第二题游戏

一、问题分析

【问题描述】

基于 STL 实现以下功能：有 n 个小朋友围成一圈玩游戏，小朋友从 1 至 n 编号，2 号小朋友坐在 1 号小朋友的顺时针方向，3 号小朋友坐在 2 号小朋友的顺时针方向，……，1 号小朋友坐在 n 号小朋友的顺时针方向。

游戏开始，从 1 号小朋友开始顺时针报数，接下来每个小朋友的报数是上一个小朋友报的数加 1。若一个小朋友报的数为 k 的倍数或其末位数（即数的个位）为 k，则该小朋友被淘汰出局，不再参加以后的报数。当游戏中只剩下一个小朋友时，该小朋友获胜。

例如，当 n=5, k=2 时：

1 号小朋友报数 1；

2 号小朋友报数 2 淘汰；

3 号小朋友报数 3；

4 号小朋友报数 4 淘汰；

5 号小朋友报数 5；

1 号小朋友报数 6 淘汰；

3 号小朋友报数 7；

5 号小朋友报数 8 淘汰；

3 号小朋友获胜。

给定 n 和 k，请问最后获胜的小朋友编号为多少？

【输入形式】

输入一行，包括两个整数 n 和 k，意义如题目所述。

【输出形式】

输出一行，包含一个整数，表示获胜的小朋友编号。

本题要处理的对象是 n 个结点，要实现的功能是从 n 个结点中按照某种顺序删除某些结点，直到剩下一个结点。通过调用函数将剩下的结点的编号输出。

1. 求解方法

从表头开始，遍历 list 中的每个结点，当循环变量 i 为 *k* 的倍数或其末位数（即数的个位）为 *k* 时，将结点从 list 中删除，当迭代器来到表尾时，再将其移至表头重新遍历，直到 list 中只剩下一个结点，for 循环结束，输出剩下的结点的 id。

2. 样例推导

样例 1：

n=5,k=2

1 号小朋友报数 1；

2 号小朋友报数 2 淘汰；

3 号小朋友报数 3；

4 号小朋友报数 4 淘汰；

5 号小朋友报数 5；

回到 1 号小朋友；

1 号小朋友报数 6 淘汰；

3 号小朋友报数 7；

5 号小朋友报数 8 淘汰；
只剩 3 号，结束，3 号小朋友获胜。
样例 2：
n=7,k=3
1 号小朋友报数 1；
2 号小朋友报数 2；
3 号小朋友报数 3 淘汰；
4 号小朋友报数 4；
5 号小朋友报数 5；
6 号小朋友报数 6 淘汰；
7 号小朋友报数 7；
回到 1 号小朋友；
1 号小朋友报数 8；
2 号小朋友报数 9 淘汰；
4 号小朋友报数 10；
5 号小朋友报数 11；
7 号小朋友报数 12 淘汰；
回到 1 号小朋友；
1 号小朋友报数 13 淘汰；
4 号小朋友报数 14；
5 号小朋友报数 15 淘汰；
只剩 4 号，结束。

二、数据结构和算法设计

1. 抽象数据类型设计

要处理的数据对象为一组结点，线性结构，构成循环，故可选择 STL 中的 list。

- 数据对象：list 是一种序列式容器。list 容器完成的功能实际上和数据结构中的双向链表是极其相似的，list 中的数据元素是通过链表指针串连成逻辑意义上的线性表，也就是 list 也具有链表的主要优点，即：在链表的任一位置进行元素的插入、删除操作都是快速的。list 的实现大概是这样的：list 的每个节点有三个域：前驱元素指针域、数据域和后继元素指针域。前驱元素指针域保存了前驱元素的首地址；数据域则是本节点的数据；后继元素指针域则保存了后继元素的首地址。其实，list 和循环链表也有相似的地方，即：头节点的前驱元素指针域保存的是链表中尾元素的首地址，list 的尾节点的后继元素指针域则保存了头节点的首地址，这样，list 实际上就构成了一个双向循环链。
- 数据关系：这种结构具有下列特点：每一个数据元素均有一个直接前驱和一个直接后继数据元素，其中表尾元素的后继元素为表头元素，表头元素的前驱元素为表尾元素。

2. 算法思想的设计

从表头开始，遍历 list 中的每个结点，当循环变量 i 为 *k* 的倍数或其末位数（即数的个位）为 *k* 时，将结点从 list 中删除，当迭代器来到表尾时，再将其移至表头重新遍历，直到 list 中只剩下一个结点，for 循环结束，输出剩下的结点的 id。

3. 关键功能的算法步骤

1) 数据输入部分

```
for(int i=1;i<=n;i++) //将数据填入 list 中
{
    Node* t=new Node(i); //构造一个 id 为 i 的结点
    l.push_back(*t); //将结点存到 list 末尾
}
```

2) 数据处理部分

```
it=l.begin(); //从第一个人开始
for(int i=1;l.size()>1;i++) //依次报数直到只剩一人
{
    if(i%k==0||i%10==k) it=l.erase(it); //这个人被淘汰
    else it++; //下一个人报数
    if(it==l.end()) it=l.begin(); //如果最后一个人报数完则回到第一个人
}
```

三、算法性能分析

1. 数据输入部分

for 循环共执行 n 次，所以时间复杂度为 O(n),空间复杂度为 O(n)。

2. 数据处理部分

for 循环共执行 n 次，所以时间复杂度为 O(n),空间复杂度为 O(1)。