***运算符重载、动态内存管理***

***1．已知字符串类MyString的定义为：***

**class MyString**

**{**

**public:**

**MyString(const char\* pData= NULL); // 普通构造函数**

**MyString(const MyString &); // 拷贝构造函数**

**~ MyString(); // 析构函数**

**MyString & operator =(const MyString &); // 赋值函数**

**MyString& operator += (const MyString &);**

**operator char\* () const; // 自动转换函数**

**private:**

**char \*mpData; // 用于保存字符串**

**};**

**全局函数：**

**const MyString operator + (const MyString &,const MyString &); //字符串连接**

**ostream& operator<<(ostream& os, const MyString& str); //定向输出**

**请完整实现MyString类和指定的全局函数。（可以使用new,delete运算以及strcpy,strlen，…等库函数，还可以定义其他辅助函数）。**

***2.对任意一个正的实数,总可以唯一地表示成a0+1/(a1+1/(a2+1/(a3+1/.....)))的形式，简记为a0+a1+a2+a3+...+ak。(ai为大于0 的正整数，长度为k。若数为无理数，则k为无穷大) ,并称这种形式的数为连分数。***

**a)请使用类的自关联方式设计并实现连分数类, 此类的每个对象代表一个有理数的连分数形式，且假定连分数的长度均小于MAXLEN（MAXLEN设为数30）。**

**该类主要功能有：**

**1)对指定的i，返回ai值；**

**2)计算连分数的前q项（a0+a1+a2+a3+..+aq）所对应分数的分子和分母；**

**3)输出此分数的前q项，格式为:a0+a1+a2+a3+..+aq；**

**b)给出主程序，使用该类计算连分数的前n项对应的分数逼近PI值的程度（计算差即可。PI可用math.h中的M\_PI常量）。**

**（由此可见祖老师给出的约率和密率是多么地精确！连分数的收敛速度多么地快！）**

**例如主程序的输出可能如下：**

**PI=3.141592653589793**

**前1项为3**

**前1项对应分数为3/1**

**前1项对应分数的值为3**

**前1项对应分数与3.141592653589793的差为0.1415926535897931**

**前2项为3+7**

**前2项对应分数为22/7**

**前2项对应分数的值为3.142857142857143**

**前2项对应分数与3.141592653589793的差为-0.001264489267349741**

**前3项为3+7+15**

**前3项对应分数为333/106**

**前3项对应分数的值为3.141509433962264**

**前3项对应分数与3.141592653589793的差为8.321962752896503e-005**

**前4项为3+7+15+1**

**前4项对应分数为355/113**

**前4项对应分数的值为3.141592920353983**

**前4项对应分数与3.141592653589793的差为-2.667641891848736e-007**

**前5项为3+7+15+1+292**

**前5项对应分数为103993/33102**

**前5项对应分数的值为3.141592653011902**

**前5项对应分数与3.141592653589793的差为5.778905119192823e-010**

**前6项为3+7+15+1+292+1**

**前6项对应分数为104348/33215**

**前6项对应分数的值为3.141592653921421**

**前6项对应分数与3.141592653589793的差为-3.316279286770529e-010**

**前7项为3+7+15+1+292+1+1**

**前7项对应分数为208341/66317**

**前7项对应分数的值为3.141592653467437**

**前7项对应分数与3.141592653589793的差为1.223564103768754e-010**

**前8项为3+7+15+1+292+1+1+1**

**前8项对应分数为312689/99532**

**前8项对应分数的值为3.141592653618936**

**前8项对应分数与3.141592653589793的差为-2.914350748575711e-011**

**前9项为3+7+15+1+292+1+1+1+2**

**前9项对应分数为833719/265381**

**前9项对应分数的值为3.141592653581078**

**前9项对应分数与3.141592653589793的差为8.715344852056051e-012**

**前10项为3+7+15+1+292+1+1+1+2+1**

**前10项对应分数为1146408/364913**

**前10项对应分数的值为3.141592653591404**

**前10项对应分数与3.141592653589793的差为-1.610862485068587e-012**

**.....**

***3．在全局函数*void f(int n,int m)中动态建立一个大小为n×m的二维整数数组，并对每个数组元素依次赋值为1，2，3，...,m\*n, 再输出每行和每列元素的和，退出函数前释放此数组。**

***4．已有类A声明*如下：**

**class A**

**{**

**public:**

**A(int n):data(n) {}**

**int Data() const {return data;}**

**private:**

**int data;**

**};**

**在全局函数void g(int n) 中动态建立一个大小为n的一维指针数组， 数组元素指向A类对象，各对象的data数据成员各不相同， 分别为1，2，3，...,n。创建数组后，使用（例如输出）各对象的data值，最后释放此数组。**

***5．如何建立一个二维整数数组*类，使得其大小可动态决定，并且访问时，可象普通数组一样使用。如 建立这个二维整数数组类的一个对象obj后, 访问其第二行、第三列元素，可写成：**

**obj[1][2] = 5;**

**cout<<obj[1][2];**