原创 STL模板

2019-10-17 21:37:08 _-Y-_-Y-_ 阅读数 1 文章标签: ACM 更多 编辑

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/weixin_44410512/article/details/102615683

lower_bound

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int main(){
  int A[14]={1,1,2,2,2,4,5,5,6,8,8,8,10,15};
  int *pos;
  int idx;
  pos=lower_bound(A,A+14,3);
  idx=distance(A,pos);
  cout<<"A["<<idx<<"]="<<*pos<<endl;//A[5]=4;
  pos=lower_bound(A,A+14,2);
  idx=distance(A,pos);
  cout<<"A["<<idx<<"]="<<*pos<<endl;//A[2]=2;
  return 0;
}

```

priority_queue

```cpp
#include<iostream>
#include<vector>
#include<queue>
using namespace std;
struct cmp{
  bool operator()(int x,int y){
    return x>y;
  }
};
struct node{
    int x,y;
    friend bool operator<(node a,node b)
    {
        return a.x>b.x;//按x从小到大排
    }
};
int main(){
  priority_queue<int> p;//默认最大的元素优先级最高
  priority_queue<int, vector<int>, cmp> q;    //定义方法
  priority_queue<node> q_node;
  return 0;
}
//empty()
//pop()
//push()
//size()
//top()

```

list

```cpp
#include <list>
using namespace std;
int main(){
  list<int> l;
  l.size();
  l.begin();
  l.end();
```

```
  l.push_front(x);
  l.push_back(x);
  l.pop_front();
  l.pop_back();
  l.insert(p,x);
  l.erase();
  l.clear();
  return 0;
}

```

map

```
#include <map>
#include <iostream>

struct  Test{
    int x;
    int y;
    bool operator < (const Test &o) const{
        return x < o.x || y < o.y;
    }
};
int main()
{
    std::map<Test, std::string> mapTest;
  int a=1,b=2;
    Test test = { a, b };
    mapTest[test] = "Test1";

    for (auto it = mapTest.begin(); it != mapTest.end();it++)
    {
        std::cout << it->first.x << " " << it->first.y << " " << it->second.c_str() << std::endl;
    }

    return 0;
}
//////////////////////////////////////////////////////////////////////////////
#include <iostream>
#include <map>
#include <string>
using namespace std;
void print(map<string, int> T){
  map<string, int>::iterator it;
  cout<<T.size()<<endl;
  for(it=T.begin();it!=T.end();it++){
    pair<string, int> itam=*it;
    cout<<itam.first<<"-->"<<itam.second<<endl;
  }
}
int main(){
  map<string, int> T;
  T["red"] = 32;
  T["blue"] = 688;
  T["yellow"] = 122;
  T["blue"] = 312;
  print(T);//3
          //blue-->312
          //red-->32
          //yellow-->122

  T.insert(make_pair("zebra", 101010));
  T.insert(make_pair("white", 0));
  T.erase("yellow");
  print(T);//4
          //blue-->312
          //red-->32
          //white-->0
          //zebra-->101010
  pair<string, int> target = *T.find("red");
```

```
  cout<<target.first<<"-->"<<target.second<<endl;
}
//size()
//clear()
//begin()
//end()
//insert(key,vel)//向map中插入元素（key， val）
//erase(key)//删除含有key的元素
//find(key)//搜索与key一致的元素，并返回该元素的迭代器，没有一致的元素，则返回末尾end（）

```

queue

```
#include <queue>
using namespace std;
int main(){
  queue<int> q;
  q.size();//返回队列的元素数
  q.front();//返回队头的元素
  q.pop();//从队列中取出并删除元素
  q.push(x);//从队列中添加元素x
  q.empty();//在队列为空时返回true
}

```

set

```
#include <iostream>
#include <set>
using namespace std;
void print(set<int> s){
  cout<<s.size()<<":";
  for(set<int>::iterator it=s.begin();it!=s.end();it++){
    cout<<" "<<(*it);
  }
  cout<<endl;
}
int main(){
  multiset<int,greater<int>> mul;
  s.insert(8);
  s.insert(1);
  s.insert(7);
  s.insert(4);
  s.insert(8);
  s.insert(4);

  print(s);//4: 1 4 7 8
  s.erase(7);
  print(s);//3: 1 4 8
  s.insert(2);
  print(s);//4: 1 2 4 8
  return 0;
}
// size()
// clear()
// begin()
// end()
// insert(key) 向set中添加元素key
// erase(key) 删除含key的元素
// find(key) 搜索与key一致的元素，并返回指向该元素的迭代器::若没有，则返回末尾end()

```

stack

```
#include <iostream>
#include <stack>
using namespace std;
int mian(){
  stack<int> S;
```

```
  S.size()//返回栈的元素数
  S.top()//返回栈顶元素
  S.pop()//从栈中取出并删除元素
  S.push(3)//向栈中添加元素x
  S.empty()//在栈为空时返回true
  //时间复杂度均为O(1)
}

```

string

```
/*
string s1;
默认构造函数，s1位空串
string s2(s1);  将s2初始化为s1的一个副本
string s3("value"); 将s3初始化为一个字符串字面值副本
string s4(n,'c')    将s4初始化为字符'c'的n个副本
/--------------------------------------------------------------------------------/
s.empty()   如果s为空串，则返回true，否则返回false
s.size()    返回s 中字符的字符个数
s[n]
返回s中位置为n的字符，位置从0开始计数

s1+s2   把s1和s2链接成一个新的字符串，返回新生成的字符串
s1=s2   把s1内容替换为s2的副本
v1==v2  判断v1与v2的内容，相等则返回true，否则返回false
!=, <, <=, >, >=    保持这些操作的惯有含义
/--------------------------------------------------------------------------------/
isalnum(c)  如果c是字母或数字，则为true
isalpha(c)  如果c是字母，则为true
isdigit(c)  如果c是数字，则为true
islower(c)  如果c是小写字母，则为ture
isupper(c)  如果c是大写字母，则为true
isspace(c)  如果c是空白字符，则为true
ispunct(c)  如果c是标点符号，则为true
iscntrl(c)  如果c是控制字符，则为true
isgraph(c)  如果c不是空格，但是可打印，则为true
isprint(c)  如果c是可打印的字符，则为true
isxdigit(c) 如果c是十六进制数，则为true
tolower(c)  如果c是大写字母，则返回其小写字母形式，否则直接返回c
toupper(c)  如果c是小写字母，则返回其大写字母形式，否则直接返回c
/--------------------------------------------------------------------------------/
string s(cp,n)  创建一个string对象，它被初始化为cp所指向数组的前n个元素副本
string s(s2,pos2)   创建一个string对象，它被初始化为一个已存在的string对象s2中从下标pos2开始的字符的副本
                如果pos2>s.size()则该操作未定义
string s(s2,pos2,len2)
创建一个string对象，它被初始化为s2中从下标pos2开始的len2个字符的副本
如果pos2>s2.size(),则该操作未定义
无论len2的值是多少，最多只能复制s2.size()-pos2个字符
/--------------------------------------------------------------------------------/
s.insert(p,t)   在迭代器p指向的元素之前插入一个值为t的新元素。返回指向新插入元素的迭代器
s.insert(p,n,t) 在迭代器p指向的元素之前插入n个值为t的新元素。返回void
s.insert(p,b,e) 在迭代器p指向的元素之前插入b和e标记范围内所有的元素，返回void
s.assign(b,e)   用迭代器b和e标记范围内的元素替换s。对于string类型，该操作返回s，对于容器类型，则返回void
s.assign(n,t)   用值为t的n个副本替换s。对于string类型，该操作返回s，对于容器类型，返回void
s.erase(p)  删除迭代器p指向的元素，返回一个迭代器，指向被删除元素后面的元素
s.erase(b,e)    删除迭代器b和e标记范围内所有的元素，返回一个迭代器，指向被删除元素段后面的第一个元素
reverse(b,e)    把迭代器b和e标记范围内的所有元素反转
/--------------------------------------------------------------------------------/
s.insert(pos,n,c)   在下表pos的元素之前插入n个字符c
s.insert(pos,s2)
在下标为pos的元素之前插入string对象s2的副本
s.insert(pos,s2,pos2,len)
在下标为pos的元素之前插入s2中从下标pos2开始len个字符
s.insert(pos,cp,len)    在下标为pos的元素之前插入s2中从下标pos2开始的len个字符
s.insert(pos,cp)    在下标为pos的元素之前插入cp所指向的以空字符结束的字符串副本
s.assign(s2)    用s2的副本替换s
s.assign(s2,pos2,len)   用s2中从下标pos2开始的len个字符副本替换s
s.assign(cp,len)    用cp所指向数组的前len个字符副本替换s
s.assign(cp)    用cp所指向的以空字符结束的字符串副本替换s
```

```
s.erase(pos,len)
删除从下标pos开始的len个字符
/---------------------------------------------------------------------------/
s.substr(pos,n) 返回一个string类型的字符串，它包含s中从下标pos开始的n个字符
s.substr(pos)   返回一个string类型的字符串，它包含从下标pos开始到s末为的所有字符
s.substr()  返回s的副本
/---------------------------------------------------------------------------/
s.find(args)    在s中查找args的第一次出现
s.rfind(args)   在s中查找args的最后一次出现
s.find_first_of(args)   在s中查args的任意字符的第一次出现
s.find_last_of(args)    在s中查找args的任意字符的最后一次出现
s.find_first_not_of(args)   在s中查找第一个不属于args的字符
s.find_last_not_of(args)    在s中查找最后一个不属于args的字符
/---------------------------------------------------------------------------/
s.compare(s2)   比较s和s2
s.compare(pos1,n1,s2)
让s中从pos下标位置开始的n1个字符与s2做比较
s.compare(pos1,n1,s2,pos2,n2)   让s中从pos1下标位置开始的n1个字符与s2中从pos2下标位置开始的n2个字符做比较
s.compare(cp)   比较s和cp所指向的以空字符结束的字符串
s.compare(pos1,n1,cp)   让s从pos1下标位置开始的n1个字符与cp所指向的字符串做比较
s.compare(pos1,n1,cp,n2)
让s中从pos1下标位置开始的n1个字符与cp所指向字符串的前n2个字符做比较
*/

```

vector

```
#include <vector>
using namespace std;
int main(){
  vector<int> v;
  v.size();//返回向量的元素数      O(1)
  v.push_back(x);//在向量末尾添加元素x      O(1)
  v.pop_back();//删除向量的最后一个元素      O(1)
  v.begin();//返回指向向量开头的迭代器      O(1)
  v.end();//返回指向向量末尾（最后一个元素的后一个位置）      O(1)
  v.insert(v.begin()+p,x);//在向量的位置p处插入元素x      O(n)
  v.erase(v.begin()+p);//删除向量中位置p的元素      O(n)
  v.clear();//删除向量中所有元素      O(n)
}

```

文章最后发布于: 2019-10-17 21:40:49

有 0 个人打赏

©2019 CSDN 皮肤主题: 大白 设计师: CSDN官方博客