编程报告

编程报告
第五周

丁进仁 戴铭瑞

第五周“多态与虚函数”

学生姓名：丁进仁 戴铭瑞

负责部分：丁　算法设计、编程实现、结果分析

戴　问题分析、解决方案、结果分析、体会总结

注：以下程序皆已通过 OpenJudge 测试

一． Fun 和 Do

（一） 问题分析

问题：程序填空输出指定结果

```
#include <iostream>
using namespace std;
class A {
    private:
    int nVal;
    public:
    void Fun()
    { cout << "A::Fun" << endl; };
    void Do()
    { cout << "A::Do" << endl; }
};
class B:public A {
    public:
    virtual void Do()
    { cout << "B::Do" << endl;}
};
class C:public B {
    public:
    void Do( )
    { cout <<"C::Do"<<endl; }
    void Fun()
    { cout << "C::Fun" << endl; }
};
void Call(
// 在此处补充你的代码
) {
    p.Fun(); p.Do();
}
int main() {
    C c;
    Call( c);
    return 0;
}
```

Input:

无

Output:

A::Fun

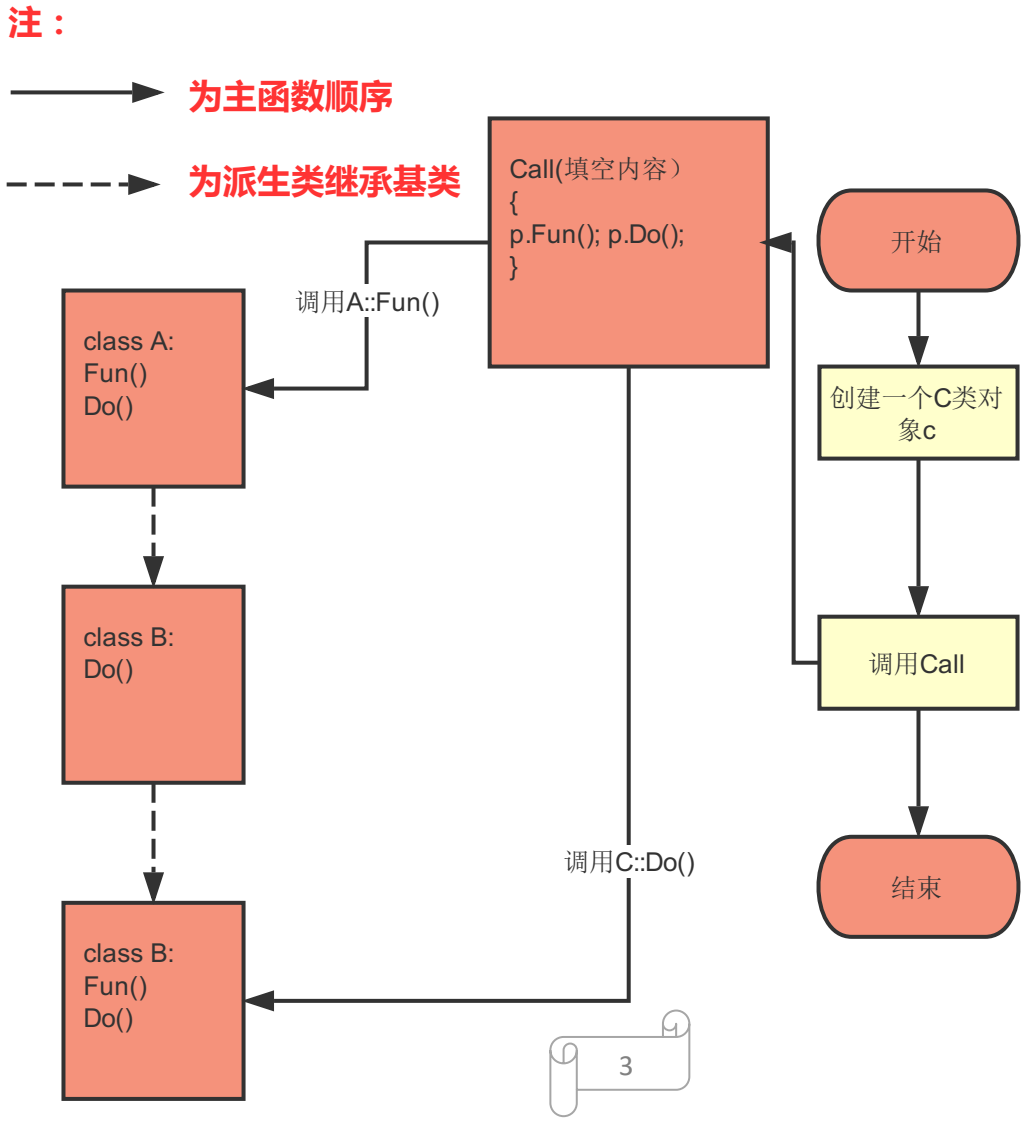
C::Do

分析：对于这种填空类题目，要先从 main 函数开始往下看，寻找代码中所缺失的函数，然后在补齐。

（二） 解决方案

方案：从输出和 Call 函数来看，这道题肯定是多态了，所以只要用基类的引用引用派生类对象即可。

（三）算法设计

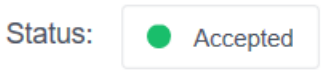


（四） 编程实现

```
B &p
```

（五） 结果分析

```
Input:
None
Output:
A::Fun
C::Do
```



Openjudge 成功通过

（六） 体会总结

体会：这题就是对多态概念和应用的测试，很基础的内容，没有什么好说的。

二． 这是什么鬼 delete

（一） 问题分析

问题：程序填空，输出指定结果

```
#include <iostream>
using namespace std;
class A
{
public:
    A() { }
// 在此处补充你的代码
};
class B:public A {
    public:
    ~B() { cout << "destructor B" << endl; }
```

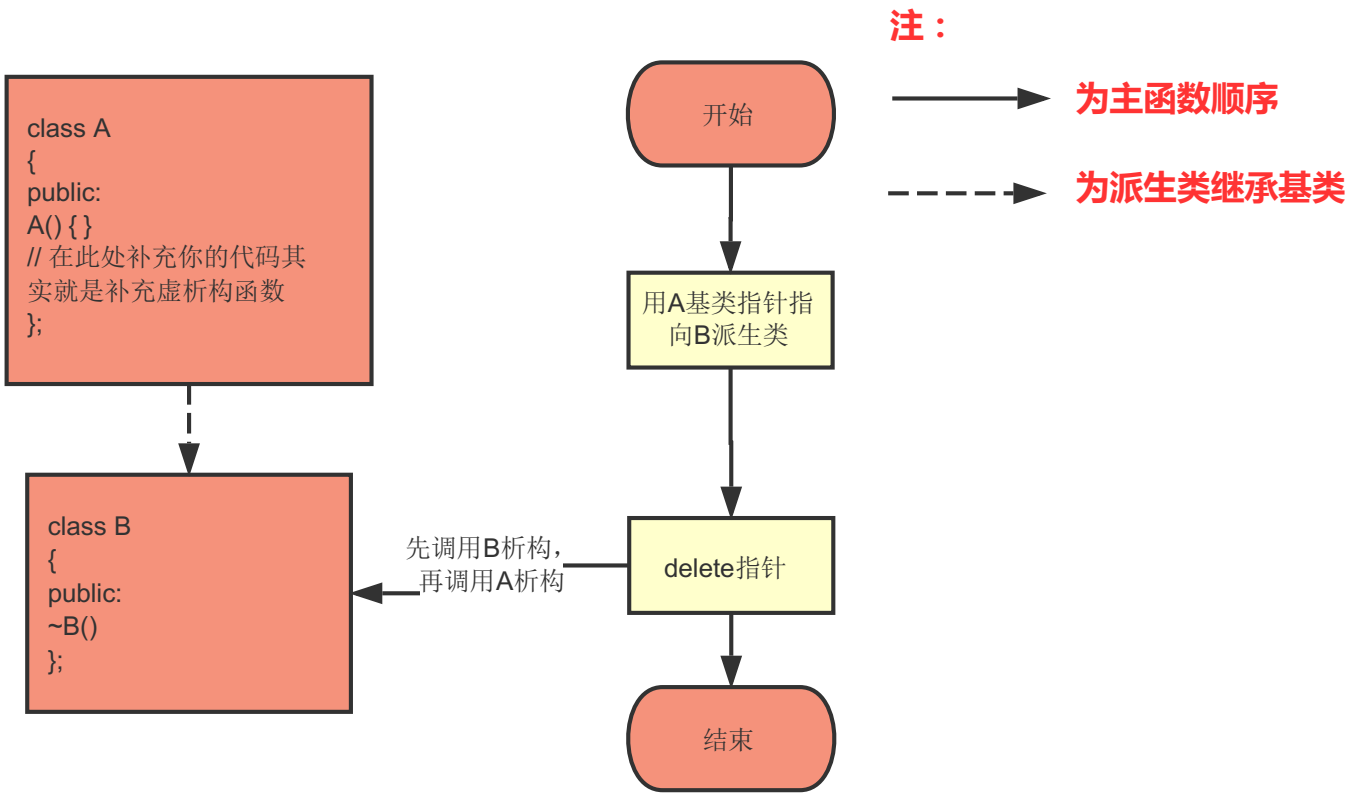
```
};
int main()
{
    A * pa;
    pa = new B;
    delete pa;
    return 0;
}
```

分析：还是从 main 函数开始寻找缺失的函数，再补齐它们。

（二） 解决方案

方案：这一题是对虚析构函数的应用，也是对多态的应用，答案是唯一的。

（三） 算法设计



（四） 编程实现

```
virtual ~A() {cout<<"destructor A"<<endl;};
```

（五） 结果分析

Input :	Output :
无	destructor B
	destructor A

Openjudge 成功通过

（六） 体会总结

体会：这道题和上一题一样考察的是多态的概念和基本应用，只要好好看视频学习都会写，没有什么好说的。

三． 魔兽世界之二：装备

（一） 问题分析

问题：题目太长了，就不复制粘贴了，简单的说下题目描述。本题较上一题魔兽世界之一：备战新增了武器，dragon 的士气，lion 的忠诚度，同样还是输出备战内容，但输出内容多了一些特别的输出内容。

```
Sample Input 1:
1
20
3 4 5 6 7
```

Sample Output 1:

```
Case:1
000 red iceman 1 born with strength 5,1 iceman in red headquarter
It has a bomb
000 blue lion 1 born with strength 6,1 lion in blue headquarter
It's loyalty is 14
001 red lion 2 born with strength 6,1 lion in red headquarter
It's loyalty is 9
001 blue dragon 2 born with strength 3,1 dragon in blue headquarter
It has a arrow,and it's morale is 3.67
002 red wolf 3 born with strength 7,1 wolf in red headquarter
002 blue ninja 3 born with strength 4,1 ninja in blue headquarter
It has a sword and a bomb
003 red headquarter stops making warriors
003 blue iceman 4 born with strength 5,1 iceman in blue headquarter
It has a bomb
004 blue headquarter stops making warriors
```

分析：既然题目的要求相比上一题魔兽世界之一：备战添加了以下新功能

1） dragon 的武器和士气

2） ninja 的两把武器

3） iceman 的武器

4） lion 的忠诚度

5） 输出的新内容

（二） 解决方案

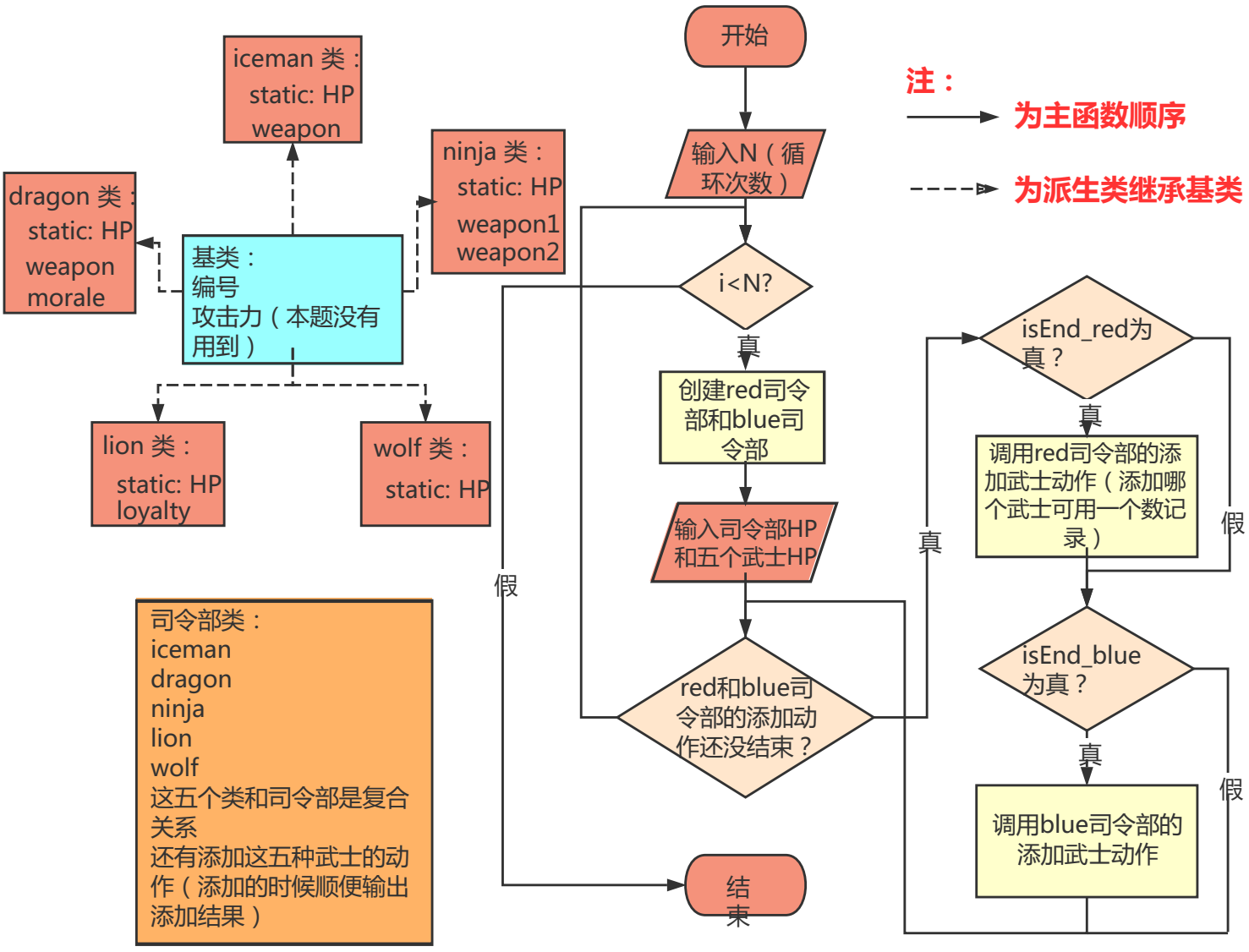
方案：既然题目的要求添加了这些新功能，那么我们也只要在上一题的代码中添加上这些新功能就 OK 了。

具体的方案就是：

1） 在对应武士的类中加上 武器 weapon，士气 morale，忠诚度 loyalty

2） 然后在司令部类中的对应添加武士动作中加上这些属性的添加行为就完成了。

（三） 算法设计



（四） 编程实现

因为代码太长了所以放在另一个附录文档

（五） 结果分析

Sample Input 1:

```
1
20
```

```
3 4 5 6 7
```

Sample Output 1:

```
Case:1
000 red iceman 1 born with strength 5,1 iceman in red headquarter
It has a bomb
000 blue lion 1 born with strength 6,1 lion in blue headquarter
It's loyalty is 14
001 red lion 2 born with strength 6,1 lion in red headquarter
It's loyalty is 9
001 blue dragon 2 born with strength 3,1 dragon in blue headquarter
It has a arrow,and it's morale is 3.67
002 red wolf 3 born with strength 7,1 wolf in red headquarter
002 blue ninja 3 born with strength 4,1 ninja in blue headquarter
It has a sword and a bomb
003 red headquarter stops making warriors
003 blue iceman 4 born with strength 5,1 iceman in blue headquarter
It has a bomb
004 blue headquarter stops making warriors004 blue headquarter stops making warriors
```

Openjudge 成功通过

（六） 体会总结

体会：这道题是在上一题的基础上增添一些新功能，按照常理来说不用半小时就可以解决了。但我在实际处理的时候才发现我上一次代码中写的复合关系根本就不对!!! new 运算符根本就不能追加空间！能通过根本就是侥幸！

在经过调查后在网上发现运用 malloc 运算符和 memcpy 函数可以达到目的，但网上的例子是针对 int 数组的，并没有说明对象数组能否实现。

于是我继续在网上调查，发现要对对象数组进行 memcpy 操作必须要是 POD 类型的类，我的类明显不符合，因此这条路走不通了。

既然新的路走不通，new 运算符又不能追加空间，只能试着改变思路了。归根究底再数组末尾追加空间是要先 new 一个更长的数组，再将旧数组搬过去。但 类* 类型的指针只能 new 无初始化的对象 才能 new 多个，有初始化的话就得逐个进行初始化。所以现有的结构肯定不能满足需求，所以我改变思路，对象不能 new 多个，但对象的指针可以 new 多个，因此我的复合关系就变成了 command 类中有五个 武士类** 类型的指针了。一切问题迎刃而解。