2021 算法竞赛新生培训

STL 标准模板库简介

应物 200 曾剑涛

2021 年 10 月 26 日

前置知识-数组

你需要定义 10 **个** int **变量?**

```
int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;
```

使用数组。

```
int a[10];
```

前置知识-数组

int a[10];
定义了一个长度为 10 的数组，包含 10 个 int 类型的元素 a[0], a[1], a[2],...., a[9]
注意下标是从 0 开始的。
可以用普通变量一样使用数组的元素。比如：
scanf("%d", &a[1]);//读入一个整数，存到 a[1]
printf("%d", a[2]);//输出变量 a[2] 的值
a[3] = a[1] + a[2]; //将 a[1] 和 a[2] 的值加起来赋给 a[3]

前置知识-数组

数组的大小不能为变量，只能为常数。以下的写法是不正确的。(虽然它能通过编译，有时候也能“正常”运行，但是请不要这么做)

```
int n = 10;
int a[n];
```

在算法竞赛中，一般直接开一个足够大的数组。

上次的作业：询问学号

https://www.luogu.com.cn/problem/P3156

```
1  #include <stdio.h>
2  #define N 2000010
3  int a[N];
4  int main() {
5      int n, m, i, x;
6      scanf("%d%d", &n, &m);
7      for(i = 1; i <= n; i++) {
8          scanf("%d", &a[i]);
9      }
10     while(m--) {
11         scanf("%d", &x);
12         printf("%d\n", a[x]);
13     }
14     return 0;
```

```
15 }
```

前置知识-函数

如果一段代码需要多次被使用，是不是每次使用时都要写一遍呢?

可以把一段代码写成一个函数，每次需要使用这段代码时，只需要直接调用。

函数的结构:

```
返回值类型 函数名() {
    函数体;
    return 返回值;
}
```

函数使用示例

```
1  #include <stdio.h>
2  void display() {
3      printf("---------\n");
4      printf("haha\n");
5      printf("hihi\n");
6      printf("hahahihi\n");
7      printf("---------\n");
8  }
9  int main() {
10     display();
11     display();
12     return 0;
13 }
```

函数使用示例-参数和返回值

```
1  int gcd(int a,int b) {//计算a和b的最大公约数
2      int r;
3      while(b) {
4          r = a % b;
5          a = b;
6          b = r;
7      }
8      return a;
9  }
10 int main() {
11     printf("%d\n", gcd(3, 5));
12     printf("%d\n", gcd(12, 16));
13     return 0;
14 }
```

STL 简介

标准模板库（Standard Template Library，STL）。是一些前辈已经写好的常用代码，包含了常用的算法、容器、数据结构等。使用时只需调用即可，不用自己再写一遍。
STL 的代码从广义上讲分为三类：algorithm（算法）、container（容器）和 iterator（迭代器），本次课主要介绍常用的算法和容器。

algorithm

要使用 STL 中的算法，要先包含头文件 algorihm 并使用命名空间 std

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

排序

用法:

```
1 int a[] = {3, 1, 4, 2, 5};
2 sort(&a[0], &a[5]); // 将数组a升序排序
3 sort(&a[0], &a[5], greater<int>()); //将数组b降序排序
```

sort(a[st], a[ed]); 表示将数组 a 中的元素 a[st], a[st+1], ..., a[ed-1] 排序。注意不含 a[ed], [st, ed) 是一个**左闭右开**区间。
例如，要将一个长度为 n 的数组 a 所有元素进行排序，应该使用 sort(&a[0], &a[n]);
也可写成 sort(a, a + n);

sort 的时间复杂度

sort 使用的是快速排序，时间复杂度为 $O(nlogn)$，在 1 秒内大约可对 $10^5 \sim 10^6$ 个元素进行排序。

例题 https://www.luogu.com.cn/problem/P1177

二分查找

STL 提供了两个二分查找函数，lower_bound 和 upper_bound。用法如下:

```
1
2 lower_bound(&a[st], &a[ed], value);
3 /*
4 [st, ed)是查询的区间，左闭右开区间(和sort类似)，value是
      要查询的值。
5 该函数的反回值是一个指针，需要减去&a[0]，得到查询位置的
      下标。
6 upper_bound和lower_bound使用方法相同，不同点在于，
      lower_bound查询的是大于等于value的第一个数，
      upper_bound查询的是大于value的第一个数。
7 */
```

二分查找-示例

```
1 int n = 10;
2 int a[10] = {1, 3, 4, 5, 5, 5, 6, 7, 10, 11};
3 int l = lower_bound(&a[0], &a[n], 5) - &a[0];//查询第
    一个大于等于5的数 (a[3] = 5)
4 int r = upper_bound(&a[0], &a[n], 5) - &a[0];//查询第
    一个大于5的数(a[6] = 6 > 5)
5 printf("%d %d\n", l, r); //输出3 6
6 //lower_bound和upper_bound得到一个左闭右开区间[l, r),
    这个区间内的值都等于value
```

例题:https://www.luogu.com.cn/problem/P2249

vector

前面我们学习的数组，数组的长度只能是一个定值，不能为变量。有时候需要用到长度不是定值的数组，或都数组的长度在使用过程中需要变化，就可以使用 vector. 需要包含头文件 vector
vector: 不定长数组。

```
1 //创建vector:
2 //1. 创建一个空的vector, 长度为0
3 vector<int> a;
4 //2. 创建一个长度为n的vector, 初始值全部为0
5 vector<int> a(n);
6 //3. 创建一个长度为n的vector, 初始值全部为v
7 vector<int> a(n, v);
```

使用 vector

```
1  //1. 获取vector的长度
2  int n = a.size();
3  //2. 使用vector中的元素
4  //像普通数组一样，使用方括号[]，下标也是从零开始,a[0],
       a[1], a[2], ..., a[n - 1]
5  //3. 遍历vector，和普通数组一样
6  for(int i = 0; i < n; i++) {
7      printf("%d ", a[i]);
8  }
9  //4. 用迭代器遍历vector
10 for(auto it = a.begin(); it != a.end(); it++) {
11     printf("%d ", *it);
12 }
```

对 vector 使用 algorithm

```
1  //1. sort
2  sort(a.begin(), a.end()); //升序
3  sort(a.begin(), a.end(), greater<int>()); //降序
4
5  //2. lower_bound
6  int l = lower_bound(a.begin(), a.end(), value) - a.
       begin();
```

vector 的特有功能

```
1 //1. 用vector最后添加一个元素v
2 a.push_back(v);
3 //2. 删除vector的最后一个元素
4 a.pop_back();
5 //3. 调整vector的大小，(如果将vector调小，会将多余的元
      素删除，如果调大，会用0填充，也可用指定的值填充)
6 a.resize(n);
```

*vector 的实现原理

倍增

set

set: 集合，set 中的元素会自动按顺序排列，set 中不允许有相等的元素。使用:

```
1  //1. 创建一个set
2  set<int> S;
3  //2. 向set中加入元素
4  S.insert(2);
5  S.insert(4);
6  S.insert(4); // 再次插入4， 4并不会两次加入到set中
7  //不管插入的顺序如何，set中的元素始终按从小到大的顺序排
       列
8  //3. 删除set中的元素
9  S.erase(4); // S中只有1个4， 将4闪除，S中就没有4了
10 //4. 查找集合中是否有某个元素
11 S.count(4); // 如果S中有4,返回1， 否则返回0
```

遍历 set

set 只能使用迭代器遍历

```
1 for(auto it = S.begin(); it != S.end(); it++) {
2     printf("%d ", *it);
3 }
```

映射 map

map 是一种特殊的容器，存储的是“键”-“值”对的映射关系（类似 Python 中的字典）。map 中存储的键值对自动按键排序。

```
1  //创建一个map,
2  map<int, int> mp;//创建了一个键和值的类型都是int的map
3  //插入一个键值对
4  mp.insert(make_pair(1, 100)); //插入了一个映射关系 1
       -> 100
5  //由键查询值，像访问数组元素一样
6  printf("%d\n", mp[1]); // 输出100
7  //也可以你数组元素一样插入映射
8  mp[2] = 200; //相当于mp.insert(make_pair(2, 200));
9  // 插入重复的键值，会覆盖原来的
10 mp[1] = 1000; // 原先的1->100 变成了 1->1000
11 //
```

映射 map

```
1 //查询map中是否存在某个键
2 mp.count(key); // 如果key存在，返回1， 否则返回0
3 //删除一个键
4 mp.erase(key);
5 //遍历map中所有的键值对
6 for(auto it = mp.begin(); it != mp.end(); it++) {
7     printf("%d %d\n", (*it).first, (*it).second);
8 }
```

栈 stack

栈是一种先进后出 (FILO) 的数据结构。可以把它看成一个一端开口的容器，第一个放进去的容器在容器底部。要取出元素时，只能取出顶部的元素。

```
1 //创建一个栈
2 stack<int> st;
3 //向栈顶添加元素
4 st.push(2);
5 st.push(100);
6 //获取栈顶元素
7 printf("%d\n", st.top());
8 //移除(弹出)栈顶元素
9 st.pop();
```

如何判断一个人是不是程序员?

问他 push 的反义词是什么。

如何判断一个人是不是程序员?

问他 push 的反义词是什么。
pull: 非程序员
pop: 程序员

队列 queue

队列是一种先进先出 (FIFO) 的数据结构。类似排队打饭，先进入队列的人，应当先打到饭并离开队列。可以看作一个两端开口的容器，一端进入元素，一端弹出元素。

```
1 //创建一个队列
2 queue<int> Q;
3 //向队尾添加元素
4 Q.push(2);
5 Q.push(100);
6 //获取队头元素
7 printf("%d\n", Q.front());
8 //弹出队头元素
9 Q.pop();
```

优先队列 priority_queue

和队列类似，但是，优先队列会把最大的元素放到顶部。每次出队时，弹出的是最大的元素。

```
1  //创建一个优先队列
2  priority_queue<int> Q;
3  //向优先队列中插入元素
4  Q.push(2);
5  Q.push(100);
6  //获取优先队列中最大的元素
7  printf("%d\n", Q.top());//虽然2比100先入队，但由于100
       是最大元素，所以输出的是100。注意是top不是front
8  //弹出最大元素
9  Q.pop();
10
11 //如何创建一个小根堆（即每次出队最小元素）？
```

容器总结

	大小	加入元素	获取元素	删除元素	查询元素是否存在
数组	固定值	不支持	a[index]	不支持	
vector	size()	push_back(v) 在尾部插入	a[index]或 a.at(index)	pop_back() 删除最后一个元素	
set	size()	insert(v)	迭代器	erase(value)	count(value)
map	size()	insert(make_pair(key, value)) 或 mp[key] = value	mp[key]	erase(key)	count(key)
stack	size()	push(v) 在栈顶插入	st.top() 获取栈顶元素	st.pop() 弹出栈顶元素	
queue	size()	push(v) 在队尾插入	Q.front() 获取队头元素	Q.pop() 弹出队头元素	
priority_queue	size()	push(v)	Q.top() 获取最大元素	Q.pop() 弹出最大元素	

作业

https://www.luogu.com.cn/contest/55131

现在是答疑时间。