

面向对象程序设计模拟试卷一答案

一解：1. A 2. D 3. C 4. B 5. C

二解：各类的成员及其存取属性如下。

1.private:	2.private:
int a;	int d;
protected:	protected:
int b;	int e;
public:	int b;
int c;	public:
	int c;
	int f;
3.private:	4.private:
int g;	int n, j;
int b;	int e, c;
int c;	int f, h;
protected:	int i;
int h;	protected:
public:	int b;
int i;	int k;

三解：main 中每行的输出结果见如下注释。

```
A a; cout<<"\n";    //输出 A
B b; cout<<"\n";    //输出 B
C c; cout<<"\n";    //输出 AC
D d; cout<<"\n";    //输出 BACD
E e; cout<<"\n";    //输出 AACE
F f; cout<<"\n";    //输出 BACACDAACEF
```

四解：错误及其原因见注释。

```
class A{
    static int a=0;    //不能在类的体内初始化
protected:
    int b;
public:
    int c;
    A(int);
    operator int( );
} a(1, 2);    //没有定义A(int, int)
class B: A{
    B(int);
    virtual int d;    //virtual不能用于数据成员
    int e;
public:
    A::b;    //修改而不是恢复访问权限
```

```

    friend int operator =(B); //不存在普通赋值函数的重载，=必须重载为双目的
    static B(int, int);      //构造函数不能定义为不带this参数的静态函数成员
} b=5;                      //B(int)是不可访问的私有构造函数

class C: B{
public:
    int operator++(double); //必须用int定义后置运算
}; //C 必须定义构造函数，因 B 定义有带参构造函数且没有无参构造函数

int main( ){
    int *A::*p, i;
    i=a.a;                //a.a 是私有的，不能被不是 A 的友元的 main 访问
    i=A(4);
    i=b.c;                //b.c 是私有的，不能被不是 B 的友元的 main 访问
    p=&A::c;              //A::c 不是一个 int *指针
    i=b;                  //B 不存在公有的 B::operator int()或 B::operator A()
    return ;              //返回 void 类型，与 main 声明的返回类型不一致
}

```

五解：变量 i 在每条赋值语句执行后的值见注解。

```

z=b.x;        //i=4
i=a.x;        //i=4
i=c.*p;       //i=35
i=a++;        //i=1
i::x+c.y;     //i=37
i=a+b;        //i=-31
//A(4,3)++将 A::x=a.x=b.x=c.x 置为 4，普通成员 a.y=4 不变
b.h(i)=7;     //i=7

```

六解：本题所涉及的概念并不复杂，值得注意的是分数的约分运算，该运算实际上是一个单目运算。在重载约分运算时，要选择只有一个操作数的运算符。另外，在对分数进行加和乘以后，最好对运算结果进行约分运算，如果不约分不算错。函数成员如下。

```

FRACTION::FRACTION(int num, int den){
    numerator=num;
    denominator=den;
}

int FRACTION::operator>(const FRACTION&f)const{
    int d= denominator *f.denominator;
    return (numerator*f.denominator/d >denominator*f.numerator/d);
}

FRACTION FRACTION::operator*( )const{
    int c=cmd(numerator, denominator);
    return FRACTION(numerator/c, denominator/c);
}

FRACTION FRACTION::operator+(const FRACTION&f)const{
    int n= numerator*f.denominator+denominator*f.numerator;
    int d= denominator*f.denominator;
    return *FRACTION(n, d); //对运算结果进行约分运算
}

```

```
}  
FRACTION FRACTION::operator*(const FRACTION&f)const{  
    return *FRACTION(numerator*f.numerator, denominator*f.denominator); //约分  
}
```