高职高专电子信息类精品课程规划教材

程序设计基础（C语言）

主　编　杨俊清
副主编　石　锋

西安电子科技大学出版社
http://www.xduph.com

精品课程

高职高专电子信息类精品课程规划教材

程序设计基础(C 语言)

主　编　杨俊清

副主编　石　锋

参　编　程传旭　陈庆荣　张少应

彭　寒　王　建　曹国震

西安电子科技大学出版社

2009

内 容 简 介

本书是为了适应高职高专教学改革的需要而编写的，全书主要内容包括 C 程序的结构，算法及算法的描述，C 语言基础，流程控制语句，函数，数组，结构体、共用体和枚举类型，文件等，较全面地反映了 C 语言的全貌。

本书在内容的组织上打破了同类教材的传统结构，突出了重点，分散了难点。主要表现在：一是将编译预处理和位运算并入 C 语言基础，降低了教学要求；二是将指针的相关知识以应用为原则，分散到相关章节中，便于学生学习和掌握。同时，本书内容完全覆盖了“全国计算机等级考试（二级 C）考试大纲”中 C 语言部分的知识点，并且在习题中选入部分历年考试的原题，为学习者参加全国计算机等级考试提供帮助。

本书可以作为高职高专类学校计算机专业的教材，也适合有关人员自学使用。

★ 本书配有电子教案，有需要者可登录出版社网站，免费下载。

图书在版编目(CIP)数据

程序设计基础：C 语言 / 杨俊清主编. —西安：西安电子科技大学出版社，2009.8

高职高专电子信息类精品课程规划教材

ISBN 978-7-5606-2318-4

Ⅰ. 程… Ⅱ. 杨… Ⅲ. C 语言—程序设计—高等学校：技术学校—教材

Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2009)第 122218 号

策　　划　杨　璠

责任编辑　杨　璠

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　陕西光大印务有限责任公司

版　　次　2009 年 8 月第 1 版　2009 年 8 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印　张　13

字　　数　303 千字

印　　数　1～4 000 册

定　　价　23.00 元

ISBN 978 - 7 - 5606 - 2318 - 4/TP • 1175

XDUP 2610001-1

*** * * 如有印装问题可调换 * * ***

前　言

本书是按照高等学校基础教育改革的要求，由西安航空技术高等专科学校计算机工程系组织编写的。

程序设计技术是信息技术人员的基本功。随着计算机技术的进步和高校教学改革的不断深入，C 语言由原来计算机专业高年级学生学习的专业课程，逐渐变成了低年级学生学习程序设计的入门课程、基础课程。教学目的和教学对象的变化，要求必须对这门课程的教学内容和教学方法进行改革。我系近年来在这方面进行了有益的尝试，对改革的成果进行了总结和提炼，编写了这本教材。

本书以《程序设计基础》命名，是为了突出这门课程作为入门性课程的特点。在这门课程中，核心是程序设计的基本思想和方法。但思想和方法又不能凭空存在，必须以某种方式表达出来，C 语言是表达这种思想和方法的一种非常优秀的工具。本书以 C 语言的语法为线索，逐渐地融入程序设计的基础知识，使学生在学习 C 语言的过程中，潜移默化地接受、理解和掌握程序设计的基本思想和方法。当然，作为入门性的课程，也不能期望在这门课程中解决所有的问题。因此，在本书中，我们只介绍最基本的、学生能够理解和掌握的程序设计思想和方法，而程序设计中一些经典的思想和方法，留到其他课程中去解决。

本书还有一个特点，就是把 C 语言中应用最灵活、学生学习时感觉难度最大的指针进行了分解。在书中，我们把指针的相关知识，按照它的作用分解到各个章节中。这样不但分散了难点，而且使学生知道什么时候必须使用指针，该如何使用指针。解决了指针这个难点，C 语言的教学就成功了一半。

本书由杨俊清教授任主编，负责全书的组织和统稿工作，并编写了第 1 章。石锋担任副主编，并编写了第 7 章。参加编写的人员还有：程传旭，编写了第 2、3 章；陈庆荣，编写了第 4 章；张少应，编写了第 5 章；彭寒，编写了第 6 章；王建，编写了第 8、9 章；曹国震，编写了第 10 章。

由于作者水平有限，书中疏漏之处在所难免，恳请读者批评指正。

编　者

2009 年 6 月

目　录

第 1 章　程序设计概述

知识要点

程序工作原理
程序设计语言的发展
算法的实现
C 语言程序的基本结构
C 语言的发展
Visual C++ 6.0 环境运行的 C 程序实例

能力要求

掌握程序工作原理
掌握 C 语言程序的基本结构
掌握在 Visual C++ 6.0 环境中运行的 C 程序调试

1.1　程序工作原理

1.1.1　冯・诺伊曼原理

一台计算机由硬件系统和软件系统两大部分组成，硬件是计算机的物质基础，而软件是计算机的灵魂。没有软件，计算机只是一台“裸机”，有了软件，计算机才能成为一台真正的“电脑”。而所有的软件，都是用计算机程序设计语言编写的。程序是指计算机可以直接或间接执行的指令的集合。计算机系统通过运行程序来实现各种不同的应用。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。一种程序设计语言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。

当今的计算机模型是由数学家冯・诺依曼(von Neumann)提出来的，我们称其为冯・诺依曼模型(von Neumann Model)或冯・诺依曼机(von Neumann Machine)。直到今天，几乎所有的计算机都是沿用这一模型设计的。冯・诺依曼机概念基于一个存储器(用来存储指令和数据)、一个控制器(负责从存储器中逐条取出指令)和一个处理器(通过算术或逻辑操作来处理数据)，最后的结果必须送回存储器中。可以把这些特点归结为以下几条：

(1) 数据或指令以二进制形式存储(数据和指令在外形上没有什么区别，但每一位二进制数字有不同的含义)。

(2) “存储程序”的工作方式(事先要编好程序，执行之前先将程序存放到存储器某个可知的地方)。

(3) 程序顺序执行(可强行改变执行顺序)。

(4) 存储器的内容可以被修改(存储器的某个单元一旦进入新的数据，则该单元原来的数据立即消失，被新数据代替)。

冯·诺依曼体系结构的作用体现在命令式语言的下述三大特性上：

(1) 变量　存储器由大量的存储单元(Memory Cell，Memory Location)组成，数据就存放在这些单元中。汇编语言通过对存储单元的命名来访问数据。在命令式语言中，存储单元及它的名称由变量(Variable)的概念代替。变量代表一个(或一组)命名的存储单元，单元可存放值，值可以被修改。

(2) 赋值　使用存储概念的另一个后果是每一计算结果必须存储，即赋值于某个存储单元，从而改变该单元的值。

(3) 重复　语句按顺序执行，指令存储在有限的存储器中，要完成任何复杂的计算，唯一的办法是重复执行某些指令序列。

冯·诺依曼型计算机的两大特征是“程序存储”和“采用二进制”。具体地说，在上述计算机中，要实现机器的自动计算，必须先根据题目的要求，编制出求解该问题的计算程序，并通过输入设备将该程序存入计算机的存储器中，称之为“程序存储”。在计算机中，计算程序及数据是用二进制代码表示的，计算机只能存储并识别二进制代码表示的计算程序和数据，称之为“采用二进制”。

冯·诺伊曼思想实际上是电子计算机设计的基本思想，奠定了现代电子计算机的基本结构，开创了程序设计的时代。

1.1.2　计算机的构成原理

电子计算机硬件通常由 5 大功能部件组成：存储器、运算器、控制器、输入设备和输出设备，5 大部件通过总线有机地连接在一起组成计算机的硬件系统。

运算器用来实现算术、逻辑等各种运算。

存储器用来存放计算程序及参与运算的各种数据。存储器可以分为内存储器(内存)和外存储器(外存)。

控制器实现对整个运算过程的有规律的控制。

输入设备实现计算程序和原始数据的输入。

输出设备实现计算结果的输出。

随着制造计算机的器件的发展，特别是微机的发展，产生了中央处理器(CPU)。CPU 中包括运算器和控制器。

计算机工作时要预先把指挥计算机如何进行操作的指令序列(通常称为程序)和原始数据通过输入设备输入到计算机的内部存储器中。每一条指令中明确规定了计算机从哪个地址取数，进行什么操作，然后送到什么地址等步骤。

计算机在运行时，先从内存中取出第一条指令，通过控制器的译码，按指令的要求，从存储器中取出数据进行指定的运算和逻辑操作，然后再按地址把结果送到内存中去。接下来，再取出第二条指令，在控制器的指挥下完成规定的操作。依此进行下去，直至遇到停止指令。简而言之，即将程序与数据一样存储，按程序编排的顺序，一步一步地取出指令，自动地完成指令规定的操作。

我们把按照冯·诺依曼原理构造的计算机叫冯·诺依曼计算机，其体系结构称为冯·诺依曼结构。目前计算机已发展到了第五代，基本上仍然遵循着冯·诺依曼原理和结构。

1.2　程序设计语言

1.2.1　程序设计语言的发展

语言是人们交流思想的工具。人类在长期的历史发展过程中，为了交流思想、表达感情和交换信息，逐步形成了语言。这类语言，如汉语和英语，通常称为自然语言。另一方面，人们为了某种专门用途，创造出种种不同的语言，这类语言通常称为人工语言。专门用于编制计算机程序的各种人工语言称为程序设计语言(Programing Language)。

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。低级语言包括机器语言和汇编语言。低级语言与特定的机器有关，其功效高，但使用复杂、繁琐、费时、易出差错。机器语言是表示成数码形式的机器基本指令集。汇编语言是机器语言部分符号化的结果。高级语言的表示方法要比低级语言更接近于待解问题的表示方法，其特点是在一定程度上与具体机器无关，易学、易用、易维护。

计算机的每一个动作、每一个步骤都是按照已经编好的程序来执行的，而程序需要用人们能掌握的语言来编写，于是出现了程序设计语言。计算机程序设计语言的发展经历了从机器语言、汇编语言到高级语言的历程。

1．机器语言

电子计算机所使用的是由“0”和“1”组成的二进制数，二进制是计算机语言的基础。计算机发明之初，人们只能用计算机的语言去命令计算机工作，也就是写出一串串由“0”和“1”组成的指令序列交由计算机执行，这种语言就是机器语言。

使用机器语言编写程序是一件十分繁琐的工作，特别是在程序有错需要修改时更加困难，而且编出的程序不便于记忆、阅读和书写，还容易出错。由于每台计算机的指令系统往往各不相同，因此在一台计算机上执行的程序要想在另一台计算机上执行，必须另编程序，可移植性较差，造成了重复工作。但由于使用的是针对特定型号计算机的语言，故而机器语言运算效率是所有语言中最高的。机器语言是第一代计算机语言。

2．汇编语言

为了克服机器语言难读、难编、难记和易出错的缺点，人们用与代码指令实际含义相近的英文缩写词、字母和数字等符号取代指令代码，例如用 ADD 代表加法，用 MOV 代表数据传递等。这样，人们能较容易读懂并理解程序，使得纠错及维护变得方便了。这种程

序设计语言称为汇编语言，即第二代计算机语言。然而计算机是不认识这些符号的，这就需要一个专门的程序负责将这些符号翻译成二进制数的机器语言，这种翻译程序称为汇编程序。

汇编语言仍然是面向机器的语言，使用起来还是比较繁琐，通用性也差。汇编语言是低级语言。但是，用汇编语言编写的程序，其目标程序占用内存空间少，运行速度快，有着高级语言不可替代的用途。

3. 高级语言

不论是机器语言还是汇编语言，都是面向硬件具体操作的，对机器过分依赖，要求使用者必须对硬件结构及其工作原理都十分熟悉，这对非计算机专业人员来说是难以达到的，对计算机的推广应用不利。计算机事业的发展促使人们寻求一些与人类自然语言相接近且能为计算机所接受的通用易学的计算机语言。这种与自然语言相近并被计算机接受和执行的计算机语言称为高级语言。高级语言是面向用户的语言。无论何种机型的计算机，只要配备上相应高级语言的编译或解释程序，则用该高级语言编写的程序就可以运行。

1954 年，第一个完全脱离机器硬件的高级语言 FORTRAN 问世了，自此之后，共有几百种高级语言出现，其中有重要意义的有几十种，影响较大、使用较普遍的有 FORTRAN、ALGOL、COBOL、BASIC、LISP、Pascal、C、PROLOG、Ada、C++、VC、VB、Delphi、Java 等。

高级语言的出现使得计算机程序设计语言不再过度地依赖某种特定的机器或环境。这是因为高级语言在不同的平台上会被编译成不同的机器语言，而不是直接被机器执行的。计算机并不能直接地接受和执行用高级语言编写的源程序。高级语言源程序在输入计算机时，通过“翻译程序”翻译成机器语言形式的目标程序，计算机才能识别和执行。这种“翻译”通常有两种方式，即编译方式和解释方式。

编译方式是指在源程序执行之前，就将程序源代码“翻译”成目标代码(机器语言)，因此目标程序可以脱离其语言环境独立执行，使用比较方便、效率较高。但应用程序一旦需要修改，必须先修改源代码，再重新编译生成新的目标文件(*.obj)才能执行。

解释方式是应用程序源代码一边由相应语言的解释器“翻译”成目标代码(机器语言)，一边执行，因此效率比较低，而且不能生成可独立执行的可执行文件，应用程序不能脱离其解释器。但这种方式比较灵活，可以动态地调整、修改应用程序。

高级语言的发展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。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各自为政，缺乏科学规范的系统规划与测试、评估标准。其结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，甚至带来巨大损失，软件给人的感觉是越来越不可靠，以致几乎没有不出错的软件。这一切，极大地震动了计算机界，历史上称之为“软件危机”。人们认识到：大型程序的编制不同于小程序，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。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 出现，标志着结构化程序设计时期的开始。

4. 面向对象程序设计语言

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式的，在一个模块被执行完成前，计算机不能干别的事，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，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，也就是面向对象(Object)。

面向对象程序设计(Object Oriented Programming)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)及事物之间的相互关系。面向对象技术强调的基本原则是直接面对客观事物本身进行抽象并在此基础上进行软件开发，将人类的思维方式与表达方式直接应用在软件设计中。

1.2.2　典型高级语言

目前有各种高级程序设计语言，其中以下几种应用非常广泛。

FORTRAN，全称为 Formula Translator，意即公式翻译。它是一种适用于科学计算的高级程序设计语言。

COBOL，全称为 Common Business Oriented Language，意即通用商业语言。它是适用于数据处理的高级程序设计语言。

BASIC，全称为 Beginner's All-purpose Symbolic Instruction Code，意即初学者通用符号指令代码。这是一种简单易学，具有会话功能的，适用于科学计算、数据处理和实时处理的程序设计语言。1964 年由美国达尔摩斯学院的基米尼和科茨完成设计并提出了 BASIC 语言的第一个版本，后经过不断丰富和发展，从基本的 BASIC 发展到 GWBASIC、Quick BASIC、True BASIC、Turbo BASIC、Visual Basic 等。Visual Basic 是一个基于 Windows 操作系统的面向对象的可视化集成开发环境和程序设计语言，它既有传统 BASIC 易学、易懂、易记、易用的特点，又有面向对象、可视化设计、事件驱动、动态数据交换等特点。

Pascal 是一种结构化程序设计语言，由瑞士苏黎世联邦工业大学的沃斯教授研制，于 1971 年正式发表。它是以 ALGOL 语言为基础，按照结构化程序设计原则设计出来的，它的优点是小巧、简洁、结构清晰、表达能力强、实现效能高。Pascal 既重视数据结构，又重视程序的结构，具有大量的控制结构，充分反映了结构化程序设计的思想和要求，直观易懂，使用灵活，既可用于科学计算，又能用来编写系统软件。它适用于科学计算、数据处理和描述系统软件。

C 语言是由美国贝尔实验室提出的。1973 年首先用于编写 UNIX 操作系统。C 程序易读，程序效率很高，适于描述操作系统、编译程序和各种软件工具，已得到广泛的应用。C 语言的主要特色是兼顾了高级语言和汇编语言的特点，简洁、丰富、可移植。C 语言提供了结构式编程所需要的各种现代化的控制结构。C 语言是一种通用编程语言，使用 C 语言编写程序，既能感觉到使用高级语言的自然，也能体会到利用计算机硬件指令的直接。

LISP 是一种表处理语言，是 20 世纪 50 年代末 60 年代初为解决人工智能问题而发展起来的一种高级程序设计语言。LISP 的理论基础是数理逻辑，它很简单又有很强的表达能力。

Java 是 Sun 公司推出的一种编程语言。它是一种通过解释方式来执行的语言，语法规则和 C++ 类似。同时，Java 也是一种跨平台的程序设计语言。Java 非常适合于企业网络和 Internet 环境，现在已成为 Internet 中最受欢迎、最有影响的编程语言之一。Java 有许多优

点：简单、面向对象、分布式、解释性、可靠、安全、结构中立性、可移植性、高性能、多线程、动态性等。

1.2.3 程序设计语言发展趋势

程序设计语言是软件的重要方面。它的发展趋势是可视化、智能化和构件化。

1．可视化程序设计技术

随着 Windows 操作系统的广泛推广与应用，它的可视化图形界面与所见即所得的视觉效果越来越成为编程语言效仿的典范。典型的可视化程序设计语言集成环境有 Microsoft 公司提供的 Visual Studio 系列等。Windows 系统本身就提供了相应的接口功能与系统调用供程序开发者使用，所以可视化程序设计技术成为程序设计语言发展的一个趋势。

2．智能化程序设计技术

程序设计技术的智能化主要体现在第 4 代程序设计语言中，它改变了传统的完全手工的编程工作方式，而将编程变成了提问式与填空式的工作方式。在系统给出部分甚至大部分代码以后，由程序员填入适当的其他内容而完成整个编程工作，这样的工作方式不仅提高了编程工作效率，而且避免了很多在传统手工方式中存在或容易出现的错误。

3．构件化程序设计技术

以面向对象程序设计为基础，在可视化程序设计语言集成环境的支持下，还出现了构件化程序设计的趋势。程序设计过程中以标准的构件为基本单位来构造一个完整的程序，这种工作类似于房屋建筑过程中以砖和瓦为基本单位，只需将它们放置到合适的位置并固定就可以了。而构件就相当于程序设计过程中的砖和瓦，完成对构件属性的定义和操作的说明就完成了所有编程工作。

1.3 算法概述

著名计算机科学家沃思提出一个公式：

$$\text{程序} = \text{数据结构} + \text{算法}$$

其中，数据结构是对程序中数据的描述，主要是数据的类型和数据的组织形式；算法是对程序中操作的描述，即操作步骤。数据是操作的对象，操作的目的是对数据进行加工处理，以得到期望的结果。算法是灵魂，数据结构是加工对象。

1．算法的基本概念

在日常生活中，人们处理问题都有一定的步骤。例如考大学就要有这样的步骤：要填报名单，交报名费，拿到准考证，再参加考试，填报志愿，得到录取通知书。这些步骤都按一定的次序进行，缺一不可，次序错了也不行。广义地说，为解决一个问题而采取的方法和步骤，就称为算法。算法解决“做什么”和“怎么做”的问题。计算机算法是用计算机求解一个具体问题或执行特定任务的一组有序的操作步骤(或指令)。

解决一个问题，可以有不同的方法和步骤。一般来说，希望采用简单的和运算步骤少的方法。不仅要保证算法正确，还要考虑算法的质量，选择合适的算法。算法是根据问题定义中的信息得来的，是对问题处理过程的进一步细化，但它不是计算机可以直接执行的，只是编制程序代码前对处理思想的一种描述，因此它是独立于计算机的，但它的具体实现是在计算机上进行的。

2．算法的特性

一个算法应该具有以下 5 个重要的特征。

(1) 有穷性：一个算法必须保证执行有限步之后结束。在执行有限步之后，计算必须终止，并得到解答。也就是说，一个算法的实现应该在有限的时间内完成。

(2) 确切性：算法的每一步骤必须有确切的定义。算法中对每个步骤的解释是唯一的。

(3) 零个或多个输入：输入指在执行算法时需要从外界取得的必要的信息。一个算法有零个(即没有)或多个输入，以刻画运算对象的初始情况。

(4) 一个或多个输出：输出是算法的执行结果。一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

(5) 有效性：又称可行性。算法中的每一个步骤能够精确地运行，并得到确定的结果，即用笔和纸通过有限次运算可完成。

3．算法的描述

算法的常用表示方法有如下几种：

(1) 使用自然语言描述算法；

(2) 使用流程图描述算法；

(3) 使用 N-S 结构图表示算法；

(4) 使用伪代码描述算法。

下面以求解 n!=1×2×3×4×5×…×(n−1)×n 为例说明算法的 3 种描述方法。

第 1 种：使用自然语言描述求 n!的算法。

① 确定一个 n 的值；

② 假设等号右边的算式项中的初始值 i 为 1；

③ 假设变量 f 中存放 n!的值，且初始值为 1；

④ i≤n 时，执行⑤，否则转出执行⑧；

⑤ 计算 f 乘以 i 的值后，重新赋值给 f；

⑥ 计算 i 加 1，然后将值重新赋值给 i；

⑦ 转去执行④；

⑧ 输出 f 的值，即 n! 的值，算法结束。

从上面的这个描述的求解过程中不难发现，使用自然语言描述算法的方法虽然比较容易掌握，但是存在着很大的缺陷。例如，当算法中含有多分支或循环操作时很难表述清楚。另外，使用自然语言描述算法还很容易造成歧义(称之为二义性)，可能使他人对相同的一句话产生不同的理解。

第 2 种：使用流程图描述求 n!的算法，如图 1-1 所示。

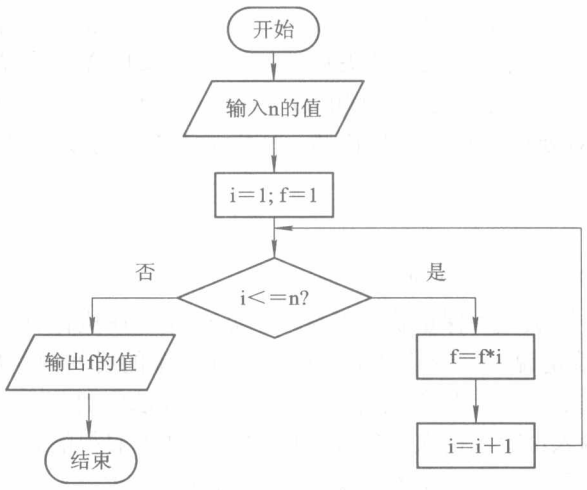
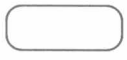
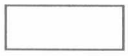


图 1-1　求 n！的算法流程图

从图 1-1 中，可以比较清晰地看出求解问题的执行过程。流程图用一些图框表示各种操作。用图形表示算法，直观形象，易于理解。在进一步学习使用流程图描述算法之前，有必要对流程图中的一些常用符号做一个解释。流程图的符号如表 1-1 所示。

表 1-1　流程图的符号

符　号	名　称	作　用
	起止框	表示算法的开始和结束
	输入/输出框	表示算法过程中，从外部获取信息(输入)，然后将处理过的信息输出
	判断框	表示算法过程中的分支结构。菱形框的 4 个顶点中，通常用上面的顶点表示入口，根据需要用其余的顶点表示出口
	处理框	表示算法过程中需要处理的内容。只有一个入口和一个出口
	流程线	在算法过程中指向流程的方向
	连接点	在算法过程中用于将画在不同地方的流程线连接起来
	注释框	对流程图中某些框的操作做必要的补充说明，可以帮助读者更好地理解流程图的作用。不是流程图中的必要部分

流程图的缺点是在使用标准中没有规定流程线的用法，因为流程线能够转移、指出流程控制方向，即算法中操作步骤的执行次序。在早期的程序设计中，曾经由于滥用流程线的转移而导致了可怕的“软件危机”，震动了整个软件业，并展开了关于“转移”用法的大讨论。

第 3 种：使用 N-S 结构图描述求 n! 的算法。

针对传统流程图存在的问题，美国学者 I.Nassi 和 B.Shneiderman 于 1973 年提出一种新的结构化流程图形式，即 N-S 结构图。Chapin 在 1974 年对其进行了进一步扩展，因此，

N-S 结构图又称为 Chapin 图或盒状图。

N-S 结构图的主要特点是完全取消了流程线，不允许有随意的控制流，全部算法写在一个矩形框内，该矩形框以 3 种基本结构(顺序、选择、循环)描述符号为基本复合而成。

无论是使用自然语言还是使用流程图描述算法，都仅仅是表述了编程者解决问题的一种思路，无法被计算机直接接受并操作。由此引进了第 4 种非常接近于计算机编程语言的算法描述方法——伪代码。

第 4 种：使用伪代码描述求 n! 的算法。

```
算法开始：
输入 n 的值;
置 i 的初值为 1;
置 f 的初值为 1;
当 i <= n 时，执行下面的操作
  使 f = f* i;
  使 i = i + 1;
  (循环体到此结束)
输出 f 的值;
算法结束
```

也可以写成以下形式：

```
BEGIN                    /*算法开始*/
输入 n 的值;
i ← 1;                   /*为变量 i 赋初值*/
f ← 1;                   /*为变量 f 赋初值*/
while  i<=n              /*当变量 i <=n 时，执行下面的循环体语句*/
  { f ← f* i;
  i ← i + 1; }
输出 f 的值;
END                      /*算法结束*/
```

伪代码是一种在书写程序或描述算法时使用的非正式、透明的表述方法，它并非是一种编程语言。这种方法针对的是一台虚拟的计算机。伪代码通常采用自然语言、数学公式和符号来描述算法的操作步骤，同时采用计算机高级语言(如 C、Pascal、VB、C++、Java 等)的控制结构来描述算法步骤的执行顺序。伪代码书写格式比较自由，容易表达出设计者的思想，写出的算法很容易修改。但是用伪代码写的算法不如流程图直观，可能会出现逻辑上的错误。

4. 算法分析

对于同一个问题可以设计出不同的算法，各种算法之间肯定有是否适合和“好”“差”之分。衡量一个适合的、“好”的算法，在其必须是正确的基础上，还应遵循下列几个方面：

(1) 易读性：算法应易于阅读和理解，以便于调试、修改和扩充。

(2) 高效性：算法应具有较高的时间效率和空间效率，即占用较短的执行时间和较少的存储空间。当然，这两者都和问题的规模有关。

(3) 健壮性：正确的输入能得到正确的输出，这是算法必须具有的特性之一。但当遇到非法输入时，算法应能做出反应或处理(如提示信息等)，而不会产生不需要的或不正确的结果。

例如，在算法中总是采用较简单的方法以及模块化函数以增强算法的易读性；采用步骤较少的求解方法以提高算法的时间效率；对可能发生的各种现象及输入形式给出应对措施以提高算法的健壮性。

要确定一个算法是适合的、“好”的算法，就需要进行算法分析。算法分析的两个主要方面是分析算法的时间效率和空间效率，目的是以求改进算法或对不同的算法进行比较。鉴于目前情况下运算空间较为充足，我们把算法的时间效率分析作为主要内容。

算法运行的时间分析和程序运行的时间分析是有区别的。同一算法由不同的程序员所编写的程序有优劣之分，程序运行的时间也就有所不同；程序在不同的机器上运行的速度又和机器本身的速度有关。而我们感兴趣的是对解决问题的算法作时间上的度量分析，或对解决同一问题的两种或两种以上的算法运行时间加以比较。

1.4 C 语言程序设计

1.4.1 C 语言的发展历史

C 语言是国际上流行的、很有发展前途的计算机高级语言。C 语言适合作为“系统描述语言”。它既可以用来编写系统软件，也可以用来编写应用程序。

以前操作系统等系统软件主要采用汇编语言编写。汇编语言依赖于计算机硬件，程序的可读性、可移植性都比较差。为了提高可读性和可移植性，人们希望采用高级语言编写这些软件，但是一般的高级语言难以实现汇编语言的某些操作，特别是针对硬件的一些操作(如内存地址的读写，直接硬件、二进制位的操作)。人们设法寻找一种既具有一般高级语言特性，又具有低级语言特性的语言，C 语言就在这种情况下应运而生。

C 语言的发展如下：

ALGOL60→CPL→BCPL→B→C→标准 C→ANSI C→ISO C

ALGOL60：一种面向问题的高级语言。ALGOL60 离硬件较远，不适合编写系统程序。

CPL(Combined Programming Language，组合编程语言)：CPL 是一种在 ALGOL60 基础上发展出来的更接近硬件的语言。CPL 规模大，实现困难。

BCPL(Basic Combined Programming language，基本的组合编程语言)：BCPL 是对 CPL 进行简化后的一种语言。

B 语言：B 语言是对 BCPL 进一步简化所得到的一种很简单的接近硬件的语言。B 语言取 BCPL 语言的第一个字母。B 语言精练、接近硬件，但过于简单，数据无类型。B 语言诞生后，UNIX 开始用 B 语言改写。

C 语言：C 语言是在 B 语言基础上增加数据类型而设计出的一种语言。C 语言取 BCPL 的第二个字母。C 语言诞生后，UNIX 很快用 C 语言改写，并被移植到其他计算机系统。

标准 C、ANSI C、ISO C：标准化的 C 语言。

最初 UNIX 操作系统是采用汇编语言编写的，B 语言版本的 UNIX 是第一个用高级语言