多态与虚函数例题

题目：水果类与虚函数

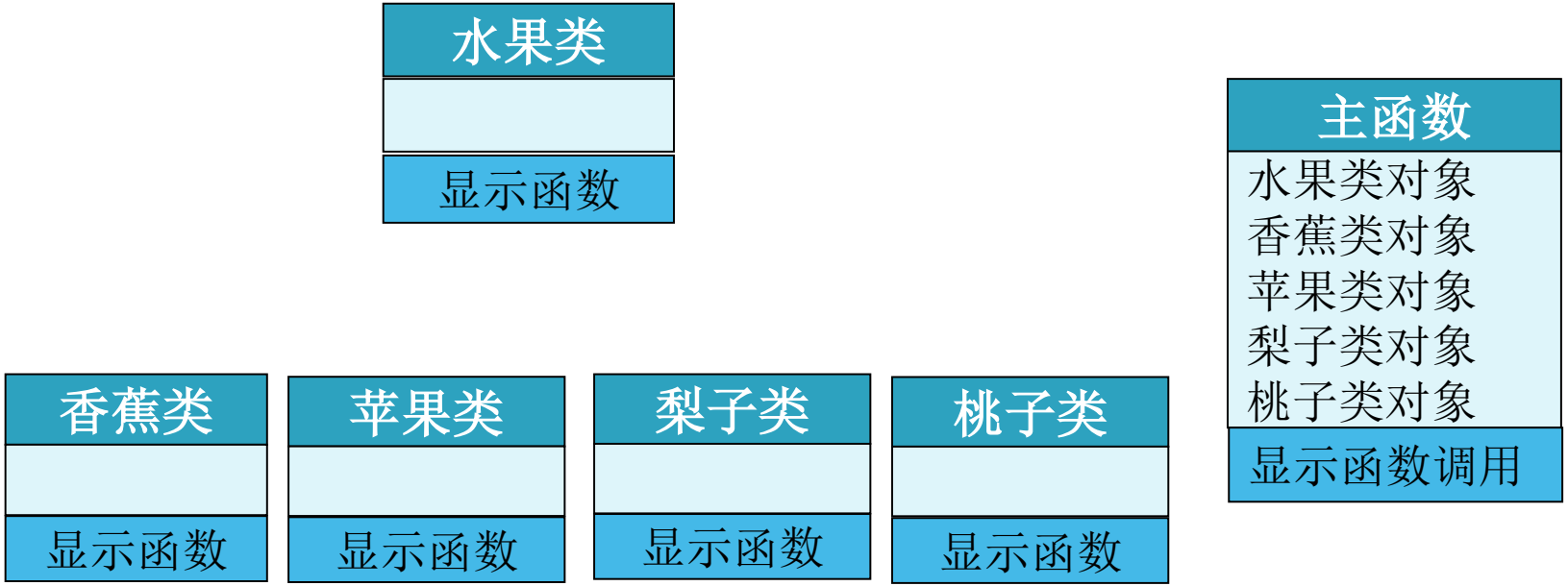
1. 题目内容与要求

① 首先设计一个水果类（Fruit）作为基类，成员函数为显示“水果”函数；

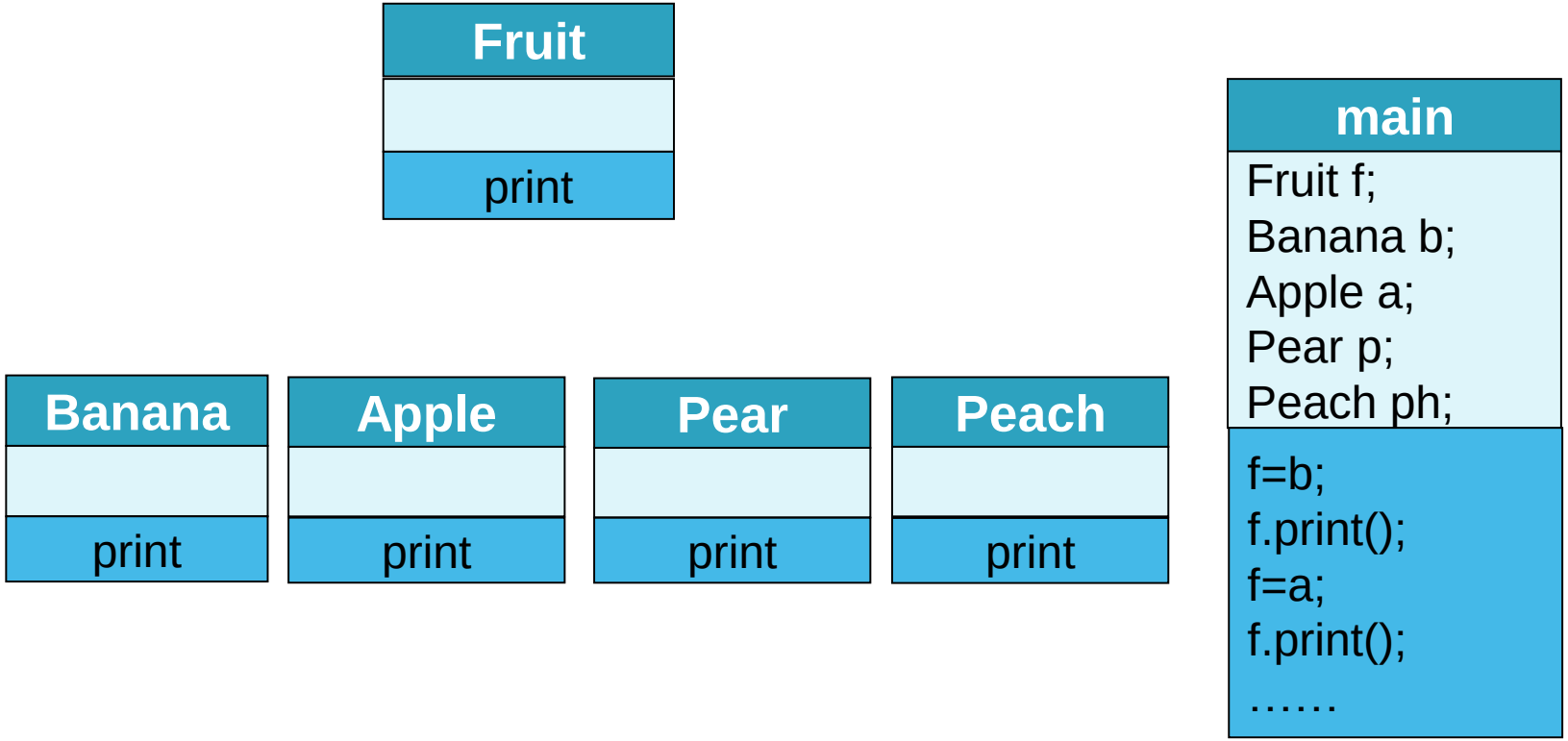
② 然后设计Fruit类的四个派生类：香蕉类（Banana）、苹果类（Apple）、梨子类（Pear）和桃子类（Peach），成员函数分别为显示“香蕉”、“苹果”、“梨子”和“桃子”函数；

③ 最后在主函数中定义这些类的对象，并调用它们的显示函数。

2. 类的分析


水果类
显示函数
主函数
水果类对象
香蕉类对象
苹果类对象
梨子类对象
桃子类对象
显示函数调用
香蕉类
显示函数
苹果类
显示函数
梨子类
显示函数
桃子类
显示函数

3. 类的设计


Fruit
print
main
Fruit f;
Banana b;
Apple a;
Pear p;
Peach ph;
f=b;
f.print();
f=a;
f.print();
……
Banana
print
Apple
print
Pear
print
Peach
print

4. 基类程序代码

```
// 基类: 水果类
class Fruit
{
public:
    void print() {
            cout<< "水果" <<endl;
    }
};
```

派生类程序代码

```
// 派生类1: 香蕉类
class Banana: public Fruit{
public:
    void print() {
            cout<< "香蕉" <<endl;
    }
};
// 派生类2: 苹果类
class Apple: public Fruit{
public:
    void print() {
            cout<< "苹果" <<endl;
    }
};
```

```
// 派生类3: 梨子类
class Pear: public Fruit{
public:
    void print() {
            cout<< "梨子" <<endl;
    }
};
// 派生类4: 桃子类
class Peach: public Fruit{
public:
    void print() {
            cout<< "桃子" <<endl;
    }
};
```

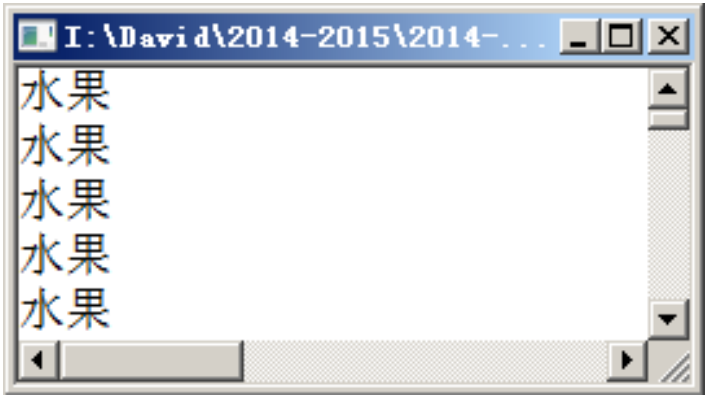
主函数程序代码

```
#include <iostream>
using namespace std;
<5个类的定义在此!>

int main() // 主函数
{
   核心代码在此!

   return 0;
}
```

```
Fruit * pFruit[] = {
        new Fruit(),
        new Banana(), new Apple(),
        new Pear(), new Peach()
};
```

```
for(int i = 0; i < 5; i++) {
                (*pFruit[i]).print();
}
```

5. 运行结果与程序分析



- *从以上5行运行结果来看，似乎调用的都是基类的print函数。*
- *究其原因是派生类函数未能覆盖基类同名函数，从而造成没有机会调用派生类函数的情况。*

6. 延伸思考

问题

怎样实现派生类函数的调用呢？

答案

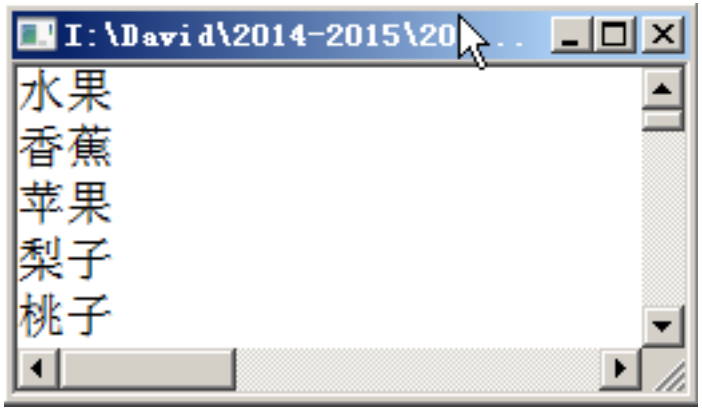
使用virtual修饰符改写基类Fruit的print为虚函数！

详细格式

```
virtual void print ()
```

修改后的基类程序代码

```
// 基类: 水果类
class Fruit
{
public:
virtual void print() {
            cout<< "水果" <<endl;
    }
};
```



7. 小结

- ✓ 虚函数是多态的一种实现形式；
- ✓ 作用是实现函数的覆盖；
- ✓ 写法是将virtual加在函数之前；
- ✓ 今后在类的继承当中的基类尽量多使用虚函数。

例题讲解就到这里！

谢谢！