万能c++头文件

```
#include<bits/stdc++.h>
#include <algorithm>
using namespace std;
```

```
#include<bits/stdc++.h>这个头文件包含以下等等C++中包含的所有头文件:
#include <iostream>
#include <cstdio>
#include <fstream>
#include <algorithm>
#include <cmath>
#include <deque>
#include <vector>
#include <queue>
#include <string>
#include <cstring>
#include <map>
#include <stack>
#include <set>
```

vector容器

```
#include<vector>
```

初始化

(1) vector a(10); //定义了10个整型元素的向量（尖括号中为元素类型名，它可以是任何合法的数据类型），但没有给出初值，其值是不确定的。

（2）vector a(10,1); //定义了10个整型元素的向量,且给出每个元素的初值为1

（3）vector a(b); //用b向量来创建a向量，整体复制性赋值

（4）vector a(b.begin(),b.begin+3); //定义了a值为b中第0个到第2个（共3个）元素

（5）int b[7]={1,2,3,4,5,9,8};

vector a(b,b+7); //从数组中获得初值

常用操作

（1）a.assign(b.begin(), b.begin()+3); //b为向量，将b的0~2个元素构成的向量赋给a

（2）a.assign(4,2); //是a只含4个元素，且每个元素为2

（3）a.back(); //返回a的最后一个元素

（4）a.front(); //返回a的第一个元素

（5）a[i]; //返回a的第i个元素，当且仅当a[i]存在2013-12-07

（6）a.clear(); //清空a中的元素

（7）a.empty(); //判断a是否为空，空则返回ture,不空则返回false

（8）a.pop_back(); //删除a向量的最后一个元素

（9）a.erase(a.begin()+1,a.begin()+3); //删除a中第1个（从第0个算起）到第2个元素，也就是说删除的元素从a.begin()+1算起（包括它）一直到a.begin()+ 3（不包括它）

（10）a.push_back(5); //在a的最后一个向量后插入一个元素，其值为5

（11）a.insert(a.begin()+1,5); //在a的第1个元素（从第0个算起）的位置插入数值5，如a为1,2,3,4，插入元素后为1,5,2,3,4

（12）a.insert(a.begin()+1,3,5); //在a的第1个元素（从第0个算起）的位置插入3个数，其值都为5

（13）a.insert(a.begin()+1,b+3,b+6); //b为数组，在a的第1个元素（从第0个算起）的位置插入b的第3个元素到第5个元素（不包括b+6），如b为1,2,3,4,5,9,8 ，插入元素后为1,4,5,9,2,3,4,5,9,8

（14）a.size(); //返回a中元素的个数；

（15）a.capacity(); //返回a在内存中总共可以容纳的元素个数

（16）a.resize(10); //将a的现有元素个数调至10个，多则删，少则补，其值随机

（17）a.resize(10,2); //将a的现有元素个数调至10个，多则删，少则补，其值为2

（18）a.reserve(100); //将a的容量（capacity）扩充至100，也就是说现在测试a.capacity();的时候返回值是100.这种操作只有在需要给a添加大量数据的时候才 显得有意义，因为这将避免内存多次容量扩充操作（当a的容量不足时电脑会自动扩容，当然这必然降低性能）

（19）a.swap(b); //b为向量，将a中的元素和b中的元素进行整体性交换

（20）a==b; //b为向量，向量的比较操作还有!=,>=,<=,>,<

（1）sort(a.begin(),a.end()); //对a中的从a.begin()（包括它）到a.end()（不包括它）的元素进行从小到大排列

（2）reverse(a.begin(),a.end()); //对a中的从a.begin()（包括它）到a.end()（不包括它）的元素倒置，但不排列，如a中元素为1,3,2,4,倒置后为4,2,3,1

（3）copy(a.begin(),a.end(),b.begin()+1); //把a中的从a.begin()（包括它）到a.end()（不包括它）的元素复制到b中，从b.begin()+1的位置（包括它）开 始复制，覆盖掉原有元素

（4）find(a.begin(),a.end(),10); //在a中的从a.begin()（包括它）到a.end()（不包括它）的元素中查找10，若存在返回其在向量中的位置

从向量中读取元素

1、通过下标方式读取

int a[6]={1,2,3,4,5,6};
vector b(a,a+4);
for(int i=0;i<=b.size()-1;i++)
cout<<b[i]<<" ";

2、通过遍历器方式读取

int a[6]={1,2,3,4,5,6};
vector b(a,a+4);
for(vector::iterator it=b.begin();it!=b.end();it++)
cout<<*it<<" ";

string标准库

常用函数

```
s.find()
s.refind()
s.substr()
s.empty()  //判断字符串是否为空,返回true或者false
s.size()   //返回字符串中字符的个数
s[n]    //返回字符串中的第n个字符,下标从0开始
s1+s2   //将s1和s2连接成一个新的字符串,返回新生成的字符串
s1=s2   //将s1的内容替换成s2的内容
v1==v2  //比较s1和s2的内容,判断其内容是否一样
!=,<,<=,>,>=  //保持这些操作符号惯有的特性
string s(str.rbegin(),str.rend());  //实现逆序
```

append追加

```
str1.append("hello"); //str1后面追加一个hello
str1.append(str2); //str1后面追加str2的字符
str1.append(10,"!"); //str1后面追加10个!
str1.append(str2,2,2);//从str2的第二位开始,添加2位,即添加

```

c_str()

```
返回一个指向正规C字符串的指针,内容与本字符串相同;
String str("helloworld");
char * char_str = str.c_str();//将字符串string转化成char*(字符指针)

```

find函数

```
size_type find( const basic_string &str, size_type index );
  size_type find( const char *str, size_type index );
  size_type find( const char *str, size_type index, size_type length );
  size_type find( char ch, size_type index );
  find()函数:
    返回str在字符串中第一次出现的位置（从index开始查找）。如果没找到则返回string:
    返回str在字符串中第一次出现的位置（从index开始查找，长度为length）。如果没找到
    返回字符ch在字符串中第一次出现的位置（从index开始查找）。如果没找到就返回strir
    如:
        string str1( "Alpha Beta Gamma Delta" );
        unsigned int loc = str1.find( "Omega", 0 );
        if( loc != string::npos ){
            cout<<loc<<endl;
        }

```

refind函数 反向查找

```
  size_type rfind( const basic_string &str, size_type index );
  size_type rfind( const char *str, size_type index );
  size_type rfind( const char *str, size_type index, size_type num );
  size_type rfind( char ch, size_type index );
1
2
3
4
rfind()函数:

返回最后一个与str中的某个字符匹配的字符，从index开始查找。如果没找到就返回string::
返回最后一个与str中的某个字符匹配的字符，从index开始查找,最多查找num个字符。如果没找
```

```
返回最后一个与ch匹配的字符，从index开始查找。如果没找到就返回string::npos
如:
int loc;
string s = "My cat's breath smells like cat food.";

loc = s.rfind( "breath", 8 );
if(loc != string::npos){

}

```

string的截取函数substr

```
string s("What we have here is a failure to communicate");
string sub = s.substr(10,5);


```

other

```
insert()
replace()
erase()
copy()
begin()
end()
```

不知道是否会用到的c头文件

```
#include <ctype.h>
#include <float.h>
#include <locale.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

标准库使用

```
#include<bits/stdc++.h>
#include<algorithm>
using namespace std;

vector<long> v_arr;
queue<long> q_arr;
map<string,int> m_arr;
set<int> set_arr;
stack<float> s_arr;

void test_vector();
void test_vector_with_class();
void test_string();

int main()
{
 //test_vector();
 //test_vector_with_class();
 test_string();
 return 0;
}

class item{
 public:
  int age;
  float height;
  string name;
  item(int a,float h, string n){
   age = a;
   height = h;
   name = n;
  }
};
void operator << (ostream &out, item &i)
{
 out<<i.age<<" "<<i.height<<" "<<i.name<<endl;
}
int operator < (item &i1, item &i2)
{
```

```
//按照age -> height -> name 的顺序排序
 if(i1.age<i2.age)
  return 1;
 else if(i1.age == i2.age)
  if(i1.height < i2.height)
   return 1;
  else if(i1.height == i2.height)
   if(i1.name <= i2.name)
    return 1;
   else
    return 0;
  else
   return 0;
 else
  return 0;
}
void test_vector()
{
 srand(123);
 for(int i=0;i<10;i++)
 {
  v_arr.push_back(rand()%100);
  q_arr.push(rand()%100+1000);
  s_arr.push((rand()%100)/100.0);
 }
 sort(v_arr.begin(),v_arr.end());
 for(int i=0;i<10;i++)
 {
  cout<<v_arr[i]<<endl;
  cout<<"\t"<<q_arr.front()<<endl;
  cout<<"\t\t"<<s_arr.top()<<endl;
  q_arr.pop();
  s_arr.pop();
 }
 string s="abc";
 m_arr[s] = 12;
 m_arr["abc"] = 11;
 cout<<m_arr[s]<<"map";



 set_arr.insert(2);
```

```
}
void test_vector_with_class()
{
 vector<item> v_class;
 v_class.push_back(item(12,12.2,"abc"));
 v_class.push_back(item(11,12.2,"abc"));
 v_class.push_back(item(13,12.2,"abc"));
 v_class.push_back(item(11,12.4,"abc"));
 v_class.push_back(item(11,12.2,"adc"));
 v_class.push_back(item(11,12.2,"abd"));
 sort(v_class.begin(),v_class.end());

 for(unsigned int i=0;i<v_class.size();i++)
  cout<<v_class[i];
 cout<<string::npos;
}

void test_string()
{
//find()
 string s = "#password:123456#ahaha";
 string p;
 cout<<s.find(':')<<endl;
 cout<<s.find('#')<<endl;
 cout<<s.find('@')<<endl;
//substr()
 if(s.find(":") != string::npos)
  p = s.substr(s.find(":")+1,s.size()-s.find(':'));
//erase()
//rfind()
 cout<<endl<<p<<endl;
 p.erase(p.rfind('h'));
 cout<<"after erase&&rfind:"<<p<<endl;
//append()
 p.append("000");
 p.append(6,'6');
 p.append(s,1,8);
 cout<<"after append:"<<p<<endl;
//c_str()
 const char *c_s = p.c_str();
 cout<<"c_str:"<<c_s<<endl;
}
```