OOP HW4 错题整理：Virtual Function & Abstract Class

Notes

2026 年 5 月 27 日

总览

本份笔记整理以下错题：1，2，7，10，16，19，22，26–27，28，32，35，37，38，39，44，45，47，50，51，54，55，57，58，62，以及一道虚函数输出分析题。

核心主线：

- **virtual function**：普通虚函数，不一定要被派生类重写。
- **pure virtual function / abstract method**：纯虚函数，若派生类要成为具体类，则必须实现。
- **abstract class**：含有至少一个纯虚函数的类，不能直接创建对象，但可以创建指针或引用。
- **runtime polymorphism**：必须通过基类指针或引用调用虚函数，才体现运行时多态。

Question 1

以下说法正确的是？

- A. 派生类可以和基类有同名成员函数，但是不能有同名成员变量
- **B. 派生类的成员函数中，可以调用基类的同名同参数表的成员函数**
- C. 派生类和基类的同名成员函数必须参数表不同，否则就是重复定义
- D. 派生类和基类的同名成员变量存放在相同存储空间

解析：正确选项是 **B**。派生类中可以定义和基类同名、同参数表的成员函数，这不是重复定义，而是**隐藏**或**重定义**。如果要在派生类成员函数中调用基类版本，可以使用作用域限定符：

```
class Base {
public:
    void show() {}
};

class Derived : public Base {
```

```
public:
    void show() {
        Base::show();    // 调用基类同名同参数函数
    }
};
```

易错点：派生类也可以和基类有同名成员变量，但它们是两个不同的成员，存放在不同的空间。

Question 2

Which among the following can show polymorphism?

- A. Overloading `&&`
- **B. Overloading <<**
- C. Overloading `||`
- D. Overloading `+=`

解析：题库预期答案是 **B**。在很多 OOP 题库中，`<<` 被当作最典型的运算符重载例子，例如：

```
cout << 1;
cout << 3.14;
cout << "hello";
cout << user_defined_object;
```

同一个运算符面对不同类型产生不同效果，因此体现了**重载多态/编译期多态**。

注意：严格 C++ 语法上，`+=` 也可以重载，例如自定义向量类的 `operator+=`。所以本题不是问“哪个能不能被重载”，而是按题库习惯选择最典型的 `<<`。

Question 7

If a virtual member function is defined _____

- A. It should not contain any body and defined by subclasses
- **B. It must contain body and overridden by subclasses**
- C. It must contain body and be overloaded
- D. It must not contain any body and should not be derived

解析：题库语境下选 **B**，但这里的英文表述不严谨。普通虚函数通常可以有函数体：

```
class Base {
public:
    virtual void f() {
        cout << "Base␣f" << endl;
    }
};
```

派生类可以重写它，但**不是必须重写**。如果题目说的是 **pure virtual function**，才是没有函数体、需要派生类实现：

```
class Base {
public:
    virtual void f() = 0;  // pure virtual function
};
```

一句话：普通 `virtual` 可以有函数体；纯虚函数写成 `=0`。

Question 10

What does a virtual function ensure for an object, among the following?

- A. Correct method is called, regardless of the class defining it
- B. Correct method is called, regardless of the object being called
- **C. Correct method is called, regardless of the type of reference used for function call**
- D. Correct method is called, regardless of the type of function being called by objects

解析：正确选项是 **C**。虚函数保证：即使使用**基类指针或引用**调用函数，也会根据对象的实际类型调用正确版本。

```
class Base {
public:
    virtual void f() { cout << "Base" << endl; }
};

class Derived : public Base {
public:
    void f() override { cout << "Derived" << endl; }
};

Base* p = new Derived;
p->f();  // 输出 Derived
```

核心：看的是对象实际类型，而不是指针/引用表面上的类型。

Question 16

Which is a must condition for virtual function to achieve runtime polymorphism?

- A. Virtual function must be accessed with direct name
- B. Virtual functions must be accessed using base class object
- **C. Virtual function must be accessed using pointer or reference**
- D. Virtual function must be accessed using derived class object only

解析：正确选项是 **C**。运行时多态必须通过**基类指针或基类引用**调用虚函数。若直接使用对象调用，通常在编译期已经确定类型，不体现动态绑定。

```
Base* p = new Derived;
p->f();   // runtime polymorphism

Base& r = derived_object;
r.f();    // runtime polymorphism
```

Question 19

It is _____ to redefine the virtual function in derived class.

- A. Necessary
- **B. Not necessary**
- C. Not acceptable
- D. Good practice

解析：正确选项应为 **B**。普通虚函数不一定要在派生类中重写。如果派生类没有重写，就继承并使用基类版本。

```
class Base {
public:
    virtual void f() {
        cout << "Base␣f" << endl;
    }
};

class Derived : public Base {
    // 不重写 f()，仍然可以实例化
};

Derived d;
d.f();  // 调用 Base::f()
```

易错点：如果是纯虚函数 `virtual void f() = 0;`，派生类要成为具体类才必须实现。普通 `virtual` 不必须重写。

Question 22

Which among the following is true?

- A. The abstract functions must be only declared in derived classes
- B. The abstract functions must not be defined in derived classes
- C. The abstract functions must be defined in base and derived class
- **D. The abstract functions must be defined either in base or derived class**

解析：题库一般选 **D**，但选项表述不够严谨。C++ 中抽象函数通常指纯虚函数：

```
virtual void f() = 0;
```

它在基类中通常只声明，不给普通函数体；若派生类要成为具体类，则必须在派生类中实现。

更准确的 C++ 说法：抽象函数在基类中声明为纯虚函数；具体派生类必须实现它。

Question 26–27

Given:

```
class A {
    A() {};
    virtual f() {};
    int i;
};
```

which statement is NOT true?

- A. `i` is private
- B. `f()` is an inline function
- C. `i` is a member of class A
- **D. `sizeof(A) == sizeof(int)`**

解析：不正确的是 **D**。因为类中含有虚函数，通常对象内部会额外包含一个虚函数表指针（vptr）。因此对象大小通常不只是一个 `int` 的大小。

纯虚函数版本：

```
class A {
    A() {}
    virtual f() = 0;
    int i;
};
```

即使 `f()` 是纯虚函数，该类对象布局中通常也仍然需要 vptr。因此 `sizeof(A) == sizeof(int)` 仍然不成立。

注意：实际大小和编译器、平台、对齐规则有关，但通常会大于 `sizeof(int)`。

Question 28

Given:

```
class X {
    int i;
    virtual void f() {};
};
```

If `sizeof(int*) == sizeof(int) == 4`, then `sizeof(X)==?`

- A. 4
- B. 6
- **C. 8**
- D. Undetermined

解析：题库答案是 **C**。因为：

$$\texttt{int i} = 4 \text{ bytes}, \quad \texttt{vptr} = 4 \text{ bytes}$$

所以：

$$\texttt{sizeof(X)} = 4 + 4 = 8$$

注意：这是在题目已经明确 `sizeof(int*) == sizeof(int) == 4` 的前提下。

Question 32

Which among the following is wrong?

- A. `class student{ }; student s;`
- **B. `abstract class student{ }; student s;`**
- C. `abstract class student{ }s[50000000];`
- D. `abstract class student{ }; class toppers: public student{ }; topper t;`

解析：题库通常考察点是：**抽象类不能实例化**。所以 `student s;` 若 `student` 是抽象类，就错误。

注意：这些选项不是标准 C++ 语法，因为 C++ 没有 `abstract class` 这种关键字写法。C++ 中抽象类通过纯虚函数形成：

```
class Student {
public:
    virtual void f() = 0;
};

Student s;  // 错，抽象类不能实例化
```

Question 35

Which among the following best describes abstract classes?

- A. If a class has more than one virtual function, it’s abstract class
- B. If a class have only one pure virtual function, it’s abstract class
- **C. If a class has at least one pure virtual function, it’s abstract class**
- D. If a class has all the pure virtual functions only, then it’s abstract class

解析：正确选项是 **C**。C++ 中只要一个类中有至少一个纯虚函数，它就是抽象类。

```
class A {
public:
    virtual void f() = 0;  // 至少一个纯虚函数
};
```

Question 37

If there is an abstract method in a class then, _____

- **A. Class must be abstract class**
- B. Class may or may not be abstract class
- C. Class is generic
- D. Class must be public

解析：正确选项是 **A**。如果一个类含有抽象方法，即纯虚函数，那么这个类就是抽象类。

```
class A {
public:
    virtual void f() = 0;
};  // A 是抽象类
```

Question 38

If a class is extending/inheriting another abstract class having abstract method, then _____

- **A. Either implementation of method or making class abstract is mandatory**
- B. Implementation of the method in derived class is mandatory
- C. Making the derived class also abstract is mandatory
- D. It’s not mandatory to implement the abstract method of parent class

解析：正确选项是 **A**。派生类继承抽象类后有两种选择：

- 实现所有纯虚函数，成为具体类；
- 不完全实现，继续作为抽象类。

```
class A {
public:
    virtual void f() = 0;
};

class B : public A {
    // 不实现 f()，所以 B 仍然是抽象类
};

class C : public A {
public:
    void f() override {}  // C 实现了 f()，C 可以实例化
};
```

Question 39

Abstract class A has 4 virtual functions. Abstract class B defines only 2 of those member functions as it extends class A. Class C extends class B and implements the other two member functions of class A. Choose the correct option below.

- A. Program won’t run as all the methods are not defined by B
- B. Program won’t run as C is not inheriting A directly
- C. Program won’t run as multiple inheritance is used
- **D. Program runs correctly**

解析：正确选项是 **D**。纯虚函数可以分层实现，不要求第一层派生类一次性全部实现。只要最终要实例化的类实现了全部纯虚函数，就可以创建对象。

```
class A {
public:
    virtual void f1() = 0;
    virtual void f2() = 0;
    virtual void f3() = 0;
    virtual void f4() = 0;
};

class B : public A {
public:
    void f1() override {}
    void f2() override {}
    // f3 和 f4 没有实现，所以 B 仍然是抽象类
};

class C : public B {
public:
    void f3() override {}
    void f4() override {}
    // f1、f2 来自 B；f3、f4 由 C 实现
};

int main() {
    // A a;  // 错
    // B b;  // 错
    C c;     // 对
}
```

Question 44

The abstract classes in Java can _____

- **A. Implement constructors**
- B. Can’t implement constructor
- C. Can implement only unimplemented methods
- D. Can’t implement any type of constructor

解析：正确选项是 **A**。Java 抽象类不能直接实例化，但它仍然可以有构造器。构造器会在创建派生类对象时被调用，用来初始化抽象父类部分。

C++ 类比：C++ 抽象类也可以有构造函数：

```
class Base {
public:
```

```
    Base() { cout << "Base constructor" << endl; }
    virtual void f() = 0;
};
```

Question 45

It is _____ to have an abstract method.

- A. Not mandatory for an static class
- B. Not mandatory for a derived class
- **C. Not mandatory for an abstract class**
- D. Not mandatory for parent class

解析：题库语境偏 Java，正确选项是 **C**。Java 中抽象类不一定必须包含抽象方法。例如一个类可以被声明为 abstract，但里面没有 abstract method。

C++ 注意：C++ 中没有 `abstract` 关键字；一个类成为抽象类的条件就是至少有一个纯虚函数。因此在 C++ 语境下，“抽象类是否必须有抽象方法”应理解为：是的，至少有一个纯虚函数。

Question 47

If single level inheritance is used and an abstract class is created with some undefined functions, can its derived class also skip some definitions?

- **A. Yes, always possible**
- B. Yes, possible if only one undefined function
- C. No, at least 2 undefined functions must be there
- D. No, the derived class must implement those methods

解析：正确选项是 **A**。派生类可以继续不实现部分纯虚函数，但代价是它自己仍然是抽象类，不能实例化。

```
class A {
public:
    virtual void f() = 0;
};

class B : public A {
    // 可以不实现 f()
    // 但 B 仍然是抽象类
};
```

Question 50

Is it compulsory to have constructor for all the classes involved in multiple inheritance?

- A. Yes, always
- B. Yes, only if no abstract class is involved
- **C. No, only classes being used should have a constructor**
- D. No, they must not contain constructors

解析：题库答案通常是 **C**。这题的关键是：**不要求所有类都手动写构造函数**。如果没有显式写构造函数，编译器会尝试生成默认构造函数。

```
class A {};
class B {};

class C : public A, public B {};

int main() {
    C c;  // 可以，编译器自动生成默认构造函数
}
```

易错点：默认构造函数当然也算构造函数。更准确说法是：创建对象时，所有基类子对象和成员对象都必须能够被成功构造，但不一定要程序员显式写构造函数。

Question 51

Which among the following best defines the abstract methods?

- A. Functions declared and defined in base class
- **B. Functions only declared in base class**
- C. Function which may or may not be defined in base class
- D. Function which must be declared in derived class

解析：题库答案是 **B**。抽象方法通常指在基类中只声明、没有普通实现的函数。在 C++ 中对应纯虚函数：

```
class Base {
public:
    virtual void f() = 0;
};
```

注意：C++ 允许纯虚函数有类外定义，但这属于高级细节；考试题一般按“只声明不实现”来理解。

Question 54

It is _____ to define the abstract functions.

- A. Mandatory for all the classes in program
- B. Necessary for all the base classes
- C. Necessary for all the derived classes
- **D. Not mandatory for all the derived classes**

解析：正确选项是 **D**。不是所有派生类都必须实现抽象函数，因为中间派生类可以继续保持抽象。

```
class A {
public:
    virtual void f() = 0;
};

class B : public A {
    // 不实现 f(), B 仍为抽象类
};

class C : public B {
public:
    void f() override {}  // C 成为具体类
};
```

Question 55

The abstract function definitions in derived classes is enforced at _____

- A. Runtime
- **B. Compile time**
- C. Writing code time
- D. Interpreting time

解析：正确选项是 **B**。如果一个类还没有实现所有纯虚函数，却尝试创建对象，编译器会直接报错。因此这是编译期检查。

```
class A {
public:
    virtual void f() = 0;
};
```

```
class B : public A {};

int main() {
    B b;  // 编译期报错：B 仍然是抽象类
}
```

Question 57

If a function declared as abstract in base class doesn't have to be defined in derived class then _____

- A. Derived class must define the function anyhow
- **B. Derived class should be made abstract class**
- C. Derived class should not derive from that base class
- D. Derived class should not use that function

解析：正确选项是 **B**。如果派生类不想实现基类中的抽象函数，那么这个派生类自身也必须保持抽象，不能创建对象。

Question 58

Which among the following is true?

- A. Abstract methods can be static
- B. Abstract methods can be defined in derived class
- **C. Abstract methods must not be static**
- D. Abstract methods can be made static in derived class

解析：正确选项是 **C**。抽象方法依赖虚函数机制，而 `static` 成员函数不属于某个对象，没有 `this` 指针，不能参与虚函数动态绑定。

```
class Base {
public:
    static virtual void f() = 0;  // 错
};
```

一句话：`virtual` 靠对象动态绑定；`static` 不依赖对象，所以二者不能同时用。

Question 62

The abstract method definition can be made _____ in derived class.

- A. Private
- B. Protected
- C. Public
- **D. Private, public, or protected**

解析：正确选项是 **D**。C++ 中判断是否重写，主要看函数名、参数表、const 修饰、返回类型是否兼容，不看访问权限。因此基类中的 public 纯虚函数，可以在派生类中被 private/protected/public 方式重写。

```
class Base {
public:
    virtual void f() = 0;
};

class Derived : public Base {
private:
    void f() override {
        cout << "Derived␣f" << endl;
    }
};
```

注意：访问权限只影响能不能直接调用，不影响是否构成 override。

```
Derived d;
// d.f();  // 错，因为 Derived::f 是 private

Base* p = &d;
p->f();   // 可以，因为 Base::f 是 public，运行时绑定到 Derived::f
```

补充题：虚函数、const 与隐藏

题目代码：

```
#include<iostream>
using namespace std;

class Base{
protected:
    int x;
public:
    Base(int b=0): x(b) { }
    virtual void display() const {cout << x << endl;}
};

class Derived: public Base{
    int y;
public:
    Derived(int d=0): y(d) { }
    void display() {cout << x << "," << y << endl;}
};

int main()
{
  Base b(1);
  Derived d(2);
  Base *p = &d;
  b.display();
  d.display();
  p->display();
  return 0;
}
```

输出：

```
1
0,2
0
```

解析 1：Base b(1)

```
Base b(1);
```

调用 Base(int b=0): x(b)，所以 b.x = 1。因此：

```
b.display();
```

调用 Base::display() const，输出：

```
1
```

解析 2：Derived d(2)

```
Derived(int d=0): y(d) { }
```

这里只初始化了 y，没有显式调用基类构造函数，所以基类部分自动调用：

```
Base()
```

也就是 x = 0。同时 y = 2。因此：

```
d.display();
```

调用 Derived::display()，输出：

```
0,2
```

解析 3：为什么 p->display() 输出 0?

基类函数是：

```
virtual void display() const
```

派生类函数是：

```
void display()
```

注意:基类版本有 const,派生类版本没有 const。因此它们的函数签名不同,Derived::display() 并没有真正 override Base::display() const，而只是隐藏了基类同名函数。

所以：

```
Base *p = &d;
p->display();
```

调用的仍然是 Base::display() const。由于 d 的基类部分 x = 0，所以输出：

```
0
```

正确重写写法

如果想让它真正发生多态，应写成：

```
class Derived: public Base{
    int y;
public:
    Derived(int d=0): Base(d), y(d) { }
```

```
    void display() const override {
        cout << x << "," << y << endl;
    }
};
```

这样：

- const 保持一致；
- 使用 override 让编译器检查是否真正重写；
- Base(d) 让基类部分的 x 也初始化为 d。

此时输出会变为：

```
1
2,2
2,2
```