问题

算法题当中对原始数据进行排序后，很大概率可以使得解题变得简单。一般情况下都是对 vector 等基本类型进行排序，这时直接使用 sort 函数即可，但是有时候我们想对自定义的结构体等类型进行排序，这时候直接调用sort就行不通了，需要另外定义一个排序规则，然后传给sort函数才行，具体怎么实现呢？

sort内部使用的是什么排序？

sort并不是简单的快速排序，它对普通的快速排序进行了优化，此外，它还结合了插入排序和堆排序。系统会根据你的数据形式和数据量自动选择合适的排序方法，这并不是说它每次排序只选择一种方法，它是在一次完整排序中不同的情况选用不同方法，比如给一个数据量较大的数组排序，开始采用快速排序，分段递归，分段之后每一段的数据量达到一个较小值后它就不继续往下递归，而是选择插入排序，如果递归的太深，他会选择堆排序。

sort的基本使用

要使用该函数需要先包含：`#include <algorithm>`
sort的函数原型为：`sort(first_pointer, first_pointer + n, cmp)`

参数1：第一个参数是数组的首地址，一般写上数组名就可以，因为数组名是一个指针常量。(左闭)
参数2：第二个参数相对较好理解，即首地址加上数组的长度n（代表尾地址的下一地址）。（右开）
参数3：默认可以不填，如果不填sort会默认按数组升序排序。也就是1,2,3,4排序。也可以自定义一个排序函数，改排序方式为降序什么的，也就是4,3,2,1这样。

下面给一个sort最常使用的场景：

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    vector<int> nums({13, 5, 3, 7, 43});
    sort(nums.begin(), nums.end());  // 默认升序排序
    for(auto i:nums) {
        cout<<i<<" ";
    }
    cout<<endl;  // 即输出：3 5 7 13 43
    return 0;
}
```

想要改为降序排序，简单这样改改就行：

```
#include <iostream>
#include <algorithm>
using namespace std;

//自定义一个降序排序函数
bool cmp(const int a, const int b) {
    return a > b;  // 前者大于后者返回true，因此为降序排序
}

```

```cpp
int main() {
    vector<int> nums({13, 5, 3, 7, 43});
    sort(nums.begin(), nums.end(), cmp);  // cmp函数作为第三个参数传进去即可
    for(auto i:nums) {
        cout<<i<<" ";
    }
    cout<<endl;
    return 0;
}
```

当然也可以给string排序：

```cpp
#include<iostream>
#include<algorithm>
#include<string>
using namespace std;

bool cmp(string a,string b)
{
    return a>b;
}
int main()
{
    string a[4]={"hhhhh","heheheh","xxxxxx","kkkkkk"};
    sort(a,a+4,cmp);
    for(int i=0;i<4;i++)
        cout<<a[i]<<endl;
    return 0;
}
```

用 sort 对结构体排序

我们可以使用 sort 实现对结构体的自定义条件的排序。

这里摘抄一个博客中的一个例子：实例化3只dog,并且按照先排公狗，后排母狗的规则排序。排序时先让年龄从大到小排序，如果年龄一样，再考按照体重从轻到重排。

```cpp
#include<iostream>
#include<algorithm>
#include<string>
using namespace std;
struct Dog{
    int age;
    int weight;
    string sex;
};
bool cmp(struct Dog a,struct Dog b)
{
    if(a.sex==b.sex){
        if(a.age==b.age){
            return a.weight<b.weight;//体重升序
        }
        else return a.age>b.age;//年龄降序
    }
    else return a.sex>b.sex;//性别字典序降序
}
```

```
int main()
{
    Dog dog[3];
    dog[0].sex="male";
    dog[0].age=3;
    dog[0].weight=20;

    dog[1].sex="male";
    dog[1].age=3;
    dog[1].weight=15;

    dog[2].sex="male";
    dog[2].age=3;
    dog[2].weight=44;

    sort(dog,dog+3,cmp);
    for(int i=0;i<3;i++){
        cout<<dog[i].sex<<" "<<dog[i].age<<" "<<dog[i].weight<<endl;
    }
    return 0;
}
```

除了这种在结构体外定义比较函数外，还可以使用在结构体内重载 `<` 运算符的方法**（注意这里重载的只能是 `<` 运算符，因为 sort 函数内部默认是降序，用的就是 `<` 运算符）**。

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
typedef struct student{
    char  name[20];
    int math;
    // 重载 < 运算符
    bool operator < (const student &x){
        return math > x.math ;
    }
}Student;
int main(){
    Student a[4]={{"apple",67},{"limei",90},{"apple",90}};
    sort(a,a+3);  // 在结构体内重载了 < 运算符后，就可以跟vector等类型一样只传入两个参数即可。
    for(int i=0;i<3;i++)
        cout<<a[i].name <<" "<<a[i].math <<" " <<endl;
    return 0;
}
```

By Yee

2020.05.14