STL容器

- 顺序容器

容器类型	说明
array	静态数组，大小固定，支持快速随机访问，不能添加或删除元素
vector	动态数组，大小可变，支持快速随机访问，在尾部之外的位置插入、删除元素速度很慢
deque	双端队列，支持快速随机访问，在头尾插入删除元素速度很快，在其他位置插入删除元素较慢
list	双向链表，只支持双向顺序访问，在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，在任何位置进行插入删除操作速度都很快

- 容器适配器
  一个容器适配器接受一种已有的容器类型，使其行为看起来像一种不同的容器类型。

容器类型	说明
stack	栈适配器，默认基于deque实现
queue	队列适配器，默认基于deque实现
priority_queue	优先队列适配器，默认基于vector实现

- 关联容器

容器类型	说明
set	有序集合，关键字即值
multiset	关键字可重复出现的set
map	关联数组，保存“关键字-值”对
multimap	关键字可重复出现的map

- 无序关联容器

容器类型	说明
unordered_set	用哈希函数组织的set
unordered_multiset	关键字可重复出现的unordered_set
unordered_map	用哈希函数组织的map
unordered_multimap	关键字可重复出现的unordered_map

1. vector

vector是一种动态数组，在内存中具有连续的存储空间，支持快速随机访问，但是在插入和删除操作方面效率比较慢。

1. 构造

功能	函数
初始化为3个0	vector <int> v1(3)
初始化为5个2	vector <int> v2(5, 2)
用另一个vector初始化	vector <int> v3(v2)
用另一个vector一定范围内的值初始化	vector <int> v4(v2.begin(), v2.begin() + 3)
列表初始化	vector<int> v5{ 1,2,3,4,5 }

2. 赋值

功能	函数
赋值为5个2	v1.assign(5, 2)
用另一个vector一定范围内的值进行赋值	v2.assign(v1.begin(), v1.end())
列表赋值	v3.assign({ 5, 6, 7 })

3. 获取长度

功能	函数
获取长度	v1.size()

4. 是否为空

功能	函数
是否为空	v1.empty()

5. 清空

功能	函数
清空	v1.clear()

6. 插入

功能	函数
末尾插入元素	v1.push_back(8)
在某一索引前插入一个元素	v1.insert(v1.begin() + 1, 12)
在某一索引前插入多个元素	v1.insert(v1.begin() + 1, 5, 12)
在某一索引前插入另一个vector中一定范围的元素	v1.insert(v1.end(),v2.begin(),v2.end())

7. 删除

功能	函数
删除末尾元素	v1.pop_back()
删除指定元素	v1.erase(v1.begin()+3)
删除一定范围元素	v1.erase(v1.begin() + 2, v1.begin() + 4)

8. 交换

功能	函数
交换内容	v1.swap(v2)

2. list

list是一个双向链表，内存空间可以不连续，无法进行随机访问，迭代器只能进行单步加减。但list可以快速地在任意地方进行插入和删除操作，而且插入操作和删除操作不会造成原有的list迭代器失效。

1. 构造

功能	函数
初始化为3个0	list<double> list1(3)
初始化为5个2	list<double> list2(5, 2)
用另一个list初始化	list<double> list3(list2)
用另一个list一定范围内的值初始化	list<double> list5(++list3.begin(),--list3.end())
列表初始化	list<double> list5{1,2,3,4,5,6}

2. 赋值

功能	函数
赋值为5个2	list1.assign(5, 2)
用另一个list一定范围内的值进行赋值	list2.assign(list1.begin(), --list1.end())
列表赋值	list3.assign({ 5, 6, 7 })

3. 获取长度

功能	函数
获取长度	list1.size()

4. 是否为空

功能	函数
是否为空	list1.empty()

5. 清空

功能	函数
清空	list1.clear()

6. 插入

功能	函数
末尾插入元素	list1.push_back(8.8)
头部插入元素	list1.push_front(1.1)
在某一索引前插入一个元素	list1.insert(list1.begin() + 1, 12)
在某一索引前插入5个元素12	list1.insert(list1.begin() + 3, 5, 12)
在某一索引前插入另一个list中一定范围的元素	list1.insert(list1.begin(), list2.begin(), --list2.end())

7. 删除

功能	函数
删除末尾元素	list1.pop_back()
删除头部元素	list1.pop_front()
删除指定元素	list1.erase(list1.begin()+3)
删除一定范围元素	list1.erase(++list1.begin() , list1.end())

8. 移除

功能	函数
移除特定值元素	list1.remove(5)
移除相邻的重复元素	list1.unique()

9. 融合有序链表

功能	函数
融合有序链表	list1.merge(list2)

10. 排序

功能	函数
排序	list1.sort()

11. 翻转

功能	函数
翻转	list1.reverse()

12. 交换

功能	函数
交换	list1.swap(list2)

3. stack

功能	函数
栈顶插入	push()
栈顶删除	pop()
获取栈顶元素	top()
获取长度	size()
是否空	empty()

4. queue

功能	函数
队尾插入	push()
队首删除	pop()
获取队首元素	front()
获取队尾元素	back()
获取长度	size()
是否空	empty()

5. deque

1. 构造

功能	函数
初始化为3个0	deque<double> deque1(3)
初始化为5个2	deque<double> deque2(5, 2)
用另一个deque初始化	deque<double> deque3(deque2)
用另一个deque一定范围内的值初始化	deque<double> deque5(++deque3.begin(),--deque3.end())
列表初始化	deque<double> deque5{1,2,3,4,5,6}

2. 赋值

功能	函数
赋值为5个2	deque1.assign(5, 2)
用另一个deque一定范围内的值进行赋值	deque2.assign(deque1.begin(), --deque1.end())
列表赋值	deque3.assign({ 5, 6, 7 })

3. 获取长度

功能	函数
获取长度	deque1.size()

4. 是否为空

功能	函数
是否为空	deque1.empty()

5. 清空

功能	函数
清空	deque1.clear()

6. 插入

功能	函数
末尾插入元素	deque1.push_back(8.8)
头部插入元素	deque1.push_front(1.1)
在某一索引前插入一个元素	deque1.insert(deque1.begin() + 1, 12)
在某一索引前插入5个元素12	deque1.insert(deque1.begin() + 3, 5, 12)
在某一索引前插入另一个deque中一定范围的元素	deque1.insert(deque1.begin(), deque2.begin(), --deque2.end())

7. 删除

功能	函数
删除末尾元素	deque1.pop_back()
删除头部元素	deque1.pop_front()
删除指定元素	deque1.erase(deque1.begin()+3)
删除一定范围元素	deque1.erase(++deque1.begin() , deque1.end())

8. 交换

功能	函数
交换	deque1.swap(deque2)

6. string

1. 构造

功能	函数
以字符串构造	string str1("wangang")
以字符串前4个字符构造	string str2("wangang", 4)
以string构造	string str3(str1)
以string从0开始的4个字符构造	string str4(str1, 0, 4)
以string一定范围内的字符构造	string str5(str1.begin() + 3, str1.begin()+6)
以5个同样的字符构造	string str6(5, 's')

2. 赋值

功能	函数
以字符串赋值	str1.assign("Hinbe")
以字符串前4个字符赋值	str1.assign("Hinbe",4)
以string赋值	str1.assign(str2)
以string从0开始的4个字符赋值	str1.assign(str3, 0, 4)
以string一定范围内的字符赋值	str1.assign(str3.begin() + 2, str3.begin() + 9)
以5个同样的字符构造	str1.assign(5, 'K')

3. 获取长度

功能	函数
获取长度	str1.size()

4. 是否为空

功能	函数
是否为空	str1.empty()

5. 清空

功能	函数
清空	str1.clear()

6. 获取子串

功能	函数
获取位置0后的6个字符	str1.substr(0, 6)

7. 转常量字符串

功能	函数
string 转常量字符串(以\0结尾）	const char *s1 = str1.c_str()

8. 转字符数组

功能	函数
string 转常量字符数组	const char *s2 = str1.data()

9. 末尾添加

功能	函数
末尾添加string	str1.append(str2)
末尾添加string从4开始的5个字符	str1.append(str3, 4, 5)
末尾添加string一定范围内的字符	str1.append(str3.begin(), str3.begin() + 6)
末尾添加字符串	str1.append("ABC")
末尾添加字符串的前3个字符	str1.append("12345678", 3)
末尾添加5个同样的字符	str1.append(5, 'V')

10. 比较

功能	函数
string比较	str1.compare(str2)
str1的子串（从索引3开始，包含4个字符） 与str2进行比较	str1.compare(3, 4, str2)
str1的子串（从索引3开始，包含4个字符）与str2的子串（从索引3开始，包含4个字符）进行比较	str1.compare(3, 4, str2, 3, 4)
str1与字符串进行比较	str1.compare("hi,hello")
str1指定子串（从索引0开始，包含2个字符） 与字符串进行比较	str1.compare(0, 2, "hi")
str1的子串（从索引0开始，包含5个字符） 与字符串的前5个字符进行比较	str1.compare(0, 5, "hi,hello", 5)

11. 插入

功能	函数
在末尾插入一个字符	str1.push_back('D')
在指定位置插入一个string	str1.insert(1, str2)
在指定位置插入一个string的子串（从索引0开始，包含2个字符）	str1.insert(0, str2, 0, 2)
在指定位置插入另一string的一定范围字符	str1.insert(str1.end(), str2.begin(), str2.begin() + 5)
在指定位置插入一个字符串	str1.insert(0, "JKLMN")
在指定位置插入一个字符串的前3个字符	str1.insert(0, st1,3)
在指定位置插5个同样的字符	str1.insert(3, 5, 'V')

12. 删除

功能	函数
删除最后一个字符	str1.pop_back()
删除指定位置字符	str1.erase(str1.begin())
从指定位置开始删除3个字符	str1.erase(1, 3)
删除一定范围字符	str1.erase(str1.begin(), str1.begin() + 3)

13. 替换

功能	函数
将从1开始的3个字符替换为str2	str1.replace(1, 3, str2)
将从1开始的3个字符替换为字符串	str1.replace(1, 3, "ABCDE")
将从1开始的3个字符替换为字符串的前3个字符	str1.replace(1, 3, "XYZUVW", 3)
将从1开始的3个字符替换为5个字符	str1.replace(1, 3, 5, 'P')
将从1开始的3个字符替换为str2从0开始的3个字符	str1.replace(1, 3, str2, 0, 3)
将一定范围的字符替换为str2	str1.replace(str1.begin()+1, str1.begin()+4, str2)
将一定范围的字符替换为字符串	str1.replace(str1.begin() + 1, str1.begin() + 4, "XYZ")
将一定范围的字符替换为字符串的前3个字符	str1.replace(str1.begin() + 1, str1.begin() + 4, "XYZUVW", 3)
将一定范围的字符替换为5个字符	str1.replace(str1.begin() + 1, str1.begin() + 4, 5, 'Q')
将一定范围的字符替换为另一string一定范围的字符	str1.replace(str1.begin(), str1.begin() + 4, str2.begin(), str2.begin() + 4)

14. 查找

- find 从源字符串起始位置pos(默认为0)处，查找有目标字符串str的位置，如果找到，则返回首次匹配的开始位置，如果没有找到匹配的内容，则返回string::npos。

- rfind 从源字符串起始位置pos(默认为npos)处，倒序查找有目标字符串str的位置，如果找到，则返回首次匹配的开始位置，如果没有找到匹配的内容，则返回string::npos。
- find_first_of 从源字符串起始位置pos(默认为0)处，依此查找每个字符。如果某字符在目标字符串中，则返回首次匹配的该字符的位置,否则返回string::npos。
- find_first_not_of 从源字符串起始位置pos(默认为0)处，依此查找每个字符。如果某字符不在目标字符串中，则返回首次不匹配的该字符的位置,如果全部匹配则返回string::npos。
- find_last_of 从源字符串起始位置pos(默认为npos)处，倒序依此查找每个字符。如果某字符在目标字符串中，则返回首次匹配的该字符的位置,否则返回string::npos。
- find_last_not_of 从源字符串起始位置pos(默认为npos)处，倒序依此查找每个字符。如果某字符不在目标字符串str中，则返回首次不匹配的该字符的位置,否则返回string::npos。

功能	函数
从指定位置开始查找string	str11.find(str22,5)
从指定位置开始查找字符串	str11.find("needle", 5)
从指定位置15开始查找子串的前6个字符	str11.find("needles are small", 15, 6)
从指定位置开始查找字符	str11.find('a')

15. 类型转换

功能	函数
讲其他类型转为string类型	string pi = to_string(3.1415926)
Convert string to integer	int num = stoi(str1)
Convert string to long int	stol
Convert string to unsigned integer	stoul
Convert string to long long	stoll
Convert string to unsigned long long	stoull
Convert string to float	stof
Convert string to double	stod
Convert string to long double	stold

16. 字符操作

功能	函数
大写字符转小写字符	char ch_low = tolower(ch_up)
小写字符转大写字符	char ch_up = toupper(ch_low)

ASCII values	characters	iscntrl	isblank	isspace	isupper	islower	isalpha	isdigit	isxdigit	isalnum	ispunct	isgraph	isprint
0x00 .. 0x08	NUL, (other control codes)	x											
0x09	tab ('\t')	x	x	x									
0x0A .. 0x0D	(white-space control codes: '\f','\v','\n','\r')	x		x									
0x0E .. 0x1F	(other control codes)	x											
0x20	space (' ')		x	x									x
0x21 .. 0x2F	!"#$%&' ()*+,-./										x	x	x
0x30 .. 0x39	0123456789							x	x	x		x	x
0x3a .. 0x40	:;<=>?@										x	x	x
0x41 .. 0x46	ABCDEF				x		x		x	x		x	x
0x47 .. 0x5A	GHIJKLMNOPQRSTUVWXYZ				x		x			x		x	x
0x5B .. 0x60	[\]^_`										x	x	x
0x61 .. 0x66	abcdef					x	x		x	x		x	x
0x67 .. 0x7A	ghijklmnopqrstuvwxyz					x	x			x		x	x
0x7B .. 0x7E	{|}~										x	x	x
0x7F	(DEL)	x											

7. map/unordered_map

1. 构造

功能	函数
构造空unordered_map	unordered_map <int, int> m1
以初始化列表构造	unordered_map<int, int> m2{ { 1,10 }, { 2, 20 } }
复制构造	unordered_map<int, int> m3(m1)
用另一个unordered_map一定范围进行构造	unordered_map<int, int> m4(m1.begin(),m1.end())

2. 获取长度

功能	函数
获取长度	m1.size()

3. 是否为空

功能	函数
是否为空	m1.empty()

4. 清空

功能	函数
清空	m1.clear()

5. 获取指定键值的元素个数（是否存在）

功能	函数
清空	m1.count(10)

6. 插入

功能	函数
插入单个元素	m4[1] = 10
插入单个元素	m4.insert({ 2,20 })
插入单个元素	m4.insert(pair<int, int>(3, 30))
插入一定范围元素	m4.insert(m3.begin(), m3.end())
插入初始化列表	m4.insert({ { 10,100 },{ 11,110 } })

7. 删除

功能	函数
由键值删除	m4.erase(11)
由迭代器删除	m4.erase(m4.begin())
删除一定范围	m4.erase(m4.begin(), ++m4.begin())

8. 查找

功能	函数
查找指定键值元素的迭代器	m4.find(10)

9. 交换

功能	函数
交换两个unordered_map的内容	m4.swap(m3)

8. set/unordered_set

1. 构造

功能	函数
构造空unordered_set	unordered_set s1
以初始化列表构造	unordered_set s2{ 11,22,33 }
复制构造	unordered_set s3(s1)
用另一个unordered_set一定范围进行构造	unordered_set s4(s1.begin(),s1.end())
利用数组、 其他容器一定范围元素进行构造	unordered_sets5(a, a + 4)

2. 获取长度

功能	函数
获取长度	s1.size()

3. 是否为空

功能	函数
是否为空	s1.empty()

4. 清空

功能	函数
清空	s1.clear()

5. 获取指定键值的元素个数（是否存在）

功能	函数
清空	s1.count(10)

6. 插入

功能	函数
插入单个元素	s6.insert(111)
插入初始化列表	s6.insert({ 222,333,444 })
插入一定范围元素	s6.insert(s1.begin(), s1.end())
插入数组、其他容器一定范围元素	s6.insert(a, a + 3)

7. 删除

功能	函数
由值删除	s6.erase(222)
由迭代器删除	s6.erase(s6.begin())
删除一定范围	s6.erase(s6.begin(), ++s6.begin())

8. 查找

功能	函数
查找指定元素的迭代器	s6.find(333)

9. 交换

功能	函数
交换两个unordered_map的内容	s6.swap(s1)