高等学校教材

计算机软件实践教程

郭浩志　主编

西北电讯工程学院出版社

高等学校教材

计算机软件实践教程

郭浩志 主编

西北电讯工程学院出版社

1985

内 容 简 介

本书旨在介绍我国高等工科院校计算机软件专业各有关课程上机实习的内容、方法和要求，并适当给出示例和实习题，但不局限于具体计算机和程序语言。书中比较全面地介绍了我国当前工科院校主要软件课程的实践性环节。全书共分两篇。第一篇系基本知识和实习入门训练。第二篇介绍了七门主要软件课程（高级语言、数值分析、汇编语言、系统实用程序、数据结构、操作系统和编译原理等）的实践性教学环节。本书的组织既注意到系统性和前后内容、形式的一致性，各章内容又具有相对的独立性，完全可以单独用来指导该门课程的实习。

本书系高等工科院校计算机软件专业各有关课程的教学用书，供大学生在大学整个学习期间使用，也可作为软件教师、研究生或工程科技人员的参考书。

高等学校教材

计算机软件实践教程

郭浩志　主编

责任编辑　刘录英

西北电讯工程学院出版社出版

西北电讯工程学院印刷厂印刷

陕西省新华书店发行　　各地新华书店经售

开本 787×1092 1/16　印张 31 6/16　印刷字数 774千字

1985年6月第一版　　1985年6月第一次印刷　　印数 1—30,000

统一书号：15322·17　　定价：6.00元

出 版 说 明

根据国务院关于高等学校教材工作分工的规定，我部承担了全国高等学校工科电子类专业课教材的编审、出版的组织工作。从一九七七年底到一九八二年初，由于各有关院校，特别是参与编审工作的广大教师的努力和有关出版社的紧密配合，共编审出版了教材159种。

为了使工科电子类专业教材能更好地适应社会主义现代化建设培养人才的需要，反映国内外电子科学技术水平，达到“打好基础、精选内容、逐步更新、利于教学”的要求，在总结第一轮教材编审出版工作经验的基础上，电子工业部于一九八二年先后成立了高等学校《无线电技术与信息系统》、《电磁场与微波技术》、《电子材料与固体器件》、《电子物理与器件》、《电子机械》、《计算机与自动控制》，中等专业学校《电子类专业》、《电子机械类专业》共八个教材编审委员会，作为教材工作方面的一个经常性的业务指导机构，并制定了一九八二～一九八五年教材编审出版规划，列入规划的教材、教学参考书、实验指导书等共217种选题。在努力提高教材质量，适当增加教材品种的思想指导下，这一批教材的编审工作由编审委员会直接组织进行。

这一批教材的书稿，主要是从通过教学实践、师生反映较好的讲义中评选择优和从第一轮较好的教材中修编产生出来的。广大编审者，各编审委员会和有关出版社都为保证和提高教材质量作出了努力。

这一批教材，分别由电子工业出版社、国防工业出版社、上海科学技术出版社、西北电讯工程学院出版社、湖南科学技术出版社、江苏科学技术出版社、黑龙江科学技术出版社和天津科学技术出版社承担出版工作。

限于水平和经验，这一批教材的编审出版工作肯定还会有许多缺点和不足之处，希望使用教材的单位、广大教师和同学积极提出批评建议，共同为提高工科电子类专业教材的质量而努力。

电子工业部教材办公室

前　言

本书是高等院校工科电子类计算机专业统编教材之一，由计算机与自动控制教材编审委员会计算机编审小组评选审定，并推荐出版。

本书是按计算机编审小组审定的编写大纲进行编写和审阅的，由国防科学技术大学郭浩志担任主编，西北电讯工程学院蔡希尧和华中工学院徐则琨担任主审。

本书以计算机软件教学计划中对大学本科生培养目标的基本要求为出发点，着眼于加强学生的基本技能和科学作风的培养和训练。全书着重介绍各种软件课程上机实习的目的、要求、内容和方法。大部分实习内容都设有一两个示例，并提供若干实习题让各校学生实习时选择，有些部分还给出必要的提示和思考题。每章最后列出该章的主要参考文献。全书共分两篇。第一篇通过软件实习的入门训练和基本知识的学习，使学生在实习基本技能方面受到较为科学和严格的基础训练，在程序设计方法、技巧、风格和素养方面得到初步培养，为以后各门软件课程的实习打下良好的基础。为加强学生学习的主动性，在第一篇第一章概述中，还简单介绍了教师组织实习的方式、特点和要求。第二篇包括七门软件课程（高级语言、数值分析、汇编语言、系统实用程序、数据结构、操作系统和编译原理等）的实习，其中大多数课程当前已列为我国高等院校软件的必修课程，其实习是软件实践性教学环节的主要内容。

本书是在总结我国若干所理工科大学软件课程近几年教学实习的经验体会和比较广泛搜集其它院校有关资料和意见的基础上编写的。编写时尽量考虑全国大多数院校的通用性。各校在采用本书的同时，宜根据各自的教学大纲和实际条件，结合具体计算机和程序语言，作适当的选择、修改和补充。

参加本书编写的还有西北电讯工程学院的金益民（第二篇第一章），国防科学技术大学的齐治昌（第二篇第二章）和陈怀义（第二篇第三章），清华大学的田金兰（第二篇第四章），南京工学院的朱静华（第二篇第五章）和上海交通大学的徐良贤（第二篇第六章）。参加各章审稿的还有南京工学院徐连信、秦振松和张月琳，清华大学吴家勋和严蔚敏，西北电讯工程学院陈家正和罗昌隆，东北工学院俞馥敏和刘栋臣等同志。

在本书编写过程中，得到了清华大学、上海交通大学、西北电讯工程学院、南京工学院、东北工学院、西安交通大学、华中工学院、国防科技大学和其它院校许多教师、研究生和大学生的大力支持和帮助，在此表示谢意。

本书系初次尝试，国内外尚未见有同类书籍出版可供借鉴，因而缺乏编写这类教科书的经验，编写中的缺点错误在所难免，热诚欢迎读者批评指正。

编　者

目　录

第一篇　基础篇

第二篇　核心篇

第一篇 基 础 篇

第一章 概 述

第一节 软件实习的目的与要求

一、原则

(1) 软件实习是计算机科学与工程有关专业的必修内容。有关教学大纲要明确指出软件实习的地位、目的与要求，使课堂教学、课外练习与上机实习，以及各有关课程之间的实习有机地结合起来。

(2) 明确各课程软件实习的分工、侧重点和衔接关系，尽量减少不必要的重复。做到循序渐进，目的明确，联系紧密。软件实习，要始终重视程序设计基本方法的训练，逐步提高要求。

(3) 确保学生在校期间有较充分的上机实习时间，每个学期都有所安排，使上机实习不断线。

(4) 精心设计和安排好每一门课程的每一次作业的实习。

(5) 实习内容一方面要有基本要求，另一方面尽量考虑有选择余地，使不同程度的学生均能适用。

二、实习目的与要求

通过软件课程的学习和一系列的实习，使学生能熟练地运用汇编语言和至少一种高级语言独立完成有关软件(系统软件、应用软件)的设计、编制、上机调试和运行，加深对课堂讲授的概念和原理的理解，加强程序设计基本方法的训练，初步具备利用计算机进行科学研究的能力。这些能力的培养是专业培养目标的一个重要组成部分。

在整个学习期间，力争每学期都结合有关课程安排学生上机实习。在制订教学计划时，应将各有关实习内容，比较均匀而有机地安排在各个学期。在不同课程中，安排不同要求的适量的上机实习。从当前情况来看，总实习时间(不包含毕业设计的实习)宜有120～180小时。有条件的学校，逐步争取到200小时，甚至更多一点。

根据需要可设置一至三种高级语言和一种汇编语言的程序设计，作为有关软件实习课的先行课程。各种程序设计课程的实习要求，应由简到繁，由易到难，逐步加大实习量，不断加深实习深度。各种语言程序设计课程的实习是其它软件实习课程的实习基础，宜单独设课，严格要求，加强实习指导，确保有较充分的实习时间。

第二节 软件实习的组织与方式

一、实习教学环节

软件实习宜根据各有关课程的特点，在不同教学环节中分别采用各种不同的方式，以培

养和不断提高学生的软件基本技能。这些实习教学环节主要指基础训练、课程应用、课程设计、生产实习和毕业设计。

（一）基础训练

各种语言程序设计课程的实习是整个软件实习的基础，应扎扎实实安排好各种语言的重要语句(指令)、数据类型、程序结构和特色以及程序设计基本方法的初步训练。尤其最初一、两次入门性训练，更应精心安排，实习量不宜过大，学生先从模拟入手。但要严格要求，加强指导，使学生在受到比较正规而扎实的基础训练的过程中，逐步达到独立上机的目的。

（二）课程应用

以各种程序设计语言为工具，结合《数值分析》和《数据结构》等课程，进行数学和数据结构等领域若干基本算法的实习，使学生初步掌握程序设计语言在数值计算和非数值计算方面的若干应用。要求学生独立编制和调试出有关应用问题的程序，并力求达到程序质量较高，资料齐全。

（三）课程设计

在诸如《系统实用程序设计》、《编译原理》、《操作系统》和《数据库》这些课程学习告一段落或结束时，可集中一段时间，以小组为单位，集体完成中等规模的大作业——简易的系统软件。例如简单的实用程序，编译程序，操作系统或小型数据库，以培养学生设计、开发软件的能力，以及集体完成中等规模软件所应具备的组织能力和协作能力。

具体实施时，可从类似上列四种课程中选择其中的一、两种安排课程设计，其余仍可按“课程应用”的教学形式进行实习。

（四）生产实习

该教学环节在专门的教学实习时间内进行，仍以(三)中所列那些课程为实习内容，但强调不能停留在单纯为了验证某种原理的常规实习上，而要从科研、生产实际中选题，让学生集体真刀真枪地设计、编制和调试出完整可用的软件。

（五）毕业设计

若在此教学环节中安排实习，可与生产实习通盘考虑。

二、实习、练习与课堂讲授

课堂讲授为实习和练习指明了范围、内容和方法。而实习和练习则进一步巩固和加深课堂讲授的知识，它们均是教学大纲中的重要组成部分。

平时练习题多面宽，各部分内容都可照顾到，但缺点是“纸上谈兵”，学生无法上机验证程序是否正确。软件实习是平时练习的继续和深入，学生亲自调试出一个程序往往胜过在纸上编写十个程序。不能以静态的程序练习替代动态的程序实习。

在加强实践环节的情况下，弄不好容易导致学时膨胀，学生负担过重的现象出现。应采取相应的措施。除了教师努力提高课堂教学质量，切忌填鸭式教学和在课外变相增加学时外，还可有意识地把实习题与练习题结合起来，以减轻实习的准备量。此外，课堂讲授时间与实习、练习时间，宜按一定比例安排。汇编语言和高级语言程序设计的讲授与实习练习比例约为 3:1～2:1，而其余课程可为 6:1～4:1。

三、教师指导

在各种语言程序设计的基础训练阶段，教师宜加强上机指导，尤其是指导学生充分利用现代计算机系统所提供的各种手段，例如交互式联机调试软件，诊断和改正程序中的错误，并学会验证程序正确性的若干技术，使学生受到比较系统和严格的训练。此后，教师可有意识地逐步减少辅导，努力培养学生的独立工作能力。在课程设计和生产实习阶段，实习可以小组为单位，使学生受到集体完成中等规模软件所应具备的组织能力和协作能力的训练。此外，教师应允许优秀学生或有特长的学生自由选题，并指导完成带探索性和创造性的独立研究活动。

四、实习程序规模

实习题应逐步由简到繁，由易到难，由小到大，实习量应逐步由少到多。各种语言程序设计的实习，初次可选择十几条指令(语句)规模的程序，甚至允许学生依样画葫芦，逐步加到100句左右。课程应用的实习可扩充到100～200句。到了课程设计和生产实习，实习量可达500句左右的规模。

实习程序的类型，从练习性程序过渡到各种应用程序和系统程序，从无接口关系的程序到有软件接口的系统实用程序，再到有软硬件接口的系统程序。从一般的程序编制到软件的设计、开发和维护。从单纯为了验证某些原理的假想程序到实际问题的程序。

五、多种形式的考核制度

(1) 每次实习均须要求每人(组)完成上机实习报告。实习报告的主要内容，除了源程序和流程图外，还应要求学生书写有关设计、调试和使用等方面的情况。最好每次实习均印发统一格式的实习报告单给学生填写。

(2) 抽样检查。教师从磁盘调出学生编写的程序，输入数据，检查结果的正确性。并适当设置故障和问题，以激发学生智力和深入考虑问题。

(3) 现场考核。汇编语言和第一高级语言的实习，可逐步试行现场考核制度，即要求学生在终端上限时完成试题。

(4) 实习应评定成绩。譬如，根据不同课程和教学要求，可分别占该门课程总成绩的20%～50%。

第二章　基本知识

第一节　软件实习环境

软件实习，与一般利用计算机算题一样，是在整个计算机系统的控制与支持下进行的。计算机系统指的是计算机硬件(简称硬件)和计算机软件(简称软件)。

一、计算机硬件

计算机的作用，主要是按人的意图代替人的部分脑力劳动。人处理问题，首先通过眼、耳、鼻等感觉器官接受外界信息，记入大脑，再经过分析、思考和计算，得出结论，最后用各种方式，如语言、文字书写等，将结果告诉外界。类似地，计算机首先从外界接受原始信息(如源程序、输入数据)，经过计算、判断，再将结果送出到外界。这整个过程是在硬件与软件密切配合下完成的。对于大脑的三种功能(指记忆，分析、思考和计算，按一定规则对信息进行控制)，计算机相应有存贮器、运算器和控制器。这三部分统称为中央处理器，简称主机或 CPU。

运算器负责对各种不同结构的代码进行各种算术运算和其它非数值运算。主存贮器（内存）是一种记忆装置，直接与运算器发生联系。它能暂存用户程序、原始数据、中间结果、最终结果以及计算机系统软件程序。控制器主要指中央控制器和指令控制器，前者指挥全机各部件相互协调地工作，后者控制一条指令的执行过程。

主存贮器的容量一般都不大够用，通常还配置外存贮器，又称辅助存贮器。常用的有磁盘和磁带。外存不直接与运算器发生联系，只与内存打交道，充当'后台'的角色。计算机系统把内存中暂时不用的代码记入外存，让内存空出来存放急用的代码。外存的代码，在需要时再一批批地调入内存。

把信息从外部介质输入到内存或从内存把信息输出到外部介质的过程称为输入输出。它是通过输入输出通道(简称为通道)来完成的。通道是中央处理部件与输入输出设备之间的桥梁，它起到一台小处理机的作用。常用的输入设备有纸带输入机、卡片输入机等等。常用的输出设备有行打印机、显示装置、绘图仪、纸带输出机、卡片输出机等等。带显示装置的键盘设备作为计算机的终端设备，主要用于人机通讯。大多数输入输出设备(统称为外部设备)在程序控制下进行操作，即由程序的指令选取所需设备，规定是输入还是输出，并指明信息传输时内存起始地址和传输数量。少数外部设备是手动的，如键盘、光笔等把信息直接送至内存去。

二、计算机软件

早期计算机是台裸机，完全由物理设备组成，仅有机器语言，即代码指令。用机器语言算题，人工干预频繁，时间耗费多，不能充分发挥计算机效率，使用范围不广泛。后来，人们利用计算机能高速完成复杂逻辑运算的这一特点，事先编制好一些程序，随计算机一同交付用户使用。后来逐步发展成为计算机系统的必备部分。任一用户题目的执行，都是在计算

机物理装置(硬件)和上述计算机程序(软件)共同控制和支持下完成的。软件代替了原先必须由人承担的部分工作，实现了合理组织整个解题过程，简化或代替解题工作流程的有关环节，从而达到既充分发挥机器效率又大大方便用户这两个目的。

(一) 操作系统

任一程序的运行，既要依赖各种硬件资源(指处理机、主存、外部设备等)，又要用到各种软件资源(指各种实用程序、应用程序、数据库等等)。操作系统是担负组织和管理各种软硬件资源，使之协调一致地高效地完成各种各样任务的一种程序。按照资源管理的观点，操作系统可由四大部分组成：

1. 处理机管理程序　处理机管理程序的主要功能是确定哪个用户的程序将在CPU上运行。处理机分两级管理。第一级为作业(用户题目)调度。在多终端分时系统中，CPU轮流执行各终端用户的程序。在实时系统中，CPU先执行有实时要求的程序，空闲时再执行其它程序。在批量多道系统中，可根据各种因素选取程序进入运行，使既能充分满足用户的要求，又能最大限度地提高计算机的利用率。当被选中的若干个作业进入运行后，操作系统再进一步进行第二级调度(进程调度)。进程处于运行、就绪、封锁三种状态之一。操作系统负责管理和转换这些状态。

2. 存贮管理程序　存贮管理程序的主要功能是负责将主存贮器分配给进程使用；在多道系统中对主存实行保护；还可提供虚拟存贮，使用户在使用计算机时不必考虑主存容量的限制。

3. 设备管理程序　设备管理程序的主要功能是将外部设备分配给进程使用，并对外设实现各种具体的操作。此外，还可实行假脱机的控制。

4. 信息管理程序　信息管理程序的主要功能是可靠地保存贮存在外存中的大量信息(通常以文件、卷宗的形式存放)，并按用户的要求对文件进行种种操作(如建立、撤销、打开、关闭、读、写……)。

(二) 系统实用程序

这是一套协助管理、维护和使用计算机的程序。在终端分时系统中，它们一般都各自配有一组命令，以利于用户交互使用。下面简单介绍一些常见的系统实用程序。

1. 汇编程序　它是一种翻译程序，专门负责把用汇编语言写的源程序翻译成目标程序(模块)。汇编语言是一种紧密依赖于具体计算机的低级语言。其核心部分的符号机器指令与计算机原有的代码机器指令一一对应。汇编程序一般通过两遍扫描(第一遍进行符号定义，第二遍生成目标代码)完成翻译。

2. 编译程序　它也是一种翻译程序，专门负责把用高级语言，例如FORTRAN、COBOL、PASCAL等书写的源程序翻译成目标程序(模块)。高级语言是一种独立于具体计算机的语言。编译程序是一种比较复杂的加工程序。一般经过词法分析、语法分析、语义分析、优化和目标代码生成等五个阶段的处理，最后译成能尽量发挥硬件资源效率，并且可在计算机上执行的目标程序。

3. 编辑程序　它提供对任一字符串进行定位、增添、替换、删去、显示等各种功能，形成或修改源程序文件、数据文件、资料表格和其它信息。

4. 连接程序　它将汇编程序或编译程序或两者共同产生的一个或多个目标模块与其它介质(如各种外存和输入设备)上的一个或多个目标模块连接成一个可在计算机上执行的程

序。

5. 调试程序(查错程序) 它允许用户随时设置断点，动态地测试、修改和输出有关主存单元的内容，以便在目标程序一级进行查错和改错。

此外，还有很多其它系统实用程序，如文件转换程序，文件转贮程序，文件比较程序，分类排序程序，目标模块修补程序，库管理程序……。

(三) 应用程序

为科学技术、工业、农业、商业和经济等各个领域而编制的有关计算、设计、控制、管理、实验以及各种事务处理的程序，统属于应用程序。

第二节 算法及其表示方法

一、什么是算法？

例：试计算两个正整数 m 和 n 的最大公因子。

求两个给定正整数 m 和 n(设 $m>n$)的最大公因子，可通过反复执行下列三个操作来实现：求 m 和 n 的余数 r；变换 m 和 n 值(以 n 取代 m，以 r 取代 n)；判别 r 是否为零。在此过程中，一旦发现 $r=0$，m 即为最大公因子；否则，重复执行以上三个操作。

计算两个正整数 m 和 n 的最大公因子的过程为：

第一步：读入两个正整数 m 和 n(设 $m>n$)

第二步：求 m 和 n 的余数 r

用 mod(m, n)表示求余数，即

$$r=\mathrm{mod}(m,\ n)=m-\lfloor m/n\rfloor*n$$

这里用$\lfloor m/n\rfloor$表示小于或等于 m/n 的最大整数。

例如：$m=24$，$n=9$ 则

$$r=\mathrm{mod}(24,9)=24-\lfloor 24/9\rfloor*9$$
$$=24-2*9=6$$

表 1-2-1

步骤	各变量的变化		
	m	n	r
第一步	24	9	未定义
第二步	24	9	6
第三步	9	6	6
第四步	9	6	6
第二步	9	6	3
第三步	6	3	3
第四步	6	3	3
第二步	6	3	0
第三步	3	0	0
第四步	3	0	0
第五步	3	0	0

第三步：变换 m 和 n 值

$m:=n;\ n:=r$

第四步：判别 r 是否为零。如果 $r=0$，则 m 为最大公因子，转入第五步；如果 $r\neq 0$，则返至第二步，重复第二、第三和第四步操作。

第五步：输出 m 值，即为最大公因子。停止。

从第一步开始，一步一步执行，直至 $r=0$，转至第五步，输出最大公因子，然后停止。例如 $m=24$，$n=9$，执行过程如表1-2-1所示。

如果给定另一组 m 和 n 值，遵循上述过程，又可得到另一确定的结果。上面叙述的求两个正整数 m 和 n 的最大公因子的过程具体给出了一个算法。它就是所谓的辗转相除法。

二、算法的表示法

算法只对解问题的方法进行了描述，但是无法将它直接输入计算机，以完成相应的计算(无论是对数值问题还是非数值问题)。程序设计就是用程序去表示和实现算法。随后，将程序输入计算机，由计算机按照程序进行计算，以求得问题的解。因此，在编写程序之前，必须清晰地描述算法。算法表示的方法通常有两类，一类是流程图(框图)，另一类是非形式的算法描述语言。

(一) 流程图

描述算法和信息处理流程的图形称为流程图。它是由各种基本图形组成的。

仍以求两个正整数 m 和 n 的最大公因子的辗转相除法为例，其流程图如图 1-2-1 所示。

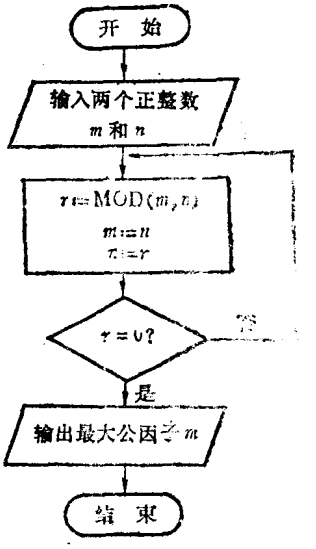

图 1-2-1 求最大公因子流程图

图中椭形图形表示流程图的开始或结束。平行四边形表示输入或输出。矩形表示一般处理。菱形表示判别或检查。箭头表示流程图的路径和方向。流程图的基本图形已有国家标准，将在本章第五节图 1-2-5 详细画出。

(二) 非形式算法描述语言

它酷似描述程序的高级语言。为便于阅读程序，只有算法的核心部分才采用高级语言的语句描述，其余部分或用自然语言(如汉语、英语等)描述，或省略(如说明语句)。对于上面求两个正整数最大公因子的例子，可用非形式算法描述语言描述如下：

```
PROCEDURE GCD
  {Finding the greatest common divisor given two positive integers m and n}
  BEGIN
    输入 m 和 n;
    REPEAT
      r:=MOD(m,n);{r is a remainder.}
      m:=n;
      n:=r
```

```
    UNTIL r=0;
    输出 m
END
```

可以看出，将用非形式算法描述语言所描述的程序，进一步加细为用高级语言所描述的程序是不难的。譬如，用 PASCAL 语言描述上列算法的程序，请见本章第四节。本章第五节介绍了一个具体的非形式算法描述语言。上列程序及以后的一些例子就是用它来描述的。

第三节 程序设计方法

一、逐步求精法

从问题的体系结构、解决策略和要求出发，先提出一个解决办法，然后对此办法逐步补充加细，使之不断明确和完善，直至整个问题可用某种程序语言明确地、完全地描述出来为止。人们一开始往往比较习惯于用公式，甚至用某种自然语言(如汉语)来描述问题，但这比较粗糙。对解决办法逐步精细化的技术有三种：第一种叫分割技术。它将一个问题划分为若干互相独立的小问题，然后依次解决每个小问题。第二种叫递推技术。每使用此方法一次，就能使问题朝着彻底解决的方向推进一步，反复使用此方法，就能得到最终的解。第三种叫分析技术。将问题进行严格分类，对不同问题采用不同的解法。这三种技术的使用也为以后的程序编制打下良好的基础。实际上，第一种技术针对复合语句，第二种技术针对各种循环语句，第三种技术针对各种条件语句。可以说，任一问题的解决都是综合运用这三种技术的结果。

逐步求精法的特点是，考虑问题时先整体后局部，先抽象后具体。因此，这种方法是一种自顶向下法。此方法安全可靠，最终得到的程序的总体结构完整清晰，含义明确，易读性、易理解性均佳，也利于程序正确性的证明。

例：求素数。

为确定[1,n](n为非 1 的正整数)范围内的所有素数，可以按递增次序查看自然数集合，从而选出满足素数定义的所有素数。该题目可用一个算法描述语言的程序表述如下：

```
PROCEDURE fprime
  {Finding primes 2,3,5,……}
  BEGIN
    READ(n);
    number:=1; {number is natural numbers 1,2,3,4,…}
    WHILE number<n DO
      BEGIN
        number 是下一个素数;
        WRITE(number)
      END
  END{FPRIME}
```

“number 是下一个素数”是一个汉语句子。它尚不具体，需要进一步精细化。为了顺序生成下一个素数，首先要按递增顺序生成自然数，然后按素数定义判定该自然数是否为素数。重复上述操作，直至选出下一个素数为止。为此，引入逻辑变量 prim。若 prim 为真，表示

number 是下一个素数，否则查看下一个自然数。为进一步展开“number 是下一个素数”，可反复执行下列语句序列：

```
{Finding the next prime}
number:=number+1;
prim:=number 是一个素数
{If the nature number is a prime, then prim:=TRUE.}
```

其中“number 是一个素数”表示按素数定义选出素数。number是素数，当且仅当number只能被 1 及其自身整除时，即若被 2、3、4、…、number－1除时，其余数均不为零。令 k 可取值 2、3、4、…、number－1，lim 为 number－1。这样，“prim:=number 是一个素数”可加细。为进一步展开“number 是下一个素数”，可反复执行下列语句序列：

```
{Finding the next prime.}
number:=number+1;
k:=2; {k—2,3,4,…,number-1.}
lim:=number-1; {lim—a limit value of k.}
REPEAT {Is the number a prime?}
  IF number 能被 k 整除
    THEN prim:=FALSE
    ELSE
      BEGIN
      k:=k+1;
      prim:=TRUE
      END
UNTIL NOT(prim) OR(k 达到 lim)
```

上列“number能被 k 整除”表示一个逻辑表达式，它可进一步精细为

$$\mathrm{MOD}(\mathrm{number},k)=0$$

若余数为零，则上列逻辑表达式成立。此时，number 不是素数，prim 为假，因而需顺序查看下一个自然数。若余数不为零，上列逻辑表达式不成立，number 可能是素数，此时应 k 加 1，继续测试 number 是否为素数。直至 k 达到 lim，且 prim为真，才表示该自然数number为素数。重复上述操作，可找出[1,n]范围内的所有素数。

此外，“k 达到 lim”可精细为

$$k\geqslant \mathrm{lim}$$

至此，我们可将上述算法描述语言程序完全精细化为

```
PROCEDURE fprime
  {Finding primes 2,3,5…}
  BEGIN
    READ(n);
    number:=1;{number is natural numbers 1,2,3,4,…}
    WRITE(2);{write prime 2}
    WHILE number+1<n DO
      BEGIN
        { Finding the next prime}
```

```
      number := number + 1;
      k := 2;{k—2,3,4,…,number - 1.}
      lim : = number - 1;{lim—a limit value of k.}
      REPEAT {Is the number a prime?}
        IF MOD(number, k) = 0
          THEN prim := FALSE
          ELSE
            BEGIN
              k := k + 1;
              prim := TRUE
            END
      UNTIL NOT(prim) OR (k≥lim);
      IF prim THEN WRITE(number)
    END
END
```

二、自底向上法

自顶向下法有时效率不太高。逐步加细的过程是逐步深入认识的过程，往往免不了要回溯，即对原来已决定的某些步骤加以修改。这时就可采用自底向上法，对原先的设想进行改错、补充、加工或优化。自底向上法往往配合自顶向下法使用，几乎不单独采用。

我们再回到上面求素数的例子。除第一个素数 2 外，其它所有素数都是奇数。因此，不必查看所有的自然数。只需修改算法的有关部分，计算量就可减少一半左右。改进如下：

```
PROCEDURE fprime
  { Finding primes 2,3,5,…}
  BEGIN
    READ(n);
    number := 1;{number is odd number 1,3,5,…}
    WRITE(2);{write prime 2}
    WHILE number + 1 < n DO
      BEGIN
        { Finding the next prime}
        number := number + 2;
        k := 2;{k—2,3,4,…,number - 1.}
        lim := number - 1;{lim—a limit value of k.}
        REPEAT {Is the number a prime?}
          IF MOD(number,k) = 0
            THEN prim := FALSE
            ELSE
              BEGIN
                k := k + 1;
                prim := TRUE
              END
```

```
        UNTIL NOT(prim) OR (k≥lim);
        IF prim THEN WRITE(number)
    END
END
```

三、模块化方法

将一个大任务划分为几个功能相对独立的小任务，每个小任务的意义单一，职责明确，各任务之间的联系简单明了。这种小任务叫模块。每个模块最好只有一个入口和一个出口。每个小任务还可继续划分为更小的任务，形成层次结构。第一层(最高层)称为主控模块，它控制整个程序流程(如其它模块的启动时机，执行次数和程序的停止等等)。高层模块可以引用低层模块，反之不行。模块能被多次调用固然很好，只被调用一次时常也是值得划分的。

模块化结构的优点：

(1) 便于不同程度的人分工编制。

(2) 易于修改，便于维护。只要模块间接口不变，每个模块内部的具体细节可以任意修改，不会给整个程序带来影响。

(3) 结构清晰，易阅读易理解。

(4) 易于移植。

(5) 利于提高软件生产率。

当今最常用的高级语言，如 FORTRAN、ALGOL、COBOL 和PASCAL等虽说不算是最理想的模块化程序设计语言，但都在不同程度上具有构成模块的机构和特性，如 FORTRAN 中的子程序和函数，ALGOL 中的分程序和过程，COBOL 中的子程序、节和段，PASCAL 中的过程等。七十年代以来提出的高级语言，如 MODULA、ADA 等一般都为模块化程序设计提供了强有力的语言特性。

例：建立一个校友通信录

通信录中应设有总标题及有关名字、代号、地址和工作单位等副标题。每人的名字、代号、地址和工作单位等信息存放在一张卡片上或现场从终端拍入。每输入一张卡片，即建立通信录一项。通信录模块结构层次图及各模块如图 1-2-2 所示。

下面给出建立通信录的 COBOL 程序：

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  KWOK.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  LEVEL-6.
OBJECT-COMPUTER.  LEVEL-6.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT I-F ASSIGN TO A1.
    SELECT O-F ASSIGN TO A4.
DATA DIVISION.
FILE SECTION.
```