort (array, array + 4, greater_equal<int> ());
  for (int i = 0; i < 4; i++)
    cout << array[i] << endl;
  return 0;
}
```

stack2.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  stack<list<int> > s;
  s.push (42);
  s.push (101);
  s.push (69);
  while (!s.empty ())
  {
    cout << s.top () << endl;
    s.pop ();
  }
  return 0;
}
```

lesseq.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  sort (array, array + 4, less_equal<int> ());
  for (int i = 0; i < 4; i++)
    cout << array[i] << endl;
  return 0;
}
```

divides.cpp

```
#include <iostream.h>
#include <stl.h>

int input [3] = { 2, 3, 4 };

int main ()
{
  int result = accumulate (input, input + 3, 48, divides<int> ());
  cout << "result = " << result << endl;
  return 0;
}
```

greater.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  sort (array, array + 4, greater<int> ());
  for (int i = 0; i < 4; i++)
    cout << array[i] << endl;
  return 0;
}
```

swap1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  int a = 42;
  int b = 19;
  cout << "a = " << a << " b = " << b << endl;
  swap (a, b);
  cout << "a = " << a << " b = " << b << endl;
  return 0;
}
```

times.cpp

```
#include <iostream.h>
#include <stl.h>

int input [4] = { 1, 5, 7, 2 };
int main ()
{
  int total = accumulate (input, input + 4, 1, times<int> ());
  cout << "total = " << total << endl;
  return 0;
}
```

less.cpp

```
#include <iostream.h>
#include <stl.h>

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

int main ()
{
  sort (array, array + 4, less<int> ());
  for (int i = 0; i < 4; i++)
    cout << array[i] << endl;
  return 0;
}
```

alg1.cpp

```
#include <iostream.h>
#include <stl.h>

int main ()
{
  int i = min (4, 7);
  cout << "min (4, 7) = " << i << endl;
  char c = max ('a', 'z');
  cout << "max ('a', 'z') = " << c << endl;
  return 0;
}
```

filln1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v (10);
  fill_n (v.begin (), v.size (), 42);
  for (int i = 0; i < 10; i++)
    cout << v[i] << ' ';
  cout << endl;
  return 0;
}
```

iota1.cpp

```
#include <stl.h>
#include <iostream.h>
```

```
int main ()
{
  int numbers[10];
  iota (numbers, numbers + 10, 42);
  for (int i = 0; i < 10; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  return 0;
}
```

nextprm0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[3] = { 0, 1, 2 };

int main ()
{
  next_permutation (v1, v1 + 3);
  for (int i = 0; i < 3; i++)
    cout << v1[i] << ' ';
  cout << endl;
  return 0;
}
```

prevprm0.cpp

```
#include <stl.h>
#include <iostream.h>

int v1[3] = { 0, 1, 2 };

int main ()
{
  prev_permutation (v1, v1 + 3);
  for (int i = 0; i < 3; i++)
    cout << v1[i] << ' ';
  cout << endl;
  return 0;
}
```

fill1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  vector <int> v (10);
  fill (v.begin (), v.end (), 42);
  for (int i = 0; i < 10; i++)
    cout << v[i] << ' ';
  cout << endl;
  return 0;
}
```

pair2.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  pair<int, int> p = make_pair (1, 10);
  cout << "p.first = " << p.first << endl;
  cout << "p.second = " << p.second << endl;
  return 0;
}
```

error1.cpp

```
#include <stl.h>

// Compile this code without defining the symbol OS_USE_EXCEPTIONS.

int main ()
{
  vector<int> v;
  v.pop_back (); // Generates an empty object error.
  return 0;
}
```

pair0.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  pair<int, int> p = make_pair (1, 10);
  cout << "p.first = " << p.first << ", p.second = " << p.second << endl;
  return 0;
}
```

pair1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  pair<int, int> p = make_pair (1, 10);
  cout << "p.first = " << p.first << ", p.second = " << p.second << endl;
  return 0;
}
```

minelem1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { -10, 15, -100, 36, -242, 42 };
int main ()
{
  cout << *min_element (numbers, numbers + 6) << endl;
  return 0;
}
```

maxelem1.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[6] = { 4, 10, 56, 11, -42, 19 };

int main ()
{
  cout << *max_element (numbers, numbers + 6) << endl;
  return 0;
}
```

max1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  cout << max (42, 100) << endl;
  return 0;
}
```

min1.cpp

```
#include <stl.h>
#include <iostream.h>

int main ()
{
  cout << min (42, 100) << endl;
  return 0;
}
```

adjdiff0.cpp

```
#include <stl.h>
#include <iostream.h>

int numbers[5] = { 1, 2, 4, 8, 16 };

int main ()
{
  int difference[5];
  adjacent_difference (numbers, numbers + 5, difference);
  for (int i = 0; i < 5; i++)
    cout << numbers[i] << ' ';
  cout << endl;
  for (i = 0; i < 5; i++)
```

```
    cout << difference[i] << ' ';
  cout << endl;
  return 0;
}
```

К сожалению, все наши попытки найти авторов этого перевода были тщетны. Тем не менее, мы сочли необходимым опубликовать этот материал ввиду скудности информации по данной тематике на русском языке.