# 《程序设计基础》第 1 次上机作业

# 范例程序

**1. 鸡和兔的数目。[此题与《C++程序设计实验指导》实验 3 编程题（3）相同]（提示：此题可以不用循环）**

**方法 1：**设鸡有 a 只，兔有 b 只。首先，如 m 为奇数，无解。如 m 为偶数，我们设 a=n, 则总腿数为 2n。然后可比较 2n 和 m 的大小。若 2n > m, 则无解。若 2n <= m，则兔有 b= (m-2n) / 2 只，鸡有 n-b 只，因此 n- b>=0, 即 b<=n, 也即 m<=4n。

**方法 2：**设鸡有 a 只，兔有 b 只。则 a+b = n, 2a+4b = m, 联立解得 a = (4n-m) /2, b = n-a。要此解可行，需要 a, b 为整数，且 a, b 必须是非负的。

```
// 解法 1：
#include <iostream>
using namespace std;
int main()
{
	int a, b, n, m;
	do {
		cout << "请输入鸡兔总数 n, 腿数 m:";
		cin >> n >> m;
	}while(n<=0 || m <=0);

	if(m % 2 == 1 || 2*n > m || m > 4*n)
	{
		cout << "无解" << endl;
		return 0;
	};

	b = (m-2*n)/2; a = n-b;
	cout << "鸡" << a << "只， 兔" << b << "只"<< endl;
	return 0;
}

// 解法 2：
#include <iostream>
```

```
using namespace std;
int main()
{
    int a, b, n, m;
    do {
        cout << "请输入鸡兔总数 n, 腿数 m:";
        cin >> n >> m;
    }while(n<=0 || m <=0);
    a = (4*n-m)/2;
    b = n - a;
    if(m % 2 == 1 || a < 0 || b < 0 )
        cout << "无解" << endl;
    else
        cout << "鸡" << a << "只， 兔" << b << "只"<< endl;
    return 0;
}
```

**2. 判断是否存在以这 3 个点为顶点的三角形（假定所有坐标值均为整数，且各坐标值的绝对值 ＜215 ） [ 根据《C++程序设计实验指导》实验 3 编程题（4）改编]**

【分析】：若三点共线，则不存在以这三点为顶点的三角形。可以从两个点连线的斜率来判断。注意：由于浮点数存在误差，因此尽量避免使用浮点数。以下程序通过求每条边的长度，用两边之和大于第三边来判断，看起来逻辑上没有问题，但实际上有错误。该程序的问题是，它不总是正确，有时会出现错误。由于错误只是偶尔出现，因此捕获错误比较困难。

【有错的程序，解法 1】此为错误的解法

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int x1, y1, x2, y2, x3, y3;
    double a, b, c;
    cout << "请输入 x1,y1:";
    cin >> x1 >> y1;
    cout << "请输入 x2,y2:";
    cin >> x2 >> y2;
    cout << "请输入 x3,y3:";
    cin >> x3 >> y3;
    a=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
    b=sqrt((x1-x3)*(x1-x3)+(y1-y3)*(y1-y3));
    c=sqrt((x2-x3)*(x2-x3)+(y2-y3)*(y2-y3));
    if((a+b>c) && (a+c>b) && (b+c >a))
        cout << "yes" << endl;
```

```
    else
        cout << "no" << endl;
    return 0;
}
```

请尝试如下测试数据，看看结果：

```
1 1
3 3
17 17

1 1
3 3
23 23
```

【解法 2，参考程序】

程序修改思路：因为输入为整数，因此尽量避免出现浮点数运算。可以考虑使用斜率相等作为测试条件，然后将条件变换为仅仅包含乘法。即，(y2-y1) / (x2-x1) == (y3-y1)/(x3-x1)
转换为：(y2-y1)(x3-x1) ==(y3-y1)(x2-x1).

```
// 程 序 如 下 ：
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int x1, y1, x2, y2, x3, y3;
    double a, b, c;
    cout << "请输入  x1,y1:";
    cin >> x1 >> y1;
    cout << "请输入  x2,y2:";
    cin >> x2 >> y2;
    cout << "请输入  x3,y3:";
    cin >> x3 >> y3;
    if ( (y2-y1) *(x3-x1) != (y3-y1)* (x2-x1) )
        cout << "yes" << endl;
    else
        cout << "no" << endl;
    return 0;
}
```

**3、计算和式的值。**

```
// 方法 1：
#include<iostream>
using namespace std;
```

```
int main()
{
	int sum=0;
	for(int i = 1; i <= 51;i++){
		sum += (i % 2) ? 2*i - 1 : 1 - 2*i;
	}
	cout<<"1 - 3 + 5 -7 + …… + 101 = "<<sum<<endl;
	return 0;
}

// 方法 2：
#include<iostream>
using namespace std;
int main()
{
	int sum=0, sign = 1;
	for(int i = 1; i <= 101; i += 2){
		sum += sign * i;
		sign *= -1;
	}
	cout<<"1 - 3 + 5 -7 + …… + 101 = "<<sum<<endl;
	return 0;
}

// 方法 3：
#include<iostream>
using namespace std;
int main()
{
	int sum=0;
	bool sign = false;
	for(int i = 1; i <= 101; i += 2){
		sum += ( sign^=true) ? i : -i;
	}
	cout<<"1 - 3 + 5 -7 + …… + 101 = "<<sum<<endl;
	return 0;
}
```



**4. 子序列的和**

【分析】当整数比较大时，譬如大于 47000 时，直接计算其平方会溢出。因此，此题需要避免计算整数的平方。可以考虑先计算一个整数的倒数，得到一个浮点数，然后再相乘。

【参考代码】

```
#include <iostream>
#include <iomanip>
```

```
using namespace std;
int main()
{
	int n,m; double S = 0;
	cin >> n >> m;

	for(int i= n; i <= m; i++)
	{
		double item = 1.0/i;
		item *= item;
		S += item;
	}
	cout.setf(ios::fixed); //控制按照小数输出
	cout << setprecision(5) << S << endl;
	return 0;
}
```

**5、打印*字符构成的金字塔程序。**

```
#include <iostream>
using namespace std;
int main(void){
	while(true){
		int rowsNumber;
		cout<<"Please Enter the Number of Rows:";
		cin>>rowsNumber;
		if(rowsNumber >0){
			for(int i=0; i< rowsNumber; i++){
			    for(int j=0; j < rowNumber - i – 1; j++)
				    cout << ‘ ‘;
			    for(int j=0; j < 2*i + 1; j++)
				    cout << ‘*‘
			    cout<<endl;
			}
		}
		else return 0;
	}
	return 1;
}
```

**6、打印数字金字塔**

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

int main(){
```

```
    int number;
    while(true){
        cout<<"请输入金字塔的层数(>0):";
        cin>>number;
        if(number>0){
            for(int i=0;i<number;i++){
                for(int j=0;j<number-i-1;j++) cout<<' ';
                for(int j=0;j<i+1;j++) cout<<j+1;
                for(int j=i;j>0;j--) cout<<j;
                cout<<endl;
            }
        }
        else
            return 0;
    }
}
```

**7、 打印 N*M 方格表中所有方格的相邻方格坐标**

```
#include <iostream>
using namespace std;
void main(){
    void searchAdjoining(int rowsNumber,int colsNumber);
    int rowsNumber,colsNumber;
    cout<<"请输入方格表的行数（N）和列数（M）:";
    cin>>rowsNumber>>colsNumber;
    if(rowsNumber>0&&colsNumber>0)
        searchAdjoining(rowsNumber,colsNumber);
    else
        cout<<"输入数据错误！";
}

void searchAdjoining(int rowsNumber,int colsNumber){
    for(int row=0;row<rowsNumber;row++)
        for (int col=0; col<colsNumber;col++){
            cout<<"当前方格坐标是"<<"["<<row<<","<<col
                <<"]，相邻方格坐标："<<endl<<"      ";
            for (int i=-1;i<2;i++){
                int adjoiningRow=row+i;
                if(adjoiningRow<0||adjoiningRow==rowsNumber) continue;
                for (int j=-1;j<2;j++){
                    int adjoiningCol=col+j;
                    if(adjoiningCol<0||adjoiningCol==colsNumber
                            ||(adjoiningRow==row&&adjoiningCol==col))
                        continue;
```

```
                    cout<<"["<<adjoiningRow<<","<<adjoiningCol<<"] ";
                }
            }
            cout<<endl;
        }
}
```