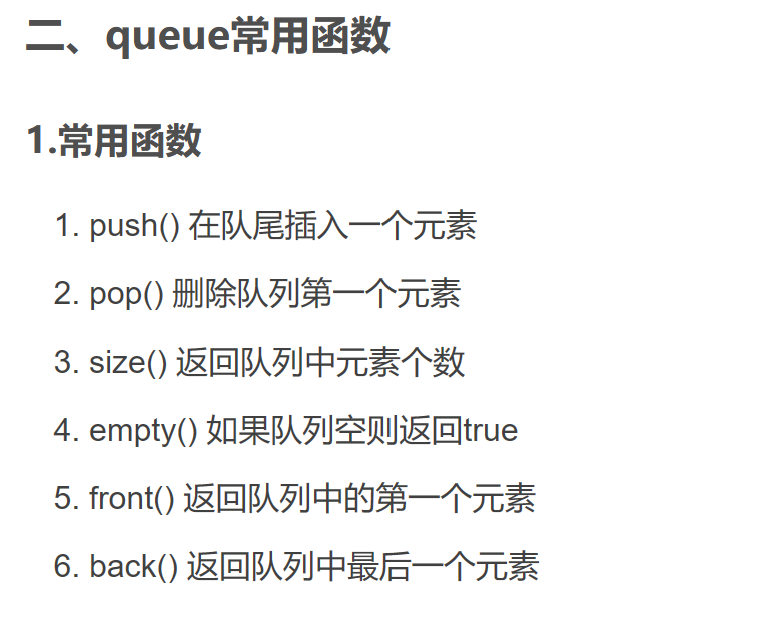
C++学习

## **std::string::**[npos](https://so.csdn.net/so/search?q=npos&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/lmb1612977696/article/details/_blank)

std::string::npos是一个常数，它等于size\_type类型可以表示的最大值，用来表示一个不存在的位置,类型一般是std::container\_type::size\_type。



**与C风格的字符串不同，C++中 string 的结尾没有结束标志'\0'。**

**C++中string 容器中也有pop\_back接口删除最后一个容器。**

**C++中map默认的value值为0；**

在C语言中，你可以使用strlen函数来计算一个以null终止的字符串的长度，这与C++是一样的。strlen函数是C语言标准库中的一个函数，可以计算一个字符串的长度，不包括最后的null字符。

例如：

**c**复制代码

|  |  |
| --- | --- |
|  | #include <string.h> |
|  |  |
|  | // ... |
|  |  |
|  | char \*a = "this { is [ a test ] } "; |
|  | size\_t length = strlen(a); |

这将会计算出字符串的长度。注意，strlen函数不会工作如果你有一个未初始化的指针或者指针没有指向一个以null终止的字符串。如果你的字符串是动态生成的，或者你是从别的地方（例如一个函数）得到的指针，你也需要确保这个字符串是以null终止的。



在C++中，小顶堆是一种特殊的二叉堆，它满足以下性质：

根节点比左右子节点小，即堆顶元素是最小的。

堆是完全二叉树，可以用数组来实现。

**当向堆中插入元素时，新元素会被放在数组的末尾，然后进行“上浮”操作，以保证堆顶元素最小。**

在进行上浮操作时，**如果一个节点比其父节点小，则将该节点与其父节点交换位置。这个交换过程会导致该节点的子节点成为新的父节点，因此还需要继续比较和交换，直到该节点比其父节点大或者到达堆顶。**

通过这个上浮操作，每次插入新元素后，堆顶仍然是堆中最小的元素。因此，在C++中，使用小顶堆函数可以实现升序操作。

mycomparision是一个类其构造如下：

class mycomparision {

public:

bool operator()(const pair<int,int> lhs, const pair<int,int> rhs) {

return lhs.second > rhs.second;

}

};

一开始我还以为是不是运算符重载写错了，后来发现这个类不是一般的类它也叫仿函数可以像一般的函数一样调用，重载了()运算符

所以说第三个函数其实是一种自定义的比较方法，可以用greater()实现，大顶堆默认用less实现

也可以用仿函数实现,

看上去仿函数好像要简单些，可以用于sort(), priority\_queue的第三个参数

#include <functional>

因为：**当向堆中插入元素时，新元素会被放在数组的末尾，然后进行“上浮”操作，以保证堆顶元素最小。**

所以构建大顶堆用降序，小顶堆用升序。

//构造一个空的优先队列（此优先队列默认为大顶堆）

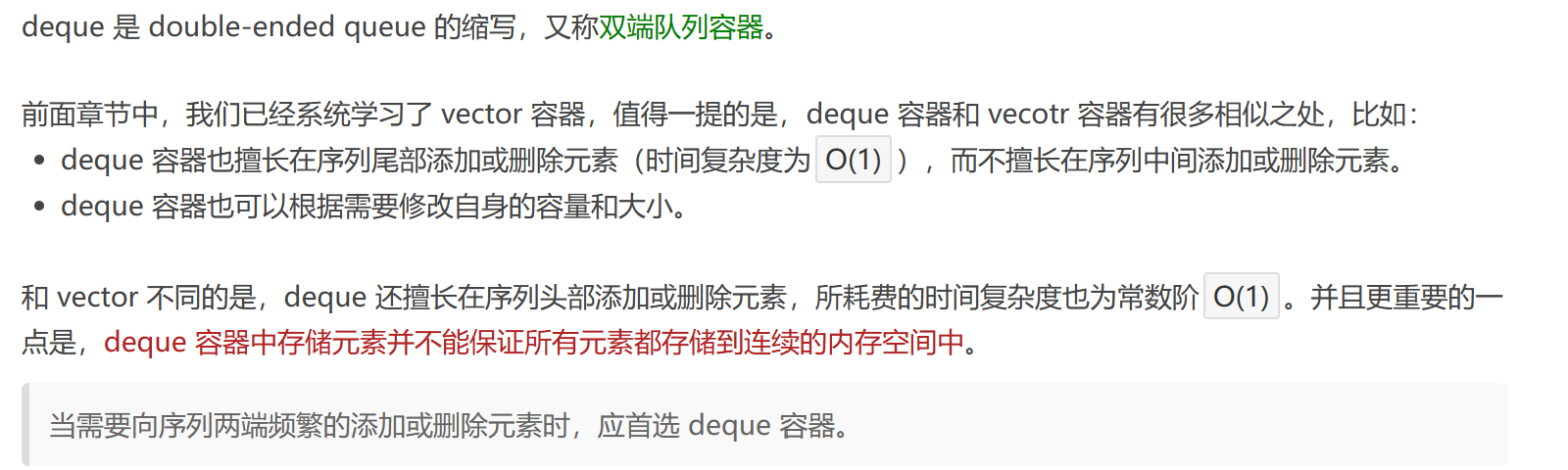
priority\_queue<int> big\_heap;

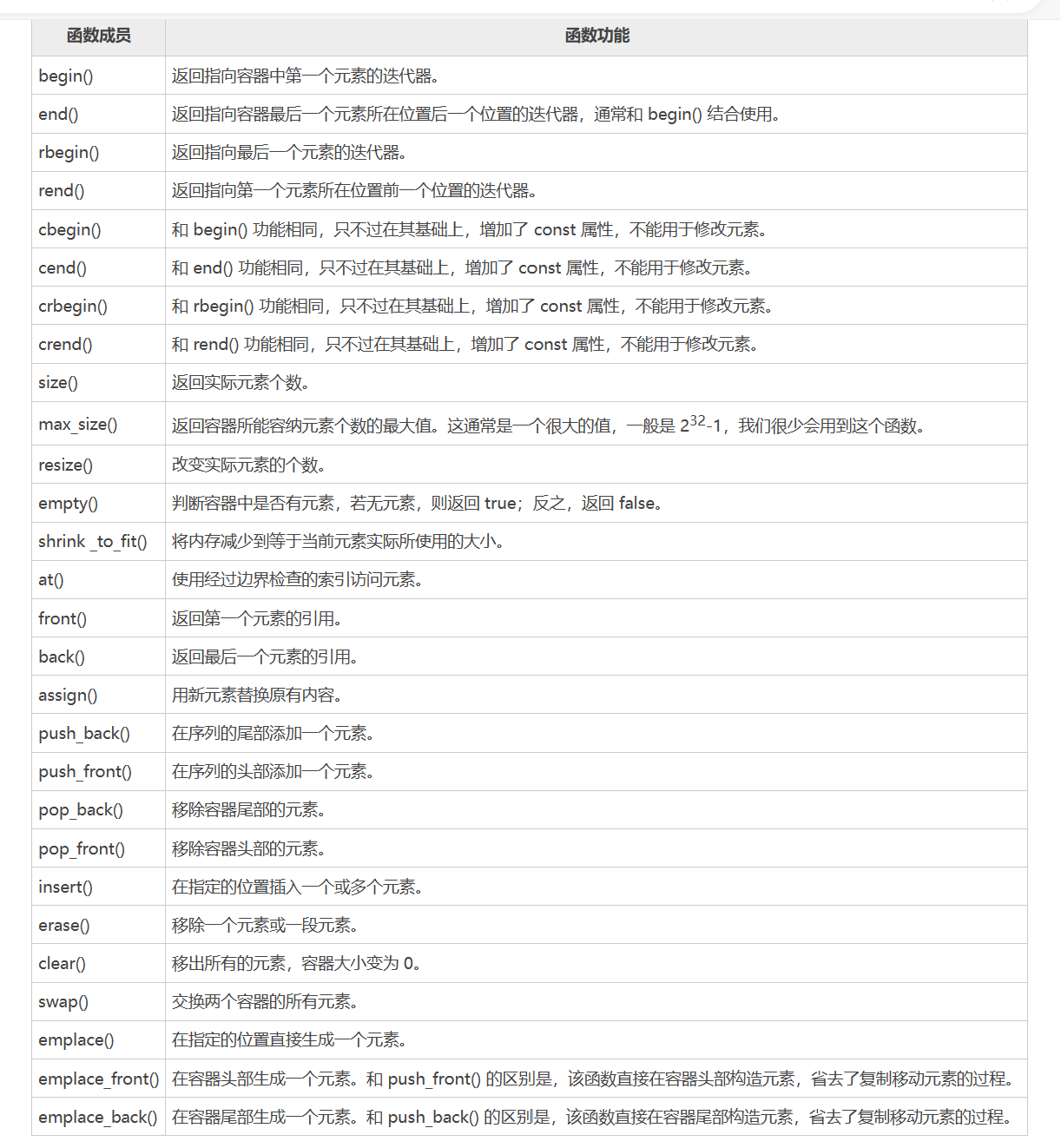
//另一种构建大顶堆的方法

priority\_queue<int,vector<int>,less<int> > big\_heap2;

//构造一个空的优先队列,此优先队列是一个小顶堆

priority\_queue<int,vector<int>,greater<int> > small\_heap;





#ifndef \_\_int8\_t\_defined

# define \_\_int8\_t\_defined

typedef signed char int8\_t;

typedef short int int16\_t;

typedef int int32\_t;

# if \_\_WORDSIZE == 64

typedef long int int64\_t;

# else

\_\_extension\_\_

typedef long long int int64\_t;

# endif

#endif

typedef unsigned char uint8\_t;

typedef unsigned short int uint16\_t;

#ifndef \_\_uint32\_t\_defined

typedef unsigned int uint32\_t;

# define \_\_uint32\_t\_defined

#endif

#if \_\_WORDSIZE == 64

typedef unsigned long int uint64\_t;

#else

\_\_extension\_\_

typedef unsigned long long int uint64\_t;

#endif

 class small{//防函数的使用

        public:

       bool operator ()(vector<int>&a,vector<int>&b){

            return a[0]<b[0];

        }

    };

sort(ans.begin(),ans.end(),small());

Map 的 value值在C++ 中被初始化为0；

C语言islower函数用于判断字符是否为小写字母（a-z）



