C/C++程序设计综合实践教程

主编　张东阳　孟力军

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内容简介

本书以培养学生程序设计能力为目标，以项目设计为主线，以 Dev-C++为开发平台，把理论教学与实践教学有机地结合起来，使学生较好地掌握大学程序设计类课程高效的学习方法，实现由高中向大学的顺利过渡，使教师切实提高课程教学质量，快速培养更多的高质量应用型人才。

本书共 12 章，主要内容包括计算机程序设计概述、C/C++开发工具 Dev-C++的使用、C/C++程序设计基础、顺序结构及其应用程序设计、选择结构及其应用程序设计、循环结构及其应用程序设计、数组及其应用程序设计、函数及其应用程序设计、结构体及其应用程序设计、链表及其应用程序设计、文件及其应用程序设计、编译预处理与源程序在线测评系统等。本书提供配套电子课件和程序代码等。

本书可作为普通高等院校本科计算机类、信息类，以及其他专业开设的 C 语言程序设计及其相关课程的教材或参考书，也可作为初学者学习 C/C++程序设计的入门教材，尤其可作为参加信息学竞赛的中小学生学习 C/C++程序设计的快速入门教材，还可供有关工程技术人员参考。

图书在版编目（CIP）数据

C/C++程序设计综合实践教程 / 张东阳，孟力军主编
. --北京：北京理工大学出版社，2022.11

ISBN 978-7-5763-1820-3

Ⅰ. ①C… Ⅱ. ①张… ②孟… Ⅲ. ①C 语言-程序设计-教材 ②C++语言-程序设计-教材 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2022）第 208202 号

出版发行 / 北京理工大学出版社有限责任公司
社　　址 / 北京市海淀区中关村南大街 5 号
邮　　编 / 100081
电　　话 / （010）68914775（总编室）
（010）82562903（教材售后服务热线）
（010）68944723（其他图书服务热线）
网　　址 / http：//www.bitpress.com.cn
经　　销 / 全国各地新华书店
印　　刷 / 唐山富达印务有限公司
开　　本 / 787 毫米×1092 毫米　1/16
印　　张 / 13.25　　责任编辑 / 李　薇
字　　数 / 311 千字　　文案编辑 / 李　硕
版　　次 / 2022 年 11 月第 1 版　2022 年 11 月第 1 次印刷　　责任校对 / 刘亚男
定　　价 / 88.00 元　　责任印制 / 李志强

前　言

高质量应用型人才培养是困扰高校教师多年的一个难题。如何对学生进行能力培养和价值塑造，切实提高学生的设计能力、实践能力、创新创业能力及其他相关能力，引领学生健康成长，快速培养大批高质量应用型人才，是摆在每个高等教育工作者案头的重大课题。

初高中阶段主要是知识的学习，教师、家长和学生的目标完全一致，都在为考上一个理想的大学全力以赴。由于时间有限，为了提升学生的创新和创造能力，只是在课程教学中增加一些实践（或实验）类教学项目，这对提升学生的实践能力和创新能力是完全不够的。为了适应信息技术的飞速发展，大幅提升学生的创新和创造能力，快速培养大批高质量的信息学人才，信息学竞赛应运而生，并成为培养和选拔信息学优秀人才的主要途径，这种人才选拔方式受到了初高中学生及其家长的普遍欢迎。

在大学阶段，除了知识的学习之外，主要是能力的培养和价值的塑造。虽然每一个专业的每一门课程都制订了较为详尽的能力培养目标，但在实施过程中大部分课程仍然只是通过一些实践（或实验）类教学项目提升学生的实践能力，这对学生的能力培养和价值塑造显然是不够的，也很难快速培养大批高质量应用型人才。为此，我们经过十多年深入的研究、探索、创新与实践，成功创建了一种切实可行、行之有效、便于大规模推广应用的基于能力培养和价值塑造的高质量应用型人才培养模式，即能力驱动人才培养模式，它较好地解决了高等教育多年来存在的本科课程教学质量问题和人才培养质量问题，让学生在理论联系实际上得到质的飞跃，迸发出强大的发展动能，较好地实现能力培养与价值塑造并举的人才培养目标，可快速培养大批高质量应用型人才。

在能力培养和价值塑造的高质量应用型人才培养模式的基础上，我们编写了《C/C++程序设计综合实践教程》，使学生快速掌握 C/C++程序，习得高效的大学课程学习方法，大幅提升设计和实践能力，实现由高中到大学的顺利过渡，奠定其他课程的学习基础。本书的主要特点如下。

（1）以解决问题为核心，为学生提供一种高效解决问题的思路和方法。

（2）以能力培养为目标，让学生在学习的过程中，有目标，有思路，有方法，有能力，有信心，有成果，有动力，有责任，有担当，从而达到事半功倍的效果。

（3）以算法设计为基础，对每一个问题都尽可能地提出较为详尽的算法分析和算法设计的思路和方法，使学生较好地掌握算法的分析和设计。

（4）以工具先行为手段，通过一个简单的程序，使学生能快速掌握 C/C++程序设计工具的使用方法，把枯燥的语言学习变为具有一定挑战性的编程项目，增强学生的学习信心，激发其学习兴趣，增强其自豪感和成就感。

（5）以程序设计为主线，通过几十个项目的练习，在反反复复的设计过程中，学生可以快速掌握 C/C++程序设计的思路和方法。

（6）课外练习以设计性作业为主，在课程教学实例的基础上，合理地设计一系列综合性的课外设计作业，进一步提高学生单片机应用系统的设计能力，激发其学习兴趣，培养其爱好，提高其自信心和成就感，让学生在反反复复的设计过程中进一步掌握 C/C++程序设计的思路和方法。

（7）以信息学竞赛为目标实现场景，为学生提供一个虚拟的、可大显身手的实践舞台。

（8）理论与实践相结合，通过几十个项目的算法分析和程序设计，学生得以在较短的时间内，快速掌握 C/C++程序设计，大幅提升分析问题和解决问题的能力，大幅提升编程能力和创新能力，快速实现由高中到大学的顺利过渡，为以后的学习奠定坚实的基础。

本书共 12 章，以高质量应用型人才培养为目标，把理论教学、实践教学、课程设计等有机地结合在一起，教学参考学时为 32～64 学时，有关章节内容可根据教学计划要求和学时情况酌情调整。

本书由沈阳理工大学张东阳、孟力军担任主编，其中第 4、5、6、7、8、9、10、11 章由张东阳编写，第 1、2、3、12 章由孟力军编写。在本书的编写过程中，清华大学、北京大学、北京师范大学、北京邮电大学、北京理工大学、中山大学、哈尔滨工业大学、吉林大学、大连理工大学、天津大学、香港大学、香港科技大学、香港城市大学、澳门大学、剑桥大学、帝国理工学院、华威大学、斯坦福大学、哥伦比亚大学等一些国内外高校的学生先后提出了许多有利于初学者学习和使用的意见和建议，在此也向他们表示衷心的感谢。

在本书的编写过程中，除了参考文献所列出的书籍和资料外，还参阅了其他书籍和网上资料，在此向所有作者表示衷心的感谢。

在本书的编写过程中，北京理工大学出版社的各位编辑为本书的编著提供了许多宝贵的意见、建议和帮助，在此一并表示感谢。

由于编著者知识水平和经验有限，书中难免存在不足和疏漏之处，恳请广大读者斧正。作者联系 Email：dongyangz@163.com。

编　者

2022 年 10 月

目 录

CONTENTS

第 1 章 计算机程序设计概述 …… 1

1.1 计算机系统组成 …… 1

1.1.1 计算机硬件系统 …… 1

1.1.2 计算机软件系统 …… 2

1.2 程序设计语言 …… 3

1.2.1 机器语言 …… 3

1.2.2 汇编语言（符号语言） …… 4

1.2.3 高级语言 …… 4

1.3 计算机算法 …… 7

1.3.1 算法的基本概念 …… 7

1.3.2 算法的表示方法（描述方式） …… 9

1.3.3 算法设计要解决的一些基本问题 …… 13

1.4 程序的基本结构和流程图 …… 14

1.4.1 顺序结构 …… 14

1.4.2 选择结构 …… 15

1.4.3 循环结构 …… 16

1.5 C/C++程序设计概述 …… 17

1.5.1 C/C++简介 …… 17

1.5.2 C/C++程序示例 …… 19

1.5.3 C/C++程序编译和执行 …… 21

1.5.4 C/C++开发工具简介 …… 21

第 2 章 C/C++开发工具 Dev-C++的使用 …… 24

2.1 C/C++开发工具 Dev-C++简介 …… 24

2.2 文件（项目）管理规划 …… 24

2.3 Dev-C++的使用 …… 25

2.3.1 启动 Dev-C++ …… 25
2.3.2 新建源程序 …… 26
2.3.3 保存源程序 …… 26
2.3.4 编译运行 …… 27
2.3.5 重新编译、运行 …… 27
2.3.6 C 语言源程序与 C++源程序比较分析 …… 28
2.4 Dev-C++的进一步使用 …… 29

第 3 章 C/C++程序设计基础 …… 32

3.1 程序设计基本知识 …… 32
3.1.1 位和字节 …… 32
3.1.2 基本数制 …… 33
3.1.3 数制相互之间的转换 …… 35
3.1.4 数值编码 …… 38
3.2 标识符与关键字 …… 42
3.3 数据类型 …… 43
3.4 变量与常量 …… 46
3.4.1 变量 …… 46
3.4.2 常量 …… 49
3.5 数据类型转换 …… 52
3.5.1 自动类型转换 …… 53
3.5.2 强制类型转换 …… 53
3.6 运算符和表达式 …… 54
3.6.1 运算符及其运算优先级 …… 54
3.6.2 算术运算符与算术表达式 …… 55
3.6.3 赋值运算符与赋值表达式 …… 56
3.6.4 逗号运算符和逗号表达式 …… 57
3.6.5 C/C++语句 …… 58

第 4 章 顺序结构及其应用程序设计 …… 59

4.1 3 种最基本的输入/输出方法 …… 59
4.1.1 变量赋值输入法 …… 59
4.1.2 C++基本输入/输出 …… 60
4.1.3 C 语言格式化输入/输出 …… 61
4.2 数学表达式应用程序设计 …… 66
4.3 综合应用题应用程序设计 …… 68

4.4 格式化输入/输出 …… 71
4.4.1 使用 C 语言标准输入/输出 …… 71
4.4.2 使用 C++流格式化输入/输出 …… 72

第 5 章 选择结构及其应用程序设计 …… 75

5.1 关系运算符和逻辑运算符 …… 75
5.1.1 关系运算符和关系表达式 …… 75
5.1.2 逻辑运算符与逻辑表达式 …… 76
5.1.3 条件运算符 …… 77
5.2 if 语句 …… 77
5.2.1 if 语句的 3 种形式 …… 78
5.2.2 嵌套 if 语句 …… 82
5.2.3 条件运算符（三目运算符） …… 85
5.3 switch 语句 …… 86
5.3.1 switch 语句的基本格式 …… 86
5.3.2 switch 语句应用程序设计 …… 87

第 6 章 循环结构及其应用程序设计 …… 90

6.1 for 语句 …… 90
6.1.1 for 语句的基本格式 …… 90
6.1.2 break 语句和 continue 语句 …… 92
6.1.3 循环嵌套 …… 94
6.2 while 语句与 do-while 语句 …… 95
6.2.1 while 语句与 do-while 语句的基本格式 …… 95
6.2.2 while 语句与 do-while 语句应用程序设计 …… 96
6.3 逻辑思维与计算机解题 …… 99
6.3.1 逻辑思维 …… 99
6.3.2 利用逻辑思维进行计算机解题 …… 100

第 7 章 数组及其应用程序设计 …… 107

7.1 一维数组及其应用程序设计 …… 107
7.1.1 一维数组的定义 …… 107
7.1.2 一维数组元素的引用 …… 108
7.1.3 一维数组的初始化 …… 108
7.1.4 一维数组的应用及其应用程序设计 …… 109
7.2 二维数组及其应用程序设计 …… 114

7.2.1 二维数组的定义 …… 114
7.2.2 二维数组的初始化 …… 115
7.2.3 二维数组的应用及其应用程序设计 …… 116
7.3 字符与字符串 …… 121
7.3.1 字符与字符的相互转换 …… 121
7.3.2 字符串的表示 …… 123
7.3.3 字符串的操作 …… 125

第8章 函数及其应用程序设计 …… 131

8.1 函数 …… 131
8.1.1 函数与函数类型 …… 131
8.1.2 函数中的形参与实参 …… 131
8.1.3 函数声明 …… 133
8.2 递归函数 …… 137
8.3 对递归函数的进一步理解 …… 141
8.3.1 队列与堆栈 …… 141
8.3.2 递归与堆栈 …… 142

第9章 结构体及其应用程序设计 …… 146

9.1 地址与指针 …… 146
9.1.1 与地址和指针相关的几个概念 …… 146
9.1.2 变量、地址与指针 …… 146
9.1.3 地址与指针操作实例 …… 147
9.2 结构体 …… 151
9.2.1 结构体的定义 …… 151
9.2.2 结构体信息静态输入 …… 151
9.2.3 结构体信息动态输入 …… 153
9.3 一个简单的学生管理系统设计 …… 155

第10章 链表及其应用程序设计 …… 162

10.1 链表 …… 162
10.1.1 链表的定义与分类 …… 162
10.1.2 链表的基本结构 …… 163
10.2 静态链表及其应用程序设计 …… 164
10.3 动态链表及其应用程序设计 …… 165
10.3.1 建立链表 …… 166

10.3.2 删除节点 …… 166
10.3.3 插入节点 …… 167
10.4 一个基于单向链表的通用管理系统设计 …… 170

第11章 文件及其应用程序设计 …… 176

11.1 文件概述 …… 176
11.1.1 文件与文件名 …… 176
11.1.2 文件的分类 …… 177
11.1.3 文件流与数据流 …… 177
11.1.4 文件的打开与关闭 …… 178
11.1.5 C/C++中带缓冲区的文件处理 …… 178
11.2 文件的3种处理形式 …… 179
11.2.1 FILE 指针 …… 179
11.2.2 重定向 …… 181
11.2.3 输入/输出流 …… 181
11.3 文件应用程序设计 …… 182

第12章 编译预处理与源程序在线测评系统 …… 187

12.1 编译预处理 …… 187
12.1.1 宏定义 …… 187
12.1.2 文件包含 …… 188
12.1.3 条件编译 …… 191
12.2 源程序在线测评系统 …… 193
12.2.1 源程序在线测评系统简介 …… 193
12.2.2 国内外主要的在线测评系统网站 …… 193
12.2.3 RealOJ 源程序在线测评系统实例 …… 194

附录一 ASCII 码一览表 …… 196

附录二 C/C++部分关键字用途及其中文释义 …… 197

附录三 ANSI C 标准关键字及其中文释义 …… 200

参考文献 …… 201

第1章　计算机程序设计概述

1.1　计算机系统组成

计算机(Computer)也称为电脑，是一种用于高速计算的电子仪器，既可以进行数值计算，又可以进行逻辑计算，还具有存储记忆功能，是能够按照程序运行，自动、高速处理海量数据的现代化智能电子设备。计算机由硬件系统和软件系统组成。没有安装任何软件的计算机称为裸机。计算机可分为超级计算机、工业控制计算机、网络计算机、个人计算机、嵌入式计算机等。较为先进的计算机有生物计算机、光子计算机、量子计算机等。

微型计算机简称“微型机”“微机”等，由于其具备人脑的某些功能，所以也称其为“微电脑”，是由大规模集成电路组成的、体积较小的电子计算机。典型的微型计算机包括控制器、运算器、存储器、输入设备、输出设备5个组成部分。如果把控制器与运算器封装在一小块芯片上，则称该芯片为微处理器(Micro Processing Unit，MPU)或中央处理器(Central Processing Unit，CPU)。如果将它与由大规模集成电路制成的存储器、输入/输出接口电路在印制电路板上用总线连接起来，那么就构成了微型计算机。其特点是体积小、灵活性好、价格便宜、使用方便。

计算机是20世纪最先进的科学技术发明，对人类的生产活动和社会活动产生了极其重要的影响，并以强大的生命力飞速发展。它的应用领域从最初的军事科研扩展到社会的各个领域，已形成了规模巨大的计算机产业，带动了全球范围的技术进步，由此引发了深刻的社会变革。目前，计算机已遍及学校、企事业单位，成为信息社会中必不可少的工具。

1.1.1　计算机硬件系统

当前计算机体系结构是由计算机的开拓者——数学家约翰·冯·诺依曼最先提出的，因此称为冯·诺依曼计算机体系结构。这种结构的硬件系统由5个部分组成，如图1.1所示，各部分的功能如下。

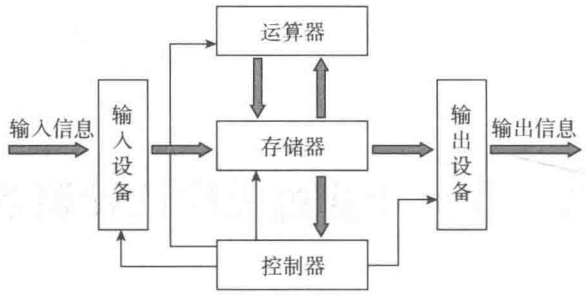

图 1.1　微型计算机的硬件系统

(1)控制器。控制器是计算机的控制中心，从存储器中读取指令并分析指令，根据指令要求完成相应操作。控制器产生一系列的控制指令，使计算机硬件系统各部分协调工作，完成程序和数据的输入、运算，并输出结果。

(2)运算器。运算器是中央处理器的执行单元，是所有中央处理器的核心组成部分。运算器在控制器的控制下接收运算数据，完成指令指定的二进制算术运算。

(3)存储器。存储器是可以被中央处理器直接访问而无须通过输入/输出设备的记忆设备。存储器用来保存程序和数据，以及存储运算时的数据和结果。

(4)输入设备。输入设备是用来完成输入功能的部件。通过输入设备可以向计算机输入程序、数据及各种信息。常用的输入设备有键盘、鼠标、扫描仪、磁盘驱动器和触摸屏等。

(5)输出设备。输出设备是用来将计算机的中间运行情况或运行后的结果进行表现的部件。常用的输出设备有显示器、打印机、绘图仪、磁盘驱动器和音响等。

根据冯·诺依曼计算机体系结构，计算机自动执行程序，即执行指令，可分为如下 4 个阶段。

(1)取指阶段：从存储器某地址处取出要执行的指令，送到中央处理器内部的指令寄存器中暂存。

(2)译码阶段：对保存在指令寄存器中的指令进行分析，翻译出该指令进行的操作。

(3)执行阶段：根据译码结果向各个部件发出相应控制信号，完成指令规定的操作。

(4)取下一条指令：为执行下一条指令做好准备，即产生下一条指令地址。

1.1.2　计算机软件系统

计算机软件系统是指挥计算机工作的程序和程序运行时所需要的数据，以及所需要的数据和文档的集合。计算机软件系统分为系统软件和应用软件两个部分。

(1)系统软件。系统软件是指计算机系统必备的基本软件，是管理、监控和维护计算机硬件和软件资源而开发应用的软件。其按功能可分为 5 类：操作系统(如 Windows、Linux 等)、语言处理程序(如汇编程序、编译程序等)、程序设计语言(如 C、C++、Java、Python 等)、系统支持和服务程序(如系统诊断程序、查杀病毒程序等)、数据库管理系统(如 Oracle、SQL Server 等)。

(2)应用软件。应用软件是由系统软件开发的，为解决计算机各类应用问题而编写的程序，具有较强的实用性。其按功能可分为两类：用户程序(如天气预报软件等)和应用软件包(如 Microsoft Office 套件等)。

1.2 程序设计语言

人与人之间的交流需要通过语言，如汉语、英语、俄语等。人与计算机之间通信，也需要语言，这种语言就是计算机语言(Computer Language)。计算机语言是人与计算机之间传递信息的媒介。

计算机语言也称为程序设计语言(Programming Language)，是一组用来定义计算机程序的语法规则。它是一种被标准化的交流技巧，用来向计算机发出指令。一种计算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地定义在不同情况下所应当采取的行动。

根据计算机程序设计语言的发展，计算机语言基本经历了3个阶段：很难理解的机器语言、较难理解的汇编语言、脱离机器的高级语言。其中高级语言的发展又经历了3个阶段：面向过程的设计语言、面向对象的设计语言、智能化语言。

1.2.1 机器语言

计算机的工作基于二进制，从根本上说，计算机只能识别由0和1组成的指令，即机器指令。

机器语言是用二进制代码表示的、计算机能直接识别和执行的一种机器指令的集合，是计算机唯一能够识别的程序设计语言。机器语言直接对计算机硬件产生作用，它是计算机的设计者通过计算机的硬件结构赋予计算机的操作功能。因此，不同类型的计算机采用的机器语言有可能是不同的。也就是说，不同的CPU具有不同的指令系统。

机器语言具有灵活、直接执行和速度快等特点。但是，机器语言也存在着程序难编写、难修改、难维护，需要用户直接对存储空间进行分配，没有通用性，编程效率极低等缺点，很难被人们掌握和推广。因此，通常只有极少数计算机专家或专业人员才能使用机器语言。

【例1.1】下面是某CPU利用机器语言计算 $a=a+b$ 的代码，其中 $a=1$，$b=2$。

```
#01:11100011 10100000 00000000 00000001
#02:11100011 10100000 00010000 00000010
#03:11100000 10000000 00000000 00000001
```

【代码解释】

```
#01:令 a=1。
#02:令 b=2。
#03:将 a 和 b 的值相加,并将结果存放在 a 中。
```

1.2.2 汇编语言(符号语言)

为了克服机器语言的上述缺点，人们创造出了符号语言，即用一些英文字母和数字表示一个指令，如 ADD 表示加、SUB 表示减等。例如：ADD A，B（执行 A+B→A）。

汇编语言(Assembly Language)是面向机器的程序设计语言，用指令助记符代替机器指令的操作码，用地址符号(Symbol)或标号(Label)代替指令或操作数的地址。像这样符号化的程序设计语言就是汇编语言，也称为符号语言。

使用汇编语言编写的程序，机器不能直接识别，还要由汇编程序或者汇编语言编译器转换成机器指令。汇编程序将符号化的操作代码组装成机器可以识别的机器指令，这个组装的过程称为组合或者汇编。因此，有时候人们也把汇编语言称为组合语言。

【例 1.2】下面是某 CPU 利用汇编语言计算 $a=a+b$ 的代码，其中 $a=1$，$b=2$。

```
#01:MOV R0,#1
#02:MOV R1,#2
#03:ADD R0,R0,R1
```

【代码解释】

```
#01:将数值 1 放入寄存器 R0。
#02:将数值 2 放入寄存器 R1。
#03:将寄存器 R0 的值和寄存器 R1 的值相加,并将结果存放在寄存器 R0 中。
```

汇编语言指令是机器指令的符号化，与机器指令存在着直接的对应关系，所以汇编语言同样存在着难学难用、容易出错、维护困难等缺点。但是汇编语言也有自己的优点，它是一种与硬件紧密相关的程序设计低级语言，其程序结构简单，执行速度快，程序易优化，编译后占用的存储空间小，是单片机应用系统开发中最常用的程序设计语言。

在实际应用中，它通常被应用在底层硬件操作和高要求的程序优化的场合，或在高级语言不能满足设计要求，且不具备支持某种特定功能的技术性能(如特殊的输入、输出)时才被使用。驱动程序、嵌入式操作系统和实时运行程序都需要汇编语言。

1.2.3 高级语言

1. 高级语言简介

计算机的发展促使人们去寻求一些与人类自然语言相接近，且能为计算机所接受的语义确定、规则明确、自然直观和通用易学的计算机语言。这种与自然语言接近、与具体的计算机指令系统无关、其表达方式更接近人们对求解问题的描述方式的计算机语言，被称为高级语言。20 世纪 50 年代，人们创造出了第一个计算机高级语言——FORTRAN 语言。目前，人们广泛使用的高级语言有 C、C++、C#、Java、Python 等。

高级语言的描述形式接近自然语言，采用类似自然语言的形式来描述问题的处理过程，用数学表达式的形式来描述对数据的计算过程。高级语言面向的是解题的算法而不是具体机

器的指令系统，故又称为算法语言。用它写出的程序对任何型号的计算机都适用(或只需进行很少的修改)。高级语言是面向用户的、基本上独立于计算机种类和结构的语言，其最大的优点是形式上接近于算术语言和自然语言，概念上接近于人们通常使用的语言。高级语言的一个命令可以代替几条、几十条甚至几百条汇编语言的指令。因此，高级语言易学易用、通用性强、应用广泛。

【例1.3】下面是利用C语言计算 $a=a+b$ 的代码，其中 $a=1$，$b=2$。

```
#01:int a=1;
#02:int b=2;
#03:a=a+b;
```

【代码解释】

```
#01:使变量 a 等于 1。
#02:使变量 b 等于 2。
#03:将变量 a 和变量 b 相加,并将结果赋值给变量 a。
```

2. 高级语言的编译与解释

使用高级语言编写的程序称为源程序。计算机并不能直接识别用高级语言编写的源程序，需要通过编译程序把高级语言“翻译”成机器语言形式的目标程序，再与有关的库程序链接成可执行程序，这样计算机才能识别和执行。这种翻译通常有两种方式：编译方式和解释方式。

编译程序的功能就是把使用高级语言书写的源程序翻译成与之等价的目标程序(汇编语言或机器语言)。编译过程包括词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成等阶段，以及符号表管理和出错处理模块。解释过程在词法分析、语法分析和语义分析方面与编译程序的工作原理基本相同，但是在运行用户程序时，它直接执行源程序或源程序的内部形式。

这两种语言处理后式的根本区别是，在编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程；而在解释方式下，解释程序和源程序(或其某种等价表示)要参与到程序的运行过程中，运行程序的控制权在解释程序上。解释器翻译源程序时不产生独立的目标程序，而编译器则需将源程序翻译成独立的目标程序。

3. 高级语言的发展

自从第一个计算机高级语言——FORTRAN语言产生之后，计算机高级语言的发展又经历了3个阶段：面向过程的程序设计语言、面向对象的程序设计语言、智能化语言。

1)面向过程的程序设计语言

面向过程的程序设计语言主要采用数组、过程(函数或模块)、指针等解决问题。数十年来，全世界涌现出了几千种面向过程的程序设计语言，每种语言都有其特定的用途，其中应用广泛的有100多种，而影响较大的有：FORTRON(适合数值计算的语言)、BASIC(适合初学者的小型会话语言)、COBOL(适合商业管理的语言)、Pascal(适合教学的结构化程序设

计语言)、C(系统描述语言，适用于解决小型程序编程，尤其是设备驱动程序和内嵌式应用程序)。

结构化程序设计(Structured Programming)是面向过程程序设计的一个子集，它对写入的程序使用逻辑结构，使理解和修改更有效、更容易。程序处理一般包括输入、处理、输出3个步骤。输入包括变量赋值、输入语句；处理包括算术运算、逻辑运算、算法处理等；输出包括打印输出、写入文件和数据库等。

结构化程序设计方法的基本思路是把一个复杂问题的求解过程分阶段进行，每个阶段处理的问题都控制在人们容易理解和处理的范围内，采用的方法如下。

(1)自顶向下。程序设计时，应先考虑整体，后考虑细节；先考虑全局目标，后考虑局部目标。不要一开始就过多追求细节，先从上层总目标开始，逐步使问题具体化。

(2)逐步细化。一个复杂的问题通常采用分而治之的思想解决，即把大任务分解为多个小任务，先解决每个小的、容易的子任务，再解决复杂的、较大的任务。

(3)模块化设计。模块化是把要解决问题的总目标分解为各个子目标，再进一步分解为具体的小目标，把每一个小目标称为一个模块。模块在计算机程序设计中往往又被称为函数或过程，函数用于完成相同的功能，在程序的不同地方通过函数名称调用执行，而不必重复书写语句。

(4)结构化编码。结构化编码的3种基本结构为顺序结构、选择结构和循环结构。

20世纪70年代以来，结构化程序设计和软件工程的思想日益为人们所接受和欣赏。在它们的影响下，先后出现了一些很有影响力的结构化语言，这些结构化语言直接支持结构化的控制结构，具有很强的过程结构和数据结构能力。其中，C语言功能丰富，表达能力强，有丰富的运算符和数据类型，使用灵活方便，应用面广，移植能力强，编译质量高，目标程序效率高，具有高级语言的优点。同时，C语言还具有低级语言的许多特点，如允许直接访问物理地址，能进行位操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作等。用C语言编译程序产生的目标程序，其质量可以与汇编语言产生的目标程序相媲美，具有“可移植的汇编语言”之称，成为编写应用软件、操作系统和编译程序的重要语言之一。

同时，C语言由于其独特的优势，也是最“经久不衰”的程序设计语言。许多操作系统，如UNIX、Windows、Linux等，都是用C语言或以C语言为基础进行编写的。迄今为止，许多在当时应用很广泛的程序设计语言，如BASIC、Pascal等，已经退出历史舞台，而C语言在目前程序设计语言的应用排名中，一直名列前茅。“学会其他语言可一时受益，学会C语言会终身受益!”已成为业界许多开发人员的共识。

2)面向对象的程序设计语言

面向对象的程序设计语言，又被称为非过程化语言。相对于面向过程的程序设计语言，面向对象的程序设计语言具有非过程性、采用图形窗口和人机对话形式、基于数据库和面向对象技术等特点，易编程、易理解、易使用、易维护。

面向对象的程序设计语言引入了对象和类的概念，对象具有属性和行为，通过消息来实现对象之间的相互操作，具有封装性、继承性和多态性三大特性。

面向对象的程序设计语言有VB(支持面向对象的程序设计语言)、C++(支持面向对象的程序设计大型语言)、Java(适用于网络的语言)、Python(一种代表简单主义思想的语言，

阅读一个良好的 Python 程序如同在读英语一样，它使你能够专注于解决问题而不是去搞明白语言本身)等。

C++是一种使用非常广泛的计算机编程语言，它支持过程化程序设计、数据抽象、面向对象程序设计、泛型程序设计等多种程序设计风格。C++引入了面向对象的概念，使开发人机交互类型的应用程序更为简单、快捷。

起初，C++是作为 C 语言的增强版出现的，从给 C 语言增加类开始，不断地增加新特性。C++由于其语言本身过度复杂，人类难于理解其语义。更为糟糕的是，C++的编译系统受到 C++的复杂性的影响，非常难编写，即使能够使用的编译器也存在大量的问题，这些问题大多难于被发现。

因此，如何快速、高效地学习 C++程序设计是一个非常值得探讨的问题。为此，我们进行了深入的探索、研究和实践，独创基于算法设计的能力驱动教学方法，也称基于信息学竞赛的能力驱动教学方法，取得了事半功倍的效果，为初学者和信息学竞赛设计者带来了很大的方便。

3)智能化语言

智能化语言，又被称为自然语言、知识库语言或人工智能语言，其目标是成为最接近日常生活所用语言的程序语言，主要应用于人工智能领域，用于编写推理、演绎程序。

真正意义上的智能化语言尚未出现，LISP 和 PROLOG 号称智能化语言，其实还远远不能达到自然语言的要求。

1.3 计算机算法

1.3.1 算法的基本概念

程序设计离不开算法设计，所以计算机程序设计人员必须掌握设计算法，并根据算法编写程序。计算机科学的各个领域都高度依赖于算法设计。设计高效的算法来解决现实应用问题，是计算机各领域的重要研究课题。

1. 算法的定义

什么是算法？广义地说，为了解决某一问题而采取的方法和步骤，就称之为算法。乐谱是乐队演奏和指挥的算法；菜谱是厨师烧菜的算法。在计算机中，算法通常是指可以用计算机来解决的某一类问题的程序或步骤，这些程序或步骤必须是明确的和有效的，而且能够在有限步之内完成。

一个算法就是一个有穷规则的集合，其中的规则规定了一个解决某一特定类型问题的运算序列。用计算机解题时，任何答案的获得都是按指定顺序执行一系列指令的结果。因此，用计算机解题时，需要将解题方法转换成一系列具体的、在计算机上可执行的步骤，然后才能将这些步骤表示成指令代码。这些步骤能清楚地反映解题方法每步“怎样做”的过程，这个过程就是通常所说的算法。

2. 算法的基本特征

一个算法应该具有如下 5 个特征。

(1)有穷性。一个算法必须保证它的执行步骤是有限的，即它是能终止的。也就是说，执行步骤不能是无限的。

(2)确定性。算法中的每个步骤必须有确切的含义，而不应当是含糊的、模棱两可的。

(3)能(可)行性。算法中的每一个步骤都要足够简单，是实际能做的，并能在有限时间内完成。

(4)输入性。算法有 0 个或多个输入。输入是指算法在执行时需要从外界获得数据，其目的是为算法建立某些初始状态。

(5)输出性。算法有一个或多个输出。算法的目的是求解问题，问题求解的结果应以一定的方式输出。

3. 算法的判断标准

在现实社会中，不同的人对于同一问题会有不同的看法或解决方法。同样，在计算机领域，对于同一问题可能存在多种算法。

判断一个算法的好坏，主要依据以下 4 个标准。

(1)正确性。正确性是设计一个算法的首要条件，如果一个算法不正确，那么其他方面也就无从谈起。一个正确的算法是指在合理的数据输入下，能在有限的时间内得出正确的结果。

(2)可读性。算法首先是为了人的阅读与交流，其次才是让计算机执行，因此算法应该易于理解；相反，晦涩难读的算法易于隐藏较多错误，而使实现该算法的程序的调试工作变得更加困难。

(3)健壮性。算法应当具备检查错误和对错误进行适当处理的能力。一般而言，处理错误的方法不应是中断程序的执行，而应是返回一个表示错误或错误性质的值，以便在更高的抽象层次上处理。

(4)效率。效率是指算法执行时所需计算机资源的多少，包括运行时间和存储空间两方面的要求。运行时间和存储空间都与问题的规模有关。存储空间指的是算法执行过程中所需的最大存储空间。

4. 算法的复杂性分析(评估算法的效率)

在设计满足问题要求的算法时，评估算法的效率，即算法复杂度的估算是非常重要的。我们不可能把每个能想到的算法都一一实现看其是否足够快，应当通过估算算法的复杂度来判断所想的算法是否足够高效。

算法复杂度分为时间复杂度和空间复杂度。时间复杂度是指执行算法所需要的计算工作量；而空间复杂度是指执行这个算法所需要的内存空间。

由于目前计算机的存储容量足够大，因此在估算算法的复杂度时，一般不考虑空间复杂度，只考虑时间复杂度。

在分析时间复杂度时，我们通常考虑它与什么成正比，并称之为算法的阶。在计算机科学中，算法的时间复杂度是一个函数，通常用大写的 O 表述，用于定量描述该算法的运行时间。算法中模块 n 的基本操作的重复执行次数计为函数 $f(n)$，算法的时间复杂度为 $T(n)=$