1. 概述

关联容器中的元素是按关键字来保存和访问的

关联容器支持高效的关键字查找和访问。两个主要的关联容器(associative-container)类型是 map 和 set。他们各自定义在相同名字的头文件中

- map 中的元素是一些关键字-值（key-value）对:关键字起到索引的作用，值则表示与索引相关联的数据。
- set 中每个元素只包含 Z 一个关键字;set 支持高效的关键字查询操作——检查一个给定关键字是否在 set 中。

例如,在某些文本处理过程中，可以用一个 set 来保存想要忽略的单词。字典则是一个很好的使用 map 的例子:可以将单词作为关键字，将单词释义作为值。

除了基础的 map 和 set 外，标准库还提供了允许重复关键字(multi)和无序(unodered)两种属

性，三者组合形成特殊的关联容器。

按关键字有序保存元素

- map: 关联数组;保存关键字-值对
- set: 关键字即值，即只保存关键字的容器

允许重复且有序保存元素

- multimap : 关键字可重复出现的 map
- multiset : 关键字可重复出现的 set

无序集合:

- unordered_map : 用哈希函数组织的 map
- unordered_set : 用哈希函数组织的 set
- unordered_multimap : 哈希组织的 map;关键字可以重复出现
- unordered_multiset : 哈希组织的 set;关键字可以重复出现

hash 本意是指把任意长度 **输入** 通过散列算法转换为固定长度的 **输出**，是一种压缩映射。
不同于加密，hash 是单向输出，不能通过输出逆转推出输入，而加密可以
应用例如下载 iso 文件时，有的网站会给出正确下载的 hash 输出，之后我们下载文件后就可以通过该 ios 文件算出自己的 hash 输出，如果二者一致则说明下载的文件没有多或者少。

数组是通过**索引**与**值**联系在一起，索引值直接对应值的内存地址，从而达到存储和访问。

哈希表示通过**键**和**值**对应，键是字符串或者整数，而值是数值，通过键在访问值。但是键不是直接对应内存地址，而是要通过**哈希函数**和键算出对应的内存地址。

哈希函数有很多实现方式，但是必须遵守以下原则

- 不可逆，因此就要用到取余 % ，
- 可变长度的输入变成相对固定长度的输出
- 相同输入得到相同输出

但是有时不同的输入会得到相同的输出，也即是冲突。解决冲突通常有两种方法:开放寻址法和封闭寻址法。
链表法就是封闭寻址法的一种，把哈希表中的位置设定为链表，当冲突时，直接把新的输入放在节点的链表中，使用哈希函数得到节点位置和索引值进入对应节点的链表中逐个查询。但是可能发生数据集

通过哈希算法可以把明文映射为键，通过键来对比文本。如果键被盗取，也不能直接得到明文。一般用于数据检索，密码，文件对比，数字指纹等。

map 是关键字(键)-值对的集合。例如，可以将一个人的名字作为关键字，将其电话号码作为值。我们称这样的数据结构为“将名字映射到电话号码”。map 类型通常被称为关联数组(associative array)。关联数组与“正常”数组类似，不同之处在于其下标不必是整数。我们通过一个关键字而不是位置来查找值。给定一个名字到电话号码的 map，我们可以使用一个人的名字作为下标来获取此人的电话号码。

```
//报告单词出现次数
map<string,int> count;//创立空map
string word;
for (int i = 0; i < 3; ++i)
{
  cin >> word;
  ++count[word];//word类似数组的索引
}
for (auto &i : count )
{
  cout << i.first << " : " << i.second << endl;
  //从count中提取成员，first代表关键字，secon代表值
}
```

当从 map 中提取一个元素时,会得到一个 pair 类型的对象，简单来说，pair 是一个模板类型，保存两个名为 first 和 second 的（公有）数据成员。map 所使用的 pair 用 first 成员保存关键字，用 second 成员保存对应的值。因此，输出语句的效果是打印每个单词及其关联的计数器。

对于一些常见的单词例如 or，and 等如果不需要统计其值，就可以使用 set 忽略
//统计输入中每个单词出现的次数

```
map<string,size_t> word_count;
mapset<string> exclude = {"The", "But","And","Or","An", "A", "the",
"but", "and", "or", "an", "a"};
string word;
while (cin >> word)
//只统计不在exclude(不包括)中的单词
if (exclude.find (word)== exclude.end ())
++word_count[word] ; //获取并递增word的计数器
```

成员函数 find 调用返回一个迭代器。如果给定关键字在 set 中,迭代器指向该关键字。否则,find 返回尾后迭代器。在此程序中，仅当 word 不在 exclude 中时我们才更新 word 的计数器。

2.定义关联容器

map 是键-值对应的模板，因此在定义 map 时既要指明 键 类型(第一个类型)，也要指明 值 类型(第二个类型)。
初始化

- 默认构造函数，初始化为空
- 列表初始化，大列表表示整个 map，其中的小列表表示单元中键名和值，二者用 , 隔开
  {key,value}
  set 只有键，因此在定义时只用给定一个参数
  初始化
- 默认初始化
- 列表初始化，只有一个大括号
  {key}

容器 multimap 和 multiset 没有限制一个关键字对应一个值，它们都允许多个元素具有相同的关键字。例如，在我们用来统计单词数量的 map 中，每个单词只能有一个元素。另一方面，在一个词典中，一个特定单词则可具有多个与之关联的词义。

```
multimap<string, int> m1{{"ok", 3}, {"ok", 4}, {"pri", 5}};
for (auto &&i : m1)
{
  cout << i.first << " : " << i.second << endl;
}
cout << endl;
```