

06 | 理论三：面向对象相比面向过程有哪些优势？面向过程真的过时了吗？

王争 · 设计模式之美



在上两节课中，我们讲了面向对象这种现在非常流行的编程范式，或者说编程风格。实际上，除了面向对象之外，被大家熟知的编程范式还有另外两种，面向过程编程和函数式编程。面向过程这种编程范式随着面向对象的出现，已经慢慢退出了舞台，而函数式编程目前还没有被广泛接受。

在专栏中，我不会对函数式编程做讲解，但我会花两节课的时间，讲一下面向过程这种编程范式。你可能会问，既然面向对象已经成为主流的编程范式，而面向过程已经不那么推荐使用，那为什么又要浪费时间讲它呢？

那是因为在过往的工作中，我发现很多人搞不清楚面向对象和面向过程的区别，总以为使用面向对象编程语言来做开发，就是在进行面向对象编程了。而实际上，他们只是在用面向对象编程语言，编写面向过程风格的代码而已，并没有发挥面向对象编程的优势。这就相当于手握一把屠龙刀，却只是把它当作一把普通的刀剑来用，相当可惜。

所以，我打算详细对比一下面向过程和面向对象这两种编程范式，带你一块搞清楚下面这几个问题（前三个问题我今天讲解，后三个问题我放到下一节课中讲解）：

1. 什么是面向过程编程与面向过程编程语言？
2. 面向对象编程相比面向过程编程有哪些优势？
3. 为什么说面向对象编程语言比面向过程编程语言更高级？
4. 有哪些看似是面向对象实际是面向过程风格的代码？
5. 在面向对象编程中，为什么容易写出面向过程风格的代码？
6. 面向过程编程和面向过程编程语言就真的无用武之地了吗？

话不多说，带着这几个问题，我们就正式开始今天的学习吧！

什么是面向过程编程与面向过程编程语言？

如果你是一名比较资深的程序员，最开始学习编程的时候，接触的是 Basic、Pascal、C 等面向过程的编程语言，那你对这两个概念肯定不陌生。但如果你是新生代的程序员，一开始学编程的时候，接触的就是面向对象编程语言，那你对这两个概念可能会比较不熟悉。所以，在对比面向对象与面向过程优劣之前，我们先把面向过程编程和面向过程编程语言这两个概念搞清楚。

实际上，我们可以对比着面向对象编程和面向对象编程语言这两个概念，来理解面向过程编程和面向过程编程语言。还记得我们之前是如何定义面向对象编程和面向对象编程语言的吗？让我们一块再来回顾一下。

面向对象编程是一种编程范式或编程风格。它以类或对象作为组织代码的基本单元，并将封装、抽象、继承、多态四个特性，作为代码设计和实现的基石。

面向对象编程语言是支持类或对象的语法机制，并有现成的语法机制，能方便地实现面向对象编程四大特性（封装、抽象、继承、多态）的编程语言。

类比面向对象编程与面向对象编程语言的定义，对于面向过程编程和面向过程编程语言这两个概念，我给出下面这样的定义。


面向过程编程也是一种编程范式或编程风格。它以过程（可以理解为方法、函数、操作）作为组织代码的基本单元，以数据（可以理解为成员变量、属性）与方法相分离为最主要的特点。面向过程风格是一种流程化的编程风格，通过拼接一组顺序执行的方法来操作数据完成一项功能。

面向过程编程语言首先是一种编程语言。它最大的特点是不支持类和对象两个语法概念，不支持丰富的面向对象编程特性（比如继承、多态、封装），仅支持面向过程编程。

不过，这里我必须声明一下，就像我们在之前讲到的，面向对象编程和面向对象编程语言并没有官方的定义一样，这里我给出的面向过程编程和面向过程编程语言的定义，也并不是严格的官方定义。之所以要给出这样的定义，只是为了跟面向对象编程及面向对象编程语言做个对比，以方便你理解它们的区别。

定义不是很严格，也比较抽象，所以，我再用一个例子进一步解释一下。假设我们有一个记录了用户信息的文本文件 `users.txt`，每行文本的格式是 `name&age&gender`（比如，小王&28&男）。我们希望写一个程序，从 `users.txt` 文件中逐行读取用户信息，然后格式化成 `name\tage\tgender`（其中，`\t` 是分隔符）这种文本格式，并且按照 `age` 从小到大排序之后，重新写入到另一个文本文件 `formatted_users.txt` 中。针对这样一个小程序的开发，我们一块来看看，用面向过程和面向对象两种编程风格，编写出来的代码有什么不同。

首先，我们先来看，用面向过程这种编程风格写出来的代码是什么样子的。注意，下面的代码是用 C 语言这种面向过程的编程语言来编写的。

 复制代码


```
1 struct User {
2     char name[64];
3     int age;
4     char gender[16];
5 };
6
7 struct User parse_to_user(char* text) {
8     // 将text("小王&28&男")解析成结构体struct User
9 }
10
11 char* format_to_text(struct User user) {
12     // 将结构体struct User格式化成本 ("小王\t28\t男")
13 }
14
```

```

15 void sort_users_by_age(struct User users[]) {
16     // 按照年龄从小到大排序users
17 }
18
19 void format_user_file(char* origin_file_path, char* new_file_path) {
20     // open files...
21     struct User users[1024]; // 假设最大1024个用户
22     int count = 0;
23     while(1) { // read until the file is empty
24         struct User user = parse_to_user(line);
25         users[count++] = user;
26     }
27
28     sort_users_by_age(users);
29
30     for (int i = 0; i < count; ++i) {
31         char* formatted_user_text = format_to_text(users[i]);
32         // write to new file...
33     }
34     // close files...
35 }
36
37 int main(char** args, int argv) {
38     format_user_file("/home/zheng/user.txt", "/home/zheng/formatted_users.txt");
39 }

```

然后，我们再来看，用面向对象这种编程风格写出来的代码是什么样子的。注意，下面的代码是用 Java 这种面向对象的编程语言来编写的。

 复制代码

```

1  public class User {
2      private String name;
3      private int age;
4      private String gender;
5
6      public User(String name, int age, String gender) {
7          this.name = name;
8          this.age = age;
9          this.gender = gender;
10     }
11
12     public static User praseFrom(String userInfoText) {
13         // 将text(“小王&28&男”)解析成类User
14     }

```

```

15
16     public String formatToText() {
17         // 将类User格式化成文本 ("小王\t28\t男")
18     }
19 }
20
21 public class UserFileFormatter {
22     public void format(String userFile, String formattedUserFile) {
23         // Open files...
24         List users = new ArrayList<>();
25         while (1) { // read until file is empty
26             // read from file into userText...
27             User user = User.parseFrom(userText);
28             users.add(user);
29         }
30         // sort users by age...
31         for (int i = 0; i < users.size(); ++i) {
32             String formattedUserText = user.formatToText();
33             // write to new file...
34         }
35         // close files...
36     }
37 }
38
39 public class MainApplication {
40     public static void main(String[] args) {
41         UserFileFormatter userFileFormatter = new UserFileFormatter();
42         userFileFormatter.format("/home/zheng/users.txt", "/home/zheng/formatted_user
43     }
44 }

```

从上面的代码中，我们可以看出，面向过程和面向对象最基本的区别就是，代码的组织方式不同。面向过程风格的代码被组织成了一组方法集合及其数据结构（struct User），方法和数据结构的定义是分开的。面向对象风格的代码被组织成一组类，方法和数据结构被绑定一起，定义在类中。

看完这个例子之后，你可能会说，面向对象编程和面向过程编程，两种风格的区别就这么一点吗？当然不是，对于这两种编程风格的更多区别，我们继续往下看。

面向对象编程相比面向过程编程有哪些优势？

刚刚我们介绍了面向过程编程及面向过程编程语言的定义，并跟面向对象编程及面向对象编程语言做了一个简单对比。接下来，我们再来看一下，为什么面向对象编程晚于面向过程编程出现，却能取而代之，成为现在主流的编程范式？面向对象编程跟面向过程编程比起来，到底有哪些优势？

1.OOP 更加能够应对大规模复杂程序的开发

看了刚刚举的那个格式化文本文件的例子，你可能会有这样的疑问，两种编程风格实现的代码貌似差不多啊，顶多就是代码的组织方式有点区别，没有感觉到面向对象编程有什么明显的优势呀！你的感觉没错。之所以有这种感觉，主要原因是这个例子程序比较简单、不够复杂。

对于简单程序的开发来说，不管是用面向过程编程风格，还是用面向对象编程风格，差别确实不会很大，甚至有的时候，面向过程的编程风格反倒更有优势。因为需求足够简单，整个程序的处理流程只有一条主线，很容易被划分成顺序执行的几个步骤，然后逐句翻译成代码，这就非常适合采用面向过程这种面条式的编程风格来实现。

但对于大规模复杂程序的开发来说，整个程序的处理流程错综复杂，并非只有一条主线。如果把整个程序的处理流程画出来的话，会是一个网状结构。如果我们再用面向过程编程这种流程化、线性的思维方式，去翻译这个网状结构，去思考如何把程序拆解为一组顺序执行的方法，就会比较吃力。这个时候，面向对象的编程风格的优势就比较明显了。

面向对象编程是以类为思考对象。在进行面向对象编程的时候，我们并不是一上来就去思考，如何将复杂的流程拆解为一个一个方法，而是采用曲线救国的策略，先去思考如何给业务建模，如何将需求翻译为类，如何给类之间建立交互关系，而完成这些工作完全不需要考虑错综复杂的处理流程。当我们有了类的设计之后，然后再像搭积木一样，按照处理流程，将类组装起来形成整个程序。这种开发模式、思考问题的方式，能让我们在应对复杂程序开发的时候，思路更加清晰。

除此之外，面向对象编程还提供了一种更加清晰的、更加模块化的代码组织方式。比如，我们开发一个电商交易系统，业务逻辑复杂，代码量很大，可能要定义数百个函数、数百个数据结构，那如何分门别类地组织这些函数和数据结构，才能不至于看起来比较凌乱呢？类就是一种非常好的组织这些函数和数据结构的方式，是一种将代码模块化的有效手段。

你可能会说，像 C 语言这种面向过程的编程语言，我们也可以按照功能的不同，把函数和数据结构放到不同的文件里，以达到给函数和数据结构分类的目的，照样可以实现代码的模块化。你说得没错。只不过面向对象编程本身提供了类的概念，强制你做这件事情，而面向过程编程并不强求。这也算是面向对象编程相对于面向过程编程的一个微创新吧。

实际上，利用面向过程的编程语言照样可以写出面向对象风格的代码，只不过可能会比用面向对象编程语言来写面向对象风格的代码，付出的代价要高一些。而且，面向过程编程和面向对象编程并非完全对立的。很多软件开发中，尽管利用的是面向过程的编程语言，也都有借鉴面向对象编程的一些优点。

2.OOP 风格的代码更易复用、易扩展、易维护

在刚刚的那个例子中，因为代码比较简单，所以只用到到了类、对象这两个最基本的面向对象概念，并没有用到更加高级的四大特性，封装、抽象、继承、多态。因此，面向对象编程的优势其实并没有发挥出来。

面向过程编程是一种非常简单的编程风格，并没有像面向对象编程那样提供丰富的特性。而面向对象编程提供的封装、抽象、继承、多态这些特性，能极大地满足复杂的编程需求，能方便我们写出更易复用、易扩展、易维护的代码。为什么这么说呢？还记得我们在上一节课中讲到的封装、抽象、继承、多态存在的意义吗？我们再来简单回顾一下。

首先，我们先来看下封装特性。封装特性是面向对象编程相比于面向过程编程的一个最基本的区别，因为它基于的是面向对象编程中最基本的类的概念。面向对象编程通过类这种组织代码的方式，将数据和方法绑定在一起，通过访问权限控制，只允许外部调用者通过类暴露的有限方法访问数据，而不会像面向过程编程那样，数据可以被任意方法随意修改。因此，面向对象编程提供的封装特性更有利于提高代码的易维护性。

其次，我们再来看下抽象特性。我们知道，函数本身就是一种抽象，它隐藏了具体的实现。我们在使用函数的时候，只需要了解函数具有什么功能，而不需要了解它是怎么实现的。从这一点上，不管面向过程编程还是面向对象编程，都支持抽象特性。不过，面向对象编程还提供了其他抽象特性的实现方式。这些实现方式是面向过程编程所不具备的，比如基于接口实现的抽象。基于接口的抽象，可以让我们在不改变原有实现的情况下，轻松替换新的实现逻辑，提高了代码的可扩展性。

再次，我们来看下继承特性。继承特性是面向对象编程相比于面向过程编程所特有的两个特性之一（另一个是多态）。如果两个类有一些相同的属性和方法，我们就可以将这些相同的代码，抽取到父类中，让两个子类继承父类。这样两个子类也就可以重用父类中的代码，避免了代码重复写多遍，提高了代码的复用性。

最后，我们来看下多态特性。基于这个特性，我们在需要修改一个功能实现的时候，可以通过实现一个新的子类的方式，在子类中重写原来的功能逻辑，用子类替换父类。在实际的代码运行过程中，调用子类新的功能逻辑，而不是在原有代码上做修改。这就遵从了“对修改关闭、对扩展开放”的设计原则，提高代码的扩展性。除此之外，利用多态特性，不同的类对象可以传递给相同的方法，执行不同的代码逻辑，提高了代码的复用性。

所以说，基于这四大特性，利用面向对象编程，我们可以更轻松地写出易复用、易扩展、易维护的代码。当然，我们不能说，利用面向过程风格就不可以写出易复用、易扩展、易维护的代码，但没有四大特性的帮助，付出的代价可能就要高一些。

3.OOP 语言更加人性化、更加高级、更加智能

人类最开始跟机器打交道是通过 0、1 这样的二进制指令，然后是汇编语言，再之后才出现了高级编程语言。在高级编程语言中，面向过程编程语言又早于面向对象编程语言出现。之所以先出现面向过程编程语言，那是因为跟机器交互的方式，从二进制指令、汇编语言到面向过程编程语言，是一个非常自然的过渡，都是一种流程化的、面条式的编程风格，用一组指令顺序操作数据，来完成一项任务。

从指令到汇编再到面向过程编程语言，跟机器打交道的方式在不停地演进，从中我们很容易发现这样一条规律，那就是编程语言越来越人性化，让人跟机器打交道越来越容易。笼统点讲，就是编程语言越来越高级。实际上，在面向过程编程语言之后，面向对象编程语言的出现，也顺应了这样的发展规律，也就是说，面向对象编程语言比面向过程编程语言更加高级！

跟二进制指令、汇编语言、面向过程编程语言相比，面向对象编程语言的编程套路、思考问题的方式，是完全不一样的。前三者是一种计算机思维方式，而面向对象是一种人类的思维方式。我们在用前面三种语言编程的时候，我们是在思考，如何设计一组指令，告诉机器去执行这组指令，操作某些数据，帮我们完成某个任务。而在进行面向对象编程时候，我们是在思考，如何给业务建模，如何将真实的世界映射为类或者对象，这让我们更加能聚焦到业务本

身，而不是思考如何跟机器打交道。可以这么说，越高级的编程语言离机器越“远”，离我们人类越“近”，越“智能”。

这里多聊几句，顺着刚刚这个编程语言的发展规律来想，如果一种新的突破性的编程语言出现，那它肯定是更加“智能”的。大胆想象一下，使用这种编程语言，我们可以无需对计算机知识有任何了解，无需像现在这样一行一行地敲很多代码，只需要把需求文档写清楚，就能自动生成我们想要的软件了。

重点回顾

今天的内容就讲完了，我们来一起总结回顾一下，你需要重点掌握的几个知识点。

1. 什么是面向过程编程？什么是面向过程编程语言？

实际上，面向过程编程和面向过程编程语言并没有严格的官方定义。理解这两个概念最好的方式是跟面向对象编程和面向对象编程语言进行对比。相较于面向对象编程以类为组织代码的基本单元，面向过程编程则是以过程（或方法）作为组织代码的基本单元。它最主要的特点就是数据和方法相分离。相较于面向对象编程语言，面向过程编程语言最大的特点就是不支持丰富的面向对象编程特性，比如继承、多态、封装。

2. 面向对象编程相比面向过程编程有哪些优势？

面向对象编程相比起面向过程编程的优势主要有三个。

对于大规模复杂程序的开发，程序的处理流程并非单一的一条主线，而是错综复杂的网状结构。面向对象编程比起面向过程编程，更能应对这种复杂类型的程序开发。

面向对象编程相比面向过程编程，具有更加丰富的特性（封装、抽象、继承、多态）。利用这些特性编写出来的代码，更加易扩展、易复用、易维护。

从编程语言跟机器打交道的方式的演进规律中，我们可以总结出：面向对象编程语言比起面向过程编程语言，更加人性化、更加高级、更加智能。

课堂讨论

在文章中我讲到，面向对象编程比面向过程编程，更加容易应对大规模复杂程序的开发。但像 Unix、Linux 这些复杂的系统，也都是基于 C 语言这种面向过程的编程语言开发的，你怎么看待这个现象？这跟我之前的讲解相矛盾吗？

欢迎在留言区写下你的答案，和同学一起交流和分享。如果有收获，也欢迎你把这篇文章分享给你的朋友。

AI智能总结

面向过程编程与面向对象编程的区别在于代码组织方式和设计思路。面向过程编程以过程为基本单元，适合简单程序开发；而面向对象编程以类或对象为基本单元，更适用于复杂程序开发。文章通过C语言和Java的代码示例展示了两种编程风格的不同之处。面向对象编程相比面向过程编程具有更好的代码组织方式、更易维护和扩展、更高的抽象能力等优势，适用于大规模复杂程序的开发。此外，面向对象编程提供了封装、抽象、继承、多态等特性，使代码更易复用、易扩展、易维护。尽管面向过程编程和面向对象编程并非完全对立，但面向对象编程已成为主流编程范式。文章深入浅出地对比了两种编程风格，帮助读者更好地理解它们的区别和优势。面向对象编程语言比起面向过程编程语言，更加人性化、更加高级、更加智能。面向对象编程更容易应对大规模复杂程序的开发，具有更丰富的特性，更易扩展、易复用、易维护。文章深入浅出地对比了两种编程风格，帮助读者更好地理解它们的区别和优势。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

全部留言 (274)

最新 精选



相逢是缘

2019-11-16

使用任何一个编程语言编写的程序，最终执行上都要落实到CPU一条一条指令的执行（无论通过虚拟机解释执行，还是直接编译为机器码），CPU看不到是使用何种语言编写的程序。对于所有编程语言最终目的是两种：提高硬件的运行效率和提高程序员的开发效率。然而这两种很难兼得。

C语言在效率方面几乎做到了极致，它更适合挖掘硬件的价值，如：C语言用数组char a[8]，经过编译以后变成了（基地址 + 偏移量）的方式。对于CPU来说，没有运算比加法更快，它的执行效率的算法复杂度是O(1)的。从执行效率这个方面看，开发操作系统和贴近硬件的底层程序，C语言是极好的选择。

C语言带来的问题是内存越界、野指针、内存泄露等。它只关心程序飞的高不高，不关心程序猿飞的累不累。为了解脱程序员，提高开发效率，设计了OOP等更“智能”的编程语言，但是开发容易毕竟来源于对底层的一层一层又一层的包装。完成一个特定操作有了更多的中间环

节, 占用了更大的内存空间, 占用了更多的CPU运算。从这个角度看, OOP这种高级语言的流行是因为硬件越来越便宜了。我们可以想象如果大众消费级的主控芯片仍然是单核600MHz为主流, 运行Android系统点击一个界面需要2秒才能响应, 那我们现在用的大部分手机程序绝对不是使用JAVA开发的, Android操作系统也不可能建立起这么大的生态。

作者回复: 🍊

共 51 条评论 >

👍 1209



养成好习惯

2019-11-18

go语言大力推举函数式编程, 这是趋势吗老师

作者回复: 函数式编程让写代码更加简单些, 封装了很多设计模式、并发处理, 可能是个趋势。

共 4 条评论 >

👍 40



未未的未来

2019-11-15

疑问:

老师举的文件那个例子, 使用面向对象编程那个, 不是封装了函数, 用函数对操作过程进行了抽象了吗, 为什么老师说没有用到封装、抽象这些特性?

思考题:

理解, C语言虽然是面向过程语言, 但是面向过程语言也可以写面向对象的, 另外, C语言更贴近底层一些, 写操作系统的话还是有性能上的优势。

作者回复: 你指出的这点很好。关于封装, 有两种理解, 一种是狭义的面向对象特性: 封装是一种信息隐藏, 需要把数据和方法放到一起, 而c语言实现的代码, 数据和方法是分离的。封装的另一种广义的理解, 可以包含你指的封装函数。抽象实际上我们前面章节中也讲到过, 比较没有特异性, 有的时候不看做面向对象的特性。

共 10 条评论 >

👍 9



椿

2019-11-15

文中这一段, “如何给业务建模, 如何将需求翻译为类, 如何给类之间建立交互关系”, 后续会有章节展开简介吗?

作者回复: 有的 实战篇

共 4 条评论 >

👍 6



Jesse

2019-11-15

想问个问题，面向对象语言是不是没有办法直接操作系统资源？而C语言可以，所以C语言成为操作系统的首选语言，是这样吗

作者回复: 这个也算是个不错的理由。

共 4 条评论 >

👍 6



hudson

2019-11-15

formatToText换成toString是不是更符合习惯？

作者回复: 会不会有歧义呢？

共 3 条评论 >

👍 5



初学者

2020-11-24

感觉现在写代码，不是面向对象开发，更感觉像是面向数据库开发

作者回复: 大部分人都是这个感觉

共 3 条评论 >

👍 3



xavier

2019-11-15

是否可以这样理解，在例子代码中，将类一层层往下剖析，单独看类中的某个函数方法，可能是面向过程的。但当封装成一个类并使用时，就是面向对象的。

作者回复: 理解的没错。



👍 2



都市鸽

2020-01-07

go 语言的函数式编程算面向过程吗

作者回复: 不算，是一种新的编程范式



1



dagecao

2019-11-17

暂时就只有这么几节课吗？啥时候发新的课程呢？

作者回复: 每周更新三篇



1