实验7: 排序

确保把类定义写在`mySort.h`中，类实现写在`mySort.cpp`中，**不要修改文件名！！！**

`mySort.h`文件中已经给出类定义和必要的函数，可以自行添加需要用到的函数，但不要变更已有的函数声明，测试文件只会调用已有的函数。

最终提交两个文件：`mySort.h`和`mySort.cpp`

注意：请务必使用在linux g++ std=c++11编译环境下可以使用的库函数

- 实现排序类`MySort`，其中`num`存储要排序的数目，`origin`存储原始数据，完成以下功能：
  - 类初始化和销毁、私有属性存取等基本功能
  - 排序类的带参数初始化`MySort(int, int*)`，参数列表为：
    - 要排序的数目`int`
    - 原始数据`int*`
  - 奇偶交换排序`string paritySort()`，将原始数据排为升序。每一次进行交换操作，都应输出当前的序列，最终合成并返回一个`string`类型字符串，格式为：

    ```
    “数字1 数字2 ...数字n\n”

    例：“1 2 3 3 5 4\n1 2 3 3 4 5\n”
    ```

    > 序列中各数字之间添加一个空格，序列末尾添加`\n`
    >
    > 排序时可以直接修改原始数据，测试时不会对同一对象的原始数据进行多次排序

  - 双向冒泡排序`string bubbleSort()`，将原始数据排为升序，输出格式同上。
- 已经给出两个辅助用函数：`void changeSwap(int&, int&)`，`string printNum()`。也可以不使用这两个辅助函数，根据自己的习惯自定义实现算法的中间函数。测试时不会调用这两个辅助函数，但请注意不要出现编译错误。

输入输出示例：

```cpp
#include <iostream>
#include "mySort.h"

using namespace std;

int main()
{
    int num = 6;
    int origin1[6] = {4, 6, 6, 5, 1, 6};
    int origin2[6] = {1, 3, 2, 5, 7, 3};
    MySort m1(num, origin1);
    MySort m2(num, origin2);

    cout << m1.bubbleSort();
    cout << m2.paritySort();

    return 0;
}
```

```
19
20
21  /*
22      控制台运行结果为：
23      4 6 5 6 1 6
24      4 6 5 1 6 6
25      4 6 1 5 6 6
26      4 1 6 5 6 6
27      1 4 6 5 6 6
28      1 4 5 6 6 6
29      1 2 3 5 7 3
30      1 2 3 5 7 1
31      1 2 3 5 1 7
32      1 2 3 1 5 7
33      1 2 3 1 5 3
34      1 2 1 3 5 3
35      1 2 1 3 3 5
36      1 1 2 3 3 5
37  */
```