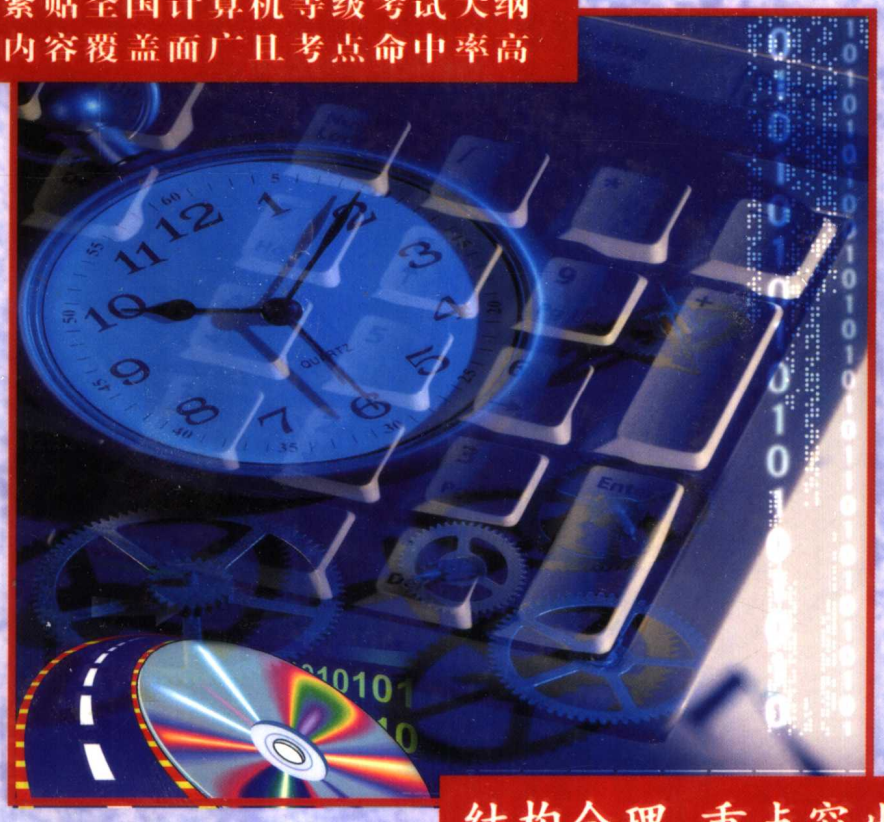
21世纪高等院校计算机教材

C/C++程序设计

黄春梅　徐宇清　主编
臧劲松　陈章　杨赞　黄小瑜　马立新　副主编
夏耘　主审

紧贴全国计算机等级考试大纲
内容覆盖面广且考点命中率高



结构合理　重点突出

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

21 世纪高等院校计算机教材

C/C++程序设计

主　编　黄春梅　徐宇清

副主编　臧劲松　陈　章
杨　赞　黄小瑜　马立新

主　审　夏　耘

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书根据国外的教学理念并结合国内程序设计教学的特点，由浅入深地介绍程序设计中的基本概念和C语言的语法，使读者掌握程序设计的典型技术，内容主要包括：程序设计基础知识、C语言的基本语法、程序控制结构、构造类型数据、函数、指针、文件，C++基础知识及6个与C语言有关的附录。此外，本书含配套光盘一张。

本书可作为高等学校本科生“C语言程序设计”课程的教材，也可作为C语言自学者的参考书。

图书在版编目（CIP）数据

C/C++程序设计/黄春梅，徐宇清等编著. —北京：中国铁道出版社，2007.1
（21世纪高等院校计算机教材）
ISBN 978-7-113-07742-6

Ⅰ.C… Ⅱ.夏… Ⅲ.C语言—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2007）第018000号

书　　名：C/C++程序设计
作　　者：黄春梅　徐宇清　等
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街8号）
策划编辑：严晓舟　秦绪好
责任编辑：苏　茜　崔晓静　黄园园
封面制作：白　雪
责任校对：王　欣
印　　刷：北京鑫正大印刷有限公司
开　　本：787×1092　1/16　**印张**：19　**字数**：444千
版　　本：2007年2月第1版　2007年2月第1次印刷
印　　数：1～5 000册
书　　号：ISBN 978-7-113-07742-6/TP·2107
盘　　号：ISBN 978-7-900144-99-7
定　　价：28.00元（含盘）

前言

编写程序是一项系统而烦琐的工作，它不仅需要程序设计人员具有一定的功底，更需要有良好的编程习惯和风格。良好的编程习惯和风格不仅可以使程序代码更易于读懂和修改，更重要的是，它可以使程序的结构更加合理，有助于提高程序的执行效率。因此，多年来，程序设计课程一直作为大学生的计算机基础课程。

C 语言以其小巧、灵活、高效等特点成为当今软件开发的主要编程语言，近年来不少高校已将 C 语言作为学生学习程序设计的入门语言，教育部考试中心及大部分省市将 C 语言程序设计纳入计算机等级考试的科目。

本书是根据 2005 年版《全国计算机等级考试二级考试大纲（C 语言程序设计）》、《上海市高等学校计算机等级考试（二级 C 程序设计）考试大纲》的要求，由全国命题组专家和一线教师结合多年的教学经验及对出题范围、重点和难点的研究，从学生学习和应试角度出发精心编写而成。

本书注重基本概念的系统化，叙述简明扼要，书中对教学的内容进行了有重点的讲解，对考试要求的知识点逐一进行了点拨，针对部分难点和重点，采用理论链接的方式，给出了相关知识和理论的分析，本书内容精炼、结构合理、重点突出，对读者可能遇到的难点做了十分清晰和详细的阐述。读者只需按本书的指引，就能在短时间内强化 C/C++程序设计考试的全部知识点。

本书每章都分为 6 个部分，分别是：第 1 部分“本章概要”，它紧扣 C 程序设计的考点；第 2 部分“学习目标”；第 3 部分教学内容；第 4 部分“应用实例”（旨在加深和巩固与本章相关的知识点，提高学生分析问题和解决问题的能力）；第 5 部分“本章小结”；第 6 部分“本章实验”。

本书既是一本教程又是一本实验指导，它整理了课上的教案，注重训练环节，体现了在理论指导下，让学生动手、动脑的基本思想方法，提出理性思维和理性实践。按照建构主义的学习理论，学生作为学习的主体，在与客观环境（指所学内容）的交互过程中构建自己的知识结构。本书引导学生在解题编程中探索其中带规律性的认识，将感性认识升华到理性高度，这样学生就能举一反三。本书可供各层面学生、教师、自学应试者阅读。

为了配合学习，本书提供了配套的光盘，其中包含一套学习自测系统、电子课件、程序

图解、算法演示、视频资料等国内外有关C/C++语言程序设计的素材文件，以及书中的源程序文件，供读者参考。

本书主编为上海理工大学计算机基础教学专家黄春梅、徐宇清，副主编为臧劲松、陈章、杨赞、黄小瑜、马立新等，参加编写的还有刘小昵、马晓旦、吴存孝、黄玉林、黄鹤鸣、高建，担任本书主审是计算机等级考试命题组专家夏耘。

在本书的编写过程中，还得到了清华大学、上海交通大学、复旦大学、华东师范大学、上海海事大学、华东理工大学、上海理工大学、上海大学等高校计算中心各位老师的帮助，在此一并致谢。

由于时间仓促和编者水平有限，书中难免存在一些不妥之处，敬请广大读者批评指正。

编　者

2007年1月

目 录

第 1 章 绪论 1
1.1 程序与程序设计 1
1.1.1 程序设计的基本概念 2
1.1.2 程序设计基本方法与原则 7
1.2 算法 10
1.2.1 算法的概念和主要特性 11
1.2.2 算法的描述 12
1.3 应用实例 13
1.4 C 语言 15
1.4.1 C 语言是程序员的语言 15
1.4.2 C 语言程序结构 15
1.4.3 C 语言编程风格 16
本章小结 17
本章实验 17
第 2 章 编程基础 18
2.1 标识符、保留字和表 18
2.1.1 标识符和保留字 18
2.1.2 表列 19
2.1.3 可列性（可数性） 20
2.2 基本的数据类型 20
2.2.1 整型 21
2.2.2 浮点型（实型） 21
2.2.3 字符型 22
2.3 常量和变量 22
2.3.1 常量 22
2.3.2 变量 22
2.3.3 变量的定义和预置初值 23
2.4 运算符与表达式 23
2.4.1 常用运算符 24
2.4.2 表达式 25
2.4.3 算术表达式 25
2.4.4 赋值表达式 26
2.4.5 自增和自减表达式 27
2.4.6 条件表达式 27

2.4.7 逗号表达式 28
2.4.8 位运算表达式 29
2.5 语句 31
2.5.1 语句概述 31
2.5.2 赋值语句 32
2.6 数据的输入和输出 32
2.6.1 转义字符 33
2.6.2 格式编辑 33
2.6.3 格式输出函数 35
2.6.4 格式输入函数 36
2.6.5 单个字符的输入/输出函数 38
2.6.6 字符串的输入/输出函数 39
2.7 应用实例 41
本章小结 42
本章实验 42
第 3 章 流程控制 **44**
3.1 逻辑问题的解决方案 44
3.1.1 穷举法 45
3.1.2 迭代法 46
3.2 关系运算 47
3.3 逻辑运算 48
3.4 选择结构控制语句 50
3.4.1 if 语句 50
3.4.2 多分支的 switch 语句和 break 中断跳转语句 57
3.5 循环结构控制语句 62
3.6 应用实例 69
本章小结 75
本章实验 75
第 4 章 构造类型数据 **78**
4.1 数组 78
4.1.1 一维数组 79
4.1.2 二维数组 83
4.1.3 字符数组与字符串 87
4.2 结构体 94
4.2.1 结构体类型的定义和变量的声明 95
4.2.2 结构体变量的存储与成员的引用 97
4.2.3 结构体数组 99

4.3 共用体 102
4.3.1 共用体类型的定义和变量的声明 102
4.3.2 共用体变量的存储和成员的引用 103
4.4 枚举类型 107
4.4.1 定义枚举类型 107
4.4.2 声明枚举类型变量 108
4.5 应用实例 109
本章小结 122
本章实验 123
第 5 章 函数 **129**
5.1 函数概述 129
5.1.1 函数的定义 130
5.1.2 函数的形式参数和实在参数 133
5.1.3 函数的返回值 135
5.1.4 函数的调用 137
5.1.5 数组作为函数参数 142
5.1.6 内部函数和外部函数 150
5.2 函数的嵌套调用 151
5.3 函数的递归调用 154
5.4 变量的作用域和存储类别 158
5.4.1 变量的作用域 158
5.4.2 变量的存储类别 163
5.4.3 多文件程序的运行 170
5.5 应用实例 171
本章小结 177
本章实验 178
第 6 章 指针 **181**
6.1 指针概述 182
6.1.1 指针变量和基类型 182
6.1.2 指针变量的定义和引用 182
6.1.3 指针变量的运算 184
6.1.4 指向一维数组或二维数组的指针 186
6.1.5 指针指向函数 191
6.1.6 指针指向字符串 193
6.1.7 指针数组 196
6.2 指向指针的指针 197
6.3 函数调用返回指针值 199
6.4 带参数的 main 函数 201

6.5 动态数据结构——链表 203
6.6 应用实例 210
本章小结 214
本章实验 215
第 7 章 文件 216
7.1 文件概述 216
7.1.1 C 的文件系统 216
7.1.2 文件指针 217
7.2 文件的打开与关闭 218
7.2.1 文件的打开 218
7.2.2 文件的关闭 219
7.3 文件的读写 220
7.3.1 fputc 和 fgetc 函数 220
7.3.2 fputs 和 fgets 函数 222
7.3.3 fwrite 和 fread 函数 223
7.3.4 fscanf 和 fprintf 函数 225
7.4 文件的定位与出错检测 226
7.4.1 文件的定位 226
7.4.2 文件的出错检测与处理 229
7.5 应用实例 230
本章小结 231
本章实验 232
第 8 章 C++基础知识 233
8.1 C++对 C 语言的改进 233
8.1.1 语法方面的变化 234
8.1.2 指针与引用 235
8.1.3 函数的改进 235
8.2 类与对象 238
8.2.1 面向对象程序设计特点 239
8.2.2 类 240
8.2.3 对象 241
8.2.4 构造函数和析构函数 241
8.3 继承与派生 249
8.3.1 继承与派生含义 249
8.3.2 派生类的定义 250
8.3.3 访问控制 251
8.3.4 派生类的构造函数和析构函数 255
8.4 多态性 258
8.4.1 函数重载 258

8.4.2 运算符重载 259
8.4.3 虚函数 263
8.4.4 纯虚函数和抽象类 265
8.5 应用实例 266
本章小结 275
本章实验 276
附录 A 标准 ASCII 编码对照表 277
附录 B C 语言主要关键字及其用途 280
附录 C C 语言运算符优先级和结合性 281
附录 D C 语言常用库函数 282
附录 E 实验报告格式 285
附录 F 编译预处理 287
参考文献 292

第1章 绪论

本章概要

计算机技术已经渗透到我们生活的各个领域，程序也不再是只有当打开计算机时才存在的东西，它深入我们的生活，有些已经成为我们生活的必需品。

所以，现代大学生要了解它、学习它。

C 语言具有语言简洁、紧凑、灵活；运算符和数据类型丰富；程序设计结构化、模块化；生成目标代码质量高；可移植性好；它将成为学习程序设计的切入点。

本章将论述程序设计的基础知识，包括：

- 程序的定义、构造、执行
- 算法的定义、特性、描述方法
- C 语言的产生及发展
- C 程序的结构

学习目标

完成本章的学习后能够：

- 了解程序存储和程序执行的基本概念，知道程序和指令是如何工作的
- 了解程序设计有哪些基本方法，可以用哪些计算机语言来实现程序设计
- 了解算法的基本概念，可以用 N-S 图解题
- 了解 C 语言的发展史
- 了解 C 的编程风格

1.1 程序与程序设计

程序是什么？程序与我们有什么关系？许多人在生活中并不会去考虑这个问题，许多人认为“程序”是一个与我们毫不相干的东西。果真是这样吗？

当在马路上出行的时候，当然会听从交通信号的指挥，那么，又是谁在指挥着这些交通信号呢？是程序！是交通控制系统的控制程序。

当在超市购买了称心的商品到付款处结账时，收银员拿起购买的物品在 POS 机上轻轻一划，很快账单就呈现在客户面前，是谁在完成这件事？程序。

当用磁卡在 ATM 机上存取款，当用交通卡乘坐地铁，当拿起手机与朋友通话，当打开电视观看最喜爱的节目，……在做这一切的时候，程序就在我们的周围，程序就在为我们服务。

本节将讨论程序设计的一些基本概念和基本方法，程序设计应遵循的主要原则和编码风格。

1.1.1 程序设计的基本概念

我们现在所使用的计算机，大部分都被称为“冯·诺依曼”型计算机，所谓“冯·诺依曼”型计算机的基本原理就是程序存储、程序执行。计算机只是一个电子设备，它那么神通广大，是因为有了程序在指挥控制它的运行，所以，程序才是计算机真正的“灵魂”，没有程序的计算机只是一堆废铁，它不能做任何事情。

如何赋予计算机“灵魂”，让它按照我们设想的要求去执行操作，去完成任务，这就是程序设计。

1. 指令与指令系统

前面提到，计算机之所以能够“聪明”地工作是因为它有程序在指挥控制它。而程序又是由一系列指令构成的，不同的指令构成了不同的程序，完成不同的任务。

在谈到指令这一概念时，必须强调：计算机每执行一条指令，只是完成了一个基本操作，一系列的指令构成了一系列的操作，这才是程序所完成的任务。所以，也有这样一句话：程序是一系列相关指令的集合。

既然一条指令执行一个基本操作，那么，计算机为了有所适从就必须从指令中得到如下信息：

（1）要执行的是什么类型的操作？

（2）被操作的数据在何处？

（3）操作后的结果放在何处？

指令中是如何给出这些信息的呢？通常一条指令被分为“操作码”和“操作数”两部分。前者用于说明操作类型，后者则用于说明操作数据的存储位置（地址）。

还要指出的是，正如前面所说程序是由一系列相关指令构成，根据“冯·诺依曼”型计算机的基本原理，程序先存储在内存储器中，计算机执行当前指令的时候，还必须获得下一条指令存放的位置（指令的地址），这一任务在计算机中由指令计数器（PC）完成。

2. 程序与程序设计语言

在有关指令的叙述中，有一个名词值得引起注意：指令系统。它是什么含义呢？所谓指令，是计算机硬件能够识别并可直接执行的操作命令，一台计算机中所有能够被识别的指令的集合就称为这台计算机的指令集，或称为“指令系统”。

如果计算机硬件不同，那么它能识别的指令也就不同，所以我们说：指令系统是“面向机器”的。换一种不同类型的计算机，也许就意味着必须换一种指令系统。

“冯·诺依曼”型计算机的另一特点是采用二进制原理编码和运算，计算机所产生的丰富多彩的图文声像和数据信息，在其内部其实只有“1”和“0”这两种状态的排列组合和逻辑运算，而且也只能被特定的计算机硬件所识别，这种符号构成的二进制代码对于程序编写者而言实在是苦不堪言，而且，一旦计算机升级换代，用二进制代码编写的程序就意味着全部失效，这是不可想象的。

所以，如何解决程序设计在这方面的困惑，如何使我们方便程序设计而又不依赖于计算机硬件，这就是所讨论的程序设计语言的要点。

计算机语言是进行程序设计的工具，它随着科技的进步不断发展，按照计算机语言的发展过程，大致可将计算机语言分成 3 种类型。

（1）机器语言

机器语言有如下特点：

① 机器语言是任何计算机能够直接识别和执行的语言。

② 因为不存在一个翻译过程，所以机器语言的执行速度快。

③ 设计人员必须熟知机器指令的所有二进制符号。

④ 用机器语言编写的程序在不同种类的计算机上不能通用。

如果说上述机器语言的特点中①和②是它的优点的话，那么机器语言特点中的③和④就是它的缺点。

下面这段代码是时钟音乐报时控制系统中的程序片段：

```
…10100000011001001001111100110110011000000101001000000000000000011100111000
0001010001010110010000010001011110001011100010010111100001100010000010101
0001100001000000011100011110011110001111001110011000011111011101100100000
0010000000011000010001100011000010000110101010101010111010010101010101111
1101010100100101010100111101010101101111111111101010101110101010101111110100
0100100101110101010100010000110100010100101011001011010101000100010101000
0101001010110011110101010100010000010100010100101011001011010101010001000
0010100010100101011001011010101010001000011010001010010101100101101010101
0001000001010001010010101100100101010101000100010101000101001010110011010
1010101000100011101000101001010110011111010101010001000101010001010011011100011101100010111011101100001010000001101010010000000000101001000001100000
0000100100000000010100100001010000000000000000100100000000001100100000011100100000011000000000000011000000000000000000000000000000000000000000000000…
```

看上去枯燥而乏味，其实以上这一大堆的数据只是该程序中极小的一个片段，由此可见，使用机器语言无论对于编写程序的方便程度或对于调试程序都存在着极大不便。

（2）汇编语言

机器语言是由一组组二进制代码组成的指令系统，如果能够用--些比较容易记住的符号来表示这些枯燥难记的二进制代码，则会给程序编写带来很大的方便。

汇编语言就是这样一种使用“助记符”的语言，下面看一下使用汇编语言编写的应用于时钟音乐报时控制系统中的程序片段：

```
;************************************************
;*      1993.11.26  FOR ZHONG NORMAL MUSIC      *
;*                  ADD DOG COUNT               *
;*      WANGWEI     SHANGHAI NORMAL UNIVERSITY  *
;************************************************
INIT:   CLR P3.5        ;START THE DOG COUNTING
        MOU A,#OOH      ;MAIN PROGRAM START HERE
        MOU RO,#1OH     ;INITIAL RAM AREA
```

```
        MOU R2,#7OH
LOP1:   MOU @RO,A
        INC RO
INIA:   DJNZ R2,LOP1
        MOU SP,#RAND+1
        SJMP INIB
        ORG OO1BH
        LJMP TM1
INIB:   MOU ZDT,#174
        MOU TEMPO,#15H
        MOU XPON,#OCH
        MOU PITN,#O8H
        MOU RO,#1OH          ;INITIAL SUBROUTINE
        MOU R1,#2OH
        MOU DATA,#OOH
        MOU R2,#O9H
LOP3:   MOU A,RO
        LCALL OUTPUT
        MOU A,R1
        LCALL OUTPUT
        INC R0
        INC R1
        DJNZ R2,LOP3
        MOU A,#OEH
        LCALL OUTPUT
        MOU A,#OOH
...
```

在汇编语言中，枯燥单调的二进制代码被一些表示指令功能的英文缩写词（助记符）所替代，显然，MOV 可以表示传送操作，ADD 可以表示加法操作，而 LJMP 则表示了程序的跳转（Long Jump），这确实给编写程序带来了很大的方便。

不过，在了解汇编语言的同时，必须知道它的如下特点：

① 汇编语言虽然相对于机器语言方便了编程，但它同机器语言一样，也属于“低级语言”，它是“面向机器”的，也就是说，不同种类的计算机有着不同的汇编语言。上述我们看到的时钟控制系统是用 8051 系列单片微机的汇编语言编写的，如果更换了其他系列的 CPU，则整个控制程序就必须重新编写。

② 用汇编语言编写的程序并不能像二进制语言那样直接被计算机识别和运行，它必须通过“编译”和“连接”，翻译成机器语言后才能被计算机执行。

（3）高级语言

汇编语言虽然克服了二进制代码枯燥难记的问题，但它毕竟只是使用了帮助记忆的“助记符”，而且，由于它“面向机器”的特性，它的通用性还是很差。

如果能够使用类似日常生活中使用的自然语言来描述程序所要实现的功能，所编写的程序在不同的计算机中都可以使用，那将给程序设计带来极大的方便。高级语言就是这样一种使用自然语言，不依赖于具体的计算机而只是面向要解决的问题的程序设计语言。

下面这段代码是自动阅卷系统中的程序片段：

```
...
**********用读二进制文件的方法读出软盘"考生信息.wei"文件中的姓名和准考证号
  If fsox.FileExists("考生信息.WEI") Then '若考生信息文件存在
    Open "a:\考生信息.wei" For Binary As #1
    Seek #1, 355
    Get #1, 355, xh
    Seek #1, 366
    Get #1, 366, mz
    Close #1
    If Trim(xh) <> "" Then
      kszkz.Caption = xh
      ksname.Caption = "待查" ' mz 'RTrim(Mid(mz, 1, 4))
      MsgBox "阅卷系统已从考生信息文件中查到该生准考证号", 64, "信息"
    Else
      MsgBox "这是一张空盘,考生未进行登录", 64, "注意"
      Exit Sub
    End If
  Else
    MsgBox "这张软盘不是考试专用盘,请核对！ ", 64, "注意"
    Exit Sub
  End If
...
```

可以观察一下上述程序片段，在用高级语言编写的程序代码中，大量使用了诸如 If（如果）怎样、Then（那么）如何、Else（否则）怎样的自然语言，如果要显示一个信息对话框，则可以在程序中给出诸如 MsgBox 的命令语句。这大大方便了程序的编写、测试。而且，其更大的好处还在于，高级语言是“面向问题”的，使用高级语言编写的程序，具有很好的通用性和可移植性。当然，使用高级语言编写的程序，需要经过专门的翻译过程才能被计算机识别和执行。

高级语言的发展也经历了从早期语言到结构化程序设计语言，从面向过程到非过程化程序语言的过程。相应地，软件的开发也由最初的个体手工作坊式的封闭式生产，发展为产业化、流水线式的工业化生产。

20 世纪 60 年代中后期，软件越来越多，规模越来越大，而软件的生产基本上是人自为战，缺乏科学规范的系统规划与测试、评估标准，其恶果是大批耗费巨资建立起来的软件系统，由于含有错误而无法使用，以致带来巨大损失，软件给人的感觉是越来越不可靠。这一切，极大地震动了计算机界，被称为“软件危机”。人们认识到：大型程序的编制不同于写小程序，它应该是一项新的技术，应该像处理工程一样处理软件研制的全过程。程序的设计应易于保证正确性，也便于验证正确性。1969 年，提出了结构化程序设计方法，1970 年，第一个结构化程序设计语言——Pascal 语言的出现，标志着结构化程序设计时期的开始。

20 世纪 80 年代初开始，在软件设计思想上，又产生了一次革命，其成果就是面向对象的程序设计。在此之前的高级语言，几乎都是面向过程的，程序的执行是流水线似的，在一个模块被执行完成前，人们不能干别的事，也无法动态地改变程序的执行方向。这和人们日常处理事物的方式是不一致的，对人而言是希望发生一件事就处理一件事，也就是说，不能面向过程，而应是面向具体的应用功能，也就是对象（Object）。其方法就是软件的集成化，如同硬件的集成电路一样，生产一些通用的、封装紧密的功能模块，称之为软件集成块，它与具体应用无关，但能相互组合，完成具体的应用功能，同时又能重复使用。对用户来说，只关心它的接口（输入量、输出量）及能实现的功能，至于如何实现的，那是内部的事，用户完全不用关心，Visual Basic（以下简称 VB）、Delphi 就是典型代表。

高级语言的下一个发展目标是面向应用，也就是说，只需要告诉程序自己要干什么，程序就能自动生成算法，自动进行处理，这就是非过程化的程序语言。

程序设计语言（Programming Language）是用于编写计算机程序的语言。语言的基础是一组记号和一组规则。根据规则由记号构成的记号串的总体就是语言。在程序设计语言中，这些记号串就是程序。程序设计语言包含 3 个方面，即语法、语义和语用。语法表示程序的结构或形式，亦即表示构成程序的各个记号之间的组合规则，但不涉及这些记号的特定含义，也不涉及用户；语义表示程序的含义，亦即表示按照各种方法所表示的各个记号的特定含义，但也不涉及用户；语用表示程序与使用的关系。

程序设计语言的基本成分有：

① 数据成分，用于描述程序所涉及的数据。

② 运算成分，用以描述程序中所包含的运算。

③ 控制成分，用以描述程序中所包含的控制。

④ 传输成分，用以表达程序中数据的传输。

程序设计语言按照语言级别可以分为低级语言和高级语言。低级语言有机器语言和汇编语言。按照用户的要求有过程式语言和非过程式语言之分。过程式语言的主要特征是，用户可以指明一系列可顺序执行的运算，以表示相应的计算过程，如 FORTRAN、COBOL、Pascal 等。

按照应用范围，有通用语言与专用语言之分。如 FORTRAN、COLBAL、Pascal、C 等都是通用语言。目标单一的语言称为专用语言，如 APT 等。

按照使用方式，有交互式语言和非交互式语言之分。具有反映人机交互作用的语言成分的语言称为交互式语言，如 BASIC 等。不反映人机交互作用的语言称为非交互式语言，如 FORTRAN、COBOL、ALGOL60、Pascal、C 等都是非交互式语言。

按照成分性质，有顺序语言、并发语言和分布语言之分。只含顺序成分的语言称为顺序语言，如 FORTRAN、C 等。含有并发成分的语言称为并发语言，如 PASCAL、Modula 和 Ada 等。

程序设计语言是软件的重要方面，其发展趋势是模块化、简明化、形式化、并行化和可视化。

1.1.2　程序设计基本方法与原则

1．程序设计方法学的研究

20 世纪 60 年代中期以后，计算机硬件技术日益进步，计算机的存储容量、运算速度和可靠性明显提高，生产硬件的成本不断降低。计算机价格的下跌为它的广泛应用创造了极好的条件。在这种形势下，迫切要求计算机软件也能与之相适应。因而，一些开发大型软件系统的要求提了出来。然而软件技术的进步一直未能满足形势发展的需要，在大型软件的开发过程中出现了三大难题：

（1）复杂程度高

（2）研制周期长

（3）正确性难以保证

由于遇到的问题找不到解决办法，致使问题堆积起来，形成了人们难以控制的局面，出现了所谓的“软件危机”。

最为典型的例子是美国 IBM 公司于 1961 年～1964 年开发的 IBM360 系列机的操作系统。该软件系统花了大约 5 000 人/年的工作量，最多时，有 2 000 人投入开发工作，写出近一百万行的源程序。尽管投入了这么多的人力和物力，得到的结果却极其糟糕。据统计，这个操作系统每次发行的新版本都是从前一版本中找出 1 000 个程序错误而修正的结果。

IBM360 操作系统的历史教训已成为软件开发项目中的典型事例被记入历史史册。

如果开发的软件隐含错误，可靠性得不到保证，那么在运行过程中很可能对整个系统造成十分严重的后果。例如，1963 年，美国用于控制火星探测器的计算机软件中的一个“,”号被误写为“.”，而使飞往火星的探测器发生爆炸，损失惨重。

为了克服这一危机，一方面需要对程序设计方法、程序的正确性和软件的可靠性等问题进行系列的研究；另一方面，也需要对软件的编制、测试、维护和管理的方法进行研究，从而产生了程序设计方法学。

什么是程序设计方法学呢？简言之，程序设计方法学是讨论程序的性质、程序设计的理论和方法的一门学科。它包含的内容比较丰富，例如，结构程序设计、程序正确性证明、程序变换、程序的形式说明与推导、程序综合、自动程序设计等。

在程序设计方法学中，结构程序设计占有十分重要的地位，可以说，程序设计方法学是在结构程序设计的基础上逐步发展和完善起来的。

2．结构化程序设计

（1）结构化程序设计

结构化程序设计的思想是在 20 世纪 60 年代末、70 年代初为解决“软件危机”而形成的。多年来的实践证明，结构化程序设计策略确实使程序执行效率提高，并且由于减少了程序的出错率，而大大减少了维护费用。

那么，什么是结构程序设计呢？至今仍众说纷纭，还没有一个既严格，又能被大家普遍接受的定义。

现在，较为认同的定义为：结构程序设计就是一种进行程序设计的原则和方法，按照这种原则和方法可设计出结构清晰、容易理解、容易修改、容易验证的程序。即结构化程序设