第四次作业

1. 有以下程序结构，请分析所有成员在各类的范围内的访问属性。

```
class A //基类
{
    public:
        void f1();
    protected:
        void f2();
    private:
        int i;
};
class B: public A //B为A的公用派生类
{
    public:
        void f3( );
        int k;
    private:
        int m;

};
class C: protected B //C为B的保护派生类
{
    public:
        void f4( );

    protected:
        int n;
    private:
        int p;
};
class D: private C //D为C的私有派生类
{
    public:
        void f5( );

    protected:
        int q;
    private:
        int r;
};
void main()
{
    A a1; //a1是基类A的对象
```

```
    B b1; //b1 是派生类B的对象
    C c1; //c1是派生类C的对象
    D d1; //d1是派生类D的对象
}
```

能否访问	f1	f2	f3	f4	f5	i	k	m	n	p	q	r
a1	✓	✓	x	x	x	✓	x	x	x	x	x	x
b1	✓	✓	✓	x	x	x	✓	✓	x	x	x	x
c1	✓	✓	✓	✓	x	x	✓	x	✓	✓	x	x
d1	✓	✓	✓	✓	✓	x	✓	x	✓	x	✓	✓

2. 分别声明Teacher (教师)类和Cadre (干部)类，采用多重继承方式由这两个类派生出新类Teacher Cadre (教师兼干部)。要求： (1) 在两个基类中都包含姓名、年龄、性别、地址、电话等数据成员。 (2) 在Teacher类中还包含数据成员title (职称)，在Cadre类中还包含数据成员post(职务)。在Teacher Cadre 类中还包含数据成员wages (工资)。 (3) 对两个基类中的姓名、年龄、性别、地址、电话等数据成员用相同的名字，在引用这些数据成员时，指定作用域。 (4) 在类体中声明成员函数，在类外定义成员函数。 (5) 在派生类Teacher Cadre的成员函数show中调用Teacher类中的display函数，输出姓名、年龄、性别、职称、地址、电话，然后再用cout语句输出职务与工资。

3. 有以下程序，请完成下面的工作： (1) 阅读程序，写出运行时输出的结果。 (2) 然后上机运行，验证结果是否正确 (3) 分析程序执行过程，尤其是调用构造函数和析构函数的过程。

```
#include <iostream>
using namespace std;
class A
{
public:
A( ){cout<<"constructing A "<<endl;}

~A( ){cout<<"destructing A "<<endl;}

};
class B:public A
public:
B( ){cout<<"constructing B "<<endl;}

~B( ){cout<<"destructing B "<<endl;}
```

```
};
class C:public B
{public:
C( ){cout<<"constructing C "<<endl;}

~C( ){cout<<"destructing C "<<end;}

};
int main( )
{C c1;

return 0;
}
```

```
constructing A
constructing B
constructing C
destructing C
destructing B
destructing A
```

分析过程：首先初始化C类的一个对象，而在子类的构造函数被调用时，都会调用父类的构造函数，因此依次初始化A,B,C

在销毁对象时，首先执行的是子类的析构函数，然后依次执行基类的析构函数

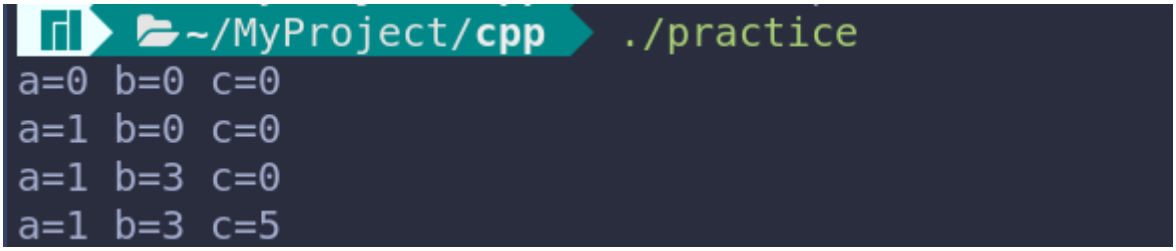
---

4. 有以下程序，请完成下面工作: (1) 阅读程序，写出运行时输出的结果。 (2) 然后上机运行，验证结果是否正确。 (3) 分析程序执行过程，尤其是调用构造函数的过程。

```
#include <iostream>
using namespace std;
class A
{
    public:
        A( ){a=0;b=0;}
```

```
        A(int i){a=i;b=0;}
        A(int i,int j){a=i;b=j;}
        void display( ){cout<<"a="<<a<<" b="<<b;}

    private:
        int a;

        int b;
};
class B: public A
{
    public:
        B( ){c=0;}
        B(int i):A(i){c=0;}
        B(int i,int j):A(i,j){c=0;}
        B(int i,int j,int k):A(i,j){c=k;}

        void display1( )
        {display( );

            cout<<" c="<<c<endl;
        }

    private:
        int c;
};
int main( )
{
    B b1;
    B b2(1);

    B b3(1,3);

    B b4(1,3,5);

    b1.display1( );

    b2.display1( );

    b3.display1( );

    b4.display1( );

    return 0;
}
```

1.
```
a = 0, b = 0, c = 0
a = 1, b = 0, c = 0
a = 1, b = 3, c = 0
a = 1, b = 3, c = 5
```

2. 

3. 分析： b1: 派生类的对象首先执行基类的构造函数，a = 0, b = 0,然后执行子类B无参数的函数c = 0

   b2: 首先执行基类构造函数，并且利用列表初始化字段，A(1)$\Longrightarrow$ a = 1, b= 0,然后执行B(1),c = 0

   b3: 首先执行基类构造函数A(1,3)$\Longrightarrow$ a = 1, b = 3，然后执行B(1,3)c = 0

   b4: 由参数的个数,首先执行基类构造函数A(1,3) ，a = 1, b = 3,然后执行B(1,3,5) c = 5