高等学校新工科应用型人才培养“十三五”规划教材

C++语言面向对象程序设计

苏日娜　王瑞琴　编著

西安电子科技大学出版社
http://www.xduph.com

高等学校新工科应用型人才培养“十三五”规划教材

C++语言面向对象程序设计

苏日娜　王瑞琴　编著

西安电子科技大学出版社

内 容 简 介

本书共 10 章，全面介绍了 C++ 语言的相关知识。第 1、2 章介绍了面向对象程序设计的基本知识，包括数据类型、运算符和表达式以及 C++ 程序设计的基本控制结构；第 3、4 章对函数、数组和字符串进行了介绍；第 5、6 章围绕面向对象程序设计的思想，深入阐述了类和对象以及数据的共享与保护；第 7、8 章分别介绍了继承与派生、多态与运算符重载；第 9 章对模板作了较详细的介绍；第 10 章对输入/输出流和异常处理作了较深入的阐述。在学习本书前，最好先学习 C 语言相关知识。

本书通过将 C 语言面向过程的程序设计方法与 C++ 语言面向对象的程序设计方法进行对比，让读者深刻体会用 C++ 语言进行面向对象程序设计的优势。通过学习 C++ 语言的知识，运用 C++ 语言的方法和技巧设计程序，能够解决综合性强和复杂度高的问题。书中也给出了相应的例题和相关程序，通过将理论和实践相结合，可使读者更好地掌握面向对象程序设计的原理和方法。

本书可作为高校计算机及相关专业的“C++ 程序设计”和“面向对象程序设计”课程的教材，也可作为读者自学 C++ 语言的参考书。

图书在版编目(CIP)数据

C++语言面向对象程序设计 / 苏日娜，王瑞琴编著. —西安：西安电子科技大学出版社，2019.7

ISBN 978-7-5606-5354-9

Ⅰ. ① C… Ⅱ. ① 苏… ② 王… Ⅲ. ① C++ 语言—程序设计 Ⅳ. ① TP312.8

中国版本图书馆 CIP 数据核字(2019)第 112106 号

策划编辑　李惠萍

责任编辑　郭　魁　李惠萍

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)88242885　88201467　　邮　　编　710071

网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com

经　　销　新华书店

印刷单位　陕西日报社

版　　次　2019 年 7 月第 1 版　　2019 年 7 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印　张　17.5

字　　数　414 千字

印　　数　1～2000 册

定　　价　39.00 元

ISBN 978-7-5606-5354-9 / TP

XDUP 5656001-1

如有印装问题可调换

前　言

计算机专业开设的第一门高级语言程序设计课程，一般是C语言或C++语言。作为第一门计算机编程语言，需要学好基础知识，为以后进一步学习和应用打下良好的基础。在实际教学过程中，学习过C语言的学习者，往往不太理解C++语言与C语言的区别，因此在编程的思维方式和程序设计方法上，很难从C语言的面向过程方式向C++语言的面向对象的方式转变。本书在详细介绍C++语言基础知识的同时，把相关知识点与C语言进行比较。通过对比C语言与C++语言，让学习者更好地掌握并巩固C++语言的语法知识，同时引导有C语言基础的学习者学会编写C++程序。本书在讲解过程中逐步渗透面向对象程序设计的思想，将需要解决的问题采用面向对象的方法进行描述和设计，把描述问题的数据和对数据的操作封装在一起形成类，而类是面向对象程序设计的重要基础，通过继承性、多态性等技术支持，使得开发程序的可重用性、开发效率大大提高。

本书以C++语言作为一门独立的课程进行全面和系统的讲述。在内容编排上，按照循序渐进的原则，第1章到第4章从C++语言的基本概念和基本语法知识讲起，将C++语言与C语言进行对比，使读者易于学习C++语法基础；第5章之后的章节则重点讲述了面向对象的程序设计思想和方法，整个内容按照从简单到复杂的思路进行设计。为了使理论结合实践，本书的每一章中都给出了大量的例题，这些例题有助于学习者更好地理解C++语言的语法和面向对象的程序设计方法。同时，考虑到面向对象程序设计语言的特点，本书在重点章的最后一节均设计了综合性的程序，有助于学习者掌握如何利用C++语言进行复杂问题的求解。

全书编写工作主要由苏日娜完成，夏麟老师负责设计、编写了第6章至第10章的人事信息管理程序，戴洪珠、王瑞琴、章春芽、胡正华编写了部分章节，陈溢聪、罗镇楠、李鹏辉等参与了部分章节的整理和代码的调试以及文字录入和校对等工作。

感谢读者选择使用本书。由于编者水平有限，书中不足之处在所难免，恳请读者批评指正，在此表示诚挚的感谢！联系E-mail：nbsrn@126.com，来信标题请包含“C++ book”。

编　者

2019年5月

于宁波工程学院

目　录

第 1 章　面向对象程序设计概述

本章介绍计算机程序设计方法，包括结构化程序设计方法和面向对象程序设计方法。通过对这两种方法的对比了解面向对象程序设计方法的优势，理解为什么要使用面向对象方法来进行程序设计。同时介绍面向对象程序设计的基本概念、面向对象程序设计的基本特征、面向对象程序设计语言以及面向对象程序的开发环境。

1.1　计算机程序设计方法

计算机程序就是一组能够被计算机识别并执行指令的集合。正确有效地设计程序不仅需要计算机语言的支持，还需要程序设计的思想和方法来指导。常用的程序设计方法包括结构化程序设计方法和面向对象程序设计方法。

1.1.1　结构化程序设计方法

结构化程序设计(Structured Programming，SP)方法诞生于 20 世纪 60 年代，盛行于 20 世纪 70 至 80 年代，它建立在 Bohm、Jacopini 证明的结构定理基础上。该结构定理指出：任何程序逻辑都可以采用顺序、选择和循环三种基本结构表示。在结构化程序设计过程中注重的是程序结构的规范性，强调的是程序设计的自顶向下、逐步求精的演化过程。这样在问题的求解过程中，首先要对解决的任务进行整体规划，将一个复杂的任务按照功能分解成一个个易于控制和处理的子任务，然后对每个子任务再进行细化，依此进行，直到不需要再细分为止。具体实现程序时，每个子任务对应一个子模块。程序设计过程就是划分模块、向下分解、再把模块划分成子模块的过程。模块间尽量相对独立，通过模块间的调用关系或全局变量有机地联系起来。因此，应用结构化程序设计方法解决问题时，遵循的原则为：自顶向下、逐步细化、模块化设计、结构化编码。

结构化程序设计方法是以数据的流向为线索，围绕实现功能处理的过程来构造系统的，也可形象地称为面向过程的程序设计方法。采用结构化程序设计方法设计的程序包括过程定义和过程调用(过程即为完成某项操作所需执行的一段代码，通常可以采用函数实现)。在设计过程中数据和处理这些数据的算法(过程)是分离的。这样对不同数据做相同处理，或者对相同数据做不同的处理，都要使用不同的模块，从而降低了程序的可维护性和可复用性。同时，这种分离还可能存在数据被多个模块共同使用和修改的情况，数据的安全性和一致性难以保证。

随着计算机处理的信息量和信息类型迅速增加，有待程序解决的问题已从数值计算扩展到人类社会的方方面面，所处理的数据也从简单的数字和字符发展为具有多种格式的数

据，如文本、图形、图像、影像、声音等，同时描述的问题也越来越复杂。显然，面向过程的程序设计方法已经远远不能满足大规模的软件开发要求。于是，人们开始寻求一种更加先进的程序设计方法，面向对象的程序设计方法因此应运而生。

1.1.2 面向对象程序设计方法

面向对象程序设计(Object-Oriented Programming，OOP)方法是吸收了软件工程领域中有益概念和有效方法而发展起来的一种软件开发方法，也常称为面向对象的编程。它更直观地描述客观世界存在的事物(即对象)及事物之间的相互关系。客观世界存在的事物可以看成是具有某些静态特征(采用数据描述)和动态特征(采用操作描述)的统一体。因此，面向对象程序设计是将数据及对数据的操作封装在一起，形成一个相互依存、不可分离的整体——类。程序设计一般由类的定义和类的使用两部分组成，以事件或消息驱动对象执行相应的处理来完成相关操作。类是对同类型对象共同特性的抽象。类中的数据大多数情况下只能被该类所封装的操作所使用，这样有助于数据的安全性和一致性。封装好的类将通过外部接口与外界进行联系。这样，程序模块之间的关系相对简单，程序的流程不再是流水线式的过程化按步执行，而是根据程序运行时各种事件的实际触发来执行。程序执行流程不遵循预定顺序，因而更符合软件设计和开发的实际情况。

同时，根据应用程序对类扩充的新的需求，在原有抽象好的已定义类的基础上，增加一些新的数据和操作即可产生一个新的类，不需要对原始类所封装的已有内容重写一遍，这样使得代码具有良好的重用性。而且，面向对象程序设计所具有的多态性使得相同操作处理不同类型数据时无需重复写功能相似的代码，从而大大提高了代码的开发效率。

1.1.3 面向对象与面向过程的比较

结构化程序设计方法解决了早期计算机程序难于阅读、理解、调试和难于设计、开发、维护等问题。对于小型和中等复杂程度的程序来说，结构化程序设计是一种较好的开发方法。基于 C 语言的面向过程程序设计在很长一段时间内成为了计算机程序开发的主流技术，其采用模块化设计以及采用顺序、选择、循环三种基本结构高效解决了许多程序设计问题。

软件开发过程中面向过程的程序设计方法以功能为基础，将数据和对数据的操作相分离，其优点是结构清晰、模块化强。但其代码的重用性差，不利于代码的维护与扩展，适用于小型算法与程序的设计开发。

结构化程序设计是由“数据结构 + 算法”组成的。

下面采用面向过程的程序设计方法，使用 C 语言编写一个简单的比较两个整数并求其中大数的程序。

```
#include "stdio.h"
int max(int x, int y)
{
    return (x>y) ? x : y;
}
void main()
```

```
{
    int a,b;
    scanf("%d,%d", &a, &b);
    printf("the max is %d\n", max(a, b));
}
```

说明：该程序由 max 和 main 两个功能模块组成。每个模块设计成一个独立的函数，程序是按照结构化方法设计的。程序以设计两个数的数据结构和求两个数中较大者的比较算法为出发点进行。在主函数中设置的数据结构为两个整型变量，自定义函数 max 用于完成两个数进行比较的算法设计。

面向对象程序设计的思想是在面向过程设计的基础上，遵循现实世界中的事物及事物之间的关系，转变为以类和对象的设计和使用为中心，将数据和对数据的操作封装在一起形成类，对类进行实例化形成对象，从而进行安全、高效、可重用性强的程序开发。面向对象的设计方法符合人类认识事物、解决问题的思维方式，更适用于大型软件的开发和维护。

下面采用面向对象的程序设计方法，使用 C++ 语言编写比较两个数并求其大数的程序。

```
#include <iostream>
using namespace std;
class Compare
{
public:
    void SetData(float x, float y);
    float MaxData();
    void OutPutMax();
private:
    float a, b, max;
};
void Compare :: SetData(float x, float y)
{
    a = x;
    b = y;
}
float Compare :: MaxData ()
{
    max = (a>b) ? a : b;
    return max;
}
void Compare :: OutPutMax()
{
    cout << "the max is" << max << endl;
```

```
}
int main()
{   Compare twodata;
    twodata.SetData(8.2, 9.9);
    twodata.MaxData ();
    twodata.OutPutMax();
    return 0;
}
```

说明：程序是由类 Compare 的定义、实现和主函数 main 组成的。虽然该程序的代码量多于面向过程方法设计的程序，但是它的思想是将问题所涉及的数据和对数据的操作组合在一起形成类。主函数不再负责繁杂的数据结构定义，而是关注对类的使用。程序可以由若干个类组成，主函数根据问题需要对类进行调用，某个类中数据和操作的改动不会影响其他类和主函数。

由此可见，相对于面向过程程序设计，面向对象程序设计在设计思想和设计方法方面发生了质的转变。其根本性的变化在于：不再将软件系统看成是工作在数据上的一系列过程或函数的集合，而是看成一系列相互协作而又彼此独立的类和对象的集合。这种方法集抽象性、封装性、继承性和多态性于一体，更符合人们的思维方式，有助于保持问题空间和解空间在结构上的一致，同时保证所要处理的数据具有良好的安全性，程序具有良好的可重用性，开发效率高，更适合非数值问题处理和图形化程序设计。

1.2　面向对象程序设计

上一节介绍了计算机两大主流程序设计方法，通过实例对采用这两种方法设计的程序进行了比较，其中涉及许多面向对象的概念，本节将着重介绍这些基本概念，并讲述面向对象的基本特征和采用面向对象思想进行软件开发的方法。

1.2.1　面向对象的基本概念

面向对象程序设计是建立在对象、类、消息传递等概念基础上的，设计的应用程序常常具备封装性、继承性、多态性等特征。下面将对这些概念和特性作详细介绍。

1．对象

从现实世界角度，对象是客观世界存在的一个事物或一个实体的描述，可以是有形的(比如一辆汽车、一名学生、一只猫)，也可以是无形的(比如一个工程、一次考试、一项计划)。从认识事物的角度，一般可以从静态的属性和动态的行为两方面来描述一个对象。例如，一名学生可以描述为“姓名：张三　性别：男　年龄：21　身高：170　体重：70”，这里姓名、性别、年龄、身高、体重都描述的是张三的静态特性(属性)，还可以描述该学生“上课、运动、看书”等动作，这些是张三的动态特性(行为)。因此，对象需要设定名字以区别于其他对象；需要通过属性来描述其某些静态特征；需要有一组操作定义其相关行为，这些行为或作用于自身，或作用于其他对象。

从面向对象程序设计角度，遵循这种对客观事物的描述方法，通过将描述属性的数据和对这些数据实施相关的操作封装在一起构成一个统一的对象概念。用数据来表示静态属性，用函数来表示动态行为。

2. 类

从现实世界角度，分类是将一类事物区别于另一类事物的方法。分类需要依靠对事物的抽象，找到事物的共同特性，从而抽象出一类事物的概念。例如，张三、李四、王五……忽略了非本质特征后，他们的基本特征都是一致的，于是把他们抽象为一个概念——“人”，即“人”类。

从面向对象程序设计角度，类也同样遵循这种抽象方法。将具有相同数据和相同操作的一组对象的集合称为类，这也是抽象数据类型的一种实现方式。类是对一类对象的抽象描述，而对象是某个类的具体实例化表示。类和对象的关系是抽象与具体的关系，就好比模具与铸件之间的关系。类就相当于模具，它定义的是一种类型，而对象则好比在该模具定义下产生的具体铸件，具有具体的实现形态。在程序设计中总是先定义类，再声明和使用属于这个类的对象。

3. 消息

从现实世界角度，事物不是孤立存在的实体，事物与事物之间存在着各种各样的联系。正是它们之间的相互作用、联系和连接，才构成了世界各种不同的系统。例如，教师在课堂上提出问题，让学生回答问题，学生收到问题信息后准备作答。在这个过程中，教师和学生两个对象之间的联系是通过问与答的消息传递建立起来的。

从面向对象程序设计角度，对象之间也需要建立联系进行交互，一个对象向另一个对象发出建立联系的消息。所谓消息，是面向对象发出的服务请求，其实也是调用该对象的一个方法的过程。它是面向对象系统中对象之间交互的途径。对象之间通过消息联系，彼此共同协作，才能形成一个有机的系统。

当一个消息发送给某一对象时，接收到消息的对象经过解读消息，然后予以执行，这种通信机制称为消息传递。消息机制为独立的对象提供了一个相互动态联系的途径，使它们的行为能互相配合，构成一个有机运行的系统。通常，一个消息由消息的发送者、消息的接收者、消息所要求的具体服务、消息所要求服务的一些参数以及消息的应答几部分组成。发送消息的对象不需要知道接收消息的对象如何对消息进行响应。通常采用调用功能函数来实现消息的传递和请求响应。

1.2.2　面向对象的基本特征

1. 封装性

从现实世界角度，所谓封装就是把某个事物包裹起来，使外界不知道其中的内容。从面向对象程序设计角度，封装是指将数据和对数据的操作集中起来放在对象内部，并尽可能隐蔽对象的内部细节。对象好比一个不透明的黑盒子，从外界是看不见内部的，更不能从外面直接访问或修改内部的数据及代码。在使用的时候，仅提供对外访问的接口而无需知道它的数据结构细节和实现操作的算法。在封装之前要做好设计数据和操作的工作，设计数据结构和功能操作，确定哪些数据需要进行隐藏，哪些是对外开放的。通过设置数据

和操作的访问权限来控制对象内部对外界的可访问性和开放程度。

封装的好处是可以将对象的使用者与设计者分开，大大降低了使用者操作对象的复杂度。使用者不必知道对象具体的内部细节，只需要使用设计者提供的接口功能，就可以自如地操作对象。封装的结果实际上隐藏了复杂性，并使得代码具有重用性，从而降低了开发软件系统的难度。

2. 继承性

从现实世界角度，很多事物具有一定的延续性和多层结构。例如，交通工具可分为汽车、火车、飞机、轮船等。其中汽车又可以分为客车和卡车。客车又可继续发展出小客车、中巴车、大客车。每一次划分所产生的新的类可以是在已有类的基础上添加一些新的特性而产生的。这个已有类我们称为基类或父类，新的类称为派生类或子类。

从面向对象程序设计角度，提供继承与派生机制，在基类的基础上仅增加或修改部分属性和操作就可以创建一个全新的类。同时，派生出的子类还可以继续派生它的子类，如此下去，可以形成树状派生关系，称为派生树或继承树。

继承性是面向对象程序设计的一个重要特征。继承体现了特殊类与一般类之间的上下分层关系，这种机制为程序员提供了一种组织、构造、重用类的手段。继承使一个类(基类或父类)的数据成员和成员函数能够被另一个类(派生类或子类)重用。在子类中只需增加一些基类中没有的数据成员和成员函数，或对基类中的某些数据成员或成员函数进行改造，这样就可以避免公共代码的重复开发，减少代码和数据的冗余。

从继承方式分类，继承可以分为单一继承和多重继承两种。单一继承是指子类只能从一个基类派生出来。比如，祖孙三代的财产继承关系，可以由爷爷传给父亲，再由父亲传给孩子。多重继承是指子类可以从多个基类派生出来。比如，一个人身上的基因信息就是来自于父亲和母亲两者的遗传。

继承性简化了人们对问题的认识和描述，同时还可以在开发新程序和扩充原程序时最大限度地利用已有程序，提高程序的可重用性，从而提高程序修改、扩充和设计的效率。

3. 多态性

从现实世界角度，多态性体现在同一种行为在不同对象发出请求时会呈现不同的响应。例如，对于学生上课这种行为，不同的老师来授课所开展的教学活动是不同的。教程序设计老师上的内容与教英语的老师上的内容是截然不同的。所以在授课或说响应学生上课这个请求事件过程中，授课行为在不同教师的执行过程中呈现了多种形态。

从面向对象程序设计角度，多态性是指在基类中定义的属性或操作被派生类继承后，针对不同的数据类型会表现出不同的行为。也就是使得同样的操作对不同的对象有不同的表现方式。当一个对象接收到一个进行某项服务的请求消息时，将根据对象所属的类，动态地选用该类中定义的操作。不同的类对消息按不同的方式解释。例如，我们定义一个图形类，这个类具有绘图这样的操作。图形类又派生出了圆类、正方形类、长方形类。当这三种具体的图形在执行绘图操作时分别画出了各自的图形，它们执行了不同操作，也就是绘图操作对不同对象产生了不同形态。多态性包括静态多态性(编译时多态性)和动态多态性(运行时多态性)两种。

多态性的意义在于同一个接口实现了不同操作。因此，面向对象的多态特性使软件开

发更科学、更方便，且更符合人类的思维习惯，能有效提高软件开发效率、缩短开发周期、提高软件可靠性，使所开发的软件更健壮。

1.2.3　面向对象的软件开发

面向对象的软件开发是把面向对象的思想应用于软件开发，指导开发活动的全过程，是面向对象程序设计方法在软件工程领域中的全面应用。开发的全过程遵循软件工程的流程，它主要包括面向对象的分析(OOA)、面向对象的设计(OOD)、面向对象的编程(OOP)、面向对象的测试(OOT)和面向对象的软件维护(OOSM)等主要内容。完备、正确地分析、理解、表达所需解决问题的内在实质为良好的设计奠定了基础，更是最终编程实现问题的解的重要保障。编写程序只是其中相对较小的一部分。

1．分析

系统分析阶段应该简明扼要地抽象出系统必须做什么，而不涉及如何做及怎样实现。这一阶段是整个软件工程的初始阶段，要求能够准确地描述需求并抽象出问题的模型，建立类和对象，确定对象的属性、方法、关联关系和对象间的通信。分析者需要与用户一起共同沟通交流，通过多层次的迭代过程完成分析任务。用分析的结果代替原始的问题描述，并作为后期设计阶段的基础。

面向对象的分析(Object-Oriented Analysis，OOA)直接用问题域中客观存在的事物建立模型中的对象，对单个事物及事物之间的关系，都保留它们的原貌，不做转换，也不打破原有界限而重新组合，因此能够很好地映射客观事物。

在用 OOA 具体地分析一个事物时，大致遵循如下五个基本步骤：

第一步，确定对象和类。这里所说的对象是对数据及其处理方式的抽象，它反映了系统保存和处理现实世界中某些事物信息的能力。类是多个对象的共同属性和方法集合的描述，它包括如何在一个类中建立一个新对象的描述。

第二步，确定结构。结构是指问题域的复杂性和连接关系。类成员结构反映了泛化—特化关系，整体—部分结构反映整体与局部之间的关系。

第三步，确定主题。主题是指事物的总体概貌和总体分析模型。

第四步，确定属性。属性就是数据元素，可用来描述对象或分类结构的实例，可在图中给出，并在对象的存储中指定。

第五步，确定方法。方法是在收到消息后必须进行的一些处理方法，方法要在图中定义，并在对象的存储中指定。

2．设计

面向对象的设计(Object-Oriented Design，OOD)是面向对象方法中一个中间过渡环节。其主要作用是对 OOA 分析的结果作进一步的规范化整理和模型扩充，以便能够被下阶段的 OOP 直接接受。这个过程是把分析阶段得到的需求转变成符合成本和质量要求的实现方案的过程，也可以说它是用面向对象的观点去解决问题域模型的过程。在这一阶段，需要对分析阶段建立的对象模型进行细化，加入必要的实现细节。设计阶段对分析的结果进行深入的加工，这个阶段更多地考虑与实现相关的因素。具体来说，设计阶段将针对以下四个方面进行：

(1) 问题域：在设计阶段将分析阶段得到的概念性的类与实际具体实现环境相结合，增加更多有利于实现的相关属性和操作。

(2) 人机交互：人机交互包括系统的输入和输出的设计。

(3) 数据管理：在分析阶段，不必关心信息是如何保存的，但到设计阶段，就需要考虑如何实现数据与问题域对象之间的接口等。

(4) 系统交互：系统除了和用户打交道外，还可能与其他外设或系统有关，设计时需要考虑这些事物的接口。

上面各个部分的设计过程都是类与类之间关系的标识和设计的过程，该过程和 OOA 分析中的各个步骤相同，也要经历类的发现、类结构的设计、类之间关系的设计等步骤，亦需要借助于一些手段如 use case、主题划分等来协助设计。

3. 编程

面向对象的编程(OOP)工作就是用一种面向对象的编程语言把OOD模型中的每个部分书写出来，是面向对象的软件开发最终得以实现的重要阶段。在软件开发的全过程中，程序的分析和设计过程是重要基础，没有正确的需求分析和模块设计，编程阶段的工作也就没有意义。所以对于软件开发人员来说不能仅仅关注程序实现的技巧，更应在真正理解和掌握面向对象程序设计的基本方法和核心思想上下功夫，把程序设计好。编程阶段的主要工作是使用选定的程序设计语言，把模块的过程性描述翻译为用语言书写的源程序。源程序要求正确可靠、简明清晰、效率高。

(1) 源程序的正确性是对程序质量的最基本要求。

(2) 源程序简明清晰，才便于验证源代码和模块规格说明的一致性，容易进行测试和维护。

(3) 源程序的清晰与效率之间常存在矛盾，要求清晰性好的程序一般效率较低，而要求效率高的程序一般清晰性较差。对于大多数模块，编码时应该把简明清晰放在首位。

(4) 除了编程阶段产生源代码外，在测试阶段也需要编写一些测试程序，用于对软件的测试。

4. 测试

面向对象测试(Object-Oriented Test，OOT)的任务是发现软件中的错误。在面向对象的软件测试中继续运用面向对象的概念与原则来组织测试，以对象的类作为基本测试单位，可以更准确地发现程序错误并提高测试效率。软件测试也是一门学科，良好有效的测试需要很多测试方法和工具才能实现。测试也是软件成为可交付使用的产品必不可少的重要环节。这个环节需要多次反复进行，确保软件产品将错误率降到最低。

5. 维护

面向对象的软件维护(Object-Oriented Software Maintenance，OOSM)是软件开发过程中不可缺少的环节。将软件交付使用后，工作并没有完结，还要根据软件的运行情况和用户的需求，不断改进系统。现代软件的规模越来越大，在交付使用后也很难保证没有各种各样的隐含错误，这就需要开发人员或专业软件维护人员进行必要和合理的维护。

使用面向对象的方法开发的软件，其程序与问题域是一致的。因此，在维护阶段运用面向对象的方法，采用以类及对象为基本单位进行维护，这样可以大大提高软件维护的效率。

1.3　程序设计语言

计算机系统包括硬件系统和软件系统两大部分。用户的需求需要依靠软件来实现。没有软件，计算机仅是一台裸机，没有什么功能。在用计算机解决问题之前，必须先把求解的问题用计算机能够理解的语言表述出来，编写成可执行的程序。编写程序所使用的语言称为程序设计语言，它是外界和计算机沟通互动的桥梁。随着程序设计方法和技术的不断发展，直接导致了一大批风格各异的程序设计语言的诞生。程序设计语言的发展经历了机器语言、汇编语言、高级语言、面向对象程序设计语言等多个阶段。

1.3.1　机器语言

在计算机刚诞生之时，使用的是最原始的穿孔卡片，这种卡片上使用的语言是只有专家才能理解的语言，它只能用二进制数编制指令控制计算机运行。每一条指令都是由“0”、“1”这两个数字按照一定的规则排列而成的，与人类的语言差别极大，这种语言被称为机器语言。机器语言也是第一代计算机语言，使用二进制位来表示程序指令。例如，计算 3+5 的机器语言程序如下：

```
10110000    00000011    //将 3 送往累加器
00000100    00000101    //将 5 与累加器中的 3 相加，结果保留在累加器中
```

这种语言本质上是计算机唯一能识别并直接执行的语言，与汇编语言或高级语言相比，其执行效率高。但机器语言很难理解，程序可读性差，编写、修改、调试难度巨大，不容易掌握和使用。在这之后的语言是在机器语言的基础上发展而来的。虽然后来发展的语言能让人类直接理解，但最终送入计算机的还是这种机器语言。

1.3.2　汇编语言

计算机语言发展到第二代，出现了汇编语言。汇编语言是由一组与机器语言指令相对应的符号指令和简单语法组成的语言，它用助记符代替了操作码，用地址符号或标号代替地址码。汇编语言使用符号代替机器语言的二进制码，因此也称为符号语言。

例如，计算 3 + 5 的汇编语言程序如下：

```
MOV   AL, 03H      //将十六进制数 3 送往累加器
ADD   AL, 05H      //将十六进制数 5 与累加器中的 3 相加，结果保留在累加器中
```

汇编语言程序不能够被计算机直接运行，需要由翻译程序将它翻译成机器语言。汇编语言的功能很强，能发挥计算机各硬件的功能。但在使用汇编语言编写程序时，要求程序编写者熟悉计算机内部的结构和组织，特别是要熟悉计算机微处理器的结构和处理器指令及相关外围硬件设备等。比起机器语言，汇编语言更易读、易写，尽管还是复杂，用起来容易出错，但在计算机语言发展史上是机器语言向高级语言进化的桥梁。

1.3.3　高级语言

机器语言和汇编语言都称为低级语言，是面向机器的，而且学习起来困难，编程效率

低，可读性、可维护性差。而高级语言更接近自然语言和数学公式的编程，基本脱离了机器的硬件系统，使人们可用更易理解的方式编写程序。

高级语言与计算机的硬件结构及指令系统无关，它有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且容易学习掌握。但高级语言编译生成的程序代码一般比用汇编程序语言设计的程序代码要长，执行的速度也慢。所以汇编语言适合编写一些对速度和代码长度要求高的程序和直接控制硬件的程序。高级语言程序“看不见”机器的硬件结构，不能用于编写直接访问机器硬件资源的系统软件或设备控制软件。为此，一些高级语言提供了与汇编语言之间的调用接口。用汇编语言编写的程序，可作为高级语言的一个外部过程或函数，利用堆栈来传递参数或参数的地址。

例如，计算 3+5 的 C++语言程序如下：

```
int sum;                              //定义整型变量 sum
sum = 3 + 5;                          //将 3 与 5 的和赋值给 sum
cout << "3+5=" << sum << endl;        //输出 3+5=8
```

本书将讨论 C++编程语言，并用它编写程序。C++也是一种高级语言。其他高级语言还有 C、C#、Java、Python、PHP、Pascal、FORTRAN 等等。高级语言更接近人类使用的语言，其设计宗旨是方便人们编写和阅读程序。

1.3.4 面向对象程序设计语言

20 世纪 80 年代，出现了面向对象的编程语言。面向对象的编程语言是为了能够更直接地描述客观世界中存在的事物以及它们之间的关系而设计的。面向对象语言是比面向过程语言更高级的一种高级语言。它更接近于自然语言和人类的表述方式，是人们对客观事物更高层次的抽象。

面向对象程序设计语言与以往各种编程语言的根本区别是程序设计思维方法的不同，面向对象程序设计可以更直接地描述客观世界存在的事物(即对象)及事物之间的相互作用关系。这使得程序能够比较直接地反映客观世界的真实情况，软件设计人员能够利用人类认识事物的规律及所采用的一般思维方法来进行软件设计。

面向对象程序设计语言经历了一个很长的发展阶段。例如，LISP 家族的面向对象语言、Simula 67 语言、Smalltalk 语言以及 Python、Java、C#、C++等语言，都不同程度地采用了面向对象的方法和基本概念。

C++语言是在应用最广泛、最深入的 C 语言基础上发展起来的，凭借 C++对 C 的兼容和 C++自身强大的功能，使得 C++语言成为广泛使用的面向对象程序设计语言之一。

1.4 C++语言面向对象程序开发

面向对象程序设计语言发展至今，一般都需要一个集成的开发环境(Integrated Development Environment，IDE)来支撑程序的设计与实现。整个程序的编写与实现需要经历源程序、目标程序、可执行程序三个阶段，经过 IDE 的编辑、编译、调试无误后才能正确执行，从而得到想要的结果。