# PART ONE

作者：2014 级电子信息工程--范娇娇

（注明：以下程序已全部通过 OJ）

## 一、分配病房

**1、问题描述：** 某个科室的病房分为重症和普通，只有当病人的疾病严重程度超过了入住重症病房的最低 严重值，才可以安排入住重症病房。 现在要求设计一个程序，给病人安排好病房。疾病的严重程度用 0 到 10 来表示，0 表示小毛病，10 表示非常严重。

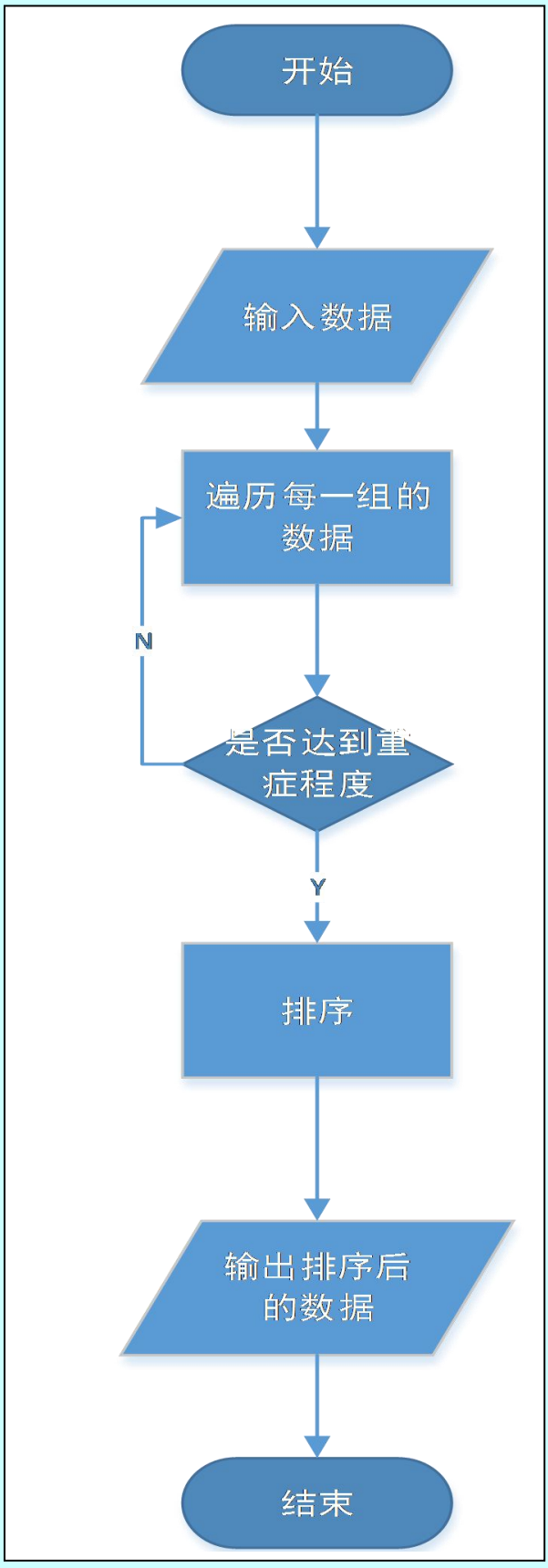
A、输入

- 第一行输入病人的个数 m（m < 50），以及安排住入重症病房的最低严重值 a。
- 紧接着 m 行，每行表示病人编号（三个位，用 0 补齐）及其疾病的严重程度（浮点数，1 位小 数）。
- 每个病人的疾病严重程度都不一样。

B、输出 要求按照病人的严重程度输出住在重症病房里的病人的编号。 注意：如果当前所有病人的严重程度并不满足住在重症病房里，则输出“None.”（不包括引号）。

**2.解决方案：** 本道题目可以分为两部分，一方面是找到符合要求的重症病人，另一方面是将其按程度的严重性从大到小排序；其中输入的每一行包含两个数据，病人的 ID 和严重程度，可以用两个一维数组表示，用下标相同来划分同一组数据。

## 3、流程图




## 4、编程实现

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
	int m;
	float a;
	int flag=0;
	int id[1000];
	float b[1000];
	cin>>m>>a;
for(int i=0;i<m;i++)
{
	cin>>id[i]>>b[i];
}
for(int x=0;x<m-1;x++){
	for(int y=1;y<m-x;y++){
		if(b[y-1]>b[y]){
	float temp1=b[y];
		b[y]=b[y-1];
		b[y-1]=temp1;
	int temp2=id[y];
	  id[y]=id[y-1];
	  id[y-1]=temp2;}}}
for(int j=m-1;j>=0;j--)
{
if(b[j]>=a)
{
cout<<setfill('0');
cout<<setw(3)<<id[j]<<" ";
flag++;
cout<<fixed<<setprecision(1)<<b[j]<<endl;
}

}
if(flag==0)
{
cout<<"None."<<endl;
}
return 0;}
```

**5、结果分析：**此种方法编程虽比较容易实现，但是将病人 ID 和 level 分为两个数组，还是会有些麻烦，尝试用一个数组来解决这个问题，但编程过程中遇到一些问题，目前还没解决，还在查询资料完善。

**6、总结体会：**拿到问题时，有了算法思路，有了流程图作为编程指导，但在编程实现的过程中还是会遇到一些比较棘手的问题，反思了一下，还是目前的所遇到的错误情况比较少，简单题目也会出现较难的问题的，果然，积累也是一部分宝贵的资源，keep practicing!

## 二、碱基配对

**1、问题描述：**脱氧核糖核酸（DNA）由两条互补的碱基链以双螺旋的方式结合而成。而构成 DNA 的碱 基共有 4 种，分别为腺瞟呤（A）、鸟嘌呤（G）、胸腺嘧啶（T）和胞嘧啶（C）。我们知道，在两条互 补碱基链的对应位置上，腺瞟呤总是和胸腺嘧啶配对，鸟嘌呤总是和胞嘧啶配对。你的任务就是根据 一条单链上的碱基序列，给出对应的互补链上的碱基序列。

A、 输入

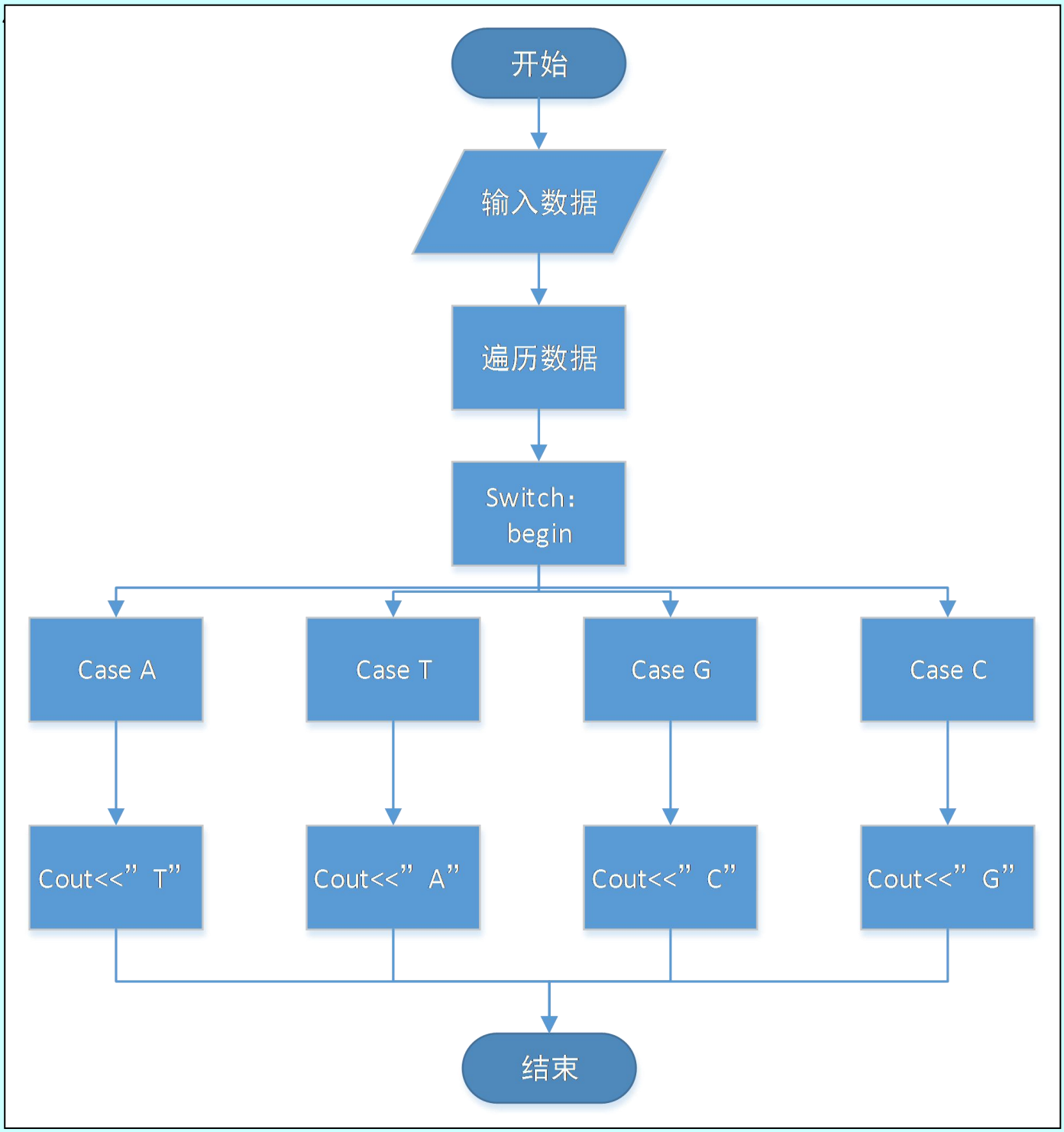
- 第一行是一个正整数 n，表明共有 n 条要求解的碱基链。
- 以下共有 n 行，每行用一个字符串表示一条碱基链。这个字符串只含有大写字母 A、T、G、C， 分别表示腺瞟呤、胸腺嘧啶、鸟嘌呤和胞嘧啶。每条碱基链的长度都不超过 255。

B、输出

- 共有 n 行，每行为一个只含有大写字母 A、T、G、C 的字符串。分别为与输入的各碱基链互补的碱基链。

**2、解决方案：**首先输入问题，可以用字符串数组来解决；其次，有四种碱基进行配对，最容易联想到的就是利用 swith case 语句来解决，具体思路如下 3 中流程图

**3、流程图**



## 4、编程实现

```
#include<iostream>
using namespace std;
int main() {
	int n;
	cin >> n;
	cin.get();
	for (int i = 0; i < n; i++)
	{
		char a[256];
		cin.getline(a,256);
		char base[256];
		for (int j = 0; j < 256; j++)
		{
			switch (base[j])
			{
			case 'A': base[j] = 'T'; break;
			case 'T': base[j] = 'A'; break;
			case 'C': base[j] = 'G'; break;
			case 'G': base[j] = 'C'; break;
			default: base[j] = '\0'; break;

			}
		}
		for (int k = 0; k< 256;k++)
		{
			if (base[k] != '\0')
				cout << base[k];
			else
			{
				cout << endl;
				break;
			}
		}
	}
	return 0;
}
```

**5、问题分析：**本题的算法思想很简单易懂，但有同学写了以下程序

```
#include<iostream>
using namespace std;
int main()
{
    int n,i,j;
    char a[256][256];
    cin>>n;
    cin.get();

    for(i=0;i<n;i++)
    cin>>a[i];

    for(i=0;i<n;i++)
    {
        for(j=0;j<256;j++)
            switch(a[i][j])
        {
            case'A':cout<<"T";break;
            case'T':cout<<"A";break;
            case'C':cout<<"G";break;
            case'G':cout<<"C";break;
        }
        cout<<endl;
    }
    return 0;}
```

其算法思想与我的相同，但在 OJ 上的运行结果为 runtime error。一般这种情况问题应该是因为数组问题，但我目前还并没有将其改正为可以通过 OJ 的程序，还在请教修改中（注明：此程序在其他编译环境下均能正常执行，且运行结果正确）

**6、总结分析：**经常问题越简单，却越容易出现难以发现或者难以轻易修正的问题，这个时候转化思想，优化程序可能是目前解决问题的最佳选择了，算法思想正确，但编程过程中还是会有不同的问题出现，突然感觉编程要经历张无忌练武功的过程，如果你把什么都忘了，那么说明你就会了，但是前提就是基础要牢，不用硬记就能灵活运用，革命尚未成功，同志还需努力哈！！！

# PART TWO

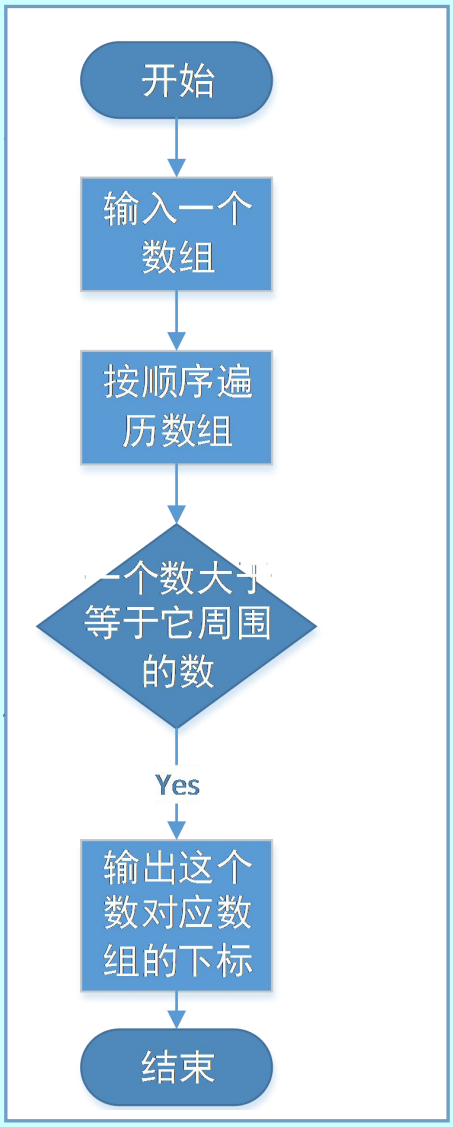
作者：14-电子信息工程-郑春瑞

## 三、寻找山顶

**1、问题描述：**在一个 m×n 的山地上（想象成一个二维数组），已知每个地块的平均高程，求出所有山顶所在的地块（所谓山顶，就是其地块平均高程不比其上下左右相邻的四个地块每个地块的平均高程小的地方），输出该数组相应下标。

**2、解决方案：**输入一个二维数组，因为题目相当于单个数组与其周围元素相比较，利用循环语句和 if 语句来实现。

**3、流程图**




```
#include<iostream>
using namespace std;
int main()
{
	int m,n;
	int mou[100][100];
	cin>>m>>n;
	for(int k=0;k<m;k++){
		for(int l=0;l<n;l++){
			cin>>mou[k][l];
		}
	}
	for(int i=0;i<m;i++){
		for(int j=0;j<n;j++){

	if((mou[i][j]>=mou[i-1][j]&&i-1>=0||i-1<0)&&(mou[i][j]>=mou[i+1][j]&&i+1<m||i+1>=m)&&(mou[i][j]>=mou[i][j-1]&&j-1>=0||j-1<0)&&(mou[i][j]>=mou[i][j+1]&&j+1<n||j+1>=n))
				cout<<i<<" "<<j<<endl;
		}
	}
	return 0;
}
```

**5、总结：**

这个程序和之前一次作业细菌感染的题有点像，但是比那个题要简单些，这个题的重点我觉得是要将情况考虑全面，一开始的时候我忘了最边上的情况代码应该怎么实现，导致遇到了一些麻烦，再仔细考虑了之后才明白，所以读题认真思考是很重要的一个过程。

**四、实验总结：**

通过最近几次作业，感觉自己还有很多不足，这次作业比之前两次的要简单一些，但是有的还是没有做到独立完成，经常出现考虑不全面甚至还经常出现一些小错误，这也暴露了程序编写的少，应该加强对各种语句的锻炼，而且明白了有些程序可能没有那么难，但是往往自己有时候有些无从下手，所以认真读题和分析才能让自己一点点入门。