C++方向编程题答案

第四周

day21

> **题目ID：46126-洗牌**
>
> 链接：https://www.nowcoder.com/practice/5a0a2c7e431e4fbbbb1ff32ac6e8dfa0?tpId=85&&tqId=29848&rp=1&ru=/activity/oj&qru=/ta/2017test/question-ranking
>
> **【题目解析】：**
>
> 本题题目的意思是模拟洗牌过程，牌被分成两组，且每组数量相等，然后每组牌从最后一个牌交叉排列，最后再把排列逆序就是一次洗牌的过程，K次需要重复K次这样的过程。
>
> **【解题思路】：**
>
> 每次读取一个数之后，算出他经过k次洗牌后的位置，只用一个长度为2n数组用来输出
>
> 根据当前数的位置，可以算出经过一次洗牌后的位置
>
> 如果当前数小于等于n（即在左手），则他下次出现的位置是 2*当前位置
>
> 与之对应的当前位置 + n（即在右手）的牌,则他下次出现的位置是 2*当前位置 + 1
>
> **【示例代码】**

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

int main()
{
    int T, n, k;
    cin >> T;
    while (T--)
    {
        cin >> n >> k;
        int num = 2 * n;
        vector<int> table(num);
        for (int i = 0; i < num; ++i)
            cin >> table[i];
  //洗k次牌
        while (k--)
        {
            vector<int> n1(table.begin(), table.end());
            for (int i = 0; i < n; ++i)
            {
          //如果当前数小于等于n（即在左手），则他下次出现的位置是 2*当前位置
                //与之对应的当前位置 + n（即在右手）的牌,则他下次出现的位置是 2*当前位置 + 1
                table[2 * i] = n1[i];

                table[2 * i + 1] = n1[i + n];
```

```
            }
        }
        for (int i = 0; i < num - 1; ++i)
            cout << table[i] << " ";
        cout << table[num - 1] << endl;
    }
    return 0;
}
```

题目ID:36888-MP3光标位置

链接：https://www.nowcoder.com/practice/eaf5b886bd6645dd9cfb5406f3753e15?tpId=37&&tqId=21287&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking

【题目解析】：

本题的意思是第一行输入歌曲数量，第二行输入指令，最后需要显式的输出也为两行，第一行为当前歌曲所在的列表，第二行为光标所指向的歌曲。

【解题思路】：

本题比较简单，通过解析指令，进行移动即可，分两种情况，歌曲数目不大于4和大于4的情况。

【示例代码】

```
#include <iostream>
#include <string>
using namespace std;
int main(){
//歌曲数量
    int n;
//命令
    string order;
    while (cin >> n >> order)
    {
  //将n首歌曲编号1: n , num为光标当前所在歌曲的编号,first为当前屏幕显示页的第一首歌曲的编号
        int num = 1, first = 1;
        if (n <= 4) //歌曲总数不超过4时，所有歌曲一页即可显示完，不需翻页，first始终不变
        {
            for (int i = 0; i < order.size(); i++)
            {
                if (num == 1 && order[i] == 'U')
                    num = n;
                else if (num == n && order[i] == 'D')
                    num = 1;
                else if (order[i] == 'U')
                    num--;
                else
                    num++;
            }
            for (int i = 1; i <= n - 1; i++)
                cout << i << ' ';
            cout << n << endl;
            cout << num << endl;
```

```
        }
        else  //歌曲总数大于4时，显示完全所有歌曲需要翻页，屏幕总是显示4首歌曲
        {
            for (int i = 0; i < order.size(); i++)
            {
                if (first == 1 && num == 1 && order[i] == 'U')
                { first = n - 3; num = n; } //特殊翻页1
                else if (first == n - 3 && num == n && order[i] == 'D')
                { first = 1; num = 1; } //特殊翻页2
                else if (first != 1 && num == first && order[i] == 'U')
                { first--; num--; } //一般翻页1
                else if (first != n - 3 && num == first + 3 && order[i] == 'D')
                { first++; num++; } //一般翻页2
                else if (order[i] == 'U')
                    num--; //其他情况1
                else
                    num++; //其他情况2
            }
            for (int i = first; i < first + 3; i++)
                cout << i << ' ';
            cout << first + 3 << endl;
            cout << num << endl;
        }
    }
    return 0;
}
```