队列

队列是一种特殊的数据结构，简单而言就是一个只能在一个端口插入元素，并且只能在另一个端口排出元素的线性数据结构，很像现实生活中的排队。

STL中的队列

STL中有专门对应于队列的数据结构queue。其主要可以进行如下操作：

1. push 【入队插到队尾】
2. pop 【队首元素出队】
3. size 【返回队列中元素的个数】
4. front 【返回队列中第一个元素】
5. back 【返回队列中最后一个元素】
6. empty 【判断队列是否为空】

举个例子：

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

int main(int argc, char const *argv[])
{
    int n=10,num;
    queue<int> a;
    for(int i=1;i<n;i++){
        num = rand()%233;
        a.push(num);
    }

    //数列长度
    cout<<a.size()<<endl;
    //数列头元素
    cout<<a.front()<<endl;
    //数列尾元素
    cout<<a.back()<<endl;
    //数列是否为空
    while(!a.empty()){
        cout<<a.front()<<ends;
        a.pop();
```

```
    }cout<<endl<<endl;

    return 0;
}
```

队列实现

通常用一个数组模拟一个队列，用两个指针：front 和 rear 分别表示队列头部和尾部。

在入队的时候将 rear 后移，在出队的时候将 front 后移。

下面我们来实现一个手动队列。它可以根据我们的输入插入或者排出元素，告诉我们队列是否为空，查询队头元素是什么，或者告诉我们队列的尺寸是多少。

```
#include <iostream>

using namespace std;

const int N=100010;

int m;
int q[N], hh, tt = -1;

int main()
{
    cin >> m;

    while (m -- )
    {
        string op;
        int x;

        cin>>op;
        if (op=="push")
        {
            cin>>x;
            q[++tt]=x;
        }
        else if (op=="pop") hh ++ ;
        else if (op=="empty") cout << (hh <= tt ? "NO" : "YES") << endl;
        else cout << q[hh] << endl;
    }
```

```
    return 0;
}
```

优先队列

每个元素具有一定优先级的队列就叫做优先队列（但是其本质是一个堆）。
试想一下，你排队买饭，正排着队，突然来了一个社会小霸王，很不讲理，非要插队，你又揍不过他，这时候他的优先级比你高，排在你的前面。但是还没有完！突然有一个人拿着军官证，为满足军人优先政策，他的优先级比所有人都高，于是，他排在你和小霸王的前面。

STL也有优先队列专门的库。它可以执行以下操作：

```
top() : 访问栈顶元素 常数复杂度
empty() : 检查底层的容器是否为空 常数复杂度
size() : 返回底层容器的元素数量 常数复杂度
push() : 插入元素，并对底层容器排序 最坏 O(n)
pop():排除一个元素，复杂度最坏O(log(n))
```

模板的具体内容需要我们在之后多多使用熟悉。但是现在我们可以先看一看优先队列的例题来熟悉它的内容：
132. 小组队列 - AcWing题库

一种使用优先队列的办法如下：

```
#include <queue>
#include <cstdio>
#include <iostream>
using namespace std;
const int N = 1000000, T = 1006;
int t, f[N], id = 0;
char s[10];
queue<int> q[T];

void Team_Queue() {
    q[0] = queue<int>();
    for (int i = 1; i <= t; i++) {
        int n;
        scanf("%d", &n);
        while (n--) {
            int x;
            scanf("%d", &x);
```

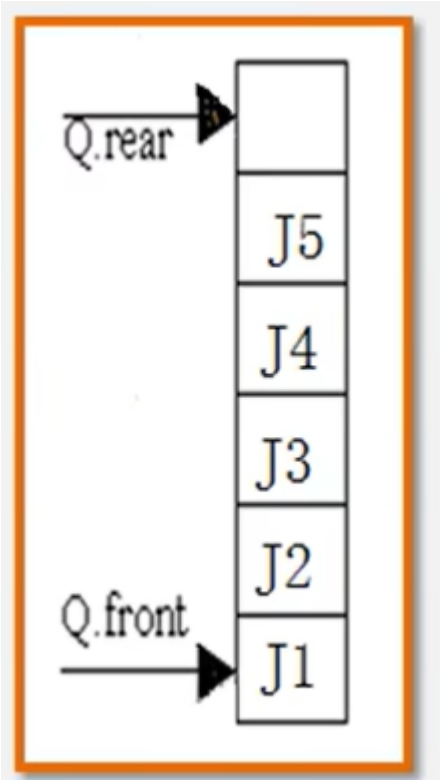
```
            f[x] = i;
        }
        q[i] = queue<int>();
    }
    cout << "Scenario #" << ++id << endl;
    while (scanf("%s", s) && s[0] != 'S') {
        if (s[0] == 'E') {
            int x;
            scanf("%d", &x);
            if (q[f[x]].empty()) q[0].push(f[x]);
            q[f[x]].push(x);
        } else {
            int x = q[0].front();
            printf("%d\n", q[x].front());
            q[x].pop();
            if (q[x].empty()) q[0].pop();
        }
    }
    cout << endl;
}

int main() {
    while (cin >> t && t) Team_Queue();
    return 0;
}
```

循环队列

循环队列一般不会在算法竞赛中出现，但是在数据结构课程的考试中是重点考察内容。在22级的数据结构期末考试中过半同学在循环队列的试题上丢分严重，故虽然不涉及编码知识，还是要了解下循环队列的内核。

（1）由于队列的是受限性线性表，一段进，一段出，因此在操作过程中可能会产生假溢出现象而不能充分利用内存。这时可以使用循环队列充分使用内存。

(2) 

(3) 循环队列求队列长度：

```
(rear-front+maxsize)%maxsize;
```

(4) 循环队列插入元素：首先判重：

```
if((q.rear+1)%maxsize== q.front) return false;
```

此时操作失败；由于要解决假溢出问题，操作转换为如下：

```
q.[ rear ]=x; rear=（rear+1）%maxsize;
```

(5) 循环队列删除元素：直接令头指针

```
front=(front+1)%maxsize;
```

(6) 循环队列出队：首先判断是否队空：

```
if(q.front== q.rear)
return false;
```

（7）循环队列除了上述的方法，还有另一种就是省出来一个空间不用。于是最后判断队列为满的操作变为：front== (rear+1)%maxsize；

单调队列

单调队列需要满足两个性质：

- 队列内具有一定的单调性（最简单的，例如元素递增或者递减）。
- 满足普通队列性质，一端进，另一端出，不可以中间插队。
  但是这样就会现矛盾了，例如一个单调增的队列：$1,5,8,9$，我们要插入4，这时如果只能从尾端进去的话就打破了其单调性。那么这时的做法，就是从队尾到队头把大于4的全部排出队伍，然后插入后的队列就变成了$1,4$。

我们来看这道例题：

给定一个大小为 $n \leq 10^6$ 的数组。

有一个大小为 k 的滑动窗口，它从数组的最左边移动到最右边。

你只能在窗口中看到 k 个数字。

每次滑动窗口向右移动一个位置。

以下是一个例子：

该数组为 `[1 3 -1 -3 5 3 6 7]`，k 为 3。

窗口位置	最小值	最大值
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

你的任务是确定滑动窗口位于每个位置时，窗口中的最大值和最小值。

最小值和最大值分开来做，两个for循环完全类似，都做以下四步：

- 解决队首已经出窗口的问题;
- 解决队尾与当前元素$a[i]$不满足单调性的问题;
- 将当前元素下标加入队尾;
- 如果满足条件则输出结果;

同时注意下以下细节：

- 上面四个步骤中一定要先3后4，因为有可能输出的正是新加入的那个元素;
- 算最大值前注意将hh和tt重置;
- hh从0开始，数组下标也要从0开始。

```
# include <iostream>
using namespace std;
const int N = 1000010;
int a[N], q[N], hh, tt = -1;

int main()
{
    int n, k;
    scanf("%d%d",&n,&k);
    for (int i = 0; i < n; ++ i)
    {
        scanf("%d", &a[i]);
        if (i - k + 1 > q[hh]) ++ hh;                   // 若队首出窗口，hh加1
        while (hh <= tt && a[i] <= a[q[tt]]) -- tt;     // 若队尾不单调，tt减1
        q[++ tt] = i;                                   // 下标加到队尾
        if (i + 1 >= k) printf("%d ", a[q[hh]]);        // 输出结果
    }
    printf("\n");

    hh = 0; tt = -1;                                    // 重置!
    for (int i = 0; i < n; ++ i)
    {
        if (i - k + 1 > q[hh]) ++ hh;
        while (hh <= tt && a[i] >= a[q[tt]]) -- tt;
        q[++ tt] = i;
        if (i + 1 >= k) printf("%d ", a[q[hh]]);
    }
```

```
    return 0;
}
```