高等学校计算机专业
面向项目实践规划教材

C++程序设计基础案例教程

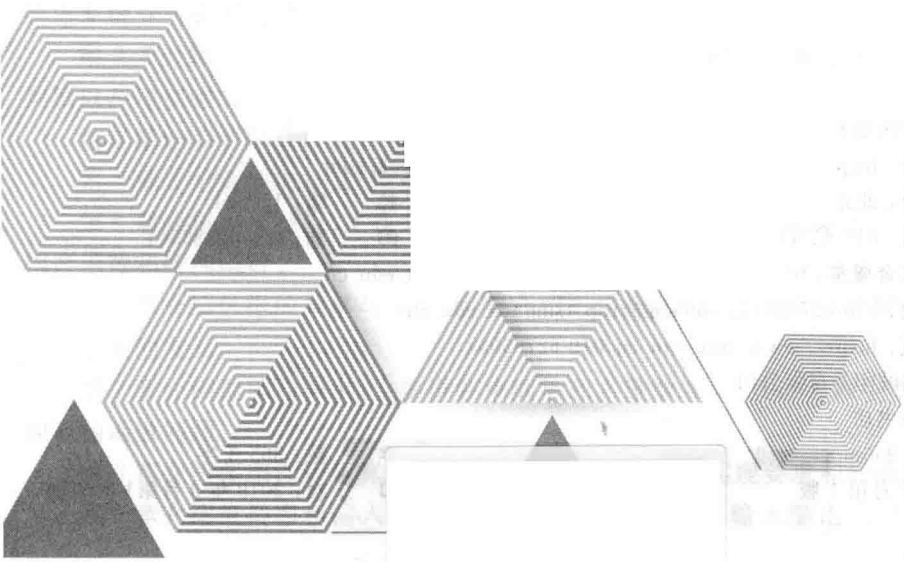
◎ 吴　艳　费如纯　主编
高　艳　副主编

清華大學出版社

高等学校计算机专业
面向项目实践规划教材

C++程序设计基础案例教程

◎ 吴　艳　费如纯　主编
高　艳　副主编



清華大學出版社
北京

内 容 简 介

本书全面介绍 C++面向对象程序设计语言，书中从软件开发过程入手，对软件采用面向对象方法进行开发做了简要介绍；着重讲解 C++面向对象语言的基础知识：数据类型、表达式、语句以及三种基本结构；介绍了面向对象的概念、构造函数和析构函数(包括特殊的构造函数)；接着介绍了面向对象的一些重要特征(抽象、继承、多态等)；最后介绍了标准输入输出流的问题，尤其是一些常用特殊格式输出以及异常处理等问题。全书提供了大量应用实例，每章后均附有习题。

本书适合作为高等院校计算机、软件工程、物联网工程专业本科生、研究生的教材，同时可供软件开发人员、广大科技工作者和研究人员参考。

图书在版编目(CIP)数据

C++程序设计基础案例教程/吴艳等主编. —北京：清华大学出版社，2019
(高等学校计算机专业面向项目实践规划教材)
ISBN 978-7-302-48383-0

Ⅰ. ①C…　Ⅱ. ①吴…　Ⅲ. ①C++语言－程序设计－高等学校－教材　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 216772 号

责任编辑：贾　斌　薛　阳
封面设计：刘　键
责任校对：胡伟民
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：15　　**字　　数**：365 千字
版　　次：2019 年 9 月第 1 版　　**印　　次**：2019 年 9 月第 1 次印刷
印　　数：1～1500
定　　价：39.80 元

产品编号：071600-01

前言

FOREWORD

面向对象程序设计(Object Oriented Programming,OOP)借助20世纪50年代的人工智能语言LISP引入,发展至今逐步成为计算机程序设计的主流,由于其设计思想符合人们解决问题的思维方式,因此逐步被越来越多的软件设计人员所接受。C++语言是在C语言的基础上发展起来的,是一门高效实用的程序设计语言,它既可以进行过程化程序设计,又可以进行面向对象程序设计。

C++不仅集成了C语言灵活高效、功能强大、可移植性好等特点,而且引入了面向对象程序设计的思想和机制,可以在很大程度上提高编程能力,减少软件维护的开销,增强软件的可扩展性和可重用性。

本书从编程的基本知识入手,以短小精悍的例题作为课内案例,针对每个章节的知识点进行详解及扩充,对有无编程基础的读者都是适用的。此外,全书以某公司人员管理系统作为实际案例,贯穿全书,通过理论知识的实际应用,更形象地诠释了知识的应用,提高读者对知识点的掌握程序,同时培养读者对实际问题的分析能力、解决能力,进一步提高读者的实践能力。

全书共10章,其各章节的内容如下:

第1章介绍程序设计的基本概念以及程序设计的基本过程,利用公司人员管理系统来阐述系统分析的理论知识。

第2章介绍C++程序基础知识,主要包括一个C++程序的开发过程,C++中预定义数据类型以及对应的表达式,系统输入输出函数的使用。

第3章介绍程序设计的三种基本结构。

第4章介绍函数的定义、声明、调用以及一些特殊函数。

第5章介绍类和对象,主要介绍面向对象的特点,类和对象的概念以及定义,最后介绍构造函数和析构函数。

第6章介绍数据的共享与保护,主要介绍标识符的作用域和定义存储类型问题,同时也介绍了类的友元。

第7章介绍继承与派生,主要讲解单继承和多重继承,以及因为派生而产生的构造函数和析构函数问题。

第8章介绍多态性和运算符的重载,主要介绍多态的实现要求和特殊的运算符重载。

第9章介绍流类库和输入输出,主要介绍C++的基本输入输出流以及对应的格式控制符。

第10章介绍异常处理,主要介绍一些简单异常对应的解决办法。

本书的每一章后均配有对本章知识点的总结——小结,对知识掌握程度的验证——习题,这些有助于提高读者的实际操作能力及运用能力。

本书由吴艳、费如纯担任主编，高艳担任副主编，其中第1章由高艳编写，第2～8章由吴艳编写，第9章和第10章由费如纯编写，吴艳、费如纯负责全书的统稿，由高艳完成习题的整理。由于编者水平有限，错误和疏漏之处在所难免，恳请广大读者批评指正。

编　者

2017年6月

目录

CONTENTS

第1章 绪论 …… 1

1.1 程序设计语言简介 …… 1
1.1.1 低级语言 …… 1
1.1.2 高级语言 …… 2
1.1.3 面向对象的语言 …… 2
1.2 面向对象程序设计基础简介 …… 2
1.2.1 面向对象方法的由来 …… 2
1.2.2 面向对象的基本概念 …… 3
1.3 面向对象软件开发简介 …… 4
1.3.1 软件分析 …… 4
1.3.2 软件设计 …… 5
1.3.3 软件编程 …… 7
1.3.4 软件测试 …… 8
1.3.5 软件维护 …… 9
1.4 综合案例——公司人员管理系统1 …… 10
1.4.1 系统描述和要求 …… 10
1.4.2 系统分析和设计 …… 10
1.5 小结 …… 10
习题1 …… 11

第2章 C++简单程序设计 …… 12

2.1 C++语言概述 …… 12
2.1.1 C++的产生 …… 12
2.1.2 C++的特点 …… 13
2.1.3 C++程序开发过程 …… 13
2.1.4 C++程序实例 …… 14
2.1.5 字符集 …… 15
2.2 基本数据类型和表达式 …… 16
2.2.1 基本数据类型 …… 16
2.2.2 自定义数据类型 …… 16
2.2.3 常量 …… 25

2.2.4 变量 …… 28
2.2.5 符号常量 …… 29
2.2.6 运算符与表达式 …… 31
2.2.7 语句 …… 38
2.3 数据的输入与输出 …… 39
2.3.1 I/O 流 …… 39
2.3.2 预定义的插入符和提取符 …… 39
2.3.3 简单的 I/O 格式控制 …… 40
2.4 综合案例——公司人员管理系统 2 …… 41
2.5 小结 …… 42
习题 2 …… 42

第 3 章 程序设计结构 …… 44

3.1 算法的基本控制结构 …… 44
3.2 顺序结构 …… 45
3.3 分支结构 …… 46
3.3.1 单分支结构 …… 46
3.3.2 双分支结构 …… 47
3.3.3 多分支结构 …… 48
3.4 循环结构 …… 53
3.4.1 for 语句 …… 53
3.4.2 while 语句 …… 54
3.4.3 do…while 语句 …… 55
3.5 其他控制语句 …… 57
3.6 多种结构的嵌套 …… 60
3.7 综合案例——公司人员管理系统 3 …… 63
3.8 小结 …… 67
习题 3 …… 67

第 4 章 函数 …… 68

4.1 函数的定义与使用 …… 68
4.1.1 函数的定义 …… 68
4.1.2 函数的声明 …… 69
4.1.3 函数的调用 …… 70
4.1.4 函数的参数传递 …… 73
4.1.5 局部变量和全局变量 …… 76
4.1.6 变量的存储类别 …… 78
4.2 内联函数 …… 79
4.3 带默认形参值的函数 …… 80

4.4 函数重载 …… 82
4.5 常用的系统函数 …… 85
4.6 综合案例——公司人员管理系统 4 …… 86
4.7 小结 …… 86
习题 4 …… 86

第 5 章 类与对象 …… 87

5.1 面向对象程序设计的基本特点 …… 87
5.1.1 抽象 …… 87
5.1.2 封装 …… 88
5.1.3 继承 …… 88
5.1.4 多态 …… 89
5.2 类和对象 …… 89
5.2.1 类的定义 …… 89
5.2.2 类成员的访问控制 …… 91
5.2.3 对象 …… 92
5.2.4 类的成员函数 …… 96
5.2.5 组合类 …… 97
5.2.6 程序实例 …… 97
5.3 构造函数和析构函数 …… 100
5.3.1 构造函数 …… 101
5.3.2 复制构造函数 …… 105
5.3.3 组合类的构造函数 …… 106
5.3.4 析构函数 …… 108
5.3.5 浅拷贝和深拷贝 …… 110
5.4 综合案例——公司人员管理系统 5 …… 113
5.5 小结 …… 114
习题 5 …… 114

第 6 章 数据的共享与保护 …… 117

6.1 标识符的作用域与可见性 …… 117
6.1.1 作用域 …… 117
6.1.2 可见性 …… 120
6.2 存储类型和标识符的生存期 …… 121
6.2.1 存储类型 …… 121
6.2.2 标识符的生存期 …… 124
6.3 类的静态成员 …… 127
6.3.1 静态数据成员 …… 127
6.3.2 静态函数成员 …… 128

6.4 类的友元 …… 130
6.4.1 友元函数 …… 130
6.4.2 友元类 …… 131
6.5 常对象与常引用 …… 133
6.6 综合案例——公司人员管理系统 6 …… 137
6.7 小结 …… 137
习题 6 …… 138

第 7 章 继承与派生 …… 139

7.1 继承 …… 139
7.1.1 单一继承 …… 140
7.1.2 多重继承 …… 146
7.2 派生 …… 148
7.2.1 派生类的构造函数 …… 148
7.2.2 派生类的析构函数 …… 150
7.3 综合案例——公司人员管理系统 7 …… 156
7.4 小结 …… 158
习题 7 …… 159

第 8 章 多态性与运算符重载 …… 161

8.1 多态性 …… 161
8.1.1 通用多态和专用多态 …… 161
8.1.2 多态的实现 …… 162
8.2 抽象类 …… 170
8.3 运算符重载 …… 171
8.3.1 运算符重载的概念 …… 171
8.3.2 运算符重载为类的成员函数 …… 173
8.3.3 运算符重载为类的友元函数 …… 175
8.4 “++”和“--”的重载 …… 178
8.5 综合案例——公司人员管理系统 8 …… 180
8.6 小结 …… 181
习题 8 …… 181

第 9 章 流类库与输入输出 …… 184

9.1 输入输出的概念 …… 184
9.2 C++的基本流类体系 …… 185
9.3 标准输入输出流 …… 186
9.3.1 输出宽度控制：setw 和 width …… 187
9.3.2 填充字符控制：setfill 和 fill …… 188

9.3.3 输出精度控制：setprecision 和 precision …… 189
9.3.4 其他格式状态 …… 189
9.4 文件输入输出流 …… 190
9.4.1 文件的打开与关闭 …… 190
9.4.2 文件的读写 …… 191
9.4.3 文件读写位置指针 …… 194
9.5 综合案例——公司人员管理系统 9 …… 195
9.6 小结 …… 197
习题 9 …… 197

第 10 章 异常处理 …… 198

10.1 异常处理的概念 …… 198
10.1.1 C++异常处理的实现 …… 198
10.1.2 异常处理举例 …… 199
10.2 异常处理的注意事项 …… 201
10.3 小结 …… 201
习题 10 …… 201

附录 A 案例综合 …… 202

附录 B 参考答案 …… 213

参考文献 …… 227

第1章

绪　论

本章学习目标

- 了解程序设计语言的发展历史与程序设计语言的特点；
- 理解并掌握面向对象程序设计方法中涉及的基本概念与设计理念；
- 理解并掌握利用面向对象方法进行程序设计的基本过程。

本章主要向读者介绍程序设计语言的发展历史，低级语言、高级语言的特点及实际应用；同时简要介绍面向对象程序设计所涉及的一些基本概念，使读者对面向对象程序设计有个概要性的理解；最后详细介绍面向对象程序设计的基本步骤，为后续学习奠定基础。

1.1　程序设计语言简介

在人类社会生活中，“自然语言”是人所熟知的用来进行交流的工具。虽然国度不同使用的语言不同，但是所有的语言都是由语音、词汇和语法等构成的。在计算机的世界里，“程序设计语言”是人与计算机进行交流的工具，所谓的程序设计语言是计算机可以识别的语言，人类利用它来指挥计算机进行工作——用于解决生活中的问题。

计算机之所以能够实现很多的功能，其主要是依靠程序来实现的，而程序是为实现特定目标或解决特定问题用程序设计语言所编写的命令行序列的集合，程序规定了计算机执行的动作以及执行的顺序。程序设计语言按照是否能够被计算机所直接识别分为低级语言和高级语言，下面分别介绍这两类语言。

1.1.1　低级语言

在计算机诞生初期，程序员使用机器语言编写程序以达到控制计算机执行的目的。机器语言是由计算机硬件系统能够直接识别的二进制指令所组成的，我们将机器语言称为低级语言(Low-level Language)。由于计算机可以直接识别机器语言，因此对计算机来说，机器语言是较好的选择，但是对于人类来讲，机器语言却有很多的缺点：比如记忆比较烦琐，理解比较困难，而且开发效率低。于是针对这些缺点，人类研发出汇编语言。汇编语言是将机器指令映射为一些可以被人读懂的助记符，如用 ADD 表示加运算、用 SUB 表示减运算等。用汇编语言编写的程序计算机是不能直接识别的，所以需要经过“翻译”后才能转换成

计算机硬件系统可以识别的机器指令，这种翻译工具称为编译器。虽然汇编语言采用了助记符的形式来完成程序的编写，在很大程度上提高了记忆、辅助了理解，但是它仍然属于低级语言，程序员在利用汇编语言编写程序时还是需要考虑大量的人机交互细节。

1.1.2 高级语言

高级语言(High-level Language)的出现使计算机编程语言开启了新篇章，它使得计算机程序设计语言不再需要过度依赖于某种特定的计算机硬件或计算机环境配置，这是因为高级语言在不同的平台上会被编译成不同的计算机语言，而不是直接被计算机执行。高级语言忽略了计算机的硬件差别，屏蔽了机器底层的细节，提高了语言的抽象层次，更接近于人类的语言，其优点凸显在易学、易用、易维护等几个方面。用高级语言编写的程序称为“源程序”，与汇编语言编写的程序一样，计算机不能直接识别源程序，必须将源程序编译成二进制的机器指令才能在计算机上运行。

常见的高级语言有 C、C++、C#、Visual FoxPro 以及 Java 等，不同的语言适用于不同的场合。

1.1.3 面向对象的语言

面向对象语言(Object oriented Language)借鉴 20 世纪 50 年代的人工智能语言 LISP，引入了动态绑定的概念和交互式开发环境的思想；始于 20 世纪 60 年代的离散事件模拟语言 Simula67，引入了类的要领和继承，成形于 20 世纪 70 年代的 Smalltalk。它是以对象作为基本程序结构单元的高级程序设计语言，用于描述的设计是以对象为核心，而对象是程序运行时的基本单位。面向对象语言中包含了类、继承、对象、封装等概念。面向对象语言的发展主要有两个方向：一个方向是纯面向对象语言，例如常见的 Smalltalk、EIFFEL 等；另一个方向是混合型面向对象语言，即在过程式语言及其他语言中加入类、继承、封装等知识，常见的有 C++、Objective-C 等。

利用面向对象语言对客观系统进行描述时较为自然、贴近人的思维，更便于软件的扩充与复用，其主要特点可归纳如下 4 个：

(1) 识认性，系统中的基本构件可识认为一组可识别的离散对象。

(2) 类别性，系统中具有相同数据结构与行为的所有对象可组成一类。

(3) 多态性，对象具有唯一的静态类型和多个可能的动态类型。

(4) 继承性，在基本层次关系的不同类中共享数据和操作。

1.2 面向对象程序设计基础简介

面向对象程序设计中涉及一些相关的专业术语，读者需要对这些术语有一个初步认识才能更好地利用面向对象语言进行程序设计，下面简要介绍一些常用的术语。

1.2.1 面向对象方法的由来

所谓面向对象，就是以对象的观点来分析现实世界中的问题。从普通人认识世界的观

点出发，把事物进行分析、归类、综合，提取其共性并加以描述。在面向对象的系统中，世界被看成是独立对象的一个集合，对象之间通过“消息”相互通信。对象是由描述该对象的数据（又称为属性）和基于这些数据的行为（又称为方法）所组成的。

面向对象实际上是一种软件系统的分析、设计和实现方法。它是围绕真实世界的概念来组织模型的一种全新思维方法，其基本思想是：对问题空间进行自然分割，以更接近人类的思维方式建立问题域模型，以便对客观实体进行结构模拟和行为模拟，使设计出的软件尽可能直接地描述现实世界，构造出模块化的、可扩充的、维护性好的软件，并能够很好地控制软件的复杂性和降低软件的开发、维护费用。在面向对象方法中，对象是核心概念，所有面向对象的技术都是建立在这个概念的基础之上的。

1.2.2 面向对象的基本概念

1. 对象

对象是一种可以看得到、摸得到或者感知得到的客观实体，如键盘、汽车、空气、人、比赛规则等。这里所描述的对象虽然是现实世界的实体，但却可以把它应用到计算机领域，作为我们解决问题的出发点。利用计算机解决实际问题时，人们可以将现实世界中的对象与计算机软件操作中的抽象对象一一对应，利用现实世界中的实体用语来描述问题、分析问题。

2. 对象的模型化

（1）属性

属性（也称为静态特征）是用来描述对象内在的、本质的特征。简单地说，属性就是用来描述对象自身状态、性质的数据名称的集合，主要用来区别一事物与另一事物的不同，一般通过变量来定义。例如学生有学号、姓名、性别、家庭住址等数据名称；钢笔有长度、颜色、品牌以及用途等数据名称；汽车有颜色、品牌、排气量以及油耗等数据名称；比赛规则有制定方、制定时效以及针对对象等数据名称。

（2）方法

方法（也称为动态特征）是指对象在“外力”作用下产生的可以改变其部分或者全部属性值的动作行为的综合描述，一般通过函数来实现。例如学生的考试；钢笔的书写；汽车的刹车；比赛规则的修订等。

（3）封装

当一个对象含有完整的属性和与之相对应的方法时，称之为封装。封装是面向对象方法的一个重要原则，它把属性和方法结合在一个对象里，对象的属性（内部信息）对外界隐藏，只能通过对象提供的方法（接口）进行访问。对于外界来说，只能知晓对象的外部行为而无法了解对象的内部实现细节，这样可以保证对象属性数据的安全性。

封装有以下两层含义：

- 结合性。把对象的全部属性和方法结合起来，形成一个独立的不可分割的单位。
- 信息隐藏性。尽可能隐蔽对象的内部细节，对外形成一个边界，只保留有限的对外接口与外界发生联系。

封装的基本单位是对象，如钢笔、学生、轿车等。封装的目的在于将对象的使用者和对象的设计者分开，使用者不必知道行为（方法）实际的实现细节，只需调用设计者提供的接口

来访问该对象。把定义模块和实现模块分开，可以大幅提高软件的可维护性、可修改性。

1.3 面向对象软件开发简介

面向对象的软件开发不仅仅限于编码(面向对象编程)，还包括系统前期的分析(面向对象分析)与设计(面向对象设计)。分析与设计实际上就是对所要研究的现实世界进行建模的过程。

面向对象的分析、设计和编程有怎样的关系呢？面向对象分析的结果是生成一个模型，基于这个模型可以进行面向对象的设计，而面向对象设计的结果就是利用面向对象编程的方法实现整个软件系统的蓝图。三者之间是相辅相成、承前启后的关系。

1.3.1 软件分析

所谓“需求分析”，是指对要解决的问题进行详细的分析，弄清楚问题的需求，主要包括需要输入的数据，要得到的结果，最后能够输出的数据。简单地说，软件工程中的“需求分析”实质上就是确定计算机要“做什么”，最终需要达到什么样的效果。也就是说，需求分析是系统开发之前必须要做的一项工作。

在软件工程中，需求分析指的是在建立一个新的或改变一个现有的软件系统时，描写新系统的目的、范围、定义和功能时所要做的所有的工作。需求分析是软件工程中的一个关键过程。在这个过程中，系统分析员和软件工程师需要明确客户的需求。只有在明确了客户的这些需求后，他们才能够分析和寻求新系统的解决方法。需求分析阶段的任务是确定软件系统功能。

在软件工程的发展过程中，很长一段时间里人们一直认为需求分析阶段是整个软件开发过程中最为简单的一个阶段。但近些年来，越来越多的人意识到，实际上需求分析是整个软件开发过程中最为关键的一个阶段。假如在需求分析时系统分析员和软件工程师未能正确地理解、认识到客户的需要，那么最后的软件在实现上也不可能达到客户的需求，或者导致软件项目无法在规定的时间内完工。

需求分析的主要任务是通过详细调查现实世界要处理的对象，充分了解原系统工作的概况，明确客户的各种需求，然后在此基础上确定新系统的功能。通常对软件系统的需求分析有以下几个方面的综合要求：

- 功能需求；
- 性能需求；
- 可靠性和可用性需求；
- 出错处理需求；
- 接口需求；
- 约束；
- 逆向需求；
- 将来可能提出的要求。

需求分析的基本步骤包括以下几项：

1. 调查组织机构情况

主要是概况了解，包括了解该组织的部门组成情况，各部门的职能等信息，为分析信息流程提供必要的依据。

2. 调查各部门的业务活动情况

软件针对性环境的了解，包括了解各个部门输入和使用什么数据，如何加工、处理这些数据，输出什么信息，输出到什么部门，输出结果的格式是什么等问题。

3. 协助用户明确对新系统的各种要求

可以通过座谈、问卷以及电邮沟通等方式与客户进行良好沟通，主要包括客户的信息要求、处理要求、完全性与完整性要求等内容。

4. 确定新系统的边界

明确新系统应该实现的必要功能，主要是确定哪些功能由计算机完成或将来准备让计算机完成，以及哪些活动由人工完成。

5. 分析系统功能

6. 分析系统数据

7. 编写分析报告

1.3.2 软件设计

软件设计是以软件需求规格说明书为依据，根据需求分析阶段确定的软件功能进行设计软件系统的整体结构、划分功能模块、确定每个模块的实现算法以及编写具体的代码，形成软件的具体设计方案。

软件设计是把许多事物和问题抽象起来，将问题或事物进行分解并模块化使得所要解决的问题变得简单、容易，分解的越细模块数量相应的也就越多，虽然将问题进行模块化可以简化问题的解决方法，但是由于模块数过多，就会对应地产生一些副作用——使得设计者在软件的合成过程中需要更多地考虑模块之间耦合度的情况。因此，在软件设计过程中需要综合考虑实际情况，进行适量的模块划分。

模块化(Modularity)指的是软件可被分割为分别命名并可寻址的组件(也叫做模块)，将模块综合起来又可以满足问题的需求的性质。软件的模块化是允许智能化管理程序的唯一属性。

软件设计中包括几个主要的设计要素：结构设计、数据设计、接口设计和过程设计。其中结构设计是指定义软件系统各主要部件之间的关系；数据设计是指将模型转换成数据结构的定义；接口设计是指软件内部，软件和操作系统间以及软件和人之间如何通信；过程设计是指系统结构部件转换成软件的过程描述。

同时在软件设计过程中要遵守如下的设计原则：

(1) 设计对于分析模型应该是可跟踪的：软件的模块可能被映射到多个需求上。

(2) 设计结构应该尽可能地模拟实际问题。

(3) 设计应该表现出一致性。

(4) 不要把设计当成编写代码。

(5) 在创建设计时就应该能够评估质量。

(6) 评审设计以减少语义性的错误。

软件的设计是一个将需求转变为软件陈述(表达)的过程。系统通过采用逐步求精的方法使得软件设计陈述逐渐接近源代码。这里有两个基本步骤：第一步是概要设计(Preliminary Design)，该阶段主要完成如何将需求分析结果转换成数据和软件框架；第二步是详细设计(Detail Design)，该阶段主要完成如何将框架逐步求精细化为具体的数据结构和软件的算法表达。求精(Refinement)又叫做逐步求精，指的是通过程序细节连续细化来开发程序体系的策略。分步骤地对程序抽象进行分解直至成为编程语言的过程同时造就了程序的层次结构。

软件设计方法每天都在进化，作为已经经过测试和细化的方法，良好的设计应具有以下的4种特性，并在这些特性之间保持一致。

(1) 将信息领域的表达转换为软件设计的表达机制。

(2) 表示功能组件及其界面的符号。

(3) 逐步求精和分割的试探。

(4) 质量评估的指导方针。

设计过程中用以促成模块化设计的4个区域：模块、数据、体系和程序设计。模块设计(Modular Design)降低了程序的复杂性、便于修改且使得支持系统不同部分的并行开发实现起来更容易。模块类型提供的操作特性通过结合时间历史、激活机制以及控制模式来表现。在程序结构内部，模块可以被分类为：

(1) 顺序(Sequential)模块，由应用程序引用和执行，但不能从表观上中断。

(2) 增量(Incremental)模块，可被应用程序先行中断，而后再从中断点重新开始。

(3) 并行(Parallel)模块，在多处理器环境下可以与其他模块同时执行。

单独的模块更容易开发，因为功能可以被划分出来，而界面只是用来确保功能的独立。功能的独立性可以使用两个定性的标准来衡量：内聚性和耦合度。

数据设计是最重要的设计行为。数据结构的影响和程序上的复杂性导致数据设计对软件质量有着深远的影响。这种质量由以下的原理来实施：

(1) 适用于功能和行为分析的系统分析原理同样应该适用于数据。

(2) 所有的数据结构，以及各自所完成的操作都应该被确定。

(3) 创建数据词典并用来详细说明数据和程序的设计。

(4) 底层的数据设计决定应该延迟至设计过程的后期。

(5) 数据结构的陈述(具体说明)应该只被那些直接使用包含在此结构内的数据的模块所知道。

(6) 有用的数据结构和操作库可以在适当的时候使用。

(7) 软件设计和编程语言应该支持抽象数据类型的规范和实现。

体系设计的主要目标是开发模块化的程序结构并表达出模块间的控制相关性。另外，体系设计融合了程序结构与数据结构，以及使得数据得以在程序中流动的界面定义。这种方法鼓励设计者关注系统的整体设计而不是系统中单独的组件。选用不同的方法会采用不同的途径来接近体系的原点，但所有这些方法都应该认识到具有软件全局观念的重要性。

程序设计在数据、程序结构以及陈述详细算法的说明都已使用类似英语的自然语言来呈现后，再确定程序设计。使用自然语言来陈述的原因是当开发小组的绝大多数成员使用

自然语言来交流的话，那么小组外的一个新手在不经学习的情况下会更容易理解这些说明。这里有个问题：程序设计必须毫无歧义地来详细说明程序，但我们都知道不含糊的自然语言也就不自然了。

软件设计的重要性表现在软件的质量上。软件设计描述了软件是如何被分解和集成为组件的，同时也描述了组件之间的接口以及组件之间是如何发挥软件构建功能的。如何设计才能保证质量？通常需要遵守以下软件设计的一般原则：

- 要有分层的组织结构，便于对软件各个构件进行控制；
- 应形成具有独立功能特征的模块(模块化)；
- 应有性质不同、可区分的数据和过程描述(表达式)；
- 应使模块之间及与外部环境之间接口的复杂性尽量地减小；
- 应利用软件需求分析中得到的信息和可重复的方法。

1.3.3 软件编程

为了使计算机能够理解人的意图，人类就必须要将需解决的问题的思路、方法和手段通过计算机能够理解的形式告诉计算机，使得计算机能够根据人的指令一步一步去工作，完成某种特定的任务。这种人和计算机之间交流的过程就是编程。软件编程就是让计算机为解决某个问题而使用某种程序设计语言编写程序代码，并最终得到相应结果的过程。使用的程序设计语言不同，编写的程序就不同。

软件编程实际上是对软件详细设计结果的一个翻译。软件编码首先要注意编码工具的选择，其中编码工具的选择主要考虑以下内容：工程特性、技术特性、运行环境、算法和数据结构的复杂性以及开发人员的知识水平和能力。此外，还可以针对项目的应用领域来考虑选择编码工具。例如用面向对象思想开发系统，则需要用面向对象的开发工具 C++、Java 等，如果想开发网站则需要使用 JSP、JavaWeb 等。

程序的质量受软件设计结果所影响，好的编码是需要遵守一定的原则的，具体原则如下所述。

1. 基本原则

(1) 严格遵循软件开发流程，在详细设计的指导下进行代码编写。

(2) 编写代码以实现软件系统功能和性能为目标，要求正确完成设计要求的功能，达到性能需求。

(3) 具有良好的程序结构，提高程序的封装性，实现程序模块内部高内聚，程序模块间低耦合。

(4) 确保程序可读性强，易于理解，方便调试和测试。

(5) 易于使用和维护，尽可能实现高可重用性。

(6) 程序执行占用资源少，以低代价完成任务。

(7) 在保证程序可读性的情况下，提高代码的执行效率。

2. 编码风格

(1) 所有变量名的定义要直观，意义鲜明，类型符合数据实际处理的要求和特点。

(2) 采用缩进的格式，体现层次和逻辑对应关系，代码整体效果更明确。

(3) 适当使用空格，如运算符左右两边均加一个空格，格式更清晰。