# **[priority\_queue的用法（含自定义排序方式）](https://www.cnblogs.com/yalphait/articles/8889221.html)**

priority\_queue本质是一个堆。

1. 头文件是#include<queue>

2. 关于priority\_queue中元素的比较

模板申明带3个参数：priority\_queue<Type, Container, Functional>，其中Type 为数据类型，Container为保存数据的容器，Functional 为元素比较方式。

Container必须是用数组实现的容器，比如vector,deque等等，但不能用 list。STL里面默认用的是vector。

2.1 比较方式默认用operator<，所以如果把后面2个参数缺省的话，优先队列就是大顶堆（降序），队头元素最大。**特别注意pair的比较函数**。

以下代码返回一个**降序**输出：

#include <iostream>

#include <queue>

using namespace std;

int main(){

priority\_queue<int> q;

for( int i= 0; i< 10; ++i ) q.push(i);

while( !q.empty() ){

cout<<q.top()<<endl;

q.pop();

}

return 0;

}

以下代代码返回pair的比较结果，先按照pair的first元素降序，first元素相等时，再按照second元素降序：

#include<iostream>

#include<vector>

#include<queue>

using namespace std;

int main(){

priority\_queue<pair<int,int> > coll;

pair<int,int> a(3,4);

pair<int,int> b(3,5);

pair<int,int> c(4,3);

coll.push(c);

coll.push(b);

coll.push(a);

while(!coll.empty())

{

cout<<coll.top().first<<"\t"<<coll.top().second<<endl;

coll.pop();

}

return 0;

}

2.2 如果要用到小顶堆，则一般要把模板的3个参数都带进去。STL里面定义了一个仿函数greater<>，基本类型可以用这个仿函数声明小顶堆。

　　以下代码返回一个升序输出：

#include <iostream>

#include <queue>

using namespace std;

int main(){

priority\_queue<int, vector<int>, greater<int> > q;

for( int i= 0; i< 10; ++i ) q.push(10-i);

while( !q.empty() ){

cout << q.top() << endl;

q.pop();

}

return 0;

}

以下代代码返回pair的比较结果，先按照pair的first元素升序，first元素相等时，再按照second元素升序：

#include<iostream>

#include<vector>

#include<queue>

using namespace std;

int main(){

priority\_queue<pair<int,int>,vector<pair<int,int> >,greater<pair<int,int> > > coll;

pair<int,int> a(3,4);

pair<int,int> b(3,5);

pair<int,int> c(4,3);

coll.push(c);

coll.push(b);

coll.push(a);

while(!coll.empty())

{

cout<<coll.top().first<<"\t"<<coll.top().second<<endl;

coll.pop();

}

return 0;

}

2.3 对于自定义类型，则必须重载operator<或者重写仿函数。

2.3.1 重载operator<的例子：返回true时，说明左边形参的优先级低于右边形参

#include <iostream>

#include <queue>

#include<stdlib.h>

using namespace std;

struct Node{

int x, y;

Node(int a=0, int b=0):

x(a),y(b){}

};

bool operator<(Node a, Node b){//返回true时，说明a的优先级低于b

//x值较大的Node优先级低（x小的Node排在队前）

//x相等时，y大的优先级低（y小的Node排在队前）

if( a.x== b.x ) return a.y> b.y;

return a.x> b.x;

}

int main(){

priority\_queue<Node> q;

for( int i= 0; i< 10; ++i )

q.push( Node( rand(), rand() ) );

while( !q.empty() ){

cout << q.top().x << ' ' << q.top().y << endl;

q.pop();

}

return 0;

}

自定义类型重载operator<后，声明对象时就可以只带一个模板参数。

但此时不能像基本类型这样声明priority\_queue<Node,vector<Node>,greater<Node> >，原因是greater<Node>没有定义，如果想用这种方法定义则可以重载operator >。

例子：返回的是小顶堆。但不怎么用，习惯是重载operator<。

#include <iostream>

#include <queue>

#include<stdlib.h>

using namespace std;

struct Node{

int x, y;

Node( int a= 0, int b= 0 ):

x(a), y(b) {}

};

bool operator>( Node a, Node b ){//返回true，a的优先级大于b

//x大的排在队前部；x相同时，y大的排在队前部

if( a.x== b.x ) return a.y> b.y;

return a.x> b.x;

}

int main(){

priority\_queue<Node,vector<Node>,greater<Node> > q;

for( int i= 0; i< 10; ++i )

q.push( Node( rand(), rand() ) );

while( !q.empty() ){

cout << q.top().x << ' ' << q.top().y << endl;

q.pop();

}

return 0;

}

2.3.2 重写仿函数的例子（返回值排序与2.3.1相同，都是小顶堆。先按x升序，x相等时，再按y升序）：

#include <iostream>

#include <queue>

using namespace std;

struct Node{

int x, y;

Node( int a= 0, int b= 0 ):

x(a), y(b) {}

};

struct cmp{

bool operator() ( Node a, Node b ){//默认是less函数

//返回true时，a的优先级低于b的优先级（a排在b的后面）

if( a.x== b.x ) return a.y> b.y;

return a.x> b.x; }

};

int main(){

priority\_queue<Node, vector<Node>, cmp> q;

for( int i= 0; i< 10; ++i )

q.push( Node( rand(), rand() ) );

while( !q.empty() ){

cout << q.top().x << ' ' << q.top().y << endl;

q.pop();

}

return 0;

}