实验三　静态成员和友元

1. 编写一个圆类 Circle，要求有函数可以求两圆周长之和，有函数可以输出创建的圆对象的个数。（注意：这里说的函数，可以是成员函数，也可以是普通函数）

```
#include<iostream>
using namespace std;
class circle
{
private:
    double r;
    static int count;
public:
    circle(double rr=0):r(rr){;count++;}
    double cal(circle &a)
  //这里一定最好用引用，如果直接用对象，就会调用默认的拷贝构造函数，执行完这个函数会自动调用析构函数，count 会减 1，但默认的拷贝构造函数中 count 没有加 1，这样算出来的圆的个数会少一个。其二，定义对象还要分配空间收回空间，是不必要的开销。
    {
        return 2*3.14*(r+a.r);
    }
    static int getcount(){return count;}
    ~circle(){count--; }
};
int circle::count=0;
int main()
{
    circle a(3),b(2),c[5];
    cout<<a.cal(b)<<endl;
    cout<<"定义的圆的个数:"<<circle::getcount()<<endl;
    return 0;
}
```

2. 定义一个学生类，有学号、姓名、三门课程的成绩，要求能够计算一个班级的各科课程的平均分。（班级人数最多 40 人）

```
#include<iostream>
using namespace std;
class Student
{
    int no;
    char s[10];
    double math,chinese,english;
    static int count,avgm,avgc,avge;
public:
    void input()
    {
```

```
            cout<<"请输入学号 姓名 语文 英语   数学:"<<endl;
            cin>>no>>s;
            cin>>chinese>>english>>math;
            count++;
            avgc=avgc+chinese;
            avgc=avgc/count;
            avgm=avgm+math;
            avgm=avgm/count;
            avge=avge+english;
            avge=avge/count;
        }
        static void showavg()
        {   cout<<"语文平均分:"<<avgc<<endl;
            cout<<"英语平均分:"<<avge<<endl;
            cout<<"数学平均分"<<avgm<<endl;

        }
};
int Student::count=0;
int Student::avgm=0;
int Student::avgc=0;
int Student::avge=0;
void main()
{   int n;
    cout<<"请输入班级人数:"<<endl;
    cin>>n;
    Student *p;
    for(int i=0;i<n;i++)
    {
        p=new Student;
        p->input();
    }
    Student::showavg();
    delete p;
}
```

3. 有一个 Time 类，其中包含数据成员 hour，minute，second；有一个 Date 类，其中包含数据成员 year，month，day，还有成员函数 display 用于输出年、月、日及时、分、秒。

```
#include<iostream>
using namespace std;
class time
{   int hour,minute,second;
public:
    time(int a=0,int b=0,int c=0):hour(a),minute(b),second(c){};
    friend class date;
```

```
};
class date
{    int year,month,day;
public:
    date(int a=0,int b=0,int c=0):year(a),month(b),day(c){};
    void show(time a)
    {        cout<<year<<"-"<<month<<"-"<<day<<endl;
             cout<<a.hour<<":"<<a.minute<<":"<<a.second<<endl;
    }
};
void main()
{
    time a(0,59,59);
    date n(2017,3,10);
    n.show(a);
}
```

4. 写出一个函数模板 findmax，函数功能为求三个数的最大值，定义主函数，调用该函数模板，可以求出各种数据类型的三个数的最大值。

```
#include<iostream>
using namespace std;
template<typename T>
T findmax(T a,T b,T c)
{
    T m=a;
    if(b>m)
            m=b;
    if(c>m)
            m=c;
    return m;
}
void main()
{
     cout<<findmax(3,5,2)<<endl;
     cout<<findmax<int>('a',56,6.55)<<endl;
}
```

5. 定义一个交换两变量值的函数模板。

```
#include<iostream>
using namespace std;
template<typename T>
void exchange(T &a,T &b)
{   T t;
    t=a;
    a=b;
    b=t;
```

```
}
void main()
{
    int a=4,b=5;
    double d1=23.234,d2=435.23;
    exchange(a,b);
    cout<<"a="<<a<<"   b="<<b<<endl;
    exchange(d1,d2);
    cout<<"d1="<<d1<<"  d2="<<d2<<endl;
}
```

6.定义一个类模板类，可以实现求一维数组中的最大值、最小值和平均值。

//类模板没讲解，请自学一下。其实和定义一般类差不多，不一样的地方就是要能适合各种不同的数据类型。我们可以先定义一个只针对整数的类，然后扩展到可以适合各种不同类型的情况，这就需要定义一些通用数据类型

如：**template <typename T >**　　//说明了 1 个通用数据类型 T

template <typename T1,typename T2>　//说明了 2 个通用数据类型 T1 和 T2

```
#include <iostream>
using namespace std;
#define N 100
template <class T1,class T2=float> //模板参数声明，第 2 个参数有默认值
class Array
{public:
        Array(T1 aa[],int nn)
        {   n=nn;
        for(int i=0;i<n;i++)
                data[i]=aa[i];
        }
        T2 getaver()                        //求数组元素平均值
        {   T1 sum=0;
            T2 aver;
            for(int i=0;i<n;i++)
                sum+=data[i];
            aver=sum/(double)n;
            return aver;
        }
private:
        T1 data[N];
        int n;
};
int main()
{   int a[]={1,2,3,4,5,6,7,8};
        Array<int,int> arr1(a,sizeof(a)/sizeof(int)); //定义类模板对象
        cout<<"aver="<<arr1.getaver()<<endl;
```

```
        Array<int> arr2(a,sizeof(a)/sizeof(int));
                    //定义类模板对象，第 2 个模板参数取默认值 float
        cout<<"aver="<<arr2.getaver()<<endl;
        return 0;
    }
```

如果成员函数写在类外，程序如下：

```
#include <iostream>
using namespace std;
#define N 100
template <class T1,class T2=float> //模板参数声明，第 2 个参数有默认值
class Array
{public:
        Array(T1 aa[],int nn);
        T2 getaver();                           //求数组元素平均值
private:
        T1 data[N];
        int n;
};
template<class T1,class T2>
Array<T1,T2>::Array(T1 aa[],int nn)
{   n=nn;
    for(int i=0;i<n;i++)
        data[i]=aa[i];
}
template<class T1,class T2>
T2 Array<T1,T2>::getaver()
{   T1 sum=0;
    T2 aver;
    for(int i=0;i<n;i++)
        sum+=data[i];
    aver=sum/(double)n;
    return aver;
}
int main()
{   int a[]={1,2,3,4,5,6,7,8};
    Array<int,int> arr1(a,sizeof(a)/sizeof(int)); //定义类模板对象
    cout<<"aver="<<arr1.getaver()<<endl;

    Array<int> arr2(a,sizeof(a)/sizeof(int));
    cout<<"aver="<<arr2.getaver()<<endl;
    return 0;
}
```