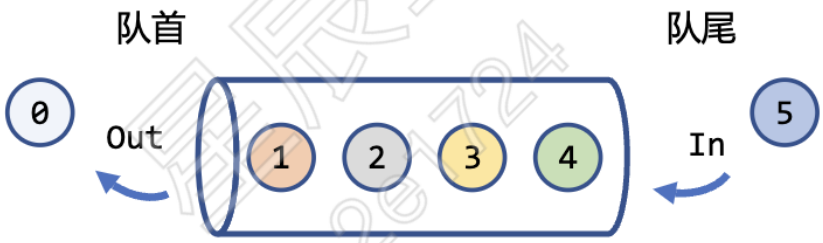
# 第十八章 队列与STL队列

## 1. 队列

### 1. 队列的定义及模拟

队列：只允许在一端进行插入数据操作，在另一端进行删除或访问操作的线性数据结构；
队列具有先进先出的特点（First In First out）

> 约定只能从数据的一端进行入队操作，即添加数据，这一端称作队尾；
>
> 对应另一端进行出队、访问操作，这一端称作队首。




- 队列的模拟

使用静态数组模拟队列及队列的相关操作

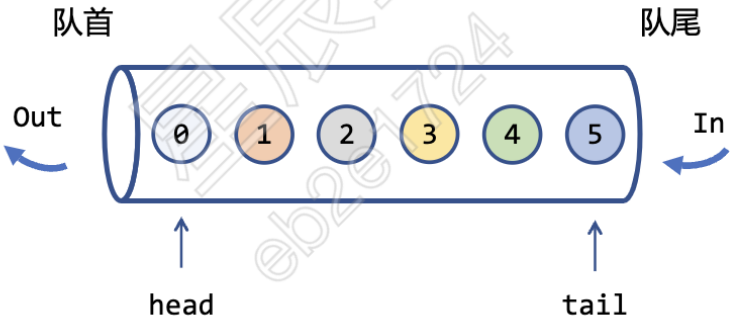
1. 导入相关头文件；
2. 定义静态数组，表示队列；
3. 定义保存的数据 x、队首 head、队尾 tail。

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

const int max_size = 1001;
int queue[max_size];
// head 队首 tail 队尾
int x, head = 0, tail = -1;
```

- 判空

  队列的元素个数为 (tail - head + 1) ，当 tail < head 时，tail - head + 1 为负，元素个数不会为负，此时队列为空队列。这也是队列创建时，为什么 head = 0，tail = -1，对应判空条件可为：tail + 1 == head。




判空操作：

1. 如果 tail 的下一个位置是 head 的位置，队列为空；
2. 否则不为空。

```
// 判空
bool empty(){
    // 1 表示队列为空，0 表示队列不为空
    return tail + 1 == head;
}
```

- 入队

入队操作：

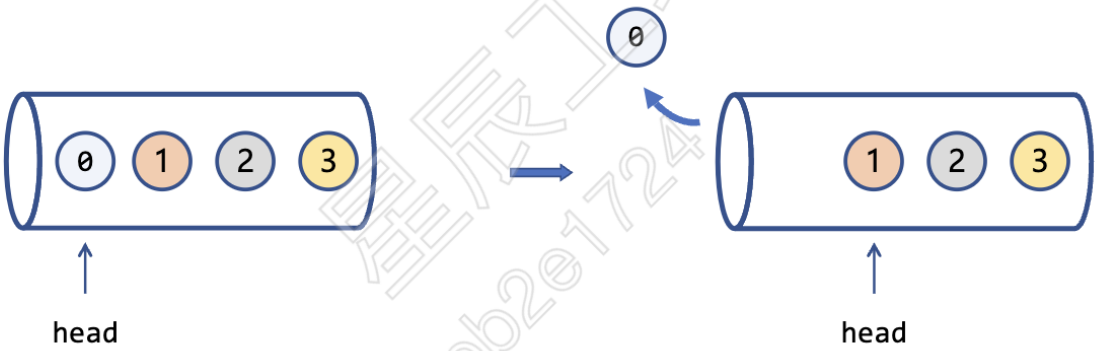
1. 队满时进行提示；
2. 否则，tail 移向下一个位置，元素入队。

注：使用数组模拟队列，注意队尾溢出问题，队尾不能超出初始定义的数组大小。

```
// 入队
void push(int x){
    if (tail == max_size - 1)
        cout << "队列已满" << endl;
    else
        queue[++ tail] = x;
}
```

- 出队

  静态数组模拟出队，可将 head + 1，head 指向数组下一个位置，队首对应变成下一个元素，模拟出队过程。



出队操作：

1. 队空时进行提示；
2. 否则，head 移向下一个位置，表示元素出队。

```
// 出队
void pop(){
    if (empty())
        cout << "队列为空" << endl;
    else
        head ++;
}
```

- 遍历
  从队首 head 开始访问，当head==tai1时，为访问的最后一个元素，打印完该元素后，结束访问，以此模拟队列的遍历过程

遍历操作：

1. 队空时进行提示；
2. 否则，打印 head 指向的每一个元素，head 移动，直至队列为空。

```
// 遍历
void print(){
    if (empty())
    cout << "队列为空" << endl;
    else{
        while (!empty())
            cout << queue[head++] << " ";
        cout << endl;
    }
}
```

- 优化
  使用数组模拟队列会导致一个问题：随着整个队列向数组的尾部移动，一旦到达数组的最末端，即使数组的前端还有空闲位置，再进行入队操作也会导致溢出 （这种数组里实际有空闲位置而发生了上溢的现象被称为假溢出）前面的程序中加入了队列已满的判断，并不能避免这种空闲位置被浪费的情况解决假溢出的办法是通过取余、利用循环的方式来存放队列元素 （数组下标为x的元素， 它移动的下一个位置为（+1）%max_size）。这样就形成了循环队列

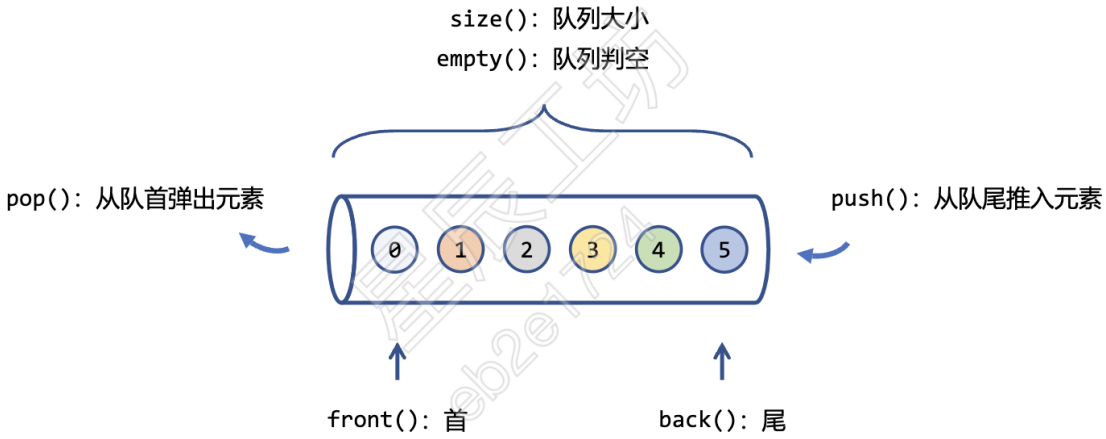
## 2. STL队列

### 2.1队列的声明和常用方法

- 队列的声明

1. 导入相关头文件 `#inlcude<queue>`
2. 格式： quene<数据类型> 队列名称 后续以int类型举例 `queue<int> q;`

- 队列的常用方法




| | |
|---|---|
| front() | 返回队列中第一个元素的引用 |
| back() | 返回队列中最后一个元素的引用 |
| push() | 在队列的尾部添加一个元素 |
| pop() | 删除队列中的第一个元素 |
| size() | 返回队列中元素的个数 |
| empty() | 判断队列中是否有元素，若无元素，则返回 true；反之，返回 false |
| emplace() | 在队列的尾部生成一个元素 |
| swap(q, other_q) | 将当前队列中的元素和参数队列中的元素交换。它们需要包含相同类型的元素 |

## 例题：

题目名称： 机器翻译

小晨的电脑上安装了一个机器翻译软件，他经常用这个软件来翻译英语文章。这个翻译软件的原理很简单，它只是从头到尾，依次将每个英文单词用对应的中文含义来替换。对于每个英文单词， 软件会先在内存中查找这个单词的中文含义，如果内存中有，软件就会用它进行翻译，如果内存中没有，软件就会在外存中的词典查找，查出单词的中文含义然后翻译，并将这个单词和译义放入内存，以备后续的查找和翻译。假设内存中有m个单元，每单元能存放一个单词和译义。每当软件将一个新单词存入内存前，如果当前内存中已存入的单词数不超过m-1，软件会将新单词存入一个未使用的内存单元，若内存中已存入m个单词，软件会清空最早进入内存的那个单词，用腾出单元来，存放新单词。假设一篇英语文章的长度为n个单词。 给定这篇待译文章，翻译软件需要去外存查找多少次词典？假设在翻译开始前，内存中没有任何单词

**输入格式**

共两行，整数之间以一个空格隔开。

第一行包含两个正整数 m、n（1≤m≤100，1≤n≤1000），表示内存容量和文章的单词数；

第二行包含 n 个正整数（1≤正整数≤1000），按照文章的顺序，每个数代表一个英文单词，相同单词用同一个正整数表示。

**输出格式**

一个整数，表示需要外查词典的次数。

**输入样例**

```
3 7
1 2 1 5 4 4 1
```

**输出样例**

```
5
```