《C++面向对象程序设计》模拟试题（8）

一、单选题(共 20 分，每题 2 分)

1. 关注重要的属性，而忽略非本质的细节被称为：
   (A) 封装　　(B) 隐藏　　(C) 抽象　　(D) 覆盖
2. 下列哪个最能体现对象与类的关系：
   (A)书和图书馆　　(B) 林肯和总统　　(C) 母亲和女儿　　(D) 学生和研究生
3. 相同的操作，但可能有不同的实现，这一面向对象特征被称为：
   (A) 复用　　(B) 继承　　(C) 多态　　(D) 名字空间
4. 如果类 C 以公有继承方式从类 B 派生，类 B 以公有继承方式从类 A 派生，那么类 C 的成员函数能访问：
   (A) 类 B 和类 A 中的私有的数据成员　　(B) 仅类 C 中公有和保护的数据成员
   (C) 类 B 和类 A 中的保护的数据成员　　(D) 仅类 C 与类 B 中公有和保护的数据成员
5. 下列哪个类不需要显式定义拷贝构造函数：
   (A) 一个字符串类，其构造函数中分配了堆中的内存空间，析构函数中进行了释放
   (B) 一个公司雇员类，其任意两个实例对象不能具有相同的员工编号
   (C) 一个日志类，其至多只能有一个实例
   (D) 一个图书馆类，其包含一个书籍对象的数组
6. 定义类 A 的***静态***成员函数 A& f(A & a)时，下列不能作为函数返回值的是：
   (A) a　　(B) 类 A 中定义的类型为 A 的静态数据成员
   (C) *this　　(D) 函数 f 中用语句 A& b=a; 定义的量 b
7. 下列关于对象存储的说法，正确的是：
   (A) 对象中不包含所有静态函数的地址，但包含所有非静态函数的地址
   (B) 对象中非静态数据成员可能存储在栈区，也可能在堆区，还可能在全局数据区
   (C) 对象中静态数据成员可能存储在栈区，也可能在堆区，还可能在全局数据区
   (D) 对象中只存放在本类中定义的所有非静态数据成员
8. 运行时多态可以通过下列哪种机制实现：
   (A) 静态联编　　(B) 虚函数　　(C) 内联函数　　(D) 运算符重载
9. 假设有如下类定义 class B{ /*略*/ }; class A{ public: void fun(B&); /*略*/ }; 则类 A、B 之间最可能的关系是：
   (A) 依赖关系　　(B) 聚合关系　　(C) 组合关系　　(D) 泛化关系
10. 下列关于抽象类的描述中，正确的是：
    (A) 抽象类中的所有成员函数必须声明为纯虚函数
    (B) 抽象类不能实例化，但可以有数据成员
    (C) 抽象类没有构造函数，所以自身不能实例化，只能实例化其派生类
    (D) 抽象类和其派生类共用一个虚函数表

二、判断正误，**对于你认为错误的论述，说明原因或举出反例。**（共 20 分，每题 2 分）

1. 在类的非静态成员函数中，可以通过(*this).x 的方式访问类的数据成员 x。
2. 在自定义类的赋值函数时，应先判断是否是自赋值。
3. 在类Book中可以同时定义多个重载的构造函数，如Book ( ); Book (char* author, char* title)。
4. 静态对象成员会在所属类的析构函数被调用时自动析构。
5. 类 A 的友元函数可以访问类 A 及其派生类的私有数据成员和成员函数。
6. 虚基类的所有成员函数都是虚函数。
7. 复用一个已有的类时，应优先选择继承方式，而不是组合方式。
8. 使用异常处理机制不能解决程序的编译时错误和逻辑错误。
9. 构造函数为常函数时，这个类就只能创建常对象。
10. 类成员的初始化是按照初始化列表中的顺序进行，因而程序员可以根据需要指定成员的初始化顺序。

三、回答下列各题（每题 4 分，共 20 分）

1. 某个 C++应用程序由 1 个.cpp 文件和 3 个.h 文件组成，.cpp 文件为 mymain.cpp，.h 文件分别为 file1.h、file2.h、head.h，文件间的包含关系如下：

//mymain.cpp
#include "file1.h"
#include "file2.h"
// 略
	// file1.h
	#include " head.h"
	// 略
		//file2.h
		#include "head.h"
		// 略
			//head.h
			int data =10；
			class A { /* 略 */ }；

请分析该程序是否存在问题，如果存在，如何改进？

2. 如果要编写一个虚函数 func，它接收自定义 FOO 类型的参数，形式可为传值、传引用、传指针，且无返回值；同时强调当对象处理消息 func 时，不会修改本对象，也不修改**实参对应的对象**。请给出至少 3 种不同写法的 func 函数原型。
3. 请说明自定义名字空间的好处。
4. 如何定义一个类，使其至多只能有一个实例？(要求给出示例代码)
5. 请用面向对象方法分析鹅(Goose)、鲤鱼(Carp)、鱼群(Shoal)、游泳(Swim)之间的关系，可以添加必要的类。

四、(8 分)公交车在固定线路上行驶，从起点 A，经 B，C，D，到达终点 E。假定每人次车票价格固定为 2 元，每辆公交车都有最大载客量限制。

1. 针对给出的 Bus 类定义，给出完整实现。

2. 编写 main 函数，利用 Bus 类，计算并输出一辆最大载客量为 80 人的公交车单向运行一趟的总收入。

```
class Bus {
public:
  Bus( int maxCapacity);
  // nDown<0 时，表示全体下车
  void ToStation(int nUp,int nDown);
  int GetIncome( ) const;
   //接右边
private:
    void Up(int num);
    void Down(int num);
    const int capacity;
    int     passagers;
    int     income;
};
```

车站	待上车人数	待下车人数
A	40	0
B	50	10
C	60	20
D	30	15
E	0	全体

五、分别写出下面两个程序的运行结果（共 10 分）

1.

```
#include<iostream.h>
class A{
   static    int    count;
public:
   static void Func(A a);
   A(int n = 0): mVal (n)                {    Show ( );}
   A(const A& a):mVal(a.mVal)      {    Show( ); }
   void Show ( ) {
         mVal += count ++;
         cout << mVal << endl;
    }
   ~A( ) {    Show ( );    }
private:
   int mVal;
};
```

```
void A::Func(A a){
      cout << a.mVal ++ << endl;
}

int A:: count =0;
int main( ) {
      A a(0);
      A b(a);
      A c(1);
      A::Func(b);
      return 0;
}
```

2.

```
#include <iostream.h>
class Base {
public:
    virtual ~Base( ) {}
    virtual void vf( ) {
          cout<<“Base::vf( )”<<endl;
    }
    virtual void vg( )    {
       cout<<“Base::vg( )”<<endl;
       vf( );
       nvh( );
    }
    void nvh( ) {
       cout<<“Base::nvh( )”<<endl;
       vf( );
    }
};
```

```
class Derived: public Base {
public:    virtual ~ Derived( ) {}
     virtual void vf( ) {
          cout<<“Derived::vf( )”<<endl;
     }
     void nvh( )    {
          cout<<“Derived::nvh( )”<<endl;
          vf( );
     }
     virtual void My( ) {    }
};
```

```
int    main( ) {
     Base * p = new Derived;
     p->vf( );
     p->vg( );
     p->nvh( );
     delete p;
     return 0;
}
```

六、(12 分) 某程序需要定义并实现具有上锁(Lock)，开锁(unlock)功能的门(Door)类。另外，还需要定义带有报警功能的门(AlarmDoor)，一个 AlarmDoor 只能有一个报警器(Alarm)，并通过这个报警器实现报警功能。

1) 针对上述需求，请给出相关类的定义，使其较好地符合面向对象设计思想。

2) 如果程序需要多种不同种类的 Door，各种类的 Door 有不同的 lock，unlock 功能实现；同时还需要多种不同的 Alarm，不同种类 Alarm 的报警功能实现也互不相同。请问你在 1)中的设计能否适应这种要求，若能，说明如何适应新要求；若不能，说明如何改进设计来适应新要求。

七、(10 分) n 维空间中的一个向量 V 可表示为($X_1,X_2,X_3,...,X_n$). 其 k 阶模定义为：

$$\|V\|_k = \left(\frac{\sum_{i=1}^{n} |X_i|^k}{n} \right)^{\frac{1}{k}}$$

特别有：$\|V\|_1 = \frac{\sum_{i=1}^{n} |X_i|}{n}$，$\|V\|_2 = \sqrt{\frac{\sum_{i=1}^{n} |X_i|^2}{n}}$，

$$\|V\|_{inf} = \max(|X_1|,|X_2|,...,|X_n|)$$

这三种 (1 阶、2 阶、无穷)也是最常用的模。对给定的 k，若$\|V\|_k=1$，则称 V 为 k 阶下的标准向量。下面是小赵针对非 0 的 3 维向量给出的定义：

```
#include <math.h>
const int N = 3;
class Vector {
public:
   Vector( double values[N]) {
       for( int i = 0; i < N; ++i )
            items[i] = values[i];
    }

   double Item( int index ) const
   {   return items[index];   }

   // 标准化
   Vector& Standard( int order )   {
       double nv = NormalValue(order);
       for( int i = 0; i < N; ++i )
                items[i] /= nv;
       return *this;
   }
```

```
// 接左侧
// 计算向量的模
double NormalValue( int order ) const
{
   if ( order <= 0 ) { //计算无穷阶模
       double result = 0.0;
       for( int i = 0; i < N; ++i )
            result = max( result, fabs( items[i] ) );
       return result;
   } else {   //计算 order 阶模
       double sum = 0.0;
       for( int i = 0; i < N; ++i )
            sum += pow( fabs( items[i] ), order );
       return pow( sum / N, 1.0 / order );
   }
}
protected:
       double items[ N ];
};
```

可是，小赵发觉上述代码有不妥之处：用 0 或负数表示无穷大阶，缺乏合理性；每次标准化和计算模值时都需要将阶数作为参数，这不符合用户习惯;而且他还知道，绝大多数应用程序使用的是 1 阶或 2 阶或无穷阶，虽然存在使用其它阶的应用程序，但这种情况极少出现。

根据上述描述，请采用***子类型化的方法***重新设计 Vector 类及相关类，使得标准化向量和计算向量的模值时，不必再指定阶数作为参数。给出你的设计方案和实现代码(不用考虑除 0、数组下标越界等异常的处理)。

（全卷完）