# 关联容器(associative-container)

**分类**：(1) 是一个map或是set (2) 关键字是否允许重复 (3) 元素是否按顺序保存 **特点**：（1）不支持位置容器的相关操作 （2）不支持构造函数或插入操作这些接受一个元素值和一个数量值的操作 （3）迭代器是双向的

## pair

**关联容器**：用于保存一对数据。 **初始化**：pair<type1, type2> variable = { key, value}; **例子**：

pair<string, int> process(vector<string> &v){  
 if (!v.empty())  
 return { v.back(), v.back().size() };  
 else   
 return pair<string, int>();  
 }

## map

**关联数组**：map类型的通称，可以把关键字当成下标使用。 **初始化**：map<type1, type2> variable = { {key1, value1} , {key2, value2} , {key3, value3} }; **例子**：单词计数程序。

map<string, size\_t> word\_count; //指定关键字和值的类型  
string word;  
while (cin >> word) //提取word的计数器并将其加一，如果word不在word\_count当中，下标运算符  
 ++word\_count[word]; //会创建一个值为零的新元素  
for (const auto &w : word\_count)  
 cout << w.first << “occurs” << w.second << endl; //pair类型有两个公有数据成员，first和second

## set

**集合**：同数学上的定义，set中不能有重复元素。 **例子**：

set<string> exclude = {“The”, “But”, “And”};  
cin >> word;  
Exclude.find(word); //查找word，如果不存在就返回尾迭代器。  
exclude.end(); //返回尾迭代器

## 三者之间的关系

**类型别名**：key*type和value*type是三者都有的类型别名，mapped*type是map独有的，它们有如下关系： `set::value*type v1;set::key*value v2;map<type1, type2>::value*type v3;map<type1, type2>::key*type v4;map<type1, type2>::mapped*type v5;` v1、v2、v3的类型是type1，v5的类型是type2，v3的类型是pair<type1, type2>。

## 无序容器

**定义**：使用哈希函数和关键值类型的==运算符来组织元素。能解决效率问题。管理桶。 **有**：unordered*map、unordered*set、unordered\_multi

## 关联容器迭代器

**迭代器是const的**：无法改变关键字，能改变的只有map中的值。 **遍历**：和vector一样使用：for (auto iter = word\_count.begin(); iter != word\_count.end(); ++iter)。 或者另一种写法：for (const auto &w : word\_count)

## 添加、删除元素

insert、erase 返回值：插入是否成功。返回map的迭代器和bool类型的pair。

## 关键字类型

1. 必须定义元素比较的方法。（例如<号）

解引用和下标操作的返回值不同。

find操作：返回一个迭代器 count操作：返回一个unsigned int。 如果关键字还未在map中，进行下标操作会导致插入这个元素。 不能直接定义关键字类型为自定义类类型的无序容器，需要提供==运算符和哈希值计算函数。