关联式容器：根据特定的排序准则，自动为其元素排序。默认情况下，都以操作符<进行比较。

set：元素根据其value自动排序，每个元素只能出现一次。

multiset：元素可以重复，可包含多个“value相同”的元素。

例：程序stl\_test30

*set*<*string*> city{ "Hanover", "Paris" };

city.*insert*("London");

// 插入一个重复的元素，不会有任何效果

city.*insert*("Paris");

for (const auto& elem : city)

{

*cout* << elem.*c\_str*() << " ";

}

*cout* << *endl*;

*multiset*<*string*> cities{ "Hanover", "Paris" };

cities.*insert*("London");

cities.*insert*("Paris"); // multiset允许重复

for (const auto& elem : cities)

{

*cout* << elem.*c\_str*() << " ";

}

*cout* << *endl*;

输出为：

Hanover London Paris

Hanover London Paris Paris

map：每个元素都是key/value pair。其中key是排序准则的基准，每个key只能出现一次。

multimap：与map的唯一区别是元素可以重复，允许其元素拥有相同的key。map可以支持[]访问，但multimap不支持[]访问。

例：程序stl\_test30

// 第2个key=1不会有效

*map*<int, *string*> coll{ { 5, "tagged" }, { 2, "a" }, { 1, "this" }, {1, "is"} };

for (const auto& elem : coll)

{

*cout* << elem.*first* << ": " << elem.*second*.*c\_str*() << ' ';

}

*cout* << *endl*;

// ok，支持[]访问

*string* tmp = coll[1];

// 第2个key=1有效

*multimap*<int, *string*> colls{ { 5, "tagged" }, { 2, "a" }, { 1, "this" }, { 1, "is" } };

for (const auto& elem : colls)

{

*cout* << elem.*first* << ": " << elem.*second*.*c\_str*() << ' ';

}

*cout* << *endl*;

// error，不支持[]访问

//string tmp = colls[1];