STL标准模板库是C++标准库的核心。

STL的基本观念是将数据和操作分离。数据有容器类加以管理，操作则由可定制的算法来实现。迭代器在两种之间充当黏合剂，是任何算法都可以和任何容器交互运作。

STL标准组件的三个核心：容器（Container），迭代器（Iterator），算法（Algorithm）。

容器（Container）：

1. 序列式容器：有序结合，每个元素均有确凿的位置--取决于插入时机和地点。STL中有5个序列式容器：array、vector、deque、list和forward\_list。通常被实现为array或linked list。
2. 关联式容器：已排序的结合。STL提供了4个关联式容器：set、multiset、map和multimap。通常被实现为binary tree。
3. 无序容器：无序集合，唯一重要的是某个特定元素是否位于此集合内。STL提供4个预定义的无序容器：unordered\_set、unordered\_multiset、unordered\_map和unordered\_multimap。通常被实现为hash table。

vector：将其元素置于一个dynamic array中管理，允许随机访问。

deque：双向队列，允许在头尾两端插入元素，允许随机访问。

例：程序stl\_test29

// 双向队列

*deque*<float> coll;

for (int i = 1; i <= 6; ++i)

{

coll.*push\_front*(i \* 1.1f);

}

for (auto f : coll)

{

*cout* << f << ' ';

}

*cout* << *endl*;

for (int j = 1; j <= 6; ++j)

{

coll.*push\_back*(j \* 1.1f);

}

for (auto f : coll)

{

*cout* << f << ' ';

}

*cout* << *endl*;

输出为：

6.6 5.5 4.4 3.3 2.2 1.1

6.6 5.5 4.4 3.3 2.2 1.1 1.1 2.2 3.3 4.4 5.5 6.6

array：固定大小的array，类似于数组，不能改变元素个数，只能改变元素值，支持随机访问。

例：程序stl\_test29

// array

// 可以使用初值列表进行初始化

array<*string*, 5> coll\_array = {"hello", "world"};

for (array<*string*, 5>::*size\_type* i = 0; i < coll\_array.*size*(); ++i)

{

*cout* << i << ": " << coll\_array[i].*c\_str*() << *endl*;

}

输出为：

0: hello

1: world

2:

3:

4:

list：C++11中包含两个list容器，list<>和forward\_list<>

list<>：由双向链表(doubly linked list)实现而成。list内的每个元素都以一部分内存指示其前导元素和后继元素。不支持随机访问。

list：在任何位置插入或删除都非常迅速，只需改变链接就好。

例：程序stl\_test29

// list<>

*list*<char> coll\_list1;

*list*<char> coll\_list2;

for (char c = 'a'; c <= 'z'; ++c)

{

coll\_list1.*push\_back*(c);

coll\_list2.*push\_back*(c);

}

// 打印链表中的所有元素

for (const auto& elem : coll\_list1)

{

*cout* << elem << ' ';

}

*cout* << *endl*;

// 打印链表中的元素并移除第一个元素

while (!coll\_list2.*empty*())

{

*cout* << coll\_list2.*front*() << ' ';

coll\_list2.*pop\_front*(); // 移除首元素

}

*cout* << *endl*;

forward\_list：单向链表，每个元素都指向下一个元素。