全国高等职业教育计算机类规划教材·工作过程系统化教程系列

软件设计与编程基础

（C 语言版）

吴艳平　岳淑玲　主　编
刘铁英　于艳华　副主编
李明革　丛得成　主　审

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是一本软件设计与编程的基础入门用书。全书以 C 语言为媒介，详细地介绍了结构化程序的开发的完整流程，主要内容包括项目背景、需求分析（项目计划书、需求规格说明书）、系统设计（概要设计说明书、详细设计说明书）、编码实现、系统测试与优化。

本书精心选择了“小学生数学选题系统”、“超市管理系统”两个项目，这些教学项目均由企业专家亲自指导，依据从简单到复杂的规律，逐步构建使用者的软件开发与设计理念，与以往的教材相比，更注重学习者能力的培养，具有一定的普遍性、实用性和可操作性。本书充分体现行为导向教学方法，在实施项目时，将面向过程的软件开发方法、算法及 C 语言相关知识均贯穿在项目这条主线上，让学生学会在工作中处理各类问题的方法，实现理论与实践一体化教学，把培养学生的能力放在首位。本教材同时提供了教材中所用项目的完整代码及配套电子课件。

本书可作为高职高专计算机专业及相关非计算机专业的教材使用，也可作为培训教材，也可供对程序设计感兴趣的初学者入门使用。

图书在版编目（CIP）数据

软件设计与编程基础：C 语言版 / 吴艳平，岳淑玲主编. —北京：电子工业出版社，2009.7
全国高等职业教育计算机类规划教材・工作过程系统化教程系列
ISBN 978-7-121- 08963-3

Ⅰ. 软… Ⅱ. ①吴…②岳… 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 086184 号

策划编辑：程超群
责任编辑：张燕虹
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：18.75　字数：480 千字
印　　次：2009 年 7 月第 1 次印刷
印　　数：4 000 册　　定价：29.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

随着企业对软件人才需求量的激增，高职院校人才培养的质量与规格也在不断提升，需要我们培养出具有较高的专业能力、方法能力和社会能力等综合能力的高素质的人才。这就要求我们开设的课程要以工作过程为导向，采用行动导向教学理念，全面培养学生的职业能力和提高学生可持续发展能力，以使他们将来能尽快适应岗位变化。但是，目前国内软件专业中还没有一门比较成熟、可行性强、适合职业人才培养的此类课程，因而也没有与此类课程相配套的教材。

本教材是在分析企业工作任务、提取典型工作任务之后，由多年从事软件开发、设计的企业人员和一线教学人员根据软件企业工作过程系统化的原理编写而成的，它的前身是《C 语言程序设计》，曾连续几年作为长春职业技术学院软件专业教材使用。近年来，根据工作过程系统化的指导思想，在企业专家的参与下，将软件开发方法、算法等多学科知识融入教材中，通过两个项目的完整设计过程对教材内容重新进行了组织、整理，根据企业需求及学生的认知规律，将项目划分成多个学习性工作任务，并针对不同的工作任务，提供相应的引导文，引领学生在任务的完成过程中逐步掌握相应的知识与技能。本教材有以下主要特色。

（1）适应工作过程系统化的教学模式。教学项目均由企业专家亲自指导，具有普遍性、实用性、可操作性。

（2）以典型工作过程项目为主线，贯穿全书始终。本教材精心选择了“小学生数学选题系统”、“超市管理系统”两个项目，又将项目分解为多个既独立又有一定联系的小任务，使学生在完成任务的过程中，掌握软件设计的基本理念和编程的基本方法。

（3）案例依据从简单到复杂的规律，逐步构建使用者的软件开发与设计理念，与以往的教材相比，更注重学习者能力的培养。本教材充分体现行为导向教学方法，在实施项目时，将面向过程的软件开发方法、算法及 C 语言相关知识均贯穿在项目这条主线上，让学生学会在工作中处理各类问题的方法，实现理论与实践一体化教学，也真正把培养学生的方法能力放在首位，使学生具有可持续发展的能力。

（4）充分体现工学结合，从教材的策划到实施，都有企业专家把关，符合企业需求。

本教材共分为 4 章，章节划分如下。

第 1 章　软件开发与程序设计，包括结构化软件开发的一般方法及开发流程。

第 2 章　小学生数学选题系统，包括了该系统的整个开发流程，即项目背景、需求分析、系统设计、编码实现、系统测试与优化。

第 3 章　超市管理系统，包括了该系统的整个开发流程（同“小学生数学选题系统”）。

第 4 章　项目赏析——学生成绩管理系统，展示了用 C 语言开发一个系统的基本方法和步骤。

本书由长春职业技术学院的吴艳平、岳淑玲任主编，刘铁英、于艳华任副主编；王军、董晶、李权威（上海朋道信息有限公司长春分公司）也参与了本书的编写工作。李明革教授、从得成工程师（启明信息技术股份有限公司）担任本教材主审工作。电子工业出版社的编辑对本书的编写提出了许多宝贵的意见，长春职业技术学院软件专业教研室主任、教务科长陈显刚同志给予了大力支持与帮助，在此表示感谢。

本教材适合作为高职高专计算机类专业软件设计与编程基础类教材，同时也可以作为培训教材，也可供对程序设计感兴趣的初学者入门使用。由于时间仓促以及编者水平有限，书中难免存在错误和疏漏之处，欢迎广大读者和同仁提出宝贵意见和建议。E-mail：wwyypp@126.com。

编　者

2009 年 4 月

目　录

第 1 章　软件开发与程序设计

1.1　任务一：软件的开发

软件被应用于世界的各个领域，对人们的生活和工作产生了深远的影响。近年来，软件开发作为现代科学研究和解决工程问题的基础，成为当今世界不可缺少的一部分。在不久的将来，软件将成为驱动社会发展的新动力。

任务描述

软件开发的目标：以客户与市场为导向、理解软件真正的需求，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性并满足用户需要的软件产品，获取最大的商业利益。

任务分析与设计

如果硬件是人的身体，那么软件就是人的思维。在开发软件过程中，开发的步骤、方法甚至技巧和习惯都会直接影响软件成品的质量。软件开发项目的成功有以下几个主要的目标作为验证标准：

（1）付出较低的开发成本。

（2）达到要求的软件功能。

（3）取得较好的软件性能。

（4）开发的软件易于移植。

（5）需要较低的维护费用。

（6）能按时完成开发工作，即时交付使用。

任务实现

1．初识“软件”

软件（software）是一系列按照特定顺序组织的计算机数据和指令的集合。

凡是能看见、能摸到的都是硬件，如键盘、显示器、鼠标、主机等。

运行于硬件上的、我们常用的瑞星、Office、操作系统等都是软件！没有软件的计算机是“裸机”，相当于废铁一堆。软件和计算机密不可分，计算机软件是为了解除人们繁重、重复的工作而产生的。软件是为了告诉计算机要做什么、如何做而编写的，是计算机能够理解的一串指令、代码、程序。

2．软件的分类

1）按功能划分

（1）贴近计算机硬件的小软件。通常“固化”在只读存储器芯片中，因此称为固件。

（2）系统软件。包括操作系统和编译器等软件，如 Windows、Linux、UNIX、Mac OS。系统软件与硬件共同搭建起一个操作“平台”，可以管理和优化计算机硬件资源的使用。

（3）支撑软件。是支撑各种软件的开发与维护的软件，又称为软件开发环境，如 Microsoft Visual C、Microsoft Visual Studio、Maromedia Dreamweaver 等。

（4）应用软件。它的种类最多，如办公软件、电子商务软件、通信软件、行业软件，游戏软件等。根据用户和所服务的领域提供不同的功能，是为了某种特定的用途而开发的软件。本书开发实例的方向就是应用软件。

2）按软件规模划分

根据开发软件所需的人员数量、时间期限以及完成的源程序（有效）行数，划分出 6 种不同规模的软件，如表 1.1 所示。

表 1.1　软件规模的分类

类　　别	参加人员数	研制期限	产品规模（源程序行数）
微型	1	1～4 周	0.5k
小型	1	1～6 月	1～2k
中型	2～5	1～2 年	5～50k
大型	5～20	2～3 年	50～100k
甚大型	100～1000	4～5 年	1M（=1000k）
极大型	2000～5000	5～10 年	1～10M

3．软件生命周期与开发阶段

软件生命周期是按时间分程的思想方法描述软件的产生直到报废的生命周期。软件生命周期包含制订计划、需求分析、系统设计、程序编码、软件测试、运行维护 6 个开发阶段，如表 1.2 所示。

表 1.2　软件生命周期的 6 个开发阶段

开 发 阶 段	关 键 问 题	结 束 标 准
制订计划	问题是什么？ 有可行的解决途径吗？	确定开发系统的目标、可行性分析报告、项目开发计划书
需求分析	系统必须做些什么？	软件需求规则说明书（系统必备功能）
软件设计	如何概括并分解出解决步骤？ 怎样具体地实现系统的模块？	HIPO 图
程序编码	如何用计算机语言表达系统设计？	源程序清单
软件测试	在设计测试用例的基础上，检验软件各个组成部分是否正确？	测试文档
运行维护	已交付的软件能否正确运行维护？	改正性维护、适应性维护、完善性维护、预防性维护

4．各个开发阶段之间的交接——开发文档

软件并非只包括可以在计算机上运行的程序，与这些程序相关的文件一般也被认为是软件的一部分。软件文档是软件工程实施中的重要成分，它不仅是软件开发各阶段的重要依据，而且也影响软件的可维护性。

各个开发阶段之间的交接实际上就是文档与文档之间的交接。换句话说，每个开发阶段

都应该产生相应的开发文档。文档是开发团队中技术人员的交流工具，可以快速地找到相关的位置，避免重复阅读程序。

一份合格的开发文档应该全面，要有条理地诠释软件开发流程，要包含技术难点的解释、逻辑判断、重要算法，要包括部分业务流程的说明。技术人员通过看文档可迅速了解程序的流程、调用关系，测试人员可以知道开发人员对用户需求了解的程度、是否有偏差，以快速定位错误，提高开发效率和开发质量。

5．软件的开发方法

1）开发方法的分类

（1）功能分解法——计算任务。

（2）结构化法——以数据为中心。

（3）面向对象法——以对象为中心。

（4）组件法——以组件为中心。

2）结构化系统开发方法（Structured System Development Methodology）

结构化系统开发方法是由 E.Yourdon 和 L.L.Constantine 提出的，首先对软件进行需求分析，然后进行总体设计，最后结构化编程（SP）。

按照程序设计方法的发展，程序设计方法可以分为：

（1）功能分解法，以计算任务为中心。

（2）结构化程序设计，以数据为中心。

（3）面向对象程序设计，以对象为中心。

（4）组件程序设计，以组件为中心。

初学者更容易理解结构化（模块化）的理念，即自顶向下对系统进行分析与设计。这种开发方法能产生清晰、易懂的程序代码，使用程序易于维护。

按用户至上的原则，将系统开发过程划分为若干个相对独立的阶段，然后再考虑局部实现。实施阶段坚持自底向上地逐步实施，即从最基层的模块做起（编程），然后按照系统设计的结构，将模块一个个拼接到一起进行调试，自底向上、逐步地构成整个系统。

引导文献

软件工程是随着软件的发展而诞生的学科。软件工程由程序、数据和文档组成，其中程序是主体，数据是使程序能正常操纵信息的数据结构，文档是与软件的开发、维护和使用有关的材料。软件开发方法应规范化。

1．软件开发的主要环节

软件开发的主要环节如图 1.1 所示。

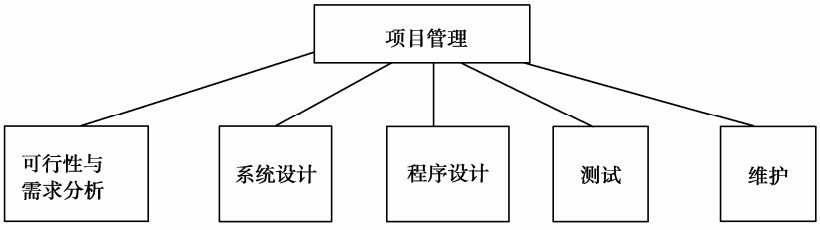


图 1.1　软件开发的主要环节

2．软件开发的策略与步骤

软件开发有三种基本策略：“复用”、“分而治之”、“优化—折中”。

（1）复用是指“利用现成的东西”。

（2）分而治之是指把一个复杂的问题分解成若干个简单的问题，然后逐个解决，如图 1.2 所示。

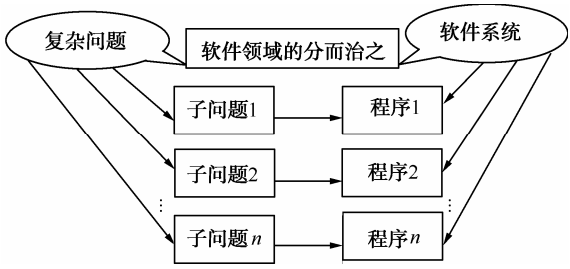


图 1.2　分而治之

软件人员在分而治之时应考虑：问题分解后能否用程序实现？程序能否集成为一个软件系统以解决问题？

（3）优化—折中。优化工作不是可有可无的事情，而是必须要做的事情。当优化工作成为一种责任时，程序员才会不断改进软件中的算法、数据结构和程序组织，从而提高软件质量。折中策略是指通过协调各个质量因素，实现整体质量的最优。

3．软件开发模型

软件开发模型是软件开发全部过程、活动和任务的结构框架。最早出现的软件开发模型是于 1970 年由 W. Royce 提出的瀑布模型，而后随着软件工程学科的发展和软件开发的实践，相继提出了演化模型、螺旋模型、增量模型、喷泉模型、智能模型等。

软件也有一个孕育、诞生、成长、成熟、衰亡的生存过程。根据这一思想，可以得到软件生命周期的 6 个开发阶段，即制订计划、需求分析、设计、程序编码、测试、运行与维护。

软件生命周期模型是从软件项目需求定义直至软件经使用后废弃为止，跨越整个生命周期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。

（1）瀑布模型：制订计划，进行需求分析和说明、软件设计、程序编码、软件测试、运行与维护，固定次序、自上而下、相互衔接。瀑布模型如图 1.3 所示。

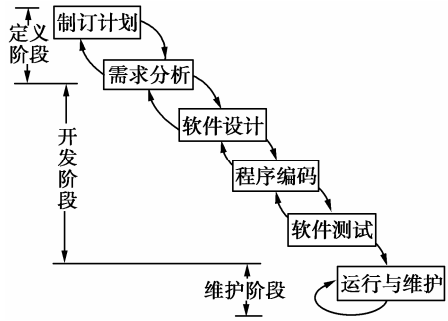


图 1.3　瀑布模型

（2）演化模型：在项目开发的初始阶段，对软件的需求认识不够清晰，可以先做试验开

发与探索可行性，弄清软件需求；在此基础上获得较为满意的软件产品。第一次得到的试验性产品称为“原型”。

（3）螺旋模型：开发复杂的大型软件时，结合瀑布模型与演化模型沿着螺线旋转，自内原型处向外拓展，每旋转一圈便更新完善一个新的版本。螺旋模型如图 1.4 所示。

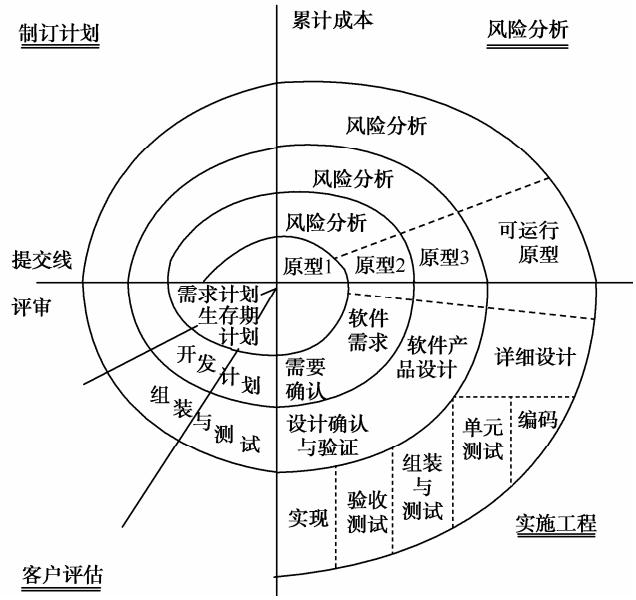


图 1.4　螺旋模型

① 制订计划——确定软件目标，选定实施方案，弄清项目开发的限制条件。

② 风险分析——分析所选方案，考虑如何识别和消除风险。

③ 实施工程——实施软件开发。测试验收产品。

④ 客户评估——评价开发工作，提出修正建议。

（4）喷泉模型：主要支持面向对象的开发方法。分析、设计和编码之间不存在明显的边界。喷泉模型如图 1.5 所示。

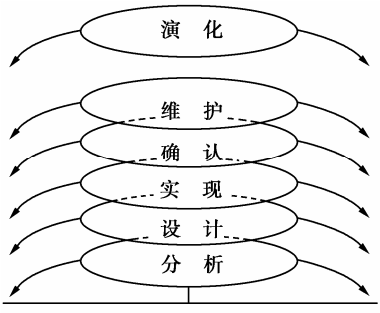


图 1.5　喷泉模型

（5）智能模型：基于知识的软件开发模型，综合以上模型，结合专家系统，采用归约和推理机制，帮助软件人员完成开发工作。

即时训练

1．软件工程实现的目标

（1）付出较低的开发成本。

（2）实现要求的软件功能。

（3）获取较好的软件性能。

（4）提高软件的可移植性。

（5）降低系统的维护费用。

（6）按时交付软件的使用。

2．软件工程工作的内容

（1）分阶段实施系统的开发。

（2）控制调整开发的进度。

（3）控制优化资金的使用。

（4）提交阶段性文档。

拓展任务

（1）较常见应用软件如下。

① 行业管理软件：如计算机行业管理软件开龙 IT200 商软 ERP 等。

② 文字处理软件：如 Office、WPS 等。

③ 信息管理软件：如 Access 数据库。

④ 辅助设计软件：如 AutoCAD、Photoshop。

⑤ 媒体播放软件：如暴风影音、豪杰超级解霸、Windows Media Player、RealPlayer 等。

⑥ 系统优化软件：如 Windows 优化大师、超级兔子魔法设置。

⑦ 图形图像软件：Coreldraw、Painter、MAX，MAYA 等。

⑧ 数学软件：Mathematica、Maple、Matlab、MathCad 等。

⑨ 统计软件：SAS、SPSS 等。

⑩ 后期合成软件：After Effects、Combustion、Digital Fusion、Shake、Flame 等。

⑪ 杀毒软件：如瑞星、金山毒霸、卡巴斯基、江民等。

（2）列举出更多的软件并归类。

1.2 任务二：结构化程序设计

程序设计是软件开发的核心，是指处理事情的先后次序。在计算机语言中，程序定义为完成特定任务的计算机指令的集合。程序设计的方法有多种，本书选用最容易理解的结构化设计方法。

任务描述

工欲善其事，必先利其器！开发软件必须树立正确的开发观念和基本的程序开发技巧，为今后专业化开发打好基础。

任务分析与设计

了解结构化程序设计的理念在软件开发中的重要地位，掌握结构化程序设计的精髓和技巧，选择最恰当的入门语言。

任务实现

1．结构化设计方法（Structured Design，SD）

从系统设计的角度出发，软件设计方法可以分为三大类。第一类是根据系统的数据流进行设计，称为面向数据流的设计或者过程驱动的设计，以结构化设计方法为代表。第二类是根据系统的数据结构进行设计，称为面向数据结构的设计或者数据驱动的设计，以 LCP（程序逻辑构造）方法、Jackson 系统开发方法和数据结构化系统开发（DSSD）方法为代表。第三类是面向对象的设计。

结构化设计方法是在模块化、自顶向下细化、结构化程序设计等程序设计技术的基础上发展起来的。该方法实施的要点是：建立数据流的类型，指明流的边界，将数据流图映射到程序结构，用“因子化”方法定义控制的层次结构，用设计测量和一些启发式规则对结构进行细化。

2．软件设计的原则

1）抽象化

对软件进行模块设计时，可以有不同的抽象层次。在最高的抽象层次上，可以使用问题所处环境的语言描述问题的解法；而在较低的抽象层次上，则采用过程化的方法。

（1）过程的抽象：在软件工程过程中，从系统定义到实现，每进展一步都可以看做是对软件解决方案的抽象化过程的一次细化。在软件计划阶段，软件被当做整个计算机系统中的一个元素来看待。在软件需求分析阶段，用“问题所处环境的为大家所熟悉的术语”来描述软件的解决方法。在从概要设计到详细设计的过程中，抽象化的层次逐次降低。当产生源程序时，到达最低的抽象层次。

（2）数据抽象：数据抽象与过程抽象一样，允许设计人员在不同层次上描述数据对象的细节。例如，可以定义一个数据对象 student，并将它规定为一个抽象的数据类型，用它的构成元素来定义它的内部细节。

student 的抽象：student { name；age；sex；}

此时，数据抽象 student 本身由另外一些数据“name（名字）、age（年龄）、sex（性别）”抽象构成。在定义 student 为一个抽象的数据类型之后，可以引用它来定义其他用数据对象，如“Rose，John”，而不必涉及 student 的内部细节。

使用 student 定义其他数据对象：student Rose，John。

（3）控制抽象：与过程抽象和数据抽象一样，控制抽象可以包含一个程序控制机制而无须规定其内部细节。控制抽象的例子就是在操作系统中用以协调某些活动的同步信号。

2）自顶向下，逐步细化

将软件的体系结构按自顶向下方式，对各个层次的过程细节和数据细节逐步细化，直到用程序设计语言的语句能够实现为止，从而最后确立整个的体系结构。最初的说明只是概念性地描述了系统的功能或信息，但并未提供有关功能的内部实现机制或有关信息的内部结构的任何信息。设计人员对初始说明仔细推敲，进行功能细化或信息细化，给出实现的细节，

划分出若干成分。然后再对这些成分施行同样的细化工作。随着细化工作的逐步展开，设计人员就能得到越来越多的细节。

3）模块化

模块又称构件，是能够单独命名并独立地完成一定功能的程序语句的集合。例如，高级语言中的过程、函数、子程序等都可作为模块。

模块化是软件的一个重要属性。模块化的特性提供了人们处理复杂问题的一种方法，同时也使得软件能够被有效地管理。

软件系统的层次结构正是模块化的具体体现。也就是说，整个软件被划分成若干单独命名和可编址的部分，称为模块。这些模块可以被组装起来以满足整个问题的需求。

模块和其他用模块之间的接口应尽可能独立。

（1）系统设计通常分为结构设计和过程设计两个阶段。

① 结构设计：确定系统由哪些模块组成，以及这些模块之间的相互关系。结构设计是总体设计阶段的任务。

② 过程设计：确定每个模块的处理过程。过程设计是详细设计阶段的任务。

（2）设计的后处理：在确定系统的软件结构以后，还必须做好下述工作。

① 为每个模块开发一份功能说明。

② 为每个模块提供一份接口说明。

③ 定义局部的和全程的数据结构。

④ 给出所有的设计限制或约束。

⑤ 进行总体设计评审。

⑥ 如果需要和可能，则进行设计“优化”。

（3）模块的独立性与信息隐蔽原理。每个模块的实现细节（过程和数据）对于其他模块来说应该是隐蔽的，即包含在模块中的信息（过程或数据）对于其他不需要这些信息的模块来说，是不能访问的，或者是“不可见”的。

有效的模块化可以通过定义来实现，模块间的通信对于软件功能来说是必要的。通过信息隐蔽，可以实施对模块的过程细节和局部数据结构的限制。

模块的独立性是软件质量的关键：模块化程度较高的软件容易开发，模块化程度较高的软件也比较容易测试和维护。

4）控制层次

控制层次也称为程序结构，它表明了程序构件（模块）的组织情况。控制层次往往用程序的层次（树形或网状）结构来表示。

（1）程序结构的深度：程序结构的层次数称为结构的深度，反映了程序的规模和复杂程度。

（2）程序结构的宽度：同层次结构中的最大模块个数称为结构的宽度。

（3）模块的扇入和扇出：扇出表示一个模块直接调用（或控制）的其他用模块数目。扇入定义为调用（或控制）一个给定模块的模块个数。多扇出意味着需要控制和协调许多下属模块。多扇入的模块通常是公用模块。

5）结构的划分

程序结构可以按水平方向或垂直方向进行划分。

（1）水平划分按程序功能定义模块各分支。顶层模块是控制模块，用来协调程序各个功

能之间的通信和运行；其下级模块的最简单的水平划分方法是建立三个分支：输入、处理（数据变换）和输出。优点是：主要的功能相互分离，易于修改、扩充。缺点是：模块接口传递更多的数据，程序流的整体控制复杂。

（2）垂直划分也称为因子划分。工作自顶向下逐层分布：顶层模块执行控制功能，而低层模块完成实际输入、计算和输出操作。优点是：低层模块的修改副作用小，程序控制结构修改的可能性小，便于维护。

6）数据结构

数据结构是数据元素间逻辑关系的表示。典型的数据种类是有限的，它们是构成一些更复杂结构的基本构件块。

7）软件过程

程序结构描述了整个程序的控制层次关系和各个部分的接口情况，而图 1.6 所示的模块 A 的软件过程则着重描述各个模块的处理细节。

软件过程必须提供详细的操作说明，如事件的顺序、正确的判定点、重复的操作等。程序结构与软件过程有关。模块的处理必须指明其上下级环境。软件过程也是层次化的。

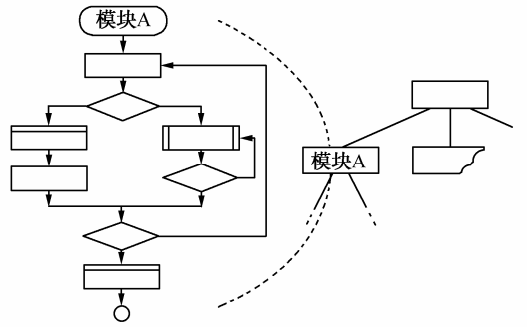


图 1.6　模块 A 的软件过程

3．编程语言

编程是一种行为，是根据用户制定的步骤用“程序语言”表示给计算机，“指挥”计算机完成任务。

算法是程序的逻辑描述，结构是逻辑的数据实体。设计出算法、描述出结构程序就有了雏形。

1）程序设计语言的分类

程序设计语言分为以下 3 类。

（1）机器语言。

（2）汇编语言。

（3）高级语言。

2）语言的选择

从描述客观系统的角度看，程序设计语言还可以分为：

（1）面向过程语言。

数据结构+算法　　（首推 C 语言）

（2）面向对象语言。

对象+消息　　（代表为 C++、Java 语言）

语言只是工具，编程语言的底层都是相似的，其差别主要是适用范围。

由于 C 语言是结构化的程序语言，适用于自顶向下的软件开发方法，在编码之前能帮助用户了解程序整体结构和操作，适合解决数据处理领域的实际问题。我们选择结构严谨、使用灵活的 C 语言作为我们的开发语言。该方法采自顶向下、逐步细化的设计方法，主要有确定问题、分析、设计算法、实现、测试、维护 6 个步骤。

引导文献

1．程序类型

源程序：用户编写的程序称为源程序。

目标程序：源程序通过翻译形成目标程序（.OBJ 文件）。

运行程序：将目标程序与函数库连接后，形成运行程序（.EXE 文件）。

程序设计：指用户通过编写源程序，翻译源程序作为目标程序，连接目标程序与函数库，形成运行程序并整理设计文档的全过程。

2．工程

把代码写到一个文件里，会使查找和修改不再麻烦，单元（Unit）文件是一个不错的选择。

3．模块与函数

要把大一些的任务分成小块，再分成更小的块，一个模块对应一个单元（函数）。这样的分工可以分散源代码，多人同步完成各个单元的代码，这就是结构化编程。在 C 语言中，各个模块一般体现为“函数”。函数是 C 语言中最基本的编程单位。

4．程序的设计风格

1）程序内部的文档

程序内部的文档包括标志符（变量和标号）、程序的注释、程序的视觉组织。

2）标志符

标志符包括模块名、变量名、常量名、标号名、子程序名以及数据区名、缓冲区名等。这些名字应含义鲜明、能正确地提示代表的实体且符合语言的命名规范。

3）程序的注释

程序的注释分为两种：序言性注释和功能性注释。

（1）序言性注释通常安排在每个程序模块的起始部分，它是对程序的整体说明。

（2）功能性注释嵌入在源程序体内，用以描述其后的语句或程序段的处理功能。

4）程序的视觉组织

程序中代码的布局对程序的可读性有很大的影响。适当地利用空格、空行和移行能使程序的逻辑结构更加清晰。空格的合理应用还可以突出运算的优先性，避免发生运算错误。

5）数据说明

为了使数据更容易理解和维护，数据说明应遵循一些简单、规范的原则。

6）输入/输出

以交互式输入/输出方式为主。对所有输入数据都进行校验，要指明可选择的值或界限，以保证每个数据的有效性/合法性；为输出加注释，并设计输出格式。

7）效率

效率是一个性能要求，应该在需求分析阶段提出要求；好的设计可提高效率；程序的效

率和程序的简明程度是一致的。

5．概要设计与详细设计

软件设计是一个把软件需求变换成软件表示的过程。最初，这种表示只是描绘出软件的总框架，然后进一步细化，把它加工成在程序细节上非常接近于源程序的软件表示。从工程管理的角度来看，软件设计分两步完成：

（1）概要设计，将软件需求转化为数据结构和软件的系统结构。

（2）详细设计，即过程设计。通过对结构表示进行细化，得到软件的详细的数据结构和算法。

概要设计建立起整个系统的体系结构框架，并给出了系统中的全局数据结构和数据库接口，人机接口，与其他硬、软件连接的接口。此外，还从系统全局的角度，考虑处理方式、运行方式、容错方式以及系统维护等方面的问题，并给出了度量和评价软件质量的方法。它奠定了整个系统实现的基础。若没有概要设计，直接考虑程序设计，则不能从全局把握软件系统的结构和质量，实现活动处于一种无序状态，程序结构划分不合理，导致系统处于一种不稳定的状态，稍做改动就会失败。因此，不能没有概要设计。

6．完成良好的软件设计应遵循的原则

软件设计既是过程又是模型。设计过程是一系列迭代的步骤，使设计人员能够描述被开发软件的方方面面。设计模型体现了自顶向下、逐步细化的思想，首先构造事物的整体；然后逐步细化，引导设计人员构造各种细节。为了给软件设计人员提供一些指导，Davis 于 1995 年提出如下一系列软件设计的原则，其中有些已被修改和补充。

（1）设计过程不应受“隧道视野”的限制。一位好的设计者应当考虑一些替代的手段。根据问题的要求，可以用基本的设计概念，如抽象、逐步求精、模块化、软件体系结构、控制层次、结构分解、数据结构、软件过程、信息隐蔽等，来决定完成工作的资源。

（2）设计应能追溯到分析模型。由于设计模型中的每一单个成分常常可追溯到多个需求上，因此有必要对设计模型如何满足需求进行追踪。

（3）设计不应当从头做起。系统是使用一系列设计模式构造起来的，很多模式很可能以前就遇到过。这些模式通常被称为可复用的设计构件。可以使用它们代替那些从头开始做的方法。时间短暂而资源有限！将应当设计时间投入到表示真正的新思想和集成那些已有的设计模式上。

（4）设计应当缩短软件和现实世界中问题的“智力差距”，也就是说，软件设计的结果应尽可能模拟问题领域的结构。

（5）设计应具有一致性和集成性。如果一个设计从整体上看像是一个人完成的，那么它就是一致的。在设计工作开始之前，设计小组应当定义风格和格式的规则。如果仔细定义了设计构件之间的接口，则该设计就是集成的。

（6）使用上述基本的设计概念，将设计构造得便于将来的修改。

（7）应将设计构造得即使遇到异常的数据、事件或操作条件，也能平滑地、轻松地降级。设计良好的计算机程序应当永不“彻底停工”，它应能适应异常的条件，并且当它必须中止处理时也能以从容的方式结束。

（8）设计不是编码，编码也不是设计。即使在建立程序构件的详细的过程设计时，设计模型的抽象级别也比源代码高。在编码级别上做出的唯一设计决策是描述如何将过程性设计转换为程序代码的小的实现细节。

（9）在开始着手设计时，就应当能够评估质量，而不是在事情完成之后。利用上述基本的设计概念和已有的设计方法，可以帮助设计者评估质量。

（10）应当坚持设计评审以减少概念上（语义上）的错误。有时，人们在设计评审时倾向于注重细节，只见树木不见森林。在关注设计模型的语法之前，设计者应能确保设计的主要概念上的成分（遗漏、含糊、不一致）都已检查过。

第 2 章　小学生数学选题系统

2.1　任务一：“小学生数学选题系统”的项目背景

任务描述

以小组为单位，对小学生系统项目背景进行分析，弄清楚该项目主要实现哪些功能。

任务分析与设计

软件项目开发的首要条件是，对开发对象的背景有清楚的了解。一般是以软件系统与企业的关系以及市场状况、调查统计作为项目背景描述依据，说明在什么环境下进行项目开发。可参考下列内容进行分析：

（1）开发人员理解的对项目的设想。

（2）企业的外部环境（如生态、文化、社会、心理、组织、技术、道德等方面）。

（3）项目发起人的情况（如姓名和地址、提供资金的可能性、在项目中的作用等）。

（4）支持该项目的经济政策和其他相关政策。

（5）具体的项目介绍。

任务实现

1．编写目的

编写项目背景，使开发人员一目了然该项目的主要功能，为以后的功能实现奠定基础。

2．项目背景文档

“小学生数学选题系统”的项目背景文档如表 2.1 所示。

表 2.1　“小学生数学选题系统”的项目背景文档

软件系统的名称	小学生数学选题系统
本项目的任务提出者	软件技术专业指导老师
用户	学生、教师、家长
开发者	软件技术专业开发小组
项目背景描述	满足小学生的四则运算（1 位或 2 位整数）的练习和考试需要，更能提高学生的算术能力，代替老师和家长对答题过程进行辅导，可机动设置题型、题量，可直接进行判卷和结果分析

引导文献

1．软件项目开发中项目背景描述的重要性

正如做策划案时要做 SWOT 分析一样，开发文档也要进行项目背景介绍。背景介绍实际

上就是开发这个软件的原因及要达到的目标。我们可以从原系统存在的问题、现实工作中存在而又可用该软件来解决的问题等方面入手，推导出要达到设计的目标。这就如同大厦的基石一样，后面的操作都是在这个基础上持续进行的，它影响到软件的整个开发运作过程。必须做到准确、真实、明晰。

通过背景分析，可以清晰地了解开发该软件的原因、存在的问题、要达到的目标、由此分析出的需求。此外，在背景分析中应明确该软件首先要实现的功能、成本的投入等，并依据经济、务实的原则，实现效益最大化。

成功地完成一个项目计划书，同其他任何工作一样，都需要深思熟虑的准备、有效的策略和清晰的计划。项目计划书可以是一个机构的内部文件，用来向董事会、理事会汇报并希望得到他们的批准与支持；也可以是机构就某一项目寻求资金上支持的对外筹款计划书。下面介绍的内容针对后一种情况。

2．项目背景的编写

在项目计划中，这一部分需要详细介绍存在的问题以及为什么要设计这个项目以解决这些问题。要充分地说明问题的严重性与紧迫性，最好能提供一些数据，这样不但可以充分地说明问题，同时还能表明你对这一项目的了解。此外，还可以引用一些相似的、真实的、典型的案例，进而引起共鸣。要说明项目的起因、逻辑上的因果关系、受益群体及其他用社会问题之间的关联等。

（1）一般来讲，这一部分包括以下主要信息：

① 项目范围（问题与事件、受益群体）。

② 导致项目产生的宏观与社会环境。

③ 提出这个项目的理由与原因。

（2）受益群体。在这一部分中，要对项目的收益群体做一个更加详细的描述。必要时，还可以把受益群体分为直接受益和间接受益群体。

分析彩票管理系统的背景。

为彩票管理系统填写背景文档。

2.2 任务二：“小学生数学选题系统”的需求分析

2.2.1 子任务一：编写项目计划书

以小组为单位，编写小学生数学选题系统项目计划书。

项目计划书是为了说明该项目计划的目的并指出预期的读者；其作用是为了说明本文档的意图和希望达到的效果；其意义是使项目相关人员了解项目开发计划书的作用、希望达到

的效果。开发计划书的作用一般都是实现项目成员以及相关人员之间的共识与约定，是项目生命周期所有活动的行动基础，项目团队根据本计划书展开工作。

任务实现

1．编写目的

为了及时、保质完成小学生数学选题系统的开发，便于项目团队成员更好地了解项目情况，使项目工作开展的各个过程合理有序，以文件化的形式，对工作任务进行分解，对相关内容以书面形式进行安排，作为项目开发过程中的共同和行动基础。

2．项目概述

1）工作内容

简要地说明在本项目的开发过程中必须进行的各项工作。

（1）组织开发小组对小学生数学选题系统的背景进行分析，了解原有工作流程。

（2）系统开发小组设计实施方案和开发流程。

（3）系统开发小组对系统进行集中开发。

（4）系统审核小组对系统进行评估、测试和审核。

（5）系统维护小组对系统进行定期维护。

2）参加人员

扼要说明参加本项目开发工作的人员的情况，包括他们的技术水平、主要经历等。

（1）项目组组长：有一定的责任心和很强的团队意识，有一定的表达能力、善于调动和管理小组成员。

（2）项目开发小组成员：具备一定的软件项目开发知识和使用相关软件的经验，了解软件的使用和操作流程，掌握 VC++6.0 开发环境。

3）产品

逐项说明本项目的预期开发成果，包括提交给用户的程序、文件和服务，以及应向本单位提交但不需向用户提交的程序和文件。

（1）小学生数学选题系统的安装包。

（2）登录系统的密码。

（3）用户使用说明书。

（4）C 源程序。

（5）相关开发文档。

4）服务

列出需向用户提供的各项服务，如培训、安装、维护和运行支持等，应逐项规定开始日期、所提供支持的级别和服务的期限。

由各项目组负责人根据自己的实际情况填写表 2.2。

表 2.2　服务项目列表

序　号	任　务	日　期	级　别	期　限
1	培训			
2	安装			
3	维护			
4	运行支持			

5）完成期限

本项目的最迟完成期限为 20 天。

6）本计划的批准者和批准日期

（1）项目的批准者：软件技术专业负责人。

（2）批准日期：2008 年 9 月 10 日。

3．项目实施计划

1）工作任务的分解与人员分工

对于项目开发中需完成的各项工作，从需求分析、设计、实现、测试到维护，包括文件的编制、审批、打印、分发工作，用户培训工作，软件安装工作等，按层次进行分解，指明每项任务的负责人和参加人员。为清晰起见，尽量采用表格的方式。

2）进度

对需求分析、设计、编码实现、测试、移交、培训和安装等工作，给出每项工作的预定开始日期、完成日期，规定各项工作任务完成的先后顺序，以及表明每项工作完成的标志性事件（即所谓的“里程碑”）。

开发过程的主要里程碑有：

（1）项目立项。

（2）需求调研结束。

（3）需求分析结束。

（4）概要设计结束。

（5）详细设计结束。

（6）编码结束。

（7）系统联调结束。

（8）系统测试结束。

（9）系统试运行结束。

（10）系统维护结束。

项目实施计划如表 2.3（各组负责人负责填写）所示。

表 2.3　项目实施计划

序　　号	负责人及所需资源	应提交成果	完 成 工 作	起止时间点	里　程　碑
1	组长	熟悉背景、拟定开发计划			
2	组员	熟悉需求分析文档			
3	组员	熟悉系统设计文档			
4	组长	编码实现			
5	组员	系统测试与维护			
6	组员	文档编写（用户操作手册）			

4．配置管理

项目开发各阶段的交付项，包括各种文档和代码，组成软件的配置。配置管理规定如何管理这些交付项。开发组会产生大量的交付项，而且由于不断地修改，每个交付项又有多个版本。如何从这些交付项建立最终系统，并保证用来生成最终系统的各交付项的一致，是配

置管理的主要任务。

每个项目组，由专门人员负责项目开发交付项的管理工作，确保系统准时交付。

5．预算

逐项列出本项目所需要的劳务和经费的预算及来源。

本项目由学生自筹经费完成。

6．关键问题

逐项列出能够影响整个项目成败的关键问题、技术难点和风险，指出这些问题对项目的影响。

（1）处理登录与主操作界面的关系。

（2）处理评分模块与判卷模块的关系。

（3）处理出题模块与题材量的变化的关系。

（4）软、硬件条件如下。

① 软件条件：VC++6.0 开发环境。

② 机器最低配置：Pentium III 450MHz 以上的 CPU 处理器，64MB 以上的内存，200MB 的自由硬盘空间，能支持 24 位真彩色的显示卡、彩色显示器。

1．需求分析

1）需求分析的目标和任务

需求分析就是分析软件用户的需求是什么。如果投入大量的人力、物力、财力、时间，开发出的软件却没人要，则所有的投入都是徒劳。如果费了很大的精力，开发一个软件，最后却不满足用户的要求，而要重新开发，则这种返工是让人痛心疾首的。

需求分析之所以重要，是因为它具有决策性、方向性、策略性的作用，它在软件开发的过程中具有举足轻重的地位。必须对需求分析具有足够的重视。在一个大型软件系统的开发中，它的作用远远大于程序设计。

需求分析的目标是深入描述软件的功能和性能，确定软件设计的约束和软件与其他系统元素的接口细节，定义软件的其他有效性需求。

需求分析究的对象是开发项目的用户及其要求。要全面理解用户的各项要求，但不可能实现所有的用户需求，最后要清晰、准确地表达需实现的用户要求，这是软件设计的基础。

软件开发项目的目标是实现系统的物理模型，解决目标系统的“做什么”的问题。系统目标模型的实现步骤如图 2.1 所示。

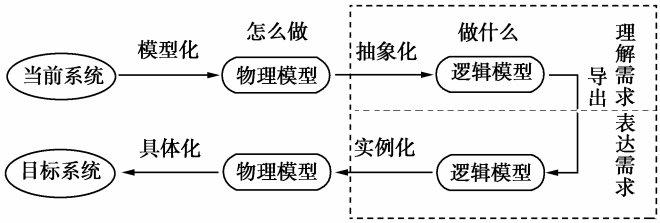


图 2.1　系统目标模型的实现步骤

2）需求分析的工作流程

（1）识别“问题”。系统分析人员综合确定对开发软件的需求，提出这些需求的实现条件和标准，如功能、性能、环境、可靠性、安全性、用户界面、资源及相应的开发进度等需求，可预先估算系统能达到的目标。同时，要确定质量控制标准、里程碑、评审验收标准及可维护性方面的需求。

（2）综合分析。分析人员应该从信息的处理入手，理清信息的结构脉络，逐渐细化软件功能，准备好封装的接口，找出设计所受的限制。对用户提交的需求进行筛选，与用户交流后消除需求误区（即不合理部分），增加真正的需要，最终形成系统的解决方案，给出目标系统的详细逻辑模型。

（3）编制需求分析阶段的文档。

（4）对需求进行分析评审。对功能的正确性、文档的一致性、完备性、准确性和清晰性，以及其他需求给予评价。以评审负责人的结论意见及签字为结束点。

3）获取需求的方法

（1）了解系统的需求。

（2）市场调查。

（3）拜访用户和用户领域的专家。对原始资料进行理解分析，结合专家信息进一步捕获用户需求。

（4）现场考察用户实际的操作环境、操作过程和操作要求。

具体操作有：下发用户调查表、召开调查会、跟踪业务流程、咨询关键岗位人员、查阅开发软件的相关资料。组长可以根据成员的特点进行分工，从多角度同步完成需求的获取。

2．项目目标的设定

设定项目目标就是把项目要完成的工作用清晰的语言描述出来，让项目团队的每一个成员都有明确的概念。注意，不要简单地说成在什么时间完成、开发什么系统或完成什么安装。“要完成一个系统”只是一个模糊的目标，还不够具体和明确。明确的项目目标应该指出服务对象，所开发系统的最主要功能和系统本身的比较深层次的社会目的或系统使用后所产生的社会效益。

3．项目计划与质量管理

项目计划是在可行性分析之后制订的，项目计划将作为依据贯穿需求分析、系统设计、程序设计、测试、维护等软件开发的环节。

项目计划要提供合理的进程表，明确参与软件开发人员的任务，保持开发步调一致，共同、及时地完成项目。项目计划是要付诸实施的，不要过于夸张，其重点在“准确”、“翔实”，而非“快速”、“完美”。

事实上，提高质量是软件开发的主要目标。软件开发是一种智力创作活动，仅仅通过执行严格的操作规范并不足以保证软件产品的质量。程序开发人员必须了解软件各个方面的元素，如正确性、性能、易用性、灵活性、可复用性、可理解性等，才能在进行系统设计、程序设计时提高软件质量。因此，软件的高质量实际上是“设计”出来的。

1）项目计划

做项目计划，就要“知己知彼”。确实了解设计项目的规模、难度与时间限制。这样，才可以确定在这个项目中要投入的人力、物力。在这方面，一个有经验的软件开发人员会为项目计划的设计带来很好的参考建议。

项目的时间限制：一种是类似于商业合同的限制，将完成日期写在合同中，延期时，开发方要做出相应的赔偿；另一种是自行开发的软件产品，只确定大致的发行日期并允许有延误，但是如果因此而拖延太久，也会失去商机而造成损失。

项目的资源分为“人”、“可复用的软构件”和“软、硬件环境”，如图 2.2 所示。

人是最有价值的资源。制订项目计划时，一定要确定开发人员，并根据成员的特点和技术进行分工。

可复用的软构件能提高软件的质量与生产效率。还可以使用“拿来”的组件，甚至可以有偿购买专业性较强的软件。

软、硬件环境常常被忽略，但却是必不可少的资源。一般来讲，只要符合项目的开发要求即可。当所开发的项目需要使用特殊的设备时，要事先做好准备。

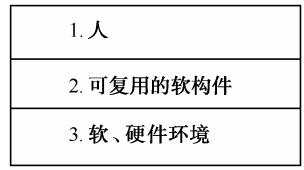


图 2.2　项目的资源

2）进度安排

我们常常会发现项目的开发进度落后了，究其原因不外如下几点：

（1）制定了不现实的期限。项目开发人员按照不合理的进度表开展工作。

（2）客户的需求发生了变化，进度表却没有做出相应的修改。

（3）低估了项目的规模和难度，投入的人力和物力不能满足开发需求。

（4）出现了没有预见到的难以克服的技术难题。

（5）开发人员安排没有冗余，没有考虑开发人员在现实中的意外事件。

（6）开发人员之间的交流出现了障碍，人员之间不能“兼容”和理解，从而导致各阶段任务不能按时完成。

基于以上的事实提出以下建议。

（1）由项目负责人制定进度表，因为他是最了解项目和开发人员的人。最终的进度表要经过开发小组的讨论和大多成员的认定。

（2）应尽可能将技术难度高的事件提前完成，进度安排并非一定要符合逻辑，应发扬中国人的美德——先苦后甜。

（3）开发软件项目，应设立为若干个里程碑。而且，一个里程碑的多个任务可同时进行。里程碑就像心灵的灯塔，指引程序员的设计进度。

（4）进度表中应该保留缓冲时间，可以借鉴 Microsoft 公司的开发小组制定的“50% 缓冲规则”。对许多项目经理而言，能够容忍进度表中存在缓冲时间，是相当不容易的。

（5）如果发现项目应交付的期限不合理时，要据理力争，放宽期限、调整进度。当客户需求发生变化时，要及时变动进度表。

3）项目实现“零缺陷”

高目标：人在做一件事情时，由于存在很多不确定的因素，一般不可能 100% 地达到目标。但是，不设定较高的目标，没有“零缺陷”的质量目标，也许会成堆出现开发陷阱。

4）可执行的规范

好规范的前提是有能力实现并执行。无论多么好的规范，一味地照搬可能会出现硬伤。在书籍中可以很容易找到软件工程的众多规范，但真正适用于当前项目的规范、可以成功执行的规范才是真正的好规范。要把握软件的灵活与严密的尺度。程序员必须深入了解软件多

方面的质量因素，把那些能提高软件质量因素的各种规范植入脑中，才能在各个实践环节自然而然地把高质量设计到软件中。

5）选题检查

质量检查并不是要等到项目结束时执行的。对应进度表，在设定的每个里程碑中进行质量检查并做出评审。

4．项目计划书的编写提纲（完整版）

项目计划书的编写没有定式，只有通用的约定。在编制时，还要注意具体“项目”具体分析。

（1）项目提出的背景和必要性：包括国内外现状、知识产权状况和发展趋势，技术突破对产业技术进步的重要意义和作用，项目可能形成的产业规模和市场前景。

（2）国内外市场分析：包括国际市场状况及该产品未来增长趋势、国际市场的竞争能力、产品替代进口或出口的可能性，国内市场需求规模和产品的发展前景、在国内市场的竞争优势和市场占有率。

（3）项目主要开发和建设内容：包括项目的主要科技攻关内容、项目目标及开发任务。

（4）项目实施的技术方案：包括项目的技术路线、工艺的合理性和成熟性，关键技术的先进性和创新点；产品技术性能水平与国内外同类产品的比较；项目承担单位在实施本项目的优势。

（5）项目实施的现有基础：包括项目承担单位注册地点、股权结构、资产和负债情况、员工构成、主要业务和主要产品、生产规模、主要装备和技术水平、近年来的经营状况，对引进技术的消化、吸收、创新的后续开发能力，企业资质、信用和融资能力等。

（6）项目组织机构和人员安排：包括项目的组织形式、产学研联盟运作机制及分工安排，项目的实施地点，项目承担单位负责人、项目领军人物的主要情况，项目开发的人员安排。

（7）项目实施进度计划：包括项目阶段考核指标（含主要技术经济指标，可能取得的专利，尤其是发明专利和国外专利情况）及时间节点安排，项目的验收指标。

（8）项目资金需求及来源：包括项目新增总投资估算、资金筹措方案（含自有资金、银行贷款、科教兴市专项资金、推进部门配套资金等）、投资使用计划。

（9）项目经济和社会效益分析：包括项目未来三年或五年的生产成本、销售收入和利税估算，财务内部收益率、投资回收期、投资利润率、财务净现值等指标的动态财务分析，社会效益分析。

（10）项目风险分析及应对措施：包括项目技术、市场、资金等风险分析及应对措施。

2.2.2　子任务二：编写需求规格说明书

在完成了针对“小学生数学选题系统”软件市场的前期调查，以及与客户进行了全面、深入的探讨和分析的基础上，制定需求规格说明书。

开发人员针对“小学生数学选题系统”用户的需求进行细致的调研分析，将用户非形式

的需求陈述转化为完整的需求定义，再由需求定义转换为相应的需求文档。

任务实现

1．编写目的

“小学生数学选题系统”需求规格说明书的预期读者为客户、业务或需求分析人员、测试人员、用户文档编写者、项目管理人员。

2．系统概述

1）功能简介

简要描述系统的主要功能，并说明本系统与其他相关系统之间的关系。建议用框图的方式说明系统的组成。

“小学生数学选题系统”主要是对使用者的身份进行验证和对数学试卷的出题、判题、评分进行管理，根据其功能分为登录模块和主控模块。“小学生数学选题系统”流程如图 2.3 所示。

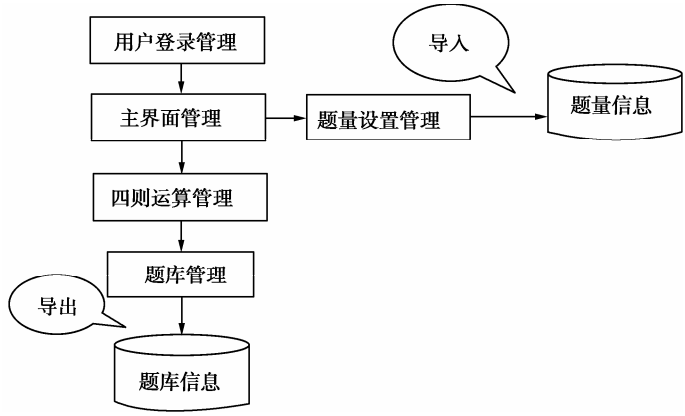


图 2.3　“小学生数学选题系统”流程

2）用户特点

描述本系统的最终用户的特点，说明操作人员、维护人员的技能，以及本系统的使用频度。

本系统的最终用户为学校，操作人员只要掌握最基本的软件操作技能即可。系统维护人员要熟练掌握系统的功能与使用方法。系统的使用频度较高。

3）系统运行环境

说明运行该系统所需要的硬件设备。

硬件条件（最低配置）：Pentium III 450MHz 以上的 CPU 处理器，64MB 以上的内存，200MB 的自由硬盘空间，能支持 24 位真彩色的显示卡、彩色显示器。

3．功能模块说明

“小学生数学选题系统”功能模块如图 2.4 所示。

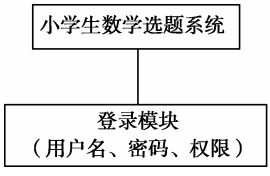


图 2.4　“小学生数学选题系统”功能模块

1）登录模块

（1）概述。本模块体现为用户使用软件的验证功能。

（2）登录模块如表 2.4 所示。

表 2.4　登录模块

输入	用户名、密码
处理	验证用户名和密码的正确性
输出	错误提示或欢迎信息

2）主控模块

（1）概述。主控模块主要由使用说明、题量设置、四则题库、评分系统模块组成，能够完成自动出题、阅卷功能。

（2）使用说明模块如表 2.5 所示。

表 2.5　使用说明模块

输入	无
处理	无
输出	在屏幕上显示小学生数学选题系统说明信息

（3）题量设置模块如表 2.6 所示。

表 2.6　题量设置模块

输入	新题量设置
处理	判断是否为 5 的倍数
输出	当前设置的题量

（4）四则题库模块如表 2.7 所示。

表 2.7　四则题库模块

输入	题型
处理	抽取相应的题型
输出	在屏幕上显示相应的试卷

（5）评分系统模块如表 2.8 所示。

表 2.8　评分系统模块

输入	无
处理	计算得分
输出	在屏幕上显示正误题目数量和得分

1．需求分析应该注意的事项

（1）最好为每个需求注释“为什么”，以便让程序员了解需求的本质，从而选用最合适的

技术来实现此需求。

（2）需求说明不可有二义性，更不能前后相矛盾。如果有二义性或前后相矛盾，则要重新分析此需求。

（3）跳出程序员的逻辑。需求分析和程序设计不尽相同，合理、可行是才是重要的。要跳出程序设计的圈子，站在系统的角度上来看问题。

2．通过什么方式了解需求

（1）直接与客户交谈，“侃”出需求。

（2）用户讲不清楚，分析人员猜不透，就要请教行家。“听君一席话，胜读十年书。”

（3）应避免幼稚需求，分析优秀和蹩脚的同类软件，对优点尽量吸取，对缺点引以为戒。

3．软件需求规格说明及评审

软件需求规格说明是分析任务的最终产物，通过建立完整的信息描述、详细的功能和行为描述、性能需求和设计约束的说明、合适的验收标准，给出对目标软件的各种需求。

作为需求分析阶段工作的复查手段，在需求分析的最后一步，应该对功能的正确性、完整性和清晰性，以及其他需求给予评价。评审的主要内容是：

（1）系统定义的目标是否与用户的要求一致。

（2）系统需求分析阶段提供的文档资料是否齐全。

（3）文档中的所有描述是否完整、清晰、准确反映用户要求。

（4）与所有其他系统成分的重要接口是否都已经描述。

（5）被开发项目的数据流与数据结构是否足够、确定。

（6）所有图表是否清楚，在不补充说明时能否理解。

（7）主要功能是否已包括在规定的软件范围之内，是否都已充分说明。

（8）软件的行为和它必须处理的信息、必须完成的功能是否一致。

（9）设计的约束条件或限制条件是否符合实际。

（10）是否考虑了开发的技术风险。

（11）是否考虑过软件需求的其他方案。

（12）是否考虑过将来可能会提出的软件需求。

（13）是否详细制定了检验标准，它们能否对系统定义是否成功进行确认。

（14）是否有遗漏，重复或不一致的地方。

（15）用户是否审查了初步的用户手册或原型。

（16）软件开发计划中的估算是否受到了影响。

为保证软件需求定义的质量，评审应由专门指定的人员负责，并按规程严格进行。评审结束应有评审负责人的结论意见及签字。除分析员之外，用户/需求者，开发部门的管理者，软件设计、实现、测试的人员都应当参加评审工作。评审的结果一般包括修改意见，待修改完成后再经评审通过，才可进入设计阶段。

以小组为单位，编写自行设计的小学生选题系统的需求说明书。

设计彩票系统的需求说明书。

2.3 任务三：“小学生数学选题系统”的设计

在进行系统设计时，要关注软件的质量因素，如正确性、精确性、性能与效率、易用性、可理解性、简法性、可复用性、可扩充性，等等。

2.3.1 子任务一：编写概要设计说明书

任务描述

（1）根据“小学生数学选题系统”需求分析的结果，从实现的角度进一步划分为模块，并组成模块的层次结构，绘制“小学生数学选题系统”总的模块层次图（H 图）。

（2）根据系统流程图进行功能分解以确定模块结构，划分功能模块，并说明各模块的功能。

（3）确定模块之间的调用关系，绘制函数关系图。

任务分析与设计

（1）绘制功能模块图的关键是对用户需求要有充分的了解。

（2）划分各功能模块，采用自顶向下逐层分解的结构化分析方法。

（3）函数关系图主要是针对各模块而设计的，体现了各函数之间的调用关系，确定了模块之间的接口，即模块之间的传递的信息。

（4）系统设计是把需求转化为软件系统的最重要的环节，包括体系结构、模块、数据与算法和用户界面的设计。

任务实现

1．编写目的

小学生数学选题系统的概要设计是总体设计的第一个阶段，这个阶段的主要任务是对小学生数学选题系统的结构设计，具体为：

（1）采用结构化设计方法，将小学生数学选题系统按功能划分成模块。

（2）确定小学生数学选题系统各模块的功能。

（3）确定小学生数学选题系统各模块之间的调用关系（这里用函数关系图表示）。

（4）确定小学生数学选题系统各模块之间的接口，即模块之间传递的信息。

2.“小学生数学选题系统”总的模块层次图（H图）

“小学生数学选题系统”H 图如图 2.5 所示。

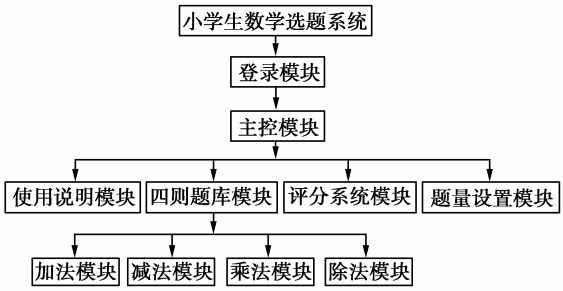


图 2.5 “小学生数学选题系统”H 图

3．功能模块划分及调用关系

功能模块划分及调用关系如表 2.9 所示。

表 2.9 功能模块划分及调用关系

模 块	函 数	模块功能	调用函数	被调用函数
登录模块	login()	控制使用权限	main()	showMainMenu()
主控模块	showMainMenu()	显示选题系统主菜单并根据选择跳转到相应处理模块	login()	systemSetting() selectProblemMenu() gradeSystem() helpInfo()
使用说明模块	helpInfo()	提供帮助信息	showMainMenu()	showMainMenu()
四则题库模块	selectProblemMenu()	选题、出题、判题	showMainMenu()	showMainMenu() addition()加法模块 subtraction()减法/模块 division()乘法模块 multiplication()除法模块
加法模块	addition()	输出相应数量的算式，判断答题正确否和提示信息	selectProblemMenu()	selectProblemMenu()
减法模块	subtraction()	输出相应数量的算式，判断答题正确否和提示信息	selectProblemMenu()	selectProblemMenu()
乘法模块	division()	输出相应数量的算式，判断答题正确否和提示信息	selectProblemMenu()	selectProblemMenu()
除法模块	multiplication()	输出相应数量的算式，判断答题正确否和提示信息	selectProblemMenu()	selectProblemMenu()
题量设置模块	systemSetting()	设置出题的个数	showMainMenu()	showMainMenu()
评分系统模块	gradeSystem()	根据用户答题情况给出成绩	showMainMenu()	showMainMenu()

1．软件设计的过程

一旦确定软件需求之后，就进入开发阶段。开发阶段由三个互相关联的步骤组成：设计、实现（编码）和测试，最后得到有效的计算机软件。

从工程管理的角度来看，软件设计分两步完成：首先做概要设计，将软件需求转化为数据结构和软件的系统结构，并建立接口；然后是详细设计，即过程设计，得到软件的详细的数据结构和算法。

软件设计是开发阶段中最重要的步骤，它是软件开发过程中质量得以保证的关键步骤。设计提供了软件的表示，使得软件的质量评价成为可能。

软件设计是一个把软件需求变换成表示的过程。

1）需求转换为设计时判断设计好坏的三条特征

（1）设计必须实现分析模型中描述的所有显式需求，必须满足用户希望的所有隐式需求。

（2）设计必须是可读、可理解的，使得将来易于编程、易于测试、易于维护。

（3）设计应从实现角度出发，给出与数据、功能、行为相关的软件全貌。

2）设计的技术标准

（1）设计出来的结构应是分层结构，从而建立软件成分之间的控制。

（2）设计应当模块化，从逻辑上将软件划分为完成特定功能或子功能的构件。

（3）设计应当既包含数据抽象，也包含过程抽象。

（4）设计应当建立具有独立功能特征的模块。

（5）设计应当建立能够降低模块与外部环境之间复杂连接的接口。

（6）设计应能根据软件需求分析获取的信息，建立可驱动、可重复的方法。

（7）软件设计过程根据基本的设计原则，使用系统化的方法和完全的设计评审来建立良好的设计。

2．概要设计在开发阶段中的重要性

在软件需求分析阶段已经完全弄清楚了软件的各种需求，较好地解决了要让所开发的软件“做什么”的问题，并已在软件需求规格说明和数据要求规格说明中详尽和充分地阐明了这些需求。下一步就要着手实现软件的需求，即要着手解决“怎么做”的问题。

3．利用HIPO图（Hierarchy Plus Input-Process-Output）进行迭代式细化设计

在软件设计时，解决设计问题通常需要经历一个认识逐步发展的过程，并且对一些问题还要经过反复的考虑才可能达到比较满意的设计效果。我们称此为迭代式细化设计。HIPO 图能很好地适应这一要求。

（1）HIPO 图是系统设计的描述工具。HIPO 图是 IBM 公司于 20 世纪 70 年代，在层次结构图的基础上推出的一种描述系统结构和模块内部处理功能的工具。HIPO 图由两部分组成——H 图和 IPO 图。

① 系统层次结构（H）图——描述整个系统的设计结构以及各类模块之间的关系，如图 2.6 所示。

② IPO 图——描述某个特定模块内部的处理过程和输入/输出关系，即模块间的联系和数据结构（用于“详细设计”部分），如图 2.7 所示。

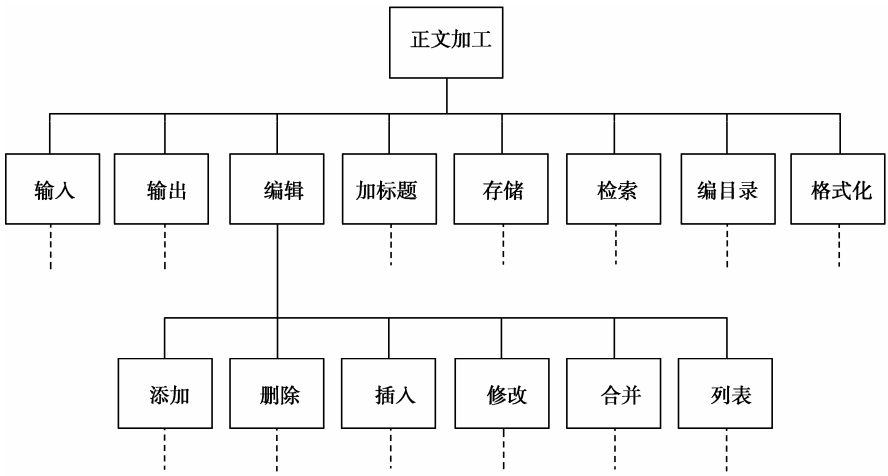


图 2.6　系统层次结构（H）图

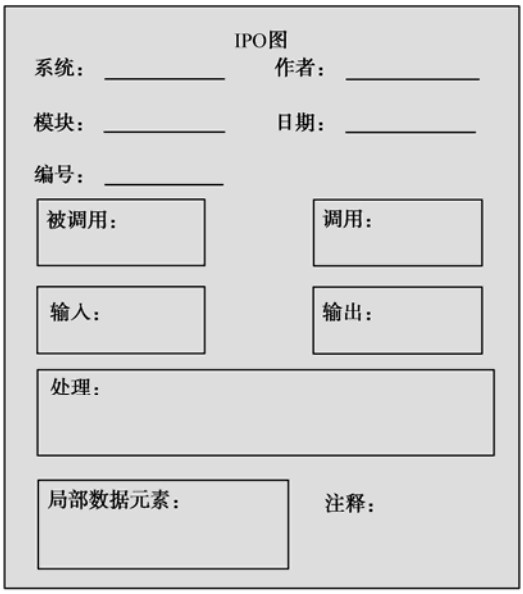


图 2.7　IPO 图

（2）利用 HIPO 图进行迭代式细化设计。把可视目录表和 IPO 图结合起来，反复交替地使用，可使得设计工作逐步深化，最终取得完满的设计结果。其实这正是自顶向下、逐步求精的结构化程序设计思想。

根据系统流程（自顶向下、逐步细化）进行功能分解，以确定模块结构、划分功能模块。

（3）将软件的体系结构采用自顶向下的方式，对各个层次的过程细节和数据细节逐层细化，直到用程序设计语言的语句能够实现为止，从而最后确立整个体系结构。

2.3.2　子任务二：编写详细设计说明书

任务描述

（1）根据“小学生数学选题系统”流程图和 H 图，绘制 IPO 图。

（2）详细设计数据结构。

任务分析与设计

（1）绘制 IPO 图的关键是，要清楚对各功能模块的输入/输出数据、处理功能和调用详细情况。

（2）数据结构设计要详细，为编码实现打下良好的基础。

任务实现

1．编写目的

小学生数学选题系统详细设计是设计的第二个阶段。这个阶段的主要任务是，在小学生数学选题系统概要设计的基础上，对概要设计中产生的功能模块进行过程描述，设计功能模

块的内部细节，包括算法和详细数据结构，为编写源代码提供必要的说明。

2．项目概述

1）模块编号

小学生数学选题系统各模块编号如表 2.10 所示。

表 2.10　小学生数学选题系统各模块编号

序　　号	编　　号	名　　称
1	01	登录模块
2	02	主控模块
3	021	使用说明模块
4	022	四则题库模块
5	0221	加法模块
6	0222	减法模块
7	0223	乘法模块
8	0224	除法模块
9	023	题量设置模块
10	024	评分系统模块

2）各模块IPO表

（1）登录模块 IPO 表如表 2.11 所示。

表 2.11　登录模块 IPO 表

系统：小学数学选题系统	作者：××××
模块：登录模块	日期：××××
模块编号：01	
上层调用模块：小学生数学选题系统	下层被调用模块：主控模块
输入：用户名、密码	输出数据：验证结果是否正确
处理：根据用户所提交的登录信息，验证用户名和密码，共提供三次机会，如前台密码验证成功则进入主控模块，否则提示相关信息退出系统	
局部数据元素：输入的用户名、密码、正确的登录名、密码，允许重复登录的次数	

（2）主控模块 IPO 表如表 2.12 所示。

表 2.12　主控模块 IPO 表

系统：小学数学选题系统	作者：×××
模块：主控模块	日期：××××
模块编号：02	
上层调用模块：登录模块	下层被调用模块： 使用说明模块 四则题库模块 评分系统模块 题量设置模块
输入：用户所需选择的项目 num（0-4）	输出数据：
处理： num=1，题量设置模块 num=2，四则题库模块 num=3，评分系统模块 num=4，使用说明模块 num=0，退出系统	
局部数据元素：int num	

（3）使用说明模块 IPO 表如表 2.13 所示。

表 2.13　使用说明模块 IPO 表

系统：小学生数学选题系统	作者：×××
模块：使用说明模块	日期：××××
模块编号：021	
上层调用模块：主控模块	下层被调用模块：
输入：任意键	输出数据：系统说明信息
处理：在屏幕上显示系统说明信息	
局部数据元素：无	

（4）四则题库模块 IPO 表如表 2.14 所示。

表 2.14　四则题库模块 IPO 表

系统：小学生数学选题系统	作者：×××
模块：四则题库模块	日期：××××
模块编号：022	
上层调用模块：主控模块	下层被调用模块： 加法模块 减法模块 乘法模块 除法模块

续表

输入：用户所需选择的项目 num（0-4）	输出数据：
处理： num=1，加法模块 num=2，减法模块 num=3，乘法模块 num=4，除法模块 num=0，返回主控模块	
局部数据元素：int num	

（5）加法模块 IPO 表如表 2.15 所示。

表 2.15　加法模块 IPO 表

系统：小学生数学选题系统	作者：××××
模块：加法模块	日期：××××
模块编号：0221	
上层调用模块：四则题库模块	下层被调用模块：
输入：答案	输出数据： 提示信息：正确或错误（包括正确答案）
处理：随机出 5 道题（题量设置决定题的数量），判断用户输入的答案的正误码率，给出相应的提示信息	
局部数据元素：int i;//控制量 int num;//判断量 int x=1,y=10;int x1,x2;//操作数 int sumsys,sumuser;	

（6）减法模块 IPO 表如表 2.16 所示。

表 2.16　减法模块 IPO 表

系统：小学生数学选题系统	作者：××××
模块：减法模块	日期：××××
模块编号：0222	
上层调用模块：四则题库模块	下层被调用模块：
输入：答案	输出数据：提示信息为正确或错误（包括正确答案）
处理：如果操作数中被减数 x1 较大，则交换 x1 和 x2 的值，随机出 5 道题（题量设置决定题的数量），判断用户输入的答案的正误码率，给出相应的提示信息	
局部数据元素：int i;//控制量 int num; //判断量 int x=1,y=10;int x1,x2,t;//操作数 int sumsys,sumuser;	

（7）乘法模块 IPO 表如表 2.17 所示。

表 2.17　乘法模块 IPO 表

系统：小学生数学选题系统	作者：××××
模块：乘法模块	日期：××××
模块编号：0223	
上层调用模块：四则题库模块	下层被调用模块：
输入：答案	输出数据： 提示信息为正确或错误（包括正确答案）
处理：随机出 5 道题（题量设置决定题的数量），判断用户输入的答案的正误码率，给出相应的提示信息	
局部数据元素：int i;//控制量 int num;//判断量 int x=1,y=10; int x1,x2;//操作数　int sumsys,sumuser;	

（8）除法模块 IPO 表如表 2.18 所示。

表 2.18　除法模块 IPO 表

系统：小学生数学选题系统	作者：××××
模块：除法模块	日期：××××
模块编号：0224	
上层调用模块：四则题库模块	下层被调用模块：
输入：答案	输出数据：提示信息为正确或错误（包括正确答案）
处理：保证操作数 x1 和 x2 为整除关系，并且除数 x2 不为零，随机出 5 道题（题量设置决定题的数量），判断用户输入的答案的正误码率，给出相应的提示信息	
局部数据元素：int i;//控制量 int num;//判断量 int x=1,y=10; int tx1,x2;//操作数　int sumsys,sumuser;	

（9）题量设置模块 IPO 表如表 2.19 所示。

表 2.19　题量设置模块 IPO 表

系统：小学生数学选题系统	作者：××××
模块：题量设置模块	日期：××××
模块编号：023	
上层调用模块：主控模块	下层被调用模块：
输入：重新设置的题量	输出数据：
处理：保证输入的数据为 5 的倍数，否则重新输入，然后将出题时的题量更新为新设置的数	
局部数据元素：无	

（10）评分系统模块 IPO 表如表 2.20 所示。

表 2.20　评分系统模块 IPO 表

系统：小学生数学选题系统	作者：×××
模块：评分系统模块	日期：××××
模块编号：024	
上层调用模块：主控模块	下层被调用模块：
输入：无	输出数据：答题总数量，对错题目数量和分数
处理：显示答题的总数量、正确数量、错误数量和最后得分	
局部数据元素：int sum,num,tsum,fsum; char choi;	

3．各模块的算法设计

各模块的算法设计如表 2.21 所示。

表 2.21　各模块的算法设计

编　　号	名　　称	主 要 算 法
01	登录模块	循环
02	主控模块	选择（多分支）
021	使用说明模块	顺序
022	四则题库模块	选择（多分支）
0221	加法模块	循环
0222	减法模块	循环
0223	乘法模块	循环
0224	除法模块	循环
023	题量设置模块	顺序
024	评分系统模块	顺序

4．数据结构设计

根据需求分析及概要设计，进一步分析数据结构。题量数目数据表（全局变量）如表 2.22 所示，评分数据表（局部变量）如表 2.23 所示。

表 2.22　题量数目数据表（全局变量）

汉 语 名 称	英 文 名 称	类　　型	长　　度	备　　注
题库默认题量	problemNumber	整型	2	初始值为 5
加法题正确数目	tnum1	整型	2	初始值为 0
加法题错误数目	fnum1	整型	2	初始值为 0
减法题正确数目	tnum2	整型	2	初始值为 0
减法题错误数目	fnum2	整型	2	初始值为 0
乘法题正确数目	tnum3	整型	2	初始值为 0
乘法题错误数目	fnum3	整型	2	初始值为 0
除法题正确数目	tnum4	整型	2	初始值为 0
除法题错误数目	fnum4	整型	2	初始值为 0

表 2.23　评分数据表（局部变量）

汉 语 名 称	英 文 名 称	类　　型	长　　度	数 据 约 定
完成总题量	sum	整型	2	做过的四则题目的数量
正确题目总数	tsum	整型	2	做过的正确四则题目数量
错误题目总数	fsum	整型	2	做过的错误四则题目数量
是否继续操作的控制变量	num	整型	2	约定的变量（相似操作）
是否确认操作的控制变量	choi	字符型	1	约定的变量（相似操作）

1．程序设计原理

（1）模块：又称构件，是能够单独命名并独立地完成一定功能的程序语句的集合。例如，高级语言中的过程、函数、子程序等都可作为模块。

（2）模块化是软件的一个重要属性。模块化的特性提供了人们处理复杂的问题的一种方法，同时也使得软件能够被有效地管理。

（3）模块的独立性是软件质量的关键。

① 模块化程度较高的软件易于开发。

② 模块化程度较高的软件也较易于测试和维护。

2．程序描述与功能描述

功能性注释嵌入在源程序体内，用以描述其后的语句或程序段的处理功能。书写功能性注释，要注意以下几点：

（1）描述的对象是一段程序，而不是每一个语句。

（2）适当使用缩进和空行，使程序与注释容易区别。

（3）注释一定要准确。不精确的或错误的注释不仅对理解程序毫无帮助，反而会妨碍对程序的理解。

3．模块（函数）设计与算法

在设计好软件的结构后，就已经在宏观上明确了各个模块应具有什么功能，应放在体系结构的哪个位置。我们习惯地从功能上划分模块，保持“功能独立”是模块化设计的基本原则。因为“功能独立”的模块可以降低开发、测试、维护等阶段的成本。但是，“功能独立”并不意味着模块之间保持绝对的孤立。一个系统要完成某项任务，需要各个模块相互配合才能实现，此时模块之间就要进行信息交流。

1）设计准则

（1）尽力提高模块独立性。

（2）选择合适的模块规模。

（3）模块的深度、宽度、扇出和扇入应适当。

（4）模块的作用范围应该在控制范围之内。

（5）降低模块接口的复杂程度。

（6）设计单入口、单出口的模块，避免“病态连接”。

2）有效的模块设计

模块化方法带来了许多好处：一方面，模块化设计降低了系统的复杂性，使得系统容易修改；另一方面，推动了系统各个部分的并行开发，从而提高了软件的生产效率。

描述一个模块时，还必须按模块的外部特性与内部特性分别描述。模块的外部特性是指模块的模块名、参数表以及给程序或整个系统造成的影响。模块的内部特性是指完成其功能的程序代码和仅供该模块内部使用的数据。

对于模块的外部环境（如需要调用这个模块的上级模块）来说，只需要了解这个模块的外部特性，不必了解它的内部特性。在软件设计阶段，通常是先确定模块的外部特性，然后再确定它的内部特性。

3）模块独立性

所谓模块独立性，是指软件系统中每个模块只涉及软件要求的具体子功能，而与软件系统中其他模块的接口是简单的。例如，若一个模块只具有单一的功能且与其他用模块没有太多的联系，那么则称此模块具有模块独立性。

4）算法的特性

（1）有穷性：一个算法应包含有限的操作步骤，而不能是无限的。

（2）确定性：算法中每一个步骤应当是确定的，而不能是含糊的、模棱两可的。

（3）有零个或多个输入。

（4）有一个或多个输出。

（5）有效性：算法中每一个步骤应当能有效地执行，并得到确定的结果。

4．需求概述与程序设计结构

按照需求分析文档中的说明要求，使用系统选题并进行答题，统计题数，算出分数使得信息传递准确、流畅。同时，系统最大限度地实现易安装性、易维护性、易操作性、运行稳定、安全可靠性。

设计一个模块时，要考虑“模块的功能”，也要考虑“模块间的信息交流”。

5．软件体系结构

1）软件体系结构类型

（1）数据流系统。这种结构中的每一个组成成分都有一套输入和输出数据，都以输入数据－处理－输出结果的方式工作。进行数据变换的构件叫做过滤器，把数据从一个过滤器的输出导入到另一个过滤器的输入，就叫做管道。

（2）调用－返回系统：

① 主程序/子程序。

② 层次结构。在层次结构中，每一层都只与上下相邻的两层通信。每一层在利用下层基础服务的条件下，为上层提供服务。

2）软件体系结构的三要素

软件设计的目标是建立软件的体系结构表示。将这个表示当做一个框架，从事更详细的设计活动。Shaw 和 Garlan 提出了在软件体系结构设计中应保持的下列性质。

（1）结构：体系结构设计应当定义系统的构件、这些构件打包的方式和交互的方式，如将对象打包以封装数据和操纵数据的处理，并通过相关操作的调用来进行交互。

（2）附属的功能：体系结构设计应当描述设计出来的体系结构如何实现对功能、性能、可靠性、安全性、适应性以及其他系统需求。

（3）可复用：体系结构设计应当描述为一种可复用的模式，以便在以后类似的系统族的

设计中使用它们。此外，设计应能复用体系结构中的构造块。

6.“黑箱”技术

在设计当前模块时，应先把这个模块的所有下层模块定义成“黑箱”，并在系统设计中利用它们，暂时不考虑它们的内部结构和实现方法。在这一步定义好了“黑箱”，由于已确定了它的功能和输入、输出，在下一步就可以对它们进行设计和加工。这样，又会导致更多的“黑箱”。最后，全部“黑箱”的内容和结构应完全被确定。这就是自顶向下、逐步求精的过程。使用黑箱技术的主要好处是，使设计人员可以只关心当前的有关问题，暂时不必考虑进一步的琐碎的次要的细节，待进一步分解时才去关心它们的内部细节与结构。

2.4 任务四：“小学生数学选题系统”的编码实现

2.4.1 子任务一：界面设计

设计小学生数学选题系统界面，根据用户要求确定界面的风格，分组绘制界面草图，编码实现各模块界面的设计。

具体要求如下：

（1）界面清晰明了、各界面之间上下衔接自然。

（2）根据系统概要设计的功能模块图确定各界面之间的关系。

分析小学生数学选题系统共需要几个独立界面。

（1）分别定义各界面的函数。

（2）具体设计各界面的风格。

（3）做好各界面的衔接。

（4）运行调试。

（1）根据系统概要设计，初步分析小学数学选题系统共计 5 个界面：登录界面、主控界面、四则题库界面、题量设置界面和使用说明界面。

（2）根据函数关系图确定各函数名称分别为 login() 、showMainMenu()、selectProblemMenu()、systemSetting()、helpMenu()。

（3）各小组画出界面风格草图。

（4）编码实现。

1．登录界面设计

```
#include <stdio.h>     /*引用预处理命令中的 stdio.h 头文件*/
//登录界面代码
main()                 /*主函数*/
{
```

```
    login();            /*调用 login()函数*/
}
void login()
{
    printf("\n\n");
    printf("\t\t*****************************************************\n");
    printf("\t\t                    选题系统登录                     \n");
    printf("\t\t*****************************************************\n");
    printf("\n\n");
    printf("\t\t\t\t 请输入用户名:" );
    printf("\n\n");
    printf("\t\t\t\t 请输入密码:");
}
```

运行结果如图 2.8 所示。

图 2.8　登录界面

2．主控界面设计

```
//主控界面代码
void showMainMenu()
{
    int num;
    printf("\n\n");
    printf("\t\t*****************************************************\n\n");
    printf("\t\t                小学生数学选题系统                   \n\n");
    printf("\t\t*****************************************************\n\n");
    printf("\n\n");
    printf("\t\t\t\t 1. 题 量 设 置\n\n");
    printf("\t\t\t\t 2. 四 则 题 库\n\n");
    printf("\t\t\t\t 3. 评 分 系 统 \n\n");
    printf("\t\t\t\t 4. 使 用 说 明\n\n");
    printf("\t\t\t\t 0. 退出\n\n\n\n");
    do{  printf("\t\t 请 选 择 操 作(0-4):");
         scanf("%d",&num);
       }while(num>4||num<0);//只能选择 0-4，否则一直要求重新操作
    switch(num)
    {    case 1:systemSetting();break;          //题量设置
```

```
                case 2:selectProblemMenu();break;   //四则题库选择
                case 3: gradeSystem()break;;        //评分系统
                case 4: helpInfo();break;           //使用说明
                case 0:exit(0);break;
    default: printf("\n\t\请按任意键重新选择!!!"); getchar(); system("cls"); showMainMenu();
        }
}
```

运行结果如图 2.9 所示。

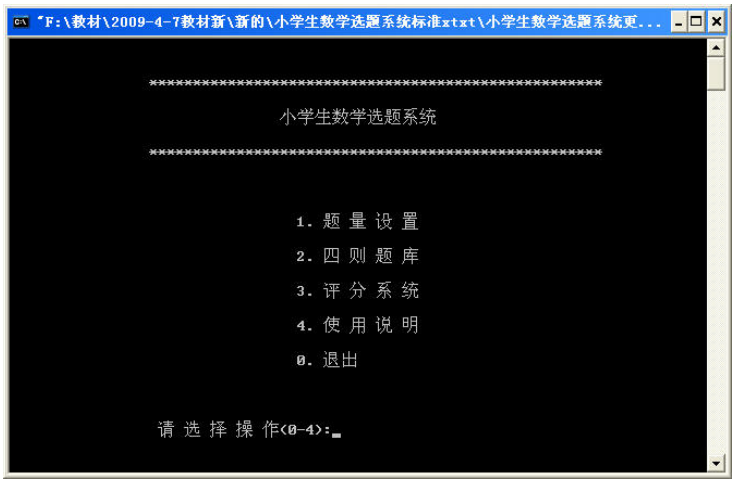


图 2.9　主控模块界面

3．四则题库界面设计

```
// 四则题库界面代码
void selectProblemMenu()
{
    int num;
    system("cls");
    printf("\n\n");
    printf("\t\t*****************************************************\n\n");
    printf("\t\t                  四则题库                           \n\n");
    printf("\t\t*****************************************************\n\n");
    printf("\n\n");
    printf("\t\t\t\t 1. 加 法 题 库\n\n");
    printf("\t\t\t\t 2. 减 法 题 库\n\n");
    printf("\t\t\t\t 3. 乘 法 题 库\n\n");
    printf("\t\t\t\t 4. 除 法 题 库\n\n");
    printf("\t\t\t\t 0. 返回主控界面\n\n");
    printf("\n\t\t 请 选 择 操 作:");
    scanf("%d",&num);
}
```

运行结果如图 2.10 所示。

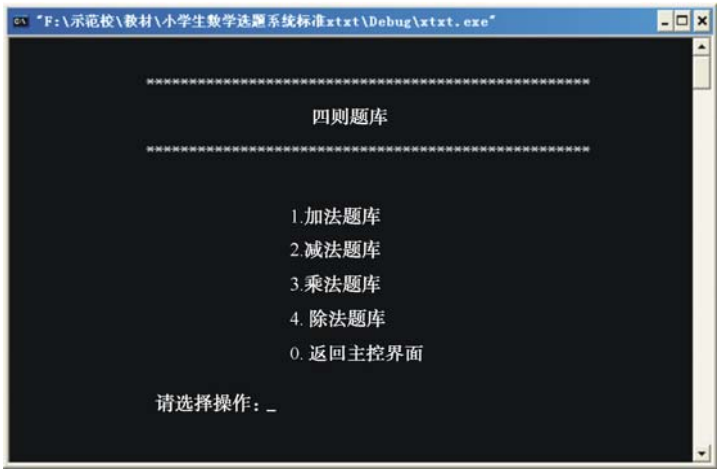


图 2.10　四则题库界面

4．题量设置界面设计

```
//题量设置界面代码
void systemSetting()
{
    system("cls");
    printf("\n\n");
    printf("\t\t**************************************************\n\n");
    printf("\t\t                  题量设置                        \n\n");
    printf("\t\t**************************************************\n\n");
    printf("\n\n");
    printf("\t\t  当前默认题量为:5");
    printf("%d\n\n",problemNumber);
    printf("\t\t  请输入你选择的题量(5 的倍数):");
    scanf("%d",&problemNumber);
    system("cls");
    showMainMenu();
}
```

运行结果如图 2.11 所示。

图 2.11　题量设置界面

5．使用说明界面设计

```
//使用说明界面代码
void helpInfo()
{
    getchar();
    system("cls");
    printf("\n\n");
    printf("\t\t**************************************************\n\n");
    printf("\t\t                    使用说明                      \n\n");
    printf("\t\t**************************************************\n\n");
    printf("\n\n");
    printf("\t1. 本系统共分为使用说明、四则题库、评分系统三个模块。\n\n");
    printf("\t2. 四则题库分为加、减、乘、除运算，每种运算各有 100 道题供练习者使用，\n\n");
    printf("\t    在练习过程中,每练习 5 道题后可选择是否退出训练。\n\n");
    printf("\t3. 评分系统的主要功能是显示每次训练的题数及正误数。\n\n");
    printf("\t\t 按任意键返回主菜单");
    getchar();
        system("cls");
    showMainMenu();
}
```

运行结果如图 2.12 所示。

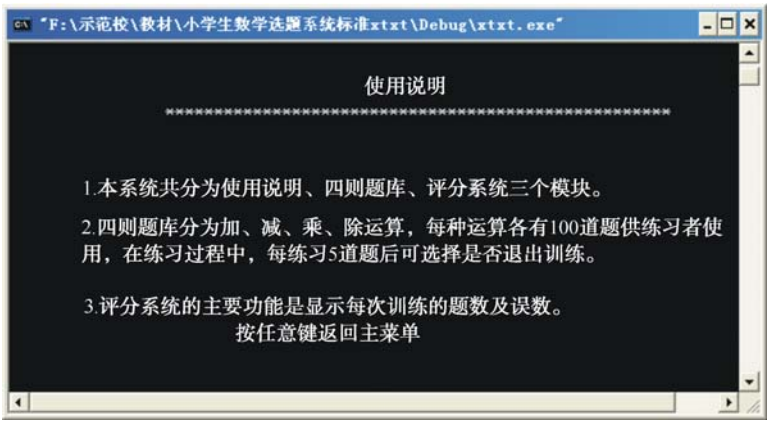


图 2.12　使用说明界面

引导文献

1．人机界面设计

目前，友好的“人机界面”（Human Computer Interface，HCI）已成为基于计算机的系统的一项重要的技术指标。因此，开发人机界面的人机工程已经成为开发基于计算机的系统的一个重要组成部分。

要设计出高质量的人机界面，不仅涉及计算机技术，还涉及美学、心理学等人文科学知识。

2．界面设计的重要性

每个新的系统对用户来说都是一次新的学习过程，如果界面风格经常变化，不保持统一，

无疑会增加用户的学习难度，也许会导致用户的厌烦。要想实际解决这个问题，主要依靠界面设计这一关键环节，一个友好、易用的界面是令用户满意的界面。因此，我们在设计界面时不但要考虑功能和美观，还要考虑用户的可操作性，并非每个用户都非常熟悉计算机，我们应以令没有计算机基础的客户满意为标准，遵循客户至上的原则。

3．人机界面开发过程的主要 5 个步骤

（1）活动分析。即分析人机交互的所有过程，标志该过程中人的活动并据此确定需要计算机执行的任务。

（2）动作定义和设计。

（3）动作的实现。用特定的人机交互语言的语句和命令去实现每一个动作，进而实现各个人机界面的交互活动。如有必要，可以设计人机交互语言，精确地定义语言的语法和语义，并实现语言中的每一个动作和命令。

（4）用户环境的设计。要设计高质量的人机界面，必须考虑将支撑人机界面的软件和硬件集成后，构成集成的用户环境的整体设计效果，还应考虑空间、光线、温度等环境因素。

（5）原型设计。软件工程师根据以上 4 个步骤的结果设计原型，并请用户对原型进行评价和审查。根据用户提出的意见修改原型，这是一个迭代过程，直至通过用户的评审。软件工程师以通过评审的原型为基础设计人机界面，就可以设计出令用户满意的高质量的人机界面。

4．初识C语言

本教材的代码实现是在 VC++环境下实现 TC 的功能，主要好处是，让学生在比较友好的界面下，完成 C 语言的学习。

1）C语言程序简介

为了认识 VC++环境，说明 C 语言源程序结构的特点，下面先介绍几个程序。这几个程序由易到难，充分体现了 C 语言源程序在组成结构上的特点。虽然还未介绍有关内容，但可以从这些例子中了解组成一个 C 语言源程序的基本部分和书写格式。

【例 2.1】

（1）选择“开始”→“程序”→“Microsoft Visual C++ 6.0/ Microsoft Visual C++6.0”。

（2）选择“文件”→“新建”命令，打开如图 2.13 所示的“新建”对话框，选择“文件”选项卡中的“Win32 Console Application”选项，输入工程名称及所在位置。

图 2.13　“新建”对话框

（3）单击“确定”按钮，弹出如图 2.14 所示的对话框。

图 2.14　创建控制台程序

（4）选择“一个空工程”选项后，单击“完成”按钮，弹出如图 2.15 所示的对话框。

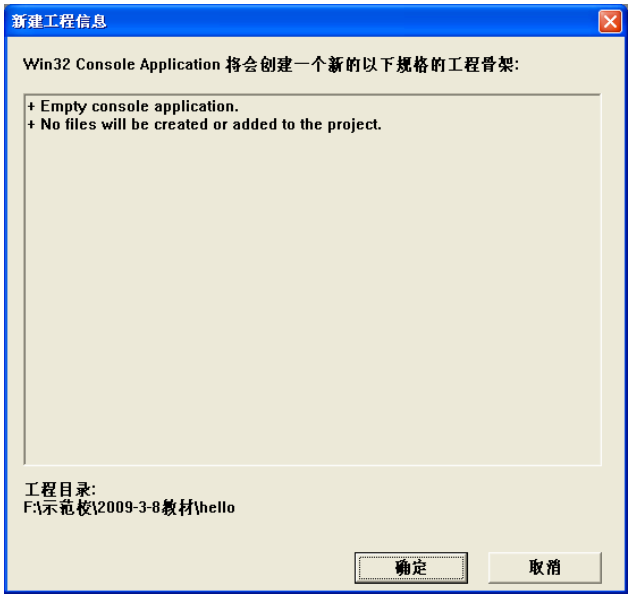


图 2.15　新建工程骨架

（5）单击“确定”按钮后，弹出如图 2.16 所示对话框。在“文件”选项卡中，选择“C++ Source File”，在右侧的“文件名”框中输入“hello.c”。

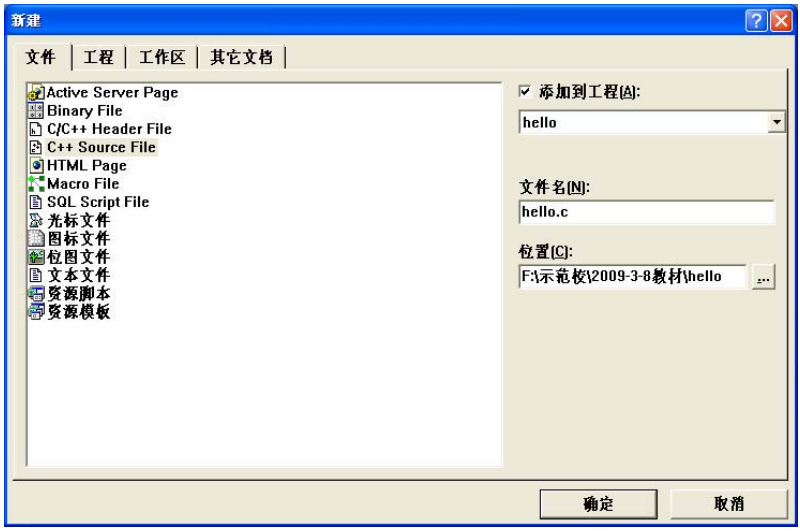


图 2.16　新建 hello.c 文件

（6）单击“确定”按钮后，弹出如图 2.17 所示的窗口。

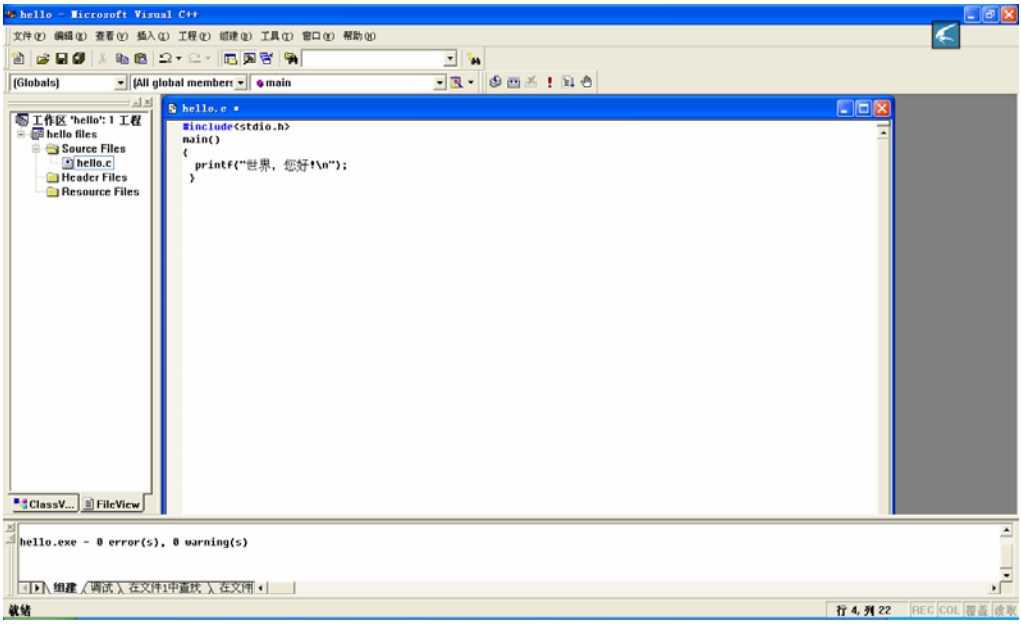


图 2.17　编码实现

（7）按图所示输入【例 2.2】所示的程序后，选择“组建”→“编译”命令，运行结果如图 2.18 所示。

【例 2.2】

```
#include<stdio.h>
main()
{
   printf("世界，您好\n");
  }
```

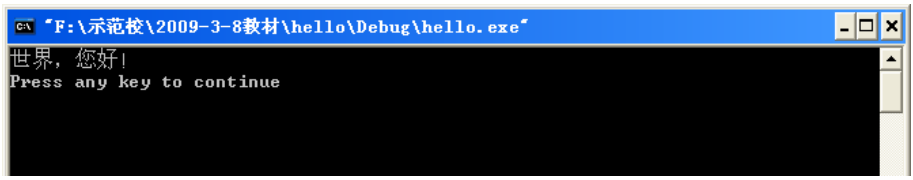


图 2.18　运行结果

说明：

（1）include 称为文件包含命令。

（2）扩展名为.h 的文件称为头文件。

（3）main 是主函数的函数名，表示这是一个主函数。

（4）每一个 C 语言源程序都必须有且只能有一个主函数（main 函数）。

（5）printf 函数是一个由系统定义的标准函数，可在程序中直接调用。

（6）printf 函数的功能是，把要输出的内容送到显示器显示。

程序的功能是在显示器上输出“世界，您好”。在 main()之前的语句称为预处理命令。预处理命令还有其他几种，这里的 include 称为文件包含命令，其意义是把尖括号<>或引号""内指定的文件包含到本程序来，成为本程序的一部分。被包含的文件通常是由系统提供的，其扩展名为.h，因此也称为头文件或首部文件。C 语言的头文件中包括了各个标准库函数的函数原型。因此，凡是在程序中调用一个库函数时，都必须包含该函数原型所在的头文件。在本例中，使用了一个库函数：输出函数 printf，因此在程序的主函数前用 include 命令包含了 stdio.h 文件。

需要说明的是，C 语言规定对标准输入函数 scanf 和标准输出函数 printf 可以省去对其头文件的包含命令。因此，在本例中也可以删去包含命令#include<stdio.h>。

【例 2.2】的主函数体分为两部分：一部分为说明部分，另一部分为执行部分。

说明是指变量的类型说明。在【例 2.2】中，未使用任何变量，因此无说明部分。C 语言规定，源程序中所有用到的变量都必须先说明，后使用，否则将会出错。这是编译型高级程序设计语言的一个特点，与解释型的 BASIC 语言不同。说明部分是 C 语言源程序结构中很重要的组成部分。

说明部分后的程序行为执行语句部分，用以完成程序的功能。在【例 2.2】中，执行部分是一条输出语句，调用 printf 函数在显示器上输出字符串。

2）输入和输出函数

在【例 2.2】中，用到了输出函数 printf，在以后的程序设计过程中还将经常用到输入函数 scanf，这里先简单介绍它们的格式，以便下面使用。

scanf 和 printf 这两个函数分别称为格式输入函数和格式输出函数，其意义是按指定的格式输入/输出值。因此，这两个函数在括号中的参数表都由以下两部分组成：

“格式控制串”，参数表。

格式控制串是一个字符串，必须用双引号括起来，它表示了输入/输出量的数据类型。各种类型的格式表示法可参阅附录 E。在 printf 函数中，还可以在格式控制串内出现非格式控制字符，这时在显示屏幕上将原样输出。参数表中给出了输入或输出的量。当有多个量时，用逗号间隔。例如：

```
printf("a=,b=",a,b);
scanf("%d%d",&x,&y);
```

【例 2.3】

```
int max(int a,int b);                    /*函数说明*/
main()                                   /*主函数*/
{
   int x,y,z;                            /*变量说明*/
   int max(int a,int b);                 /*函数说明*/
   printf("请输入两个数:\n");
   scanf("%d%d",&x,&y);                  /*输入 x,y 值*/
   z=max(x,y);                           /*调用 max 函数*/
   printf("大数是：=%d",z);              /*输出*/
 }
int max(int a,int b)                     /*定义 max 函数*/
{
   if(a>b)return a;
   else return b;                        /*把结果返回主调函数*/
 }
```

在【例 2.3】中，程序的功能是由用户输入两个整数，程序执行后输出其中的大数。本程序由两个函数（主函数和 max 函数）组成。函数之间是并列关系。可从主函数中调用其他函数。max 函数的功能是比较两个数，然后把大数返回给主函数。max 函数是一个用户自定义函数。因此，在主函数中要给出说明。可见，在程序的说明部分中，不仅可以有变量说明，还可以有函数说明。在程序的每行后用/*和*/括起来的内容为注释部分，程序不执行注释部分。

在【例 2.3】中，程序的执行过程是，首先在屏幕上显示提示串，请用户输入两个数，回车后由 scanf 函数语句接收这两个数送入变量 x 和 y 中，然后调用 max 函数，并把 x 和 y 的值传送给 max 函数的参数 a 和 b。在 max 函数中，比较 a 和 b 的大小，把大者返回给主函数的变量 z，最后在屏幕上输出 z 的值。

3）清屏

```
格式：system("cls")
```

该函数包含在头文件 #include <stdlib.h>中，作用是清除屏幕上的显示内容。

4）C语言源程序的结构特点

（1）一个 C 语言源程序可以由一个或多个源文件组成。

（2）每个源文件可由一个或多个函数组成。

（3）一个源程序不论由多少个文件组成，都有一个且只能有一个 main 函数，即主函数。

（4）源程序中可以有预处理命令（include 命令仅为其中的一种），预处理命令通常放在源文件或源程序的最前面。

（5）每一个说明、每一个语句都必须以分号结尾，但预处理命令、函数头和花括号“}”之后不能加分号。

（6）标志符、关键字之间必须至少加一个空格以示间隔。若已有明显的间隔符，也可不再加空格来间隔。

5）书写程序时应遵循的规则

从书写清晰，便于阅读、理解、维护的角度出发，在书写程序时应遵循以下规则。

（1）一个说明或一个语句占一行。

（2）用{} 括起来的部分，通常表示了程序的某一层次结构。{}一般与该结构语句的第一个字母对齐，并单独占一行。

（3）低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写，以便看起来更加清晰，增加程序的可读性。

在编程时应力求遵循这些规则，以养成良好的编程风格。

6）C语言字符集

字符是组成语言的最基本的元素。C 语言字符集由字母、数字、空白符、标点和特殊字符组成。在字符常量、字符串常量和注释中，还可以使用汉字或其他可表示的图形符号。

（1）字母：

① 小写字母 a～z 共 26 个。

② 大写字母 A～Z 共 26 个。

（2）数字：0～9 共 10 个。

（3）空白符。空格符、制表符、换行符等统称为空白符。空白符只在字符常量和字符串常量中起作用。在其他地方出现时，只起间隔作用，编译程序对它们忽略不计。因此，在程序中使用空白符与否，对程序的编译不发生影响，但在程序中适当的地方使用空白符将增加程序的清晰性和可读性。

（4）标点和特殊字符。

7）C语言词汇

在 C 语言中使用的词汇分为 6 类：标志符、关键字、运算符、分隔符、常量、注释符。

（1）标志符。在程序中使用的变量名、函数名、标号等统称为标志符。除库函数的函数名由系统定义外，其余都由用户自定义。C 语言规定，标志符只能是字母（A～Z，a～z）、数字（0～9）、下画线（_）组成的字符串，并且其第一个字符必须是字母或下画线。

以下标志符是合法的：z、y、c6、Name_1、total5 等。

以下标志符是非法的。

① 4d：以数字开头。

② x*y：出现非法字符*。

③ -8x：以减号开头。

在使用标志符时，还必须注意以下事项。

① C 语言不限制标志符的长度，但它受各种版本的 C 语言编译系统的限制，同时也受具体机器的限制。例如，在某版本的 C 语言中规定标志符的前 8 位有效，当两个标志符的前 8 位相同时，则被认为是同一个标志符。

② 在标志符中，大小写是有区别的。例如，name 和 NAME 是两个不同的标志符。

③ 标志符虽然可由程序员随意定义，但标志符是用于标志某个量的符号。因此，命名应尽量有相应的意义，以便于阅读理解，实现“见名知义”。

（2）关键字。关键字是由 C 语言规定的具有特定意义的字符串，通常也称为保留字。用户定义的标志符不应与关键字相同。C 语言的关键字分为以下 3 类。

① 类型说明符：用于定义、说明变量、函数或其他数据结构的类型，如【例 2.3】中用到的 int。

② 语句定义符：用于表示一个语句的功能，如【例 2.3】中用到的 if else 就是条件语句

的语句定义符。

③ 预处理命令：用于表示一个预处理命令，如前面各例中用到的 include。

（3）运算符。C 语言中含有相当丰富的运算符。运算符与变量、函数一起组成表达式，表示各种运算功能。运算符由一个或多个字符组成。

（4）分隔符。在 C 语言中采用的分隔符有逗号和空格两种。逗号主要用在类型说明和函数参数表中，分隔各个变量。空格多用于语句各单词之间，作为间隔符。在关键字、标志符之间必须有一个以上的空格符做间隔，否则将会出现语法错误，例如把 char a 写成 chara，C 语言编译器会把 chara 当成一个标志符处理，其结果必然出错。

（5）常量。C 语言中使用的常量可分为数字常量、字符常量、字符串常量、符号常量、转义字符常量等多种，将在后面章节中给予专门介绍。

（6）注释符。C 语言的注释符是以“/*”开头并以“*/”结尾的串。在“/*”和“*/”之间的即为注释。程序编译时，不对注释做任何处理。注释可出现在程序中的任何位置。注释用来向用户提示或解释程序的意义。在调试程序中对暂不使用的语句也可用注释符括起来，使编译跳过不做处理，待调试结束后再去掉注释符。

即时训练

（1）用 scanf 函数通过键盘向变量 A、B、C 输入数据，其中 A 为整型变量，B 为浮点型变量，C 为字符型变量，写出实现的代码。

（2）已知圆的半径 R=2.5，求圆的周长及面积。用 scanf 函数输入数据，用 printf 函数在显示器上输出结果。

（3）优化“小学生选题系统”界面，可适当对功能模块进行调整，使其更加友好、易用。

建议教学时以组为单位选出代表对本组优秀作品进行阐述并演示，最后选择出最佳设计 1～2 件，作为最后的界面设计方案。

拓展任务

设计体育彩票销售系统的界面。要求有登录模块、体育彩票服务站模块、自行投注模块、添加顾客模块，具体界面设计如图 2.19 至图 2.22 所示。

图 2.19　体育彩票销售系统登录界面

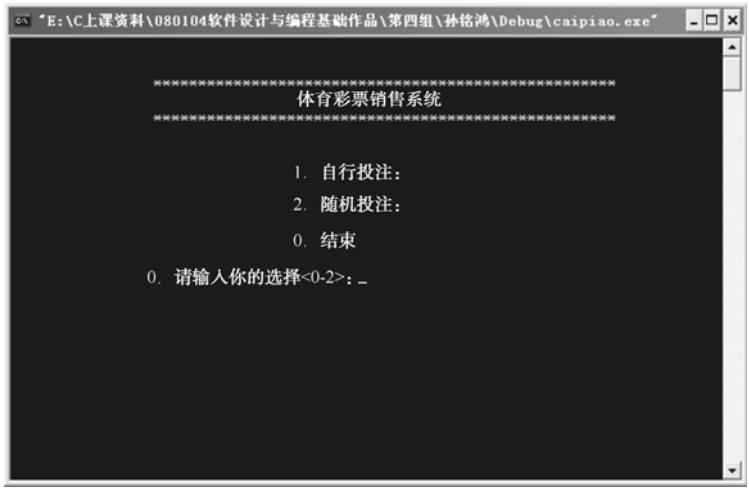


图 2.20　体育彩票销售系统主界面

图 2.21　自行投注界面

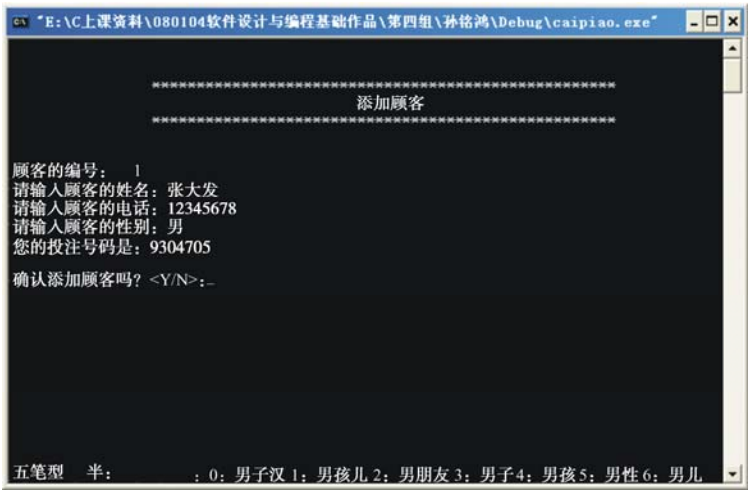


图 2.22　添加顾客界面

2.4.2 子任务二：登录模块的功能实现

任务描述

在登录模块界面设计的基础上，具体实现登录模块的功能，即当用户输入用户名和密码后，系统进行判断。如果用户名和密码正确，则进入主界面；如果错误，则要求重新输入；如果三次错误，则退出系统。

任务分析与设计

1．基本流程

登录模块的基本流程如表 2.24 所示。

表 2.24 登录模块的基本流程

1．登录函数	2．呈现界面如图 2.8 所示，显示“请输入用户名”的提示
3．输入用户名（键盘输入整型数据）	4．显示“请输入密码”的提示
5．输入密码（键盘输入整型数据）	6．判断用户名称和密码是否为 8888，如果正确，则清屏并调用主界面函数 showMainMenu()如图 2.23
7. 否则将累计用户名与密码错误次数的变量 number 加 1	8．判断 number 是否小于 3，如果正确，则清屏并返回步骤 2，否则退出系统

2．自然语言描述

S1：定义用户名、密码、累计用户名与密码错误次数的变量分别为 username、 password 、number，初始化 number 的值为 0。

S2：接收从键盘输入的用户名和密码。

S3：用户明确自己的用户名和密码，如可以分别为 8888。

S4：判断用户名、密码与自己已确定的用户名、密码是否相同。如果相同，则执行 S5，否则执行 S6。

S5：调用主界面 showMainMenu()函数。

S6：清屏。

S7：累计用户名与密码错误次数的变量 number，判断变量 number 是否小于 3。如果小于 3，则执行 S2，否则执行 S8。

S8：退出系统。

3．设计流程图

登录模块的设计流程图如图 2.23 所示。

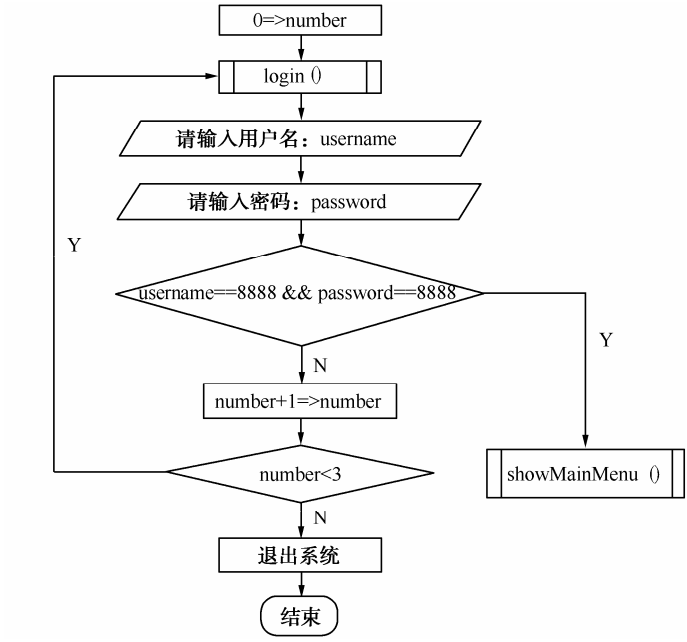


图 2.23　登录模块的设计流程图

任务实现

```
void login();                          /*声明 login 函数*/
void login()                           /*定义 login 函数*/
{
    int username,password;   /*定义用户名 username 和密码 password 两个整型变量*/
    //long j;
    printf("\n\n"); /*\n 为换行符*/
    printf("\t\t**************************************************\n");
    printf("\t\t                选题系统登录                      \n");
    printf("\t\t**************************************************\n");
    printf("\n\n");
    printf("\t\t\t\t 请输入用户名:" );
    scanf("%d",&username);
    printf("\n\n");
    printf("\t\t\t\t 请输入密码:");
    scanf("%d",&password);
    printf("\n\n");
    printf("\n\n");
        if (username==8888 && password==8888)
    {
        system("cls");
        showMainMenu();
    }
        else
        {
            number++;
```

```
                    if(number<3)
                    {
                        system("cls"); /*清除屏幕上的内容*/
                        login();
                    }
                else
                    exit(0);
        }
        }
```

1．数据类型

数据是程序处理的对象。程序设计的过程就是数据加工的过程。一种语言支持的数据类型越丰富，其功能越强、应用范围越广。C 语言规定在程序中使用的每个数据都必须属于某种类型，它还提供了丰富的数据类型，主要有基本数据类型、构造数据类型、指针类型、空类型四大类。基本数据类型有整型、实型（单精度型、双精度型）、字符串型、指针型等。本节重点介绍以下 5 种常用的数据类型。

1）整型（int）

int 代表整型，如 123、−567、75、90000 等。

2）实型

带小数点的数据就是实型，根据数的范围可为分 float、double 两种类型。

（1）float（单精度型）。float 代表的是单精度数据类型，如 7.78、−556.444 等。

（2）double（双精度型）。double 代表的是双精度数据类型，数的范围比 float 类型大，如 1.22554545、222555.0155 等。

3）字符型（char）

char 代表的是字符型，用一对单引号引起的一个数据就是 char 类型，如'311'、'A'、'b'等。

4）字符串型

用双引号引起来的数据类型就是字符串型，如"王丹"、"stud"、"F435"等。注意：在 C 语言中，系统没有提供字符串这种数据类型，一般用数组来表示，关于字符串的详细内容将在后面介绍。

5）指针型

内存单元的地址编号是一种非常特殊的数据类型，称为指针型，简单地说，内存单元的地址就是指针。关于指针的详细内容将在后面介绍。

2．常量与变量

对于基本数据类型，按其取值是否可改变又分为常量和变量两种。在程序执行过程中，其值不发生改变的量称为常量，其值可变的量称为变量。它们可与数据类型结合起来分类。例如，可分为整型常量、整型变量、浮点常量、浮点变量、字符常量、字符变量、枚举常量、枚举变量。在程序中，常量是可以不经说明而直接引用的，而变量则必须先定义后使用。

整型变量包括整型常量、整型变量。

1）常量和符号常量

在程序执行过程中，其值不发生改变的量称为常量。

（1）直接常量（字面常量）。

① 整型常量：34、0、−13。

② 实型常量：14.7、−4.23。

③ 字符常量：'x'、'y'。

（2）标志符：用来标志变量名、符号常量名、函数名、数组名、类型名、文件名的有效字符序列。

（3）符号常量：用标示符代表一个常量。在 C 语言中，可以用一个标志符来表示一个常量，称为符号常量。

符号常量在使用之前必须先定义，其一般形式为：

#define 标志符 常量

其中，#define 也是一条预处理命令（预处理命令都以"#"开头），称为宏定义命令（在后面预处理程序中将进一步介绍），其功能是把该标志符定义为其后的常量值。一经定义，以后在程序中所有出现该标志符的地方均以该常量值代之。

（4）习惯上，符号常量的标志符用大写字母，变量标志符用小写字母，以示区别。

例如：符号常量的使用。

【例 2.4】

```
#define PRICE 50
main()
{
   int num,sum;
   num=20;
   sum=num* PRICE;
   printf("sum=%d",sum);
}
```

说明：

（1）用标志符代表一个常量，称为符号常量。

（2）符号常量与变量不同，它的值在其作用域内不能改变，也不能再被赋值。

（3）使用符号常量的好处是：含义清楚；能做到“一改全改”。

2）变量

其值可以改变的量称为变量。一个变量应该有一个名字，在内存中占据一定的存储单元，如图 2.24 所示。变量定义必须放在变量使用之前。一般放在函数体的开头部分。区分变量名和变量值是两个不同的概念：

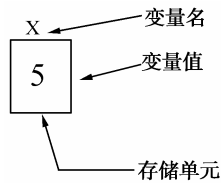


图 2.24　变量与变量名

变量定义的一般形式为：

类型说明符　变量名标志符,变量名标志符,…;

例如：

```
int x,y,z; (x,y,z 为整型变量)
long x1,y1; (x1,y1 为长整型变量)
unsigned x2,y2; (x2,y2 为无符号整型变量)
```

在书写变量定义时，应注意以下几点：

（1）允许在一个类型说明符后，定义多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至少用一个空格间隔。

（2）最后一个变量名之后必须以“;”号结尾。

（3）变量定义必须放在变量使用之前。一般放在函数体的开头部分。

变量赋初值：在程序中常常需要对变量赋初值，以便使用变量。语言程序中可有多种方法为变量提供初值。本小节先介绍在做变量定义的同时给变量赋以初值的方法。这种方法称为初始化。在变量定义中赋初值的一般形式为：

类型说明符　变量 1=值 1,变量 2=值 2,…;

【例 2.5】

```
int a=13;
int b,c=15;
float x=4.2,y=6,z=10.1;
char ch1='T',ch2='H';
```

应注意，在定义中不允许连续赋值，如 a=b=c=5 是不合法的。

3．字符型数据

1）字符常量

用一对单引号括起来的单个字符，称为字符常量，例如，'A'、'2'、'+'、'%'等都是有效的字符型常量。一个字符常量的值是该字符集中对应的编码值，例如在 ASCII 字符集中，字符常量 0～9 的 ASCII 编码值是 48～57。

2）字符变量

字符变量的类型关键字为 char，一般占用 1 字节内存单元。字符变量用来存储字符常量。将一个字符常量存储到一个字符变量中，实际上是将该字符的 ASCII 码值（无符号整数）存储到内存单元中。字符数据在内存中存储的是字符的 ASCII 码——一个无符号整数，其形式与整数的存储形式一样，所以 C 语言允许字符型数据与整型数据之间通用。

字符变量具有以下特点。

（1）一个字符型数据，既可以字符形式输出，也可以整数形式输出。

【例 2.6】

```
//用字符形式和整数形式输出字符变量
main()
  {
      char ch1,ch2;
      ch1='a'; ch2='b';
      printf("ch1=%c,ch2=%c\n",ch1,ch2);
      printf("ch1=%d,ch2=%d\n",ch1,ch2);
  }
```

程序运行结果：

```
ch1=a,ch2=b
ch1=97,ch2=98
```

（2）允许对字符数据进行算术运算，此时就是对它们的 ASCII 码值进行算术运算。

【例 2.7】

```
//字符数据的算术运算
  main()
    {
        char ch1,ch2;
        ch1='a'; ch2='B';
        printf("ch1=%c,ch2=%c\n",ch1-32,ch2+32);
        printf("ch1+100=%d\n", ch1+100);
        printf("ch1+256=%c\n", ch1+256);
    }
```

程序运行结果：

```
ch1=A,ch2=b
ch1+100=197
ch1+256=a
```

3）字符串常量

字符串常量是用一对双引号括起来的若干字符序列。字符串中字符的个数称为字符串长度。长度为 0 的字符串（即一个字符都没有的字符串）称为空串，表示为" "（一对紧连的双引号）。

例如，"How are you"、"student"等，都是字符串常量，其长度分别为 11 和 7（空格也是一个字符）。

C 语言规定：在存储字符串常量时，由系统在字符串的末尾自动加一个'\0'作为字符串的结束标志。

字符常量与字符串常量的区别是：

（1）定界符不同：字符常量使用单引号，而字符串常量使用双引号。

（2）长度不同：字符常量的长度固定为 1，而字符串常量的长度，可以是 0，也可以是某个整数。

（3）存储要求不同：字符常量存储的是字符的 ASCII 码值，而字符串常量，除了要存储有效的字符外，还要存储一个结束标志'\0'。

4．转义字符

转义字符是一种特殊的字符常量。转义字符以反斜线“\”开头，后跟一个或几个字符。转义字符具有特定的含义，不同于字符原有的意义，故称“转义”字符。例如，在前面各例题 printf 函数的格式串中用到的“\n”就是一个转义字符，其意义是“回车换行”。转义字符主要用来表示那些用一般字符不便于表示的控制代码。常用的转义字符及其含义如表 2.25 所示。

表 2.25　常用的转义字符及其含义

转 义 字 符	转义字符的意义	ASCII 代码
\n	回车换行	10
\t	横向跳到下一制表位置	9
\b	退格	8

续表

转 义 字 符	转义字符的意义	ASCII 代码
\r	回车	13
\f	走纸换页	12
\\	反斜线符"\"	92
\'	单引号符	39
\"	双引号符	34
\a	鸣铃	7
\ddd	1～3 位八进制数所代表的字符	
\xhh	1～2 位十六进制数所代表的字符	

广义地讲，C 语言字符集中的任何一个字符均可用转义字符来表示。表中的\ddd 和\xhh 正是为此而提出的。ddd 和 hh 分别为八进制和十六进制的 ASCII 代码。例如，\101 表示字母“A”，\102 表示字母“B”，\134 表示反斜线，\XOA 表示换行等。

例如：转义字符的使用。

【例 2.8】

```
main()
{
   int a,b,c;
   a=5; b=6; c=7;
   printf("abc\tde\rf\n");
   printf("hijk\tL\bM\n");
}
```

5．运算符和表达式

C 语言的运算符和表达式数量之多，在高级语言中是少见的。丰富的运算符和表达式使 C 语言功能更加完善。这也是 C 语言的主要特点之一。

C 语言的运算符不仅具有不同的优先级，而且还有一个特点，就是它的结合性。在表达式中，各运算量参与运算的先后顺序不仅要遵守运算符优先级别的规定，还要受运算符结合性的制约，以便确定是自左向右进行运算还是自右向左进行运算。这种结合性是其他高级语言的运算符所没有的，因此也增加了 C 语言的复杂性。

1）C语言的运算符

C 语言的运算符可分为以下几类。

（1）算术运算符：用于各类数值运算，包括加（+）、减（-）、乘（*）、除（/）、求余（或称模运算，%）、自增（++）、自减（--）共 7 种。

（2）关系运算符：用于比较运算，包括大于（>）、小于（<）、等于（=）、大于等于（>=）、小于等于（<=）和不等于（≠）共 6 种。

（3）逻辑运算符：用于逻辑运算，包括与（&&）、或（||）、非（!）共三种。

（4）赋值运算符：用于赋值运算，分为简单赋值（=）、复合算术赋值（+=，-=，*=，/=，%=）和复合位运算赋值（&=，|=，^=，>>=，<<=）三类共 11 种。

（5）条件运算符：这是一个三目运算符，用于条件求值（?:）。

（6）指针运算符：用于取内容（*）和取地址（&）两种运算。

（7）求字节数运算符：用于计算数据类型所占的字节数（sizeof）。

（8）特殊运算符：有括号()，下标［］，成员（→）等几种。

2）算术运算符和算术表达式

（1）基本的算术运算符。

① 加法运算符“+”：加法运算符为双目运算符，即应有两个量参与加法运算，如 x+y、2+3 等，具有右结合性。

② 减法运算符“-”：减法运算符为双目运算符。但“-”也可作负值运算符，此时为单目运算，如-a，-15 等，具有左结合性。

③ 乘法运算符“*”：双目运算，具有左结合性。

④ 除法运算符“/”：双目运算，具有左结合性。参与运算量均为整型时，结果也为整型，舍去小数。如果运算量中有一个是实型，则结果为双精度实型。

【例 2.9】

```
main(){
   printf("\n\n%d,%d\n",30/7,-30/7);
   printf("%f,%f\n",30.0/7,-30.0/7);
 }
```

说明：在本例中，30/7、-30/7 的结果均为整型，小数全部舍去；而 30.0/7 和-30.0/7 由于有实数参与运算，因此结果也为实型。

⑤ 求余运算符（模运算符）“%”：双目运算，具有左结合性。要求参与运算的量均为整型。求余运算的结果等于两数相除后的余数。

【例 2.10】

```
main(){
   printf("%d\n",200%3);
 }
```

（2）算术表达式、运算符的优先级和结合性。表达式是由常量、变量、函数和运算符组合起来的式子。一个表达式有一个值及其类型，它们等于计算表达式所得结果的值和类型。表达式求值按运算符的优先级和结合性规定的顺序进行。单个的常量、变量、函数可以看成是表达式的特例。

算术表达式是由算术运算符和括号连接起来的式子。

① 算术表达式：用算术运算符和括号将运算对象（也称操作数）连接起来的、符合 C 语法规则的式子。

以下是算术表达式的例子：

```
x+y
(a*2)/b
(x+y)*5-(a+b)/3
++I
sin(x)+sin(y)
(++i)-(j++)
```

② 运算符的优先级：在 C 语言中，运算符的运算优先级共分为 15 级。1 级最高，15 级最低。在表达式中，优先级较高的先于优先级较低的进行运算。在一个运算量两侧的运算符

优先级相同时，则按运算符的结合性所规定的结合方向处理。

③ 运算符的结合性：在 C 语言中，各运算符的结合性分为两种，即左结合性（自左至右）和右结合性（自右至左）。例如，算术运算符的结合性是自左至右，即先左后右。若有表达式 x-y+z，则 y 应先与“-”符号结合，执行 x-y 运算，然后再执行+z 的运算。这种自左至右的结合方向就称为“左结合性”；而自右至左的结合方向称为“右结合性”。最典型的右结合性运算符是赋值运算符。例如 x=y=z，由于“=”的右结合性，应先执行 y=z，再执行 x=(y=z)运算。C 语言运算符中有不少为右结合性，应注意区别，以避免理解错误。

④ 强制类型转换运算符：其一般形式如下。

（类型说明符） （表达式）

它的功能是把表达式的运算结果强制转换成类型说明符所表示的类型。例如：

```
(float) z        把 z 转换为实型
(int)(a+b)       把 a+b 的结果转换为整型
```

（3）自增、自减运算符。自增 1、自减 1 运算符：自增 1 运算符记为“++”，其功能是使变量的值自增 1。自减 1 运算符记为“--”，其功能是使变量值自减 1。

自增 1、自减 1 运算符均为单目运算，都具有右结合性。可有以下 4 种形式。

① ++i：i 自增 1 后再参与其他用运算。

② --i：i 自减 1 后再参与其他用运算。

③ i++：i 参与运算后，i 的值再自增 1。

④ i--：i 参与运算后，i 的值再自减 1。

在理解和使用上容易出错的是 i++和 i--。特别是，当它们出在较复杂的表达式或语句中时，常常难以弄清，因此应仔细分析。

【例 2.11】

```
main(){
   int i=2;
   printf("%d\n",++i);
   printf("%d\n",--i);
   printf("%d\n",i++);
   printf("%d\n",i--);
}
```

3）赋值运算符和赋值表达式

（1）赋值运算符。简单赋值运算符和赋值表达式：简单赋值运算符记为“=”。由“=”连接的式子称为赋值表达式。一般形式为：

变量=表达式

例如：

```
x=a+b
y=i++
```

赋值表达式的功能是计算表达式的值再赋予左边的变量。赋值运算符具有右结合性。因此

```
x=y=z=8
```

可理解为

```
x=(y=(z=8))
```

在其他用高级语言中，赋值构成了一个语句，称为赋值语句；而在 C 语言中，把“=”定义为运算符，从而组成赋值表达式。凡是表达式可以出现的地方均可出现赋值表达式。

例如，下面的式子

```
x=(a=5)+(b=8)
```

是合法的。它的意义是把 5 赋予 a，把 8 赋予 b，再把 a 与 b 相加，把和赋予 x，故 x 应等于 13。

在 C 语言中也可以组成赋值语句，按照 C 语言规定，任何表达式在其末尾加上分号就构成为语句。因此如下式子

```
x=8;a=b=c=5;
```

是赋值语句，在前面各例中已使用过了。

（2）类型转换。如果赋值运算符两边的数据类型不相同，系统将自动进行类型转换，即把赋值号右边的类型换成左边的类型。具体规定如下：

① 实型赋予整型，舍去小数部分。前面的例子已经说明了这种情况。

② 整型赋予实型，数值不变，但将以浮点形式存放，即增加小数部分（小数部分的值为 0）。

③ 字符型赋予整型，由于字符型为一个字节，而整型为两个字节，故将字符的 ASCII 码值放到整型量的低 8 位中，高 8 位为 0。整型赋予字符型，只把低 8 位赋予字符量。

【例 2.12】

```
main(){
int a,b=15;
float x,y=1.86;
char z1='p',z2;
a=y;
x=b;
a=z1;
z2=b;
printf("%d,%f,%d,%c",a,x,a,c2);
}
```

本例表明了上述赋值运算中类型转换的规则。a 为整型，赋予实型量 y 值 1.86 后只取整数 1。x 为实型，赋予整型量 b 值 15。字符型量 z1 赋予 a 变为整型，整型量 b 赋予 z2 后取其低 8 位成为字符型（b 的低 8 位为 01000010，即十进制 66，按 ASCII 码对应于字符 B）。

（3）复合的赋值运算符。在赋值符“=”之前加上其他用二目运算符可构成复合赋值符，如+=，-=，*=，/=，%=，<<=，>>=，&=，^=，|=。

构成复合赋值表达式的一般形式为：

变量　双目运算符=表达式

它等效于

变量=变量 运算符 表达式

例如：

```
p+=6          等价于 p=p+6
x*=y+8        等价于 x=x*(y+8)
a%=b          等价于 a=a%b
```

对于复合赋值符这种写法，初学者可能不习惯，但十分有利于编译处理，能提高编译效率并产生质量较高的目标代码。

6．简单分支语句——if介绍

用 if 语句可以构成分支结构。它根据给定的条件进行判断，以决定执行某个分支程序段。

1）第一种形式为基本形式：if

```
if(表达式) 语句
```

其语义是：如果表达式的值为真，则执行其后的语句，否则不执行该语句。第一种 if 语句的执行流程如图 2.25 所示。

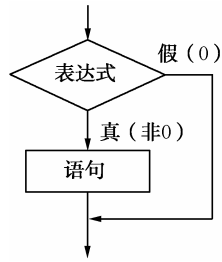


图 2.25　第一种 if 语句的执行流程

【例 2.13】

```
main()
{
    int x,y,max;
    printf("\n 请输入两个数:");
    scanf("%d%d",&x,&y);
    max=x;
    if (max<y) max=y;
    printf("max=%d",max);
}
```

在本例程序中，输入 x 和 y 两个数。把 x 先赋予变量 max，再用 if 语句判别 max 和 y 的大小。若 max 小于 y，则把 y 赋予 max。因此，max 中总是大数，最后输出 max 的值。

2）第二种形式为：if-else

```
if(表达式)
        语句 1;
    else
        语句 2;
```

语义是：如果表达式的值为真，则执行语句 1，否则执行语句 2 。

第二种 if 语句的执行流程如图 2.26 所示。

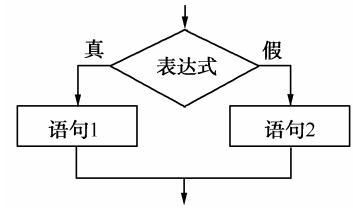


图 2.26　第二种 if 语句的执行流程

【例 2.14】

```
main()
{
    int x,y;
    printf("请输入两个数:");
    scanf("%d%d",&x,&y);
```

```
        if(x>y)
            printf("max=%d\n",x);
        else
            printf("max=%d\n",y);
    }
```

本例输入两个整数，输出其中的大数。

即时训练

（1）输入 A、B、C 三个数，要求按由小到大的顺序输出。

（2）判断用户输入的数是奇数还是偶数，然后在屏幕上显示出相应的信息。

（3）如果将原来代码中的判断条件由 if(number<3)变为 if(number>3)，则重新编写代码实现登录模块。

建议完成代码后，将更改前后的程序进行比较，看哪种实现方法用的代码更少，更能体现结构化设计的优势。

拓展任务

实现体育彩票销售系统的登录模块。

要求：

（1）用户名和密码都为字符型数据。

（2）密码错误给出提示。

（3）密码正确显示主界面。

2.4.3 子任务三：题量设置模块的功能实现

任务描述

题量设置模块主要是实现在四则题库训练选择题量的功能，每次训练前可以按照自己的要求设置本次练习的题量（要求所输入的题量为 5 的倍数），系统会根据所设置的题量，自动出题。如果不设置题量，则系统默认为 5。

任务分析与设计

（1）题量设置模块的基本流程如表 2.26 所示。

表 2.26 题量设置模块的基本流程

1．题量设置函数	2．呈现如图 2.11 所示的界面
3．显示“请输入你选择的题量（5 的倍数）”提示信息	4．输入题量（键盘输入整型数据）
5．判断 amount 是否为 5 的倍数；若不是，则执行第 3 步	

（2）自然语言描述。

S1：显示当前默认题量为 5。

S2：显示提示信息“请输入你选择的题量（5 的倍数）”。

S3：用户输入新设置的题量。

S4：判断题量是否为 5 的倍数。如果是，则更新题量设置，否则执行 S2。

S5：调用主界面 showMainMenu()函数。

S6：退出系统。

（3）设计流程图如图 2.27 所示。

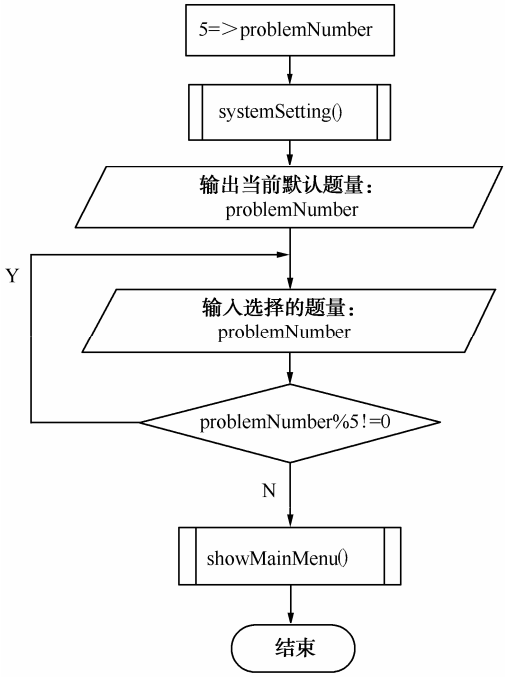


图 2.27　题量设置模块的设计流程图

任务实现

（1）根据系统设计，题量设置需要定义全局变量 amount。

（2）根据函数关系图确定题量设置函数名称 systemSetting()。

（3）实现函数功能。

（4）编码实现。

题量设置设计：

```
void systemSetting();//函数声明

 //定义题量设置函数
void systemSetting()
{
    system("cls");
    printf("\n\n");
    printf("\t\t**************************************************\n\n");
    printf("\t\t                  题量设置                        \n\n");
    printf("\t\t**************************************************\n\n");
    printf("\n\n");
    printf("\t\t  当前默认题量为:");
```

```
        printf("%d\n\n",problemNumber);
        do
        {
            printf("\t\t  请输入你选择的题量(5 的倍数):");
            scanf("%d",&problemNumber);
        }while(problemNumber%5!=0);
        system("cls");
        showMainMenu();
    }
```

1．局部变量和全局变量

1）局部变量

局部变量也称为内部变量。局部变量是在函数内做定义说明的，其作用域仅限于函数内，离开该函数后再使用这种变量是非法的。

说明：

（1）主函数中定义的变量也只能在主函数中使用，不能在其他用函数中使用。因为主函数也是一个函数，它与其他函数是平行关系。这一点与其他语言不同，应予以注意。

（2）形参变量是属于被调函数的局部变量，实参变量是属于主调函数的局部变量。

（3）允许在不同的函数中使用相同的变量名，它们代表不同的对象，分配不同的单元，互不干扰，也不会发生混淆。

（4）在复合语句中也可定义变量，其作用域只在复合语句范围内。

2）全局变量

全局变量是定义在函数外面的变量，即在所有{}外面，其作用范围是整个程序，也就是说，用户可以在任何函数内直接使用，不必每次都定义；但是，如果在一个函数内，局部变量与全局变量名相同，使用这个变量时，以定义的局部变量的值为准，但不改变全局变量的值！局部变量是定义在一个函数里的变量，即在某一个{}里面，只能在这个函数范围内使用。

例如：

```
int a=3; /*  变量 a 就是全局变量*/
main()
{ int a=10; /*局部与全局变量名相同，在本函数中以局部值为准，但不改变全局变量值*/
printf("%d\n",fun(5)*a); /*这里的 a 值是 10 /
}
fun(int b)
{ if(b==0) return(a); /*这里的 a 还是全局变量值*/
return(fun(b-1)*b);
}
```

程序运行的结果是 3600。

首先，上述程序是要求我们输出 fun(5)*a，这里的 a 的值应该是主函数里定义的 10，而不是外面的全局变量 3（变量的作用域）；然后程序去调用子函数 fun()来计算 fun(5)，根据题目，即返回 fun(4)*5，…，递归调用，相当于 fun(0)*1*2*3*4*5，而到 fun(0)时应返回 a，这里的 a 就是全局变量 3。因为该 fun()中没有再次定义 a 这个变量，所以 fun(5)最后的结果应该

是 3*1*2*3*4*5=360。因此，最终结果为 fun(5)*10=3600。

2．变量的存储类别

变量的存储方式可分为“静态存储”和“动态存储”两种。

1）静态存储变量

静态存储变量通常是在变量定义时就分配存储单元并一直保持不变，直至整个程序结束。

2）动态存储变量

动态存储变量是在程序执行过程中，使用它时才分配存储单元，使用完毕立即释放。

典型的例子是函数的形式参数，在函数定义时并不给形参分配存储单元，只是在函数被调用时，才予以分配，调用函数完毕立即释放。如果一个函数被多次调用，则反复地分配、释放形参变量的存储单元。从以上分析可知，静态存储变量是一直存在的，而动态存储变量则时而存在时而消失。我们又把这种由于变量存储方式不同而产生的特性称为变量的生存期。生存期表示了变量存在的时间。生存期和作用域从时间和空间这两个不同的角度来描述变量的特性，这两者既有联系，又有区别。一个变量究竟属于哪一种存储方式，并不能仅从其作用域来判断，还应有明确的存储类型说明。

在 C 语言中，对变量的存储类型说明有以下 4 种。

（1）auto：自动变量。

（2）register：寄存器变量。

（3）extern：外部变量。

（4）static：静态变量。

自动变量和寄存器变量属于动态存储方式，外部变量和静态变量属于静态存储方式。在介绍变量的存储类型之后，可以知道对一个变量的说明不仅应说明其数据类型，还应说明其存储类型。因此，变量说明的完整形式应为：

存储类型说明符　数据类型说明符　变量名,变量名…;

【例 2.15】

```
static int x,y;                  说明 x,y 为静态类型变量
auto char c1,c2;                 说明 c1,c2 为自动字符变量
static int s[5]={1,2,3,4,5};     说明 s 为静态整型数组
extern int a,b;                  说明 a,b 为外部整型变量
```

下面介绍自动变量的存储类型。

自动变量的存储类型说明符为 auto。这种存储类型是 C 语言程序中使用最广泛的一种类型。C 语言规定，函数内凡未加存储类型说明的变量均视为自动变量，也就是说，自动变量可省去说明符 auto。在前面各章的程序中所定义的变量凡未加存储类型说明符的都是自动变量。

【例 2.16】

```
{ int i,j,k;
char c;
……
}等价于：  { auto int i,j,k;
auto char c;
……
}
```

自动变量具有以下特点：自动变量的作用域仅限于定义该变量的个体内。在函数中定义的自动变量，只在该函数内有效。在复合语句中定义的自动变量只在该复合语句中有效。

【例 2.17】

```
int kv(int a)
{
auto int x,y;
{ auto char c;
} /*c 的作用域*/
……
} /*a,x,y 的作用域*/
```

自动变量属于动态存储方式，只有在使用它，即定义该变量的函数被调用时才给它分配存储单元，开始它的生存期。函数调用结束，释放存储单元，结束生存期。因此，函数调用结束之后，自动变量的值不能保留。在复合语句中定义的自动变量，在退出复合语句后也不能再使用，否则将引起错误。

【例 2.18】

```
main()
{ auto int a,s,p;
printf("\ninput a number:\n");
scanf("%d",&a);
if(a>0){
s=a+a;
p=a*a;
}
printf("s=%d p=%d\n",s,p);
}
{ auto int a;
printf("\ninput a number:\n");
scanf("%d",&a);
if(a>0){
auto int s,p;
s=a+a;
p=a*a;
}
printf("s=%d p=%d\n",s,p);
}
```

s、p 是在复合语句内定义的自动变量，只能在该复合语句内有效。上面程序的第 9 行却是退出复合语句之后用 printf 语句输出 s、p 的值，这显然会引起错误。

由于自动变量的作用域和生存期都局限于定义它的个体内（函数或复合语句内），因此不同的个体中允许使用同名的变量而不会混淆。即使在函数内定义的自动变量也可与该函数内部的复合语句中定义的自动变量同名。

【例 2.19】

```
main()
{
```

```
auto int a,s=100,p=100;
printf("\ninput a number:\n");
scanf("%d",&a);
if(a>0)
{
auto int s,p;
s=a+a;
p=a*a;
printf("s=%d p=%d\n",s,p);
}
printf("s=%d p=%d\n",s,p);
}
```

本程序在 main 函数中和复合语句内两次定义了变量 s、p 为自动变量。按照 C 语言的规定，在复合语句内，应由复合语句中定义的 s、p 起作用，故 s 的值应为 a+a，p 的值为 a*a。退出复合语句后的 s、p 应为 main 所定义的 s、p，其值在初始化时给定，均为 100。从输出结果可以分析出两个 s 和两个 p 虽变量名相同，但却是两个不同的变量。

对构造类型的自动变量（如数组等），不可做初始化赋值。

3．while循环语句

一般形式为：

```
while(表达式)语句
```

其中表达式是循环条件，语句为循环体。

该循环语句的执行过程是：计算表达式的值，当值为真（非 0）时，执行循环体语句。while 循环语句的执行过程如图 2.28 所示。

【例 2.20】　用 while 循环语句求 $\sum_{n=1}^{100} n$。

用 while 循环语句求 $\sum_{n=1}^{100} n$ 的算法流程图如图 2.29 所示。

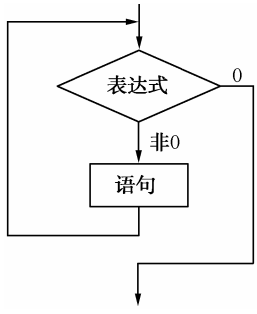


图 2.28　while 循环语句的执行过程

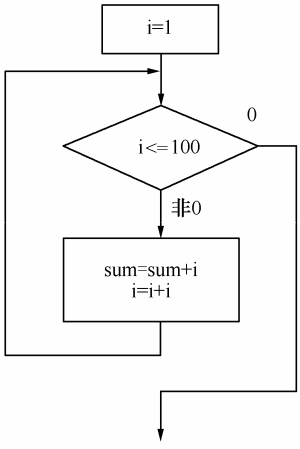


图 2.29　用 while 循环语句求 $\sum_{n=1}^{100} n$ 的算法流程图

```
main()
{
    int i,sum=0;
    i=1;
    while(i<=100)
          {
          sum=sum+i;
          i++;
          }
    printf("%d\n",sum);
}
```

【例 2.21】　统计从键盘输入一行字符的个数。

```
#include <stdio.h>
main(){
     int n=0;
     printf("请输入一行字符（回车结束）:\n");
     while(getchar()!='\n') n++;
     printf("%d",n);
}
```

本例程序中的循环条件为 getchar()!='\n'，其意义是，只要从键盘输入的字符不是回车就继续循环。循环体 n++完成对输入字符的个数计数。该程序实现了对输入一行字符的个数计数。

说明：

（1）循环体如果包含一个以上的语句，则必须用花括号括起来，以复合语句的形式出现，否则 while 语句范围只到 while 后面第一个分号处。

（2）在循环体中应有使循环趋向于结束的语句，即设置修改循环条件的语句。

（3）while 语句先判断表达式的值，然后执行循环体中的语句。如果表达式的值一开始为假（值为 0），则不执行循环体，直接执行循环体以外的语句。表达式是控制循环的条件，它可以是任何类型的表达式。

4．do while循环语句

一般形式为：

```
do
     语句
while(表达式);
```

该循环语句的执行过程是：先执行循环中的语句，然后再判断表达式是否为真。如果为真，则继续循环；如果为假，则终止循环。因此，do while 循环至少要执行一次循环语句。do while 循环语句的执行过程如图 2.30 所示。

【例 2.22】　用 do while 循环语句求 $\sum_{n=1}^{100} n$。

用 do while 循环语句求 $\sum_{n=1}^{100} n$ 的算法流程图如图 2.31 所示。

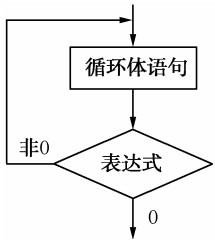


图 2.30　do while 循环语句的执行过程

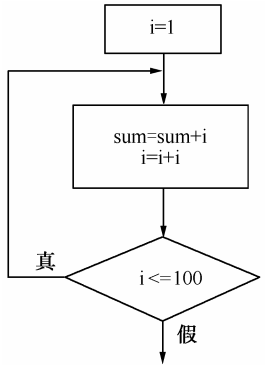


图 2.31　用 do while 循环语句求 $\sum_{n=1}^{100} n$ 的算法流程图

```
//求 1 到 100 的累加和
void main()
{
    int i,sum=0;
    i=1;
    do
          {
            sum=sum+i;
            i++;
           }
while(i<=100);
     printf("%d\n",sum);
}
```

当有许多语句参加循环时，要用“{”和“}”把它们括起来。

说明：

（1）do while 语句的表达式是任意表达式，是控制循环的条件。

（2）先执行后判断。因此，循环体至少执行一次，直到表达式为“假”时才退出循环。因此，在循环体语句中一定要有改变表达式的值的操作，最终使其表达式的值变为 0，结束循环。否则将成为“死”循环。

（3）如果 do while 语句的循环体部分是由多个语句组成的，则必须用左、右花括号括起来，使其形成复合语句。

（4）在关键字 while 的小括号的后面，一定要加分号“;”，千万不能忘记，它表示 do while 语句到此结束。

【例 2.23】　while 和 do while 循环语句的比较。

```
（1）用 while 循环求 1-100 的和
main()
{int sum=0,i;
scanf("%d",&i);
while(i<=100)
      {sum=sum+i;
       i++;
}
```

```
printf("sum=%d",sum);
}
（2）用 do-while 循环求 1-100 的和
main()
{int sum=0,i;
scanf("d",&i);
do
    {sum=sum+i;
     i++;
    }while(i<=100);
printf("sum=%d",sum);
}
```

5．switch case语句

```
switch(num)
    {
        case 0:
            system("cls");
            showMainMenu();
            break;
        case 1:
            system("cls");
            addition();//加法题库
            break;
        case 2:
            system("cls");
            subtration();//减法题库
            break;
        case 3:
            system("cls");
            multiplication();//乘法题库
            break;
        case 4:
            system("cls");
            division();//除法题库
            break;
            default:
            printf("\n\t\t 请按任意键重新选择！！！");
            getchar();
            system("cls");
            selectProblemMenu();
}
```

即时训练

设计红、绿、灯程序，输入 R 显示红灯；输入 G 显示绿灯；输入 Y 显示黄灯。

拓展任务

设计体育彩票销售系统自行投注模块，添加顾客后，顾客自选投注号码，具体效果如图 2.32 和图 2.33 所示。

图 2.32　自行投注界面

图 2.33　产生投注号码

2.4.4　子任务四：四则题库模块的功能实现

任务描述

（1）具体实现四则题库模块的题型选择功能，即显示题库选择界面，当用户选择题型后，系统进行判断，如果选择的是加法题库，进入加法题库界面，以此类推；如果选择的是返回操作，则返回主控界面；如果输入其他值，则返回题库选择界面要求重新输入选择。

（2）在四则题库模块界面设计的基础上，实现加法模块的功能。

加法题库：每次训练的题量为系统全局变量 problem 的值，结束后可以选择“继续”或者“取消”，如果选择“继续”选项，系统会再出一套题继续训练；否则进行其他操作。

（3）在四则题库模块界面设计的基础上，实现减法模块的功能。

减法题库：输出算式时要求操作数 x1>x2，即结果为大于或等于 0 的结果。

（4）在四则题库模块界面设计的基础上，实现乘法模块的功能。

（5）在四则题库模块界面设计的基础上，实现除法模块的功能。

除法题库：输出的操作数 x1、x2 做整除运算，并且除数 x2 不能为 0。

任务分析与设计

1．四则题库模块

（1）基本流程如表 2.27 所示。

表 2.27　四则题库模块的基本流程

1．四则题库函数
2．呈现如图 2.10 所示的界面，显示“请选择操作”提示
3．输入题库类型（键盘输入整型数据）
4．判断 num 的值，如果是 1-4，则调用对应数学题库；如果是 0，则返回主控界面；否则返回四则题库界面重新选择

（2）自然语言描述。

S1：输出四则题库界面。

S2：输入选择给变量 num。

S3：判断 num 的值，为 0，调用主控函数；为 1，调用加法题库；为 2，调用减法题库；为 3，调用乘法题库；为 4，调用除法题库；输入其他任意值，返回四则题库界面重新选择。

S4：退出系统。

（3）四则题库模块的设计流程图如图 2.34 所示。

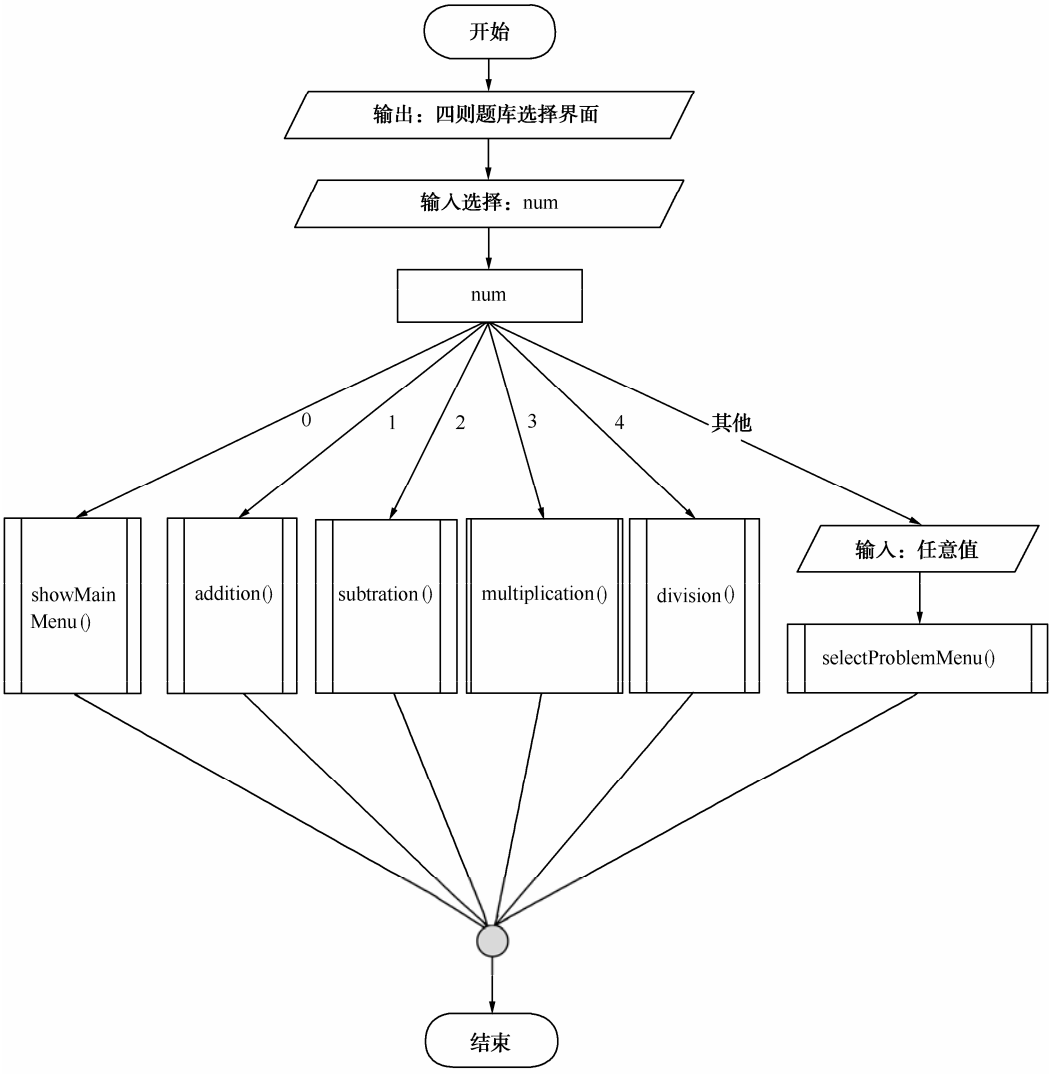


图 2.34　四则题库模块的设计流程图

2．加法题库模块

（1）基本流程如表 2.28 所示。

表 2.28　加法题库模块的基本流程

1．加法题库函数	2．循环控制变量 I 初始化为 1
3．判断 i 是否小于或等于题量设置 problemNumber，若不符合，则执行第 8 步	4．呈现如图 2.35 所示的界面，显示加法题
5．输入答案	6．判断答案正误，显示相应的提示信息。当答案正确时，全局变量 tnum 自加（记录正确题目数量）；错误时附加显示正确答案
7．i 自加，然后执行第 3 步	8．显示是否继续的提示信息
9．输入选择（键盘输入整型数据）	10．判断用户输入的信息，如果为继续，则调用加法函数；如果为取消，则返回四则题库函数，否则执行第 8 步

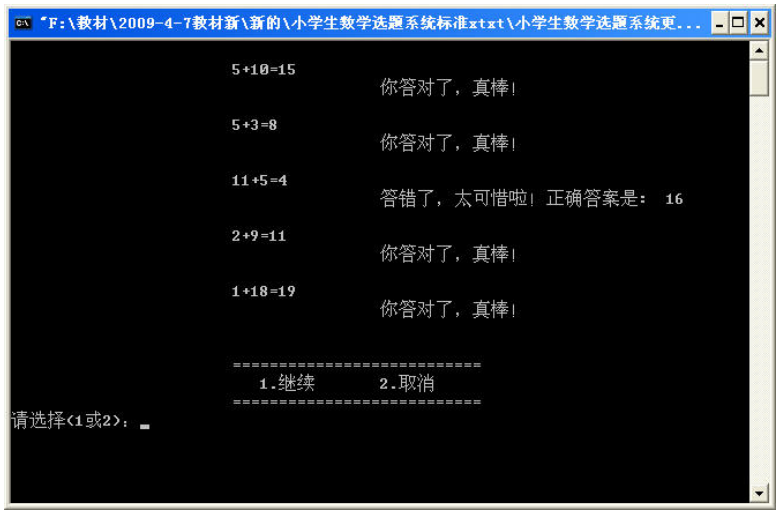


图 2.35　加法题库操作界面

（2）自然语言描述。

S1：定义整型变量。i 为循环控制变量，x1、x2 为随机产生 1～20 的操作数，sumsys 为正确答案，sumuser 为用户答案，num 为是否继续操作的控制变量。

S2：变量 i 初始化为 1。

S3：判断 i 是否小于或等于题量设置 problemNumber。若不符合，则执行 S9。

S4：产生 1～20 之间的随机整数分别赋给 x1、x2。

S5：输出加法算式。

S6：用户输入答案。

S7：判断用户输入答案与正确答案是否相同。若相同，则输出正确的提示信息，同时全局变量 tnum 自加；否则输出错误的提示信息和正确答案，全局变量 fnum 自加。

S8：i 自加，然后执行 S3。

S9：输出是否继续的提示信息。

S10：用户输入选择的操作。

S11：判断用户输入的信息，如果为继续（1），则调用加法函数 addition；如果为取消（2），则返回四则题库函数 selectProblemMenu()；否则执行 S8。

（3）设计流程图如图 2.36 所示。

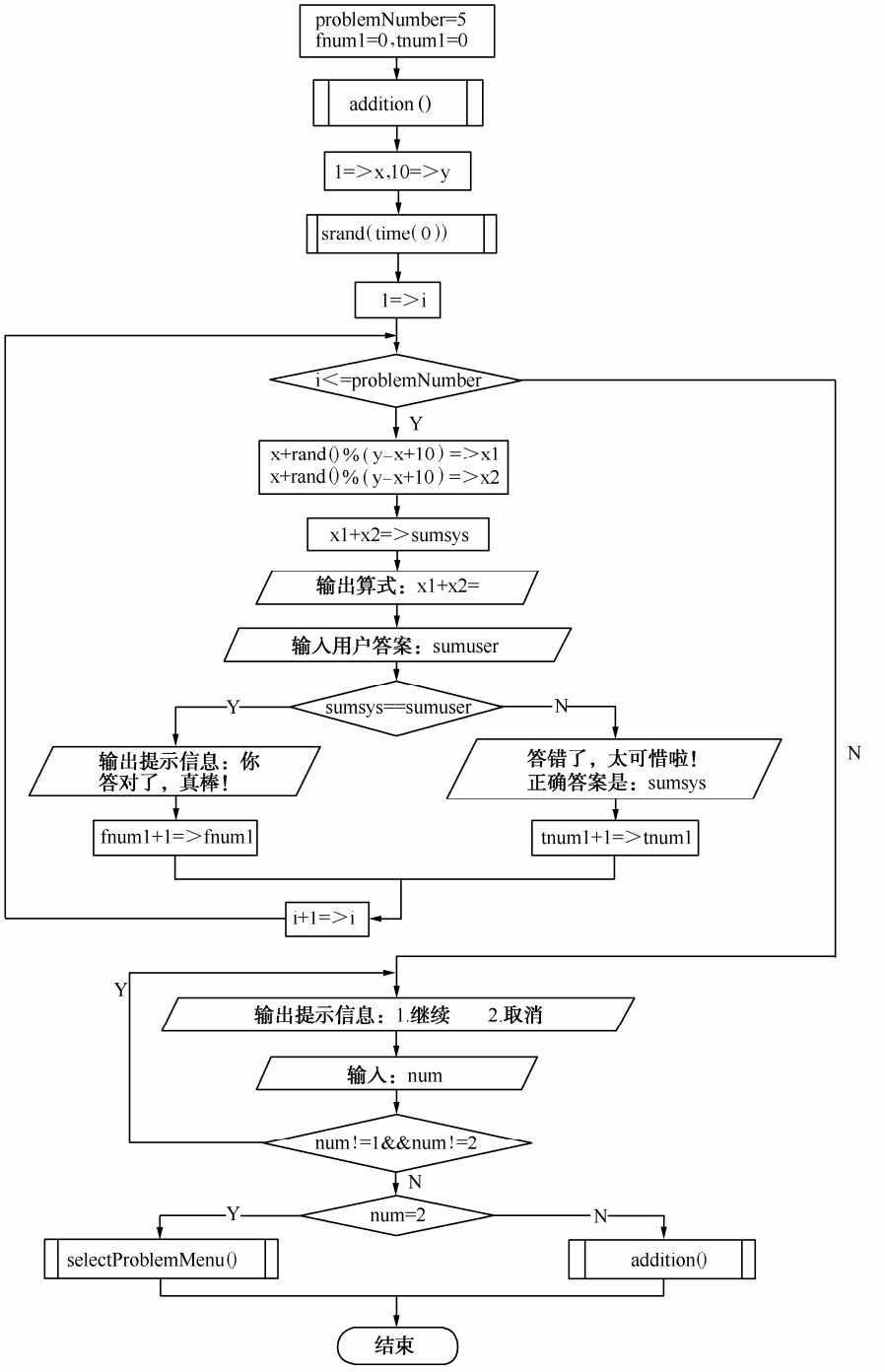


图 2.36　加法题库模块的设计流程图

3．减法题库模块

（1）基本流程如表 2.29 所示。

表 2.29　减法题库模块的基本流程

1．减法题库函数	2．循环控制变量 i 初始化为 1
3．判断 i 是否小于或等于题量设置 problemNumber，若不符合，则执行第 8 步	4．呈现界面（参照加法题库），显示减法题（保证操作数 x1>x2）
5．输入答案	6．判断答案正误，显示相应的提示信息。当答案正确时，全局变量 tnum 自加（记录正确题目数量）；错误时附加显示正确答案， 全局变量 fnum 自加（记录错误题目数量）
7．i 自加，执行第 3 步	8．显示是否继续的提示信息
9．输入选择（键盘输入整型数据）	10．判断用户输入的信息，如果是继续，则调用减法函数；如果是取消，则返回四则题库函数，否则执行第 8 步

（2）自然语言描述。

S1：定义整型变量。i 为循环控制变量，x1、x2 为随机产生 1～20 的操作数，sumsys 为正确答案，sumuser 为用户答案，num 为是否继续操作的控制变量。

S2：变量 i 初始化为 1。

S3：判断 i 是否小于或等于题量设置 problemNumber。若不符合，则执行 S10。

S4：产生 1～20 之间的随机整数分别赋给 x1、x2。

S5：判断是否 x1<x2，若符合，则互换两变量的值。

S6：输出算式。

S7：用户输入答案。

S8：判断用户输入答案与正确答案是否相同。若相同，则输出正确的提示信息，同时全局变量 tnum 自加；否则输出错误的提示信息和正确答案，全局变量 fnum 自加。

S9：是 i 自加，执行 S3。

S10：输出是否继续的提示信息。

S11：用户输入选择的操作。

S12：判断用户输入的信息。如果为继续，则调用减法函数 subtration；如果为取消，则返回四则题库函数 selectProblemMenu()；否则执行 S10。

（3）设计流程图如图 2.37 所示。

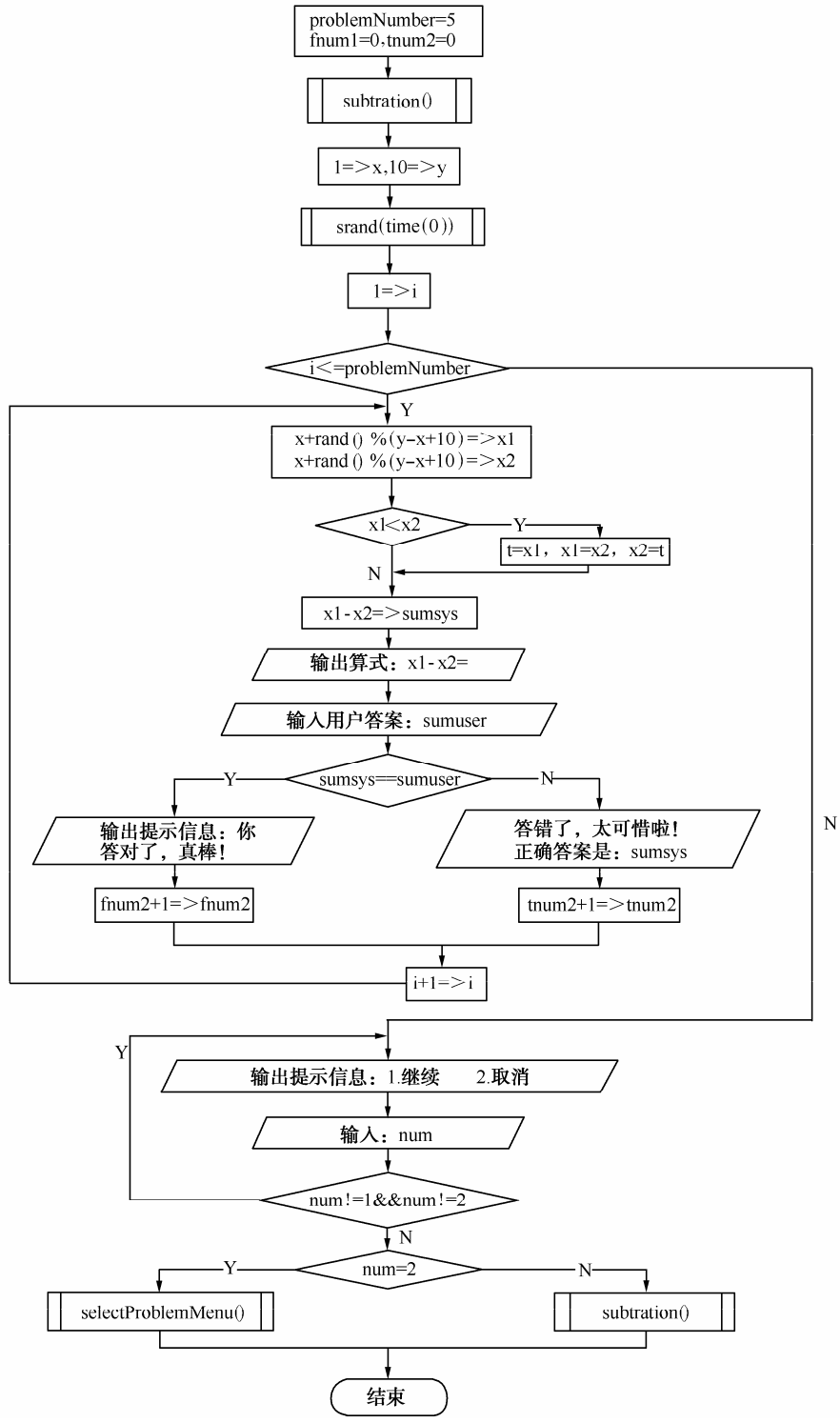


图 2.37　减法题库模块的设计流程图

4．乘法题库模块

（1）基本流程如表 2.30 所示。

表 2.30　乘法题库模块的基本流程

1．乘法题库函数	2．循环控制变量 i 初始化为 1
3．判断 i 是否小于或等于题量设置 problemNumber。若不符合，则执行第 8 步	4．呈现界面（参照加法题库），显示乘法题
5．输入答案	6．判断答案正误，显示相应的提示信息。当答案正确时，全局变量 tnum 自加（记录正确题目数量）；错误时显示错误信息和正确答案，全局变量 fnum 自加（记录错误题目数量）
7．i 自加，执行第 3 步	8．显示是否继续的提示信息
9．输入选择（键盘输入整型数据）	10．判断用户输入的信息。如果是继续，则调用乘法函数；如果是取消，则返回四则题库函数，否则执行第 8 步

（2）自然语言描述。

S1：定义整型变量。i 为循环控制变量，x1、x2 为随机产生 1～20 的操作数，sumsys 为正确答案，sumuser 为用户答案，num 为是否继续操作的控制变量。

S2：变量 i 初始化为 1。

S3：判断 i 是否小于或等于题量设置 problemNumber。若不符合，则执行 S9。

S4：产生 1～20 之间的随机整数分别赋给 x1、x2。

S5：输出算式。

S6：用户输入答案。

S7：判断用户输入答案与正确答案是否相同。若相同，则输出正确的提示信息，同时全局变量 tnum 自加；否则输出错误的提示信息和正确答案，全局变量 fnum 自加。

S8：i 自加，执行 S3。

S9：输出是否继续的提示信息。

S10：用户输入选择的操作。

S11：判断用户输入的信息。如果为继续，则调用乘法函数 multiplication；如果为取消，则返回四则题库函数 selectProblemMenu()；否则执行 S9。

（3）设计流程图如图 2.38 所示。

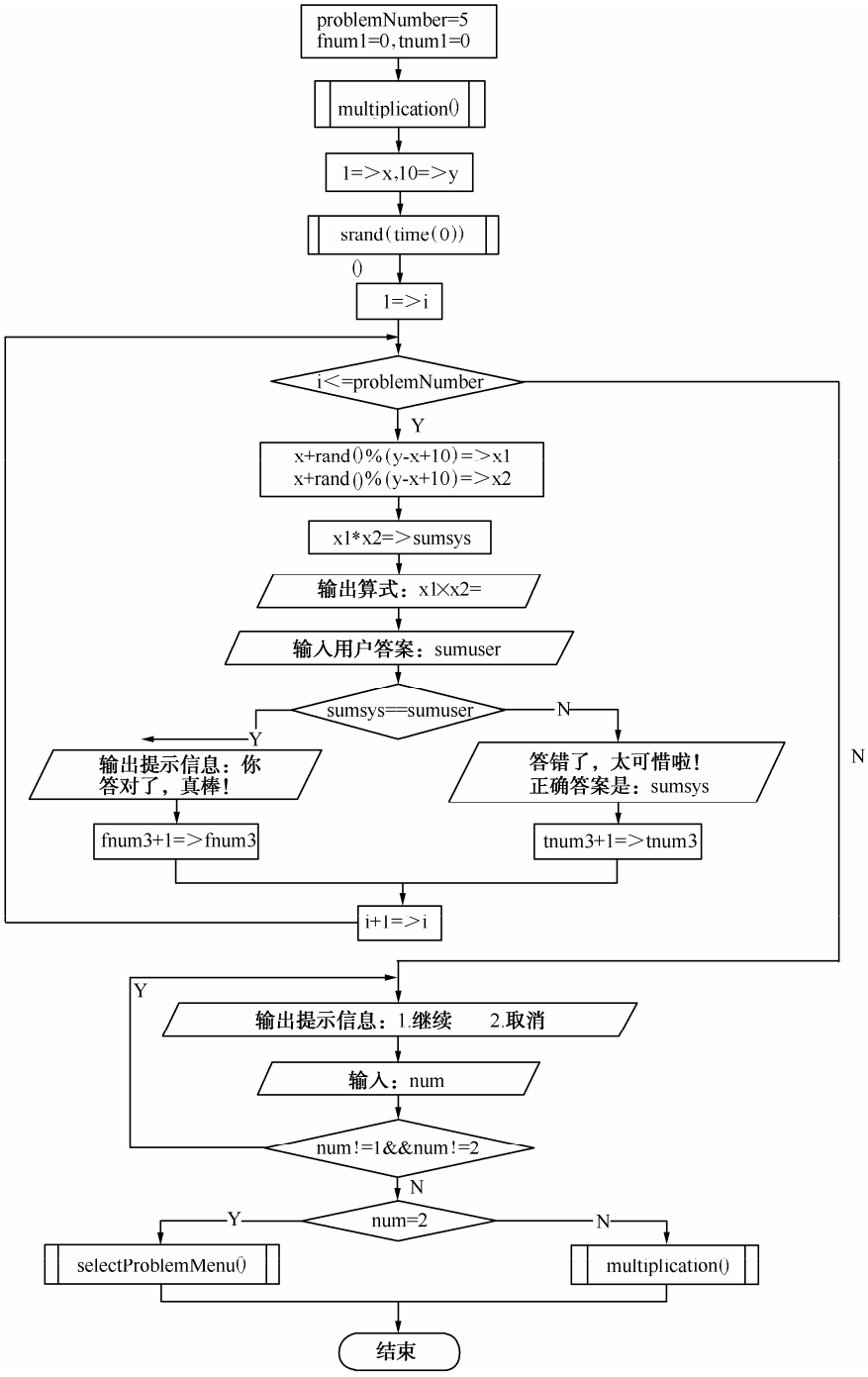


图 2.38　乘法题库模块的设计流程图

5．除法题库模块

（1）基本流程如表 2.31 所示。

表 2.31　除法题库模块的基本流程

1．除法题库函数	2．循环控制变量 i 初始化为 1
3．判断 i 是否小于或等于题量设置 problemNumber。若不符合，则执行第 8 步	4．呈现界面（参照加法题库），显示除法题（保证操作数 x1、x2 整除且 x2 不为零）
5．输入答案	6．判断答案正误，显示相应的提示信息。答案正确时，全局变量 tnum 自加（记录正确题目数量）；错误时显示提示信息和正确答案，全局变量 fnum 自加（记录错误题目数量）
7．i 自加，执行第 3 步	8．显示是否继续的提示信息
9．输入选择（键盘输入整型数据）	10．判断用户输入的信息。如果是继续，则调用加法函数；如果是取消，则返回四则题库函数，否则执行第 8 步

（2）自然语言描述。

S1：定义整型变量。i 为循环控制变量，x1、x2 为随机产生操作数，sumsys 为正确答案，sumuser 为用户答案；num 为是否继续操作的控制变量。

S2：变量 i 初始化为 1。

S3: 判断 i 是否小于或等于题量设置 problemNumber。若不符合，则执行 S10。

S4：产生随机整数分别赋给 sumsys、x2（不为零）。

S5：给变量 x1 赋值为 sumsys*x2 的值。

S6：输出算式 x1÷x2=。

S7：用户输入答案。

S8：判断用户输入答案与正确答案是否相同。若相同，则输出正确的提示信息，同时全局变量 tnum 自加；否则输出错误的提示信息和正确答案，同时全局变量 fnum 自加。

S9：i 自加，执行 S3。

S10：输出是否继续的提示信息。

S11：用户输入选择的操作。

S12：判断用户输入的信息。如果为继续，则调用除法函数 division；如果为取消，则返回四则题库函数 selectProblemMenu()；否则执行 S10。

（3）设计流程图如图 2.39 所示。

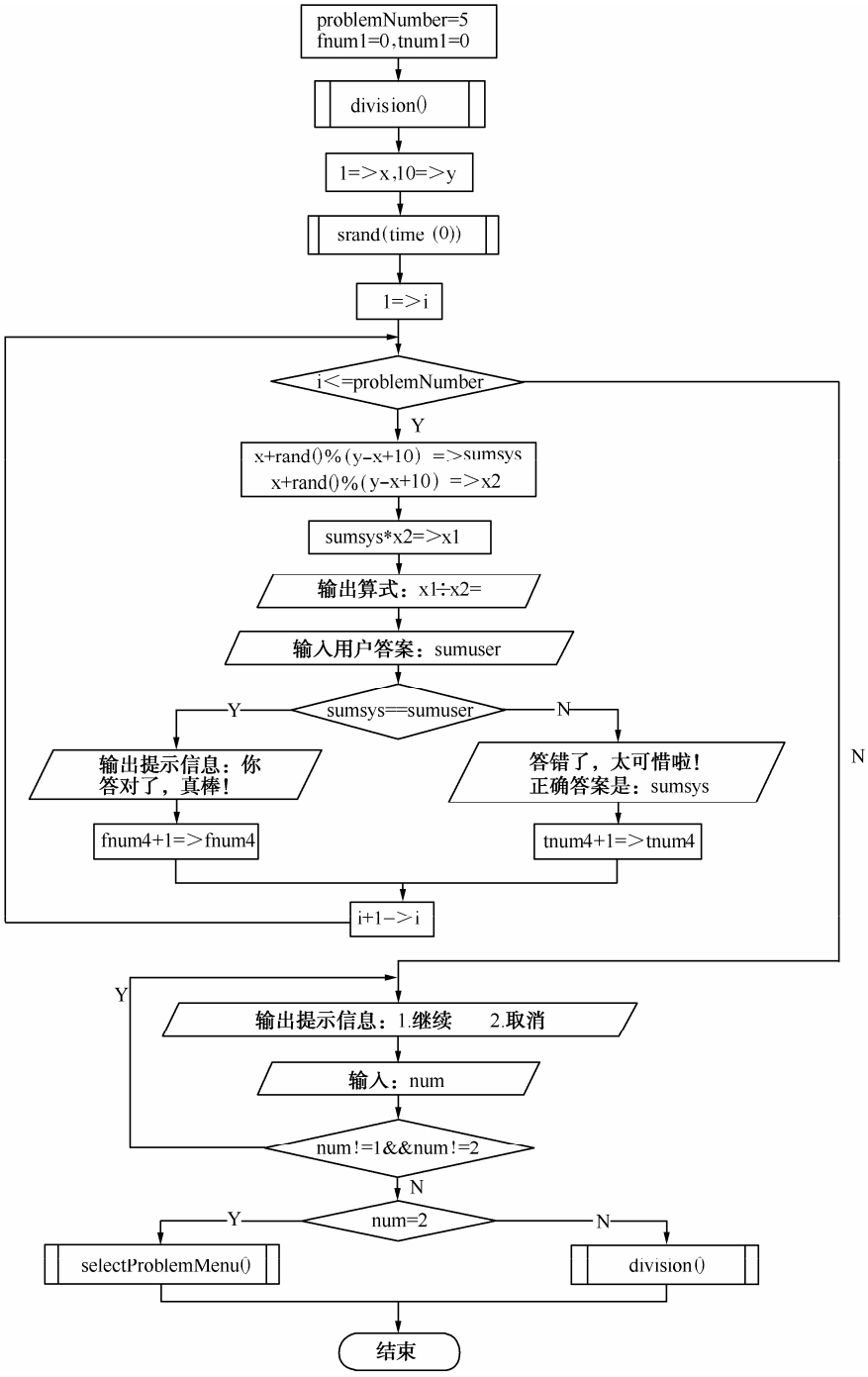


图 2.39　除法题库模块的设计流程图

```
void selectProblemMenu();        //声明 void selectProblemMenu 函数
void addition()                  //加法函数的声明
void subtration()                //减法函数的声明
void multiplication()            //乘法函数的声明
void division()                  //除法函数的声明

//四则题库界面代码
void selectProblemMenu()
{
    int num;
    system("cls");
    printf("\n\n");
    printf("\t\t*****************************************************\n\n");
    printf("\t\t                  四则题库                           \n\n");
    printf("\t\t*****************************************************\n\n");
    printf("\n\n");
    printf("\t\t\t\t 1. 加 法 题 库\n\n");
    printf("\t\t\t\t 2. 减 法 题 库\n\n");
    printf("\t\t\t\t 3. 乘 法 题 库\n\n");
    printf("\t\t\t\t 4. 除 法 题 库\n\n");
    printf("\t\t\t\t 0. 返回主控界面\n\n");

    printf("\n\t\t 请 选 择 操 作(0-4):");
    scanf("%d",&num);

    switch(num)
    {
            case 0:
                system("cls");
                showMainMenu();
                break;
            case 1:
                system("cls");
                addition();//加法题库
                break;
            case 2:
                system("cls");
                subtration();//减法题库
                break;
            case 3:
                system("cls");
                multiplication();//乘法题库
                break;
            case 4:
```

```
                    system("cls");
                    division();//除法题库
                    break;
                default:
                    printf("\n\t\t 请按任意键重新选择！！！");
                    getchar();
                    system("cls");
                    selectProblemMenu();
            }
}

//加法题库
void addition()
{
    int i;//控制变量
    int num; //判断变量
    int x=1,y=10;//1-10
    int x1,x2;//操作数
    int sumsys,sumuser;//sumsys 为正确答案,sumuser 为用户答案
    srand(time(0));
    for(i=1;i<=problemNumber;i++)
    {

        x1=x+rand()%(y-x+10);
        x2=x+rand()%(y-x+10);

        sumsys=x1+x2;

        printf("\n\t\t\t%d+%d=",x1,x2);
        scanf("%d",&sumuser);

        if(sumsys==sumuser)
        {
            printf("\t\t\t\t\t 你答对了，真棒！\n");
            tnum1++;
        }
        else
        {
            printf("\t\t\t\t\t 答错了，太可惜啦！正确答案是:  %d \n",sumsys);
            fnum1++;
        }
}//循环结束

    do
    {
        printf("\n\n\t\t\t==========================\n");
        printf("\t\t\t    1.继续          2.取消\n");
```

```
            printf("\t\t\t============================\n");
            printf("请选择(1 或 2)：");
            scanf("%d",&num);
        }while(num!=1&&num!=2);
        if(num==2)
        {
            system("cls");
            selectProblemMenu();
        }
        else
        {
            system("cls");
            addition();
        }
        system("cls");
    }

    //减法题库
    void subtration()
    {
        int i;  //控制变量
        int num;
        int x=1,y=10;
        int x1,x2,t;//操作数
        int sumsys,sumuser;//sumsys 为正确答案,sumuser 为用户答案
        srand(time(0));
        for(i=1;i<=problemNumber;i++)
        {
            x1=x+rand()%(y-x+10);
            x2=x+rand()%(y-x+10);
            if(x1<x2)
            {
                t=x1;
                x1=x2;
                x2=t;
            }
            sumsys=x1-x2;

            printf("\n\t\t\t%d-%d=",x1,x2);
            scanf("%d",&sumuser);
            if(sumsys==sumuser)
            {
                printf("\t\t\t\t\t 你答对了，真棒！\n");
                tnum2++;
            }
            else
            {
```

```
                printf("\t\t\t\t\t 答错了,太可惜啦!正确答案是:   %d \n",sumsys);
                fnum2++;
            }
        }//循环结束

        do
        {
            printf("\n\n\t\t\t==========================\n");
            printf("\t\t\t     1.继续            2.取消\n");
            printf("\t\t\t==========================\n");
            printf("请选择(1 或 2)：");
            scanf("%d",&num);
        }while(num!=1&&num!=2);
        if(num==2)
        {
            system("cls");
            selectProblemMenu();
        }
        else
        {
            system("cls");
            subtration();
        }
        system("cls");
}

//乘法题库
void multiplication()
{
        int i;//控制变量
        int num;
        int x=1,y=10;
        int x1,x2;//操作数
        int sumsys,sumuser;//sumsys 为正确答案,sumuser 为用户答案
        srand(time(0));
        for(i=1;i<=5;i++)
        {
            x1=x+rand()%(y-x+1);
            x2=x+rand()%(y-x+1);
            sumsys=x1*x2;

            printf("\n\t\t\t%d×%d=",x1,x2);
            scanf("%d",&sumuser);
            if(sumsys==sumuser)
            {
                printf("\t\t\t\t\t 你答对了,真棒!\n");
```

```
                tnum3++;
            }
            else
            {
                printf("\t\t\t\t\t 答错了,太可惜啦!正确答案是:   %d \n",sumsys);
                fnum3++;
            }
        }//循环结束

        do
        {
            printf("\n\n\t\t\t==========================\n");
            printf("\t\t\t     1.继续            2.取消\n");
            printf("\t\t\t==========================\n");
            printf("请选择(1 或 2)：");
            scanf("%d",&num);
        }while(num!=1&&num!=2);
        if(num==2)
        {
            system("cls");
            selectProblemMenu();
        }
        else
        {
            system("cls");
            multiplication();
        }
        system("cls");
    }

    //除法题库
    void division()

    {
        int i;
        int num; //控制变量
        int x=1,y=10;
        int x1,x2;//操作数
        int sumsys,sumuser;//sumsys 为正确答案,sumuser 为用户答案
        srand(time(0));
        for(i=1;i<=problemNumber;i++)
        {
            printf("\n");

            //实现整除
            sumsys=x+rand()%(y-x+10);//商
            x2=x+rand()%(y-x+1);  //除数不为 0
```

```
            x1=x2*sumsys;                    //被除数=除数*商

            printf("\n\t\t\t%d÷%d=",x1,x2);
            scanf("%d",&sumuser);
            if(sumsys==sumuser)
            {
                printf("\t\t\t\t\t 你答对了,真棒!\n");
                tnum4++;
            }
            else
            {
                printf("\t\t\t\t\t 答错了,太可惜啦!正确答案是:   %d \n",sumsys);
                fnum4++;
            }
        }//循环结束
        do
        {
            printf("\n\n\t\t\t==========================\n");
            printf("\t\t\t     1.继续          2.取消\n");
            printf("\t\t\t==========================\n");
            printf("请选择(1 或 2)：");
            scanf("%d",&num);
        }while(num!=1&&num!=2);
        if(num==2)
        {
            system("cls");
            selectProblemMenu();
        }
        else
        {
            system("cls");
            division();
        }
        system("cls");
    }
```

1．逻辑运算符和逻辑表达式

1）逻辑运算符及其优先次序

C 语言中提供下列三种逻辑运算符。

（1）&&：与运算。

（2）||：或运算。

（3）!：非运算。

与运算符&&和或运算符||均为双目运算符，具有左结合性。非运算符!为单目运算符，具有右结合性。逻辑运算符与其他运算符的优先级关系可表示如下：

!（非）→算术运算符→关系运算符→&&和 ||→赋值运算符

低 ——————————→ 高

由上可知，非、与、或的优先顺序为：!（非）→&&（与）→||（或）。

2）逻辑运算的值

逻辑运算的值也为“真”和“假”两种，用“1”和“0”来表示，其求值规则如下。

与运算 &&：参与运算的两个量都为真时，结果才为真，否则为假。

例如：15>0 && 14>4。

由于 15>0 为真，14>4 也为真，相与的结果也为真。

或运算||：参与运算的两个量只要有一个为真，结果就为真。两个量都为假时，结果为假。

例如：15>0||15>5。

由于 15>0 为真，相或的结果也就为真。

非运算!：参与运算量为真时，结果为假；参与运算量为假时，结果为真。

例如：!（15>0）。

由于 15>0 为真，非运算后的结果也就为假。

虽然 C 编译在给出逻辑运算值时，以“1”代表“真”，以“0”代表“假”，但反过来，在判断一个量是为“真”还是为“假”时，以“0”代表“假”，以非“0”的数值作为“真”。

例如：15&&5。

由于 15 和 5 均为非“0”，因此表达式“15&&5”的值为“真”，即为 1。

又如：15||0。

由于 15 为非“0”，因此表达式“15||0”的值为“真”，即为 1。

通过以上的分析，我们总结出如表 2.32 所示的逻辑运算的真值表，用它表示当 a 和 b 为不同组合时，各种逻辑运算所得到的值。

表 2.32　逻辑运算的真值表

A	b	!a	!b	a&&b	a||b
真	真	假	假	真	真
真	假	假	真	假	真
假	真	真	假	假	真
假	假	真	真	假	假

3）逻辑表达式

逻辑表达式是指用逻辑运算符将 1 个或多个表达式连接起来，进行逻辑运算的式子。在 C 语言中，用逻辑表达式表示多个条件的组合。

逻辑表达式的一般形式为：

表达式　逻辑运算符　表达式

其中的表达式可以又是逻辑表达式，从而组成了嵌套的情形。

例如：（a&&b）&&c。

根据逻辑运算符的左结合性，上式也可写为：a&&b&&c。

逻辑表达式的值是式中各种逻辑运算的最后值，以“1”和“0”分别代表“真”和“假”。

例如，下面的表达式都是逻辑表达式：

```
(x>=0) && (x<210)
(x<9) || (x>3)
! (x= =0)
(year%4==0)&&(year%100!=0)||(year%400==0)
```

【例 2.24】

```
main(){
    char c='h';
    int i=10,j=20,h=30;
    float a=5e+3,b=0.68;
    printf("%d,%d\n",!a*!b,!!!a);
    printf("%d,%d\n",a||i&&j-3,i<j&&a<b);
    printf("%d,%d\n",i==28&&c&&(j=29),a+b||i+j+h);
}
```

在本例中，因!a 和!b 分别为 0，!a*!b 也为 0，故其输出值为 0。因 a 为非 0，故!!!a 的逻辑值为 0。对 a|| i && j-3 式，先计算 j-3 的值为非 0，再求 i && j-3 的逻辑值为 1，故 a||i&&j-3 的逻辑值为 1。对于 i<j&&a<b 式，由于 i<j 的值为 1，而 a<b 为 0，故表达式的值为 1，0 相与，最后为 0；对于 i=28&&c&&(j=29)式，由于 i=28 为假，即值为 0，该表达式由两个与运算组成，所以整个表达式的值为 0。对于 a+b||i+j+h 式，由于 a+b 的值为非 0，故整个或表达式的值为 1。

2．多分支结构

1）if-else-if语句

一般形式为：

```
if(表达式 1)
        〈语句 1〉;
  else  if(表达式 2)
        〈语句 2〉;
  else  if(表达式 3)
        〈语句 3〉;
        …
  else  if(表达式 m)
        〈语句 m〉;
  else
      〈语句 n〉;
```

该语句的执行过程是：依次判断表达式的值，当出现某个值为真时，则执行其对应的语句；然后跳到整个 if 语句之外继续执行程序。如果所有的表达式均为假，则执行语句 n；然后继续执行后续程序。if-else-if 语句的执行过程如图 2.40 所示。

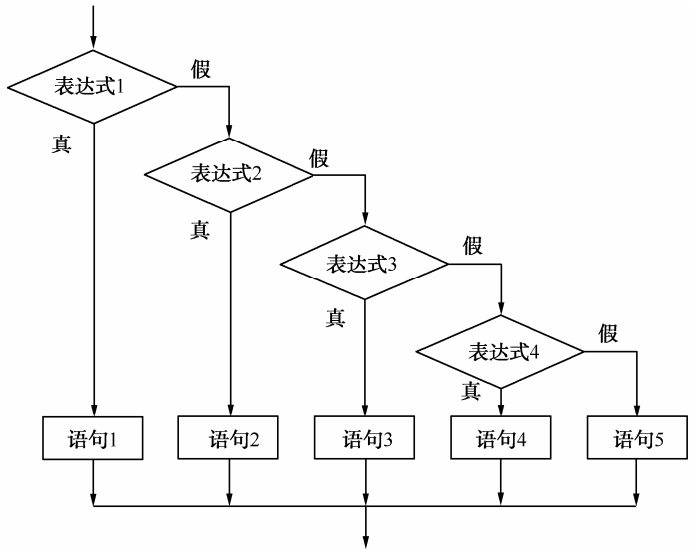


图 2.40　if-else-if 语句的执行过程

【例 2.25】

```
#include"stdio.h"
main()
{
    char s;
    printf("请输入一个字符: ");
    s=getchar();
    if(s<32)
       printf("这是一个控制符\n");
    else if(s>='0'&&s<='9')
       printf("这是一个数\n");
    else if(s>='A'&&s<='Z')
       printf("这是一个大写字母\n");
    else if(s>='a'&&s<='z')
       printf("这是一个小写字母\n");
    else
       printf("这是一个其他形式的字符\n");
}
```

本例的功能是判断从键盘上输入字符的类别。可以根据输入字符的 ASCII 码来判断其类型。由 ASCII 码表可知，ASCII 值小于 32 的为控制字符。在“0”和“9”之间为数字，在“A”和“Z”之间为大写字母，在“a”和“z”之间为小写字母，其余则为其他用字符。这是一个多分支选择的问题，用 if-else-if 语句编程，判断输入字符 ASCII 码所在的范围，分别给出不同的输出。例如，输入为“A”，输出显示它为大写字符。

说明：

（1）在三种形式的 if 语句中，if 关键字之后均为表达式。该表达式通常是逻辑表达式或关系表达式，但也可以是其他用表达式，如赋值表达式等，甚至也可以是一个变量。

例如：

```
if(x=3) 语句;
if(y) 语句;
```

都是允许的。只要表达式的值为非 0，即为“真”。

例如在 if（a=5）;语句中，因为表达式的值永远为非 0，所以其后的语句总是要执行的。当然，这种情况在程序中不一定会出现，但在语法上是合法的。

（2）在 if 语句中，条件判断表达式必须用括号括起来，在语句之后必须加分号。

（3）在 if 语句的三种形式中，所有的语句应为单个语句。如果想在满足条件时执行一组（多个）语句，则必须把这一组语句用{}括起来组成一个复合语句。但是，要注意在}之后不能加分号。

例如：

```
if(x>y)
{
x=x-10;
y=y+10;
  }
else
{
x=x+10;
y=y-10;
}
```

2）switch语句

一般形式为：

```
switch(表达式)
        {   case     常量表达式 1:语句组;[break;]
            case     常量表达式 2:语句组;[break;]
              …
            case     常量表达式 n:语句组;[break;]
            [default:语句组;[break; ]]
        }
```

switch 语句的执行过程是：

（1）当 switch 后面“表达式”的值，与某个 case 后面的“常量表达式”的值相同时，就执行该 case 后面的语句（组）；当执行到 break 语句时，跳出 switch 语句，转向执行 switch 语句的下一条。

（2）如果没有任何一个 case 后面的“常量表达式”的值与“表达式”的值匹配，则执行 default 后面的语句（组）；然后，再执行 switch 语句的下一条。

switch 语句的执行过程如图 2.41 所示。

说明：

（1）switch 后面的“表达式”，可以是 int、char 和枚举型中的一种。

（2）每个 case 后面“常量表达式”的值，必须各不相同，否则会出现相互矛盾的现象（即对表达式的同一值，有两种或两种以上的执行方案）。

（3）case 后面的常量表达式仅起语句标号作用，并不进行条件判断。因为系统一旦找到入口标号，就从此标号开始执行，不再进行标号判断，所以必须加上 break 语句，以便结束 switch 语句。

（4）在 case 后，允许有多个语句，可以不用{}括起来。

（5）各 case 和 default 子句的先后顺序可以变动，而不会影响程序执行结果。

（6）default 子句可以省略不用。

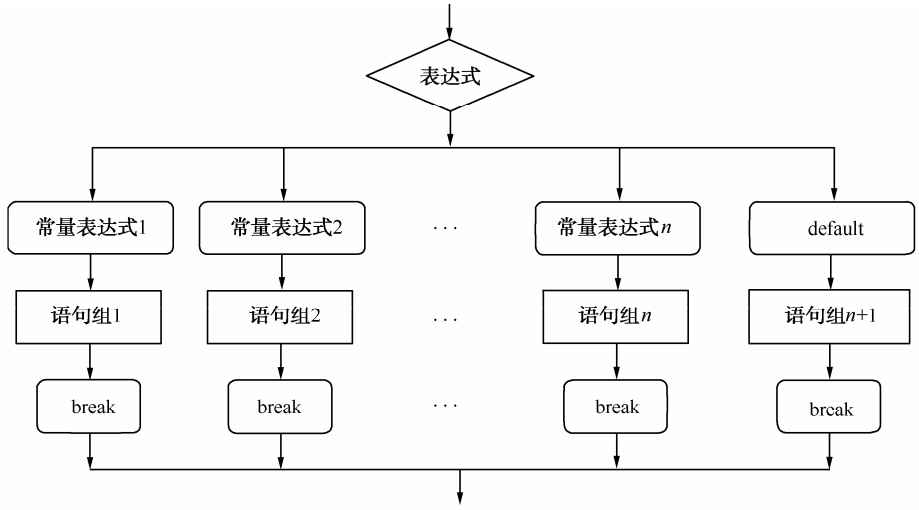


图 2.41　switch 语句的执行过程

【例 2.26】

```
main()
{
int month;
printf("请输入一个月份值(1-12)" );
scanf("% d"，&month);
switch(month)          /*根据 month 的当前取值，做出多分支选择*/
{
    case 1:printf("January\n");break;
    case 2:printf("February\n");break;
    case 3:printf("March\n");break;
    case 4:printf("April\n");break;
    case 5:printf("May\n");break;
    case 6:printf("June\n");break;
    case 7:printf("July\n");break;
    case 8:printf("August\n");break;
    case 9:printf("Septembert\n");break;
    case 10:printf("October\n");break;
    case 11:printf("November\n");break;
    case 12:printf("December\n");break;
    default:printf("Data Error！ \n");break;
  }
}
```

该程序的功能是从键盘输入一个月份（1～12），并显示该月份的英文名称。

3．随机函数和随机种子

1）sand()函数

功能：用于产生一个伪随机数。

例如：x+rand()%(y-x+1)可以产生 x～y 的随机数，其中 x 和 y 为整型数据。

2）srand()函数

功能：srand(seed)函数用于给 rand()函数设定随机种子。

说明：

（1）rand()函数产生的是伪随机数，不是真正意义上的随机数，这个伪随机数是根据一个公式算出来的，每次运行程序，产生的伪随机数都一样。

（2）要想产生真正意义上的随机数，应将 srand()和 rand()函数配合使用，srand()函数用来设置随机数的种子，一般以时间作为种子。

（3）srand()和 rand()函数应该组合使用。一般来说，srand()函数是对 rand()函数进行设置。

例如：

```
srand(time(0));   //以系统时间作为随机种子
```

4．for循环结构

一般形式为：

```
for (表达式 1;表达式 2;表达式 3)
〈语句〉
```

该语句的执行过程是：

（1）先求解表达式 1。

（2）求解表达式 2，若其值为真（非 0），则执行 for 语句中指定的内嵌语句，然后执行下面第（3）步；若其值为假（0），则结束循环，转到第（5）步。

（3）求解表达式 3。

（4）转回上面第（2）步继续执行。

（5）循环结束，执行 for 语句下面的一个语句。

for 循环执行过程如图 2.42 所示。

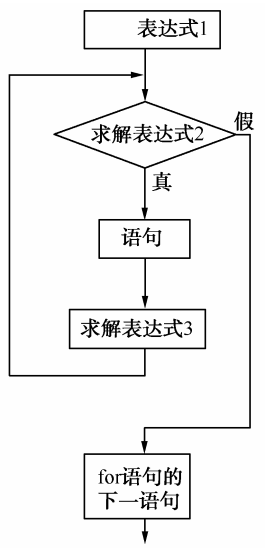


图 2.42　for 循环执行过程

for 语句最简单的应用形式也是最容易理解的形式如下：

for（循环变量赋初值;循环条件;循环变量增量）语句

循环变量赋初值总是一个赋值语句，它用来给循环控制变量赋初值；循环条件是一个关系表达式，它决定什么时候退出循环；循环变量增量，定义循环控制变量每循环一次后按什么方式变化。这三个部分之间用“;”分开。

例如：

```
for(i=1; i<=50; i++)sum=sum+i;
```

先给 i 赋初值 1，判断 i 是否小于等于 100。若是，则执行语句，之后值增加 1。再重新判断，直到条件为假，即 i>50 时，结束循环。

相当于：

```
    i=1;
    while（i<=50）
    { sum=sum+i;
      i++;
}
```

对于 for 循环中语句的一般形式，就是如下的 while 循环形式：

```
表达式 1;
while(表达式 2)
    {语句
     表达式 3;
}
```

说明：

（1）在 for 语句中，条件测试总是在循环开始时进行；如果循环体部分是由多个语句组成的，则必须用左、右花括号括起来，使其成为一个复合语句。

（2）在 for 语句中，表达式 1 通常是为循环变量赋初值的表达式；表达式 2 是控制循环的表达式，一般是关系表达式或逻辑表达式，但也可是数值表达式或字符表达式，只要其值是非 0，就执行循环体；表达式 3 通常是改变循环变量值的表达式。表达式 1 和表达式 3 既可以是一个简单的表达式，也可以是逗号表达式。

例如：

```
for(sum=0;i<=50;i++)sum=sum+i;
for(sum=0,i=1;i<=100;i++)sum=sum+i;
for(i=0;(c=getchar())!='\n';i+=c);
for(;(c=getchar())!='\n';)
printf("%c",c);
```

（3）for 语句的表示形式相当灵活，可以部分或全部省略，但“;”不能省略，如 for(;;)。

例如：

```
for(;;)语句
```

相当于：

```
while(1)语句
```

（1）随时产生 1～7 的整数，输出相应星期的英文表示。

（2）使用 for 循环打印九九表。

（3）优化“四则题库模块”的算法，大胆用自己的想法修改该模块的功能。

建议教学时以组为单位选出代表对本组优秀作品进行阐述并演示，最后选择出最佳设计 1～2 件，作为最后的四则题库算法。

拓展任务

设计体育彩票销售系统随机投注模块，添加顾客后，自动产生 7 位的投注号码，具体效果如图 2.43 和图 2.44 所示。

图 2.43　随机投注界面

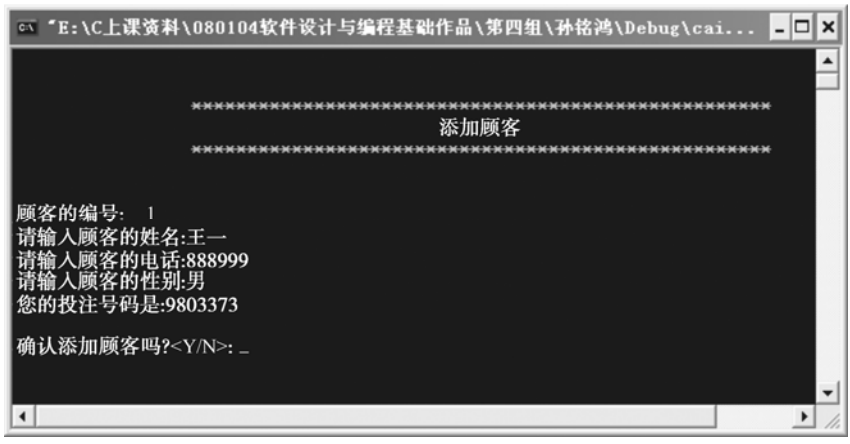


图 2.44　产生投注号码

2.4.5　子任务五：评分系统模块的功能实现

任务描述

在主控模块界面设计的基础上，实现评分系统模块的功能。评分系统模块的主要功能是，实现学生进行题库训练后，显示当次训练的成绩清单，即训练的总题数、正确的题数和错误的题数，根据答题情况给出相应分数。该模块的目标是，让使用者了解自身对各题型掌握的情况，以便根据自己的实际情况选择训练的强度，从而提高四则运算的技能。

（1）基本流程如表 2.33 所示。

表 2.33　评分系统模块的基本流程

1．评分系统函数	2．呈现评分系统界面，如图 2.45 所示
3．显示当前答题总量、正确题目数量、错误题目数量和得分	4．显示“1.返回主控界面　2.退出：”提示信息
5．输入选择（键盘输入整型数据）	6．判断 num 为 1 则返回主控界面，为 2 则执行第 7 步，否则执行第 4 步
7．显示“确认要退出吗？(Y/N)”提示信息	8．输入选择（键盘输入字符型数据）
9.判断输入的值为'Y'或'y'则退出系统，为'N'或'n'则返回主控界面，否则执行第 7 步	

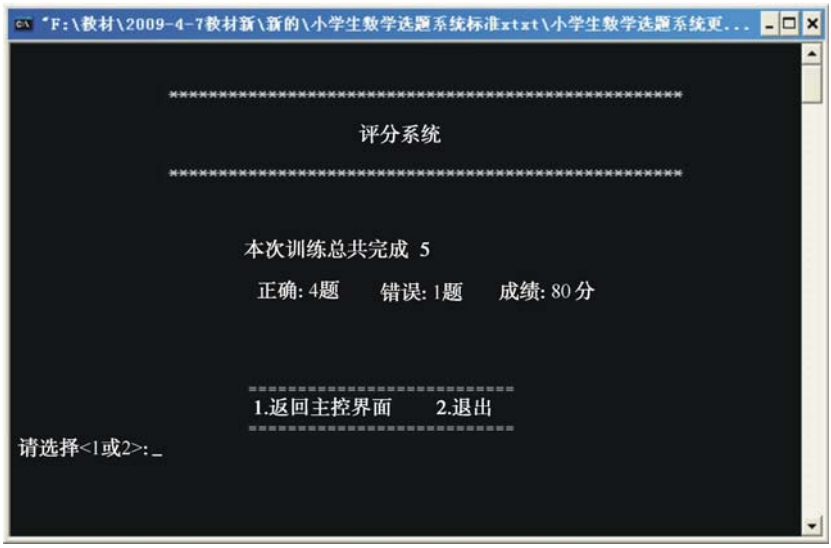


图 2.45　评分系统模块

（2）自然语言描述。

S1：定义整型变量——sum 为题目总数量，tsum 为正确题目总数量，fsum 为错误题目总数量，num 为是否继续操作的控制变量；定义字符型变量——choi 为是否退出系统操作的控制变量。

S2：计算 sum、tsum、fsum。

S3：输出评分系统的结果。

S4：输出显示当前答题总量、正确题目数量、错误题目数量和得分（tsum*100/sum）。

S5：输出是否“退出”的提示信息。

S6：用户输入选择。

S7：判断用户输入的信息。如果为 1，则返回主控界面；如果为 2，则执行 S8，否则执行 S5。

S8：输出是否“确认退出”的提示信息。

S9：判断用户输入的信息。如果为'Y'或'y'，则退出系统；如果为'N'或'n'，则返回主控界面，否则执行 S8。

（3）设计流程图如图 2.46 所示。

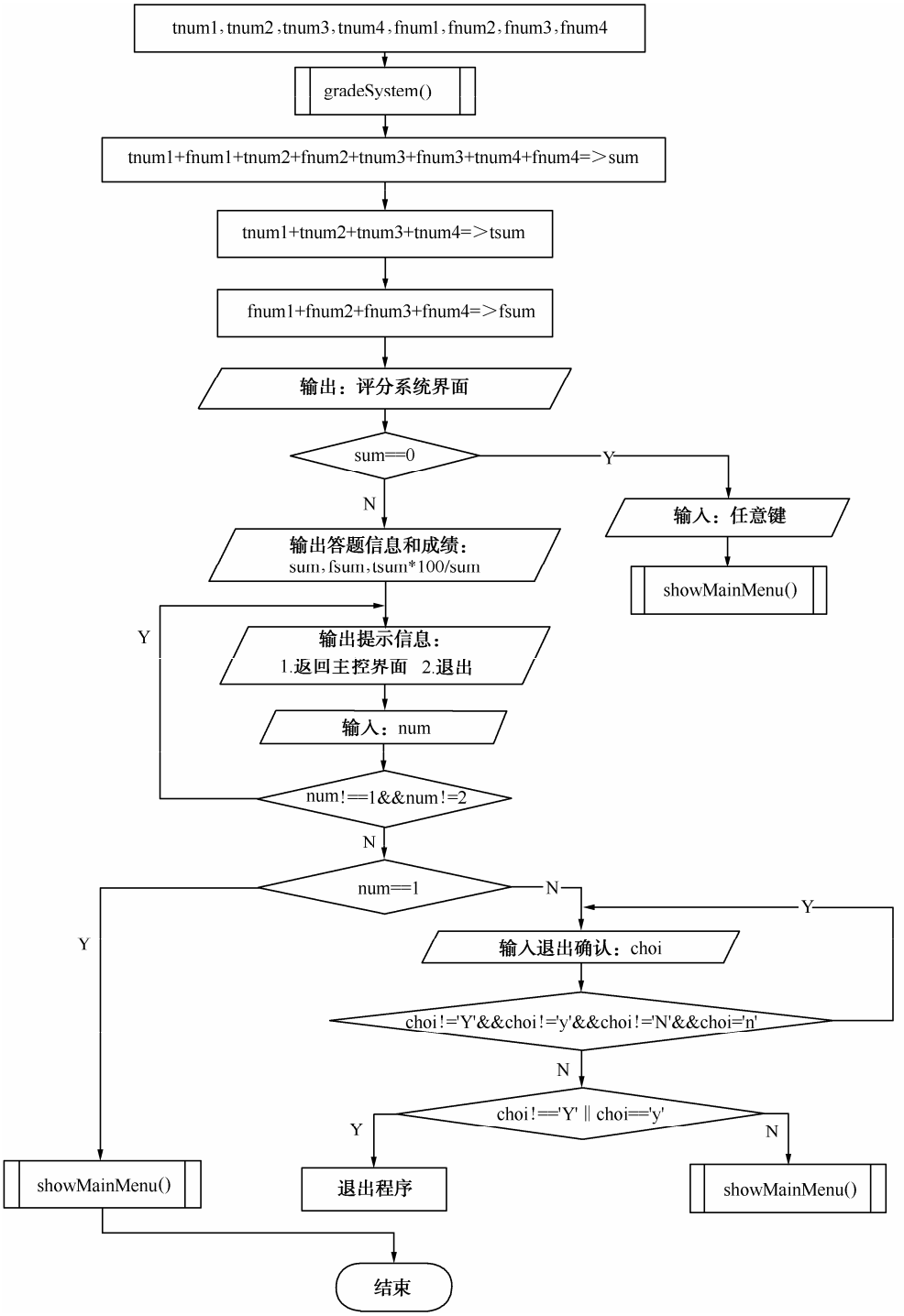


图 2.46　评分模块的设计流程图

```
void gradeSystem();//评分系统函数的声明
void gradeSystem()//评分系统函数的定义
{
    int sum,num,tsum,fsum;
    char choi;
    sum=tnum1+fnum1+tnum2+fnum2+tnum3+fnum3+tnum4+fnum4;
    tsum=tnum1+tnum2+tnum3+tnum4;
    fsum=fnum1+fnum2+fnum3+fnum4;
    system("cls");
    printf("\n\n");
    printf("\t\t**************************************************\n\n");
    printf("\t\t                    评分系统                      \n\n");
    printf("\t\t**************************************************\n\n");
    printf("\n\n");
    if(sum==0)
    {
        printf("您还没有开始进行答题，按任意键回到主控界面");
        getchar();
        getchar();
        system("cls");
        showMainMenu();
    }
    printf("\t\t\t 本次训练总共完成 %d   \n\n",sum);
    printf("\t\t\t 正确: %d 题    ",tsum);
    printf("错误: %d 题 ",fsum);
    printf("成绩: %d 分 ",tsum*100/sum);
    do
    {
        printf("\n\n\n\n\n\t\t\t==========================\n");
        printf("\t\t\t 1.返回主控界面        2.退出\n");
        printf("\t\t\t==========================\n");
        printf("请选择(1 或 2)：");
        scanf("%d",&num);
    }while(num!=1&&num!=2);
    if(num==1)
    {
        system("cls");
        showMainMenu();
    }
    else
    {
        do
        {
            printf("\t\t 确认要退出吗？(Y/N)：");
```

```
                scanf(" %c",&choi);
            }while(choi!='Y'&&choi!='y'&&choi!='N'&&choi!='n');
            if(choi=='Y' || choi=='y')
                exit(0);//退出
            else
            {
                system("cls");
                showMainMenu();
            }
        }

    }
```

1．条件运算符和条件表达式

在条件语句中，只执行单个的赋值语句时，可使用条件表达式来实现，不但使程序简洁，也提高了运行效率。条件运算符为?和:，它是一个三目运算符，即有三个参与运算的量。

1）一般形式

```
表达式 1?  表达式 2:表达式 3
```

2）用法与运算规则

如果表达式 1 的值为真，则以表达式 2 的值作为条件表达式的值，否则以表达式 2 的值作为整个条件表达式的值。

例如条件语句

```
if(a>b)   max=a;
else max=b;
```

可用条件表达式写为:

```
max=(a>b)?a:b;
```

执行该语句的语义是:

```
如 a>b 为真,则把 a 赋予 max,否则把 b  赋予 max。
```

3）说明

（1）条件运算符的运算优先级低于关系运算符和算术运算符，但高于赋值符。因此，对于 max=(a>b)?a:b 可以去掉括号而写为 max=a>b?a:b。

（2）条件运算符?和:是一对运算符，不能分开单独使用。

（3）条件运算符的结合方向是自右至左。

例如：

```
a>b?a:c>d?c:d
```

应理解为：

```
a>b?a:(c>d?c:d)
```

这也就是条件表达式嵌套的情形，即其中的表达式 3 又是一个条件表达式。

【例 2.27】

```
//输入两个数，并将中较大者显示出来
main()
{
    int a,b,max;
    printf("\n  请输入两个数:     ");
scanf("%d%d",&a,&b);
max=a>b?a:b;
    printf("max=%d",max);
}
```

用条件表达式对上例重新编程，输出两个数中的大数。

2．自增（++）、自减（--）运算

1）作用

自增运算使单个变量的值增 1，自减运算使单个变量的值减 1。

2）用法与运算规则

（1）前置运算——运算符放在变量之前：++变量、--变量。

先使变量的值增（或减）1，然后再以变化后的值参与其他用运算，即先增减、后运算。

（2）后置运算——运算符放在变量之后：变量++、变量--。

变量先参与其他用运算，然后再使变量的值增（或减）1，即先运算、后增减。

【例 2.28】

```
//自增、自减运算符的用法与运算规则示例
main()
{ int x=6, y;
   printf("x=%d\n",x);
   y =++x;
   printf("y=++x: x=%d,y=%d\n",x,y);
   y = x--;
   printf("y=x--: x=%d,y=%d\n",x,y);
}
```

程序运行结果：

```
x=6
y=++x: x=7,y=7
y=x--: x=6,y=7
```

3）说明

（1）自增、自减运算，常用于循环语句中，使循环控制变量加（或减）1，以及指针变量中，使指针指向下（或上）一个地址。

（2）自增、自减运算符，不能用于常量和表达式。

例如，9++、--(x+y)等都是非法的。

（3）在表达式中，因为连续使同一变量进行自增或自减运算时，很容易出错，所以最好避免这种用法。

3．逗号运算（ ,）及其表达式

C 语言提供一种用逗号运算符“,”连接起来的式子，称为逗号表达式。逗号运算符又称顺序求值运算符。

1）一般形式

```
表达式 1, 表达式 2,…, 表达式 n
```

2）用法与运算规则

自左至右，依次计算各表达式的值，“表达式 n”的值即为整个逗号表达式的值。

例如，逗号表达式“a=2*5,a*3”的值等于 30 的求解过程是：

先求解 a=2*5，得 a=10；再求 a*3=30，逗号表达式的值等于 30。

再例如，逗号表达式“(a=2*5,a*3),a+5”的值等于 15 的求解过程是：

先求解 a=2*5，得 a=10；再求 a*3=30；最后求解 a+5=15，逗号表达式的值等于 15。

3）说明

并不是任何地方出现的逗号，都是逗号运算符。要分析是它是作用是用做分隔符还是运算符。

4．If语句嵌套

当 if 语句中的执行语句又是 if 语句时，就构成了 if 语句嵌套。共有以下三种嵌套形式。

（1）嵌套形式 1 为：

```
if(表达式 1)
if(表达式 2)语句 1;
else        语句 2;
else  语句 3;
```

该语句的执行过程是：执行流程如图 2.47 所示，在 if-else 语句中包含另一个 if-else 语句，即第一个 else 与第二个 if 结合，而最后一个 else 与第一个 if 结合。

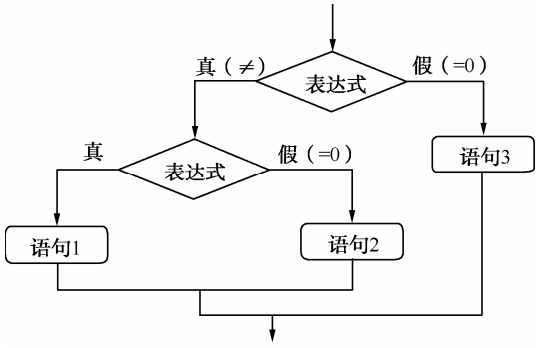


图 2.47　嵌套形式 1 的执行流程

（2）嵌套形式 2 为：

```
if(表达式 1)
        {if(表达式 2)语句 1 ;}
        else 语句 2;
```

该语句的执行过程是：执行流程如图 2.48 所示，在 if-else 语句中包含一个单分支结构复合语句，即 else 与第一个 if 结合。因为第二个 if 在复合语句中，复合语句是一条语句，不能与复合语句外的 else 结合。如果把{}去掉，则 else 与第二个 if 结合。

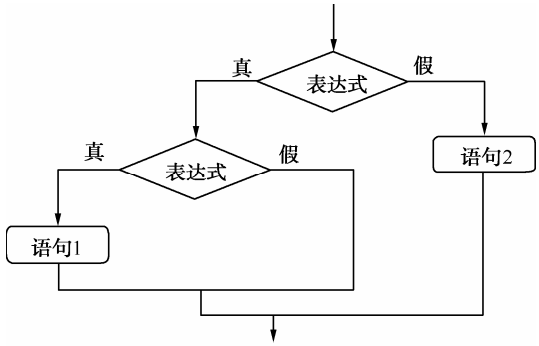


图 2.48　嵌套形式 2 的执行流程

（3）嵌套形式 3 为：

```
if(表达式 1)语句 1;
    else if(表达式 2)语句 2;
    else 语句 3;
```

该语句的执行过程是：执行流程如图 2.49 所示，在 if-else 语句的 else 后紧跟另一个 if-else 语句。C 语言规定：else 总是与它前面最近的同一复合语句内的不带 else 的 if 结合。在 if 语句嵌套形式 2 中可以看到，else 与 if 在同一复合语句内才能结合。

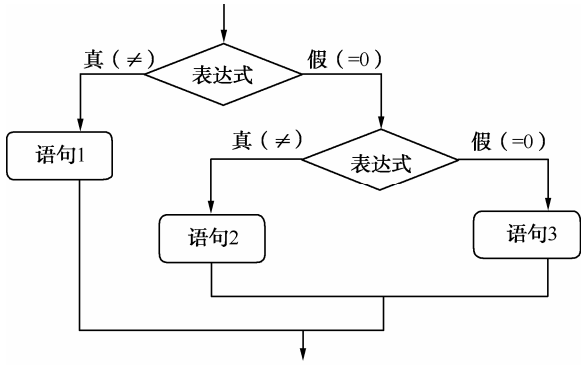


图 2.49　嵌套形式 3 的执行流程

【例 2.29】

```
//比较两个数的大小关系示例
main(){
    int A,B;
    printf("请输入 A,B:");
    scanf("%d%d",&A,&B);
    if(A==B) printf("A=B\n");
    else if(A>B)    printf("A>B\n");
    else    printf("A<B\n");
```

```
}
```

本例中采用了 if 语句的嵌套结构。采用嵌套结构实质上是为了进行多分支选择，实际上有三种选择，即 A>B、A<B 或 A=B。这种问题用 if-else-if 语句也可以完成，而且程序更加清晰。因此，在一般情况下较少使用 if 语句的嵌套结构，以使程序更便于阅读理解。

【例 2.30】

```
//判断男士与女士因退休标准不同而采用的嵌套分支结构示例
#include<stdio.h>
 main()
 {
  char sex;
  int age;
  printf("请输入性别");
  printf("(男士用 M,女士用 F):");
  scanf("%c",&sex);
  printf("请输入年龄:");
  scanf("%d",&age);
  if(sex=='M'||sex=='m')/*是男士*/
  {
       if(age>=60)
       {
          printf("已退休\n");
       }
      else
      {
          printf("在工作\n");
      }
  }
  else/*是女士*/
  {
       if(age>=55)
       {
          printf("已退休\n");
       }
      else
      {
          printf("在工作\n");

      }
  }
 }
```

即时训练

（1）编程实现任意输入三个整数，将其按从小到大的顺序输出。

（2）编程实现输入一个 4 位的整数，将其反向输出。

（3）求解方程 $ax^2+bx+c=0$。

（4）学校进行成绩分级管理，取消分数制，改为成绩分级评定。

具体办法是：90 分（含 90）以上为 A 类，80（含 80）～90 分为 B 类，70（含 70）～80 分为 C 类，60（含 60）～70 分为 D 类，60 分以下为 E 类。设计一个程序，对输入的成绩进行等级划分。

（5）自行设计“评分系统模块”，假设你是一名小学生，你最希望在进行了百题训练后想看到的评价，并将其设计出来。

拓展任务

对“体育彩票销售系统”的投注总数量、自行投注数量和随机投注数量进行统计。

2.5 任务五：“小学生数学选题系统”的测试与优化

任务描述

本次测试主要针对本小组开发的小学生数学选题系统的主要模块进行测试。

任务分析与设计

对在小学生数学选题系统需求规格说明书中列出的系统功能和性能都需要完成测试，在对测试工作期间发现的所有缺陷都需要改正并确认。

任务实现

自底向上依次进行单元测试（采用黑盒测试），组装测试（采用黑盒测试），测试用例的设计应包括合理的和不合理的输入条件。

（1）编写目的：为保证四则题库模块的各项功能可靠的实现，针对该模块内的各子模块进行测试。

（2）测试用例名如下。

stuPart1：四则题库模块的组装测试。

stuPart1.1：加法模块的测试。

stuPart1.2：减法模块的测试。

stuPart1.3：乘法模块的测试。

stuPart1.4：除法模块的测试。

（3）测试内容如表 2.34～表 2.38 所示。

表 2.34　四则题库模块的组装测试（stuPart1）

模　块　名	输　　入	输　　出
加法模块		
减法模块		
乘法模块		
除法模块		

表 2.35　加法模块的测试（stuPart1）

测试目标	验证加法模块能否进行正常的加法出题和判卷
方法	运行加法题库，在算式后输入答案
完成标准	显示有两个操作数的加法算式，若输入正确答案，则提示“真棒!”；若输入错误答案，则提示“真可惜”和正确答案。共显示 5 道题，每道题的操作数为 1～20 的整数
需要考虑的注意事项	

表 2.36　减法模块的测试（stuPart2）

测试目标	验证减法模块能否进行正常的减法出题和判卷
方法	运行减法题库，在算式后输入答案
完成标准	显示有两个操作数的减法算式，若输入正确答案，则提示“真棒!”；若输入错误答案，则提示“真可惜”和正确答案。共显示 5 道题，每道题的操作数为 1～20 的整数
需要考虑的注意事项	被减数大于减数

表 2.37　乘法模块的测试（stuPart3）

测试目标	验证乘法模块能否进行正常的乘法出题和判卷
方法	运行乘法题库，在算式后输入答案
完成标准	显示有两个操作数的乘法算式，若输入正确答案，则提示“真棒!”，若输入错误答案，则提示“真可惜”和正确答案。共显示 5 道题，每道题的操作数为 1～20 的整数
需要考虑的注意事项	

表 2.38　除法模块的测试（stuPart4）

测试目标	验证除法模块能否进行正常的除法出题和判卷
方法	运行除法题库，在算式后输入答案
完成标准	显示有两个操作数的除法算式，若输入正确答案，则提示“真棒!”，若输入错误答案，则提示“真可惜”和正确答案。共显示 5 道题，每道题的操作数为 1～20 的整数
需要考虑的注意事项	除数不能为 0

（4）执行测试，得出结论。

（5）修改 Bug 并再次测试修改处，优化软件。

1．测试的目的

测试是为了发现尽可能多的缺陷。在这里，缺陷是一种泛称，指功能的错误、功能低下、易用性差等。一个成功的测试在于发现了至今尚未发现的缺陷。

2．产品的内部测试（称为α测试）

（1）需要开发人员与独立的测试小组共同参与。

（2）“黑盒测试”必须由独立的测试人员执行。

3．β测试

软件产品正式发行前，邀请一些用户对产品进行测试，称为β测试。β测试的涉及面最广，最能反映用户的真实愿望，但花费的时间最长，不好控制。

4．软件测试过程

如果软件开发过程采用严格的瀑布模型，那么开发与测试应该有“V”形的对应关系。

通过图 2.50 中水平箭头对应阶段的反馈，软件本身及相关开发文档的质量不断提升。每一轮测试都重复这一步骤，软件的质量也就在这个过程中实现螺旋上升。

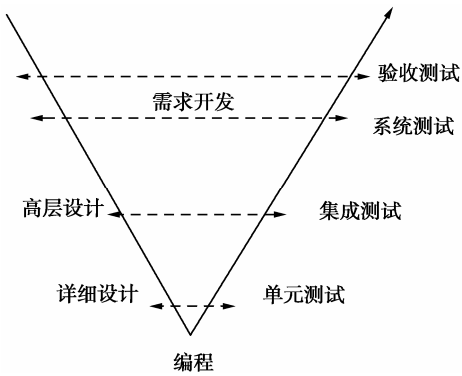


图 2.50　“V”形模型

5．测试策略

软件测试的计划性和系统性，决定了测试的一系列过程，这一系列过程称为测试策略。软件测试策略包括底层测试（Low Level Testing）和高层测试（High Level Testing），以证实主要系统功能满足用户需求。因此，按测试策略，软件测试被分为：

（1）单元测试（Unit Testing）。

（2）集成测试（Integration Testing）。

（3）验收测试（Validation Testing）。

（4）系统测试（System Testing）。

测试过程与类别如表 2.39 所示。

表 2.39　测试过程与类别

测 试 阶 段	主 要 依 据	测试人员、测试方式	主要测试内容
单元测试	系统设计文档	由开发小组执行白盒测试	接口测试、路径测试
集成测试	系统设计文档 需求文档	由开发小组执行白盒测试和黑盒测试	接口测试、路径测试
验收测试	系统设计文档	由用户执行黑盒测试	功能测试、健壮性测试、性能测试、用户界面测试、安全性测试、压力测试、可靠性测试、安装/反安装测试
系统测试	需求文档	由独立测试小组执行黑盒测试	

说明：

（1）白盒测试。测试人员（要求充分理解程序）直接在软件源程序上测试、修改、复测。

（2）黑盒测试。测试人员不必深入了解软件内部设计，只从终端正用户的角度按产品说明书从外部测试软件功能。

（3）灰盒测试。介于白盒、黑盒测试之间。

6．测试的基本原则

Zero Bug 是指软件没有任何 Bug。Good Enough 是指软件达到一定的质量要求可以停止测试。Zero 是不可能的，而要达到 Good Enough，就要制定最低标准和测试内容。测试的原则如下：

（1）不要试图穷举测试。

（2）开发人员不能既是运动员又是裁判员。

（3）要尽早执行软件测试。

（4）软件测试应该追溯需求。

（5）缺陷的“二八”原理（一般来说，软件的 80%缺陷集中在 20%的模块中）。

（6）缺陷具有免疫性（在缺陷修改后再用相同的测试用例测试，效果会很差）。

7．测试的工作过程

第一步：制定测试计划。该计划被批准后转向第二步。

（1）制定测试项目的标准。

（2）制定测试策略。

（3）指出测试的细节和要点。

第二步：设计测试用例。该用例被批准后转向第三步。

测试设计指根据系统说明书、系统设计文档、测试范围、技术特点和程序结构设计测试用例。

第三步：如果满足“启动准则”，则执行测试。

按照测试用例的要求进行人工操作或使用自动测试工具进行。由于界面测试主要考虑的是图元和色彩风格等元素，所以一般使用手工测试的方式。

第四步：撰写测试报告。

测试报告不仅要提供测试结果的实际数据，同时要对结果进行分析和评估。

第五步：消除软件缺陷。如果满足“完成准则”，则正常结束测试。

8．集成测试

增量式集成法：一段一段地扩展程序，一步一步地增大测试的范围，以使错误易于定位和纠正，使界面的测试完全彻底。下面讨论自底向上集成测试法。

自底向上集成测试是从“原子”模块（即软件结构最底层的模块）开始组装测试，因为测试到较高层模块时，所需的下层模块功能均已具备，所以不再需要桩模块。

自底向上集成测试的步骤如下：

（1）把底层模块组织成实现某个子功能的模块群（Cluster）。

（2）开发一个测试驱动模块，控制测试数据的输入和测试结果的输出。

（3）对每个模块群进行测试。

（4）删除测试使用的驱动模块，用较高层模块把模块群组织成为完成更大功能的新模块群，如图 2.51 所示。

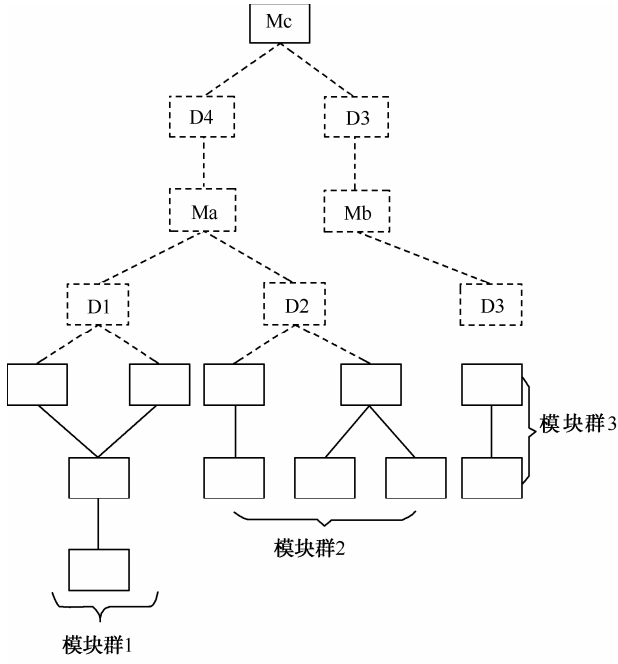


图 2.51　测试模块划分

即时训练

继续实现使用说明模块、四则题库模块、题量设置模块的、评分系统模块的单元测试和组装测试，以完成整个系统的测试。

拓展任务

实现整个小学生数学选题系统的测试和优化。

第 3 章　超市管理系统

3.1　任务一 :“超市管理系统”的项目背景

任务描述

以小组为单位，对超市管理系统的项目背景进行分析，弄清楚该项目主要实现哪些功能。

任务分析与设计

软件项目开发的关键是，了解所开发的项目背景，以开发出令用户满意的产品。

任务实现

1．编写目的

编写项目背景，使开发人员对该项目的主要功能一目了然，为以后的功能实现奠定基础。

2．项目背景文档

项目背景文档如表 3.1 所示。

表 3.1　项目背景文档

软件系统的名称	超市管理系统
本项目的任务提出者	软件技术专业指导老师
用户	小型超市管理者、营业员和库存管理员
开发者	软件技术专业开发小组
项目背景描述	随着信息技术的飞速发展，信息化管理已经引入并应用到各行业管理领域，尤其是零售业。放眼四周，各种形式的百货商场、大型仓储超市、便利店、连锁超市和专卖店等零售业不断出现，并不断改变、影响着我们的观念和生活方式。企业若想在激烈的市场经济中立足并胜出，就必须拥有一套完善的并适合自身特点的信息化管理系统，以实现降低企业成本、获取市场反馈信息、完善服务质量、提高经济效益、分析市场需求、制定销售计划和目标等管理目的。如何才能实现一个超市的信息化建设呢？除了必要的硬件设备外，还必须有一个优秀的信息管理系统软件的支持 超市管理系统是一套功能完善的管理信息系统，不仅能满足业务人员的日常事务处理的需要，提高企业经营全过程的数字化管理水平，还能满足管理人员对决策分析的需要，更能提高超市对公司经营反馈信息的响应速度，从而加快了公司资金的流通，减少了库存的积压，提高了经济效益。 在超市经营中，商品的管理至关重要。本系统对商品的管理包括进、销、存 3 个方面。超市管理系统对于超市的管理者、营业员、库存管理员等至关重要，不但减轻了他们的繁重工作，也大大提升了经营者管理的水平

3.2 任务二 :“超市管理系统”的需求分析

3.2.1 子任务一：编写项目计划书

任务描述

以小组为单位，编写超市管理系统的项目计划书。

任务分析与设计

项目计划书是开展和检查开发工作的依据。编制项目计划书的目的是用文档的形式，对开发过程中的人员组织与分工、工作内容、开发进度等方面进行规划。

任务实现

1．编写目的

编写项目计划书的目的是：说明超市管理系统的各项工作；说明超市管理系统软件的功能、性能；说明完成超市管理系统应具备的条件；说明完成期限及其他条件限制；说明完成项目的具体实施计划。

2．项目概述

1）工作内容

简要地说明在本项目的开发过程中必须进行的各项工作。

（1）组织开发小组对超市管理系统有一个基本认识，熟悉整个系统的流程。

（2）系统开发小组在对超市管理系统有基本认识后，拟定实现方案。

（3）系统开发小组对系统进行集中开发。

（4）系统审核小组对系统进行评定、审核。

（5）系统维护小组对系统进行定期维护。

2）参加人员

扼要说明参加本项目开发工作的人员的情况，包括他们的技术水平、主要经历等。

参与此项目开发的所有人员应具有一定的 C 语言项目开发的基础，都开发过“小学生数学选题”系统，掌握了软件系统开发的基本技术。各项目组组长都是思维敏捷、管理能力强的学生。

3）产品

逐项说明本项目的预期开发成果，包括提交给用户的程序、文件和服务，以及应向本单位提交但不需向用户提交的程序和文件。

（1）超市前台管理程序。

（2）超市后台管理程序。

（3）C 语言源文件。

（4）用户使用说明。

4）服务

列出需向用户提供的各项服务，如培训、安装、维护和运行支持等，应逐项规定开始日期、所提供支持的级别和服务的期限。

各组填写表 3.2。

表 3.2　服务项目列表

序　号	任　务	日　期	级　别	期　限
1	培训			
2	安装			
3	维护			
4	运行支持			

说明：由各项目组根据自己的实际情况填写表 3.2。

5）完成期限

本项目的最迟完成期限为 20 天。

6）本计划的批准者和批准日期

（1）项目的批准者：软件技术专业负责人。

（2）批准日期：2008 年 11 月 8 日。

3．项目实施计划

1）工作任务的分解与人员分工

对于项目开发中需完成的各项工作，从需求分析、设计、实现、测试直到维护，包括文件的编制、审批、打印、分发工作，用户培训工作，软件安装工作等，按层次进行分解，指明每项任务的负责人和参加人员。为清晰起见，尽量采用表格的方式。

2）进度

对需求分析、设计、编码实现、测试、移交、培训和安装等工作，给出每项工作的预定开始日期、完成日期，规定各项工作任务完成的先后顺序，以及表明每项工作完成的标志性事件（即“里程碑”）。

开发过程的主要里程碑有：

（1）项目立项。

（2）需求调研结束。

（3）需求分析结束。

（4）概要设计结束。

（5）详细设计结束。

（6）编码结束。

（7）系统联调结束。

（8）系统测试结束。

（9）系统试运行结束。

（10）系统维护结束。

项目实施计划如表 3.3 所示。

表 3.3　项目实施计划

序　号	任　务	负 责 人	时　间
1	熟悉背景、拟定开发计划	组长	1 天
2	熟悉需求分析文档	组员	1 天

续表

序　号	任　务	负 责 人	时　间
3	熟悉系统设计文档	组员	5 天
4	编码实现	组长	10 天
5	系统测试与维护	组员	3 天

4．配置管理

项目开发各阶段的交付项，包括各种文档和代码，组成软件的配置。配置管理规定如何管理这些交付项。开发组会产生大量的交付项，而且，由于不断地修改，每个交付项又有多个版本。如何从这些交付项建立最终系统，并保证用来生成最终系统的各交付项的一致性，是配置管理的主要任务。

每个项目组由专门人员负责项目开发、交付项的管理工作，确保系统准时交付。

5．预算

逐项列出本项目所需要的劳务和经费的预算及来源。

本项目由学生自筹经费完成。

6．关键问题

逐项列出能够影响整个项目成败的关键问题、技术难点和风险，指出这些问题对项目的影响。

（1）处理好销售与库存的关系。

（2）处理好销售与会员的关系。

（3）处理好商品销售后数量的变化。

7．软硬件条件

（1）软件条件：VC6.0 开发环境。

（2）硬件条件：Pentium III 450MHz 以上的 CPU 处理器、64MB 以上的内存、200MB 的自由硬盘空间、能支持 24 位真彩色的显示卡、彩色显示器。

3.2.2　子任务二：编写需求规格说明书

在完成针对超市管理系统软件市场前期调查的基础上，在与客户进行了全面深入的探讨和分析的前提下，编写需求规格说明书。

开发人员需要理解超市管理系统的用户需求，进行细致的调查分析，将用户的非形式的需求陈述转化为完整的需求定义，再由需求定义转换为相应的需求文档。

1．编写目的

超市管理系统需求规格说明书的预期读者为客户、业务或需求分析人员、测试人员、用户文档编写者、项目管理人员。

2．系统概述

1）功能简介

简要描述系统的主要功能，并说明本系统与其他相关系统之间的关系。建议用框图的方式说明系统的组成。

超市管理系统主要是对超市的进货、销售和库存这三个方面进行管理，根据其功能分为前台管理和后台管理。前台管理包括商品销售、库存预警和退出功能；后台管理包括商品维护、会员管理、库存预警和退出功能。超市管理系统的前台（销售）管理系统流程图如图 3.1 所示，后台（进货）管理系统流程图如图 3.2 所示。

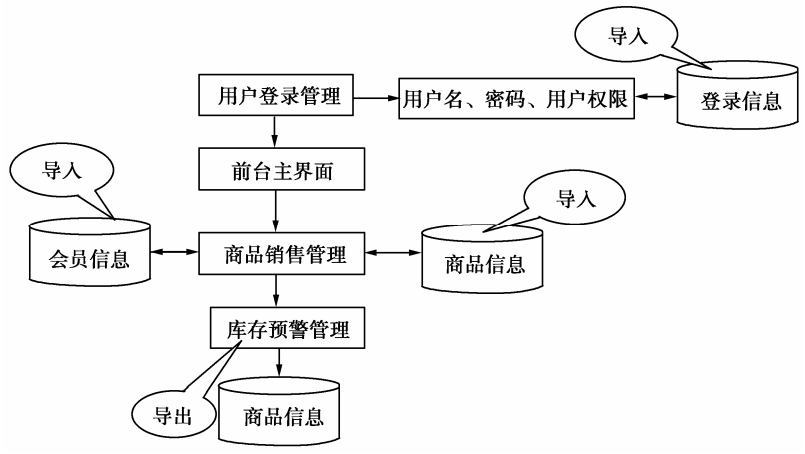


图 3.1　前台（销售）管理系统流程图

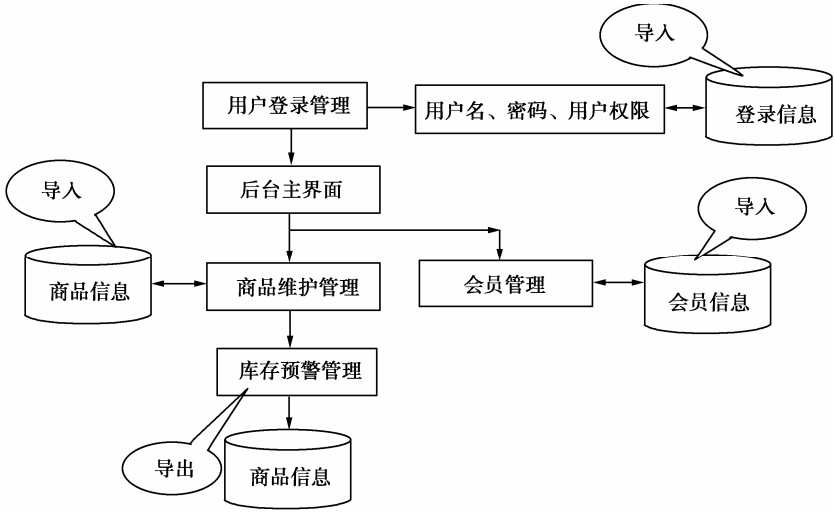


图 3.2　后台（进货）管理系统流程图

2）用户特点

描述本系统的最终用户的特点，说明操作人员、维护人员的技能以及本系统的使用频度。

本系统的最终用户为小型超市，系统操作人员只需能够熟练操作计算机，系统维护人员要熟练掌握系统的功能与用法。本系统的使用频度较高。

3）系统运行环境

说明运行该系统所需要的硬件设备。

硬件条件（最低配置）：Pentium III 450MHz 以上的 CPU 处理器、64MB 以上的内存、200MB 的自由硬盘空间、能支持 24 位真彩色的显示卡、彩色显示器。

3．功能模块说明

1）后台（进货）管理

（1）概述。本模块主要包括商品维护、会员管理、库存预警和退出功能。当库存数量不足 100 时，添加商品，每次添加商品后，库存数量都随之发生改变。

（2）商品维护模块说明如表 3.4 所示。

表 3.4　商品维护模块说明

输入	商品信息
处理	将商品信息写入商品文件中
输出	商品写入是否成功

（3）会员管理——会员添加模块说明如表 3.5 所示。

表 3.5　会员管理——会员添加模块说明

输入	会员信息
处理	将会员信息写入会员文件中
输出	会员写入是否成功

（4）会员管理——会员查询模块说明如表 3.6 所示。

表 3.6　会员管理——会员查询模块说明

输入	输入会员查询方式（编号、姓名或身份证号）
处理	通过系统内部数据判断所输入的信息与文件中的信息是否相符
输出	查询结果

（5）会员管理——会员统计模块说明如表 3.7 所示。

表 3.7　会员管理——会员统计模块说明

输入	输入会员排序方式（降序或升序）
处理	系统进行排序
输出	输出排序结果

（6）会员管理——会员删除模块说明如表 3.8 所示。

表 3.8　会员管理——会员删除模块说明

输入	输入会员编号
处理	删除会员
输出	删除是否成功的提示

（7）会员管理——库存预警模块说明如表 3.9 所示。

表 3.9　会员管理——库存预警模块说明

输入	商品信息
处理	商品数量是否充足
输出	数量不足 100 的商品列表

2）前台（销售）管理

（1）概述。本模块主要包括商品销售、库存预警和退出功能。每次商品销售后，库存数量都随之发生改变。

逐一描述功能集中的各项功能，说明输入、处理过程和输出。

（2）商品销售模块说明如表 3.10 所示。

表 3.10　商品销售模块说明

输入	商品种类与商品数量
处理	每种商品的价格，计算出所购买的所有商品总价
输出	购买清单、商品总价和商品结算清单

（3）库存预警模块说明如表 3.11 所示。

表 3.11　库存预警模块说明

输入	商品信息
处理	商品数量是否充足
输出	数量不足 100 的商品列表

3.3　任务三 :“超市管理系统”的设计

3.3.1　子任务一：编写概要设计说明书

（1）根据“超市管理系统”的需求分析结果，从实现的角度进一步划分为模块，并组成模块的层次结构，绘制超市管理系统的模块层次图（H 图）。

（2）根据系统流程图进行功能分解以确定模块结构，划分功能模块，并说明各模块的功能。

（3）确定模块之间的调用关系，绘制函数关系表。

（1）绘制功能模块图的关键是对用户需求要有充分的了解。

（2）划分各功能模块，采用自顶向下逐层分解的结构化分析方法。

（3）函数关系表主要针对各模块而设计的，体现了各函数之间的调用关系，确定了模块之间的接口，即模块之间传递的信息。

任务实现

1．编写目的

超市管理系统的概要设计是设计的第一个阶段，这个阶段的主要任务是对超市管理系统的结构设计，具体为：

（1）采用结构化设计方法，将超市管理系统按功能划分成模块。

（2）确定超市管理系统各模块的功能。

（3）确定超市管理系统模块之间的调用关系（这里用函数关系表表示）。

（4）确定超市管理系统各模块之间的接口，即模块之间传递的信息。

2．超市管理系统的模块层次设计（H图）

超市管理系统的 H 图如图 3.3 所示。

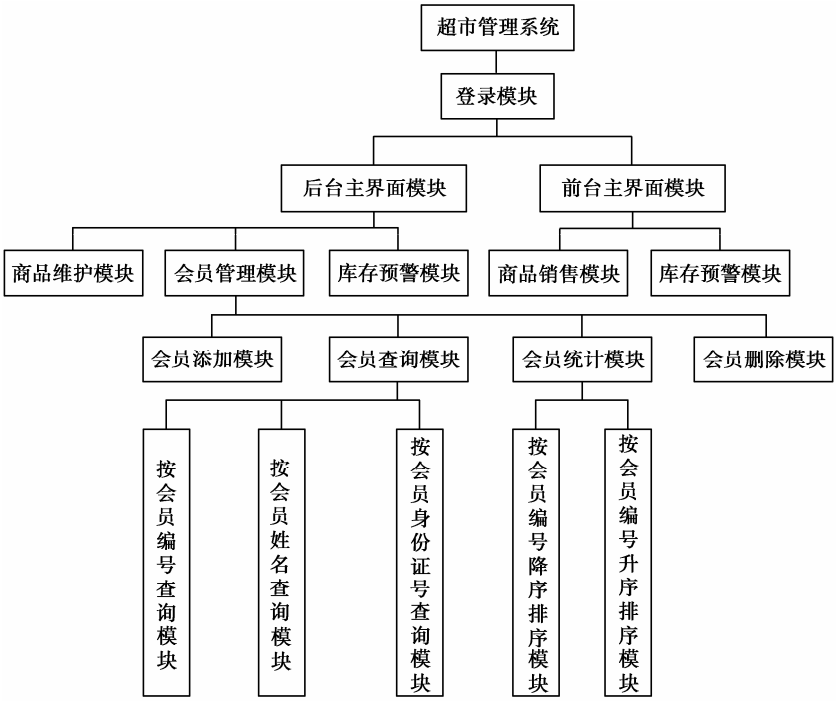


图 3.3　超市管理系统的 H 图

3．各功能模块的划分

根据系统流程图进行功能分解以确定模块结构，逐步划分各功能模块。

（1）登录模块如图 3.4 所示。

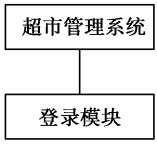


图 3.4　登录模块

功能说明：登录模块的特点是，按权限分为前台用户和后台用户，前台用户只能对商品

销售和库存预警进行管理；后台用户负责商品进货、会员管理和库存预警更新的管理。

（2）登录模块向下分为前台主界面模块和后台主界面模块两个部分，如图 3.5 所示。

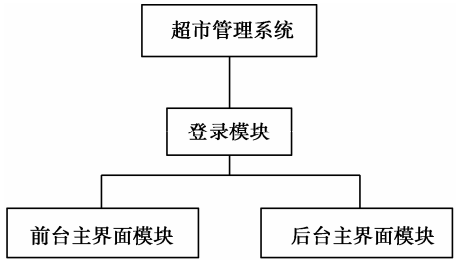


图 3.5　前台主界面模块和后台主界面模块

功能说明：超市管理系统主要分为前台管理和后台管理两大模块，分别对商品销售、库存、商品进货和会员进行管理。

（3）前台主界面模块如图 3.6 所示。

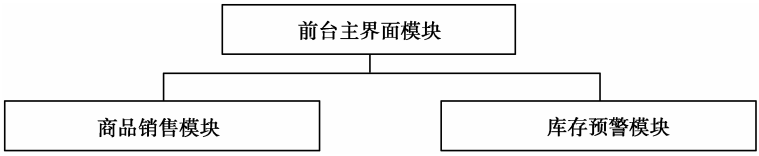


图 3.6　前台主界面模块

功能说明：前台主界面模块包括商品销售和库存预警两个模块。商品销售模块能够实现顾客对多种商品的选购。在结算时，将每种商品的单价和数量相乘，合计出商品总价进行结算，打印出购物清单；库存预警模块能实现当商品库存数量不足 100 时，显示商品清单。

（4）后台主界面模块如图 3.7 所示。

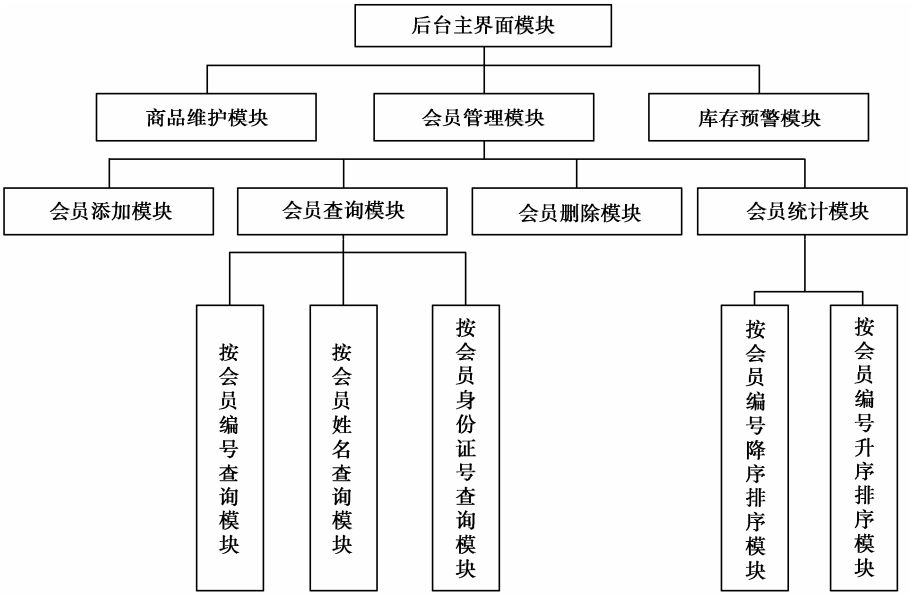


图 3.7　后台主界面模块

功能说明：后台主界面模块包括商品维护、会员管理和库存预警三个模块。其中，会员管理模块包括会员添加、会员查询、会员统计和会员删除模块；会员查询模块可以实现按会员编号、按会员姓名和按会员身份证号查询三种功能；会员统计模块可以实现按会员编号降序排序和按会员编号升序排序两种功能。

4．函数调用关系表

函数调用关系表如表 3.12 所示。

表 3.12　函数调用关系表

模　块	函　数	模 块 功 能	调 用 函 数	被调用函数
登录模块	login()	控制使用权限	main()	backMainMenu() beforeMainMenu()
后台主界面模块	backMainMenu()	显示后台界面	login()	goodsAdd() memberManage() stockWarn()
商品维护模块	goodsAdd()	添加商品	backMainMenu()	
会员管理模块	memberManage()	实现会员添加、查询、统计和删除功能	backMainMenu()	memberAdd() memberQuery() memberStat() memberDelete()
会员添加模块	memberAdd()	添加会员	memberManage()	
会员查询模块	memberQuery()	按会员编号、姓名和身份证号进行会员查询	memberManage()	numberQuery() nameQuery() identityQuery()
按会员编号查询模块	numberQuery()	按会员编号查询，并显示结果	memberQuery()	
按会员姓名查询模块	nameQuery()	按会员姓名查询，并显示结果	memberQuery()	
按会员身份证查询模块	identityQuery()	按会员身份证查询，并显示结果	memberQuery()	
会员统计模块	memberStatMenu()	按会员编号进行降序排序或升序排序	memberManage()	MemberStat()
会员删除模块	memberDelete()	删除指定编号的会员	memberManage()	
库存预警模块	stockWarn()	列出库存不足商品列表	backMainMenu()	
前台主界面模块	beforeMainMenu()	显示前台主界面	Login()	goodsSale() stockWarn()
商品销售模块	goodsSale()	销售商品、统计商品总价、结算找零	beforeMainMenu()	numberQuery() amountDeal()
库存预警模块	stockWarn()	列出库存不足商品列表	beforeMainMenu()	

3.3.2　子任务二：编写详细设计说明书

（1）根据“超市管理系统”的流程图和 H 图，绘制 IPO 图。

（2）详细设计数据结构。

任务分析与设计

（1）绘制 IPO 图的关键是，清楚各功能模块的输入/输出数据、处理功能和调用详细情况。

（2）通过程序流程图将各主要模块的算法表达出来。

（3）数据结构设计要详细，以便为编码实现打下良好的基础。

任务实现

1．编写目的

超市管理系统的详细设计是设计的第二个阶段，这个阶段的主要任务是在超市管理系统的概要设计基础上，对概要设计中产生的功能模块进行过程描述，设计功能模块的内部细节，包括算法和详细数据结构，为编写源代码提供必要的说明。

2．项目概述

1）模块编号

超市管理系统各模块编号表如表 3.13 所示。

表 3.13　超市管理系统各模块编号表

序　　号	编　　号	名　　称
1	01	登录模块
2	02	后台主界面模块
3	021	商品维护模块
4	022	会员管理模块
5	0221	会员添加模块
6	0222	会员查询模块
7	02221	按会员编号查询模块
8	02222	按会员姓名查询模块
9	02223	按会员身份证号查询模块
10	0223	会员统计模块
11	02231	按会员编号降序排序模块
12	02232	按会员编号升序排序模块
13	0224	会员删除模块
14	023	库存预警模块
15	03	前台主界面模块
16	031	商品销售模块
17	032	库存预警模块

2）各模块IPO表

（1）登录模块 IPO 表如表 3.14 所示。

表 3.14　登录模块 IPO 表

系统：超市管理系统	作者：×××
模块：登录模块	日期：××××
模块编号：01	
上层调用模块：超市管理系统	下层被调用模块： 后台主界面模块 前台主界面模块
输入：用户名、密码	输出数据：验证结果是否正确
处理：根据用户所提交的登录信息，找到相应权限的数据文件，验证用户名和密码，共提供三次机会。如果前台密码验证成功，则进入前台主界面；如果后台密码验证成功，则进入后台主界面；否则三次错误提示相关信息退出系统	
局部数据元素：输入的用户名、密码、权限，权限对应的数据文件中读取的登录名、密码，允许重复登录 3 次	

（2）后台主界面模块 IPO 表如表 3.15 所示。

表 3.15　后台主界面模块 IPO 表

系统：超市管理系统	作者：×××
模块：后台主界面模块	日期：××××
模块编号：02	
上层调用模块：登录模块	下层被调用模块： 商品维护模块 会员管理模块 库存预警模块
输入：用户所需选择的项目 choi(0-3)	输出数据：
处理： choi=1，调用商品维护模块 choi=2，调用会员管理模块 choi=3，调用库存预警模块 choi=0，返回后台主界面	
局部数据元素：	

（3）商品维护模块 IPO 表如表 3.16 所示。

表 3.16　商品维护模块 IPO 表

系统：超市管理系统	作者：×××
模块：商品维护模块	日期：××××
模块编号：021	
上层调用模块：后台主界面模块	下层被调用模块：
输入：商品信息，即商品编号、商品名称、生产日期、商品进价、商品数量、商品销售单价	输出数据：商品添加是否正确

处理：将商品信息写入商品文件中
局部数据元素：

（4）会员管理模块 IPO 表如表 3.17 所示。

表 3.17　会员管理模块 IPO 表

系统：超市管理系统	作者：×××
模块：会员管理模块	日期：××××
模块编号：022	
上层调用模块：后台主界面模块	下层被调用模块： 会员添加模块 会员查询模块 会员统计模块 会员删除模块
输入：用户所需选择的项目 choi(0-4)	输出数据：
处理： choi=1，调用会员添加模块 choi=2，调用会员查询模块 choi=3，调用会员统计模块 choi=4，调用会员删除模块 choi=0，返回会员管理界面	
局部数据元素：	

（5）会员添加模块 IPO 表如表 3.18 所示。

表 3.18　会员添加模块 IPO 表

系统：超市管理系统	作者：×××
模块：会员添加模块	日期：××××
模块编号：0221	
上层调用模块：会员管理模块	下层被调用模块：
输入：会员信息，即会员编号（自动产生）、会员姓名和会员身份证号	输出数据：添加会员是否成功
处理：将会员信息写入会员文件中	
局部数据元素：会员编号要求自动生成	

（6）会员查询模块 IPO 表如表 3.19 所示。

表 3.19　会员查询模块 IPO 表

系统：超市管理系统	作者：×××
模块：会员查询模块	日期：××××
模块编号：0222	
上层调用模块：会员管理模块	下层被调用模块：

	按会员编号查询模块 按会员姓名查询模块 按会员身份证号查询模块

续表

输入：用户所需选择的项目 choi(0-3)	输出数据：添加会员是否成功
处理： choi=1，调用按会员编号查询模块 choi=2，调用按会员姓名查询模块 choi=3，调用按会员身份证号查询模块 choi=0，返回会员管理界面	
局部数据元素：	

（7）按会员编号查询模块 IPO 表如表 3.20 所示。

表 3.20　按会员编号查询模块 IPO 表

系统：超市管理系统	作者：×××
模块：按会员编号查询模块	日期：××××
模块编号：02221	
上层调用模块：会员查询模块	下层被调用模块：
输入：会员编号	输出数据：会员查询结果
处理：将输入的会员编号与会员文件中的编号进行比较	
局部数据元素：	

（8）按会员姓名查询模块 IPO 表如表 3.21 所示。

表 3.21　按会员姓名查询模块 IPO 表

系统：超市管理系统	作者：×××
模块：按会员姓名查询模块	日期：××××
模块编号：02222	
上层调用模块：会员查询模块	下层被调用模块：
输入：会员姓名	输出数据：会员查询结果
处理：将输入的会员姓名与会员文件中的姓名进行比较	
局部数据元素：	

（9）按会员身份证号查询模块 IPO 表如表 3.22 所示。

表 3.22　按会员身份证号查询模块 IPO 表

系统：超市管理系统	作者：×××
模块：按会员身份证号查询模块	日期：××××
模块编号：02223	
上层调用模块：会员查询模块	下层被调用模块：
输入：会员身份证号	输出数据：会员查询结果

处理：将输入的会员身份证号与会员文件中的身份证号进行比较
局部数据元素：

（10）会员统计模块 IPO 表如表 3.23 所示。

表 3.23　会员统计模块 IPO 表

系统：超市管理系统	作者：×××
模块：会员统计模块	日期：××××
模块编号：0223	
上层调用模块：会员管理模块	下层被调用模块： 按会员编号降序排序模块 按会员编号升序排序模块
输入：用户所需选择的项目 choi(0-2)	输出数据：会员排序后列表
处理： choi=1，调用按会员编号降序排序模块 choi=2，调用按会员编号升序排序模块 choi=0，返回会员管理界面	
局部数据元素：	

（11）按会员编号降序排序模块 IPO 表如表 3.24 所示。

表 3.24　按会员编号降序排序模块 IPO 表

系统：超市管理系统	作者：×××
模块：按会员编号降序排序模块	日期：××××
模块编号：02231	
上层调用模块：会员统计模块	下层被调用模块：
输入：	输出数据：
处理：将会员文件中的会员信息读取出来，对其按编号进行降序排序	
局部数据元素：	

（12）按会员编号升序排序模块 IPO 表如表 3.25 所示。

表 3.25　按会员编号升序排序模块 IPO 表

系统：超市管理系统	作者：×××
模块：按会员编号升序排序模块	日期：××××
模块编号：02232	
上层调用模块：会员统计模块	下层被调用模块：
输入：	输出数据：
处理：将会员文件中的会员信息读取出来，对其按编号进行升序排序	
局部数据元素：	

（13）会员删除模块 IPO 表如表 3.26 所示。

表 3.26　会员删除模块 IPO 表

系统：超市管理系统	作者：××××
模块：会员删除模块	日期：××××
模块编号：0224	

续表

上层调用模块：会员管理模块	下层被调用模块：
输入：会员编号	输出数据：会员删除是否成功
处理：将会员文件中的所有数据读入结构体数组中，并把用户输入的会员编号与结构体数组中的会员编号进行比较。如果相同，则将后边的数据依次向前移一位。最后将改变后的结构体数组元素全部写入会员文件中	
局部数据元素：	

（14）库存预警模块 IPO 表如表 3.27 所示。

表 3.27　库存预警模块 IPO 表

系统：超市管理系统	作者：××××
模块：库存预警模块	日期：××××
模块编号：023、032	
上层调用模块：前台主界面模块	下层被调用模块：
输入：	输出数据：
处理：如果商品数量不足 100，列出需进货商品清单	
局部数据元素：	

（15）前台主界面模块 IPO 表如表 3.28 所示。

表 3.28　前台主界面模块 IPO 表

系统：超市管理系统	作者：××××
模块：前台主界面模块	日期：××××
模块编号：03	
上层调用模块：登录模块	下层被调用模块： 商品销售模块 库存预警模块
输入：用户所需选择的项目 choi(0-2)	输出数据：
处理： choi=1，调用商品销售模块 choi=2，调用库存预警模块 choi=0，退出系统	
局部数据元素：	

（16）商品销售模块 IPO 表如表 3.29 所示。

表 3.29　商品销售模块 IPO 表

系统：超市管理系统	作者：××××

模块：商品销售模块	日期：××××
模块编号：031	

续表

上层调用模块：前台主界面模块	下层被调用模块： 会员查询模块 商品查询模块
输入：商品编号和商品数量	输出数据：商品清单和商品销售总价
处理： 商品清单 商品销售总价 结算找零	
局部数据元素：输入顾客所选购的商品编号和数量，列出商品清单；商品数量×商品单价，计算出商品总价；判断是否是会员，如果是会员，则打九五折。商品结算找零后，对库存中的商品数量进行处理	

3．数据结构设计

根据需求分析及概要设计，进一步分析数据结构，数据详细情况如表 3.30 和表 3.31 所示。

表 3.30　商品目录表

汉 语 名 称	英 文 名 称	类　　型	长　　度
商品编号	goodsNumber	整型	2
商品名称	goodsName	字符	20
商品类型	goodsType	字符	8
生产日期	produceDate	日期	10
保质期	assureDate	整型	2
商品进价	goodsPrice	浮点型	8
商品数量	goodsAmount	整型	2
销售单价	salePrice	浮点型	4

表 3.31　会员表

汉 语 名 称	英 文 名 称	类　　型	长　　度
会员编号	memberNumber	整型	2
身份证号	identity	字符	20
会员名称	memberName	字符	8

3.4　任务四 :“超市管理系统”的编码实现

3.4.1　子任务一：界面设计

设计超市管理系统界面，根据用户要求确定界面的风格，分组绘制界面草图，编码实现各模块界面的设计。

具体要求：

（1）界面清晰明了、各界面之间上下衔接自然。

（2）根据系统概要设计的功能模块图确定各界面之间的关系。

任务分析与设计

分析超市管理系统共需要几个独立界面。

（1）分别定义各界面函数。

（2）具体设计各界面的风格。

（3）做好各界面的衔接。

（4）运行调试。

任务实现

（1）根据系统概要设计，初步分析超市管理系统共有如下界面：登录界面、主界面（前台主界面和后台主界面）、商品维护界面、会员管理界面（会员添加、会员查询、会员删除、会员统计界面）、商品销售界面。

（2）根据函数关系表确定各界面函数名称为 login()、backMainMenu()、beforeMain Menu()、goodsAdd()、memberManage()、memberAdd()、memberQuery()、memberStat()、memberDelete()、goodsSale()。

（3）编码实现。

```
#include <stdio.h>     /*引用预处理命令中的 stdio.h 头文件*/
#include <string.h>
#include <stdlib.h>
main()                 /*主函数*/
{
    login();           /*调用 login()函数*/
}
```

1．登录界面编码

```
void login()
  {
      char username[20],password[20]; //定义字符数组变量 username 和 password
      system("cls");//清屏
      printf("\n\n");
      printf("\t\t*****************************************************\n");
      printf("\t\t                    登录                             \n");
      printf("\t\t*****************************************************\n");
      printf("\n\n");
      printf("\t\t\t\t1.用户名称：");
      scanf("%s",username);
      printf("\n\n");
      printf("\t\t\t\t2.密码：");
```

```
        scanf("%s",password);
    }
```

运行结果如图 3.8 所示。

图 3.8　登录界面

2．后台主界面编码

```
void backMainMenu(){
    int choice;//代表选择的功能项
    char con;//是否继续
    while(1){//实现循环选择
        //程序的主界面
        system("cls");//清屏
        printf("\n\n");
        printf("\t\t*****************************************************\n");
        printf("\t\t                  后台主界面                         \n");
        printf("\t\t*****************************************************\n");
        printf("\n\n");
        printf("\t\t\t\t1.商品维护：\n\n");
        printf("\t\t\t\t2.会员管理：\n\n");
        printf("\t\t\t\t3.库存预警：\n\n");
        printf("\t\t\t\t0.退出系统：\n\n");
        printf("\t\t 请输入你的选择：");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                goodsAdd();//商品维护
                break;
            case 2:
                memberManage();//会员管理
                break;
            case 3:
                stockWarn();//库存预警
                break;
```

```
                case 0:
                    //返回确认处理
                    printf("\t\t 确认要退出吗？(Y/N)：");
                    scanf(" %c",&con);
                    if(con=='Y' || con=='y'){
                        exit(0);//整个程序结束
                    }
                    break;
                default:
                    system("cls");
                    printf("请输入正确的选择！！！（按任意键返回）");
                    getchar();getchar();//实现按任意键返回
            }
        }
    }
```

运行结果如图 3.9 所示。

图 3.9　后台主界面

3．前台主界面编码

```
void beforeMainMenu(){
    int choice;//代表选择的功能项
    char con;//是否继续
    while(1){//实现循环选择
        //程序的主界面
        system("cls");//清屏
        printf("\n\n");
        printf("\t\t**********************************************************\n");
        printf("\t\t                    前台主界面                            \n");
        printf("\t\t**********************************************************\n");
        printf("\n\n");
        printf("\t\t\t\t1、商品销售：\n\n");
```

```
            printf("\t\t\t\t2、库存预警：\n\n");
            printf("\t\t\t\t0、退出系统：\n\n");
            printf("\t\t 请输入你的选择：");
            scanf("%d",&choice);
            switch(choice){
                case 1:
                    goodsSale();//商品销售
                    break;
                case 2:
                    stockWarn();//库存预警
                    break;
                case 0:
                    //返回确认处理
                    printf("\t\t 确认要退出吗？(Y/N)：");
                    scanf(" %c",&con);
                    if(con=='Y' || con=='y'){
                        exit(0);//整个程序结束
                    }
                    break;
                default:
                    system("cls");
                    printf("请输入正确的选择！！！（按任意键返回）");
                    getchar();getchar();//实现按任意键返回
            }
        }
    }
```

运行结果如图 3.10 所示。

图 3.10　前台主界面

4．商品维护界面编码

```
    void goodsAdd()
    {
```

```
            system("cls");//清屏
            printf("\n\n");
            printf("*************************************************\n");
            printf("                    商品维护                     \n");
            printf("*************************************************\n");
            printf("\n\n");
            int i =1;
            //商品录入
                printf("请输入商品的编号：");
    }
```

运行结果如图 3.11 所示。

图 3.11　商品维护界面

5．会员管理界面编码

```
    void memberManage(){

        int choice;//代表选择的功能项
        char con;//是否继续
        while(1){//实现循环选择
            //程序的主界面
            system("cls");//清屏
            printf("\n\n");
            printf("\t\t*******************************************************\n");
            printf("\t\t                  会员管理                             \n");
            printf("\t\t*******************************************************\n");
            printf("\n\n");
            printf("\t\t\t1.会员添加：\n\n");
            printf("\t\t\t2.会员查询：\n\n");
            printf("\t\t\t3.会员统计：\n\n");
            printf("\t\t\t4.删除会员：\n\n");
            printf("\t\t\t0.返回主界面：\n\n");
            printf("\t\t 请输入你的选择：");
            scanf("%d",&choice);
            switch(choice){
                case 1:
```

```
                goodsAdd();//会员添加
                break;
            case 2:
                memberQuery();//会员查询
                break;
            case 3:
                memberStat();//会员统计
                break;
            case 4:
               memberDeleteMenu();//删除会员
               break;
            case 0:
                //确认退出处理
                printf("\t\t 确认要返回主界面吗？(Y/N)：");
                scanf(" %c",&con);
                if(con=='Y' || con=='y'){
                    beforeMainMenu();//返回主界面
                }
                break;
            default:
                system("cls");
                printf("请输入正确的选择！！！（按任意键返回会员管理）");
                getchar();getchar();//实现按任意键返回
        }
    }
}
```

运行结果如图 3.12 所示。

图 3.12　会员管理界面

6．会员查询界面编码

```
void memberQuery(){

    int choice;
```

```
        char con;//是否继续
        int number;
        while(1){//实现循环选择


            system("cls");//清屏
            printf("\n\n");
            printf("\t\t*****************************************************\n");
            printf("\t\t                    会员查询                         \n");
            printf("\t\t*****************************************************\n");

            printf("\n\n");
            printf("\t\t\t\t1.按会员编号查：\n\n");
            printf("\t\t\t\t2.按身份证号查：\n\n");
            printf("\t\t\t\t3.按会员姓名查：\n\n");
            printf("\t\t\t\t0.返回会员管理：\n\n");
            printf("\t\t 请输入你的选择：");
            scanf("%d",&choice);
            switch(choice){
                case 1:
                    printf("请输入会员的编号：");
                    scanf("%d",&number);
                    numberQuery(number,0);//按会员编号查
                    break;
                case 2:
                    identityQuery();//按身份证号查

                    break;
                case 3:
                    nameQuery();//按会员姓名查
                    break;
                case 0:
                    //返回确认处理
                    printf("\t\t 确认要返回吗？(Y/N)：");
                    scanf(" %c",&con);
                    if(con=='Y' || con=='y'){
                        return;
                    }
                    break;
                default:
                    system("cls");
                    printf("请输入正确的选择！！！（按任意键返回）");
                    getchar();getchar();//实现按任意键返回
            }
        }
    }
```

运行结果如图 3.13 所示。

图 3.13　会员查询界面

7．会员统计界面编码

```
void memberStatMenu()
{
    int choice;

    //定义是否继续统计变量 con
    char con;

    //实现循环统计
    while(1){


        system("cls");//清屏
        printf("\n\n");
        printf("\t\t*****************************************************\n");
        printf("\t\t                    会员统计                         \n");
        printf("\t\t*****************************************************\n");

        printf("\n\n");
        printf("\t\t\t\t1.按会员编号降序排列：\n\n");
        printf("\t\t\t\t2.按会员编号升序排列：\n\n");
        printf("\t\t\t\t0.返回会员管理：\n\n");
        printf("\t\t 请输入你的选择：");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                memberStat(1);//降序
                break;
            case 2:
                memberStat(2);//升序
```

```
                    break;
                case 0:
                    //返回确认处理
                    printf("\t\t 确认要返回吗？(Y/N)：");
                    scanf(" %c",&con);
                    if(con=='Y' || con=='y'){
                        return;
                    }
                    break;
                default:
                    system("cls");
                    printf("请输入正确的选择！！！（按任意键返回）");
                    getchar();getchar();
            }
        }
    }
```

运行结果如图 3.14 所示。

图 3.14　会员统计界面

说明：会员添加界面、会员删除界面、商品销售界面编码见功能模块设计。

引导文献

1．指针

指针是 C 语言中广泛使用的一种数据类型。运用指针编程是 C 语言最主要的风格之一。利用指针变量可以表示各种数据结构；能很方便地使用数组和字符串；并能像汇编语言一样处理内存地址，从而编写出精练而高效的程序。指针极大地丰富了 C 语言的功能。

1）指针的概念

简单地说，指针就是地址。这里的地址是指内存地址，与我们日常生活中的地址很相似，它说明了某一个对象（对程序来说是数据，对生活中的对象可以是人，也可以是物等）所在的地方。我们通常会向某人询问他的住址，然后把这个地址记录在纸上。在程序的世界里，也可以找到这样的模型，如下面的程序段所示：

```
int nSomething = 100;
```

```
int *pSomethingAddress = &nSomething;
```

其中，第一条语句声明了一个整型变量 nSomething 并赋初值 100，第二条语句声明了一个整型指针变量 pSomethingAddress，并指向 nSomething。与现实世界模型相对应的计算机世界模型是：nSomething 相当于一个人所在的房子，其值 100 相当于这个人，其地址&nSomething 就是这个房子的地址，变量 pSomethingAddress 相当于记录地址的纸，其值（现在是&nSomething，它是由程序动态确定的）就是记录在纸上的地址。

指针的内存逻辑结构图如图 3.15 所示。

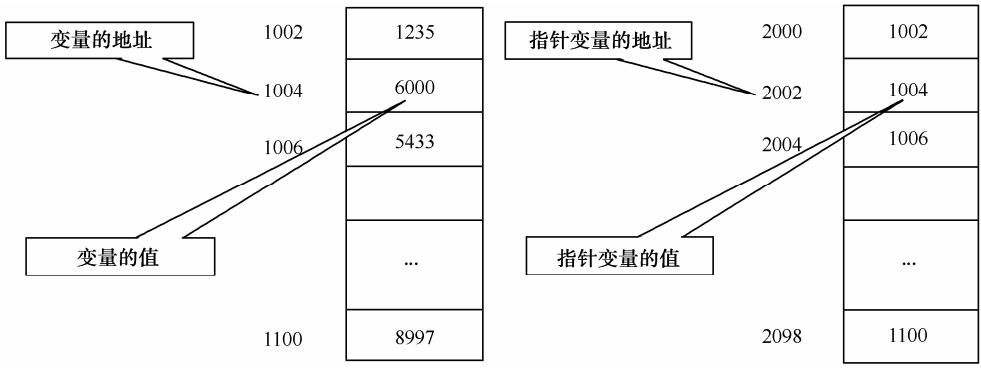
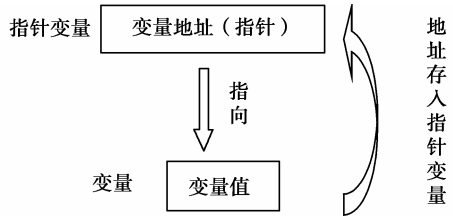


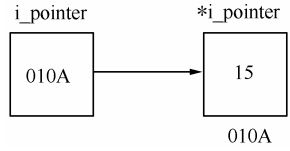
图 3.15　指针的内存逻辑结构图

2）变量的指针和指针变量

变量的指针就是变量的地址。存放变量地址的变量是指针变量。在 C 语言中，允许用一个变量来存放指针，这种变量称为指针变量。因此，一个指针变量的值就是某个变量的地址或称为某变量的指针。



为了表示指针变量和它所指向的变量之间的关系，在程序中用“*”符号表示“指向”，例如，i_pointer 代表指针变量，而*i_pointer 是 i_pointer 所指向的变量。



因此，下面两个语句的作用相同：

```
i=15;
*i_pointer=15;
```

第二个语句的含义是将 15 赋给指针变量 i_pointer 所指向的变量。

3）指针变量的声明

在 C 语言中，声明一个指针变量与声明其他的变量相似，只需要在对应的变量类型后面加一个“*”号，如声明一个整型指针变量（如前面所述）。你可以声明任何类型的指针变量，包括 C 语言内置的变量类型和自定义类型，如结构、联合或枚举类型，声明的方法同上。假如有一个结构类型 Student 定义如下：

```
struct Student
{
        char szName[50];
        char szGender[2];
         int   nAge;
};
```

那么声明一个 Student 类型的指针变量如下：

```
struct Student *pStudent;
```

指针变量的命名与其他变量的命名一样是任意的，但要求符合 C 语言的命名习惯。一般来说，为了程序的可读性，我们总以字母 p 开头来声明指针变量名，以便当我们一看到这个变量时就知道这是一个指针变量，减少出错的可能性。

在这里需要说明的是，对于某一类型的指针变量应这样理解：该变量是一个指针，该指针指向该类型变量的内存首地址，该指针也可以为空指针，即暂时（也许永远）不指向任何变量，但如果将来某一时刻为其分配变量，则它将指向该类型的变量。也可以这样想：该变量保存了一个地址（记住是地址，而不是一般的值，也许某一变量的值与该地址的值相等，但不相干），该地址是该类型的某一变量的首地址。

以上只说明了指针变量的声明，还没有为该变量的赋值。习惯上，我们在声明一个指针变量的同时也为其赋初值。如果我们已经明确知道该指针指向的变量，就可以直接把该变量的地址赋给该指针变量。取得某一变量的首地址的方法是在该变量的名字前加上“&”符号。这样，我们就可以这样声明一个指针变量：

```
struct Student stJordan;
struct Student*pJordan=&stJordan;
```

stJordan 是 Student 类型的变量，pJordan 是 Student 类型的指针变量，声明的同时给它赋初值指向 stJordan。

如果在声明指针变量时，我们不知道该指针应该指向哪一个变量，那么习惯上把它置空，即 NULL 值。例如：

```
int *pAccount = NULL;
struct Student *pStudent=NULL;
```

这样，在以后的某一时刻引用该变量时，需要判断该指针变量是否有效，即是否已经指向属于应用程序内存空间中的该类型的某一变量或内存空间（可以是数组或动态分配的内存空间）。

4）指针变量的使用

在声明了指针变量以后，就可以利用该指针变量对所指向的对象进行操纵。在进行操纵前，必须保证所操纵的对象是有效的，即该指针不为空，否则会引起“内存访问冲突”的错误，导致程序崩溃。

判断一个指针是否有效的办法是判断它是否为空指针：

```
if (pSomePoint != NULL)
{//指针有效
}
else
{//指针无效
}
```

然而，这样的判断方法需要有一个前提，即在声明指针时如果没有明确指向一个已知的变量，则将其置空，在不再需要使用这个指针时也将其置空。如果该指针指向一块动态分配的内存空间，则在释放后也应该将其置空。在这个前提下，前面的判断才是正确的。

对一个有效的指针，引用该指针所指向的对象的办法是在该指针变量前使用“*”号，如下所示：

```
int nAccount = 100;
int *pAccount = &nAccount;//pAccount 指向 nAccount
 (*pAccount) = 500;//nAccount = 500
 (*pAccount) = (*pAccount )+ 500;//nAccount = 1000
```

各运算符的运算顺序按 C 语言的规定，但是最好加上括号明确指明各个运算的顺序。这里可以看到(*pAccount)与 nAccount 等价。

对于指向结构变量的指针的使用也与此相似，按以上 pJordan 的定义，我们可以这样来访问其结构成员：

```
strcpy(pJordan->szName,"Jordan");
strcpy(pJordan->szGender, "男");
pJordan->nAge = 24;
```

其中的“->”运算符表示取该结构指针所指向的结构对象的成员变量（即域）。另一种与之等价的办法是：

```
strcpy((*pJordan).szName, "Jordan");
strcpy((*pJordan).szGender, "男");
 (*pJordan).nAge = 24;
```

在这里，(*pJordan)与 stJordan 等价。

对于指针，除了可以进行上述的各种运算以外，还可以自加（++）、自减（--）、加一个整数、减一个整数、两个指针相减等操作。与对整型数的自加和自减操作一样，指针的自加、自减操作与指针的加一、减一操作相同，自加与自减运算符出现在指针变量的前面或后面对指针变量的影响与该运算符出现在整型变量的前面或后面对整型变量的影响一样。

指针变量可以与一整型数相加或相减。指针变量 p 与整型变量 n 相加（p+n）表示指针变量 p 的值增加 n*sizeof(pointer_type)，其中的 pointer_type 表示该指针变量的类型，而 p-n 表示指针变量 p 减少 n*sizeof(pointer_type)。

两个指针变量不可以相加，因为其相加的结果是没有意义的，但是相同类型的两个指针变量可以相减。两个指针变量的差的意义是：(p2-p1)*sizeof(pointer_type)表示两个指针变量之间的内存空间的字节数。要注意的是，并不是在任何两个指针变量之间都可以进行相减操作。

指针变量的自加、自减、与整数的加减运算以及两个指针变量的差的运算通常用于利用指针访问一片连续的内存空间的场合，比如利用指针遍历数组，给动态分配的内存空间赋值或从动态分配的内存空间中取值。

2．字符数据的输入和输出

1）putchar()函数（字符输出函数）

函数功能：putchar()函数是向标准输出设备输出一个字符。

函数格式：

```
putchar(ch);
```

其中，ch 为一个字符变量或常量。

函数说明：

（1）putchar()函数的作用等同于 printf("%c", ch); 。

（2）对控制字符则执行控制功能，不在屏幕上显示。

（3）使用该函数前，必须使用以下命令：

```
#include<stdio.h>
```

或

```
#include "stdio.h"
```

【例 3.1】

```
//输出单个字符
#include<stdio.h>
main()
{
char c: /*定义字符变量*/
c='B'; /*给字符变量赋值*/
putchar(c); /*输出字母 B */
putchar('\x42'); /*也输出字母 B*/
putchar(0x42); /*直接用 ASCII 码值输出字母 B*/
}
```

2）getchar()函数（键盘输入函数）

函数功能：getchar()函数的功能是从键盘上输入一个字符。

函数格式：

```
getchar();
```

通常把输入的字符赋予一个字符变量，构成赋值语句，例如：

```
char c;
c=getchar();
```

【例 3.2】

```
//输入单个字符并在显示器上输出
#include<stdio.h>
  void main()
```

```
{
   char c;
   printf("input a character\n");
   c=getchar();
   putchar(c);
}
```

函数说明：getchar()函数只能接收单个字符，输入数字也按字符处理。输入多于一个字符时，只接收第一个字符。

使用本函数前，必须包含文件“stdio.h”。

程序的最后两行可用下面两行的任意一行代替：

```
putchar(getchar());
printf("%c",getchar());
```

3）getch()函数和getchar()函数的区别

（1）getch()函数直接从键盘获取键值，输入的字符不会回显在屏幕上，一般用来暂停屏幕；而 getchar()函数要接收一个字符且在屏幕上显示。

（2）头文件不一样：getch()函数要有 conio.h 头文件，getchar()函数要有 stdio.h 头文件。

【例 3.3】

```
//getchar()函数、putchar()函数和 getch()函数的用法
#include<stdio.h>
main()
{
char c;
c=getchar();      /*从键盘读入字符直到回车结束*/
putchar(c);       /*显示输入的第一个字符*/
getch();          /*等待按任一键*/
```

【例 3.4】

```
//置换法加密
#include <stdio.h>
#include <stdlib.h>
void password(int keycode);
Int main(void)
{
    int key;
    printf("\n:请输入密钥");
    password(key);
    return(0);
}
void password(int keycode)
{
    char ch;
    printf("\n:l 输入明文，将得到对应的密文如下：");
```

```
    while((ch=getch())!='\r')
    (ch+keycode)>122?putchar(ch-122+33+keycode);
    ((ch+keycode)<33?putchar(ch+122+keycode));
}
```

程序说明：最简单的数据加密称为置换法。有以字符为单位置换的，也有以句为单位置换的。本题采用的是一种最简单的以字符为单位的置换加密方法。上例方法就是在 ASCII 码表中，把一个要变换的字符加上（或减去）一个小常数使其变成另外一个字符，如字符'a'+3，变成了'd'。这里的 3 就称为密钥（KEY）。

即时训练

（1）写出下列程序的运行结果：

```
#include<stdio.h>
main()
{
int a=2,b=3;
int *p1,*p2;
p1=&a;
p2=&b;
printf("a=%d\n",a);
printf("b=%d\n",b);
printf("p1=%d\n",*p1);
printf("p2=%d\n",*p2);
}
```

（2）说明下列程序完成的功能：

```
#include<stdio.h>
main()
{
int    *p1,*p2,*p;
int a,b;
scanf("%d,%d",&a,&b);
p1=&a;
p2=&b;
if(a<b)
{
      p=p1;p1=p2;p2=p;
}
printf("a=%d\n",a);
printf("b=%d\n",b);
printf("p1=%d\n",*p1);
printf("p2=%d\n",*p2);
}
```

拓展任务

为超市管理系统添加一个用户管理模块，其功能有添加用户、查询用户、修改用户、删除用户。试设计出该模块的界面。

3.4.2 子任务二：登录模块的功能实现

任务描述

在登录模块界面设计的基础上，具体实现登录模块的功能，即当用户输入用户名和密码后，系统进行判断。如果用户名和密码正确，进入主界面，如果错误，要求重新输入，三次错误退出系统。

具体要求：

（1）在小学生数学选题系统的登录模块实现的基础上，将用户名和密码改成字符型数据。

（2）因为超市管理系统分为前台管理和后台管理，所以登录时要分别定义两个用户。

任务分析与设计

（1）登录模块的基本流程如表 3.32 所示。

表 3.32 登录模块的基本流程

1．定义登录函数	2．呈现界面，如图 3.8 所示，显示“请输入用户名称”提示
3．输入用户名（键盘输入字符型数据）	4．显示“请输入密码”的提示
5．输入密码（键盘输入字符型数据）	6．判断用户名称和密码是否正确
7．如果是前台用户，则调用前台主界面函数 beforeMainMenu()	8．如果是后台用户，则调用后台主界面函数 backMainMenu()
9．否则将累计用户名与密码错误次数的变量 number 加 1	10．判断 number 是否小于 3，如果正确，则清屏并返回步骤 2，否则退出系统

（2）自然语言描述。

S1：定义用户名、密码、累计用户名与密码错误次数的变量分别为 username、password、number，初始化 number 的值为 0，将 username 和 password 定义为字符数组。

S2：接收从键盘输入的用户名和密码。

S3：判断用户名和密码是否正确。

S3.1：username 和 passwod 两者均为“ht”，调用后台主界面 backMainMenu()函数。

S3.2：username 和 passwod 两者均为“qt”，调用前台主界面 beforeMainMenu()函数。

S3.3：两者都不是，计数器 number 累加，判断计数器的值。

S3.3.1：如果 number<3，返回登录界面，重新输入用户名和密码。

S3.3.2：否则，退出系统。

（3）设计登录模块流程图，如图 3.16 所示。

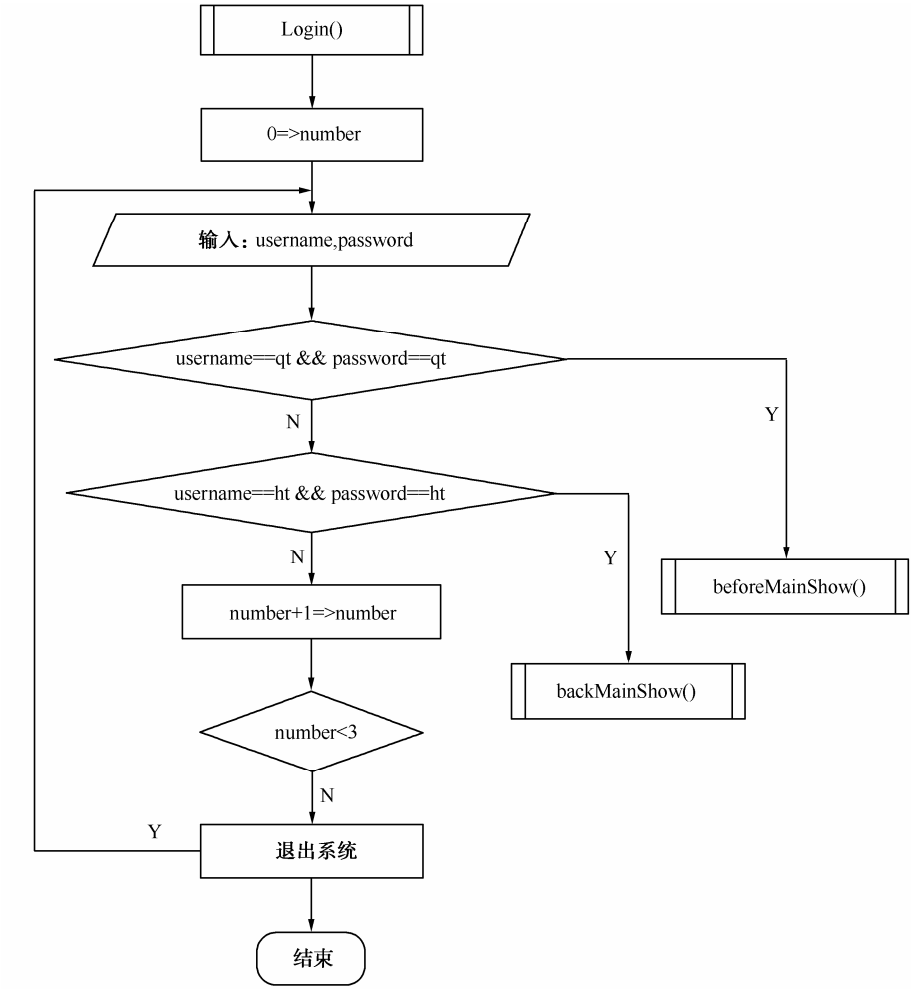


图 3.16　登录模块流程图

任务实现

```
void login()
    {
    int number=0;
    char username[20],password[20];//定义字符数组变量 username 和 password
    while (number<3)
    {
        system("cls");//清屏
        printf("\n\n");
        printf("\t\t*****************************************************\n");
        printf("\t\t                    登录                             \n");
        printf("\t\t*****************************************************\n");
        printf("\n\n");
        printf("\t\t\t\t1.用户名称：");
        scanf("%s",username);
        printf("\n\n");
```

```
            printf("\t\t\t\t2、密码：");
            scanf("%s",password);
            if(strcmp(username, "qt") == 0 && strcmp(password, "qt") == 0)
                {
                    beforeMainMenu();//strcmp 字符串比较函数
                    break;
                }
            else if(strcmp(username, "ht") == 0 && strcmp(password, "ht") == 0)
                {
                    backMainMenu();
                    break;
                }
            else
                {
                    system("cls");
                    printf("请输入正确的密码！！！（按回车键返回）");
                    getchar();getchar();//实现按回车键返回
                    number++;
                }
        }
    }
```

1．数组的概念

在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。在 C 语言中，数组属于构造数据类型。一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或构造类型。因此，按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组、结构数组等各种类别。

2．一维数组声明

声明数组的语法为在数组名后加上用方括号括起来的维数说明。这里仅介绍一维数组。一般形式如下：

```
类型说明符 数组名[常量表达式];
```

（1）类型说明符是任一种基本数据类型或构造数据类型。

（2）数组名是用户定义的数组标志符。

（3）方括号中的常量表达式表示数据元素的个数，也称为数组的长度。

例如：

```
int array[10];
```

这条语句定义了一个具有 10 个整型元素的名为 array 的数组。这些整数在内存中是连续存储的。数组的大小等于每个元素的大小乘上数组元素的个数。方括号中的维数表达式可以包含运算符，但其计算结果必须是一个长整型值。这个数组是一维的。下面的声明是合法的：

```
int array[5+3];
```

```
float count[5*2+3];
```

下面的声明是不合法的：

```
int n=10;
int array[n]; /*在声明时，变量不能作为数组的维数*/
```

对数组类型的说明应注意：

（1）数组的类型实际上是数组元素的取值类型。

（2）数组名的书写应符合标志符的书写规范。

（3）数组名不能与其他变量名相同。

（4）常量表达式表示数组元素的个数，但是其下标从 0 开始计算。

（5）不能在方括号中用变量来表示元素的个数，但是可以用符号常数或常量表达式。

（6）允许在同一个类型说明中，说明多个数组和多个变量。

3．一维数组的初始化

初始化赋值一般形式为：

```
类型说明符 数组名[常量表达式]={值，值.....};
```

C 语言对数组的初始化赋值还有以下规定：

（1）可以只给部分元素赋值。

（2）只能给元素逐个赋值，不能给数组整体赋值。

（3）如不给可初始化的数组赋初值，在数组说明中，可以不给出数组元素的个数。

例如，以下的赋值是合法的：

```
int array[10]={1,2,3,4};
```

给部分元素的赋值表如表 3.33 所示。

表 3.33　给部分元素的赋值表

a[0]	a[1]	a[2]	a[3]
1	2	3	4

```
Int array[6]={1,2,3,4,4,5};
```

给所有元素的赋值表如表 3.34 所示。

表 3.34　给所有元素的赋值表

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
1	2	3	4	4	5

```
Int array[]={1,1,1,1,1,1 };
```

不给出元素个数的赋值表如表 3.35 所示。

表 3.35　不给出元素个数的赋值表

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
1	1	1	1	1	1

4．一维数组的引用

数组元素是组成数组的基本单元。数组元素也是一种变量，其标志方法为数组名后跟一个下标。下标表示了元素在数组中的顺序号。

一般形式如下：

```
数组名[下标]
```

其中，下标只能为整型常量或整型表达式。如果是小数，则直接自动取整。

数组元素通常也称为下标变量。必须先定义数组，后使用下标变量。应先定义后使用，而不能一次引用整个数组。

例如，输出 5 个元素的数组必须使用循环语句逐个输出各下标变量：

```
for(i=0; i<5; i++)
printf("%d",b[i]);
```

而不能用一个语句输出整个数组。

下面的写法是错误的：

```
printf("%d",a);
```

【例 3.5】

```
//将 0-9 读入数组后按逆序输出
main()
{
   int i,a[10];
   for(i=0;i<=9;i++)
         a[i]=i;
   for(i=9;i>=0;i--)
         printf("%d ",a[i]);
}
```

5．字符数组

1）字符数组的声明

字符数组，用于存储和处理一个字符串，其定义格式与一维数值数组一样。

例如：

```
char str[10];
```

由于字符型和整型通用，也可以定义为 int str[10]，但这时每个数组元素占 2 个字节的内存单元。

字符数组也可以是二维或多维数组。

2）字符数组的初始化

（1）定义的时候直接用字符给数组赋值。

例如：

```
char str[12]= "how are you";
```

赋值后各元素的值如表 3.36 所示。

表 3.36　赋值后各元素的值

str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]	str[8]	str[9]	str[10]
h	o	w		a	r	e		y	o	u

注意：不能先定义后给它赋值，如 char a[12]; a[12]= "how are you";是错误的！

（2）对数组中字符逐个赋值。例如：

```
char str[]={'h','e','l','l','o'};
```

这时，str 数组的长度值自动定为 5，逐个元素赋值后的结果如表 3.37 所示。

表 3.37　逐个元素赋值后的结果

str[0]	str[1]	str[2]	str[3]	str[4]
h	e	l	l	o

3）字符数组的引用

字符数组的逐个字符引用，与引用数值数组元素类似。

【例 3.6】

```
//通过逐个字符输出来实现字符串的输出
main()
{
    int i;
    char str[5]={'h','e','l','l','o'};
    for(i=0;i<=4;i++)
        printf ("%c",str[i]);
}
```

4）字符串及其结束标志

C 语言的运算符根本无法操作字符串。在 C 语言中，把字符串当做数组来处理，因此，对字符串的限制方式与对数组的一样，特别是它们都不能用 C 语言的运算符进行复制和比较操作。

所谓字符串，是指若干有效字符的序列。C 语言中的字符串，可以包括字母、数字、专用字符、转义字符等。

C 语言规定：以'\0'作为字符串结束标志（'\0'代表 ASCII 码为 0 的字符，表示一个“空操作”，只起一个标志作用）。因此，可以对字符数组采用另一种方式进行操作——字符数组的整体操作。

注意：由于系统在存储字符串常量时，会在串尾自动加上 1 个结束标志，所以无须人为地再加 1 个。

另外，由于结束标志也要在字符数组中占用一个元素的存储空间，因此在说明字符数组长度时，至少为字符串所需长度加 1。

C 语言允许用字符串的方式对数组做初始化赋值。例如：

```
char c[]={'C', ' ','p','r','o','g','r','a','m'};
```

可写为：

```
char c[]={"C program"};
```

或去掉{}写为：

```
char c[]="C program";
```

用字符串方式赋值比用字符逐个赋值要多占一个字节，用于存放字符串结束标志'\0'。上面的数组 c 在内存中的实际存放情况为：

'\0'是由 C 编译系统自动加上的。由于采用了'\0'标志，所以在用字符串赋初值时一般无须指定数组的长度，而由系统自行处理。

C		p	r	o	g	r	a	m	\0

5）字符数组的输入/输出

在采用字符串方式后，字符数组的输入/输出将变得简单方便。

除了上述用字符串赋初值的方法外，还可用 printf()函数和 scanf()函数一次性输入/输出一个字符数组中的字符串，而不必使用循环语句逐个地输入/输出每个字符。

【例 3.7】

```
//通过字符数组实现字符串输出
main()
{
   char str[]="Visual Basic";
   printf("%s\n",str);
}
```

注意在本例的 printf()函数中，使用的格式字符串为“%s”，表示输出的是一个字符串，而在输出表列中给出数组名即可。不能写为：

```
printf("%s",str[]);
```

【例 3.8】

```
//用字符数组接收从键盘上输入的字符串，然后输出
main()
{
   char str[15];
   printf("input string:\n");
   scanf("%s",str);
   printf("%s\n",str);
}
```

在本例中，由于定义数组长度为 15，因此输入的字符串长度必须小于 15，以留出一个字节用于存放字符串结束标志'\0'。应该说明的是，对一个字符数组，如果不做初始化赋值，则必须说明数组长度。还应该特别注意的是，当用 scanf()函数输入字符串时，字符串中不能含有空格，否则将以空格作为串的结束符。

例如，当输入的字符串中含有空格时，运行情况为：

```
input string:
```

```
this is a book
```

输出为：

```
this
```

从输出结果可以看出，空格以后的字符都未能输出。为了避免这种情况，可多设几个字符数组分段存放含空格的串。

程序可改写成如下程序。

【例 3.9】

```
//输出带空格的字符串
main()
{
   char st1[6],st2[6],st3[6],st4[6];
   printf("请输入字符串:\n");
   scanf("%s%s%s%s",st1,st2,st3,st4);
   printf("%s %s %s %s\n",st1,st2,st3,st4);
}
```

该程序分别设了四个数组，输入了一行字符，用空格分隔分别存入四个数组。然后，分别输出这四个数组中的字符串。

scanf 的各输入项必须以地址方式出现，如&a,&b 等。但是，在例 3.9 中却是以数组名方式出现的，这是为什么呢?

这是因为在 C 语言中规定，数组名就代表了该数组的首地址。整个数组是以首地址开头的一块连续的内存单元。

如有字符数组 char c[10]，在内存中可如下表示。

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]

设数组 c 的首地址为 2000，也就是说，c[0]单元地址为 2000，则数组名 c 就代表这个首地址。因此，在 c 前面不能再加地址运算符&。例如，写成 scanf("%s",&c);是错误的。在执行函数 printf("%s",c)时，按数组名 c 找到首地址，然后逐个输出数组中各个字符，直到遇到字符串终止标志'\0'为止。

6）常用字符串处理函数

C 语言提供了丰富的字符串处理函数，大致可分为字符串的输入、输出、合并、修改、比较、转换、复制、搜索几类。使用这些函数可大大减轻编程的负担。用于输入/输出的字符串函数，在使用前应包含头文件"stdio.h"；若使用其他字符串函数，则应包含头文件"string.h"。

下面介绍 8 个最常用的字符串函数。

（1）字符串输出函数 puts()。

格式：puts (字符数组名)。

功能：把字符数组中的字符串输出到显示器，即在屏幕上显示该字符串。

说明：字符串中允许包含转义字符，输出时产生一个控制操作。

该函数一次只能输出一个字符串，而 printf()函数也能用来输出字符串，并且一次能输出多个。puts()函数完全可以由 printf()函数取代。当需要按一定格式输出时，通常使用 printf()

函数。

【例 3.10】

```
#include"stdio.h"
main()
{
   char str[]="Visual Basic";
   puts(str);
}
```

（2）字符串输入函数 gets()。

格式：gets(字符数组名)。

功能：从标准输入设备（键盘）上输入一个字符串。

说明：gets()函数读取的字符串长度没有限制，编程者要保证字符数组有足够大的空间，以存放输入的字符串。

该函数输入的字符串中允许包含空格，而 scanf()函数不允许。

该函数得到一个函数值，即为该字符数组的首地址。

【例 3.11】

```
#include <stdio.h>
main()
{
      char str[20];
      printf("please input a string:");
      gets(str);
      puts(str);
}
```

（3）字符串比较函数 strcmp()。

格式：strcmp(字符数组名 1,字符数组名 2)。

功能：按照 ASCII 码顺序比较两个数组中的字符串，并由函数返回值返回比较结果。

字符串 1=字符串 2，返回值等于 0。

字符串 2>字符串 2，返回值为正整数。

字符串 1<字符串 2，返回值为负整数。

说明：如果一个字符串是另一个字符串从头开始的子串，则母串为大。

不能使用关系运算符“= =”来比较两个字符串，只能用 strcmp()函数来处理。

该函数也可用于比较两个字符串常量，或比较字符型数组和字符串常量。

【例 3.12】

```
#include"string.h"
main()
{
      int answer;
      char str1[20],str2[20]="I am a student";
      printf("请输入一个字符串:");
```

```
        gets(str1);
        answer=strcmp(str1,str2);
        if (answer==0)
            printf("两个字符串相等");
        if(answer>0)
            printf("第一个字符串大");
        if(answer<0)
            printf("第一个字符串小");
    }
```

（4）字符串连接函数 strcat()。

格式：strcat(字符数组名 1,字符数组名 2)。

功能：把字符数组 2 中的字符串连接到字符数组 1 中字符串的后面，并删去字符串 1 后的串标志"\0"。该函数返回值是字符数组 1 的首地址。

说明：由于没有边界检查，编程者要注意保证“字符数组”定义得足够大，以便容纳连接后的目标字符串；否则，会因长度不够而产生问题。

连接前两个字符串都有结束标志'\0'，连接后的“字符数组”中存储的字符串的结束标志'\0'被舍弃，只在目标串的最后保留一个'\0'。

【例 3.13】

```
#include"string.h"
main()
{
    char str1[40]="你的职业是：",str2[20];
    printf("你的职业");
    gets(str2);
    strcat(str1,str2);
    puts(str1);
}
```

（5）字符串复制函数 strcpy()。

格式：strcpy (字符数组名 1,字符数组名 2)。

功能：把字符数组 2 中的字符串复制到字符数组 1 中。串结束标志'\0'也一同复制。字符数组名 2，也可以是一个字符串常量。这时，相当于把一个字符串赋予一个字符数组。

说明：字符数组必须定义得足够大，以便容纳复制过来的字符串。复制时，连同串结束标志'\0'一起复制。

不能用赋值运算符“=”将一个字符串直接赋值给一个字符数组，只能用 strcpy()函数来处理。

【例 3.14】

```
#include"string.h"
main()
{
    char str1[10],str2[10]="请跟我学";
    strcpy(str1,str2);
    printf("str1=%s",str1);
```

```
}
```

（6）测字符串长度函数 strlen()。

格式：strlen(字符数组名)。

功能：测字符串的实际长度（不含字符串结束标志'\0'）并作为函数返回值。

【例 3.15】

```
#include"string.h"
main()
{
    int len;
    char str[]="I am a student";
    len=strlen(str);
    printf("这个字符串的长度是:%d",len);
}
```

（7）将字符串中大写字母转换成小写字母——strlwr()函数。

格式：strlwr(字符串)。

功能：将字符串中的大写字母转换成小写字母，其他字符（包括小写字母和非字母字符）不转换。

【例 3.16】

```
#include"string.h"
main()
{
    char str[20]="I AM A STUDENT";
    printf("我变小写了:%s",strlwr(str));
}
```

（8）将字符串中小写字母转换成大写字母——strupr()函数。

格式：strupr(字符串)。

功能：将字符串中的小写字母转换成大写字母，其他字符（包括大写字母和非字母字符）不转换。

【例 3.17】

```
main()
{
    char str[20]="i am a student";
    printf("我变大写了:%s",strupr(str));
}
```

（1）将 1，2，3，…，50 依次存在数组元素 num[1],num[2],…,num[50]中，并打印输出数组元素的值。

（2）定义一个有 10 个元素的一维数组 count，从键盘上输入 8 个整数，将其按从大到小的顺序排列，并将排列后的数组输出。

（3）统计一行字符串中每个小写英文字符出现的次数。

（4）写出下列程序的运行结果：

```
#include<stdio.h>
main()
{
int i,max;
int a[10]={6,44,3,66,56,7,90,22,45,83};
max=a[0];
for(i=1;i<=9;i++)
if(a[i]>max)
max=a[i];
printf("max=%d\n",max);
}
```

拓展任务

实现登录时密码以“*”的方式显示。

3.4.3 子任务三：数据结构设计

任务描述

在超市管理系统中，有商品维护、商品销售、库存预警、会员管理模块，前三个模块都是对商品数据的处理，后一个模块是对会员数据的处理，在数据处理之前，要对数据进行分类、整理并优化，形成完整的数据结构。

具体要求：

（1）减少数据冗余。

（2）保证数据的完整性和安全性。

任务分析与设计

（1）根据“超市管理系统”项目开发的背景分析、系统的详细设计中的数据结构设计，确定超市管理系统的数据结构，分别见表 3.30 和表 3.31（参见 3.3.2 节）。

（2）从数据类型中分析得出，生产日期为日期型，也应该定义成结构体。

（3）设计商品结构体。

（4）设计会员结构体。

任务实现

步骤一：自定义头文件，该文件中只用来存放结构体。

步骤二：

（1）定义日期结构体。

（2）定义商品结构体。

（3）定义会员结构体。

步骤三：编码实现。

```
//定义日期结构体
struct date
{
    int year;//年
    int month;//月
    int day;//日
};
//定义商品结构体
struct goods
{
    int goodsNumber;//商品编号
    char goodsName[20];//商品名称
    char goodsType[8];//商品类型
    struct date produceDate;//生产日期
    int assureDate;//保质期（月）
    float stockPrice;//商品进价
    int amount;//商品数量
    float salePrice;//销售单价
};
//定义会员结构体
struct member{
    int memberNumber;//会员编号
    char identity[19];//身份证编号
    char memberName[20];//会员名称
};
```

1．自定义头文件

1）头文件的作用

头文件的主要作用是，统一命名多个源程序文件中都要用到的符号常量、函数声明、用户自定义的结构类型名、类型的别名定义等。

例如，对于“超市管理系统”，可以由 4 个程序（如商品管理、销售管理、会员管理、库存预警）开发小组合作。每个小组都建立一个 C 源程序文件，存放自己小组的所有程序，但各小组可以共享多个结构变量的定义、函数的声明。

只需建立一个扩展名为.h 的纯文本文件即可，本任务的所有结构体都可以放在一个.h 文件中。

2）自定义头文件的建立

（1）选择“文件”→“新建”命令，打开“新建”对话框，选择“文件”选项卡中的“C/C++ Header File”，如图 3.17 所示。

（2）单击“确定”按钮，function.h 文件建立成功。

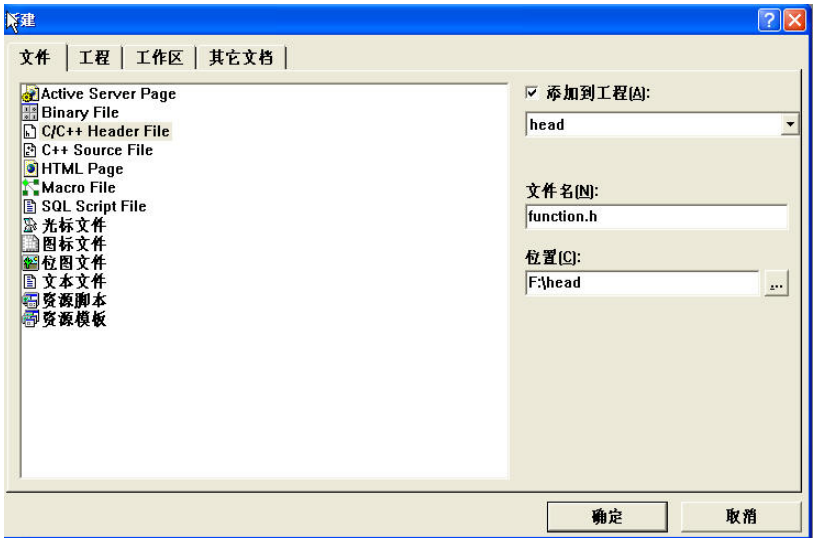


图 3.17　“新建”对话框

3）说明

（1）自定义头文件，在进行编译预处理时，使用如下格式：

```
#include "filepath"
```

如果头文件放在与源文件相同的目录下，则只需给出文件名（头文件的默认路径就是源文件所在的路径），如#include "f.h"；如果源程序文件与头文件在不同的目录中，则需要给出头文件的文件路径，如#include "c:\fun\f.h"。

（2）如果要包含库函数所使用的头文件，则一般使用如下格式：

```
#include <filepath>
```

（3）如果在一个源文件里同时包含自己的头文件和系统的头文件，则一般把包含系统的头文件的命令写在前面，这样做的目的是防止本程序的局部定义影响到库文件里的定义。

2．结构体

1）结构体的定义

```
struct   结构类型名   /*struct 是结构类型关键字*/
            {
                数据类型   数据项 1;
                数据类型   数据项 2;
                ……
                数据类型   数据项 n;
                };    /*此处分号不能省！*/
```

说明：

（1）“结构体名”用做结构体类型的标志，它又称“结构体标记”。类型与变量是不同的概念，不要混同。只能对变量赋值、存取或运算，而不能对一个类型赋值、存取或运算。在编译时，对类型是不分配空间的，只对变量分配空间。类型名的取名规则遵从标志符。

（2）成员列表为本结构体类型所包含的若干个成员的列表，必须用“{ }”括起来，并以

分号结束。每个成员的形式为：

```
类型标志符　成员名;
```

成员的类型可为除该结构体类型外的任何一种类型，如基本类型、指针类型和结构体类型。

若定义结构体类型 struct date 如下：

```
struct date
{
int year;
    int month;
    int day;
  };
```

则结构体类型 struct student 中的成员 birthday 就可定义为：

```
struct date birthday;
```

这样，birthday 就不是字符型数组，而是结构体类型 struct date 的变量。

（3）成员名命名规则与变量名相同。同一结构体类型中的各成员不可互相重名，但不同结构体类型间的成员可以重名，并且成员名还可与程序中的变量重名，因为它们代表着不同的对象。

（4）数据类型相同的数据项，既可逐个、逐行分别定义，也可合并成一行定义。

例如，上述对 date 的定义也可改为如下形式：

```
struct  date
    {
      int  year, month, day;
    };
```

（5）结构类型中的数据项，既可以是基本数据类型，也允许是另一个已经定义的结构类型。例如：

```
struct  std_info
        {char  no[7];
         char  name[9];
         char  sex[3];
         struct date birthday;
        };
```

2）结构变量的定义

用户自己定义的结构类型，与系统定义的标准类型（int、char 等）一样，可用来定义结构变量的类型。定义结构变量的方法，可概括为以下两种。

（1）先定义结构类型，后定义结构变量，一般形式为：

```
struct  结构体名  变量名表;
```

例如：

```
struct  student  stud1, stud2;
```

```
struct date    date_1;
```

定义了 3 个结构体类型的变量 stud1、stud2 和 date_1，前两个是 struct student 类型的结构体变量，后者是 struct date 类型的结构体变量。

（2）在声明结构体类型的同时定义变量，一般形式是：

```
struct   结构体名
  {
     成员表列
   }变量名表;
```

例如：

```
struct date
{ int year;
  int month;
  int day;
} date_2, date_3;
```

同样可定义 date_2, date_3 为 struct date 类型的变量。

（3）直接定义结构体类型变量，其一般形式为：

```
struct
  {
     成员表列;
   }变量名表;
```

即不出现结构体名。例如：

```
struct
      {char num[5];
        char name[10];
        char sex;
        struct date birthday;
        float score[3];
        char mobiletel[11];
       }stud3,stud4;
```

定义了 2 个结构体类型的变量 stud3 和 stud4。

3）结构变量成员的表示方法

在程序中使用结构变量时，往往不把它作为一个整体来使用。在 ANSI C 中除了允许具有相同类型的结构变量相互赋值外，一般对结构变量的使用，包括赋值、输入、输出、运算等都是通过结构变量的成员来实现的。

表示结构变量成员的一般形式是：

```
结构变量名.成员名
```

例如：

```
stud1.num          即第一个人的学号
stud2.sex          即第二个人的性别
```

在 C 语言的运算符中，因取成员运算符“.”优先级最高，故以上语句均为对引用之后的成员变量进行操作。若结构体定义是嵌套的，则只能引用最低级的成员（用若干“.”运算符，逐级引用到最低级）。例如：

```
stud3.birthday.year
```

是合法的，而

```
stud3.year
```

是非法的。

4）结构变量的赋值

结构变量的赋值就是给各成员赋值，可用输入语句或赋值语句来完成。

【例 3.18】 给结构变量赋值并输出其值。

```
main()
{
    struct student
    {
       int num;
       char *name;
       char sex;
       float score;
    } stud1,stud2;
    stud1.num=1001;
    stud1.name="王一";
    printf("请输入性别及成绩\n");
    scanf("%c %f",&boy1.sex,&stud1.score);
    stud2=stud1;
    printf("学号=%d\n 姓名=%s\n",stud2.num,stud2.name);
    printf("性别=%c\n 成绩=%f\n",stud2.sex,stud2.score);
}
```

在本程序中，用赋值语句给 num 和 name 两个成员赋值，name 是一个字符串指针变量。用 scanf()函数动态地输入 sex 和 score 成员值，然后把 stud1 的所有成员的值整体赋予 stud2，最后分别输出 stud2 的各个成员值。本例表示了结构变量的赋值、输入和输出的方法。

5）结构变量的初始化

与其他类型变量一样，对结构变量可以在定义时进行初始化赋值。

【例 3.19】 初始化结构变量。

```
main()
{
    struct student      /*定义结构*/
    {
       int num;
       char *name;
       char sex;
       float score;
```

```
    }stud2,stud1={1001,"王一",'男',98.5};
  stud2=stud1;
 printf("学号=%d\n 姓名=%s\n",boy2.num,boy2.name);
 printf("性别=%c\n 成绩=%f\n",boy2.sex,boy2.score);
}
```

在本例中，stud2、stud1 均被定义为外部结构变量，并对 stud1 做了初始化赋值。在 main() 函数中，把 stud1 的值整体赋予 stud2，然后用两个 printf 语句输出 stud2 各成员的值。

6）结构数组的定义

数组的元素也可以是结构类型的。因此，可以构成结构型数组。结构数组的每一个元素都是具有相同结构类型的下标结构变量。在实际应用中，经常用结构数组来表示具有相同数据结构的一个群体，如一个班的学生档案、一个车间职工的工资表等。

方法和结构变量相似，只需说明它为数组类型即可。

例如：

```
struct student
{
    int num;
    char *name;
    char sex;
    float score;
}stud[5];
```

定义了一个结构数组 stud，共有 5 个元素，stud[0]～stud[4]。每个数组元素都具有 struct student 的结构形式。对结构数组可以做初始化赋值。

例如：

```
struct student
    {
        int num;
        char *name;
        char sex;
        float score;
    }stud [5]={
            {1001,"王一 ","男",95},
            {1002,"张键","男",82.5},
            {1003,"李平","女",92.5},
            {1004,"刘玲","女",87},
            {1005,"孙浩","男",68};
}
```

当对全部元素做初始化赋值时，也可不给出数组长度。

【例 3.20】

计算学生的平均成绩和不及格的人数。

```
struct student
    {
        int num;
```

```
        char *name;
        char sex;
        float score;
    }stud [5]={
            {1001,"王一 ","男",95},
            {1002,"张键","男",82.5},
            {1003,"李平","女",92.5},
            {1004,"刘玲","女",87},
            {1005,"孙浩","男",68};
            }
main()
{
    int i,c=0;
    float ave,s=0;
    for(i=0;i<5;i++)
    {
      s+=stud[i].score;
      if(stud[i].score<60) c+=1;
    }
    printf("s=%f\n",s);
    ave=s/5;
    printf("平均分=%f\n 人数=%d\n",ave,c);
}
```

在本例程序中，定义了一个外部结构数组 stud，共 5 个元素，并做了初始化赋值。在 main() 函数中用 for 语句逐个累加各元素的 score 成员值存于 s 之中，如 score 的值小于 60（不及格）即计数器 C 加 1，循环完毕后计算平均成绩，并输出全班总分、平均分及不及格人数。

即时训练

（1）阅读下面程序，说明其功能：

```
#include"stdio.h"
#define NUM 3
struct mem
{
    char name[20];
    char phone[10];
};
main()
{
    struct mem man[NUM];
    int i;
    for(i=0;i<NUM;i++)
      {
        printf("input name:\n");
        gets(man[i].name);
        printf("input phone:\n");
```

```
            gets(man[i].phone);
          }
        printf("name\t\t\tphone\n\n");
        for(i=0;i<NUM;i++)
          printf("%s\t\t\t%s\n",man[i].name,man[i].phone);
}
```

（2）阅读下面程序，说明其功能：

```
#include <stdio.h>
  void main()
{
    struct   childrec/*定义结构体*/
    {
      char initial;              /*姓名首字母     */
      int age;                   /*年龄           */
      int grade;                 /*考试成绩       */
    }boy,girl;

    boy.initial = 'R';
    boy.age = 15;
    boy.grade = 75;

    girl.age = boy.age - 1;     /*女孩比男孩小一岁  */
    girl.grade = 82;
    girl.initial = 'H';

    printf(" girl:   %c is %d years old and got a grade of %d\n",
              girl.initial, girl.age, girl.grade);

    printf("   boy:   %c is %d years old and got a grade of %d\n",
              boy.initial, boy.age, boy.grade);
}
```

（3）班级选举班长，现要求编写程序，对候选人得票进行统计，设有 5 个候选人，每次输入一个得票的候选人的名字，要求最后输出各人得票结果。

（4）有 5 个学生的信息，其中包括学号、姓名、3 门课的成绩。要求编写程序，实现从键盘输入该 5 名学生的信息后，即可打印出他们的相关信息（包括学号、姓名、3 门课的成绩、平均分）。

拓展任务

设计一个头文件，将“超市管理系统”涉及的结构体都设计在其中。

3.4.4 子任务四：商品维护模块的功能实现

任务描述

商品维护是超市管理系统中的主要模块，任何操作都与商品分不开。在这个模块中，我们重点实现添加商品的功能。在第一次添加商品时，需要新建商品（goods）文件，以后的添加是对文件的追加。

具体要求：

（1）在源文件所在文件夹下，新建一个 data 文件夹，专门用于存放各种数据文件。

（2）添加每一项时都要有具体的提示信息，以保证用户添加的准确性。

（3）商品编号要唯一，因此当添加相同商品编号时，要给出不能添加的提示。

任务分析与设计

（1）商品维护模块的基本流程如表 3.38 所示。

表 3.38 商品维护模块的基本流程

1．定义商品维护函数	2．呈现界面（如图 3.11 所示），显示“请输入商品的编号”
3．键盘输入商品编号（要求是数值数据）	4．显示“请输入商品的名称”的提示
5．键盘输入商品名称（要求是字符型数据）	6．显示“请输入商品的生产日期”的提示，格式为 YYYY-MM-DD
7．键盘输入生产日期（要求是 date 型数据）	8．显示“请输入商品的保质期（以天为单位）”
9．键盘输入商品的保质期（要求是数值型数据）	10．显示“请输入商品的类型”
11．键盘输入商品的类型（要求是字符型数据）	12．显示“请输入商品的进价”
13．键盘输入商品的进价（要求是浮点型数据）	14．显示“请输入商品的数量”
15．键盘输入商品的数量（要求是整型数据）	16．显示“请输入商品的销售单价”
17．键盘输入商品销售单价（要求是数值型数据）	18．显示“确认添加商品吗？（Y/N）”
19．如果选择“Y”，则将该商品添加到商品文件（goods）中； 如果选择“N”，则显示“继续添加商品吗？（Y/N）”	20．如果选择“Y”，则继续添加商品； 如果选择“N”，则返回后台主界面

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义商品（goods）变量，指针变量。

S1.2：初始化商品进价和销售单位的初始值为 0。

S2：输入商品信息。

S2.1：输入商品的编号。

S2.2：输入商品的名称。

S2.3：输入商品的生产日期。

S2.4：输入商品的保质期。

S2.5：输入商品的类型。

S2.6：输入商品的进价。

S2.7：输入商品的数量。

S2.8：输入商品的销售价格。

S3：确认商品添加。

S3.1：如果添加，则将该商品写入商品（goods）文件中。

S3.2：如果不添加，则执行 S4。

S4：确认是否继续添加。

S4.1：如果继续添加，则返回 S2。

S4.2：否则，返回后台主界面。

（3）商品维护流程图如图 3.18 所示。

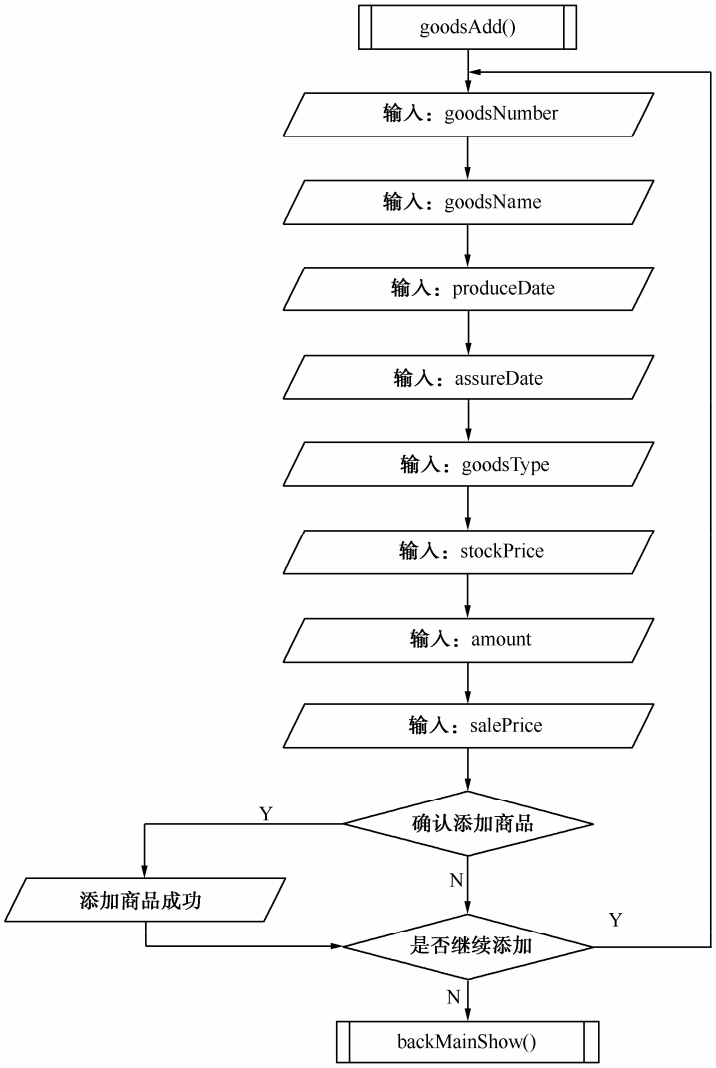


图 3.18　商品维护流程图

任务实现

步骤一：引入自定义头文件 struct.h，格式为#include "struct.h"。

步骤二：

（1）定义商品维护函数 goodsAdd()。

（2）声明商品维护函数 goodsAdd()。

（3）调用商品维护函数，在后台主界面中进行调用。

步骤三：编码实现。

```
    void goodsAdd()
{
    int i =1;
    struct goods sp;
    FILE *fp;
    char con;//是否继续
    sp.stockPrice=0;
    sp.salePrice=0;
    /*有些编译器是根据程序中有浮点数操作才会将浮点运算库链接进来的，如果没有浮点数操作，比如浮点数赋值、加减运算等，编译器可能就没有将浮点运算库链接到程序中，然后调用 scanf 输入浮点数的时候就会出错。为此，我们对商品进价和销售单位进行了初始化。sp.stockPrice=0;和 sp.salePrice=0;这两条语言，可以根据自己编译器的情况自行取舍*/
    while(1) //实现循环添加商品
    {
        //程序的主界面
        system("cls");//清屏
        printf("\n\n");
        printf("\t\t*****************************************************\n");
        printf("\t\t                  商品维护                           \n");
        printf("\t\t*****************************************************\n");
        printf("\n\n");
        //商品录入
        printf("请输入商品的编号：");
       scanf("%d",&sp.goodsNumber);
        printf("请输入商品的名称：");
        scanf("%s",&sp.goodsName);
        printf("请输入商品的生产日期(YYYY-MM-DD)：");

scanf("%d-%d-%d",&sp.produceDate.year,&sp.produceDate.month,&sp.produceDate.day);
        printf("请输入商品的保质期：");
        scanf("%d",&sp.assureDate);
        printf("请输入商品的类型：");
        scanf("%s",&sp.goodsType);
        printf("请输入商品的进价");
        scanf("%f",&sp.stockPrice);
        printf("请输入商品的数量：");
        scanf("%d",&sp.amount);
        printf("请输入商品的销售单价：");
        scanf("%f",&sp.salePrice);
        printf("\n 确认添加商品吗？(Y/N)：");
        scanf(" %c",&con);
        if(con=='Y' || con=='y')
        {
```

```
                    if((fp=fopen("data\\goods.dat","ab"))==NULL)
                    {
                        fp=fopen("data\\goods.dat","wb");
                          break;
                    }
                  if((fwrite(&sp,sizeof(struct goods),1,fp))!=1)
                  {
                        printf("\n\n 添加失败！！！\n");
                    }
                    else
                    {
                        printf("\n\n 添加商品成功！\n");//保存商品
                    }
                  fclose(fp);
              }
              //返回确认处理
              printf("\n\n 继续添加商品吗？(Y/N)：");
              scanf(" %c",&con);
              if(con=='N' || con=='n')
              {
                  break;//返回后台主界面
              }
          }

    }
```

1．文件的概念

文件是指一组相关数据的有序集合，在 C 程序设计中具有重要作用。操作系统以文件为单位对计算机内的数据进行管理，在开发一些较大的应用程序时，程序中的数据可以以文件的形式存储在外部设备上，以减轻程序的负载。

2．文件的类型

从文件编码方式来看，文件可分为 ASCII（也称文本文件）码文件和二进制文件两种。

1）文本文件

文本文件在磁盘上的存储形式是每个字符对应一个字节，字符以对应的 ASCII 形式存储。例如，源程序文件就是 ASCII 文件，ASCII 文件可以在终端显示出来，因为它们都是可打印的字符。

2）二进制文件

二进制文件是按二进制的编码方式来存放文件的。由于一些数据的编码可能是不可打印字符的计算机内部码，因此二进制文件一般无法读懂。

把一个文本文件读入内存时，要将 ASCII 码转换成二进制码，而把文件以文本方式写入磁盘时，也要把二进制码转换成 ASCII 码，因此文本文件的读/写要花费较多的转换时间。对二进制文件的读/写不存在这种转换。

3．文件的打开与关闭

文件在进行读/写操作之前要先打开，使用完毕要关闭。所谓打开文件，实际上是建立文件的各种有关信息，并使文件指针指向该文件，以便进行其他操作。关闭文件则断开指针与文件之间的联系，也就禁止再对该文件进行操作。

在 C 语言中，文件操作都是由库函数来完成的。

1）文件的打开（fopen函数）

功能：fopen 函数用来打开文件。

格式：文件指针名=fopen（文件名,使用文件方式）。

说明：

（1）“文件指针名”必须是被说明为 FILE 类型的指针变量。

（2）“文件名”是被打开文件的文件名。

（3）“使用文件方式”是指文件的类型和操作要求。

（4）“文件名”是字符串常量或字符串数组。

例如：

```
FILE *fp;
fp=("file_a","r");
```

其意义是在当前目录下打开文件 file_a，只允许进行“读”操作，并使 fp 指向该文件。

例如：

```
FILE *fphzk;
fphzk=("c:\\hzk16","rb");
```

其意义是打开 C 驱动器磁盘的根目录下的文件 hzk16，这是一个二进制文件，只允许按二进制方式进行读操作。两个反斜线“\\”中的第一个表示转义字符，第二个表示根目录。文件使用方式共有 12 种，下面给出了它们的符号和意义。

文件使用方式如下。

（1）"rt"：只读打开一个文本文件，只允许读数据。

（2）"wt"：只写打开或建立一个文本文件，只允许写数据。

（3）"at"：追加打开一个文本文件，并在文件末尾写数据。

（4）"rb"：只读打开一个二进制文件，只允许读数据。

（5）"wb"：只写打开或建立一个二进制文件，只允许写数据。

（6）"ab"：追加打开一个二进制文件，并在文件末尾写数据。

（7）"rt+"：读、写打开一个文本文件，允许读和写。

（8）"wt+"：读、写打开或建立一个文本文件，允许读、写。

（9）"at+"：读、写打开一个文本文件，允许读，或在文件末尾追加数据。

（10）"rb+"：读、写打开一个二进制文件，允许读和写。

（11）"wb+"：读、写打开或建立一个二进制文件，允许读和写。

（12）"ab+"：读、写打开一个二进制文件，允许读，或在文件末尾追加数据。

对于文件使用方式有以下说明。

（1）文件使用方式由 r、w、a、t、b、+这 6 个字符拼成，各字符的含义如下。

① r（read）：读。

② w（write）：写。

③ a（append）：追加。

④ t（text）：文本文件，可省略不写。

⑤ b（banary）：二进制文件。

⑥ +：读和写。

（2）凡用"r"打开一个文件时，该文件必须已经存在，并且只能从该文件读出。

（3）用"w"打开的文件只能向该文件写入。若打开的文件不存在，则以指定的文件名建立该文件；若打开的文件已经存在，则将该文件删去，重建一个新文件。

（4）若要向一个已存在的文件追加新的信息，只能用"a"方式打开文件，但此时该文件必须是存在的，否则将会出错。

（5）在打开一个文件时，如果出错，fopen 则返回一个空指针值 NULL。在程序中可以用这一信息来判别是否完成打开文件的工作，并做相应的处理。因此，常用以下程序段打开文件：

```
if((fp=fopen("c:\\hzk16","rb")==NULL)
{
printf("\nerror on open c:\\hzk16 file!");
getch();
exit(1);
}
```

这段程序的意义是，如果返回的指针为空，则表示不能打开 C 盘根目录下的 hzk16 文件，将给出提示信息“error on open c:\ hzk16 file!”，下一行 getch()的功能是从键盘输入一个字符，但不在屏幕上显示。

2）文件的关闭

fclose 函数

功能：fclose 函数用来关闭文件。

格式：fclose(文件指针);

例如：fclose(fp);

说明：正常完成关闭文件操作时，fclose 函数返回值为 0。如返回非零值，则表示有错误发生。

4．文件的块读/写函数（fread和fwrite）

C 语言还提供了用于整块数据的读/写函数，可用来读/写一组数据，如一个数组元素、一个结构变量的值等。

1）写数据块函数

调用的一般形式为：fwrite(buffer,size,count,fp);

例如，有如下结构体：

```
struct st
{
char num[8];
float mk[5];
}pers[30];
```

以下循环将把这 30 个元素中的数据输出到 fp 所指文件中。

```
for(i=0;i<30;i++)
fwrite(&pers[i],sizeof(struct st),1,fp);
```

2）读数据块函数（fread）

调用的一般形式为：fread(buffer,size,count,fp);

例如，以下语句从 fp 所指的文件中再次将每个学生数据逐个读入到 pers 数组中。

```
i=0;
fread(&pers[i],sizeof(struct st),1,fp);
while(!feof(fp))
{ i++;
fread(&pers[i],sizeof(struct st),1,fp);
}
```

说明：

（1）buffer 是一个指针，在 fread()函数中，它表示存放输入数据的首地址。在 fwrite()函数中，它表示存放输出数据的首地址。

（2）size 表示数据块的字节数。

（3）count 表示要读/写的数据块的块数。

（4）fp 表示文件指针。

例如，fread(fa,4,5,fp); 的意义是从 fp 所指的文件中，每次读 4 个字节（一个实数）送入实数组 fa 中，连续读 5 次，即读 5 个实数到 fa 中。

【例 3.21】

```
//从键盘输入两个学生数据，写入一个文件中，再读出这两个学生的数据显示在屏幕上
#include<stdio.h>
struct stu
{
    char name[10];
    int num;
    int age;
    char addr[15];
    }boya[2],boyb[2],*pp,*qq;

main()
{
    FILE *fp;
    char ch;
    int i;
    pp=boya;
    qq=boyb;
    if((fp=fopen("stu_list","wb+"))==NULL)
    {
    printf("Cannot open file strike any key exit!");
    getch();
```

```
        exit(1);
    }
    printf("input data");
    for(i=0;i<2;i++,pp++)
        scanf("%s%d%d%s",pp->name,&pp->num,&pp->age,pp->addr);
        pp=boya;
    fwrite(pp,sizeof(struct stu),2,fp);
    rewind(fp);
    fread(qq,sizeof(struct stu),2,fp);
    printf("name number age addrn");
    for(i=0;i<2;i++,qq++)
        printf("%st%5d%7d%sn",qq->name,qq->num,qq->age,qq->addr);
    fclose(fp);
    }
```

本例程序定义了一个结构 stu，说明了两个结构数组（boya 和 boyb）和两个结构指针变量（pp 和 qq）。pp 指向 boya，qq 指向 boyb。程序的第 16 行以读/写方式打开二进制文件“stu_list”，输入两个学生数据之后，写入该文件中，然后把文件内部位置指针移到文件首，读出两块学生数据后，在屏幕上显示。

5．sizeof操作符

sizeof 是 C 语言的一种单目操作符，如 C 语言的其他操作符++、--等。它并不是函数。sizeof 操作符以字节形式给出了其操作数的存储大小。操作数可以是一个表达式或括在括号内的类型名。操作数的存储大小由操作数的类型决定。

1）sizeof 的使用方法

（1）用于数据类型。

sizeof 的使用形式：sizeof(type)。

数据类型必须用括号括住，如 sizeof(int)。

（2）用于变量。

sizeof 的使用形式：sizeof(var_name)或 sizeof var_name。

变量名可以不用括号括住，如 sizeof(var_name)，sizeof var_name 等都是正确形式。带括号的用法更普遍，大多数程序员采用这种形式。

注意：sizeof 操作符不能用于函数类型、不完全类型或位字段。不完全类型指具有未知存储大小的数据类型，如未知存储大小的数组类型、未知内容的结构或联合类型、void 类型等。

例如 sizeof(max)，若此时变量 max 定义为 int max(),sizeof(char_v)，char_v 定义为 char char_v [MAX]且 MAX 未知，则 sizeof(void)都不是正确形式。

2）sizeof 的结果

sizeof 操作符的结果类型是 size_t，它在头文件中，typedef 为 unsigned int 类型。该类型保证能容纳实现所建立的最大对象的字节大小。

（1）若操作数具有类型 char、unsigned char 或 signed char，则其结果等于 1。

ANSI C 正式规定字符类型为 1 字节。

（2）int、unsigned int、short int、unsigned short、long int、unsigned long、float、double、long double 类型的 sizeof 在 ANSI C 中没有具体规定，大小依赖于实现，一般可能分别为 2、

2、2、2、4、4、4、8、10。

（3）当操作数是指针时，sizeof 依赖于编译器。例如在 Microsoft C/C++7.0 中，near 类指针字节数为 2，far、huge 类指针字节数为 4。一般 UNIX 的指针字节数为 4。

（1）程序填空：

```
//有 5 个学生，每个学生有 3 门课的成绩，从键盘输入以上数据（包括学号、姓名、3 门课的成绩），计算出平均成绩，将原有的数据和计算出的平均分数存放在磁盘文件"stud"中
#include "stdio.h"
struct student
{
    char num[6];
    char name[8];
    int score[3];
    float avr;
} stu[5];
main()
{
    int i,j,sum;
    FILE *fp;
    /*输入 5 个学生的相关数据*/
    for(i=0;i<5;i++)
      {
          printf("\n 请输入第 %d 位学生的信息:\n",i);
          printf("学号:");
          scanf("%s",stu[i].num);
          printf("姓名:");
          scanf("%s",stu[i].name);
          sum=0;
              for(j=0;j<3;j++)
              {
                  printf("score %d.",j+1);
                  scanf("%d",&stu[i].score[j]);
                  sum+=stu[i].score[j];
              }
          stu[i].avr=sum/3.0;
      }
    fp=fopen("stud","w");
    for(i=0;i<5;i++)
    if(fwrite(&stu[i],sizeof(struct student),1,fp)!=1)
    printf("文件写错误\n");
    fclose(fp);
}
```

（2）将 5 名职员的数据信息从键盘输入，然后送入磁盘文件“employee”中保存。设计

员工数据包括员工号、姓名、性别、年龄、工资；再从磁盘调入这些数据，依次打印出来（用fread()函数和 fwrite()函数编写）。

（3）在“employee”的基础上，增加一个新员工的数据，要求按工资高低顺序插入到原有的文件中，然后存放到 employee_sort 中。

拓展任务

实现“超市管理系统”的“用户管理模块——添加用户”的功能。

3.4.5 子任务五：会员管理——会员添加模块的功能实现

任务描述

会员添加模块主要实现新会员的注册功能。

具体要求：

（1）实现会员自动编号。

（2）添加会员基本信息，即顾客姓名和身份证号。

任务分析与设计

（1）会员添加模块的基本流程如表 3.39 所示。

表 3.39 会员添加模块的基本流程

1．定义会员添加函数	2．呈现界面（如图 3.19 所示），显示会员编号,系统提示“请输入会员的姓名”
3．键盘输入会员的姓名（要求是字符数据）	4．显示“请输入会员的身份证号码”
5．键盘输入会员的身份证号（要求是字符型数据）	6．确认是否添加会员（Y/N）
7．如果选择“Y”，则将会员信息写入会员（member）文件中	8．如果选择“N”，则不添加该会员
9．确认是否继续添加会员（Y/N）	10．如果选择“Y”，则继续添加会员，返回第 2 步
11．如果选择“N”，则返回会员管理界面	

图 3.19　会员添加界面

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义会员结构体（member）变量、指针变量 fp 和 bhfp。

S1.2：初始化编号的初值为 0。

S2：输入会员信息。

S2.1：会员自动编号。

S2.1.1：定义会员编号文件指针（*bhfp）。

S2.1.2：打开会员编号文件（memberNumber）。

S2.1.3：向会员编号文件写入数据“1”，作为会员的第一个编号，以后分别在这个基础上进行累加。

S2.1.4：关闭会员编号文件。

S2.2：输入会员的姓名。

S2.3：输入会员的身份证号。

S3：确认会员添加。

S3.1：如果添加，则将该会员写入会员（member）文件中。

S3.2：如果不添加，则执行 S4。

S4：确认是否继续添加。

S4.1：如果继续添加，则返回 S2。

S4.2：否则，返回会员管理界面。

（3）会员添加流程图如图 3.20 所示。

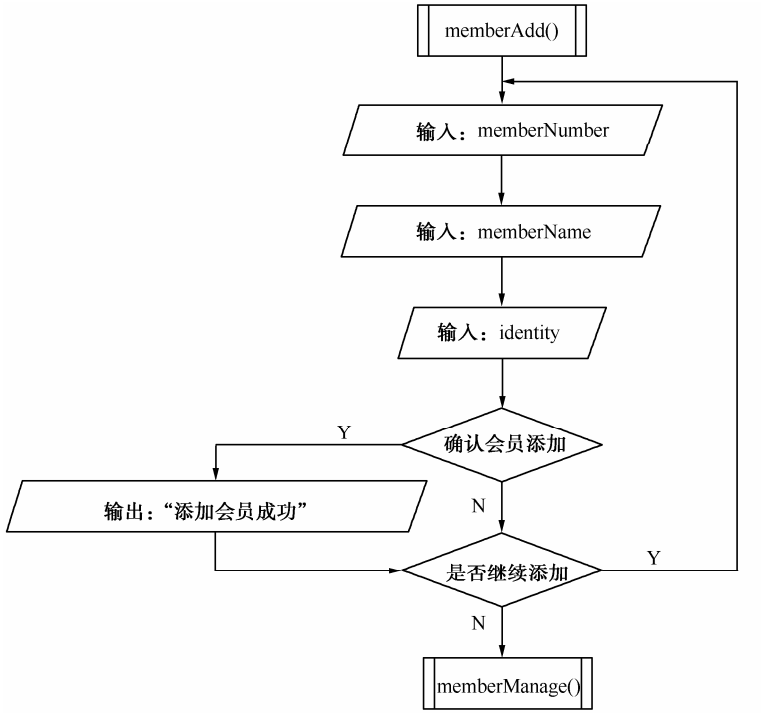


图 3.20　会员添加流程图

任务实现

步骤一：

（1）定义会员添加函数 memberAdd()。

（2）声明会员添加函数 memberAdd()。

（3）调用会员添加函数，在会员管理模块中进行调用。

步骤二：编码实现。

```
//添加会员
  void memberAdd()
{
    //初始化数据
    struct member hy;
    FILE *fp,*bhfp;
    int bh=1;
    char con;

    //追加打开一个二进制文件，并在文件末尾写数据
    if((fp=fopen("data\\member.dat","ab"))==NULL)
    {
        fp=fopen("data\\member.dat","wb");
        return;
    }

    //实现循环添加功能
    while(1)
    {
    if((bhfp=fopen("data\\memberNumber.dat","r"))==NULL)
        {
            printf("cannot open file,press any key create file!\n");
            bhfp=fopen("data\\memberNumber.dat","wb");

            //因为 getw 返回值是-1，所以在运行这条语句前要初始化编号为 1
            putw(1,bhfp);
            fclose(bhfp);
            getchar();getchar();
            memberManage();
        }
        else
        {bh = getw(bhfp);}
        hy.memberNumber = bh;
        fclose(bhfp);
```

```
//程序的主界面
system("cls");

printf("\n\n");
printf("\t\t*****************************************************\n");
printf("\t\t                    会员添加                         \n");
printf("\t\t*****************************************************\n");
printf("\n\n");

//会员录入
printf("会员的编号：\t%d\n",bh);
printf("请输入会员的姓名：");
scanf("%s",&hy.memberName);
printf("请输入会员的身份证号码：");
scanf("%s",&hy.identity);
printf("\n 确认添加会员吗？(Y/N)：");
scanf(" %c",&con);
if(con=='Y' || con=='y')
{
    if((fwrite(&hy,sizeof(struct member),1,fp))!=1)
    {
        printf("\n\n 添加失败！！！\n");
    }
    else
    {
        printf("\n\n 添加会员成功！\n");

        //保存会员
        if((bhfp=fopen("data\\memberNumber.dat","w"))==NULL)
        {
           printf("cannot open file\n");
           putw(1,bhfp);
           return;
        }

        //实现会员编号加 1
        putw(bh+1,bhfp);
        fclose(bhfp);
        getchar();
        getchar();
    }
}

//返回确认处理
printf("\n\n 继续添加会员吗？(Y/N)：");
scanf(" %c",&con);
if(con=='N' || con=='n')
```

```
                {
                    //返回会员管理界面
                    break;
                }
            }
            fclose(fp);
        }
```

1．getw()函数和putw()函数

1）getw()函数

功能：从流中取一整数。

用法：int　getw(FILE　*s　trem);

2）putw()函数

功能：把一字符或字送到流中。

用法：int　putw(int w, FILE　*stream);

【例 3.22】

```
//向流中写入一个整数，然后再取出来
#include <stdio.h>
#include <stdlib.h>
#define　MYNAME "myfile.txt"
int main(　void)
 {
FILE *fp;
int　number;
 / * place the number　in a　file */
fp =fopen(MYNAME, "wb");
if　(fp==NULL)
 {
 printf("Error opening　file　　%s\n", MYNAME);
exit(1);
 }
number=50;
putw(number,fp);
if　(ferror(fp))
printf("Error writing to file\n");
else
printf("Successful　write\n");
fclose(fp);
/ * reopen the　file */
fp =fopen(MYNAME, "rb");
if (fp == NULL)
{
printf("Error opening file　%s\n", MYNAME);
```

```
exit(1);
}

/ * extract  the  number  */
number=getw(fp);
if (ferror(fp))
printf("Error  reading  file\n");
else
printf("Successful  read: number = %d\n", number);
/ *clean  up  */
fclose(fp);
unlink(MYNAME);
return  0;
}
```

2．fgetc()函数和fputc()函数

1）fputc()函数

功能：将字符数据输出到“文件指针”所指向的文件中，同时将读/写位置指针向前移动1 个字节（即指向下一个写入位置）。如果输出成功，则函数返回值就是输出的字符数据；否则，返回一个符号常量 EOF（其值在头文件 stdio.h 中，被定义为-1）。

用法：int fputc(字符数据,文件指针);

2）fgetc()函数

功能：从“文件指针”所指向的文件中，读入一个字符，同时将读写位置指针向前移动1 个字节（即指向下一个字符）。该函数无出错返回值。

用法：int fgetc(文件指针);

例如，fgetc(fp)表达式，从文件 fp 中读一个字符，同时将 fp 的读/写位置指针向前移动到下一个字符。

【例 3.23】

```
//将键盘上输入的一个字符串（以#作为结束字符），以 ASCII 码形式存储到一个磁盘文件中
#include "stdio.h"
main(int strc, char *strv[])
    { FILE *fp;
      char ch;
      if(strc!=2)
          {  printf("参数个数不对\n\n");
             printf("用法: 可执行文件名  filename \n");
       exit(0);
          }
if ((fp=fopen(strv[1],"w"))==NULL)
       {  printf("打开文件失败\n");
          exit(0);
       }
  for( ; (ch=getchar()) != '#' ; )
       fputc(ch,fp);
  fclose(fp);
```

【例 3.24】

```
//把一个已存在磁盘上的 file_1.dat 文本文件中的内容，原样输出到终端屏幕上
#include"stdio.h"
   void main()
{
FILE *fpin;
char ch;
if((fpin=fopen("file_1.dat","r"))==NULL)
{ printf("Cann't open this file!\n");exit(0);}
ch=fgetc(fpin);
while (ch!=EOF)
{ putchar(ch); ch=fgetc(fpin);}
fclose(fpin);
}
```

3．判断文件结束函数feof

EOF 可以作为文本文件的结束标志，但不能作为二进制文件的结束符。feof 函数既可以判断二进制文件，又可以判断文本文件。

【例 3.25】

编写程序，用于把一个文本文件（源）复制到另一个文件（目的）中，源文件名和目的文件名由命令行输入，命令形式如下：

```
可执行程序名 源文件名 目的文件名
#include<stdio.h>
   void filecopy(FILE *,FILE *);
   void main(int argc,char *argv[])
{
FILE *fpin,*fpout;
if(argc==3)
{fpin=fopen(argv[1],"r");
fpout=fopen(argv[2],"w");
filecopy(fpin,fpout);
fclose(fpin);fclose(fpout);
}
else if(argc>3)
printf("The file names too many!!\n";
else
printf("There are no file names for input or output!!\n );
}
   void filecopy(FILE *fpin,FILE *fpout)
{
char ch;
ch=getc(fpin);
```

```
while(!feof(fpin))
{putc(ch,fpout); ch=getc(fpin);}
}
```

4．fscanf函数和fprintf函数

1）fscanf函数

fscanf 函数只能从文本文件中按格式输入，与 scanf 函数相似，只不过输入的对象是磁盘上文本文件中的数据，其调用形式为：

```
fscanf(文件指针,格式控制字符串,输入项表)
```

例如：fscanf(fp,"%d%d",&a,&b);

fscanf(stdin,"%d%d",&a,&b);

等价于 scanf("%d%d",&a,&b);

2）fprintf函数

fprintf 函数按格式将内存中的数据转换成对应的字符，并以 ASCII 代码形式输出到文本文件中。fprintf 函数和 printf 函数相似，只是将输出的内容按格式存放到磁盘的文本文件中。调用形式如下：

```
fprintf(文件指针,格式控制字符串,输出项表)
```

例如：fprintf(fp,"%d %d",x,y);

5．fgets函数和fputs函数

1）fgets函数

fgets 函数用来从文件中读入字符串，调用形式如下：

```
fgets(str,n,fp);
```

函数功能是：从 fp 所指文件中读入 $n-1$ 个字符放入 str 为起始地址的空间内；如果在未读满 $n-1$ 个字符时，遇到换行符或一个 EOF 则结束本次读操作，并把 str 作为函数值返回。

2）fputs函数

fputs 函数把字符串输出到文件中。函数调用形式如下：

```
fputs(str,fp);
```

注意：为了便于读入，在输出字符串时，应当人为地加诸如"\n"这样的字符串。

【例 3.26】

```
//将键盘上输入的一个长度不超过 20 的字符串，以 ASCII 码形式存储到一个磁盘文件中；然后再输出到屏幕上
#include "stdio.h"
main(int strc, char *strv[])
    { FILE *fp;
      char string[21];
      if(strc>2)
        {   printf("参数太多\n\n");
            printf("用法: 可执行文件名   filename\n");
            exit(0);
```

```
                }
        if(strc= =1)
                { printf("请输入文件名: ");
                  gets(string);
                  strv[1]=(char *)malloc(strlen(string)+1);
                  strcpy(strv[1],string);
                }
          if ((fp=fopen(strv[1],"w"))==NULL)
                {   printf("打开文件失败\n");
                  exit(0);
                }
          /*从键盘上输入字符串，并存储到指定文件中*/
          printf("请输入一个字符串: "); gets(string);
          fputs(string, fp);
          fclose(fp);
        /*重新打开文件，读出其中的字符串，并输出到屏幕上*/
            if ((fp=fopen(strv[1],"r"))==NULL)
                  {   printf("打开文件失败\n");
                      exit(0);
                  }
            fgets(string, strlen(string)+1, fp);
            printf("输出字符串: ");
            puts(string);
            fclose(fp);
          }
```

6．文件定位函数

1）fseek函数

fseek 函数用来移动文件位置指针到指定的位置上，接着的读或写操作将从此位置开始。函数的调用形式如下：

```
fseek(fp,offset,origin)
```

fp：文件指针。

offset：以字节为单位的位移量，为长整型。

origin：是起始点，用来指定位移量是以哪个位置为基准的。

fseek 函数位移量的表示方法：标志符 数字 代表的起始点。

SEEK_SET 0：文件开始。

SEEK_END 2：文件末尾。

SEEK_CUR 1：文件当前位置。

假设 fp 已指向一个二进制文件，则

```
fseek(fp,30L,SEEK_SET)
fseek(fp,-10L*sizeof(int),SEEK_END)
```

对于文本文件，位移量必须是 0；例如：

```
fseek(fp,0L,SEEK_SET)
```

```
fseek(fp,0L,SEEK_END)
```

2）ftell函数

ftell 函数用以获得文件当前位置指针的位置，函数给出当前位置指针相对于文件开头的字节数。例如：

```
long t;
t=ftell(fp);
```

当函数调用出错时，函数返回-1L。

可以通过以下方式来测试一个文件的长度：

```
fseek(fp,0L,SEEK_END);
t=ftell(fp);
```

3）rewind函数

调用形式为：rewind(fp);

函数没有返回值。函数的功能是使文件的位置指针回到文件的开头。

【例 3.27】

有一个文件 file.txt，下列程序实现第一次使它显示在屏幕上，第二次把它复制到另一个文件 newfile.txt 上。

```
#include<stdio.h>
main()
{
FILE *fp1,*fp2;
   fp1=fopen ("file.txt","r");
   fp2=fopen ("newfile.txt","w");
   while (!feof(fp1));
         putchar (fgetc (fp1));
   rewind (fp1);
   while (!feof (fp1));
         fputc(fgetc (fp1),fp2);
   fclose (fp1);
   fclose (fp2);
}
```

即时训练

（1）将某一指定磁盘文件的内容复制到另一个指定的磁盘文件中。

（2）从键盘上输入两行字符，并存入指定的磁盘文件中。

（3）从磁盘文件中读取一个字符串，并显示在屏幕上。

（4）统计文件中的单词个数。

（5）输入 5 个学生的语、数、外三门课的成绩，统计各科的平均分后，将所有的数据存入文件 STU.DAT 中。用 fscanf 和 fprintf 函数实现题目的要求，并实现以下功能：输入一个学生的学号，即可显示该学生的各科成绩及平均分。

拓展任务

完善“商品维护”模块的功能，添加商品时能实现商品自动编号。

3.4.6 子任务六：会员管理——会员查询模块的功能实现

任务描述

会员查询模块主要实现会员查询功能。会员查询包括按编号查询、按姓名查询、按身份证查询三种查询方式。

任务分析与设计

（1）基本流程如表 3.40 所示。

表 3.40 会员查询模块的基本流程

1. 定义会员查询函数	2. 呈现界面，如图 3.21 所示
3. 键盘输入你的选择（0-4）	4. 如果选择“1”，则按会员编号查询
5. 键盘输入要查询会员的编号	6. 显示查询结果，如图 3.22 所示
7. 如果选择“2”，则按身份证号查询	8. 键盘输入要查询的会员身份证号
9. 显示查询结果，如图 3.22 所示	10. 如果选择“3”，则按会员姓名查询
11. 键盘输入要查询的会员姓名	12. 显示查询结果，如图 3.22 所示
13. 如果选择“0”，则返回会员管理界面	

图 3.21 会员查询界面

图 3.22 会员查询结果

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义会员结构体（member）变量。

S1.2：定义指针变量 fp。

S2：打开会员文件（member）。

S3：按编号进行查询。

S3.1：从文件中逐条读取信息。

S3.2：将文件中的会员编号与用户输入的会员编号进行比较。

S3.2.1：如果相同，显示会员查询结果，包括会员编号、会员姓名和身份证号。按任意键返回会员查询界面。

S3.2.2：如果不同，显示无此会员编号，按任意键继续查询。

S4：返回会员查询界面。

说明：此算法为按编号查询的算法，与其他两种查询算法相似，由学生自己完成。

（3）设计流程图。

① 会员查询界面流程图如图 3.23 所示。

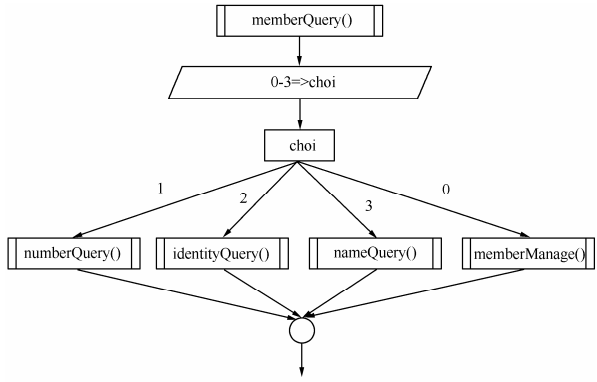


图 3.23　会员查询界面流程图

② 按会员编号查询流程图如图 3.24 所示。

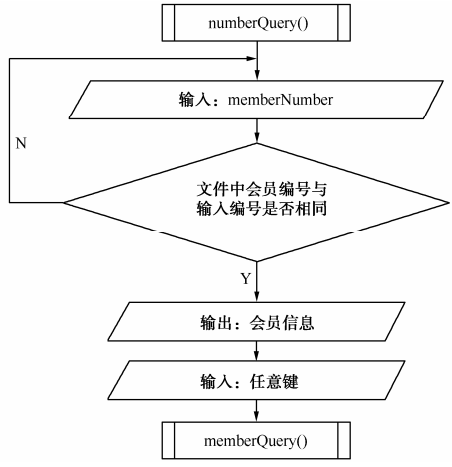


图 3.24　按会员编号查询流程图

③ 按会员姓名查询流程图如图 3.25 所示。

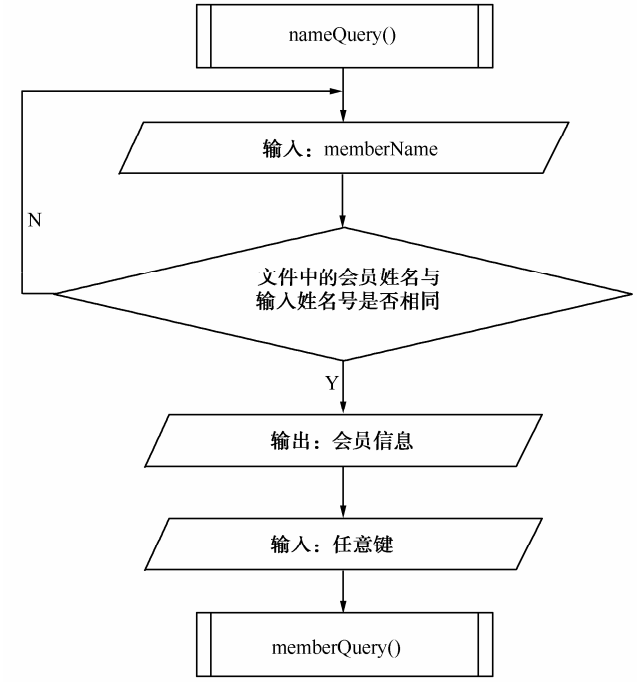


图 3.25　按会员姓名查询流程图

④ 按会员身份证号查询流程图如图 3.26 所示。

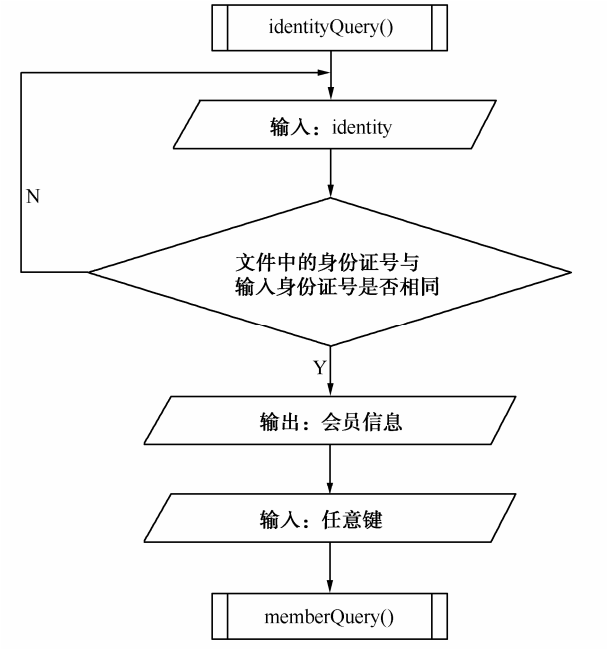


图 3.26　按会员身份证号查询流程图

实现任务

步骤一：定义按会员编号查询、按会员姓名查询、按会员身份证号查询函数 numberQuery()、nameQuery()、identityQuery()。

声明按编号查询、按姓名查询和按身份证号查询函数 numberQuery()、nameQuery()和identityQuery()。

调用按会员编号查询、按会员姓名查询和按会员身份证号查询函数，在会员查询界面中进行调用。

步骤二：编码实现。

```
//按会员编号查询
int numberQuery(int number)
{
    struct member hy;
    FILE *fp;
    if((fp=fopen("data\\member.dat","rb"))==NULL)
    {
        printf("cannot open file\n");
        return 0;
    }
    while(!feof(fp))
    {
        fread(&hy,sizeof(struct member),1,fp);
        if(hy.memberNumber==number)
        {

                system("cls");
                printf("\n\n\n\n\n\t\t\t 查询结果如下：\n\n");
                printf("\t\t 会员编号：\t%d\n\n",hy.memberNumber);
                printf("\t\t 会员名称：\t%s\n\n",hy.memberName);
                printf("\t\t 身份证号：\t%s\n\n",hy.identity);
                printf("\n\n\n 按任意键返回会员查询界面");
                getchar();getchar();
                fclose(fp);
                return 1;

        }
}

        system("cls");
        printf("\n\n 无此会员编号");
        printf("\n\n\n 按任意键继续查询");
        getchar();getchar();
        fclose(fp);
        return 0;
}
```

```
//按会员姓名查询
  void nameQuery()
{
    struct member hy;
    int found=0;//是否找到，0：未找到，1：找到
    char memberName[20];
    FILE *fp;
    if((fp=fopen("data\\member.dat","rb"))==NULL)
    {
        printf("cannot open file\n");
        return;
    }
    printf("请输入会员姓名：");
    scanf("%s",&memberName);
    while(!feof(fp))
    {
        fread(&hy,sizeof(struct member),1,fp);
        if(strcmp(hy.memberName,memberName)==0)
        {
            system("cls");
            found=1;
            printf("\n\n\n\t\t\t 查询结果如下：\n\n");
            printf("\t\t 会员编号：\t%d\n\n",hy.memberNumber);
            printf("\t\t 会员名称：\t%s\n\n",hy.memberName);
            printf("\t\t 身份证号：\t%s\n\n",hy.identity);
            printf("\n\n\n 按任意键返回会员查询界面");
            getchar();getchar();
            break;
        }
    }
    if(found==0)
    {
        system("cls");
        printf("\n\n 无此会员");
        printf("\n\n\n 按任意键继续查询");
        getchar();getchar();
    }
    fclose(fp);
}
//按会员身份证号查询
  void identityQuery()
{
    struct member hy;
    //found 是否找到，0：未找到，1：找到
    int found=0;
    char identity[19];
    FILE *fp;
```

```
        if((fp=fopen("data\\member.dat","rb"))==NULL)
        {
            printf("cannot open file\n");
            return;
        }
        printf("请输入会员身份证号：");
        scanf("%s",&identity);
        while(!feof(fp))
        {
            fread(&hy,sizeof(struct member),1,fp);
            if(strcmp(hy.identity,identity)==0)
            {
                system("cls");
                found=1;
                printf("\n\n\n\t\t\t 查询结果如下：\n\n");
                printf("\t\t 会员编号：\t%d\n",hy.memberNumber);
                printf("\t\t 会员名称：\t%s\n",hy.memberName);
                printf("\t\t 身份证号：\t%s\n",hy.identity);
                printf("\n\n\n 按任意键返回会员查询界面");
                getchar();getchar();
                break;
            }
        }
        if(found==0)
        {
            system("cls");
            printf("\n\n 无此会员身份证号");
            printf("\n\n\n 按任意键继续查询");
            getchar();getchar();
        }
        fclose(fp);
    }
```

1．C语言中的结构化程序设计

任何复杂的算法，都可以由顺序结构、选择（分支）结构和循环结构三种基本结构组成。在构造算法时，也仅以这三种结构作为基本单元，同时规定基本结构之间可以并列和互相包含，不允许交叉和从一个结构直接转到另一个结构的内部中。结构清晰、易于正确性验证和纠正程序中的错误的方法就是结构化方法，遵循这种方法的程序设计，就是结构化程序设计。遵循这种结构的程序只有一个输入口和一个输出口。

结构化程序的概念首先是从以往编程过程中无限制地使用转移语句而提出的。转移语句可以使程序的控制流程强制性地转向程序的任一处，在传统流程图中，用“很随意”的流程来描述转移功能。如果一个程序中多处出现这种转移情况，将会导致程序流程无序可寻，程序结构杂乱无章，这样的程序是令人难以理解和接受的，并且容易出错。尤其是在实际软件

产品的开发中，更多地追求软件的可读性和可修改性，像这种结构和风格的程序是不允许出现的。为此提出了程序的三种基本结构。

在讨论算法时，我们列举了程序的顺序、选择和循环三种控制流程，这就是结构化程序设计方法强调使用的三种基本结构。算法的实现过程是由一系列操作组成的，这些操作之间的执行次序就是程序的控制结构。1996 年，计算机科学家 Bohm 和 Jacopini 证明了这样的事实：任何简单或复杂的算法都可以由顺序结构、选择结构和循环结构这三种基本结构组合而成。因此，这三种结构就被称为程序设计的三种基本结构，也是结构化程序设计必须采用的结构。

结构化程序中的任意基本结构都具有唯一入口和唯一出口，并且程序不会出现死循环。在程序的静态形式与动态执行流程之间具有良好的对应关系。

结构化程序设计要求在设计程序时采用“分而治之，各个击破”的策略，用不同的函数实现功能相对单一的功能模块，这样不仅可以减少重复编程，而且还可以提高程序的可读性、可维护性和可扩充性。

2．函数的概述

前面已经介绍了函数的应用，下面对函数的概念进行阐述。函数就是能够实现特定功能的程序模块，函数是 C 程序的基本功能单位，通过对函数的调用实现特定的功能。在 C 程序设计中，要使得程序结构化，就应当将一些具有特定功能的程序段编制成函数，由主函数来调用。这样不仅能使程序简洁、便于调试、可读性强，而且还可以减少代码的重复（因为一个函数可以被调用多次）。

C 语言不仅提供了极为丰富的库函数（如 Turbo C、MS C 都提供了 300 多个库函数），还允许用户建立自己定义的函数。用户可把自己的算法编成一个个相对独立的函数模块，然后用调用的方法来使用函数。因为 C 程序的全部工作都是由各式各样的函数完成的，所以也把 C 语言称为函数式语言。

由于采用了函数模块式的结构，C 语言易于实现结构化程序设计，使程序的层次结构清晰，便于程序的编写、阅读、调试。

C 语言的程序结构如图 3.27 所示。

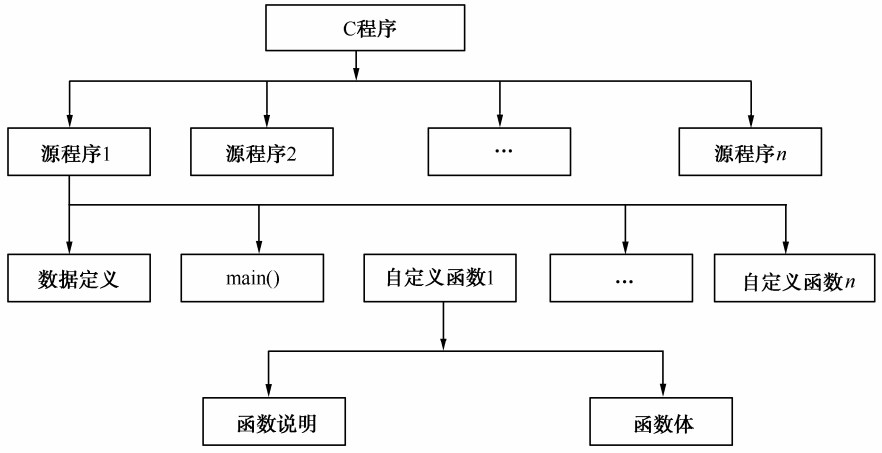


图 3.27　C 语言的程序结构

在 C 语言中，可从不同的角度对函数分类。

从函数定义的角度看，函数可分为库函数和用户定义函数两种。

（1）库函数：由 C 系统提供，用户无须定义，也不必在程序中做类型说明，只需在程序前包含该函数原型的头文件即可在程序中直接调用。在前面各章的例题中反复用到的 printf、scanf、getchar、putchar、gets、puts、strcat 等函数均属此类。

（2）用户定义函数：由用户按需要写的函数。对于用户自定义函数，不仅要在程序中定义函数本身，而且在主调函数模块中还必须对该被调函数进行类型说明，然后才能使用。

C 语言的函数兼有其他语言中的函数和过程两种功能，从这个角度看，又可把函数分为有返回值函数和无返回值函数两种。

（1）有返回值函数：此类函数被调用执行完后将向调用者返回一个执行结果，称为函数返回值，如数学函数就属于此类函数。由用户定义的这种要返回函数值的函数，必须在函数定义和函数说明中明确返回值的类型。

（2）无返回值函数：此类函数用于完成某项特定的处理任务，执行完成后不向调用者返回函数值。这类函数类似于其他语言的过程。由于函数无须返回值，用户在定义此类函数时可指定它的返回为“空类型”， 空类型的说明符为“void”。

从主调函数和被调函数之间数据传送的角度看，又可分为无参函数和有参函数两种。

（1）无参函数：函数定义、函数说明及函数调用中均不带参数。主调函数和被调函数之间不进行参数传送。此类函数通常用来完成一组指定的功能，可以返回或不返回函数值。

（2）有参函数：也称为带参函数。在函数定义及函数说明时都有参数，称为形式参数（简称为形参）。在函数调用时也必须给出参数，称为实际参数（简称为实参）。进行函数调用时，主调函数将把实参的值传送给形参，供被调函数使用。

3．函数的定义

函数的定义确立了函数的功能，即该函数能做什么；函数功能的实现必须通过函数调用，即函数调用是实现函数功能的唯一手段。函数的定义只需一次，而函数的调用可以有多次，这就是使用函数可减少程序代码长度、使程序结构清晰的原因之一。

1）无参函数的定义

（1）一般形式：

```
类型标志符  函数名()
    {声明部分
    语句
}
```

（2）说明：

类型标志符和函数名称为函数头。

类型标志符指明了本函数的类型，函数的类型实际上是函数返回值的类型。

函数名是由用户定义的标志符，函数名后有一个空括号，其中无参数，但括号不可少。

{}中的内容称为函数体。在函数体中声明部分，是对函数体内部所用到的变量的类型说明。

在很多情况下都不要求无参函数有返回值，此时函数类型符可以写为 void。

可以改写一个函数定义：

```
void Hello()
  {
     printf ("Hello,world \n");
```

```
}
```

这里，只把 main 改为 Hello，作为函数名，其余不变。Hello 函数是一个无参函数，当被其他函数调用时，输出 Hello world 字符串。

2）有参函数的定义

（1）一般形式：

```
类型标志符 函数名(形式参数表列)
    {声明部分
    语句
}
```

（2）说明：

有参函数比无参函数多了一个内容，即形式参数表列。在形参表中给出的参数称为形式参数，它们可以是各种类型的变量，各参数之间用逗号间隔。

在进行函数调用时，主调函数将赋予这些形式参数实际的值。形参既然是变量，就必须在形参表中给出形参的类型说明。

例如，定义一个函数，用于求两个数中的大数，可写为：

```
int max(int x, int y)
{
  if (x>y)
     return x;
 else
return y;
}
```

第一行说明 max 函数是一个整型函数，其返回的函数值是一个整数。形参为 x,y，均为整型量。x,y 的具体值是由主调函数在调用时传送过来的。在{}中的函数体内，除形参外，没有使用其他变量，因此只有语句，而没有声明部分。在 max 函数体中的 return 语句是把 x（或 y）的值作为函数的值返回给主调函数。在有返回值函数中，应至少有一个 return 语句。

在 C 程序中，一个函数的定义可以放在任意位置，既可放在主函数 main 之前，也可放在 main 之后。

4．函数的参数和返回值

1）函数的参数

函数的参数分为形参和实参两种，放在主调函数中的参数称为实参，放在被调函数中的参数称为形参。形参和实参的功能是实现参数的传递。发生函数调用时，主调函数把实参的值传送给被调函数的形参，从而实现主调函数向被调函数的数据传送。

说明：

（1）在调用函数时，函数的实参和形参的个数、类型和次序要一一对应。

（2）对于非 void 类型的函数，函数的调用只能通过 return 语句得到一个返回值；对于 void 类型的函数，调用函数不能返回值。

（3）形参在函数未被调用时，系统并不给它们分配存储单元，只有在发生函数调用时形参才被分配临时存储单元，但在调用结束后，系统自动释放形参占用的存储单元。

（4）实参可以是常量、变量或表达式，但必须有确定的值，以便传递给形参。

（5）形参一定是变量，也可以是指针类型，如数组名（它代表数组的首地址）和指针变量。

（6）值传递。如果实参是一个非指针类型的变量（包括数组名），此时的形参和实参的数值传递是“单向的值传递”，即实参仅把它的值传递给对应的形参，形参的值即使在函数中发生了改变，形参的值也不能传递给实参。

2）函数的返回值

函数的值是指函数被调用之后，执行函数体中的程序段所取得的并返回给主调函数的值。

说明：

（1）函数的值只能通过 return 语句返回主调函数。

return 语句的一般形式为：

```
return 表达式;
```

或者为：

```
return (表达式);
```

（2）该语句的功能是计算表达式的值，并返回给主调函数。在函数中，允许有多个 return 语句，但每次调用只能有一个 return 语句被执行，因此只能返回一个函数值。

（3）函数值的类型和函数定义中的函数类型应保持一致。如果两者不一致，则以函数类型为准，自动进行类型转换。

（4）如果函数值为整型，则在函数定义时可以省去类型说明。

（5）不返回函数值的函数，可以明确定义为“空类型”，类型说明符为“void”。

例如，超市管理系统中的会员管理界面函数如下：

```
void memberManage()
{ ……
    }
```

（6）一旦函数被定义为空类型后，就不能在主调函数中使用被调函数的函数值了。

（7）为了使程序有良好的可读性并减少出错，凡不要求返回值的函数都应定义为空类型。

5．函数的声明（函数原型）

在主调函数中调用某函数之前应对该被调函数进行说明（声明），这与使用变量之前要先进行变量说明是一样的。在主调函数中对被调函数进行说明的目的是，使编译系统知道被调函数返回值的类型，以便在主调函数中按此种类型对返回值进行相应的处理。

函数声明担负着下列三个特殊的任务：

（1）确定函数返回值的类型，使编译程序能产生函数返回数据类型的正确代码。

（2）确定了函数使用的参数的类型、个数和顺序。

（3）依据函数完成的功能确定该函数的函数名。

函数声明有下列两种格式。

（1）无参声明格式：

```
函数类型 函数名();
```

（2）有参声明格式：

```
函数类型 函数名(数据类型 形式参数 数据类型 形式参数, .....);
```

说明：函数类型是该函数返回值的数据类型，可以是以前介绍的整型（int）、长整型（long）、字符型（char）、单浮点型（float）、双浮点型（double）以及无值型（void），也可以是指针，包括结构指针。无值型表示函数没有返回值。

当被调函数的返回值是整型或字符型时，可以不对被调函数进行说明，而直接调用。

当被调函数的函数定义出现在主调函数之前时，在主调函数中也可以不对被调函数再进行说明而直接调用。

若在所有函数定义之前，在函数外预先说明了各个函数的类型，则在以后的各主调函数中，可不再对被调函数进行说明。

6．函数的调用

在 C 语言中，可以用以下几种方式调用函数。

（1）函数语句：函数调用的一般形式加上分号即构成函数语句。这种方式函数不需要返回值。

例如，getchar();printf ("%d",a);scanf ("%d",&b);都是以函数语句的方式调用函数。

（2）函数表达式：函数作为表达式中的一项出现在表达式中，以函数返回值参与表达式的运算。这种方式要求函数必须有返回值。

（3）函数实参：函数作为另一个函数调用的实际参数出现。这种情况是把该函数的返回值作为实参进行传送，因此要求该函数必须有返回值。

【例 3.28】

```
//函数定义
int max(int x,int y)
{
    if(x>y)
       return x;
    else
       return y;
}
main()
{
//函数声明（函数原型）
    int max(int x,int y);
    int a,b,c;
    printf("请输入两个整数:\n");
    scanf("%d%d",&a,&b);
//函数调用
    c=max(a,b);
    printf("最大值=%d",c);
}
```

下面可以从函数定义、函数说明及函数调用的角度来分析整个程序，从中进一步了解函数的各种特点。

程序执行过程：用户输入两个整数，调用 max 函数，将实参 a,b 的值送给形参 x,y，max 函数执行的结果返回给 c。

【例 3.29】

```
//函数调用的另一种方式
int f(int a,int b,int c)
      {
         int z;
         z=a+b*c;
         return z;
      }
main()
    {
       int x=2,y;
       y=f(x++,x++,x++);
       printf("%d\n",y);
}
```

7．函数的嵌套调用

在 C 语言中，不允许做嵌套的函数定义。因此，各函数之间是平行的，不存在上一级函数和下一级函数的问题。但是，C 语言允许在一个函数的定义中出现对另一个函数的调用。这样，就出现了函数的嵌套调用，即在被调函数中又调用其他函数。

例如，在超市管理系统中，会员管理函数可以调用会员添加、会员删除、会员查询和会员统计等函数，同时会员查询还可以调用按会员编号查询函数、按会员姓名查询函数、按会员身份证号查询函数。会员管理模块的函数调用关系如图 3.28 所示。

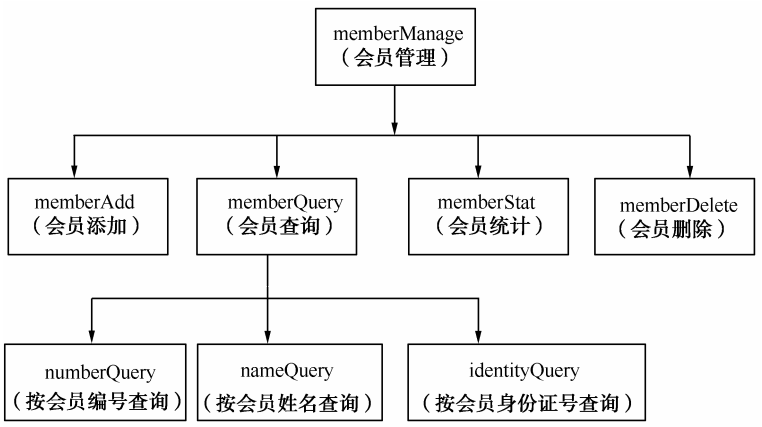


图 3.28　会员管理模块的函数调用关系

8．函数的递归调用

函数递归是指一个函数直接或者间接调用自己的现象。C 语言允许函数的递归调用。在递归调用中，主调函数又是被调函数。执行递归函数将反复调用其自身，每调用一次就进入新的一层。有时，递归是非常有用的一种调用方式，因为它可以减少代码量。不过，递归代码一般不容易理解，并且执行时对资源消耗较大。

递归调用分为两种调用方式：一种是直接递归，另一种是间接递归。

直接递归：定义一个函数时，在函数定义内部出现了对本函数的调用。

间接递归：在定义两个函数时，出现了相互调用。

【例 3.30】　求 n 的阶乘 $n!$。

算法分析：阶乘问题可以用循环语句轻松解决。但实际上，在数学上阶乘是应用递归形式定义的，n 的阶乘定义为：

$$f(n)=\begin{cases}1 & (n=01)\\ nf(n-1) & (n>1)\end{cases}$$

当 $n>1$ 时，n 的阶乘为 n 乘以 $n-1$ 的阶乘。因此，可以用函数的递归调用来求解它。

```
#include <stdio.h>
long f(int n)
{
    if(n==1)
        return 1;
    else
        return n*f(n-1);
}
main()
{
    int n;
    long fac;
    printf("请输入一个正整数：\n");
    scanf("%d",&n);
    fac=f(n);
    printf("f(%d)=%d",n,fac);
    getchar();
    return 0;
}
```

【例 3.31】　求 fibonacci 数列的递归函数。

```
fib(int n)
{
  long f;
  if(n<1) printf("n<1，dataerror!")
  else if(n==1||n=2)f＝1;
  else f＝fib(n-1)+fib(n-2);
  return(f);
  }
main()
{
int n,s;
  printf("input an integer number");
```

```
    scanf("%d"，&n)；
    s=fib(n)；
    printf("fib(%d)=%ld\n",n,s)
    }
```

运行情况如下：

```
    input an integer number:10
    fib(10)= 55
```

对递归算法的进一步说明：

（1）递归算法是解决复杂问题的有力工具，递归是有条件的，即必须有递归原则和停止准则。

（2）递归算法在实现时要用到堆栈，算法的时间和空间效率较差，但算法的结构清晰，符合结构化程序设计的要求。

（3）有些问题既可以用递归方法解决，也可以用非递归方法解决。例如，阶乘问题、斐波纳奇数列问题和表达式求值问题都可以用非递归方法解决。

即时训练

（1）编写函数，求出 1～*N* 的素质之和。

（2）输出 1～5 的阶乘值。

（3）编写程序，利用函数调用的方式，输入 10 名学生的成绩，并求出他们的平均成绩。

（4）编写求 1+2+…+*N* 的函数，并在主程序中输入 *N* 的值后调用该函数。

拓展任务

实现“超市管理系统”的“用户管理模块—查询用户”的功能。

3.4.7 子任务七：会员管理——会员统计模块的功能实现

任务描述

会员统计模块主要实现会员排序的功能。会员统计包括按会员编号降序排序和按会员编号升序排序两种排序方式，主要目的是实现对会员信息的统计管理，根据所列会员的信息，让管理者一目了然，对会员情况了如指掌。

任务分析与设计

（1）会员统计模块的基本流程如表 3.41 所示。

表 3.41 会员统计模块的基本流程

1．定义会员统计函数	2．呈现界面，如图 3.14 所示
3．键盘输入你的选择（0-2）	4．如果选择“1”，则按会员编号降序排序，排序结果如图 3.29 所示
5．如果选择“2”，则按会员编号升序排	6．如果选择“0”，则返回会员管理界面

序，排序结果如图 3.30 所示	

图 3.29　按会员编号降序排序

图 3.30　按会员编号升序排序

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义会员结构体（member）变量 hy，用于存放从文件中读取出来的信息。

S1.2：定义会员结构体（member）数组，长度为 200，用于存放所有从会员文件中读取出来的信息。

S1.3：定义指针变量 fp 和 countNum、k、j 三个变量并初始化，分别用来移动文件记录指针和排列会员编号。

S2：打开会员文件（member）。

S3：按编号进行查询。

S3.1：从文件中逐条读取信息。

S3.1.1：从文件中读取第一条信息，存入 hy 变量中。

S3.1.2：将 hy 中的信息存入 memberInfo(0)中。

S3.1.3：再次从文件中读取数据存入 hy 中。

S3.1.4：将 hy 中的信息存入 memberInfo(1)中。

S3.1.5：以此类推，读取会员文件中的所有信息，存入结构体数组 memberInfo 中。

S3.2：对所读取的数据进行比较排序。

S3.2.1：按降序排序。

S3.2.2：按升序排序。

S4：显示排序列表。

任务实现

步骤一：声明会员统计函数 int memberStat(int i)。

步骤二：编码实现。

```
int memberStat(int i)
{
    struct member hy;
    //定义会员信息为结构体数组
    struct member memberInfo[200];
   int countNum = 0;
    int k,j;
    FILE *fp;
    if((fp=fopen("data\\member.dat","rb"))==NULL)
    {
        printf("cannot open file\n");
        return 0;
    }

    if(!feof(fp))
    {
        fread(&hy,sizeof(struct member),1,fp);
    }
    while(!feof(fp)){
        memberInfo[countNum] = hy;
        countNum++;
        fread(&hy,sizeof(struct member),1,fp);
    }
    if( i == 1)
    {
      for (j=0;j<countNum;j++)
        {
          for(k=j+1;k<countNum;k++)
            {
              if(memberInfo[j].memberNumber < memberInfo[k].memberNumber)
                {
```

```
                    struct member temp;
                    temp = memberInfo[j];
                    memberInfo[j] = memberInfo[k];
                    memberInfo[k] = temp;
                }
            }
        }
    }
    else
    {
        for (j=0;j<countNum;j++)
        {
            for(k=j+1;k<countNum;k++)
            {
                if(memberInfo[j].memberNumber > memberInfo[k].memberNumber)
                {
                    struct member temp;
                    temp = memberInfo[j];
                    memberInfo[j] = memberInfo[k];
                    memberInfo[k] = temp;
                }
            }
        }

    }
        system("cls");
        printf("\n\n");
        printf("\t\t*************************************************\n");
        printf("\t\t                  会员列表                     \n");
        printf("\t\t*************************************************\n");

        printf("\n\n 会员总数为:%d",countNum);
        printf("\n\n");
        printf("\t\t 会员编号");
        printf("\t 会员名称");
        printf("\t 身份证号");
        printf("\t\n\n");
      for(j=0;j<countNum;j++)
        {
            printf("\t\t%d",memberInfo[j].memberNumber);
            printf("\t\t%s",memberInfo[j].memberName);
            printf("\t\t%s",memberInfo[j].identity);
            printf("\t\n\n");
        }
        printf("\n\n\n 按任意键返回");
        getchar();getchar();
        fclose(fp); return 1;
```

```
}
```

1．常用排序法

1）选择排序算法

每次从数组中选出一个最大的数放在已排好序的前面。第一次循环时，从第二个元素开始找比第一个元素大的元素，然后两者交换（第一次循环即找到第一大的数）；第二次循环从第三个元素找比第二个元素大的元素，两者交换（第二次循环即找到第二大的数）；如果有 N 个数，循环 $N-1$ 次即可。

【例 3.32】

```
// 输入 N 个数后，用选择排序法把数组元素从大到小排序
# include <stdio.h>
#define N 12
main()
{
        int i,j,a[n],temp;
        for (i=0;i<N; i++)
              {
              scanf("%d",&a[i]);
              }
        for (i=0;i<N-1; i++)
        {
              for (j=i+1;j<N; j++)
              {
                    if (a[i]<a[j])
                    {
                          temp=a[i];
                          a[i]=a[j];
                          a[j]=temp;
                    }
              }
        }
        for (i=0;i<N; i++)
              {
              printf("%d",&a[i]);
              }
}
```

2）冒泡排序法算法

冒泡排序算法是：比较相邻的两个元素，如果后面的比前面的小，就对调二者。反复比较，到最后两个元素。排序的结果是最大值位于最末位置。

【例 3.33】

```
//输入 N 个数后，用选择排序法把数组元素从小到大排序
# include <stdio.h>
```

```
#define N 12
main()
{
int i,j,a[n],temp;
    int i,j,a[n],temp;
    for (i=0;i<N; i++)
        {
        scanf("%d",&a[i]);
        }
    for (i=0;i<N-1; i++)
    {
        for (j=0;j<N-i-1; j++)
        {
            if (a[j]<a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
    for (i=0;i<N; i++)
        {
        printf("%d",&a[i]);
        }
        }
```

2．结构体数组

与一般类型的变量一样，结构体类型的变量也可以组织成数组，称为结构体数组。不同之处是，以前介绍的数组中，数组元素是一个基本类型的变量，而结构体数组中的每个数组元素都是用户自定义的结构体类型变量。

1）结构体数组的定义

结构体数组的定义与一般类型的数组定义类似，只需说明它为数组类型即可。

有下列两种定义形式。

（1）第一种形式。例如：

```
struct stu student [10];
```

（2）第二种形式。例如：

```
struct stu
{
    int num;
    int number;
    char name[20];
    char sex;
    float score;
```

```
}student[10];
```

表示定义了一个 struct stu 结构体类型的结构体数组 student，其中有 10 个元素，在 student[0]～student[9]中，每个元素都是一个 struct stu 类型的结构体变量。

2）结构体数组的初始化

对结构体数组也可以进行初始化，例如：

```
Struct stu
{
    char numbe[3];
    char *name;
    char sex;
    float score;
}student[5]={
{"001","李丽","男",80},
            {"002","张华","女",73.5},
            {"003","王野","男",90.5},
            {"004","陈小小","女",65},
            {"005","赵可心","女",87.5};
}
```

即在定义结构体数组的同时对结构数组中的元素进行初始化。当对全部元素做初始化赋值时，也可不给出数组长度。例如：

```
struct stu
{
    char numbe[3];
    char *name;
    char sex;
    float score;
}student[]={
{"001","李丽","男",80},
            {"002","张华","女",73.5},
            {"003","王野","男",90.5},
            {"004","陈小小","女",65},
            {"005","赵可心","女",87.5};
}
```

3）结构体数组举例

【例 3.34】

计算 5 名学生的 5 门功课成绩的平均分并输出。

```
struct stu
{
    char numbe[3];
    char *name;
    char sex;
    float score;
```

```
}student[]={
{"001","李丽","男",80},
                {"002","张华","女",73.5},
                {"003","王野","男",90.5},
                {"004","陈小小","女",65},
                {"005","赵可心","女",87.5};
}
main()
{
    int i,c=0;
    float ave,s=0;
    for(i=0;i<5;i++)
    {
      s+=student[i].score;
      if(student[i].score<60) c+=1;
    }
    printf("s=%f\n",s);
    ave=s/5;
    printf("average=%f\ncount=%d\n",ave,c);
}
```

【例 3.35】

对学生信息记录按从小到大的顺序排序。

```
#include <stdio.h>
#define MAX 5
struct stu
{
    int number;
    char name[10];
}
 main()
{
    int i,j;
    struct stu sort[MAX],min,t;
    printf("请输入 5 名学生的记录:\n");
    for (i=0;i<MAX-1;i++)
    {
        printf("请输入学号:");
        scanf("%d",&sort[i].number);
        printf("请输入姓名:");
        scanf("%s",sort[i].name);
    }
    for(i=0;j<MAX-1;j++)
    {
        min=sort[i];
        for(j=i+1;j<MAX;j++)
```

```
                    if(sort[j].number<min.number)
                    {
                        t=min;
                        min=sort[j];
                        sort[j]=t;
                    }
                sort[i]=min;
        }
    printf("排序的结果如下:\n");
    for (i=0;i<MAX;i++)
        printf("%d,%s\n",sort[i].number,sort[i].name);
    getchar();
    }
```

3．指向结构体的指针

与其他类型的变量一样，也存在指向结构体变量的指针。

1）指向结构体变量的指针

定义指向结构体类型数据的指针变量的方法与定义结构体类型的变量的方法相同，只需在变量名标志符前加上“*”即可，表示该变量是指针类型。例如：

```
struct  student  stu, *p;
```

定义了一个 struct student 类型的结构体变量 stu 和结构体类型的指针变量 p。但是，此时的 p 值为空，即无具体的值，在执行语句 p=&stu;后，p 才真正指向了结构体变量 stu，即 p 的值是 stu 的首地址。当然，也可在定义 p 时赋初值。

可以通过指向结构体变量的指针来引用结构体变量的成员，形式如下：

```
结构体变量的指针->结构体变量的成员
```

这里，“->”是指向运算符，运算优先级与“()”、“[]”和“.”的运算优先级相同，是最高优先级别。

阅读下面的例子，熟悉用指向结构体变量的指针来引用结构体变量的成员的方法。

2）指向结构体数组的指针

使用指向结构体数组的指针来访问数组，既方便了数组元素的引用，又提高了数组的利用率。

指针变量可以指向一个结构数组，这时的结构指针变量的值是整个结构数组的首地址。结构指针变量也可指向结构数组的一个元素，这时的结构指针变量的值是该结构数组元素的首地址。

如果指针变量 point 已指向某结构数组，则 point+1 指向结构数组的下一个元素，而不是当前元素的下一个成员。

另外，如果指针变量 point 已经指向一个结构变量（或结构数组），就不能再使之指向结构变量（或结构数组元素）的某一成员。

【例 3.36】

```
//使用指向结构数组的指针来访问结构数组
#include"struct.h"
```

```
/*定义并初始化一个外部结构数组 student */
struct   std_info   student[5]=
{
     {"001","李丽","男",{1980,5,20}},
     {"002","张华","女",{1980,8,15}},
     {"003","王野","男",{1980,3,10}}
     {"004","陈小小","女",{1980,4,10}}
     {"005","赵可心","女",{1980,6,20}}
};
main()
{
struct   std_info   *p_std=student;
int i=0;
printf("学号  姓名  性别  出生日期\n");
/*输出结构数组内容*/
for(;i<5;i++,p_std++)
{
     printf("%-7s%-9s%-4s",p_std->no, p_std->name, p_std->sex);
     printf("%4d-%2d-%2d\n", p_std->birthday.year,
     p_std->birthday.month,   p_std->birthday.day);
}
}
```

4．结构指针变量作为函数参数

在 ANSI C 标准中，允许用结构变量作为函数参数进行整体传送。但是，这种传送要将全部成员逐个传送，特别是成员为数组时将会使传送的时间和空间开销很大，严重地降低了程序的效率。因此，最好的办法就是使用指针，即用指针变量作为函数参数进行传送。这时，由实参传向形参的只是地址，从而减少了时间和空间的开销。

【例 3.37】

```
//编写一个专门的显示函数 display()，通过主函数调用来实现显示
#include"struct.h"
/*定义并初始化一个外部结构数组 student */
struct   std_info   student[3]=
{
     {"001","李丽","男",{1980,5,20}},
     {"002","张华","女",{1980,8,15}},
     {"003","王野","男",{1980,3,10}}
     {"004","陈小小","女",{1980,4,10}}
     {"005","赵可心","女",{1980,6,20}}
};
/*主函数 main()*/
main()
{
        void display();/*函数说明*/
     int i=0;
     printf("学号   姓名   性别   出生日期\n");
```

```
        for(;i<5; i++)
        {
            display(student + i);
            printf("\n");
        }
    }

    void  display(struct  std_info  *p_std)
    {
        printf("%-7s%-9s%-4s",p_std->no,
        p_std->name,p_std->sex);
        printf("%4d-%2d-%2d\n",p_std->birthday.year,
        p_std->birthday.month,p_std->birthday.day);
    }
```

5．动态存储分配

用变量表示长度，想对数组的大小做动态说明，这是错误的。但是，在实际的编程中，往往会发生这种情况，即所需的内存空间取决于实际输入的数据，而无法预先确定。对于这种问题，用数组的办法很难解决。为了解决上述问题，C 语言提供了一些内存管理函数，这些内存管理函数可以按需要动态地分配内存空间，也可把不再使用的空间回收待用，为有效地利用内存资源提供了手段。

常用的内存管理函数有以下三个。

1）分配内存空间函数malloc

调用形式：

```
(类型说明符*)malloc(size)
```

功能：在内存的动态存储区中分配一块长度为“size”字节的连续区域。函数的返回值为该区域的首地址。

说明：

（1）“类型说明符”表示把该区域用于何种数据类型。

（2）(类型说明符*)表示把返回值强制转换为该类型指针。

（3）“size”是一个无符号数。

例如：

```
point=(char *)malloc(50);
```

表示分配 50 个字节的内存空间，并强制转换为字符数组类型，函数的返回值为指向该字符数组的指针，把该指针赋予指针变量 point。

2）分配内存空间函数 calloc

calloc 函数也用于分配内存空间。

调用形式：

```
(类型说明符*)calloc(n,size)
```

功能：在内存动态存储区中分配 *n* 块长度为“size”字节的连续区域。函数的返回值为该区域的首地址。

说明：

（1）(类型说明符*)用于强制类型转换。

（2）calloc 函数与 malloc 函数的区别仅在于一次可以分配 *n* 块区域。

例如：

```
point=(struct stu*)calloc(5,sizeof(struct stu));
```

其中的 sizeof(struct stu)是求 stu 的结构长度。

因此，该语句的意思是，按 stu 的长度分配 5 块连续区域，强制转换为 stu 类型，并把其首地址赋予指针变量 point。

3）释放内存空间函数free

调用形式：

```
free(void*ptr);
```

功能：释放 ptr 所指向的一块内存空间，ptr 是一个任意类型的指针变量，它指向被释放区域的首地址。被释放区应是由 malloc 或 calloc 函数所分配的区域。

【例 3.38】

```
//分配一块区域，输入一个学生数据
main()
{
    struct stud
    {
       int num;
       char *name;
       char sex;
       float score;
    }   *point;
    point =(struct stud*)malloc(sizeof(struct stud));
    point ->num=1001;
    point ->name="王丹";
    point ->sex='女';
    point ->score=86.5;
    printf("学号=%d\n 姓名=%s\n", point ->num, point ->name);
    printf("性别=%c\n 成绩=%f\n", point ->sex, point ->score);
    free(point);
}
```

在本例中，定义了结构 stud，定义了 stud 类型指针变量 point。然后分配一块 stud 的内存区，并把首地址赋予 point，使 point 指向该区域。再以 point 为指向结构的指针变量对各成员赋值，并用 printf 输出各成员值。最后用 free 函数释放 point 指向的内存空间。整个程序包含了申请内存空间、使用内存空间、释放内存空间三个步骤，实现存储空间的动态分配。

即时训练

（1）用其他循环嵌套语句重新编写程序，用两种排序方法分别实现将 10 个数从大到小地排序。

（2）有一个数组，内放 10 个数，要求找出最小的数和它的下标。然后把它和数组中最前面的元素对换位置。

（3）在一个已排好序的数列（由小到大）中再插入一个数，要求仍然有序。

（4）有 5 个学生，每个学生的数据包括学号、姓名和 3 门课程的的成绩，从键盘输入每个学生的数据，计算每个学生 3 门课程的平均成绩，计算 5 个学生每门课程的平均成绩，然后要求按学生平均成绩从低到高的次序输出每个学生的各课程成绩、3 门课程的平均成绩，最后输出每门课程的平均分（采用结构体数组）。

（5）建立一个学生信息的结构体，包括学号、姓名和成绩，输入数据后，打印出成绩最高的学生的基本信息。

拓展任务

用冒泡排序法重新编写“会员统计模块”的代码。

3.4.8 子任务八：会员管理——会员删除模块的功能实现

任务描述

会员删除模块主要实现会员删除的功能。通过输入会员编号，查询会员是否存在。如果存在，则删除该会员；如果不存在，则给出相应提示。

任务分析与设计

会员删除模块涉及会员删除界面函数、会员查询函数和会员删除函数。因为会员删除界面函数不同于其他界面的操作，所以将会员删除界面的设计放在会员删除模块中。会员在删除前需要对会员进行查询，确定会员是否存在，确保删除操作的可靠性。

（1）会员删除的基本流程如表 3.42 所示。

表 3.42 会员删除的基本流程

1．定义会员删除界面函数	2．呈现界面，如图 3.31 所示
3．在会员删除界面，键盘输入你要删除的会员编号	4．输入“0”，返回会员管理界面
5．输入其他数字，在文件中查询该会员编号是否存在（Y/N），即调用会员查询函数 numberQuery()	6．如果存在，则给出提示“确定要删除该会员吗？(Y/N)”； 如果选择“Y”，则进入会员删除界面 如果选择“N”，则返回会员删除界面
7．如果不存在，则给出提示“无此会员编号，按任意键继续”	8．返回会员删除界面

图 3.31　会员删除界面

（2）自然语言描述。

① 删除会员界面自然语言描述。

S1：定义变量。

S2：删除会员。

S2.1：输入要删除的会员编号。

S2.2：判断所输入的编号，如果是 0，则调用会员管理界面函数 memberManage()。

S2.3：如果输入的会员编号非 0，则判断会员是否存在，即调用会员编号查询函数 numberQuery()。

S2.3.1：如果查询到该会员编号，则系统提示“确定要删除该会员吗？(Y/N)”

S2.3.1.1：如果选择“Y”，则调用会员删除函数 deleteMember()。

S2.3.1.2：如果选择“N”，则调用会员删除界面函数 memberDeleteMain()。

S2.3.2：如果该会员不存在，系统提示“该会员不存在，按任意键继续”。

② 会员删除自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义文件指针变量 fp，目的是操作会员文件 member.dat。

S1.2：定义会员结构体数组 hy[100]。

S1.3：定义两个整型变量 i 和 j，作为循环变量。

S1.4：定义整型变量会员编号 memberNumber，初始化数值为 0。

S2：以只读的方式打开会员文件 member.dat。

S3：循环将文件中的数据写入结构体，即判断是否到文件结束标志。

S3.1：如果没到文件结束标志，从会员文件 member.dat 中读取数据流，写入结构体数组中，同时将会员编号加 1。

S3.2：如果没有到文件结束标志，则关闭文件。

S4：将结构体数组中要删除的元素删除。

S4.1：循环判断结构体数组中的会员编号与要删除会员编号是否相同。

S4.1.1：如果相同，将该编号之后的结构体元素依次替换前一个元素，从而实现删除的目的。

S4.1.2：如果不同，判断下一个元素是否满足条件，即返回 S4.1。

S5：将数据结构的数据写入会员文件 member.dat 中。

S6：以写的方式打开会员文件 member.dat。

S7：循环将结构体数组中的数据写入会员文件 member.dat 中。

S8：关闭文件。

③ 会员编号查询补充描述。

S1：给会员编号查询函数 memberQuery()添加一个作为标志的参数，即 int flag。

S1.1：当 flag 的值等于 0 时，做普通会员编号查询。

S1.2：当 flag 的值等于 1 时，做会员删除查询。

（3）会员删除流程图如图 3.32 所示。

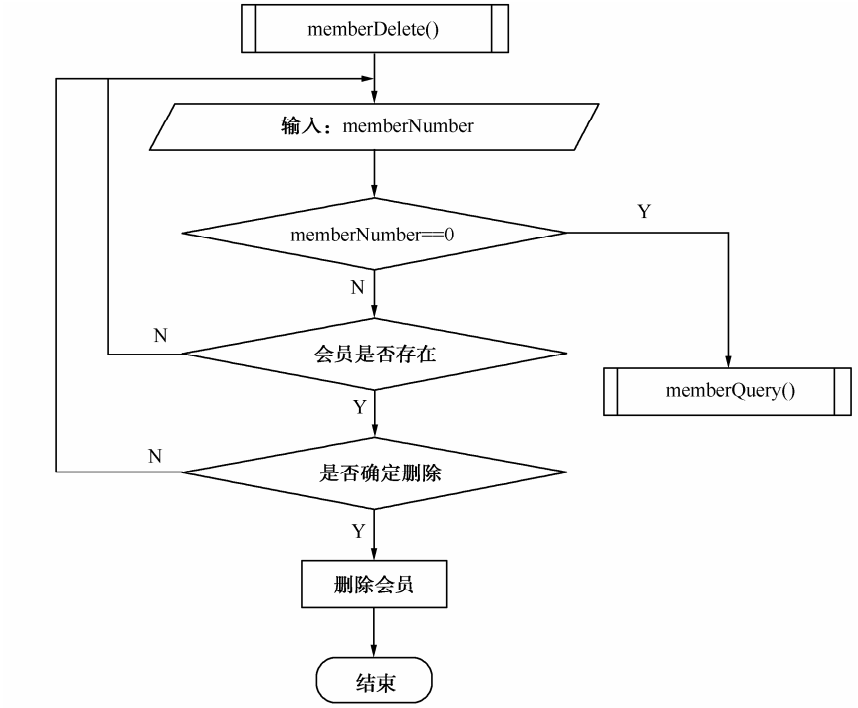


图 3.32　会员删除流程图

任务实现

步骤一：

（1）声明会员删除界面函数 void deleteMemberMenu()。

（2）定义会员删除界面函数 deleteMemberMenu()。

（3）调用会员删除界面函数，在会员管理模块调用会员删除界面函数。

步骤二：

（1）声明会员删除函数：int deleteMember(int member,int flag)。

（2）定义会员删除函数：int deleteMember(int member,int flag)。

（3）调用会员删除函数：在会员删除界面函数中调用会员删除函数。

步骤三：给会员编号查询函数添加一个标志的参数，实现多功能查询。

步骤四：编码实现。

```
//按编号查询
int numberQuery(int number,int flag)
{
    struct member hy;
    FILE *fp;
    if((fp=fopen("data\\member.dat","rb"))==NULL)
    {
        printf("cannot open file\n");
        return 0;
    }
    while(!feof(fp))
    {
        fread(&hy,sizeof(struct member),1,fp);

            if(hy.memberNumber==number)
            {
                if(flag==0)
                {
                    system("cls");
                    printf("\n\n\n\n\n\t\t\t 查询结果如下：\n\n");
                    printf("\t\t 会员编号：\t%d\n\n",hy.memberNumber);
                    printf("\t\t 会员名称：\t%s\n\n",hy.memberName);
                    printf("\t\t 身份证号：\t%s\n\n",hy.identity);
                    printf("\n\n\n 按任意键返回会员查询界面");
                    getchar();getchar();
                    fclose(fp);
                    return 1;
                }
                else
                {
                    fclose(fp);
                    return 1;
                    break;
                }

            }
}


        if(flag==0)
        {
```

```
                system("cls");
                printf("\n\n 无此会员编号!");
                printf("\n\n\n 按任意键继续");
                getchar();getchar();

            }
                fclose(fp);
                return 0;

}
//会员删除界面
  void deleteMemberMenu()
{
    char con;
    int choice;
    while(1)
    {//实现循环选择


        system("cls");//清屏
        printf("\n\n");
        printf("\t\t*****************************************************\n");
        printf("\t\t                    会员删除                         \n");
        printf("\t\t*****************************************************\n");

        printf("\n\n");
        printf("\t\t\t\t0、返回会员管理：\n\n");
        printf("\t\t 请输入你要删除的会员编号：");
        scanf("%d",&choice);
        if(choice == 0)
            {
                return;
            }
        else
        {
            if(numberQuery(choice,1)==1)

            {
                printf("\n\n\t\t 确定要删除该会员吗?（Y/N）");
                //fflush(stdin);
                scanf(" %c",&con);
                if(con=='n' || con=='N')
                {
                    break;
                }
                deleteMember(choice);
            }
```

```
                else
                {
                    printf("\n\n\t\t 该会员不存在，按任意键继续");
                    getchar();
                    getchar();

                    system("cls");
                    deleteMemberMenu();
                }
            }


        }
    }


    //会员删除
      void deleteMember(int bianhao)
    {
        FILE *fp;
        struct member hy[100];
        char con;
        int i,j;
        int memberNumber=0;
        if((fp=fopen("data\\member.dat","rb"))==NULL)
        {
            printf("cannot open file\n");
            return ;
        }
        //将信息写入结构体数组
        while (!feof(fp))
        {
            if(fread(&hy[memberNumber],sizeof(struct member),1,fp)==1)
            memberNumber++;
        }
        fclose(fp);
        //将数据结构中的要删除的元素删除
        for(i=0;i<memberNumber;i++)
        {
            if(hy[i].memberNumber==bianhao)
            {
                if (i==-1)//文件中已没数据
                {
                    printf("已无会员记录,按回车键返回会员管理界面");
                    memberManage();
                }
                for(j=i;j<=memberNumber;j++)
```

```
                {
                    hy[j]=hy[j+1];
                }
            }
        }
        //将数据结构的数据写入文件中
        if((fp=fopen("data\\member.dat","wb"))==NULL)
        {
            printf("cannot open file\n");
            return ;
        }
        for(i=0;i<memberNumber-1;i++)
        {
            if(fwrite(&hy[i],sizeof(struct member),1,fp)!=1)
                printf("write error");
        }
        fclose(fp);
    }
```

1．循环的嵌套

一个循环体内又包含另一个完整的循环结构，称为循环的嵌套。三种循环（while 循环、do while 循环和 for 循环）可以互相嵌套构成多重循环。下面几种互相嵌套都是合法的形式。

第一种：
while()
{…
while()
{…}
}
	第二种：
	do
	{…
	do
	{…}
	while();
	}
		第三种：
		for(;;)
		{…
		for(;;)
		{…}
		}

第四种：
while()
{…
do
{…}
while();
…
}
	第五种：
	for(;;)
	{…
	while()
	{…}
	}
		第六种：
		do
		{…
		for(;;)
		{…}
		…
		}
		while();

2．循环的嵌套的应用

【例 3.39】

```
//用"*"号打印 8 行 7 列的星号矩形
#include <stdio.h>
main()
 {
    int i,k;

    /*外循环控制星号矩形的行*/
    for(i=0;i<8;i++)
    {
        printf("\t");
          /*内循环打印每一行的 7 个星号*/
        for(k=0;k<7;k++)
            printf("*");

        printf("\n"); /*换行*/
    }
 }
```

【例 3.40】

```
//用"*"号打印三角形
#include <stdio.h>
main()
 {
    int i,k;

    //外循环控制星号三角形的行
    for(i=0;i<8;i++)
    {
        //内循环控制星号三角形的列
        for(k=0;k<=i;k++)
            printf("*");
        //换行
        printf("\n");
    }
 }
```

【例 3.41】

```
//求 100 以内的所有素数
#include <stdio.h>
main()
{
     int i,k,flag;
     k=2;
     while(k<=100);
```

```
        {
            flag=1;
            i=2;
            while((i<=sqrt(k))&&(flag!=0))
                {
                    If(k%i= =0)
                    flag=0;
                    i=i+1;
                }
            if(flag= =1)
            printf("%d,\n",k);
            k=k+1;
        }
        return 0;
    }
```

即时训练

（1）打印如下图形：

```
   *
  ***
 *****
  ***
   *
```

（2）输出九九表，格式如下：

1*1=1

1*2=2　2*2=4

1*3=3　2*3=6　3*3=8

…　　…　　…

1*9=9　2*9=18　3*9=27　…　9*9=81

（3）编程求 100 以内的所有“完数”。一个数如果恰好等于它的因子之和，这个数就称为“完数”。

（4）用 for 循环求 100 以内的所有素数。

（5）打印下列图形：

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

拓展任务

（1）在进行会员删除时，如果不用“将要删除的编号之后的结构体元素依次替换前一个元素”的方法进行删除，而是采用中间临时文件的方法（即将不删除的结构体元素存放到临

时文件），试重新编写该模块代码。

（2）实现“超市管理系统”的“用户管理模块—删除用户”的功能。

3.4.9 子任务九：商品销售——购物车清单的功能实现

任务描述

商品销售模块涉及商品购物车清单、动态处理商品数量和商品结算等功能。本节任务的主要目的是将顾客选购的商品编号、商品名称、商品单价和商品数量以清单的形式列出，以便在结算时使用。

任务分析与设计

（1）购物车清单的基本流程如表 3.43 所示。

表 3.43 购物车清单的基本流程

1．定义商品销售函数	2．呈现界面，如图 3.33 所示
3．输入购买商品的编号和数量	4．确认是否继续购买（Y/N）： 如果选择“Y”，则返回步骤 3； 如果选择“N”，则列出购物车清单，如图 3.34 所示

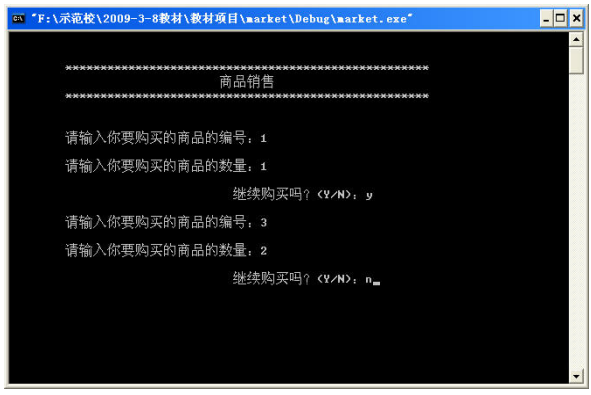


图 3.33 商品销售界面

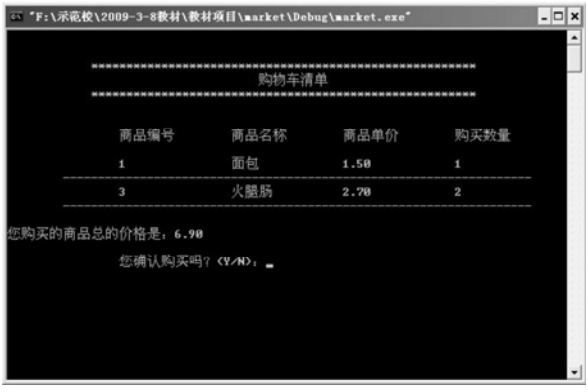


图 3.34 购物车清单界面

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义商品结构体变量 sp。

S1.2：定义选购商品 chooseGoods[100][2]二维数组。

S1.3：定义文件指针变量 fp1，用于操作商品 goods.dat 文件。

S1.4：定义商品种类变量 buyKind，并赋初始值为 0。

S1.5：定义浮点型变量：应付金额变量 dueMoney、实付金额变量 payMoney 和找零变量 change，将应付金额变量赋初值 0。

S2：以只读的方式打开商品文件 goods。

S3：输入所要选购的商品编号和数量。

S4：显示购物车清单。

S4.1：判断商品库存数量是否充足。

S4.1.1：如果购买数量大于商品库存数量，则给出“库存不足！商品未计价！按回车键返回!”的提示。

S4.1.2：如果库存充足，则系统显示购买商品的总价格。

S4.2：确定是否购买（Y/N）。

S4.2.1：如果购买，则调用结算功能。

S4.2.2：如果取消购买，则返回前台主界面。

（3）商品销售流程图如图 3.35 所示。

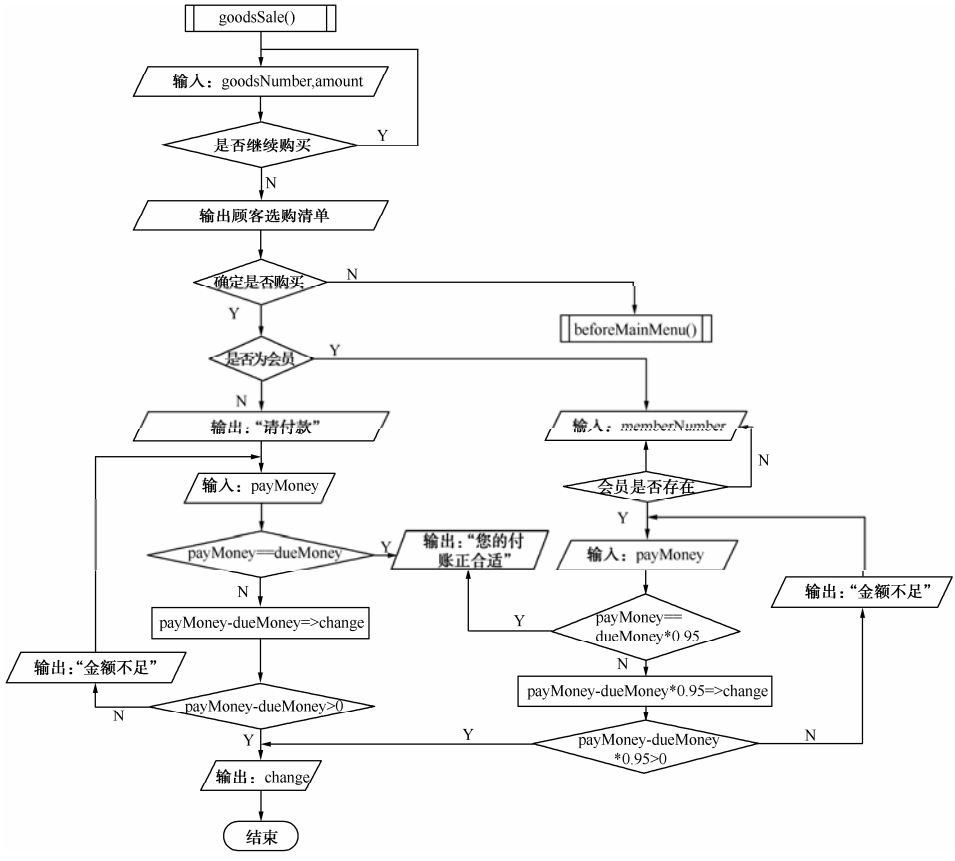


图 3.35　商品销售流程图

任务实现

步骤一：

（1）声明商品销售函数 void goodsSale()。

（2）定义商品销售函数 void goodsSale()。

（3）调用商品销售函数，在前台主界面调用。

步骤二：编码实现。

```
//商品销售——购物车清单
   void goodsSale()
{
      struct goods sp;
      int chooseGoods[100][2];//chooseGoods 用于保存客户选购的商品编号和数量
      FILE *fp1;
      char con;//是否继续购买
      int stock=0,buyKind=0,i;//buyKind 用于记录客户购买商品的种类
      float dueMoney=0,payMoney,change ;//定义应付金额、实付金额和找零变量
      if((fp1=fopen("data\\goods.dat","rb"))==NULL)//提取商品信息
      {
            printf("cannot open file\n");
            return;
      }
      system("cls");//清屏
      printf("\n\n");
      printf("\t*********************************************************\n");
      printf("\t                    商品销售                             \n");
      printf("\t*********************************************************\n\n");
//选购商品
      while(1){
            printf("\n\t 请输入你要购买的商品的编号：");
            scanf("%d",&chooseGoods[buyKind][0]);
            printf("\n\t 请输入你要购买的商品的数量：");
            scanf("%d",&chooseGoods[buyKind][1]);
            printf("\n\t\t\t\t 继续购买吗？(Y/N)：");
            scanf(" %c",&con);
            if(con=='n' || con=='N'){
                  break;
            }
            buyKind++;
      }


            //显示客户购买商品的清单
      system("cls");//清屏
      printf("\n\n");
      printf("\t      *********************************************************\n");
```

```
            printf("\t                                    购物车清单                                    \n");
            printf("\t ***********************************************************\n\n\n");
            printf("\t\t 商品编号\t 商品名称\t 商品单价\t 购买数量\n\n");
            for(i=0;i<=buyKind;i++)
            {
                while(!feof(fp1)){
                    fread(&sp,sizeof(struct goods),1,fp1);
                    if(chooseGoods[i][0]==sp.goodsNumber)
                    {
                       int tempAmount = sp.amount;

                       if(tempAmount < chooseGoods[i][1])
                           {

printf("\n\n\t\t%d\t\t%s\t\t%.2f\t\t%d\n",chooseGoods[i][0],sp.goodsName,sp.salePrice,sp.amount+1);//在库存没有的情况下，amount 的值为-1。
                                printf("\t       ------------------------------------------------------------\n");
                                printf("\n\n\t\t 库存不足！ 商品未计价！ 按回车键返回!");
                                getchar();
                                getchar();
                                beforeMainMenu();
                       }
                           else
                           {
                           dueMoney=dueMoney+chooseGoods[i][1]*sp.salePrice;
        printf("\t\t%d\t\t%s\t\t%.2f\t\t%d\n",chooseGoods[i][0],sp.goodsName,sp.salePrice,chooseGoods[i][1]);
        printf("\t----------------------------------------------------------------------\n");
                                rewind(fp1);
                       }
                           break;
                    }
                }
            }
        fclose(fp1);fclose(fp2);
        }
```

1．二维数组定义

C 语言允许构造多维数组。因为多维数组元素有多个下标，以标志它在数组中的位置，所以也称为多下标变量。顾名思义，二维数组是有两个下标的数组。

（1）二维数组定义的一般形式是：

类型说明符 数组名[常量表达式 1][常量表达式 2] = {{常量列表}，…}

其中，常量表达式 1 表示第一维下标的长度，常量表达式 2 表示第二维下标的长度。

（2）功能：定义一个二维数组，并给每个数组元素赋初值。

（3）说明：

① 数据类型是全体数组元素的数据类型，数组名用标志符表示。

② 格式中的两个整型常量表达式分别代表数组具有的行数和列数。注意，左边的常量表达式表示数组的行，右边的常量表达式表示数组的列。

③ 与数组一样，二维数组的下标值也是从 0 开始的。

④ 数组的各元素占有连续的存储空间，数组的各元素按行的顺序排列。

例如：

```
int a[3][3];
```

说明了一个 3 行 3 列的数组，数组名为 a，其下标变量的类型为整型。该数组的下标变量共有 3×3 个，即

```
a[0][0],a[0][1],a[0][2]
a[1][0],a[1][1],a[1][2]
a[2][0],a[2][1],a[2][2]
```

（4）二维数组的初始化：在｛｝中给出各数组元素的初值，各初值之间用逗号分开。把｛｝中的初值依次赋给各数组元素。有如下 5 种初始化方式。

① 初值按行的顺序依次排列，每行都用一对花括号括起来，各行之间用逗号隔开。例如：

```
int a[2][3]={{2,3,4},{7,8,9}};
```

② 不分行的初始化。二维数组可以像一维数组那样，将所有元素的初值写在一对花括号内。编译系统将会为这些有序数据按数组元素在内存中排列的顺序按行依次为各元素赋初值。例如：

```
int a[3][3]={1,2,3,4,5,6,7,8,9};
```

③ 对二维数组中的部分元素进行初始化。例如：

```
int a[2][3]={{1,2},{3}};
```

④ 如果对二维数组的全部元素都进行初始化，可以省略第一维的定义，但不能省略第二维的定义。例如：

```
static int a[ ][3]={1,2,3,4,5,6};
```

⑤ 如果只对部分数组元素提供初值，又省略了第一维的长度，那么应分行赋初值，即使某行没有对应的初值，也必须保留对应该行的一对花括号。例如：

```
int a[ ][4]={{1,3},{ },{5,6,7},{8,9,10,11}};
```

2．二维数组应用

【例 3.42】

```
//功能：从键盘上给 2*3 数组赋值，并在屏幕上显示出来
#define Row 2
#define A 2
#define B 3
#include "stdio.h"
main()
{
```

```
        int i, j, array[A][B];
        for(i=0; i<A; i++)
                for(j=0; j<B; j++)
                {
                printf("请输入数组 array[%2d][%2d]:",i,j);
                scanf("%d",&array[i][j]);
              }
    printf("\n");
    for(i=0;i<A;i++)
            {
            for(j=0;j<B;j++)
                    printf("%d\t",array[i][j]);
            printf("\n");
            }
    getch();
}
```

【例 3.43】

```
    //输入 4 名学生 3 门课程的成绩，算出每位学生的平均分，打印出成绩表。然后再分别统计出每门课程的平均分
    #include "stdio.h"
    main( )
    {
        int a[4][4];
        int i,j,t;
        float x;
        printf("请输入成绩：\n");
        printf("课程 1   课程 2   课程 3 \n");
        for( i=0; i<4; i++)
                for(j=0; j<3; j++)
                scanf("%d",&a[i][j]);
        for( i=0; i<4; i++)
            {
                t=0;
                for(j=0; j<3; j++)
                t=t+a[i][j];
                a[i][3]=t/3;
        }
        printf(" 课程 1      课程 2      课程 3      平均分\n");
         for(i=0; i<4; i++)
        {
                for(j=0; j<4; j++)
                printf("%6d",a[i][j]);
                printf("\n");
        }
        for(i=0; i<3; i++)
```

```
        {
            x=0;
            for(j=0; j<4; j++)
            x=x+a[j][i];
            printf("第%d 门课的平均分是:%4.1f\n",i+1,x/4.0);
    }
}
```

（1）编写一个程序，将一个 2 行 3 列的矩阵转置。

例如：

1 2 3

4 5 6

转置后为：

1 4

2 5

3 5

（2）编程求一个 3×3 的矩阵中主对角线中的最大值。

（3）阅读下面的程序，说明其功能。

```
main( )
{
int i,k,row=0,colum=0,max;
static int a[3][4]={{5,2,1,4},{10,12,6,5},{0,7,-2,3}};
max=-10;
for (i=0;i<=2;i++)
    for (k=0;k<=3;k++)
        if (a[i][k]>max)
        {
          max= a[i][k];
          row=i;
          colum=k;
          }
printf("max=%d, row=%d, colum=%d\n",max,row+1,colum+1);
}
```

（4）输出以下杨辉三角形（要求输出 8 行）。

1

1　1

1　2　1

1　3　3　1

1　4　6　4　1

1　5　10　10　5　1

… … …

拓展任务

试重新设计“购物车清单”的格式，然后进行编码实现。

3.4.10 子任务十：商品销售——动态处理商品数量的功能实现

任务描述

商品销售模块涉及商品购物车清单、动态处理商品数量和商品结算等功能。本节任务的主要目的是实现购买商品后，原商品文件中的商品数量随之动态地改变。

任务分析与设计

（1）动态处理商品数量的基本流程如表 3.44 所示。

表 3.44 动态处理商品数量的基本流程

1．定义动态处理商品数量函数	2．打开商品文件 goods.dat
3．将商品文件中的数据写入结构体数组中	4．将库存商品的数量减去已销售的商品数量
5．将结构体数组中的数据写入商品文件 goods.dat 中	6．关闭商品文件 goods.dat

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义商品结构体数组 sp[100]。

S1.2：定义文件指针变量 fp，用于操作商品 goods.dat 文件。

S1.3：定义整型变量 i 和 j 作为循环变量。

S1.4：定义整型变量商品编号 goodsNumber，并初始化为 0。

S2：以只读的方式打开商品文件 goods.dat。

S3：将商品文件中的数据写入结构体数组中。

S3.1：判断文件是否结束？

S3.1.1：如果没有遇到文件结束标志，从商品文件中读取数据写入结构体数组，同时将商品数量加 1。

S3.1.2：如果遇到文件结束标志，则关闭文件。

S4：确定销售后商品的数量，即将库存商品的数量减去已销售的商品数量。

S4.1：判断所选购的商品编号与结构体数组中的商品编号是否相同？

S41.1：如果相同，将结构体中的商品数量减去已销售的商品数量。

S4.1.2：如果不同，判断下一条。

S4.2：直到销售的所以商品数量都改变为止。

S5：将结构体数组中的数据写入商品文件中。

S5.1：以写的方式打开商品文件 goods.dat。

S5.2：将结构体中的数据循环写入商品文件 goods.dat 中。

S6：关闭商品文件 goods.dat。

任务实现

步骤一：

（1）声明商品数量处理函数 void amountDeal(int chooseGoods[][2],int buyKind);其中有两个参数，即 chooseGoods[][2]和 buyKind，前者为顾客所选购商品的编号，后者为顾客所选购商品的种类。

（2）定义商品数量处理函数 void amountDeal(int chooseGoods[][2],int buyKind)。

（3）调用动态处理商品数量函数，在商品销售中调用。

步骤二：编码实现。

```
//商品销售——动态处理商品数量
  void amountDeal(int chooseGoods[][2],int buyKind)
{
    struct goods sp[100];
    FILE *fp;
    int i,j;
    int goodsNumber=0;
    if((fp=fopen("data\\goods.dat","rb"))==NULL)
    {
        printf("cannot open file\n");
        return ;
    }
//将信息写入结构体数组
while (!feof(fp))
    {
        if(fread(&sp[goodsNumber],sizeof(struct goods),1,fp)==1)
        goodsNumber++;
    }
    fclose(fp);


    //将库存商品的数量减去已销售的商品数量
        for(j=0;j<goodsNumber;j++)
    {
        for(i=0;i<=buyKind;i++)
        {
            if(sp[j].goodsNumber==chooseGoods[i][0])
            {
                sp[j].amount=sp[j].amount-chooseGoods[i][1];
                break;
            }
        }

    }
```

```
    //将结构体数组中的数据写入商品文件中
    if((fp=fopen("data\\goods.dat","wb"))==NULL)
    {
        printf("cannot open file\n");
        return ;
    }
    for(i=0;i<goodsNumber;i++)
    {
        if(fwrite(&sp[i],sizeof(struct goods),1,fp)!=1)
            printf("write error");
    }
    fclose(fp);
}
```

1．goto语句

格式：goto〈语句标号〉。

功能：当程序执行到 goto 语句时，转到语句标号指定的语句去执行。

（1）使用 goto 语句时，需要预先指定一个有标号的可执行语句作为目的位置，这个有标号的语句必须与 goto 语句在同一个函数内。〈语句标号〉必须用标志符表示（格式为标号句行），不能用整数作为标号。

（2）与 if 语句一起构成循环结构。从结构化程序设计角度考虑，一般不主张使用 goto 语句，以免造成程序流程的混乱，使程序不易理解与调试。

（3）goto 语句为无条件转向语句，会跳过程序中的一段代码并转到一个指定的目的位置。

【例 3.44】

使用 goto 语句实现求解 1～100 累计和的程序如下：

```
main()
{
int    n=1, sum=0;
loop
sum += n;
if (n<=100)
{
n++;
goto    loop;
}
    printf("sum=%d\n", sum);
}
```

其中，“loop”语句标号（格式为标号:句行）的命名遵循标志符命名规则。

2．几种循环的比较

（1）各种循环语句均可以用来处理同一问题，在一般情况下，它们可以互相代替。但是，基于结构化程序设计的思想，一般不提倡用 goto 语句和 if 语句构成循环。

（2）while 和 do while 循环，只在 while 后面指定循环条件，在循环体中包含反复执行的操作语句，包括使循环趋于结束的语句（如 i++或 i--等）；for 循环可以在表达式 3 中包含使循环趋于结束的语句，甚至可以将循环体中的操作全部放到表达式 3 中。因此，for 语句的功能更强，凡用 while 循环能完成的，用 for 循环都能实现。

（3）while 语句和 do while 语句只有一个表达式，用于控制循环是否进行；for 语句有三个表达式，不仅可以控制循环是否进行，而且能为循环变量赋初值及不断修改循环变量的值。for 语句比 while 和 do while 语句的功能更强，更灵活。for 语句中的三个表达式可以是任何合法的 C 语言表达式，而且可以部分省略或全部省略，但其中的两个分号不能省略。

（4）用 while 和 do while 循环时，循环变量初始化的操作应在 while 和 do while 语句之前完成。for 语句可以在表达式 1 中实现循环变量的初始化。

（5）for 循环和 while 循环语句结构均是先判断循环条件，条件成立，才执行循环体，具有“先判断，后执行”的特点，for 语句和 while 语句可能一次也不执行循环体；而 do while 循环语句则是先执行循环体，然后再判断循环条件，具有“先执行，后判断”的特点。do while 语句至少执行一次循环体。for 和 while 循环属于“当型”循环，而 do while 循环属于“直到型”循环。

（6）while 语句和 do while 语句多用于循环次数不定的情况，而 do while 语句更适合用于第一次循环肯定执行的场合。在初值、增量控制条件明显或循环次数已经给定的情况下，最好选用 for 语句。

（7）对 while 循环、do while 循环和 for 循环，可以用 break 语句跳出循环，用 continue 语句结束本次循环；而对 goto 语句和 if 语句构成的循环，不能用 break 语句和 continue 语句进行控制。

【例 3.45】

```
//用 for 循环求 100 以内自然数中能被 3 整除的数
#include<stdio.h>
main()
{
int   i,s=0;
for(i=3,i<=100;i+=3)
      if(i%3==0) s+=I;
printf("\n   %d",s);
}
//用 while 循环输出 100 以内能被 3 整除且至少一位是 5 的所有整数
#include<stdio.h>
main()
{
int   i,n1,n2,n3;
i=3;
while(i<=100)
{
n3=i/100;
n2=(i-n3*100)/10;
n1=i%10;
if((i%3= =0)&&(n3= =5 || (n2= =5b || (n1= =5))
```

```
printf("%d,",i);
i++;
}
return 0;
}
//用 do_while 循环设计程序，使用键盘接收字符时以“#”号结束
#include<stdio.h>
main()
{
char c;
printf("\n 请输入一个字符串并以"#"结束");
do
{
    c=getchar();
    putchar(c);
}while(c!='#');
}
```

（1）求 S=1+1/2+1/3+…+1/100 的值。

（2）求 2/1+3/2+4/3+…+20/19 的值。

（3）打印出所有的“水仙花数”。所谓“水仙花数”是指一个三位数，其中各位数字的立方和等于该数的本身。

（4）编写程序，输入若干学生的数学成绩，以-1 作为终止值，计算这些学生的数学平均成绩并输出。

（5）阅读下面的程序，说明其功能。

```
#incluce<stdio.h>
 #define FLAG 58

    void main()
 {
    int data;
    int count;

    count=0;
    do
    {
        printf("请输入一个整数:");
        scanf("%d",&data);
        if(data>FLAG)
            printf("%d 太大了,请再试一试。\n",data);
        else if(data<FLAG)
            printf("%d 太小了，请再试一试。\n",data);
        else
```

```
            printf("你猜中了。\n");

        count++;
    }while(data!=FLAG);

    printf("你一共猜了%d 次才猜对。",count);
}
```

拓展任务

用其他形式的循环格式改写“动态处理商品数量”模块的代码。

3.4.11 子任务十一：商品销售——商品结算的功能实现

任务描述

商品销售模块涉及商品购物车清单、动态处理商品数量和商品结算等功能。本节任务的主要目的是实现商品结算功能。如果顾客是会员，则所有商品一律为 9.5 折优惠。

任务分析与设计

（1）商品结算的基本流程如表 3.45 所示。

表 3.45 商品结算的基本流程

1. 该任务为商品销售函数中的一部分	2. 打开商品文件 goods.dat 和临时文件 temp.dat
3. 将商品文件中的数量处理之后导入临时文件中	4. 删除原有商品文件
5. 将临时文件中的商品数据复制到新的商品文件 goods.dat 中	6. 删除临时文件

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：在商品销售函数的基础上，定义文件指针变量 fp2，用于操作会员文件 member.dat。

S1.2：定义一个整型变量 flag 作为会员标志文件。如果 flag=0，则代表不是会员；如果 flag=1，则代表是会员。将变量 flag 的初值设置为 0。

S2：以只读的方式打开会员文件 member.dat。

S3：在商品销售——购物车清单的基础上，列出商品总价。

S4：确认是否购买所选购的商品。

S4.1：如果不购买，则关闭所有文件，返回商品销售界面。

S4.2：如果购买，则进行结算。

S5：商品结算。

S5.1：判断顾客是否为会员？（Y/N）

S5.1.1：输入会员编号，调用会员编号查询函数 numberQuery()。

S5.1.2：如果是会员，则输入顾客实付金额。

S5.1.3：将实付金额与商品应付金额进行比较。

S5.1.3.1：如果相同，则系统提示“您的付账正合适!!! 谢谢您的光临，再见!!!”。

S5.1.3.2：如果不同，则进行找零，即零钱＝实付金额－应付金额×0.95。

S5.1.3.3：当零钱<0 时，则系统提示“金额不足，请重新操作!!!”。

S5.1.3.4：当零钱>0 时，则系统提示所找的零钱数。

S5.1.4：如果不是会员，则直接付款，不打折。

任务实现

编码实现。

```
//商品销售——商品结算的功能实现
……上接商品销售——购物车清单代码
    printf("\n 您购买的商品的总价格是：%.2f\n",dueMoney);
    printf("\n\t\t 您确认购买吗？(Y/N)：");
    scanf(" %c",&con);
    if(con=='y' || con=='Y')
    {
        printf("\n\t\t 您是会员吗？(Y/N)：");
        scanf(" %c",&con);
        if(con=='Y' || con=='y')
        {
            //检查是否为会员
            while(1)
            {
                printf("请输入您的会员编号；");
                scanf("%d",&number);
                if(!numberQuery(number,1))
                {
                    flag=0;
                    printf("无此会员编号；\n");
                    continue;
                }
                else
                {
                    flag=1;
                    break;
                }
            }
        }
        //完成商品的总数量处理工作
        amountDeal(chooseGoods,buyKind);
        //结算找零
        printf("\n\n 请您付账：");
        scanf("%f",&payMoney);
        if(flag==0)
        {//处理非会员的情况
            if(payMoney==dueMoney)
```

```
                {
                    printf("您的付账正合适！！！谢谢您的光临，再见！！！\n");
                    getchar();getchar();
                }
                else
                {
                  change = payMoney-dueMoney;
                  if(change >0)
                     {
                        printf("找您:%.2f\n",change);
                        printf("谢谢您的光临，再见！！！\n");
                        getchar();getchar();
                    }
                    else
                    {
                       printf("金额不足，请重新操作！！！\n");
                       getchar();getchar();
                  }
                }
            }
            else
            {//处理会员付费情况，9.5 折
                if(payMoney==dueMoney*0.95)
                {
                    printf("您的付账正合适！！！谢谢您的光临，再见！！！\n");
                    getchar();getchar();
                }
                else
                {
                     change = payMoney-dueMoney*0.95;
                     if(change >0)
                     {
                      printf("找您:%.2f\n",change);
                         printf("谢谢您的光临，再见！！！\n");
                         getchar();getchar();
                     }
                    else
                    {
                      printf("金额不足，请重新操作！！！\n");
                      getchar();getchar();
                    }

                }
            }
        }
```

说明：此编码放在商品销售——购物车清单代码的后面。

1．链表结构

链表是最简单也是最常用的一种数据结构。它是对动态获得的内存进行组织的一种结构。用数组存放数据时，必须事先定义固定的长度，链表则没有这种限制，它可以根据需要开辟内存单元。单链表的结构如图 3.36 所示。

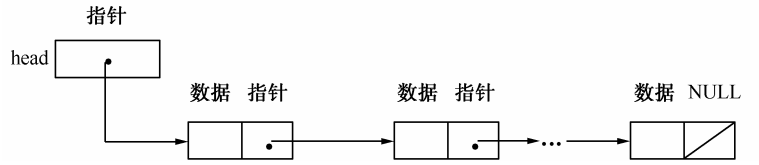


图 3.36　单链表的结构

（1）头指针变量 head：它存放一个地址。该地址指向一个链表元素。

（2）链表中的每个元素称为结点，由下列 2 个域组成。

① 数据域：存储结点本身的信息，即用户需要的实际数据。

② 指针域：指向后继结点的指针，即下一个结点的地址。

（3）尾结点的指针域置为“NULL（空）”，作为链表结束的标志。

2．对链表的基本操作

对链表的基本操作主要有以下 4 种。

（1）创建链表是指从无到有地建立起一个链表，即往空链表中依次插入若干结点，并保持结点之间的前驱和后继关系。

（2）检索操作是指按给定的结点索引号或检索条件，查找某个结点。如果找到指定的结点，则称为检索成功；否则，称为检索失败。

（3）插入操作是指在结点 ki-1 与 ki 之间插入一个新的结点 k'，使线性表的长度增 1，并且 ki-1 与 ki 的逻辑关系发生如下变化：插入前，ki-1 是 ki 的前驱，ki 是 ki-1 的后继；插入后，新插入的结点 k'成为 ki-1 的后继、ki 的前驱。

（4）删除操作是指删除结点 ki，使线性表的长度减 1，并且 ki-1、ki 和 ki+1 之间的逻辑关系发生如下变化：删除前，ki 是 ki+1 的前驱、ki-1 的后继；删除后，ki-1 成为 ki+1 的前驱，ki+1 成为 ki-1 的后继。

3．C语言对链表结点结构的描述

在 C 语言中，用结构类型来描述结点结构。例如：

```
struct   grade
{
    char   no[5];                    /*学号*/
    int    score;                    /*成绩*/
    struct   grade   *next;          /*指针域*/
};
```

【例 3.46】

建立链表是指一个一个地输入各结点数据，并建立起各结点前后相链接的关系。建立单

向链表有两种方法：插入表头（先进后出）方法和链表尾（先进先出）方法。插入表头的特点是，新产生的结点作为新的表头插入链表。链表尾方法的特点是，新产生的结点接到链表的表尾。

编写一个 creat()函数，按照规定的结点结构，创建一个单链表（链表中的结点个数不限）。

基本思路：首先向系统申请一个结点的空间，然后输入结点数据域的（2 个）数据项，并将指针域置为空（链尾标志），最后将新结点插入到链表尾。对于链表的第一个结点，还要设置头指针变量。

另外，案例代码中的 3 个指针变量 head、new 和 tail 的说明如下。

（1）head——头指针变量，指向链表的第一个结点，用做函数返回值。

（2）new——指向新申请的结点。

（3）tail——指向链表的尾结点，用 tail->next=new，实现将新申请的结点，插入到链表尾，使之成为新的尾结点。

```
#define    NULL   0
#define    LEN        sizeof(struct grade)          /*定义结点长度*/
                                                    /*定义结点结构*/
struct   grade
   { char   no[7];                                  /*学号*/
     int    score;                                  /*成绩*/
     struct   grade   *next;                        /*指针域*/
    };
                                                    /*create()函数: 创建一个具有头结点的单链表*/
                                                    /*形参：无*/
                                                    /*返回值：返回单链表的头指针*/
struct grade *create(void )
    { struct grade *head=NULL, *new, *tail;
        int count=0;                                /*链表中的结点个数(初值为 0)*/
        for(   ;    ;   )                          /*默认 3 个表达式的 for 语句*/
            { new=(struct grade *)malloc(LEN);      /*申请一个新结点的空间*/
                                                    /*1.输入结点数据域的各数据项*/
printf("Input the number of student No.%d(6 bytes): ", count+1);
scanf("%6s", new->no);
if(strcmp(new->no,"000000")==0)                     /*如果学号为 6 个 0，则退出*/
       { free(new);                                 /*释放最后申请的结点空间*/
         break;                                     /*结束 for 语句*/
        }
printf("Input the score of the student No.%d: ", count+1);
scanf("%d", &new->score);
count++;                                            /*结点个数加 1*/
                                                    /*2.置新结点的指针域为空*/
new->next=NULL;
                                                    /*3.将新结点插入到链表尾，并设置新的尾指针*/
if(count==1)   head=new;                            /*是第一个结点，置头指针*/
       else   tail->next=new;                       /*不是第一个结点，将新结点插入到链表尾*/
        tail=new;                                   /*设置新的尾结点*/
```

```
        }
    return(head);
}
```

【例 3.47】

链表的插入操作是要将一个结点插入到一个已有链表中的某个位置。该操作可以分两步完成，先找到插入点，再插入结点。

编写一个 insert()函数，完成在单链表的第 *i* 个结点后插入 1 个新结点的操作。当 *i*=0 时，表示新结点插入到第一个结点之前，成为链表新的首结点。

基本思路：通过单链表的头指针，首先找到链表的第一个结点；然后顺着结点的指针域找到第 *i* 个结点，最后将新结点插入到第 *i* 个结点之后。

```
/*函数功能：在单链表的第 i 个结点后插入 1 个新结点*/
/*函数参数：head 为单链表的头指针，new 指向要插入的新结点，i 为结点索引号*/
/*函数返回值：单链表的头指针*/
struct grade *insert(struct grade *head, struct grade *new, int i)
    { struct grade *pointer;
                                                    /*将新结点插入到链表中*/
if(head==NULL) head=new, new->next=NULL;            /*将新结点插入到 1 个空链表中*/
else                                                /*非空链表*/
        if(i==0) new->next=head,   head=new;        /*使新结点成为链表新的首结点*/
        else                                        /*其他位置*/
            {   pointer=head;
                                                    /*查找单链表的第 i 个结点(pointer 指向它)*/
                for(;   pointer!=NULL && i>1;   pointer=pointer->next,   i--)   ;
                    if(pointer==NULL)               /*越界错*/
                        printf("Out of the range, can't insert new node!\n");
                    else                            /*一般情况：pointer 指向第 i 个结点  */
                  new->next=pointer->next, pointer->next=new;
                  }
        return(head);
    }
```

即时训练

创建一个带有头结点的链表，将头结点返回给主调函数。链表用于存储学生的学号和成绩。新产生的结点总是位于链表的尾部。

拓展任务

试用多分支语句 switch 改写商品结算模块的程序代码。

3.4.12 子任务十二：库存预警模块的功能实现

任务描述

库存预警模块主要用于显示库存商品数量不足 100 的商品的信息，包括商品编号、商品

名称和商品数量，该模块主要是为库存管理员提供需要进货的列表，以免出现商品不足的现象。该商品列表随着商品销售的变化而动态改变。

任务分析与设计

（1）库存预警的基本流程如表 3.46 所示。

表 3.46　库存预警的基本流程

1．定义库存预警函数	2．呈现界面，如图 3.37 所示
3．打开商品文件 goods.dat，读取商品信息	4．判断各商品的数量是否不足 100。 如果不足 100，则显示该商品列表； 如果超过 100（包括 100），则不显示该商品列表
5．提示“按任意键返回”	6．关闭商品文件

图 3.37　库存预警界面

（2）自然语言描述。

S1：定义变量，初始化数据。

S1.1：定义商品结构体变量 sp。

S1.2：定义文件指针变量 fp，用于操作商品文件 goods.dat。

S1.3：初始化库存标志 stock 为 0。

S2：以只读的方式打开商品文件 goods.dat。

S3：判断文件是否结束。

S3.1：如果没结束，则从商品文件中读取数据，判断商品数量是否小于 100。

S3.1.1：如果小于 100，则将库存标志变量改为 1，列出该商品的编号、名称和数量。

S3.1.2：如果大于 100（包括 100），则不显示该商品。

S3.2：如果文件结束，则系统提示“按任意键返回”。

S4：关闭文件。

任务实现

```
//库存预警
```

```
void stockWarn()
{
    struct goods sp;
    FILE *fp;
    int stock=0;
    if((fp=fopen("data\\goods.dat","rb"))==NULL)
    {
        printf("cannot open file\n");
        return;
    }
    system("cls");//清屏
    printf("\n\n");
    printf("\t*************************************************************\n");
    printf("\t                          库存预警                           \n");
    printf("\t*************************************************************\n");

    while(!feof(fp))
    {
        if(fread(&sp,sizeof(struct goods),1,fp)==1)//不等 1 说明读错误，但系统并不给出提示
        {
            if(sp.amount<100)
        {
            if(stock==0)
            {
                printf("\n\n\t\t\t 如下商品数量不足 100，需要进货!\n\n");
            }
            stock=1;
            printf("\t 商品编号：%d\t",sp.goodsNumber);
            printf("\t 商品名称：%s\t",sp.goodsName);
            printf("\t 商品数量：%d\n",sp.amount);
            printf("        ------------------------------------------------------------------------\n");
        }
        }
    }

    printf("\n\n\n 按任意键返回");
    getchar();getchar();
    fclose(fp);

}
```

引导文献

位运算：前面介绍的各种运算都是以字节作为最基本位进行的。但是，在很多系统程序中，常要求在位（bit）一级进行运算或处理。C 语言提供了位运算的功能，这使得 C 语言也能像汇编语言一样用来编写系统程序。

位运算是指进行二进制位的运算。在系统软件中，常要处理二进制位的问题。例如，将一个存储单元中的各二进制位左移或右移 1 位，两个数按位相加等。在学习位运算前，应该对整数的补码存储方式有一定的了解，因为现在的微机都采用补码表示数。

C 语言提供下列 6 种位运算符。

（1）&：按位与运算符。

（2）|：按位或运算符。

（3）^：按位异或运算符。

（4）~ ：取反运算符。

（5）<<：左移运算符。

（6）>>：右移运算符。

1．按位与运算

按位与运算符“&”是双目运算符，其功能是参与运算的两个数各对应的二进位相与。只有对应的两个二进位均为 1 时，结果位才为 1；否则为 0。参与运算的数以补码方式出现。

例如，设 a=11，b=10，由于 11 和 10 的补码分别为 0000000000001011 和 0000000000001010，其对应的按位与过程如下式所示：

```
                0000000000001011
(&)             0000000000001010
----------------------------------
                0000000000001010
```

求真值得 a&b 的值为 10。

用下列程序可实现上述操作：

```
main()
{
int a=11,b=10,c;
c=a&b;
printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

按位与运算通常用来对某些位清 0 或保留某些位。例如，把 a 的高 8 位清 0，保留低 8 位，可做 a&255 运算（255 的二进制数为 0000000011111111）。

2．按位或运算

按位或运算符“|”是双目运算符，其功能是参与运算的两个数各对应的二进位相或。只要对应的两个二进位有一个为 1 时，结果位就为 1。参与运算的两个数均以补码出现。

例如，设 a=11，b=10，其对应的按位或过程如下式所示：

```
                0000000000001011
(|)             0000000000001010
----------------------------------
                0000000000001011
```

求真值得 a|b 的值为 11。

用下列程序可实现上述操作：

```
main()
{
int a=11,b=10,c;
c=a|b;
```

```
printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

3．按位异或运算

按位异或运算符“^”是双目运算符，其功能是参与运算的两数各对应的二进位相异或，当两对应的二进位相异时，结果为 1。参与运算的数仍以补码出现。

例如，设 a=11，b=10，其对应的按位异或过程如下式所示：

```
                0000000000001011
(^)             0000000000001010
----------------------------------
                0000000000000001
```

求真值得 a^b 的值为 1。

用下列程序可实现上述操作：

```
main()
{
int a=11,b=10,c;
c=a^b;
printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

4．求反运算

求反运算符“~”为单目运算符，具有右结合性，其功能是对参与运算的数的各二进位按位求反。

例如，~9 的运算为~(0000000000001001)。

结果为：1111111111110110。

5．左移运算

左移运算符“<<”是双目运算符，其功能把“<<”左边的运算数的各二进位全部左移若干位，由“<<”右边的数指定移动的位数，高位丢弃，低位补 0。

例如，a<<4 表示把 a 的各二进位向左移动 4 位。如 a=00000011（十进制数为 3），左移 4 位后为 00110000（十进制数为 48）。

6．右移运算

右移运算符“>>”是双目运算符，其功能是把“>>”左边的运算数的各二进位全部右移若干位，“>>”右边的数指定移动的位数。

例如，设 a=15，a>>2 表示把 000001111 右移为 00000011（十进制数为 3）。

应该说明的是，对于有符号数，在右移时，符号位将随同移动。当为正数时，最高位补 0；当为负数时，符号位为 1，最高位补 0 或补 1 取决于编译系统的规定。Turbo C 和很多系统规定为补 1。

阅读下列程序，说明其功能：

（1）

```
main()
{
```

```
unsigned a,b;
printf("input a number: ");
scanf("%d",&a);
b=a>>5;
b=b&15;
printf("a=%d\tb=%d\n",a,b);
}
```

（2）

```
main()
{
char a='a',b='b';
int p,c,d;
p=a;
p=(p<<8)|b;
d=p&0xff;
c=(p&0xff00)>>8;
printf("a=%d\nb=%d\nc=%d\nd=%d\n",a,b,c,d);
}
```

拓展任务

试编写“商品进货功能模块”，在商品库存不足时，能及时补充商品，并对库存量进行修改。

3.5 任务五 :“超市管理系统”的测试与优化

任务描述

本次测试主要针对本小组开发的超市管理系统的主要模块进行测试。

任务分析与设计

在“超市管理系统”需求规格说明书中列出的系统功能和性能都需要完成测试，在测试工作期间发现的所有缺陷都需要改正并确认。

“超市管理系统”的测试与优化的基本流程如表 3.47 所示。

表 3.47 “超市管理系统”的测试与优化的基本流程

1．登录模块调试	2．后台主界面调试
3．商品维护模块调试	4．会员管理模块调试
5．库存预警模块调试	6．前台主界面调试
7．商品销售模块的调试	8．上交测试文档

1．测试

自底向上依次进行单元测试（采用黑盒测试）、组装测试（采用黑盒测试），测试用例的设计应包括合理的和不合理的输入条件。

（1）编写目的：为保证超市管理系统的各模块功能的可靠实现，针对各模块内的各子模块进行测试。

（2）测试用例名如下。

supPart1：后台主界面模块的组装测试。

supPart1.1：商品维护模块的测试。

supPart1.2：会员管理模块的组装测试。

supPart1.2.1：会员添加模块的测试。

supPart1.2.2：会员查询模块的组装测试。

supPart1.2.2.1：按会员编号查询模块的测试。

supPart1.2.2.2：按会员姓名查询模块的测试。

supPart1.2.2.3：按会员身份证号查询模块的测试。

supPart1.2.3：会员统计模块的测试。

supPart1.2.4：会员删除模块的测试。

supPart1.3：库存预警模块的测试。

supPart2：前台主界面模块的组装测试。

supPart2.1：商品销售模块的测试。

supPart2.2：商品库存预警模块的测试。

（3）测试内容如表 3.48～表 3.60 所示。

表 3.48　后台主界面模块的组装测试（supPart1）

功　能	输　入	输　出
商品维护模块		
会员管理模块		
库存预警模块		

表 3.49　商品维护模块的测试（supPart1.1）

测试目标	验证商品添加能否正常写入文件中，并给出写入成功或写入失败的提示
方法	运行商品添加模块，输入要添加的商品的信息并确定
完成标准	系统提示输入商品编号、商品名称、商品生产日期、商品保质期、商品类型、商品进价、商品数量、商品销售价格等信息，系统提示“确定是否添加商品？（Y/N）”。输入“Y”确认后，系统提示“商品添加成功！”；否则，系统提示“继续添加商品吗？（Y/N）”；输入“Y”，则继续添加商品，输入“N”，返回后台主界面
需要考虑的注意事项	出现商品编号重复的时候，要给出提示“此编号已经存在，请重新输入编号”

表 3.50　会员管理模块的组装测试（supPart1.2）

功　能	输　入	输　出
会员添加模块		
会员查询模块		
会员统计模块		
会员删除模块		

表 3.51　会员添加模块的测试（supPart1.2.1）

测试目标	验证会员添加能否正常写入文件，并给出写入成功或失败的提示
方法	运行会员添加模块，输入要添加的会员的信息并确定
完成标准	会员编号自动生成，系统提示输入会员的姓名和会员身份证号，输入信息后，系统显示“确认会员添加吗？（Y/N）”输入 “Y”，则显示“添加会员成功！”、“继续添加会员吗？（Y/N）”；如果输入“N”，系统提示“继续添加会员吗？（Y/N）”；如果输入“Y”，则继续输入添加会员信息；否则，返回会员管理界面
需要考虑的注意事项	在处理会员编号时，需要实现自动编号功能

表 3.52　会员查询模块的组装测试（supPart1.2.2）

功　能	输　入	输　出
按会员编号查询模块		
按会员姓名查询模块		
按会员身份证号查询模块		

表 3.53　按会员编号查询模块的测试（supPart1.2.2.1）

测试目标	验证能否正常按会员编号进行查询
方法	运行按会员编号查询模块，输入要查询会员的编号
完成标准	在会员文件中搜索所输入的会员是否存在。如果存在，则显示该会员信息；如果不存在，则显示“无此会员编号，按任意键继续查询”
需要考虑的注意事项	输入的会员编号要为整数

表 3.54　按会员姓名查询模块的测试（supPart1.2.2.2）

测试目标	验证能否正常按会员姓名进行查询
方法	运行按会员姓名查询模块，输入要查询的会员姓名
完成标准	在会员文件中搜索所输入的会员是否存在。如果存在，则显示该会员信息；如果不存在，

	则显示“无此会员姓名，按任意键继续查询”
需要考虑的注意事项	

表 3.55　按会员身份证号查询模块的测试（supPart1.2.2.3）

测试目标	验证能否正常按会员身份证号进行查询
方法	运行按会员身份证号查询模块，输入要查询的会员身份证号

续表

完成标准	在会员文件中搜索所输入的会员是否存在。如果存在，则显示该会员信息；如果不存在，则显示“无此会员身份证号，按任意键继续查询”
需要考虑的注意事项	

表 3.56　会员统计模块的测试（supPart1.2.3）

测试目标	验证按会员编号降序或升序排序能否显示正常排序结果
方法	运行会员统计模块，系统提示选择会员排序的方式。确定排序方式后，显示会员列表
完成标准	显示“请输入你的选择:1.按会员编号降序排序 2.按会员编号升序排序 0.返回会员管理”，如果选择“1”，则显示会员降序排序列表；如果选择“2”，则显示会员编号升序排序列表；如果选择“0”，则返回会员管理界面
需要考虑的注意事项	

表 3.57　会员删除模块的测试（supPart1.2.4）

测试目标	验证会员删除是否成功
方法	运行会员删除模块，输入要删除的会员的编号
完成标准	系统提示“请输入会员编号：”，确定后，系统提示“确定要删除该会员吗？（Y/N）”。如果选择“Y”，则调用按会员编号查询模块，判断会员是否存在。如果存在，则删除会员；如果不存在，则系统提示“该会员不存在，按任意键继续”；否则返回会员删除界面
需要考虑的注意事项	

表 3.58　前台主界面模块的组装测试（supPart2）

功　能	输　入	输　出
商品销售模块		
库存预警模块		

表 3.59　商品销售模块的测试（supPart2.1）

测试目标	验证顾客选购商品后能否正常结算，商品销售后，库存是否随之改变

方法	运行商品销售模块，输入所选购的商品编号和商品数量，计算出商品总价，结算找零
完成标准	系统提示“请输入你要购买的商品编号和商品数量：”。用户输入商品编号和商品数量后，系统提示“是否继续购买？（Y/N）”。如果购买，则继续输入下一种商品的编号和数量；否则，列出顾客购买清单。用户确定清单后，系统提示“您确认购买吗？（Y/N）”。如果确认购买，系统提示“您是会员吗？（Y/N）”。如果是会员，系统提示“请输入会员编号：”调用会员编号查询模块。若是会员，则系统提示“请付账：”。输入钱款后，按所有商品打 95 折后收款。如果不是会员，则全额收款
需要考虑的注意事项	在销售模块中调用了商品数量处理函数和会员编号查询模块

表 3.60　商品库存模块的测试（supPart2.2）

测试目标	验证商品数量不足 100 的商品在库存预警模块中是否列出
方法	运行商品库存预警模块，列出库存不足商品的清单

续表

完成标准	查找商品文件中库存数量不足 100 的商品，将其信息列出
需要考虑的注意事项	

C 语言的常见错误如下。

1．书写标志符时，忽略了大小写字母的区别

```
main()
{
int a=5;
printf("%d",A);
}
```

编译程序把 a 和 A 认为是两个不同的变量名，而显示出错信息。C 认为大写字母和小写字母是两个不同的字符。习惯上，符号常量名用大写，变量名用小写表示，以增加可读性。

2．忽略了变量的类型，进行了不合法的运算

```
main()
{
float a,b;
printf("%d",a%b);
}
```

%是求余运算，得到 a/b 的整余数。整型变量 a 和 b 可以进行求余运算，而实型变量则不允许进行求余运算。

3．将字符常量与字符串常量混淆

```
char c;
c="a";
```

在这里就混淆了字符常量与字符串常量，字符常量是由一对单引号括起来的单个字符，字符串常量是一对双引号括起来的字符序列。C 规定以“\”作为字符串结束标志，它是由系统自动加上的，所以字符串“a”实际上包含两个字符：'a'和'\'，而把它赋给一个字符变量是不行的。

4．忽略了“=”与“==”的区别

在许多高级语言中，用“=”符号作为关系运算符“等于”。例如在 BASIC 程序中可以写为 if (a=3) then …

但是在 C 语言中，“=”是赋值运算符，“==”是关系运算符。例如：

```
if (a==3) a=b;
```

前者是进行比较，a 是否与 3 相等；后者表示如果 a 与 3 相等，则把 b 值赋给 a。由于习惯问题，初学者往往会犯这样的错误。

5．忘记加分号

分号是 C 语句中不可缺少的一部分，语句末尾必须有分号。

```
a=1
b=2
```

编译时，编译程序在“a=1”后面没发现分号，就把下一行“b=2”也作为上一行语句的一部分，这就会出现语法错误。改错时，有时在被指出有错的一行中未发现错误，就需要检查上一行是否漏掉了分号。

```
{ z=x+y;
t=z/100;
printf("%f",t);
}
```

对于复合语句来说，最后一个语句中的最后句尾分号不能忽略不写（这与 PASCAL 不同）。

6．多加分号

对于一个复合语句，例如：

```
{ z=x+y;
t=z/100;
printf("%f",t);
};
```

复合语句的花括号后不应再加分号，否则将会画蛇添足。

又如：

```
if (a%3==0);
i++;
```

本意是，如果 3 整除 a，则 i 加 1。但是，由于 if (a%3==0)后多加了分号，则 if 语句到此结束，程序将执行 i++语句，不论 3 是否整除 a，i 都将自动加 1。

再如：

```
for (i=0;i<5;i++);
```

```
{scanf("%d",&x);
printf("%d",x);}
```

本意是先后输入 5 个数，每输入一个数后再将它输出。由于 for()后多加了一个分号，使循环体变为空语句，此时只能输入一个数并输出它。

7．输入变量时，忘记加地址运算符“&”

```
int a,b;
scanf("%d%d",a,b);
```

这是不合法的。scanf 函数的作用是：按照 a、b 在内存的地址将 a、b 的值存进去。“&a”指 a 在内存中的地址。

8．输入数据的方式与要求不符

（1）scanf("%d%d",&a,&b);

输入时，不能用逗号作为两个数据间的分隔符，如下面的输入是不合法的：3，4。

输入数据时，在两个数据之间以一个或多个空格间隔，也可用回车键、Tab 键。

（2）scanf("%d,%d",&a,&b);

C 语言规定：如果在“格式控制”字符串中除了格式说明以外还有其他字符，则在输入数据时应输入与这些字符相同的字符。下面的输入是合法的：3，4。

此时，不用逗号而用空格或其他字符是不对的，如 3　4；3：4。

又如：

```
scanf("a=%d,b=%d",&a,&b);
```

输入应采用如下形式：

```
a=3,b=4
```

9．输入字符的格式与要求不一致

在用“%c”格式输入字符时，“空格字符”和“转义字符”都作为有效字符输入。

```
scanf("%c%c%c",&c1,&c2,&c3);
```

例如输入 a b c，字符“a”送给 c1，字符“b”送给 c2，字符“c”送给 c3，因为%c 只要求读入一个字符，所以后面不需要用空格作为两个字符的间隔。

10．输入/输出的数据类型与所用格式说明符不一致

例如，a 已定义为整型，b 定义为实型。

```
a=3;b=4.5;
printf("%f%d\n",a,b);
```

编译时不给出出错信息，但运行结果与原意不符。尤其需要注意这种错误。

11．输入数据时，企图规定精度

```
scanf("%7.2f",&a);
```

这样做是不合法的，输入数据时不能规定精度。

12．在switch语句中漏写break语句

例如，根据考试成绩的等级打印出百分制数段。

```
switch(grade)
{
    case 'A':printf("85~100\n");
    case 'B':printf("70~84\n");
    case 'C':printf("60~69\n");
    case 'D':printf("&lt;60\n");
    default:printf("error\n");
}
```

由于漏写了 break 语句，case 只起标号的作用，而不起判断作用。因此，当 grade 值为 A 时，printf 函数在执行完第一个语句后接着执行第二个至第五个 printf 函数语句。正确写法是，应在每个分支后再加上“break;”。例如：

```
case 'A':printf("85~100\n");
break;
```

13．忽视了while和do while语句在细节上的区别

（1）main()

```
{
int a=0,i;
scanf("%d",&i);
while(i==10)
{
    a=a+i;
    i++;
    }
printf("%d",a);
}
```

（2）main()

```
{
int a=0,i;
scanf("%d",&i);
do
{
    a=a+i;
    i++;
}while(i==10);
printf("%d",a);
}
```

可以看到，当输入 i 的值小于或等于 10 时，二者得到的结果相同；而当 i==10 时，二者结果就不同了。因为 while 循环是先判断后执行，而 do while 循环是先执行后判断。对于大于 10 的数，while 循环一次也不执行循环体，而 do while 语句则要执行一次循环体。

14．定义数组时误用变量

```
int n;
```

```
scanf("%d",&n);
int a[n];
```

数组名后用方括号括起来的是常量表达式，可以包括常量和符号常量，即 C 语言不允许对数组的大小做动态定义。

15．在定义数组时，将定义的“元素个数”误认为是可使用的最大下标值

```
main()
{static int a[10]={1,2,3,4,5,6,7,8,9,10};
printf("%d",a[10]);
}
```

C 语言规定：定义时使用 a[10]，表示 a 数组有 10 个元素，其下标值由 0 开始，所以数组元素 a[10]是不存在的。

16．字符数组只能在定义时初始化

```
char str[2];
str[2]= {"ab"};
```

这种写法会输出乱码。原因是字符数组与整型数组不同，整型数组可以在定义后初始化，但字符字组必须在定义时初始化，应改成 char str[2]={"ab"};。

17．在不应加地址运算符&的位置加了地址运算符

```
scanf("%s",&str);
```

C 语言编译系统对数组名的处理是：数组名代表该数组的起始地址，并且 scanf 函数中的输入项是字符数组名，不必再加地址运算符&。应改为：

```
scanf("%s",str);
```

18．同时定义了形参和函数中的局部变量

```
int max(x,y)
int x,y,z;
{z=(x>y)?x:y;
return(z);
}
```

形参应该在函数体外定义，而局部变量应该在函数体内定义。应改为：

```
int max(x,y)
int x,y;
{int z;
z=(x>y)?x:y;
return(z);
}
```

在进行系统的测试过程中，是否有如下问题？如果有，应加以解决。

（1）输入的实际参数与形式参数的个数是否相同？
（2）输入的实际参数与形式参数的属性是否匹配？
（3）输入的实际参数与形式参数的数量是否一致？
（4）调用其他模块时，所给实际参数的个数是否与被调模块的形参个数相同？
（5）调用其他模块时，所给实际参数的属性是否与被调模块的形参属性匹配？
（6）调用其他模块时，所给实际参数的数量是否与被调模块的形参数量一致？
（7）调用预定义函数时，所用参数的个数、属性和次序是否正确？
（8）是否存在与当前入口点无关的参数引用？
（9）是否修改了只读型参数？
（10）对全程变量的定义，各模块是否一致？
（11）是否把某些约束作为参数传递？
（12）是否有不合适或不相容的类型说明？
（13）是否有变量初始化或默认值错误？
（14）是否有不正确的变量名？
（15）文件属性是否正确？
（16）OPEN/CLOSE 语句是否正确？
（17）格式说明与输入/输出语句是否匹配？
（18）文件在使用前是否已经打开？
（19）是否处理了文件尾？
（20）是否处理了输入/输出错误？
（21）输出信息中是否有文字性错误？
（22）是否有输出的出错信息难以理解？

编写“用户管理模块”的测试文档。

第 4 章　项目赏析——学生成绩管理系统

前面已经介绍了用 C 语言进行软件开发的基本方法和流程，并介绍了数据输入、数据处理、数据输出的基本技术。本章通过对“学生成绩管理系统”项目开发过程的赏析，对用 C 语言解决实际问题的基本方法和步骤做一个完整的概括。

4.1　概述

4.1.1　学生成绩管理系统的背景

某个班级共有学生 36 名，假设只学习了外语、C 语言程序设计两门功课。现在，要求为这个班级编写一个学生成绩管理系统。在没有这个系统以前采用的是人工操作，操作基本流程是：教务科准备好空白成绩单（如表 4.1 所示）和空白总成绩单（如表 4.2 所示）；在期末考试结束后，每科老师领回一张空白成绩单，按要求填写单科成绩，并算出总成绩，如表 4.3 所示的外语成绩单、表 4.4 所示的 C 语言程序设计成绩单。各位老师把填好的单科成绩单，送到教务科，教务科的老师对这些成绩单进行汇总，并将汇总后的总成绩单（如表 4.5 所示），发到各班级。同时，教务科还对单科进行各分数段人数的统计，对总成绩进行排序等操作。单科总成绩计算方式为：平时成绩×20%+期中成绩×20%+期末成绩×60%。

表 4.1　空白成绩单

科成绩单

学　　号	姓　　名	平 时 成 绩	期 中 成 绩	期 末 成 绩	总　成　绩
050101001	王丽				
050101002	李丽				
050101005	张丽				
050101031	赵丽				

表 4.2　空白总成绩单

班总成绩单

学　　号	姓　　名	C　语　言	外　　语	总　成　绩	平 均 成 绩
050101001	王丽				
050101002	李丽				
050101005	张丽				
050101031	赵丽				

表 4.3　外语成绩单

外语　　科成绩单

学　号	姓　名	平时成绩	期中成绩	期末成绩	总　成　绩
050101001	王丽	90	76	85	84.2
050101002	李丽	100	100	93	95.8
050101005	张丽	90	76	85	84.2
050101031	赵丽	100	100	93	95.8

表 4.4　C 语言程序设计成绩单

C 语言程序设计　　科成绩单

学　号	姓　名	平时成绩	期中成绩	期末成绩	总　成　绩
050101001	王丽	80	66	81	77.8
050101002	李丽	90	87	75	80.4
050101005	张丽	65	96	65	71.2
050101031	赵丽	54	91	67	69.2

表 4.5　汇总后的总成绩单

050101　　班总成绩单

学　号	姓　名	C 语 言	外　语	总　成　绩	平均成绩
050101001	王丽	77.8	84.2	162	81
050101002	李丽	80.4	95.8	176.2	88.1
050101005	张丽	71.2	84.2	155.4	77.7
050101031	赵丽	69.2	95.8	165	82.5

4.1.2　系统流程概要

运行主程序后，输入密码（88888），弹出学生成绩管理系统主菜单，如图 4.1 所示。本系统有三个主要功能：成绩单管理（生成空白成绩单）、科目管理（录入单科成绩）、成绩汇总（把单科成绩汇总到班级总成绩单中）。当选择 1 时，弹出生成成绩单子菜单，如图 4.2 所示。选择 1 录入学生基本信息（姓名、学号），选择 2 添加学生信息，选择 3 删除学生信息，选择 4 显示所有学生信息，选择 5 返回到学生成绩管理系统主菜单。在图 4.1 中选择 2 时，进入科目管理模块子菜单。输入科目编号 1 时，进入如图 4.3 所示的 C 语言科目管理模块子菜单；输入科目编号 2 时，进入如图 4.4 所示的外语科目管理模块子菜单。科目管理模块主要有两个功能：录入成绩、修改成绩。输入科目编号 3 时，可返回到如图 4.2 所示的生成成绩单模块子菜单。在图 4.1 中选择 3 时，进入如图 4.5 所示的成绩汇总模块子菜单，共有四种功能：汇总、排序、查询、统计。选择相关选项，进入对应的模块。在系统运行中，产生四个文件：存储学生基本信息的文件（stu.dat）、存储外语成绩的文件（foreign.dat）、存储 C 语言的文件（c.dat）、存储汇总成绩的文件（total.dat）。

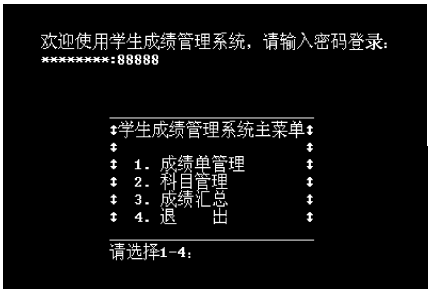


图 4.1　学生成绩管理系统主菜单

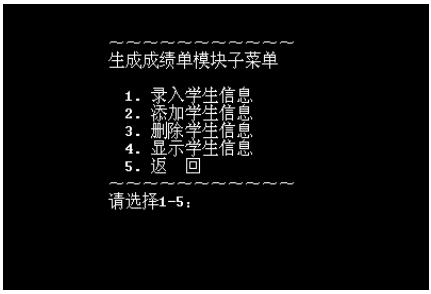


图 4.2　生成成绩单模块子菜单

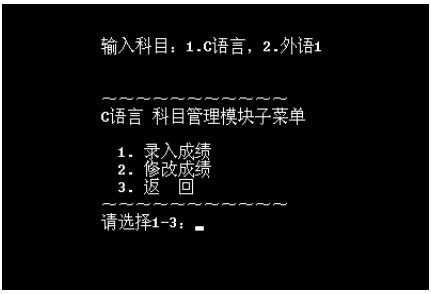


图 4.3　C 语言科目管理模块子菜单

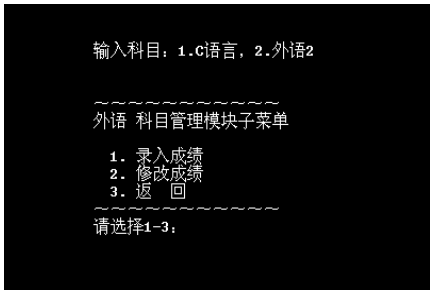


图 4.4　外语科目管理模块子菜单

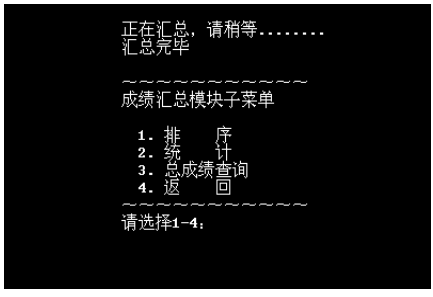


图 4.5　成绩汇总模块子菜单

4.2　明确问题

在设计系统前，必须清楚地知道该系统要解决什么问题，即要确定系统的目标和范围，从系统陈述中可以看出：此系统主要有三大功能：生成空白成绩单、填写单科成绩单、生成总成绩单，如图 4.6 所示。

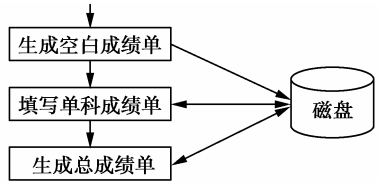


图 4.6　成绩管理的基本流程

4.3 分析

输入数据：
学生信息（学号、姓名）
单科成绩（平时成绩、期中成绩、期末成绩）
输出数据：
空白成绩单（学号、姓名、平时成绩、期中成绩、期末成绩）
外语成绩单（学号、姓名、平时成绩、期中成绩、期末成绩、总成绩）
C 语言成绩单（学号、姓名、平时成绩、期中成绩、期末成绩、总成绩）
总成绩单（学号、姓名、C 语言、外语、总成绩、平均成绩）
数据需求分析如图 4.7 所示。

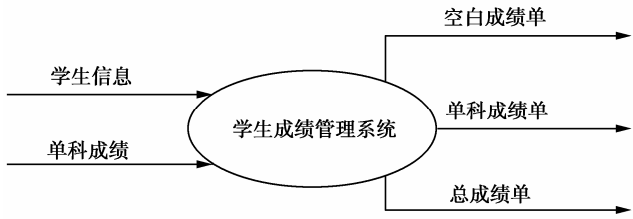


图 4.7　数据需求分析

计算单科总成绩公式：平时成绩×20%+期中成绩×20%+期末成绩×60%。
计算总成绩单中的总成绩公式：C 语言总成绩+外语总成绩。
计算总成绩单中的平均成绩公式：总成绩/2。

4.4 设计算法

4.4.1 概要设计

1．给出基本算法

S1：定义变量。
S2：密码处理。
S3：主菜单管理。
S4：空白成绩单管理。
S5：单科成绩单管理。
S6：总成绩汇总。

2．逐步细化

下面依据自顶向下、逐步求精的思路，采用模块化的方法逐步细化。
1）对 S1 步求精
S1.1：定义学生信息变量。
S1.2：定义单科成绩变量。
S1.3：定义总成绩变量。

学生信息变量、单科成绩变量、总成绩变量都是由多种类型的数据组成的，因此可以定义为结构体类型变量。

在这里不对 S2 进行求精。

2）对 S3 求精

S3.1：显示主菜单。

S3.2：处理子菜单。

显示主菜单之后，输入相关编号进入对应的子菜单处理。因为显示菜单、处理菜单是两个功能，所以用两个模块表述。

3）对 S4 求精

S4.1：显示子菜单。

S4.2：处理子菜单。

S4.3：录入学生信息。

S4.4：添加学生信息。

S4.5：删除学生信息。

S4.6：显示学生信息。

本模块是生成空白成绩单和空白总成绩单，先录入学生信息之后再进行添加、删除、显示等操作。

4）对 S5 求精

S5.1：显示子菜单

S5.2：处理子菜单

S5.3：录入单科成绩

S5.4：修改单科成绩

本模块是单科成绩的录入，把成绩录入到空白成绩单中，并计算出总成绩。由于有两科，所以对 S5.3、S5.4 进一步细化。

（1）S5.3 再次求精。

S5.3.1：录入外语成绩。

S5.3.2：录入 C 语言成绩。

（2）对 S5.4 再次求精。

S5.4.1：修改外语成绩。

S5.4.2：修改 C 语言成绩。

5）对 S6 求精

S6.1　显示子菜单。

S6.2　处理子菜单。

S6.3　汇总。

S6.4　排序。

S6.5　统计。

S6.6　查询。

把上面的分析结构用系统结构图的形式呈现，如图 4.8 所示。

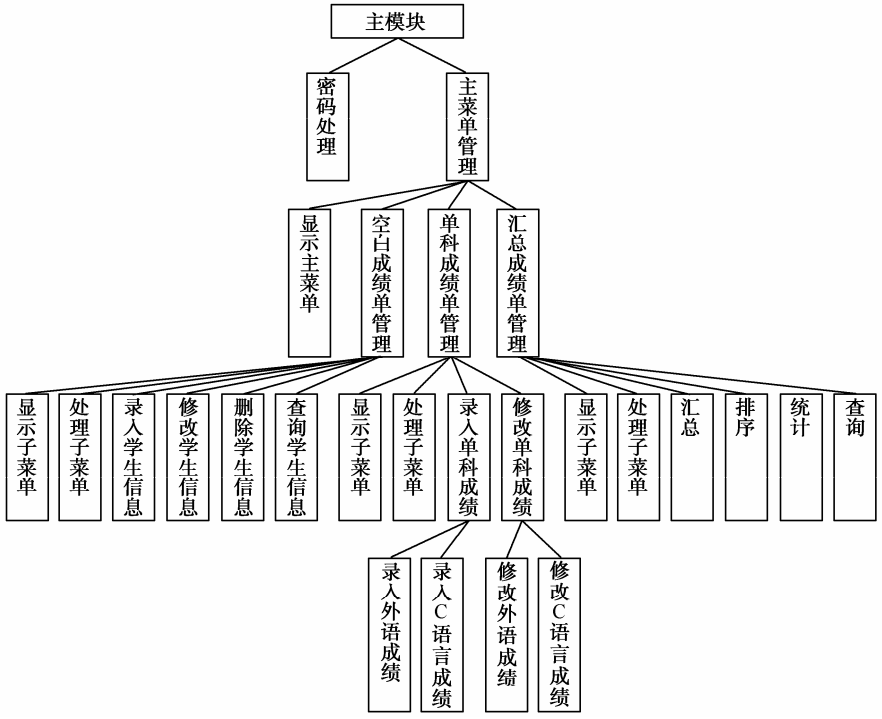


图 4.8　学生成绩管理系统结构图

4.4.2　详细设计

图 4.8 只给出了功能模块、部分模块之间的关系，对详细的模块之间的关系、每个模块的内部情况并没有给出。因此，下面对每个模块做进一步详细设计。

主模块的设计如表 4.6 所示。

表 4.6　主模块的设计

模　块　名	主　模　块	模 块 编 号	S0
函数名	main		
上层调用模块	系统	下层调用模块	S2：密码处理，S3：主菜单管理
函数参数	无	函数返回数据	无
功能	系统入口、密码处理、显示主菜单		
功能描述	S0.1：调用密码处理函数 S0.2：调用处理主菜单函数 S0.3：返回 S0.2		

主菜单处理模块详细设计，如表 4.7。

表 4.7　主菜单处理模块详细设计

模　块　名	处理选择菜单项	模 块 编 号	S3.2
函数名	selectMainMenu		
上层调用模块	int main（void）	下层调用模块	S3.1：显示主菜单 S4.2：处理子菜单 S5.2：处理选择菜单项 S6.2：处理选择菜单项
函数参数	无	函数返回数据	无
功能	显示主菜单、选择主菜单		
功能描述	S3.2.1：定义局部变量 selectItem S3.2.2：显示主菜单（模块） S3.2.3：输入变量 selectItem S3.2.4：当 selectItem=1 时，调用 S4.2：处理子菜单 当 selectItem=2 时，调用 S5.2：处理选择菜单项 当 selectItem=3 时，调用 S6.2：处理选择菜单项 当 selectItem=4 时，退出本系统 S3.2.5：返回 S3.2.2		

由于篇幅有限，读者可以用同样的方法完成其他模块的设计，详细情况见源码。

4.5　实现

```
/* stuAchiMange.c */
/*
```

学生成绩管理综合实例：一个班级考试有两门科目（C 语言、外语）。相关部门先给出空白成绩单，有姓名、学号、平时成绩、期中成绩、期末成绩等项目，要求把 C 语言、外语成绩按此成绩单的要求分别填写。最后，把全班所有科目的成绩汇总成一张总成绩单。

```
作者：陈显刚

版本 1.0
*/

#include <stdio.h>
#include <stdlib.h>                    /* exit() */
#include <string.h>

struct StuInfo                         /*  学生信息结构体变量  */
{
    char stuID[11];                    /*  学生学号  */
    char stuName[8];                   /*  学生姓名  */
};
```

```
struct SubjectList                                  /* 某科目成绩单 */
{
    struct StuInfo stuInfo;
    float subjectAchi[4];                           /* 科目成绩单 */
};

struct FinalAchiList                                /* 汇总后的成绩单 */
{
    struct StuInfo stuInfo;
    float classAchi[4];                             /* 两科成绩，总成绩，平均成绩 */
};


int handlePassword(void);                           /* 处理密码 */
void showMainMenu(void);                            /* 显示主菜单 */
void selectMainMenu (void);                         /* 选择处理主菜单 */

/* 生成成绩单模块所用函数：                          */
void displayMenuItem1(void);                        /* 处理生成成绩单菜单 */
void handleCreateAchiMenu(void);                    /* 处理生成成绩子菜单 */
void inputStuInfo(void);                            /* 录入学生信息 */
void addStuInfo(void);                              /* 添加学生信息 */
void deleteStuInfo(void);                           /* 删除学生信息 */
void displayStuInfo(void);                          /* 显示学生信息 */

/* 科目管理模块所用函数：                            */
void handleSubjectMenu(void);                       /* 科目管理处理菜单 */
void displaySubjectMenu(int subjectCode);           /* 显示科目管理菜单 */
void achievementInput(int subjectCode);             /* 输入相关科目成绩 */
void achievementmodify(int subjectCode);            /* 科目成绩修改 */
void updateAchievement(FILE *fPtr);                 /* 为单科录入成绩下一级函数 */
void editAchievement(FILE *fPtr);                   /* 修改单科成绩下一级函数 */

/* 成绩汇总所用函数： */
void handleAchievementMenu(void);                   /* 处理成绩汇总菜单 */
void displayAchievementMenu(void);                  /* 显示成绩汇总菜单 */
int   computeAchievement(struct FinalAchiList fin[]);                /* 汇总 */
void sortAchievement(struct FinalAchiList fin[],int n);              /* 排序 */
/* 分析、统计成绩 */
void statisticsAchievement(struct FinalAchiList fin[],int n);
void showGradeNumber(int cfLangueTotel[]); /* 显示统计信息      */
/* 查询成绩，依据个人姓名 */
void queryAchievement(struct FinalAchiList fin[],int n);
/* 显示汇总后的总成绩单 */
void displayTotelAchievement(struct FinalAchiList fin[],int n);
/* 统计两科：C 语言和外语各分数段人数 */
```

```
void totelGradeNumber(struct FinalAchiList fin[],
int cfLangueTotel[],int n,int m);
void    finishedInfo(void);

int main(void)                                          /* 主函数 */
{
    /* 密码处理，为 0 表示密码输入三次还有误，退出系统 */
    if(handlePassword()==0)
    {
        exit(1);
    }

    do
    {
        selectMainMenu ();                              /* 选择处理主菜单 */
    }while(1);

    return 0;
}

/*****************************************************************
函数名:handlePassword
函数功能:通过输入密码登录系统,输入若不正确，给三次机会
函数输入参数:无
函数返回值:当输入密码正确时返回 1,错误时返回 0
*****************************************************************/
int handlePassword(void)
{
    char secretCode[6];                                 /* 用于输入密码 */
    int secretcodeNumber=3;                             /* 输入密码三次后退出系统 */

    printf("\n\n\n");
    printf("\t 欢迎使用学生成绩管理系统，请输入密码登录：\n" );

    printf ("\t*********:");
    scanf("%s", secretCode);                            /* 输入密码  */
    for(;strcmp(secretCode,"88888")!=0 ;)
    {
        secretcodeNumber--;
        if ( secretcodeNumber <=0 )
        {
            break;                                      /* 密码输入三次后结束输入 */
        }
        printf( "\t 还有%d 次机会*********:",secretcodeNumber) ;
        scanf("%s", secretCode);
    }
```

```
    if ( secretcodeNumber <=0 )
    {
        return 0;
    }
    else
    {
        return 1;
    }
}

/*****************************************************************
函数名:showMainMenu
函数功能:显示主菜单
函数输入参数:无
函数返回值:无
*****************************************************************/
void showMainMenu(void)
{

    printf("\n\n");
    printf("\t\t_____\n");
    printf("\t\t\22 学生成绩管理系统主菜单\22\n");
    printf("\t\t\22                    \22\n");
    printf("\t\t\22   1. 成绩单管理     \22\n");
    printf("\t\t\22   2. 科目管理       \22\n");
    printf("\t\t\22   3. 成绩汇总       \22\n");
    printf("\t\t\22   4. 退    出       \22\n");
    printf("\t\t_____\n");
    printf("\t\t 请选择 1-4：");

}

/*****************************************************************
函数名:selectMainMenu
函数功能:输入菜单选项,1-4 进行相关操作
函数输入参数:无
函数返回值:无
*****************************************************************/
void selectMainMenu (void)
{
    int selectItem=0;

    do
    {
        showMainMenu();
        scanf("%d",&selectItem);
```

```
            switch(selectItem)
            {
               case 1:
                    handleCreateAchiMenu();                    /* 产生成绩单模块函数 */
                    break;
               case 2:
                    handleSubjectMenu();                       /* 科目管理 */
                    break;
               case 3:
                    handleAchievementMenu();                   /* 成绩汇总 */
                    break;
               case 4:
                    printf("谢谢使用，再见\n");
                    exit(0) ;
                    break;
               default:
                    printf("请输入 1-4 之间的整数");
            }

      }while(1);
}

/*****************************************************************
函数名:displayMenuItem1
函数功能:显示生成成绩单模块子菜单
函数输入参数:无
函数返回值:无
*****************************************************************/
void displayMenuItem1(void)
{
      printf("\n\n");
      printf("\t\t～～～～～～～～～～～\n");
      printf("\t\t 生成成绩单模块子菜单\n\n");
      printf("\t\t    1. 录入学生信息\n");
      printf("\t\t    2. 添加学生信息\n");
      printf("\t\t    3. 删除学生信息\n");
      printf("\t\t    4. 显示学生信息\n");
      printf("\t\t    5. 返    回\n");
      printf("\t\t～～～～～～～～～～～\n");
      printf("\t\t 请选择 1-5：");
}

/*****************************************************************
函数名:handleCreateAchiMenu
函数功能:处理生成成绩单子菜单,选择 1-5 进行相关操作
函数输入参数:无
函数返回值:无
```

```
**********************************************************/
void handleCreateAchiMenu(void)
{
    int selectItem=0;

    do
    {

        displayMenuItem1();
        scanf("%d",&selectItem);

        switch(selectItem)
        {
        case 1:
        /*输入学生信息，同时生成 C 语言、外语、成绩汇总、学生信息文件*/
            inputStuInfo();
            break;
        case 2:
        /* 添加班级学生姓名和学号，同时更新到 C 语言、外语、成绩汇总、学生信息文件 */
            addStuInfo();
            break;
        case 3:
        /* 由于退学等原因，删除学生信息，同时更新到 C 语言、外语、成绩汇总、学生信息文
           件 */
            deleteStuInfo();
            break;
        case 4:
        /* 显示学生姓名和学号，不显示成绩 */
            displayStuInfo();
            break;
        case 5:
            return ;
            break;
            default:
                printf("请输入 1-5 之间的整数");
        }

    }while(1);

}

/*****************************************************************
函数名:inputStuInfo
函数功能:录入学生信息,录入学生姓名、学号分别生成四个文件（学生基本信息、C 语言、外语、
总成绩单）。若已有信息，则删除原来的信息重新输入
函数输入参数:无
函数返回值:无
```

```
**********************************************************/
void inputStuInfo(void)
{
    struct StuInfo stu;          /* 学生信息（姓名和学号）结构体 */
    /* （C 语言、外语、姓名、学号、平时成绩、期中成绩、期末成绩、总成绩）结构体 */
    struct SubjectList subc,subf;
    /* 总成绩（姓名、学号、C 语言成绩、外语成绩、总成绩、平均成绩）结构体 */
    struct FinalAchiList fin;

    char ch;
    int j;

    FILE *ptStu;                        /* 学生信息文件指针 */
    FILE *ptClanguage;                  /* C 语言成绩单文件指针 */
    FILE *ptForeignLanguage;            /* 外语成绩单文件指针 */
    FILE *ptTotelList;                  /* 总成绩单文件指针 */

    printf("\n");
    printf("\t\t 执行此操作，原来存入文件的内容会被全都删除\n
          \t\t 是否要继续输入(y/n)");
    ch=getch();
    if(ch!='y')
        return;

    /* 建立学生信息文件,若之前有信息，则删除 */
    if((ptStu=fopen("stu.dat","wb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }
    /* 建立 C 语言成绩单文件,若之前有信息，则删除 */
    if((ptClanguage=fopen("c.dat","wb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }
    /* 建立外语成绩单文件,若之前有信息，则删除 */
    if((ptForeignLanguage=fopen("foreign.dat","wb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }
    /* 建立总成绩单文件,若之前有信息，则删除 */
    if((ptTotelList=fopen("totel.dat","wb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
```

```
    }

    printf("\n\n\t\t 录入学生的学号和姓名，输入 no 结束\n");

    printf("\t\t 输入姓名:");
    scanf("%s",stu.stuName);

    printf("\t\t 输入学号(最多 11 位):");
    scanf("%s",stu.stuID);
    while(strcmp(stu.stuName,"no")!=0)/* 输入姓名为 no 时结束 */
    {

        subc.stuInfo=stu;                    /* 把学生信息分别存入其他文件中 */
        subf.stuInfo=stu;
        fin.stuInfo=stu;

        for(j=0;j<4;j++)                     /* 为成绩单文件赋初值，单精度 */
        {
            subc.subjectAchi[j]=0.0;
            subf.subjectAchi[j]=0.0;
            fin.classAchi[j]=0.0;
        }

        fwrite(&stu,sizeof(struct StuInfo),1,ptStu);      /* 写入文件中 */
        fwrite(&subc,sizeof(struct SubjectList),1,ptClanguage);
        fwrite(&subf,sizeof(struct SubjectList),1,ptForeignLanguage);
        fwrite(&fin,sizeof(struct FinalAchiList),1,ptTotelList);

        printf("\n\t\t 录入学生的学号和姓名，输入 no 结束\n");

        printf("\t\t 输入姓名:");
        scanf("%s",stu.stuName);

        printf("\t\t 输入学号(最多 11 位):");
        scanf("%s",stu.stuID);
    }

    fclose(ptStu);                  /* 关闭所有文件 */
    fclose(ptClanguage);
    fclose(ptForeignLanguage);
    fclose(ptTotelList);

    finishedInfo();                 /* 输入字符继续，注意回车也是一个字符 */

}

/*****************************************************************
```

```
函数名:addStuInfo
函数功能:添加学生信息，追加信息，分加添加到上述四个文件中
函数输入参数:无
函数返回值:无
*****************************************************************/
void addStuInfo(void)
{
    struct StuInfo stu;
    struct SubjectList sub;
    struct FinalAchiList fin;

    FILE *ptStu;
    FILE *ptClanguage;
    FILE *ptForeignLanguage;
    FILE *ptTotelList;

    char ch;
    int i;

    printf("\n");
    printf("\t\t 进行添加操作，是否要继续(y/n):");
    ch=getch();
    if(ch!='y')
        return;

    if((ptStu=fopen("stu.dat","ab"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }

    if((ptClanguage=fopen("c.dat","ab"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }

    if((ptForeignLanguage=fopen("foreign.dat","ab"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }

    if((ptTotelList=fopen("totel.dat","ab"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
```

```
        }

        printf("\n\t\t 正在进行添加操作,输入 no 结束\n");

        printf("\t\t 请输入姓名:");
        scanf("%s",stu.stuName);

        printf("\n\t\t 请输入学号(11 位):");
        scanf("%s",stu.stuID);

        while(strcmp(stu.stuName,"no")!=0)
        {
            sub.stuInfo=stu;
            fin.stuInfo=stu;

            for(i=0;i<4;i++)
            {
                sub.subjectAchi[i]=0.0;

                fin.classAchi[i]=0.0;
            }

            fwrite(&stu,sizeof(struct StuInfo),1,ptStu);
            fwrite(&sub,sizeof(struct SubjectList),1,ptClanguage);
            fwrite(&sub,sizeof(struct SubjectList),1,ptForeignLanguage);
            fwrite(&fin,sizeof(struct FinalAchiList),1,ptTotelList);

            printf("\n\t\t 正在进行添加操作,输入 no 结束\n");
            printf("\t\t 请输入姓名:");
            scanf("%s",stu.stuName);
            printf("\n\t\t 请输入学号(最多 11 位):");
            scanf("%s",stu.stuID);
        }

        fclose(ptStu);
        fclose(ptClanguage);
        fclose(ptForeignLanguage);
        fclose(ptTotelList);

        finishedInfo();                         /* 提示已完成本次操作 */
    }

    /*************************************************************
    函数名:deleteStuInfo
    函数功能:删除学生信息，从四个文件中分别删除
    函数输入参数:无
    函数返回值:无
```

```
**************************************************************/
void deleteStuInfo(void)
{
    struct StuInfo stu[50];              /* 定义四个数固定数组把信息读入数组中再操作 */
    struct SubjectList subc[50];         /*也可以定义动态数组 */
    struct SubjectList subf[50];
    struct FinalAchiList fin[50];

    FILE *ptStu;
    FILE *ptClanguage;
    FILE *ptForeignLanguage;
    FILE *ptTotelList;

    char ch;
    int i;
    int deletePosition=-1;               /* 存储被删除学生信息位置 */
    int stuNumber=0;                     /* 学生实际人数，从 0 开始 */
    char deleteName[8];

    printf("\n");
    printf("\t\t 进行删除操作，是否要继续(y/n)");
    ch=getch();
    if(ch!='y')
        return;

    if((ptStu=fopen("stu.dat","rb+"))==NULL)     /* 以读/写方式打开文件 */
    {
        printf("\t\t 文件打不开\n");
        return;
    }
    if((ptClanguage=fopen("c.dat","rb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }
    if((ptForeignLanguage=fopen("foreign.dat","rb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }
    if((ptTotelList=fopen("totel.dat","rb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }

    /* 信息分别读入数组 */
```

```
fread(&stu[stuNumber],sizeof(struct StuInfo),1,ptStu);
fread(&subc[stuNumber],sizeof(struct SubjectList),1,ptClanguage);
fread(&subf[stuNumber],sizeof(struct SubjectList),1,ptForeignLanguage);
fread(&fin[stuNumber],sizeof(struct FinalAchiList),1,ptTotelList);
while(!feof(ptStu))
{
    stuNumber++;

    fread(&subc[stuNumber],sizeof(struct SubjectList),1,ptClanguage);
    fread(&subf[stuNumber],sizeof(structSubjectList),
        1,ptForeignLanguage);
    fread(&fin[stuNumber],sizeof(struct FinalAchiList),1,ptTotelList);
    fread(&stu[stuNumber],sizeof(struct StuInfo),1,ptStu);
}


fclose(ptStu);                                  /* 关闭文件 */
fclose(ptClanguage);
fclose(ptForeignLanguage);
fclose(ptTotelList);

printf("\n");
printf("\t\t 请输入删除学生姓名：");
scanf("%s",deleteName);

for(i=0;i<stuNumber;i++)
{
    if(strcmp(deleteName,stu[i].stuName)==0)
    {
        deletePosition=i;                       /* 记录删除学生信息位置 */
        break;
    }

}

if(deletePosition!=-1)                          /* 位置不等于-1，说明找到信息*/
{

    for(i=deletePosition;i<stuNumber-1;i++)     /* 删除操作 */
    {
        stu[i]=stu[i+1];
        subc[i]=subc[i+1];
        subf[i]=subf[i+1];
        fin[i]=fin[i+1];
    }

    if((ptStu=fopen("stu.dat","wb+"))==NULL)    /* 以写的方式打开文件 */
```

```
            {
                printf("\t\t 文件打不开\n");
                return;
            }
            if((ptClanguage=fopen("c.dat","wb+"))==NULL)
            {
                printf("\t\t 文件打不开\n");
                return;
            }
            if((ptForeignLanguage=fopen("foreign.dat","wb+"))==NULL)
            {
                printf("\t\t 文件打不开\n");
                return;
            }
            if((ptTotelList=fopen("totel.dat","wb+"))==NULL)
            {
                printf("\t\t 文件打不开\n");
                return;
            }

            /* 写入文件，每次只写一条 */
            fwrite(stu,sizeof(struct StuInfo),stuNumber-1,ptStu);
            fwrite(subc,sizeof(struct SubjectList),stuNumber-1,ptClanguage);
            fwrite(subf,sizeof(structSubjectList),
                   stuNumber-1,ptForeignLanguage);
            fwrite(fin,sizeof(struct FinalAchiList),stuNumber-1,ptTotelList);

            printf("\t\t 删除成功\n");
        }
        else
        {
            printf("\t\t 没有找到要删除的信息\n");
        }

        fclose(ptStu);
        fclose(ptClanguage);
        fclose(ptForeignLanguage);
        fclose(ptTotelList);

        finishedInfo();                                    /* 提示已完成本次操作 */

    }

    /*****************************************************************
    函数名:displayStuInfo
    函数功能:显示学生基本信息
    函数输入参数:无
```

```
函数返回值:无
*****************************************************************/
void displayStuInfo(void)
{
    struct StuInfo stu;
    FILE *ptStu;

    if((ptStu=fopen("stu.dat","rb+"))==NULL)
    {
        printf("\t\t 文件打不开\n");
        return;
    }

    printf("\n\n\t\t 学生基本信息表\n");
    printf("\n\t\t 学生姓名\t\t 学生编号\n\n");

    fread(&stu,sizeof(struct StuInfo),1,ptStu);
    while(!feof(ptStu))
    {
        printf("\t\t%s\t\t\t%s\n",stu.stuName,stu.stuID);
        fread(&stu,sizeof(struct StuInfo),1,ptStu);
    }

    fclose(ptStu);
    finishedInfo();                                    /* 提示已完成本次操作 */
}

/*****************************************************************
函数名:handleSubjectMenu
函数功能:处理科目子菜单，先选择科目，1 代表 C 语言，2 代表外语，后显示处理菜单
函数输入参数:无
函数返回值:无
*****************************************************************/
void handleSubjectMenu(void)
{
    int selectItem=0;
    int selectSubject;                                 /* 用于存储的是 C 语言不是外语科目 */

    printf("\n\n");
    printf("\n\t\t 输入科目：1.C 语言，2.外语");
    scanf("%d",&selectSubject); /* 输入选择的科目，传到相关处理函数 */

    /* 如果输入的不是 1 或 2，请重新输入 */
    while(selectSubject!=1 && selectSubject!=2)
    {
        printf("\n");
        printf("\t\t 请选择输入 1 或 2,重新输入");
```

```
        scanf("%d",&selectSubject);
    }

    do
    {
        displaySubjectMenu(selectSubject);          /* 显示菜单 */
        scanf("%d",&selectItem);                    /* 输入项目，最好把它放在显示菜单函数中 */

        switch(selectItem)
        {
            case 1:
                achievementInput(selectSubject);    /* 录入成绩 */
                break;
            case 2:
                achievementmodify(selectSubject);   /* 修改成绩 */
                break;
            case 3:
                return ;
                break;
            default:
                printf("请输入 1-3 之间的整数");
        }

    }while(1);

}

/*****************************************************************
函数名:displaySubjectMenu
函数功能:显示科目管理模块子菜单
函数输入参数:选择的科目，1 代表 C 语言，2 代表外语
函数返回值:无
*****************************************************************/
void displaySubjectMenu(int subjectCode)
{
    char *temp=NULL;

    if(subjectCode==1)
        temp="C 语言";
    else
        temp="外语";

    printf("\n\n");
    printf("\t\t～～～～～～～～～～～～\n");
    printf("\t\t%s 科目管理模块子菜单\n\n",temp);
    printf("\t\t   1. 录入成绩\n");
    printf("\t\t   2. 修改成绩\n");
```

```
    printf("\t\t   3. 返   回\n");
    printf("\t\t～～～～～～～～～～～～\n");
    printf("\t\t 请选择 1-3：");
}

/*************************************************************
函数名:achievementInput
函数功能:录入单科成绩
函数输入参数:已选择的科目代号，1 代表 C 语言，2 代表外语
函数返回值:无
*************************************************************/
void achievementInput(int subjectCode)
{
    FILE *ptSubject;

    if(subjectCode==1)
    {
        if((ptSubject=fopen("c.dat","rb+"))==NULL) /* 打开 C 语言文件 */
        {
            printf("\t\t 文件打不开\n");
            return;
        }
    }
    else
    {
        if((ptSubject=fopen("foreign.dat","rb+"))==NULL)   /* 打开外语文件 */
        {
            printf("\t\t 文件打不开\n");
            return;
        }
    }

    updateAchievement(ptSubject);                          /* 录入成绩 */

    fclose(ptSubject);

    finishedInfo();                                        /* 提示已完成本次操作 */
}

/*************************************************************
函数名:updateAchievement
函数功能:为单科目录入分数，录入所有学生的成绩
函数输入参数:文件指针，以读/写方式打开选择科目的文件
函数返回值:无
*************************************************************/
void updateAchievement(FILE *fPtr)
{
```

```
int recordNumber=0;                                    /* 记录号，为定位 */

struct SubjectList subjectInfo;

/* 文件指针定位 */
fseek(fPtr,recordNumber*sizeof(struct SubjectList),SEEK_SET);
fread(&subjectInfo,sizeof(struct SubjectList),1,fPtr);

while(!feof(fPtr))
{
    printf("\t\t 第 %d 条 信息\n",recordNumber);        /* 显示已有信息 */
    printf("\n\t 学号\t 姓名\t 平时成绩\t 期中成绩\t
           期末成绩\t 总成绩\n",recordNumber);
    printf("\t%s\t%s\t%.1f\t\t%.1f\t\t%.1f\t\t%.1f\n",
           subjectInfo.stuInfo.stuID,
           subjectInfo.stuInfo.stuName,
           subjectInfo.subjectAchi[0],
           subjectInfo.subjectAchi[1],
           subjectInfo.subjectAchi[2],
           subjectInfo.subjectAchi[3]);

    printf("\t\t 请输入成绩:\n");
    printf("\t\t 学号：%s\t 姓名：%s",
           subjectInfo.stuInfo.stuID,
           subjectInfo.stuInfo.stuName);

    printf("\n\t\t 平时成绩(输入-1 结束录入)：");
    scanf("%f",&subjectInfo.subjectAchi[0]);
    if (subjectInfo.subjectAchi[0]==-1)
      return;

    printf("\t\t 期中成绩(输入-1 结束录入)：");
    scanf("%f",&subjectInfo.subjectAchi[1]);
    if (subjectInfo.subjectAchi[1]==-1)
      return;

    printf("\t\t 期末成绩(输入-1 结束录入)：");
    scanf("%f",&subjectInfo.subjectAchi[2]);
    if (subjectInfo.subjectAchi[2]==-1)
      return;

    /* 计算成绩公式：平时成绩*20%+期中成绩*20%+期末成绩*60% */
    subjectInfo.subjectAchi[3]=subjectInfo.subjectAchi[0]*(float)0.2
                              +subjectInfo.subjectAchi[1]*(float)0.2
                              +subjectInfo.subjectAchi[2]*(float)0.6;

    fseek(fPtr,recordNumber*sizeof(struct SubjectList),SEEK_SET);
```

```
            /* 写入文件中 */
            fwrite(&subjectInfo,sizeof(struct SubjectList),1,fPtr);
            recordNumber++;

            fseek(fPtr,recordNumber*sizeof(struct SubjectList),SEEK_SET);
            fread(&subjectInfo,sizeof(struct SubjectList),1,fPtr);

        }

}

/*****************************************************************
函数名:achievementmodify
函数功能:处理单科成绩的修改
函数输入参数:已选择科目代码。1 代表 C 语言，2 代表外语
函数返回值:无
*****************************************************************/

void achievementmodify(int subjectCode)
{
    FILE *ptSubject;

    if(subjectCode==1)
    {
        if((ptSubject=fopen("c.dat","rb+"))==NULL)          /* 打开 C 语言文件 */
        {
            printf("\t\t 文件打不开\n");
            return;
        }
    }
    else
    {
        if((ptSubject=fopen("foreign.dat","rb+"))==NULL)    /* 打开外语文件 */
        {
            printf("\t\t 文件打不开\n");
            return;
        }
    }

    editAchievement(ptSubject);                             /* 修改成绩 */

    fclose(ptSubject);

    finishedInfo();                                         /* 提示已完成本次操作 */

}
```

```
/*************************************************************
函数名:editAchievement
函数功能:修改单科成绩
函数输入参数:已选择的科目的文件指针
函数返回值:无
*************************************************************/
void editAchievement(FILE *fPtr)
{
    int modifyPosition=-1;                          /* 修改位置 */
    int recordNumber=0;                             /* 总人数 */
    char modifiedName[8];                           /* 输入修改学生姓名 */

    struct SubjectList subjectInfo;                 /* 科目结构体 */

    printf("\n");
    printf("\t\t 请输入要修改的学生姓名");
    scanf("%s",modifiedName);

    fseek(fPtr,recordNumber*sizeof(struct SubjectList),SEEK_SET);
    fread(&subjectInfo,sizeof(struct SubjectList),1,fPtr);
    while(!feof(fPtr))
    {
        if((strcmp(subjectInfo.stuInfo.stuName,modifiedName)==0))
        {
            modifyPosition=recordNumber;
            break;
        }
        recordNumber++;

        fseek(fPtr,recordNumber*sizeof(struct SubjectList),SEEK_SET);
        fread(&subjectInfo,sizeof(struct SubjectList),1,fPtr);

    }

    if(modifyPosition!=-1)
    {
        printf("\t\t 第 %d 条 信息\n",recordNumber);
        printf("\n\t 学号\t 姓名\t 平时成绩\t 期中成绩\t
                期末成绩\t 总成绩\n",recordNumber);
        printf("\t%s\t%s\t%.1f\t\t%.1f\t\t%.1f\t\t%.1f\n",
                subjectInfo.stuInfo.stuID,
                subjectInfo.stuInfo.stuName,
                subjectInfo.subjectAchi[0],
                subjectInfo.subjectAchi[1],
                subjectInfo.subjectAchi[2],
                subjectInfo.subjectAchi[3]);
```

```
            printf("\n\t\t 请输入成绩:\n");
            printf("\t\t 学号：%s\t 姓名：%s",
                    subjectInfo.stuInfo.stuID,
                    subjectInfo.stuInfo.stuName);

            printf("\n\t\t 平时成绩(输入-1 结束录入)：");
            scanf("%f",&subjectInfo.subjectAchi[0]);
            if (subjectInfo.subjectAchi[0]==-1)
               return;

            printf("\t\t 期中成绩(输入-1 结束录入)：");
            scanf("%f",&subjectInfo.subjectAchi[1]);
            if (subjectInfo.subjectAchi[1]==-1)
               return;

            printf("\t\t 期末成绩(输入-1 结束录入)：");
            scanf("%f",&subjectInfo.subjectAchi[2]);
            if (subjectInfo.subjectAchi[2]==-1)
               return;

            subjectInfo.subjectAchi[3]=subjectInfo.subjectAchi[0]*(float)0.2+
                                       subjectInfo.subjectAchi[1]*(float)0.2+
                                       subjectInfo.subjectAchi[2]*(float)0.6;

            fseek(fPtr,recordNumber*sizeof(struct SubjectList),SEEK_SET);
            fwrite(&subjectInfo,sizeof(struct SubjectList),1,fPtr);
        }
        else
        {
            printf("\n\t\t 没有找到要修改的信息\n");
        }

}

/*****************************************************************
函数名:handleAchievementMenu
函数功能:成绩汇总
函数输入参数:无
函数返回值:无
*****************************************************************/
void handleAchievementMenu(void)
{
    int selectItem=0;
    struct FinalAchiList fin[50];
    int stuNumber;

    printf("\n\t\t 正在汇总，请稍等.....\n");
```

```
    stuNumber=computeAchievement(fin);                  /* 调用成绩汇总函数 */
    if(stuNumber==-1)                                   /* 返回-1 表示文件没有打开 */
      return;

    printf("\t\t 汇总完毕");

    do
    {
        displayAchievementMenu();                       /* 显示成绩汇总模块子菜单 */
        scanf("%d",&selectItem);

        switch(selectItem)
        {
            case 1:
                sortAchievement(fin,stuNumber);         /* 排序 */
                break;
            case 2:
                statisticsAchievement(fin,stuNumber);   /* 统计 */
                break;
            case 3:
                queryAchievement(fin,stuNumber);        /* 查询 */
                break;
            case 4:
                return ;
                break;
            default:
                printf("请输入 1-4 之间的整数");
        }

    }while(1);
}

/*************************************************************
函数名:displayAchievementMenu
函数功能:显示成绩汇总模块子菜单
函数输入参数:无
函数返回值:无
*************************************************************/
void displayAchievementMenu(void)
{

    printf("\n\n");
    printf("\t\t～～～～～～～～～～～～\n");
    printf("\t\t 成绩汇总模块子菜单\n\n");
    printf("\t\t   1. 排      序\n");
    printf("\t\t   2. 统      计\n");
```

```
        printf("\t\t   3. 总成绩查询\n");
        printf("\t\t   4. 返       回\n");
        printf("\t\t～～～～～～～～～～～～\n");
        printf("\t\t 请选择 1-4：");



    }

    /*************************************************************
    函数名:computeAchievement
    函数功能:成绩汇总，存入数组中，在排序、统计、查询操作中使用，在显示成绩汇总模块子菜单之前进行成绩汇总，以便每一次修改后都能及时更新
    函数输入参数:无
    函数返回值:无
    *************************************************************/
    int   computeAchievement(struct FinalAchiList fin[])
    {
        struct SubjectList subc[50];
        struct SubjectList subf[50];

        FILE *ptClanguage;
        FILE *ptForeignLanguage;
        FILE *ptTotelList;
        int stuNumber=0;

        if((ptClanguage=fopen("c.dat","rb+"))==NULL)
        {
            printf("\t\t 文件打不开\n");
            return -1;
        }
        if((ptForeignLanguage=fopen("foreign.dat","rb+"))==NULL)    /* 打开文件 */
        {
            printf("\t\t 文件打不开\n");
            return -1;
        }
        if((ptTotelList=fopen("totel.dat","wb+"))==NULL)                /*重新建一个统计文件*/
        {
            printf("\t\t 文件打不开\n");
            return -1;
        }

        fread(&subc[stuNumber],sizeof(struct SubjectList),1,ptClanguage);
        fread(&subf[stuNumber],sizeof(struct SubjectList),1,ptForeignLanguage);

        while(!feof(ptClanguage))
        {
            fin[stuNumber].stuInfo=subc[stuNumber].stuInfo;
```

```
            fin[stuNumber].classAchi[0]=subc[stuNumber].subjectAchi[3];
            fin[stuNumber].classAchi[1]=subf[stuNumber].subjectAchi[3];
            fin[stuNumber].classAchi[2]=subc[stuNumber].subjectAchi[3]
                                        +subf[stuNumber].subjectAchi[3];

            fin[stuNumber].classAchi[3]=fin[stuNumber].classAchi[2]/(float)2.0;

            stuNumber++;

            fread(&subc[stuNumber],sizeof(struct SubjectList),1,ptClanguage);
            fread(&subf[stuNumber],sizeof(struct SubjectList),
                  1,ptForeignLanguage);
        }

        fwrite(fin,sizeof(struct FinalAchiList),stuNumber,ptTotelList);

        fclose(ptClanguage);
        fclose(ptForeignLanguage);
        fclose(ptTotelList);

        return stuNumber;                                   /* 返回数组元素个数 */

    }

    /*****************************************************************
    函数名:sortAchievement
    函数功能:按每名学生的总成绩从大到小排序
    函数输入参数:总成绩数组 fin[]、班级人数 n
    函数返回值:无
    *****************************************************************/
    void sortAchievement(struct FinalAchiList fin[],int n)
    {
        int i,j;
        int p;
        struct FinalAchiList temp;
        for(i=0;i<n-1;i++)                                  /* 选择法排序 */
        {
            p=i;
            for (j=i;j<n;j++)
            {
                if(fin[p].classAchi[3]<fin[j].classAchi[3])
                {
                    p=j;
                }
            }
            if(p!=i)
            {
```

```
                temp=fin[i];
                fin[i]=fin[p];
                fin[p]=temp;
            }

        }

        displayTotelAchievement(fin,n);                    /* 显示排序后的总成绩 */


}

/*****************************************************************
函数名:statisticsAchievement
函数功能:分析、统计成绩处理
函数输入参数:存班级所有学生的总成绩数组 fin[]、班级人数 n
函数返回值:无
*****************************************************************/
void statisticsAchievement(struct FinalAchiList fin[],int n)
{
    int cf[11];                                            /* 存储各分数段人数 */

    printf("\n\t\tC 语言分数段人数：");
    totelGradeNumber(fin,cf, n, 0);                        /* 调用 C 语言分数段统计函数 */
    showGradeNumber(cf);                                   /* 调用显示函数 */

    printf("\n\n\t\t 外语分数段人数：");
    totelGradeNumber(fin,cf, n, 1);                        /* 调用外语分数段统计函数 */
    showGradeNumber(cf);                                   /* 调用显示函数 */

    finishedInfo();                                        /* 提示已完成本次操作 */

}

/*****************************************************************
函数名:totelGradeNumber
函数功能:统计单科各分数段人数
函数输入参数:学生总成绩数组、分数段数组、班级人数，科目代码
函数返回值:无
*****************************************************************/
void totelGradeNumber(struct FinalAchiList fin[],int cfLangueTotel[],int n,int m)
{
    int i;
    int temp;

    for(i=0;i<11;i++)                                      /* 初始化数组 */
    {
```

```
            cfLangueTotel[i]=0;
        }

        for(i=0;i<n;i++)                                    /* 统计分数段人数 */
        {
            temp=(int)fin[i].classAchi[m]/10;
            cfLangueTotel[temp]=cfLangueTotel[temp]+1;
        }
    }

    /*****************************************************************
    函数名:showGradeNumber
    函数功能:显示单科统计结果
    函数输入参数:单科统计数组
    函数返回值:无
    *****************************************************************/
    void showGradeNumber(int cfLangueTotel[])
    {
        int i;

        for(i=0;i<11;i++)                                   /* 显示分数段人数 */
        {

            if(i!=10)
            {
                printf("\n\t\t%d  分~%d 分数段人数:%d",
                        (i*10),(9+i*10),cfLangueTotel[i]);
            }
            else
            {
                printf("\n\t\t%d  分~100 分数段人数:%d",
                        (i*10),cfLangueTotel[i]);
            }
        }


    }
    /*****************************************************************
    函数名:queryAchievement
    函数功能:依据个人姓名，查询汇总后的总成绩
    函数输入参数:学生成绩数组、班级人数
    函数返回值:无
    *****************************************************************/
    void queryAchievement(struct FinalAchiList fin[],int n)
    {
        int i,sFlag=0;
        char queryName[8];
```

```
    printf("\n\t\t 请输入要查的学生的姓名：");
    scanf("%s",queryName);

    printf("\n\n");
    for(i=0;i<n;i++)
    {
        if(strcmp(fin[i].stuInfo.stuName,queryName)==0)
        {
            printf("\t 学号\t 姓名\tC 语言成绩\t 外语成绩\t
                总成绩\t 平均成绩\n");
            printf("\t%s\t%s\t%.1f\t\t%.1f\t\t%.1f\t%.1f\n",
                fin[i].stuInfo.stuID,
                fin[i].stuInfo.stuName,
                fin[i].classAchi[0],
                fin[i].classAchi[1],
                fin[i].classAchi[2],
                fin[i].classAchi[3]);
            sFlag=1;
        }
    }

    if(sFlag==0) printf ("\t\t 没有此人");
    finishedInfo();                                    /* 提示已完成本次操作 */

}
/*****************************************************************
函数名:displayTotelAchievement
函数功能:显示总成绩单
函数输入参数:学生总成绩数组、班级人数
函数返回值:无
*****************************************************************/
void displayTotelAchievement(struct FinalAchiList fin[],int n)
{

    int i;
    printf("\n\n\t 学号\t 姓名\tC 语言成绩\t 外语成绩\t 总成绩\t 平均成绩\n");
    for (i=0;i<n;i++)
    {
     printf("\t%s\t%s\t%.1f\t\t%.1f\t\t%.1f\t%.1f\n",
            fin[i].stuInfo.stuID,
            fin[i].stuInfo.stuName,
            fin[i].classAchi[0],
            fin[i].classAchi[1],
            fin[i].classAchi[2],
            fin[i].classAchi[3]);
    }
```

```
    finishedInfo();                                    /* 提示已完成本次操作 */

}

/*****************************************************************
函数名:finishedInfo
函数功能:提示用户，已完成本次操作，按任意键继续
函数输入参数:无
函数返回值:无
*****************************************************************/
void   finishedInfo(void)
{
    char ch;
    printf("\n");
    printf("\t\t 本次操作完毕，按任意键返回");
    ch=getch();
}
```

4.6 测试

测试数据如表 4.3、表 4.4、表 4.5 所示。

4.7 设计说明

4.7.1 设计内容的综合性

此项目设计涉及的知识和技术比较广泛，其中包括软件开发的基本方法，基本数据类型、构造数据类型、指针、空类型；函数的分析设计方法等。此项目为综合实训项目，为了充分利用以前学习过的知识和技能，我们针对不同的模块所采用的技术也不尽相同。例如，“空白成绩单管理”模块采用文件操作的方式是每一次都直接对文件进行操作，这增加了读/写文件次数，效率不高；而在“汇总成绩单管理”模块中，除了“汇总”子模块需要对文件进行写操作外，其他子模块都对文件进行读操作，因此建立了一个数组，将数据存入数组中，通过对数组操作实现数据处理来提高效率。

4.7.2 分析方法

分析方法采用的是结构化软件设计方法，符合自顶向下、逐步求精的策略，根据数据流变化进行分析和设计功能模块，最终通过函数来实现各功能模块的功能。该系统在实现过程中，函数部分始终遵循“三步骤”原则，即从分析函数、设计函数到实现函数。

4.7.3 测试

因为一个应用程序在编写过程中难免会有各种各样的错误，所以对程序进行测试是必要的。测试方法有白盒测试和黑盒测试。白盒测试是结构测试，就是对源码进行测试，检查程

序是否准确运行；黑盒测试就是功能测试，把程序看成一个黑盒子，不需要测试源码，只通过运行可执行程序进行测试，检查是否完成预期功能。测试时往往结合两种方法以达到测试目的。

4.7.4 待完善的问题

本项目采用的技术也不是很复杂。有些技术使用得并不是特别合理。这样设计目的是，使初学者易于接受。比如数组的使用，我们采用的是静态数组。合理的技术是使用动态数组或链表，原因是当成绩管理的人数不确定、学生的科目数量发生变化时能够动态管理。再如菜单的设计，本项目中采用字符界面的菜单，为了提高美观度、用户满意度，也可以通过使用 C 语言中的绘图技术或 Windows Api 函数设计图形界面的菜单。读者可以参考有关资料，自行设计。

从功能角度来看，相对简单，本项目设计只是一个班的成绩管理，也可以实现多个班级的管理。读者可以根据自己学院的实际情况自行设计，以便更有针对性。

附录A　常用字符与ASCII码对照表

ASCII 值	字　符	ASCII 值	字　符	ASCII 值	字　符	ASCII 值	字　符
000	NULL	032	SP	064	@	096	'
001	SOH	033	!	065	A	097	a
002	STX	034	"	066	B	098	b
003	ETX	035	#	067	C	099	c
004	EOT	036	$	068	D	100	d
005	ENQ	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	'	071	G	103	g
008	BS	040	(	072	H	104	h
009	HT	041	)	073	I	105	i
010	LF	042	*	074	J	106	j
011	VT	043	+	075	K	107	k
012	FF	044	,	076	L	108	l
013	CR	045	-	077	M	109	m
014	SO	046	.	078	N	110	n
015	SI	047	/	079	O	111	o
016	DLE	048	0	080	P	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	s
020	DC4	052	4	084	T	116	t
021	NAK	053	5	085	U	117	u
022	SYN	054	6	086	V	118	v
023	ETB	055	7	087	W	119	w
024	CAN	056	8	088	X	120	x
025	EM	057	9	089	Y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	{
028	FS	060	<	092	\	124	|
029	GS	061	=	093	]	125	}
030	RS	062	>	094	^	126	~
031	US	063	?	095	_	127	

附录B　C语言运算符的优先级与结合方向

优　先　级	运　算　符	功　　能	要求运算量的个数	结 合 方 向
1	()	提高运算优先级		自左至右
	[]	下标运算符		
	–>	指向结构体的成员		
	.	取结构体成员		
2	!	逻辑非	1（单目运算符）	自右至左
	~	按位取反		
	++	自增		
	--	自减		
	(类型符)	强制类型转换		
	*	间接访问		
	&	取地址		
	size of	测试数据字节数		
3	*	乘法	2（双目运算符）	自左至右
	/	除法		
	%	求余数运算		
4	+	加法	2（双目运算符）	自左至右
	-	减法		
5	<<	左移位	2（双目运算符）	自左至右
	>>	右移位		
6	<	小于	2（双目运算符）	自左至右
	>	大于		
	<=	小于等于		
	>=	大于等于		
7	= =	等于	2（双目运算符）	自左至右
	!=	不等于		
8	&	按位与	2（双目运算符）	自左至右
9	^	按位异或	2（双目运算符）	自左至右
10	|	按位或	2（双目运算符）	自左至右
11	&&	逻辑与	2（双目运算符）	自左至右
12	||	逻辑或	2（双目运算符）	自左至右
13	?:	条件运算	3（3 目运算符）	自右至左
14	= += -= *= /= %= ^= |= &= >>= <<=	赋值运算	2（双目运算符）	自右至左
15	,	逗号运算		自左至右

附录C　C语言常用库函数

1．数学函数

使用数学函数时，应该在源文件中使用预编命令：

```
#include <math.h> 或 #include "math.h"
```

数学函数如表 C1 所示。

表 C1　数学函数

函 数 名	函 数 原 型	功　　能	返 回 值
acos	double acos(double x);	计算 arccos *x* 的值，其中−1≤*x*≤1	计算结果
asin	double asin(double x);	计算 arcsin *x* 的值，其中−1≤*x*≤1	计算结果
atan	double atan(double x);	计算 arctan *x* 的值	计算结果
atan2	double atan2(double x,double y);	计算 arctan *x*/*y* 的值	计算结果
cos	double cos(double x);	计算 cos *x* 的值，其中 *x* 的单位为弧度	计算结果
cosh	double cosh(double x);	计算 *x* 的双曲余弦 cosh *x* 的值	计算结果
exp	double exp(double x);	求 e^x 的值	计算结果
fabs	double fabs(double x);	求 *x* 的绝对值	计算结果
floor	double floor(double x);	求出不大于 *x* 的最大整数	该整数的双精度实数
fmod	double fmod(double x,double y);	求整除 *x*/*y* 的余数	余数的双精度实数
frexp	Double frexp(double val,int*eptr);	把双精度数 val 分解成数字部分和以 2 为底的指数，即 val=$x\times2^n$，*n* 存放在 eptr 指向的变量中	数字部分 x 0.5<=x<1
log	double log(double x);	求 ln*x* 的值	计算结果
log10	double log10(double x);	求 $\log_{10}x$ 的值	计算结果
modf	double modf(double val,int*iptr);	把双精度数 val 分解成数字部分和小数部分，把把整数部分存放在 ptr 指向的变量中	val 的小数部分
pow	double pow(double x,double y);	求 x^y 的值	计算结果
sin	double sin(double x);	求 sin *x* 的值，其中 *x* 的单位为弧度	计算结果
sinh	double sinh(double x);	计算 *x* 的双曲正函数 sinh *x* 的值	计算结果
sqrt	double sqrt(double x);	计算 *x* 的平方根，其中 *x* 的单位为弧度	计算结果
tan	double tan(double x);	计算 tan *x* 的值，其中 *x* 的单位为弧度	计算结果
tanh	double tanh(double x);	计算 *x* 的双曲正切函数 tanh *x* 的值	计算结果

2．字符函数

使用字符函数时，应该在源文件中使用预编译命令：

```
#include <ctype.h> 或 #include"ctype.h"
```

字符函数如表 C2 所示。

表 C2　字符函数

函　数　名	函 数 原 型	功　　能	返　回　值
isalnum	int isalnum(int ch);	检查 ch 是否为字母或数字	是字母或数字返回 1，否则返回 0
isalpha	int isalpha(int ch);	检查 ch 是否为字母	是字母返回 1，否则返回 0
iscntrl	int iscntrl(int ch);	检查 ch 是否为控制字符（其 ASCII 码在 0 和 0xlF 之间）	是控制字符返回 1，否则返回 0
isdigit	int isdigit(int ch);	检查 ch 是否为数字	是数字返回 1，否则返回 0
isgraph	int isgraph(int ch);	检查 ch 是否为除空格外的可打印字符（不含空格），其 ASCII 码在 0x21 和 0x7e 之间	是可打印字符返回 1，否则返回 0
islower	int islower(int ch);	检查 ch 是否是小写字母（a～z）	是小字母返回 1，否则返回 0
isprint	int isprint(int ch);	检查 ch 是否是可打印字符（含空格），其 ASCII 码在 0x20 和 0x7e 之间），不包括空格	是可打印字符返回 1，否则返回 0
ispunct	int ispunct(int ch);	检查 ch 是否是标点字符（不包括空格），即除字母、数字和空格以外的所有可打印字符	是标点返回 1，否则返回 0
isspace	int isspace(int ch);	检查 ch 是否是空格、跳格符（制表符）或换行符	是则返回 1，否则返回 0
isupper	int isupper(int ch);	检查 ch 是否是大写字母（A～Z）	是大写字母返回 1，否则返回 0
isxdigit	int isxdigit(int ch);	检查 ch 是否是一个十六进制数字（即 0～9 或 A～F，a～f）	是则返回 1，否则返回 0
tolower	int tolower(int ch);	将 ch 字符转换为小写字母	返回 ch 对应的小写字母
toupper	int toupper(int ch);	将 ch 字符转换为大写字母	返回 ch 对应的大写字母

3．字符串函数

使用字符串函数时，应该在源文件中使用预编译命令：

```
#include <string.h> 或 #include"string.h"
```

字符串函数如表 C3 所示。

表 C3　字符串函数

函数名	函 数 原 型	功　　能	返　回　值
memchr	void memchr(void *buf,char ch,unsigned count);	在 buf 的前 count 个字符里搜索字符 ch 首次出现的位置	返回指向 buf 中 ch 的第一次出现的位置指针。 若没有找到 ch，则返回 NULL

续表

函 数 名	函 数 原 型	功 能	返 回 值
memcmp	int memcmp(void *bufl,void *buf2,unsigned count);	按字典顺序比较由 buf1 和 buf2 指向的数组的前 count 个字符	buf1<buf2，为负数 buf1=buf2，返回 0 buf1>buf2，为正数
memcpy	void *memcpy(void *to,void *from,unsigned count);	将 from 指向的数组中的前 count 个字符复制到 to 指向的数组中。from 和 to 指向的数组不允许重叠	返回指向 to 的指针
memove	void *memove(void *to,void *from,unsigned count);	将 from 指向的数组中的前 count 个字符移动到 to 指向的数组中。from 和 to 指向的数组不允许重叠	返回指向 to 的指针
memset	void *memset(void *buf,char ch,unsigned count);	将字符 ch 复制到 buf 指向的数组前 count 个字符中	返回 buf
strcat	char *strcat(char *strl,char *str2);	把字符 str2 接到 str1 后面，取消原来 str1 最后面的串结束符"\0"	返回 strl
strchr	char *strchr(char *str,int ch);	找出 str 指向的字符串中第一次出现字符 ch 的位置	返回指向该位置的指针。若找不到，则返回 NULL
strcmp	int *strcmp(char *strl,char *str2);	比较字符串 str1 和 str2	strl<str2，为负数 str1=str2，返回 0 str1>str2，为正数
strcpy	char *strcpy(char *strl,char *str2);	把 str2 指向的字符串复制到 str1 中	返回 str1
strlen	unsigned int strlen(char *str);	统计字符串 str 中字符的个数（不包括终止符"\0"）	返回字符个数
strncat	char *strncat(char *strl,char *str2,unsigned count);	把字符串 str2 指向的字符串中最多 count 个字符连到字符串 str1 后面，并以 NULL 结尾	返回 str1
strncmp	int strncmp(char *strl, *str2,unsigned count);	比较字符串 str1 和 str2 中最多前 count 个字符	strl<str2，为负数 str1=str2，返回 0 str1>str2，为正数
strncpy	char *strncpy(char *strl, *str2,unsigned count);	把 str2 指向的字符串中最多前 count 个字符复制到字符串 strl 中	返回 str1
strnset	void *strnset(char *buf,char ch,unsigned count);	将字符 ch 复制到 buf 指向的数组前 count 个字符中	返回 buf

续表

函 数 名	函 数 原 型	功 能	返 回 值
strset	void *strset(void *buf,char ch);	将 buf 所指向的字符串中的全部字符都变为字符 ch	返回 buf
strstr	char *strstr(char *strl,*str2);	寻找 str2 指向的字符串在 str1 指向的字符串中首次出现的位置	返回 str2 指向的字符串首次出现的地址。若没有找到，则返回 NULL

4．输入/输出函数

使用输入/输出函数时，应该在源文件中使用预编译命令：

```
#include <stdio.h>  或  #include"stdio.h"
```

输入/输出函数如表 C4 所示。

表 C4　输入/输出函数

函 数 名	函 数 原 型	功 能	返 回 值
clearerr	void clearerr(FILE*fp);	清除文件指针错误指示器	无
close	int close(int fp);	关闭文件（非 ANSI 标准）	关闭成功，返回 0；不成功，返回-1
creat	int creat(char*filename,int mode);	以 mode 所指定的方式建立文件（非 ANSI 标准）	成功，返回正数；否则返回-1
eof	int eof(int fp);	判断 fp 所指的文件是否结束	文件结束，返回 1；否则返回-1
fclose	int fclose(FILE*fp);	关闭 fp 所指的文件，释放文件缓冲区	关闭成功返回 0，不成功返回非 0
feof	int feof(FILE*fp);	检查文件是否结束	文件结束返回非 0，否则返回 0
ferror	int ferror(FILE*fp);	测试 fp 所指的文件是否有错误	无错返回 0，否则返回非 0
fflush	int fflush(FILE*fp);	将 fp 所指的文件的全部控制信息和数据存盘	存盘正确返回 0，否则返回非 0
fgets	char *fgets(char *buf,int n,FILE *fp);	从 fp 所指的文件读取一个长度为（n−1）的字符串，存入起始地址为 buf 的空间	返回地址 buf。若遇文件结束或出错，则返回 EOF
fgetc	int fgetc(FILE*fp);	从 fp 所指的文件中取得下一个字符	返回所得到的字符。出错，返回 EOF
fopen	FILE *fopen(char *filename,char *mode);	以 mode 所指定的方式打开名为 filename 的文件	成功，返回一个文件指针；否则返回 0
fprintf	int fprintf(FILE *fp,char *format,args,…);	把 args 的值以 format 指定的格式输出到 fp 所指的文件中	实际输出的字符数

续表

函 数 名	函 数 原 型	功 能	返 回 值
fputc	int fputc(char ch,FILE*fp);	将字符 ch 输出到 fp 所指的文件中	成功，返回该字符；出错，返回 EOF
fputs	int fputs(char str,FILE*fp);	将 str 指定的字符串输出到 fp 所指的文件中	成功，返回 0；出错，返回 EOF
fread	int fread(char *pt,unsigned size,unsigned n,FILE *fp);	从 fp 所指文件中读取长度为 size 的 *n* 个数据项，存到 pt 所指向的内存区	返回所读的数据项个数。若文件结束或出错，则返回 0
fscanf	int fscanf(FILE *fp,char *format,args,…);	从 fp 指定的文件中按给定的 format 格式将读入的数据送到 args 所指向的内存变量中（args 是指针）	返回已输入的数据个数
fseek	int fseek(FILE*fp,long offset,int base);	将 fp 指定的文件的位置指针移到以 base 所指出的位置为基准，以 offset 为位移量的位置	返回当前位置，否则返回-1
ftell	long ftell(FILE *fp);	返回 fp 所指向的文件中的读/写位置	返回文件中的读/写位置，出错返回-1
fwrite	int fwrite(char *ptr,unsigned size,unsigned n,FILE *fp);	把 ptr 所指向的 n*size 个字节输出到 fp 所指向的文件中	写到 fp 文件中的数据项的个数
getc	int getc(FILE *fp);	从 fp 所指向的文件中读出下一个字符	返回读出的字符。若文件出错或结束，则返回 EOF
getchar	int getchar();	从标准输入设备中读取下一个字符	返回字符。若文件出错或结束，则返回-1
gets	char *gets(char *str);	从标准输入设备中读取字符串存入 str 指向的数组	成功返回 str，否则返回 NULL
open	int open(char *filename,int mode);	以 mode 指定的方式打开已存在的名为 filename 的文件（非 ANSI 标准）	返回文件号（正数），若打开失败，则返回-1
printf	int printf(char *format,args,…);	在 format 指定的字符串的控制下，将输出列表 ages 的值输出到标准设备	输出字符的个数。若出错，则返回负数
prtc	int prtc(int ch,FILE *fp);	把一个字符 ch 输出到 fp 所指的文件中	输出字符 ch。若出错，则返回 EOF
putchar	int putchar(char ch);	把字符 ch 输出到 fp 标准输出设备中	返回换行符。若失败，则返回 EOF
puts	int puts(char *str);	把 str 指向的字符串输出到标准输出设备中，将“\0”转化为回车行	返回换行符。若失败，则返回 EOF

续表

函 数 名	函 数 原 型	功 能	返 回 值
putw	int putw(int w,FILE *fp);	将一个整数 i（即一个字）写到 fp 所指定的文件中(非 ANSI 标准)	返回读出的字符。若文件出错或文件结束，则返回 EOF
read	int read(int fd,char *buf,unsigned count);	从文件号 fp 所指定文件中读 count 个字节到由 buf 指示的缓冲区（非 ANSI）标准	返回真正读出的字节个数。若文件结束，则返回 0；出错，则返回-1
remove	int remove(char *fname);	删除以 fname 为文件名的文件	成功，返回 0；出错，返回-1
rename	int rename(char *oname, char *nname);	把 oname 所指定的文件名改为由 nname 所指的文件名	成功，返回 0；出错，返回-1
rewind	void rewind(FILE *fp);	将 fp 指定的文件指针置于文件头，并清除文件结束标志和错误标志	无
scanf	int scanf(char *format,args,…);	从标准输入设备按 format 指示的格式字符串规定的格式，输入数据给 args 所指示的单元。args 为指针	读入并赋给 args 数据个数。若文件结束，则返回 EOF；若出错，则返回 0
write	int write(int fd,char *buf,unsigned count);	从 buf 指示的缓冲区输出 count 个字符到 fd 所指的文件中（非 ANSI 标准）	返回实际写入的字节数。若出错，则返回-1

5．动态存储分配函数

使用动态存储分配函数时，应该在源文件中使用预编译命令：

```
#include <stdio.h> 或 #include"stdio.h"
```

动态存储分配函数如表 C5 所示。

表 C5　动态存储分配函数

函 数 名	函 数 原 型	功 能	返 回 值
calloc	void *calloc(unsigned n,unsigned size);	分配 *n* 个数据项的内存连续空间，每个数据项的大小为 size	分配内存单元的起始地址。若不成功，返回 0
free	void free(void *p);	释放 p 所指内存区	无
malloc	void *malloc(unsigned size);	分配 size 字节的内存区	所分配的内存区地址。若内存不够，返回 0
realloc	void *realloc(void *p,unsigned size);	将 p 所指的以分配的内存区的大小改为 size。size 可以比原来分配的空间大或小	返回指向该内存区的指针。若重新分配失效，返回 NULL

6．其他函数

使用其他函数时，应该在源文件中使用预编译命令：

```
#include <stdlib.h> 或 #include"stdlib.h"
```

其他函数如表 C6 所示。

表 C6　其他函数

函　数　名	函 数 原 型	功　　能	返　回　值
abs	int abs(int num);	计算整数 num 的绝对值	返回计算结果
atof	double atof(char *str);	将 str 指向的字符串转换为一个 double 型的值	返回双精度计算结果
atoi	int atoi(char *str);	将 str 指向的字符串转换为一个 int 型的值	返回转换结果
atol	long atol(char *str);	将 str 指向的字符串转换为一个 long 型的值	返回转换结果
exit	void exit(int status);	中止程序运行。将 status 的值返回调用的过程	无
itoa	char *itoa(int n,char *str,int radix);	将整数 *n* 的值按照 radix 进制转换为等价的字符串，并将结果存入 str 指向的字符串中	返回一个指向 str 的指针
labs	long labs(long num);	计算 long 型整数 num 的绝对值	返回计算结果
ltoa	char *ltoa(long n,char *str,int radix);	将长整数 *n* 的值按照 radix 进制转换为等价的字符串，并将结果存入 str 指向的字符串	返回一个指向 str 的指针
rand	int rand();	产生 0 到 RAND_MAX 之间的伪随机数。RAND_MAX 在头文件中定义	返回一个伪随机（整）数
random	int random(int num);	产生 0 到 num 之间的随机数	返回一个随机（整）数
randomize	void randomize();	初始化随机函数，使用时包括头文件 time.h	

附录D　VC++ 6.0 常用菜单功能说明

VC++ 6.0 共有 9 个菜单：File、Edit、View、Insert、Project、Build、Tools、Window、Help。每个菜单都有下拉菜单，用鼠标单击菜单项可弹出其下拉菜单，下拉菜单中的每个菜单项执行不同的功能。下面对各菜单项进行详细介绍。

1．File菜单

File 菜单如表 D1 所示。

表 D1　File 菜单

菜　单　项	快　捷　键	功 能 说 明
New	Ctrl+N	创建一个新的文件、项目或工作区
Open	Ctrl+O	打开一个已存在的文件
Close	—	关闭当前打开的文件
Open Workspace	—	打开一个已存在的工作区
Save Workspace	—	保存当前打开的工作区
Close Workspace	—	关闭当前打开的工作区
Save	Ctrl+S	保存当前打开的文件
Save As	—	将当前文件另存为一个新的文件
Save All	—	保存所有打开文件
Page Setup	—	对页面的布局进行设置
Print	Ctrl+P	打印当前打开的文件
Recent Files	—	最近使用的文件列表
Recent Workspaces	—	最近使用的工作区列表
Exit	—	退出集成开发环境

2．Edit菜单

Edit 菜单如表 D2 所示。

表 D2　Edit 菜单

菜　单　项	快　捷　键	功 能 说 明
Undo	Ctrl+Z	撤销上一次的操作
Redo	Ctrl+Y	恢复被撤销的操作
Cut	Ctrl+X	将所选内容剪切至剪贴板中
Copy	Ctrl+C	将所选内容复制到剪贴板中
Paste	Ctrl+V	将当前剪贴板中的内容粘贴到当前插入点
Delete	Del	删去所选内容
Select All	Ctrl+A	选定当前窗口中的全部内容

续表

菜 单 项		快 捷 键	功 能 说 明
Find		Ctrl+F	查找指定的字符串
Find in Files		—	在多个文件中查找指定字符串
Replace		Ctrl+H	替换指定字符串
Go To		Ctrl+G	光标自动转移到指定位置
Bookmarks		Ctrl+F2	设置书签或书签导航
Advanced	Incremental Search	Ctrl+I	开始向前搜索
	Format Selection	Ctrl+F8	对选中对象进行快速缩排
	Tabify Selection	—	在选中对象中用制表位替代空格
	Untabify Selection	—	在选中对象中用空格替代制表符
	Make Selection Uppercase	Ctrl+Shift+U	把选中部分改成大写
	Make Selection Lowercase	Ctrl+U	把选中部分改成小写
	a-b View Whitespace	Ctrl+Shift+8	显示或隐藏空格点
Breakpoints		Alt+F9	编辑程序中的断点
List Member		Ctrl+Alt+T	显示出全部关键字
Type Info		Ctrl+T	显示变量、函数或方法的语句
Parameter Info		Ctrl+Shift+Space	显示函数的参数
Complete Word		Ctrl+Space	给出相关关键字的全称

3．View菜单

View 菜单如表 D3 所示。

表 D3　View 菜单

菜 单 项		快 捷 键	功 能 说 明
ClassWizard		Ctrl+W	编辑应用程序的类
Resource Symbols		—	浏览和编辑资源文件中的资源标志符（ID 号）
Resource Includes		—	编辑和修改资源文件名及预处理命令
Full Screen		—	切换到全屏幕显示方式
Workspace		Alt+O	激活项目工作区窗口
Output		Alt+2	激活输出窗口
Debug Windows	Watch	Alt+3	激活监视窗口
	Call Stack	Alt+7	激活调用栈窗口
	Memory	Alt+6	激活内存窗口
	Variables	Alt+4	激活变量窗口
	Registers	Alt+5	激活寄存器窗口
	Disassembly	Alt+8	激活反汇编窗口
Refresh		—	更新选中区域
Properties		Alt+Enter	打开源文件属性窗口

4．Insert菜单

Insert 菜单如表 D4 所示。

表 D4　Insert 菜单

菜　单　项	快　捷　键	功 能 说 明
New Class	—	在项目中添加一个新类
New Form	—	在项目中添加一个新表单
Resource	Ctrl+R	创建各种新资源
Resource Copy	—	对选定的资源进行复制
File As Text	—	将一个已存在的文件插入到当前焦点中
New Atl Object	—	在项目中添加一个新的 Atl 对象

5．Project菜单

Project 菜单如表 D5 所示。

表 D5　Project 菜单

菜　单　项		快　捷　键	功 能 说 明
Set Active Project		—	选定指定项目为当前工作区中的活动项目
Add To Project	New	—	在项目中增加文件
	New Folder	—	在项目中增加新文件夹
	Files	—	在项目中插入已存在的文件
	Date Connection	—	在当前项目中增加数据连接
	Components and Controls	—	在当前项目中插入一个部件或 ActiveX 控件
Dependencies		—	编辑项目组件
settings		Alt+F7	编译及调试的设置
Export Makefile		—	以制作文件（.mak）形式输出可编译项目
Insert Project into Workspace		—	将项目插入到项目工作区窗口中

6．Build菜单

Build 菜单如表 D6 所示。

表 D6　Build 菜单

菜　单　项	快　捷　键	功 能 说 明
Compile	Ctrl+F7	编译当前编辑窗口中打开的文件
Build	F7	生成一个可执行文件，即编译一个项目
ReBuild All	—	编译和连接多个项目文件
Batch Build	—	一次编译和连接多个项目文件
Clean	—	删除当前项目中所有中间文件及输出文件
Start Debug	F5	开始或继续调试程序
	F11	单步运行调试
	Ctrl+F10	运行程序到光标所在处
	—	连接正在运行的进程
Debugger Remote Connection	—	编辑远程调试连接设置

续表

菜 单 项	快 捷 键	功 能 说 明
Excute	Ctrl+F5	运行可执行文件
Set Active Configuration	—	选择激活的项目及配置
Configurations	—	编辑项目配置
Profile	—	选中该菜单项，用户可以检查代码的执行情况

7．Tools菜单

Tools 菜单如表 D7 所示。

表 D7　Tools 菜单

菜 单 项	快 捷 键	功 能 说 明
Source Browser	Alt+F12	浏览对指定对象的查询及相关信息
Close Source Browser File	—	关闭信息浏览文件
Visual Component Manager	—	激活组件管理器
Register Control	—	激活注册控件
Error Lookup	—	激活错误查找器
ActiveX Control Text Container	—	激活 ActiveX 控件测试器
OLE/COM Object Viewer	—	激活 OLE/COM 对象查看器
Spy++	—	激活 Spy++工具包
MFC Tracer	—	激活 MFC 跟踪器
Customize	—	定制 Tools 菜单和工具栏
Options	—	改变集成开发环境的各项设置
Macro	—	创建和编辑宏
Record Quick Macro	Ctrl+Shift+R	记录宏
Play Quick Macro	Ctrl+Shift+P	运行宏

8．Window菜单

Window 菜单如表 D8 所示。

表 D8　Window 菜单

菜 单 项	快 捷 键	功 能 说 明
New Window	—	为当前文档打开另一个窗口
Split	—	将窗口拆分为多个窗口
Docking View	Alt+F6	启动或关闭 Docking View 模式
Close	—	关闭当前窗口
Close All	—	关闭所有打开的窗口
Next	—	激活下一个窗口
Previous	—	激活上一个窗口
Cascade	—	将工作区中所有打开的窗口重叠排列
Tile Horizontally	—	将工作区中所有打开的窗口按照纵向平铺
Tile Vertically	—	将工作区中所有打开的窗口按照横向平铺
Windows	—	管理当前打开的窗口

9．Help菜单

Help 菜单如表 D9 所示。

表 D9　Help 菜单

菜　单　项	快　捷　键	功 能 说 明
Contents	—	显示所有帮助信息的内容列表
Search	—	利用在线查询获得帮助信息
Index	—	显示在线文件的索引
Use Extension Help	—	打开或关闭 Extension Help
Keyboard Map	—	显示所有键盘命令
Tip of the Day	—	显示 Tip of the Day
Technical Support	—	显示 Visual Studio 的支持信息
Microsoft the Web	—	有关 Microsoft 的网站或网页
About Visual C++	—	显示版本的有关信息

附录E　scanf、printf函数格式字符表

1．scanf函数格式字符说明

scanf 函数格式字符说明如表 E1 所示。

表 E1　scanf 函数格式字符说明

格 式 字 符	功　　能
d	输入十进制整数
o	输入八进制整数
x	输入十六进制整数
c	输入单个字符
s	输入字符串
f	输入浮点数（小数或指数形式）
e	输入浮点数（指数形式）
hd，ho，hx	输入短整型数据
ld，lo，lx	输入长整型数据
lf，le	输入长浮点型数据（双精度）

2．printf函数格式字符说明

printf 函数格式字符说明如表 E2 所示。

表 E2　printf 函数格式字符说明

格 式 字 符	功　　能
d	按十进制形式输出带符号的整数（正数前无+号）
o	按八进制形式无符号输出（无前导 0）
x	按十六进制形式无符号输出（无前导 0x）
u	按十进制形式无符号输出
c	按字符形式输出一个字符
f	按十进制形式输出单、双精度浮点数（默认 6 位小数）
e	按指数形式输出单、双精度浮点数
s	输出以“\0”结尾的字符串
ld	长整型输出
lo	长八进制整型输出
lx	长十六进制整型输出
lu	按无符号长整型输出
m	按宽度 m 输出，右对齐
-m	按宽度 m 输出，左对齐
m.n	按宽度 m，n 位小数，或截取字符串前 N 个字符输出，右对齐
-m.n	按宽度 m，n 位小数，或截取字符串前 N 个字符输出，左对齐

参 考 文 献

[1]（美）肯格著，朱剑平等译．软件开发：编程与设计（C 语言版）．北京：清华大学出版社，2006．
[2] 谭浩强．C 程序设计（第 3 版）．北京：清华大学出版社，2005．
[3] 何勤．轻松学习 C 程序设计．北京：中国电力出版社，2008．
[4] 姜灵芝，余键．C 语言程序设计案例精编．北京：清华大学出版社，2008．
[5] 于文强，毛慧凤等．C/C++程序设计教程上机实训．北京：中国铁道出版社，2007．

反侵权盗版声明

《软件设计与编程基础（C 语言版）》读者意见反馈表

尊敬的读者：

感谢您购买本书。为了能为您提供更优秀的教材，请您抽出宝贵的时间，将您的意见以下表的方式（可从 http://www.huaxin.edu.cn 下载本调查表）及时告知我们，以改进我们的服务。对采用您的意见进行修订的教材，我们将在该书的前言中进行说明并赠送您样书。

姓名：_____ 电话：_____

职业：_____ E-mail：_____

邮编：_____ 通信地址：_____

1．您对本书的总体看法是：

□很满意 □比较满意 □尚可 □不太满意 □不满意

2．您对本书的结构（章节）：□满意 □不满意 改进意见_____

_____

_____

3. 您对本书的例题： □满意 □不满意 改进意见_____

_____

_____

4．您对本书的习题： □满意 □不满意 改进意见_____

_____

_____

5．您对本书的实训： □满意 □不满意 改进意见_____

_____

_____

6． 您对本书其他的改进意见：

_____

_____

_____

_____

7．您感兴趣或希望增加的教材选题是：

_____

_____

_____

请寄：100036 北京市万寿路 173 信箱高等职业教育分社 收

电话：010-88254565 E-mail：gaozhi@phei.com.cn