## 顺序容器

### 简介：

顺序容器提供了元素存储和访问的能力，这种顺序不依赖于元素的值，而是与元素加入容器时的位置相对应。

|  |  |
| --- | --- |
| **顺序容器类型表** | |
| vector | 可变大小数组。支持快速随机访问。在尾部之外的位置插入或删除元素可能很慢。 |
| deque | 双端队列。支持快速随机访问。在头尾位置插入/删除速度很快。 |
| list | 双向链表。只支持双向顺序访问。在list中任何位置进行插入/删除操作速度都很快。 |
| forward\_list | 单向链表。只支持单向顺序访问。在链表任何位置进行插入/删除操作速度都很快。 |
| array | 固定大小数组。支持快速随机访问。不能添加或删除元素。 |
| string | 与vector相似的容器，但专门用于保存字符。随机访问速度快。在尾部插入/删除速度快。 |

### 操作：

#### 访问：

c.back(); 返回c中尾元素的引用。若c为空，函数行为未定义。

c.front(); 返回c中首元素的引用。若c为空，函数行为未定义。

c[n]; 返回c中下标为n的元素的引用，n为一个无符号的整数。

若n>=c.size(), 则函数行为未定义。

c.at[n]; 返回下标为n的元素的引用。如果下标越界，则抛出一out\_of\_range异常。

#### 大小：

c.size(); c中元素的数目（不支持forward\_list）。

c.max\_size(); c可保存的最大元素数目。

c.empty(); 若c中存储了元素，返回false，否则返回true。

c.resize(n); 调整c的大小为n个元素。若n<c.size(); 则多出的元素被丢弃。

若必须添加新元素，对新元素进行值的初始化。

c.resize(n, t); 调整c的大小为n个元素。任何新添加的元素都初始化为值为t。

shrink\_to\_fit(); 只适用于vector、string、deque。

capacity和reserve只适用于vector和string。

c.shrink\_to\_fit(); 请将capacity()减少为与size()相同大小。

c.capacity(); 不重新分配内存空间的话，c可以保存多少元素。

c.reserve(n); 分配至少能容纳n个元素的内存空间。

#### 添加元素（不适用于array）：

c.insert(args); 将args中的元素拷贝进c。

c.emplace(inits); 使用inits构造c中的一个元素。

c.push\_front(); 将元素插到容器的头部。

c.push\_back(); 将元素追加到容器的尾部。

c.append(args); 将args追加到c，返回一个指向c的引用。

c.replace(range, args); 删除c中范围range内的字符，替换为args指定的字符。

range或者是一个下标和一个长度，或者是一对指向c的迭代器。返回一个指向s的引用。

#### 删除元素（不适用于array）：

c.erase(args); 删除args指定的元素。返回一个指向被删元素之后元素的迭代器，若args指向尾元素，则返回尾后（off\_the\_end）迭代器。若args是尾后迭代器，则函数行为未定义。

c.erase(b,e); 删除迭代器b和e所指定范围内的元素。返回一个指向最后一个被删除元素之后的迭代器，若e本身就是尾后迭代器，则函数也返回尾后迭代器。

c.clear(); 删除c中所有的元素，返回void。

c.pop\_back(); 删除c中的尾元素。若c为空，则函数行为未定义。函数返回void。

c.pop\_front(); 删除c中的首元素。若c为空，则函数行为未定义。函数返回void。

#### 获取迭代器

c.begin(), c.end(); 返回指向c的首元素和尾元素之后的迭代器。

c.cbegin(),c.cend(); 返回const\_inerator。

反向容器的额外成员（不支持forword\_list）

reverse\_inerator; 按逆序寻址元素的迭代器。

const\_reserse\_iterator; 不能修改元素的逆序迭代器。

c.rbegin(),c.rend(); 返回指向c的尾元素和首元素之前位置的迭代器。

c.crbegin(),c.crend(); 返回const\_reverse\_iterator。

#### 其他：

swap(c1, c2); 交换c1和c2中的元素。c1和c2必须具有相同的类型。swap通c1.swap(c2); 常比从c2向c1拷贝元素快的多。

assign 操作不适用于关联容器和array。

seq.assign(b,e); 将seq中的元素替换为迭代器b和e所表示范围中的元素。迭代器b和e不能指向seq中的元素。

seq.assign(il); 将seq中的元素替换为初始化列表il中的元素。

seq.assign(n,t); 将seq中的元素替换为n个值为t的元素。

### string搜索操作：

|  |  |  |
| --- | --- | --- |
| **string搜索操作**  (搜索操作返回指定字符出现的下标，如果未找到则返回npos。) | | |
| s.find(args); | 查找s中args第一次出现的位置。 | |
| s.rfind(args); | 查找s中args最后一次出现的位置。 | |
| s.find\_first\_of(args); | 在s中查找args中任何一个字符第一次出现的位置。 | |
| s.fing\_last\_of(args); | 在s中查找args中任何一个字符最后一次出现的位置。 | |
| s.find\_first\_not\_of(args); | 在s中查找第一个不在args中的字符。 | |
| s.find\_last\_not\_of(args); | 在s中查找最后一个不在args中的字符。 | |
|  | | |
| args必须是以下形式之一 | | |
| c, pos | | 从s中位置pos开始查找字符c。pos默认为0。 |
| s2,pos | | 从s中位置pos开始查找字符串s2。pos默认为0。 |
| cp,pos | | 从s中位置pos开始查找指针cp指向的以空字符结尾的C风格字符串。pos默认为0。 |
| cp,pos,n | | 从s中位置pos开始查找指针cp指向的数组的前n个字符。pos和n无默认值。 |

### compare函数：

这些函数与C标准库的strcmp函数很相似，类似strcmp，根据s是等于、大于还是小于参数指定的字符串，s.compare返回0、正数或负数。

|  |  |
| --- | --- |
| **s.compare的几种参数形式** | |
| s2 | 比较s和s2。 |
| pos1，n1，s2 | 将s中从pos1开始的n1个字符与s2进行比较。 |
| pos1，n1，s2，pos2，n2 | 将s中从pos1开始的n1个字符与cp指向的以空字符结尾的字符数组进行比较。 |
| cp | 比较s与cp指向的以空字符结尾的字符数组。 |
| pos1，n1，cp | 将s中从pos1开始的n1个字符与cp指向的以空字符结尾的字符数组进行比较。 |
| pos1，n1，cp，n2 | 将s中从pos1开始的n1个字符串与指针cp指向的地址开始的n2个字符进行比较。 |

### 数值转换：

|  |  |
| --- | --- |
| **string和数值之间的转换** | |
| to\_string(val) | 一组重载函数，返回数值val的string表示。val可以是任何算术类型，对每个浮点类型和int或更大的整型，都有相应的版本的to\_string。与往常一样，小整形会被提升。 |
| stoi(s, p, b)  stol(s, p, b)  stoul(s, p, b)  stoll(s, p, b)  stoull(s, p, b) | 返回s的起始子串（表示整数内容）的数值，返回值类型分别是int、long、unsigned long、long long、unsigned long long。b表示转换所用的基数，默认值为10。p是size\_t指针，用来保存s中的第一个非数值字符的下标，p默认为0，即，函数不保存下标。 |
| stof(s, p)  stof(s, p)  stold(s, p) | 返回s起始子串（表示浮点数内容）的数值，返回值类型分别是float、double或long double。参数p的作用于整数转换函数中一样。 |

## 二、关联容器：

关联容器和顺序容器有着根本的不同：关联容器中的元素是按关键字来保存和访问的。与之相对，顺序容器中的元素是按它们在容器中的位置来顺序保存和访问的。

|  |  |
| --- | --- |
| **关联容器类型** | |
| 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，即只保存关键字容器。 |
| 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。 |

### 关联容器操作：

|  |  |
| --- | --- |
| **关联容器额外的类型别名** | |
| key\_type | 此容器类型的关键字类型。 |
| mapped\_type | 每个关键字关联的类型；只适用于map。 |
| value\_type | 对于set，与key\_type相同  对于map，为pair<const key\_type, mapped\_type> |

### 操作：

#### 添加元素：

|  |  |
| --- | --- |
| **关联容器insert 操作** | |
| c.insert(v) | v是value\_type类型的对象；args用来构造一个元素。 |
| c.emplace(args) | 对于map和set，只有当元素的关键字不在c中是才可以插入（或构造）元素。函数返回一个pair，包含一个迭代器，指向具有指定关键字的元素，以及一个指向插入是否成功的bool值。  对于multimap和multiset，总会插入（或构造）给定元素，并返回一个指向新元素的迭代器。 |
| c.insert(b, e)  c.insert(il) | b和e是迭代器，表示一个c::value\_type类型值的范围；il是这种值的花括号列表。函数返回void。  对于map和set，只插入关键字不在c中的元素。对于multimap和multimap，则会插入范围中的每一个元素。 |
| c.insert(p, v)  c.emplace(p, args) | 类似insert(v)(或emplace(args))，但将迭代器p作为一个提示，指出从哪里开始搜索新元素应该存储的位置。返回一个迭代器，指向具有给定给定关键字的元素。 |

#### 删除元素：

|  |  |
| --- | --- |
| **从关联容器删除元素** | |
| c.erase(k) | 从c中删除每个关键字为k的元素。返回一个size\_type的值，值出删除的元素的数量。 |
| c.erase(p) | 从c中删除迭代器p指向的元素。p必须指向c中一个真实元素，不能等于c.end()。返回一个指向p之后元素的迭代器，若p指向c中的尾元素，则返回c.end()。 |
| c.erase(b, e) | 删除迭代器对b和e所表示范围中的元素。返回e。 |

#### map的下标操作：

|  |  |
| --- | --- |
| **map和unordered\_map的下标操作** | |
| c[k] | 返回关键字为k的元素；如果k不在c中，添加一个关键字为k的元素，对其进行值初始化。 |
| c.at(k) | 访问关键字为k的元素，带参数检查；若k不在c中，抛出一个out\_of\_range异常。 |

#### 访问元素：

|  |  |
| --- | --- |
| **在一个关联容器中查找元素的操作** | |
| lower\_bound和upper\_bound不适用于无序容器。  下标和at操作只适用于非const的map和unordered\_map。 | |
| c.find(k) | 返回一个迭代器，指向第一个关键字为k的元素，若k不在容器中，则返回尾后迭代器。 |
| c.count(k) | 返回关键字等于k的元素的数量。对于不允许重复关键字的容器，返回值永远是0或1。 |
| c.lower\_bound(k) | 返回一个迭代器，指向第一个关键字不小于k的元素。 |
| c.upper\_bound(k) | 返回一个迭代器，指向第一个关键字大于k的元素。 |
| c.equal\_range(k) | 返回一个迭代器pair，表示关键字等于k的元素的范围。若不存在，pair的两个成员均等于c.end()。 |