

高等学校教材·计算机科学与技术

C 语言程序设计教程习题解答与实验指导

王敬华 林 萍 张 维 编著

清华大学出版社

北 京

内 容 简 介

本书是《C 语言程序设计教程》的配套教学用书。第 1 章包括主教材中全部习题及其详细解答,涵盖了计算机程序设计语言考试的主要题型(选择题、程序填空题、阅读程序写运行结果题和编程题等),综合运用数据类型、程序结构和典型算法。对每道习题不仅仅只是给出参考答案,而且还给出了详细的解释。第 2 章包括主教材中各章节思考题及其解答。第 3 章包括在目前最为流行的两大开发环境 Borland C++ 3.1 和 Visual C++ 6.0 下的程序编辑、编译及调试的具体方法。第 4 章针对主教材中每个章节的主要内容,精心设计了 10 个具有广泛代表性的实验,以帮助读者通过实验更好地理解和把握 C 语言程序设计的特点和方法,并为每个实验题目给出了参考答案,另外还附加了期末实验上机考试的有关题目。

本书可作为高校各专业 C 语言教辅教材和全国计算机等级考试参考书。

版权所有,翻印必究。举报电话:010-62782989 13501256678 13801310933

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

本书防伪标签采用特殊防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪。

图书在版编目(CIP)数据

C 语言程序设计教程习题解答与实验指导 / 王敬华, 林萍, 张维编著. —北京: 清华大学出版社, 2006.2
(高等学校教材·计算机科学与技术)

ISBN 7-302-12441-8

I. C… II. ①王… ②林… ③张… III. C 语言-程序设计-高等学校-教学参考资料 IV. TP312

中国版本图书馆 CIP 数据核字(2006)第 005793 号

出 版 者: 清华大学出版社
<http://www.tup.com.cn>
社 总 机: 010-62770175

地 址: 北京清华大学学研大厦
邮 编: 100084
客户服务: 010-62776969

责任编辑: 魏江江

封面设计:

印 刷 者:

装 订 者:

发 行 者: 新华书店总店北京发行所

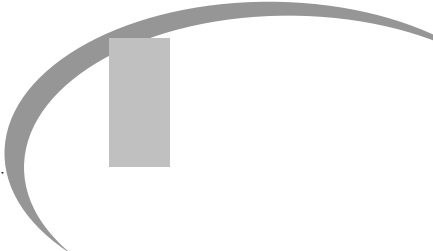
开 本: 185×260 印张: 14 字数: 328 千字

版 次: 2006 年 2 月第 1 版 2006 年 2 月第 1 次印刷

书 号: ISBN 7-302-12441-8/TP·7980

印 数: 1~ 000

定 价: 元



书是《C 语言程序设计教程》的配套教学用书，可作为高校各专业 C 语言教辅教材和全国计算机等级考试参考书。

如何让学生深刻理解和掌握 C 语言的语法规则及编程特点，克服“懂 C 语言，但不会做题、不会编程序”的通病，一直是本书作者多年来不断思考的问题。为此，本书从如下几个方面进行了精心的组织和详细介绍，希望能给读者在学习 C 语言的过程中提供一定的帮助。

1. 习题详解

本书对《C 语言程序设计教程》中全部习题，不仅给出了参考答案，而且还进行了详细的解释，让读者不仅知其然，而且还知其所以然。对于编程题来说，注重程序设计的方法和思路，习题程序按照目前最流行的编码规范以及与教材例题相统一的编码风格编写的。

2. 思考题详解

本书对《C 语言程序设计教程》中每个章节的思考题同样是做出了详细的解释，并给出了参考答案。以帮助读者对主教材中相关知识点的把握和理解。

3. 上机环境的系统介绍

为了帮助读者更快更好地熟悉 C 语言上机编程环境，本书重点介绍了目前最为流行的两大开发环境 Borland C++ 3.1 和 Visual C++ 6.0 下的标准 C 语言程序编辑、编译及调试的具体方法。

4. 上机实验题目

为了配合授课进度，加强实验上机环节，提高读者 C 语言的编程能力，本书精心设计了 10 个平时上机的实验题。这些以主要知识点为主线设计的实验题目，具有广泛的代表性和实用性，并以循序渐进的任务驱动方式，指导读者完成程序设计实验。建议实验为 20 学时，可以要求学生每 2 学时做一个实验，实验完成后，写出上机实验报告。另外，本书的最后还为实验上机考试精心编制了 15 道实验题目，每道

题目均有一定的难度和综合性，为了能顺利解答这些题目，希望读者在平时的 C 语言学习中打好基础。

本书全部习题解答和实验程序均由作者本人在 Visual C++ 和 Borland C++ 环境下调试通过。

《C 语言程序设计教程》多媒体教学课件将于教材正式出版以后制作，届时连同全部例题与习题的源程序文件一起免费提供给使用本教材的教学单位或个人。有需要者可与出版社或作者本人直接联系。

本书的编者为一本书的撰写花费了大量的心血，第 1、2 章由王敬华编写，第 3 章由张维编写，第 4 章由林萍编写，全书的统稿工作由王敬华负责。

在本书的写作过程中，魏开平副教授对本书提出了许多宝贵意见，陈静也做了大量的工作。在此向他们表示衷心的感谢。

由于作者水平有限，书中难免会有错误，恳请读者批评指正。

本书作者 E-mail 地址：jhuawang@126.com

欢迎广大读者和我们交流。

编 者

2006 年 2 月于桂子山

第 1 章 习题解答	1
1.1 习题 1 参考答案和解释	1
1.2 习题 2 参考答案和解释	4
1.3 习题 3 参考答案和解释	8
1.4 习题 4 参考答案和解释	15
1.5 习题 5 参考答案和解释	24
1.6 习题 6 参考答案和解释	36
1.7 习题 7 参考答案和解释	54
1.8 习题 8 参考答案和解释	73
1.9 习题 9 参考答案和解释	90
1.10 习题 10 参考答案和解释	111
1.11 习题 11 参考答案和解释	114
1.12 习题 12 参考答案和解释	135
第 2 章 思考题解答	148
2.1 第 3 章思考题参考答案和解释	148
2.2 第 4 章思考题参考答案和解释	149
2.3 第 6 章思考题参考答案和解释	150
2.4 第 7 章思考题参考答案和解释	150
2.5 第 8 章思考题参考答案和解释	151
2.6 第 9 章思考题参考答案和解释	151
2.7 第 11 章思考题参考答案和解释	152
2.8 第 12 章思考题参考答案和解释	152
第 3 章 C 语言上机开发环境介绍	153
3.1 Borland C++ 3.1 开发环境	153
3.1.1 Borland C++ 的启动及准备工作	153

3.1.2	Borland C++编辑器的使用	155
3.1.3	程序的编译、链接、运行	158
3.1.4	程序调试方法	160
3.1.5	工程文件的使用方法	163
3.1.6	带参数的 main 函数的运行方法	165
3.2	Visual C++ 6.0 开发环境	166
3.2.1	启动 Visual C++ 6.0 环境	166
3.2.2	建立或打开源程序文件	167
3.2.3	程序的编辑、编译、链接、运行	169
3.2.4	程序调试方法	170
3.2.5	建立工程	172
3.2.6	向已有工程中加入新文件	174
第 4 章	C 语言上机实验题	175
4.1	平时上机实验题目	175
4.1.1	实验 1 熟悉上机环境和基本数据类型编程练习	175
4.1.2	实验 2 输入/输出与顺序结构编程练习	176
4.1.3	实验 3 选择结构编程练习	177
4.1.4	实验 4 循环结构编程练习	178
4.1.5	实验 5 数组编程练习	178
4.1.6	实验 6 函数编程练习	179
4.1.7	实验 7 指针编程练习	180
4.1.8	实验 8 数组、指针和函数综合编程练习	180
4.1.9	实验 9 复杂数据类型编程练习	181
4.1.10	实验 10 文件编程练习	181
4.2	平时上机实验题目参考答案	181
4.2.1	实验 1 熟悉上机环境和基本数据类型编程练习	181
4.2.2	实验 2 输入/输出与顺序结构编程练习	183
4.2.3	实验 3 选择结构编程练习	185
4.2.4	实验 4 循环结构编程练习	189
4.2.5	实验 5 数组编程练习	191
4.2.6	实验 6 函数编程练习	194
4.2.7	实验 7 指针编程练习	197
4.2.8	实验 8 数组、指针和函数综合编程练习	199
4.2.9	实验 9 复杂数据类型编程练习	204
4.2.10	实验 10 文件编程练习	207
4.3	期末上机实验考试题目	209

习题解答

☆ 学习要点

- 完成主教材中的全部习题。
- 熟悉 C 语言程序设计考试的各种题型：选择题、程序填空题、阅读程序写出运行结果题、编程题等。
- 注意综合运用数据类型、程序结构和典型算法。

1.1 习题 1 参考答案和解释

1. 简答题

(1) 冯·诺依曼计算机模型有哪几个基本组成部分？各部分的主要功能是什么？

【参考答案】 冯·诺依曼计算机模型是由运算器、控制器、存储器、输入设备、输出设备五大功能部件组成的。

运算器又称算术逻辑部件，简称 ALU，是计算机用来进行数据运算的部件。数据运算包括算术运算和逻辑运算。

控制器是计算机的指挥系统，计算机就是在控制器的控制下有条不紊地协调工作的。

存储器是计算机中具有记忆能力的部件，用来存放程序或数据。

输入设备是用来输入程序和数据的部件。

输出设备正好与输入设备相反，是用来输出结果的部件。

(2) 简述计算机的工作原理。

【参考答案】 计算机的工作原理可简单地概括为：各种各样的信息，通过输入设备，进入计算机的存储器，然后送到运算器，运算完毕把结果送到存储器存储，最后通过输出设备显示出来。整个过程由控制器进行控制。

(3) 计算机软件系统分为哪几类？

【参考答案】 软件内容丰富，种类繁多，通常根据软件用途将其分为两大类：系统

软件和应用软件。

系统软件是指管理、监控、维护计算机正常工作和供用户操作使用计算机的软件。这类软件一般与具体应用无关，是在系统一级上提供的服务。系统软件主要包括以下两类：一类是面向计算机本身的软件，如操作系统、诊断程序等。另一类是面向用户的软件，如各种语言处理程序（像 BC、VC 等）、实用程序、字处理程序等。

应用软件是指某特定领域中的某种具体应用，供最终用户使用的软件，它必须在操作系统的基础上运行。

(4) 什么叫软件？说明软件与硬件之间的相互关系。

【参考答案】 软件是指计算机程序及有关程序的技术文档资料。两者中更为重要的是程序，它是计算机进行数据处理的指令集，也是计算机正常工作最重要的因素。在不太严格情况下，认为程序就是软件。硬件与软件是相互依存的，软件依赖于硬件的物质条件，而硬件则需在软件支配下才能有效地工作。在现代，软件技术变得越来越重要，有了软件，用户面对的将不再是物理计算机，而是一台抽象的逻辑计算机，人们可以不必了解计算机本身，可以采用更加方便、更加有效地手段使用计算机。从这个意义上说，软件是用户与机器的接口。

2. 填空题

(1) 运算器通常又称为 ALU，是计算机用来进行数据运算的部件。数据运算包括算术运算和逻辑运算。

(2) 目前计算机最常用的输入设备有键盘和鼠标。

(3) 计算机的 CPU 主要是由控制器和运算器构成的。

(4) 十进制的基数为10，二进制的基数为2。

(5) 在 C 语言中，表示一个八进制数用前缀0标记，表示一个十六进制数用前缀0x 或 0X标记。

(6) 机器数的三种表示形式是原码、反码和补码。

(7) 十进制数 23 和 -23 的 8 位二进制补码分别是00010111和11101001。

(8) 用 8 位二进制码表示有符号的定点整数，可表示的最大整数是127，最小整数是-128。

3. 选择题

(1) 计算机工作时，内存储器用来存储（ ）。

A. 程序和指令

B. 数据和信号

C. 程序和数据

D. ASCII 码和数据

【答案】 C。

【解释】 计算机内存按所存信息的类别一般分为两大类，即程序和数据。程序是用来控制计算机完成某项任务的指令的集合，而数据是程序运行处理的对象。A 只说明是程序。B 和 D 只说明是数据。因为信号和 ASCII 码均为数据，所以选 C。

(2) 语言编译程序若按软件分类则是属于（ ）。

A. 系统软件

B. 应用软件

C. 操作系统

D. 数据库管理系统

【答案】 A。**【解释】** 软件根据其用途分为两大类：系统软件和应用软件。各种语言的编译程序都是属于系统软件。C 和 D 都是具体的软件，均属于系统软件。所以选 A。

(3) 在计算机内一切信息的存取、传输和处理都是以 () 形式进行的。

A. ASCII 码

B. 二进制

C. 十进制

D. 十六进制

【答案】 B。**【解释】** 计算机只能识别二进制数。所有的信息（包括指令和数据）都是以二进制形式来存放，也是以二进制形式来进行处理的。所以选 B。

(4) 十进制数 35 转换成二进制数是 ()。

A. 100011

B. 0100011

C. 100110

D. 100101

【答案】 A。**【解释】** $(35)_{10} = (32)_{10} + (3)_{10} = (100000)_2 + (11)_2 = (100011)_2$ ，所以选 A。

(5) 十进制数 268 转换成十六进制数是 ()。

A. 10B

B. 10C

C. 10D

D. 10E

【答案】 B。**【解释】** $(268)_{10} = (256)_{10} + (12)_{10} = (100000000)_2 + (1100)_2 = (1\ 0000\ 1100)_2 = (10C)_{16}$ ，所以选 B。

(6) 下列无符号整数中最大的数是 ()。

A. $(10100011)_2$ B. $(FF)_{16}$ C. $(237)_8$

D. 789

【答案】 B。**【解释】** $(10100011)_2 = (163)_{10}$ ， $(FF)_{16} = (65535)_{10}$ ， $(237)_8 = (183)_{10}$ ，所以选 B。

(7) 与二进制数 0.1 等值的十六进制小数为 ()。

A. $(0.2)_{16}$ B. $(0.1)_{16}$ C. $(0.4)_{16}$ D. $(0.8)_{16}$ **【答案】** D。**【解释】** $(0.1)_2 = (0.1000)_{10} = (0.8)_{16}$ ，所以选 D。

(8) 将 250 与 5 进行按位与的结果是 ()。

A. 0

B. 1

C. $(FF)_{16}$ D. $(F0)_{16}$ **【答案】** A。**【解释】** $(250)_{10} \& (5)_{10} = (11111010)_2 \& (00000101)_2 = (00000000)_2 = (0)_{10}$ ，所以选 A。(9) 将 $(AF)_{16}$ 与 $(78)_{16}$ 进行按位异或的结果是 ()。A. $(D7)_{16}$ B. $(28)_{16}$ C. $(D8)_{16}$ D. $(27)_{16}$ **【答案】** A。**【解释】** $(AF)_{16} \wedge (78)_{16} = (10101111)_2 \wedge (01111000)_2 = (11010111)_2 = (D7)_{16}$ ，所以选 A。(10) 将 $(717)_8$ 进行按位求反的结果是 ()。A. $(110001)_2$ B. $(060)_8$ C. $(60)_{10}$

D. 都不正确

【答案】 B。

【解释】 $\sim(717)_{10} = \sim(111001111)_2 = (000110000)_2 = (060)_8$ ，所以选 B。

(11) 将二进制数 10110010 的最高位求反的操作是 ()。

- A. 与 $(7F)_{16}$ 按位与
- B. 与 $(7F)_{16}$ 按位异或
- C. 与 $(80)_{10}$ 按位或
- D. 都不正确

【答案】 A。

【解释】 A: $(10110010)_2 \& (7F)_{16} = (10110010)_2 \& (01111111)_2 = (00110010)_2$

B: $(10110010)_2 \wedge (7F)_{16} = (10110010)_2 \wedge (01111111)_2 = (11001101)_2$

C: $(10110010)_2 \mid (80)_{16} = (10110010)_2 \mid (10000000)_2 = (10110010)_2$

所以选 A。严格来讲，应当是与 $(80)_{16}$ 按位异或为好。

(12) 将二进制数 10110010 的高 4 位求反，低 4 位不变的操作是 ()。

- A. 与 $(0F)_{16}$ 按位与
- B. 与 $(F0)_{16}$ 按位异或
- C. 与 $(0F)_{10}$ 按位异或
- D. 与 $(0F)_{16}$ 按位或

【答案】 B。

【解释】 高 4 位求反，可以将高 4 位与 $(1111)_2$ 进行按位异或得到，低 4 位不变，可将低 4 位与 $(0000)_2$ 进行按位异或得到，故可与 $(11110000)_2$ 进行按位异或来实现题目的要求。所以选择 B。其他均不正确。

1.2 习题 2 参考答案和解释

1. 简答题

(1) C 语言有哪些特点？

【参考答案】

- a) 简洁紧凑、灵活方便
- b) 运算符丰富
- c) 数据结构丰富
- d) C 语言是结构式语言
- e) C 语言的语法限制不太严格、程序设计自由度大
- f) C 语言允许直接访问物理地址，可以直接对硬件进行操作
- g) C 语言程序生成代码质量高，程序执行效率高
- h) C 语言适用范围大，可移植性好
- i) 具有预处理功能
- j) 具有递归功能

(2) C 语言的主要应用有哪些？

【参考答案】

- a) 许多系统软件和大型应用软件都是用 C 语言编写的，如 UNIX, Linux 等操作系统。
- b) 在软件需要对硬件进行操作的场合，用 C 语言明显优于其他高级语言。例如，各种硬件设备的驱动程序（像显卡驱动程序、打印机驱动程序等）一般都是用 C 语言编

写的。

c) 在图形、图像及动画处理方面, C 语言具有绝对优势, 特别是游戏软件的开发主要就是使用 C 语言。

d) 随着计算机网络飞速发展, 特别是 Internet 的出现, 计算机通信就显得尤其重要, 而通信程序的编制首选就是 C 语言。

e) C 语言适用于多种操作系统, 像 Windows、UNIX、Linux 等绝大多数操作系统都支持 C 语言, 其他高级语言未必能得到支持, 所以在某个特定操作系统下运行的软件用 C 语言编制是最佳选择。

(3) 列举几种程序设计语言。

【参考答案】 C 语言、Java 语言、Pascal 语言、BASIC 语言、LOGO 语言等。

(4) 编写一个实现某种功能的 C 语言程序, 必须经历哪几个步骤?

【参考答案】

- a) C 语言编程工具的安装
- b) 程序编辑
- c) 程序编译
- d) 程序链接
- e) 程序运行

2. 填空题

(1) 计算机程序设计语言的发展, 经历了从 机器语言、汇编语言 到 高级语言 的历程。

(2) 计算机能惟一识别的语言是 机器语言。

(3) C 语言最初是在 ALGOL 60 语言的基础上发展而来的。

(4) C 语言程序是由 多个函数 构成的。

(5) 每个 C 语言程序中有且只有一个 main 函数, 它是程序的入口和出口。

(6) 引用 C 语言标准库函数, 一般要用 #include 预处理命令将其头文件包含进来。

(7) 用户自定义的函数, 必须先 定义 后 使用。

(8) 用户自定义函数包含两个部分, 即 声明部分 和 执行部分。声明部分 在前, 执行部分 在后, 这两部分的顺序不能颠倒, 也不能有交叉。

3. 选择题

(1) C 语言属于 ()。

- A. 机器语言 B. 低级语言 C. 中级语言 D. 高级语言

【答案】 D。

【解释】 机器语言是计算机惟一能识别的语言, 是二进制语言, 其他语言程序都必须将其编译为机器语言才能运行。低级语言是一种机器语言的符号化语言, 像汇编语言。而其他语言一般是高级语言, C 语言就是高级语言。所以选择 D。

(2) C 语言程序能够在不同的操作系统下运行, 这说明 C 语言具有很好的 ()。

- A. 适应性 B. 移植性 C. 兼容性 D. 操作性

【答案】 B。

【解释】 所谓移植性就是在某操作系统下编写的程序能够在其他操作系统下编译运行, 而程序几乎不需要做任何修改。所以选择 B。

(3) 一个 C 语言程序是由 ()。

- A. 一个主程序和若干子程序组成 B. 函数组成
C. 若干过程组成 D. 若干子程序组成

【答案】 B。

【解释】 C 语言程序是由函数构成的, 所谓函数是指功能相对独立的可以反复执行的一段程序, 在某些程序设计语言中也称为过程, 但 C 语言中叫函数。所以选择 B。

(4) C 语言规定, 在一个源程序中, main 函数的位置 ()。

- A. 必须在最开始 B. 必须在系统调用的库函数的后面
C. 可以任意 D. 必须在最后

【答案】 C。

【解释】 根据 C 语言的规定, 任何程序有且仅有一个 main 函数, main 函数可以出现在程序的任何地方, 没有位置的限制。所以选择 C。

(5) C 语言程序的执行, 总是起始于 ()。

- A. 程序中的第一条可执行语句 B. 程序中的第一个函数
C. main 函数 D. 包含文件中的第一个函数

【答案】 C。

【解释】 在一个 C 语言源程序中, 无论 main 函数书写在程序的前部, 还是后部, 程序的执行总是从 main 函数开始, 并且在 main 函数中结束。所以选择 C。

(6) 下列说法中正确的是 ()。

- A. C 语言程序书写时, 不区分大小写字母
B. C 语言程序书写时, 一行只能写一个语句
C. C 语言程序书写时, 一个语句可分成几行书写
D. C 语言程序书写时每行必须有行号

【答案】 C。

【解释】 C 语言严格区分大小写字母, 如 "A1" 和 "a1" 被认为是两个不同的标识符, C 语言程序的书写非常灵活, 既可以一行多句, 又可以一句多行, 且每行不加行号。所以选择 C。

(7) 以下叙述不正确的是 ()。

- A. 一个 C 源程序可由一个或多个函数组成
B. 一个 C 源程序必须包含一个 main 函数
C. C 语言程序的基本组成单位是函数
D. 在 C 语言程序中, 注释说明只能位于一条语句的后面

【答案】 D。

【解释】 C语言是由函数组成的，且有且仅有一个 `main` 函数，所以 C 语言程序的基本组成单位是函数。故 A、B、C 的说法都是正确的。对于 C 语言中的注释可以出现在一条语句的后面，也可以出现在一条语句或函数之前，故 D 是错误的。所以选择 D。

(8) 下面对 C 语言特点，不正确描述的是 ()。

- A. C 语言兼有高级语言和低级语言的双重特点，执行效率高
- B. C 语言既可以用来编写应用程序，又可以用来编写系统软件
- C. C 语言的可移植性较差
- D. C 语言是一种结构式模块化程序设计语言

【答案】 C。

【解释】 C 语言是介于汇编语言和高级语言之间的一种语言，由于它可以直接访问物理地址，对硬件操作，所以 C 语言既可以编写应用程序，又可以开发系统软件，而且 C 语言程序可移植性好于汇编语言，程序清晰，具有模块化的特点。所以选择 C。

(9) C 语言源程序的最小单位是 ()。

- A. 程序行
- B. 语句
- C. 函数
- D. 字符

【答案】 D。

【解释】 程序行、语句、函数都是由字符构成的，字符是 C 语言的最小单位。所以选择 D。

(10) C 语言程序的注释是 ()。

- A. 由 “/*” 开头，“*/” 结尾
- B. 由 “/*” 开头，“/*” 结尾
- C. 由 “//” 开头
- D. 由 “/*” 或 “//” 开头

【答案】 A。

【解释】 在标准 C 语言程序中，注释是由 “/*” 开头，“*/” 结尾。在 C++ 程序中，也可以由 “//” 开头对单行进行注释。所以选择 A。

(11) C 语言程序的语句都是以 () 结尾。

- A. “.”
- B. “;”
- C. “,”
- D. 都不是

【答案】 B。

【解释】 根据 C 语言的规定，在程序中所有的语句均必须由 “;” 结尾。所以选择 B。

(12) 标准 C 语言程序的文件名的后缀为 ()。

- A. .c
- B. .cpp
- C. .obj
- D. .exe

【答案】 A。

【解释】 .c 是标准 C 语言程序文件名的后缀；.cpp 则是 C++ 程序文件名的后缀；.obj 是源程序经编译后所生成的目标文件的扩展名；.exe 则是源程序经编译、链接后所生成的执行文件的扩展名。所以选择 A。

(13) C 语言程序经过编译以后生成的文件名的后缀为 ()。

- A. .c
- B. .obj
- C. .exe
- D. .cpp

【答案】 B。

【解释】 C 语言源程序经编译后生成目标 (object) 文件，其文件名后缀为 .obj。所以选择 B。

(14) C 语言程序经过链接以后生成的文件名的后缀为 ()。

- A. .c B. .obj C. .exe D. .cpp

【答案】 C。

【解释】 C 语言源程序经链接后生成可执行 (execute) 文件, 其文件名后缀为 .exe。所以选择 C。

(15) C 语言编译程序的首要工作是 ()。

- A. 检查 C 语言程序的语法错误 B. 检查 C 语言程序的逻辑错误
C. 检查程序的完整性 D. 生成目标文件

【答案】 A。

【解释】 C 语言编译程序的首要工作就是检查 C 语言程序中是否存在语法错误, 如果有则给出错误的提示信息, 如果没有则生成的目标文件 (.obj)。编译程序对程序中的逻辑错误和程序的完整性是不检查的。所以选择 A。

1.3 习题 3 参考答案和解释

1. 填空题

(1) 在 C 语言中, 基本数据类型主要有 整型、字符型、实型 三种。

(2) 根据 C 语言标识符的命名规则, 标识符只能由 字母、数字、下划线 组成, 而且第一个字符必须是 字母 或 下划线。

(3) C 语言中的常量分为 直接 常量和 符号 常量两种。定义 符号 常量需要使用预处理命令 #define。

(4) 在 C 语言中, 八进制整型常量以 0 作为前缀, 十六进制整型常量以 0x 或 0X 作为前缀。

(5) 在 C 语言中 (以 16 位 PC 为例), 一个 char 型数据在内存中所占的字节数为 1; 一个 int 型数据在内存中所占的字节数为 2。

(6) 在 C 语言中 (以 16 位 PC 为例), 一个 float 型数据在内存中所占的字节数为 4; 一个 double 型数据在内存中所占的字节数为 8。

(7) C 语言中, 设一个 int 型数据在内存中占 2 个字节, 则 int 型数据的取值范围为 -32768~+32767。

(8) 已知 int m=5, y=2; 则计算表达式 y+=y-=m*=y 后的 y 值是 -16。

(9) 语句: x++; ++x; x = x + 1; x = 1 + x; , 执行后都使变量 x 中的值增 1, 请写出一条同一功能的赋值语句 (不得与列举的相同) x += 1;。

(10) 若 a 为整型变量, 则表达式 “(a = 4 * 5, a * 2), a + 6” 的值为 26。

(11) 假设 m 是一个三位数, 从左到右用 a, b, c 表示各位的数字, 则从左到右各个数字是 bac 的三位数的表达式是 (m/10)%10*100+m/100*10+m%10。

2. 选择题

(1) 在 C 语言系统中, 假设 int 类型数据占 2 个字节, 则 double、long、unsigned int、

char 类型数据所占字节数分别为 ()。

- A. 8, 2, 4, 1 B. 2, 8, 4, 1 C. 4, 2, 8, 1 D. 8, 4, 2, 1

【答案】 D。

【解释】 C 语言系统中, 如果 int 型数据占 2 字节, 则说明该系统是 16 位的系统, 此时 double 型数据占 8 字节, long 型数据占 4 字节, unsigned int 占 2 字节, char 型数据占 1 字节, 所以选择 D。

(2) 下面四个选项中, 均是不合法的用户标识符的选项是 ()。

- A. A P_0 do B. float la0 _A
C. b-a sizeof int D. _123 temp int

【答案】 C。

【解释】 根据 C 语言中对标识符的规定: A 中的 A、P_O 是合法的, do 是关键字, 非法; B 中 la0、_A 是合法的, float 是关键字, 非法; C 中 b-a 非法, 因“-”不是标识符中的有效字符, sizeof 和 int 均是关键字, 非法; D 中 _123、temp 是合法的, int 是关键字, 非法。故只有 C 全错, 所以选择 C。

(3) 下面四个选项中, 均是合法整型常量的选项是 ()。

- A. 160 -0xffff 011 B. -0xcdf 01a 0xe
C. -01 986,012 0668 D. -0x48a 2e5 0x

【答案】 A。

【解释】 A 中 160 是十进制数, -0xffff 是十六进制数, 011 是八进制数, 均合法; B 中 01a 非法, 因为 a 不是八进制数码; C 中 986,012 非法, 不能包含“,”, 0668 非法, 因为 8 不是八进制数码; D 中 0x 非法, 因为后面没有有效的十六进制数码。所以选择 A。

(4) 下面四个选项中, 均是不合法的浮点数的选项是 ()。

- A. 160. 0.12 e3 B. 123 2e4.2 .e5
C. -.18 123e4 0.0 D. -e3 .234 1e3

【答案】 B。

【解释】 C 语言中的浮点数有两种形式, 一种为十进制小数形式, 一种为指数形式, 其一般形式为 $a E n$, a 为十进制数, n 为十进制整数, 都不可省略。A 中 e3 非法, 因为只有阶码 3 没有尾数, 其余两数都是合法的浮点数; B 中 123 是整数, 不是浮点数, 2e4.2 阶码部分 4.2 是浮点数, 不是整数, 故是非法的, .e5 尾数部分不能只有小数点, 也是非法的; C 中的三个数均是合法的浮点数; D 中的 .234 和 1e3 也是合法的, 只有 -e3 非法。所以选择 B。

(5) 下面四个选项中, 均是不合法的转义字符的选项是 ()。

- A. \" \"' \"xf' B. \"1011' \"' \"ab'
C. \"011' \"f' \"j' D. \"abc' \"101' \"xlf'

【答案】 B。

【解释】 A 中均为合法的转义字符; B 中 \"1011' 的 \"后面多于 3 位八进制数是非法的, \"不能标识 \" 字符, 是非法的, \"ab' 的 \"后面漏掉了 x 是非法的; C 中 \"011' 是合法的; D 中 \"101' 是合法的; 故都不合法的只有 B, 所以选择 B。

(6) 下面四个选项中, 均是正确的数值常量或字符常量的选项是 ()。

- A. 0.0 0f 8.9e '&' B. "a" 3.9e-2.5 1e1 \"
- C. '3' 011 0xff00 0a D. +001 0xabcd 2e2 50.

【答案】 D。

【解释】 A 中 0f、8.9e 是非法的数值常量; B 中 "a" 是字符串常量, 是非法的数值常量或字符常量; C 中 0a 是非法的数值常量; D 中均是合法的数值常量; 所以选择 D。

(7) 下面程序段输出结果是 ()。

```
int i = 5, k;  
k = (++i) + (++i) + (i++);  
printf ("%d,%d", k, i);
```

- A. 24,8 B. 21,8 C. 21,7 D. 24,7

【答案】 B。

【解释】 $k = (++i) + (++i) + (i++)$ 表达式中, "++" 号在 i 前面的有两个, 所以在计算 k 之前, i 要先加两次 1, 即 i 变为 7, 然后再将 3 个 7 相加, 使得 k 的值为 21, 表达式中 "++" 号在 i 后面的有一个, 所以得出 k 的值以后 i 又增 1 次变为 8。所以正确答案为 B。

(8) 下面程序段输出结果是 ()。

```
short int i = 32769;  
printf ("%d\n", i);
```

- A. 32769 B. 32767
- C. -32767 D. 输出不是确定的数

【答案】 C。

【解释】 因 $(32769)_{10} = (1000\ 0000\ 0000\ 0001)_2$, 所以 i 的值在内存中补码形式表示为 1000 0000 0000 0001, 最高位是 1 表示负数, 其表示的有符号数是 $-(0111\ 1111\ 1111\ 1111)_2$, 即十进制数 -32767。所以正确答案是 C。

(9) 若有说明语句: `char c = '\72';` 则变量 c ()。

- A. 包含 1 个字符 B. 包含 2 个字符
- C. 包含 3 个字符 D. 说明不合法, c 的值不确定

【答案】 A。

【解释】 因为 '\72' 是转义字符, 表示其 ASCII 码为八进制数 72 的字符, 即 ':' 字符, 所以正确答案是 A。

(10) 若有定义: `int a = 7; float x = 2.5, y = 4.7;` 则表达式 `x + a % 3 * (int)(x + y) % 2 / 4` 的值是 ()。

- A. 2.500000 B. 2.750000 C. 3.500000 D. 0.000000

【答案】 A。

【解释】 本题考查运算符的优先级概念, 式中要先算 $(x+y)$ 的值, 再进行强制类型变换, `*`、`/`、`%` 是同级的运算符, 要从左到右计算, 最后算加法。所以正确的结果是 A。

(11) 设变量 a 是整型, f 是实型, i 是双精度型, 则表达式 `10 + 'a' + i * f` 值的数据

类型为 ()。

- A. int B. float C. double D. 不确定

【答案】 C。

【解释】 不同类型数据混合运算的转换规律是：运算前 float 型数据会自动转换为 double 型，char 型数据会自动转换为 int 型。运算时 int 型数据和 double 型数据要先化为相同类型，即 double 型，运算结果也为 double 型。所以正确答案为 C。

(12) sizeof (float)是 ()。

- A. 一个双精度型表达式 B. 一个整型表达式
C. 一种函数调用 D. 一个不合法的表达式

【答案】 B。

【解释】 sizeof 是计算某数据类型的变量所占内存大小的运算符，其值是占用的字节数，为一整型值，所以正确答案是 B。

(13) 设变量 n 为 float 类型，m 为 int 类型，则以下能实现将 n 中的数值保留小数点后两位，第三位进行四舍五入运算的表达式是 ()。

- A. $n = (n * 100 + 0.5) / 100.0$ B. $m = n * 100 + 0.5, n = m / 100.0$
C. $n = n * 100 + 0.5 / 100.0$ D. $n = (n / 100 + 0.5) * 100.0$

【答案】 B。

【解释】 要实现将 n 中的数值保留小数点后两位，第三位进行四舍五入，则首先可将 n 乘以 100，此时 $n*100$ 的最低两位整数将是 n 的前两位小数， $n*100$ 的第一位小数将是 n 的第三位小数，再将 $n*100$ 加上 0.5，如果 $n*100$ 的第 1 位小数（即 n 的第三位小数）大于或等于 0.5，则整数部分加 1，否则整数部分不变，然后再将结果赋值给一整型变量其实就是将 $n*100+0.5$ 的整数部分赋值给整型变量，从而完成了 n 的第三位小数的四舍五入操作，再将该整型变量除以 100.0 就得到了所希望的结果。所以正确答案是 B。

(14) 在 C 语言中，要求运算数必须是整型的运算符是 ()。

- A. / B. ++ C. != D. %

【答案】 D。

【解释】 /、++、!=运算符都可以对浮点数进行运算，但%只能对整型数据进行运算。所以应选择 D。

(15) 若变量已正确定义并赋值，下面符合 C 语言语法的表达式是 ()。

- A. $a:=b+1$ B. $a=b=c+2$ C. $\text{int } 18.5\%3$ D. $a=a+7=c+b$

【答案】 B。

【解释】 赋值运算符=的左边必须是变量，不能是表达式，所以 D 是非法的；对于 C 来说，int 掉了括号，应写成(int)18.5%3，所以也是错误的；A 中的:=是非法的运算符。而 B 的功能是将 c+2 的值赋值给 b，再将 b 的值赋值给 a，是正确的，所以应选择 B。

(16) 若有定义: $\text{int } k = 7, x = 12;$ ，则能使值为 3 的表达式是 ()。

- A. $x \% = (k \% = 5)$ B. $x \% = (k - k \% 5)$
C. $x \% = k - k \% 5$ D. $(x \% = k) - (k \% = 5)$

【答案】 D。

【解释】 A 的计算结果为 6; B 的计算结果为 2; C 的计算结果为 2; D 的计算结果为 3, 因为 $(x\%k)$ 的结果是 5, 而 $(k\%5)$ 的结果是 2。所以应选择 D。

(17) 若变量 a、i 已正确定义, 且 i 已正确赋值, 合法的语句是 ()。

- A. $a==1$ B. $++i$; C. $a = a++ = 5$; D. $a = \text{int}(i)$;

【答案】 B。

【解释】 A 是关系表达式, 不是 C 语言的语句; C 中的 $a++$ 是表达式不能出现在赋值运算符的左边, 故是非法的; D 中 int 不可作为函数名, 应写为 $(\text{int})i$ 的形式; B 是将 i 的值增 1, 是合法的, 所以应选择 B。

(18) 若有如下程序

```
void main( )
{
    int y = 3, x = 3, z = 1;
    printf ("%d %d\n", (++x, y++), z+2);
}
```

运行该程序的输出结果是 ()。

- A. 3 4 B. 4 2 C. 4 3 D. 3 3

【答案】 D。

【解释】 逗号表达式的求值顺序是从左向右依次计算用逗号分隔的各表达式的值, 最后一个表达式的值就是整个逗号表达式的值, 所以 $(++x, y++)$ 的值将是 $y++$, 因为是先输出 y 的值, 再将 y 增 1, 所以首先输出 3, 再输出 $z+2$ 的值, 即 3。所以应选择 D。

(19) 下面正确的字符常量是 ()。

- A. "c" B. "\\" C. 'W' D. "

【答案】 C。

【解释】 C 语言中的字符常量必须由单引号括起来的字符。A 表示的是一字符串常量, 不是字符常量; B 中右边不是单引号是错误的; D 中单引号之间没有任何字符也是非法的; C 表示的是 W 字符, 是正确的, 所以应选择 C。

(20) 在 C 语言中(以 16 位 PC 为例), 5 种基本数据类型的存储空间长度的排列顺序为 ()。

- A. $\text{char} < \text{int} < \text{long int} < \text{float} < \text{double}$
B. $\text{char} = \text{int} < \text{long int} < \text{float} < \text{double}$
C. $\text{char} < \text{int} < \text{long int} = \text{float} = \text{double}$
D. $\text{char} = \text{int} = \text{long int} < \text{float} < \text{double}$

【答案】 A。

【解释】 在 16 位的 PC 中, char 类型占 1 字节, int 占 2 字节, long int 占 4 字节, float 占 4 字节, double 占 8 字节。所以正确答案是 A。

(21) 假设所有变量均为整型, 则表达式 $(a = 2, b = 5, b++, a + b)$ 的值是 ()。

- A. 7 B. 8 C. 6 D. 2

【答案】 B。

【解释】 逗号表达式的求值顺序是从左向右依次计算用逗号分隔的各表达式的值，最后一个表达式的值就是整个逗号表达式的值。所以表达式最后的值是 $a+b$ ，但经过前面的计算此时 a 的值为 2， b 的值是 6，所以最后的结果是 8，应选择 B。

(22) 以下正确的叙述是 ()。

- A. 在 C 语言程序中，每行中只能写一条语句
- B. 若 a 是实型变量，C 语言程序中允许赋值 $a=10$ ，因此实型变量中允许存放整型数
- C. 在 C 语言程序中，无论是整数还是实数，都能被准确无误地表示
- D. 在 C 语言程序中， $\%$ 是只能用于整数运算的运算符

【答案】 D。

【解释】 在 C 语言程序中，每行中可以写多行语句，故 A 的说法错误；整型数可以赋值给实型变量，其实是将整数转换成实型数后再赋值给实型变量的，并不是实型变量中允许存放整型数，故 B 也是错误的；在 C 语言程序中，整型数据可以准确的表示，但实数因有其精度的限制，不可能准确无误地表示，故 C 也是错误的； $\%$ 运算符只能用于整数的运算，所以 D 是正确的。

(23) 假定 x 和 y 为 `double` 型，则表达式 $x=2, y=x+3/2$ 的值是 ()。

- A. 3.500000
- B. 3
- C. 2.000000
- D. 3.000000

【答案】 D。

【解释】 逗号表达式的值是最后一个表达式的值，即 $y=x+3/2$ 的值，因 x 的值是 2.000000， $3/2$ 的值是 1，所以 y 的值是 3.000000，应选择 D。

(24) 以下程序的输出结果是 ()。

```
void main( )
{
    int a = 3;
    printf ("%d\n", (a+=a-=a*a) );
}
```

- A. -6
- B. 12
- C. 0
- D. -12

【答案】 D。

【解释】 $a+=a-=a*a$ 等价于 $a=a+(a-a*a)$ ，即先计算 $a=a-a*a$ ，所以此时 a 的值为 $3-3*3=-6$ ，再计算 $-6+(-6)=-12$ 赋值给 a ，所以 a 的值为 -12，也就是整个表达式的值。所以应选择 D。

(25) 在 16 位 C 语言编译系统上，若定义 `long a;`，则能给 a 赋 40000 的正确语句是 ()。

- A. $a=20000+20000;$
- B. $a=4000*10;$
- C. $a=30000+10000;$
- D. $a=4000L*10L;$

【答案】 D。

【解释】 在 16 位 C 语言编译系统上, A、B、C 中的计算结果都是 40000, 是一整型数据, 即占 16 个二进制位, 其 16 位二进制补码为: 1001 1100 0100 0000, 最高为 1, 其实表示一负数, 则赋给变量 a 后其值为: 1111 1111 1111 1111 1001 1100 0100 0000, 表示的值为-25536, 而不是 40000, 所以 A、B、C 都是错误的; 对于 D 来说, 因为 $4000L * 10L = 40000L$ 是一长整型数, 没有超出长整型数的表示范围, 所以是正确的, 结果应选择 D。

(26) 若变量 a 是 int 类型, 并执行了语句: `a = 'A' + 1.6;`, 则正确的叙述是 ()。

- A. a 的值是字符 C
- B. a 的值是浮点型
- C. 不允许字符型和浮点型相加
- D. a 的值是字符'A'的 ASCII 码值加上 1

【答案】 D。

【解释】 `a = 'A' + 1.6;` 其功能是将字符'A'的 ASCII 码的值加上 1.6, 然后将运算结果的整数部分赋值给整型变量 a, 所以 a 的值是字符'A'的 ASCII 码值加上 1。正确答案为 D。

(27) 语句 `printf("a\\bre\\'hi\\'y\\\\"bou\\n");` 的输出结果是 ()。

- A. `a\\bre\\'hi\\'y\\\\"bou`
- B. `a\\bre\\'hi\\'y\\bou`
- C. `re\\'hi\\'you`
- D. `abre\\'hi\\'y\\bou`

【答案】 C。

【解释】 首先要明确该字符串中包含的转义字符, "`a \\b re \\ ' hi \\ ' y \\ \\b ou \\n`" 中的转义字符(带下划线的部分)共有 6 个, 其中 `\\b` 是退格符, 输出时将光标往左边回退一个位置, `\"` 为单引号字符, `\\` 为 `\\` 字符, `\\n` 为回车换行符。所以显示的结果为 C。

(28) 下列程序执行后的输出结果是 ()。

```
void main( )
{ int x = 'f'; printf("%c \\n", 'A'+(x-'a'+1)); }
```

- A. G
- B. H
- C. I
- D. J

【答案】 A。

【解释】 因为 x 为 'f', 所以 `x-'a'+1` 的值为 6, 'A'+6 对应的是 'G', 所以输出的结果是 A。

(29) 下列程序执行后的输出结果是 ()。

```
void main( )
{ char x = 0xFFFF; printf ("%d \\n", x--); }
```

- A. -32767
- B. FFFE
- C. -1
- D. -32768

【答案】 C。

【解释】 因为 x 是字符型, 所以将 0xFFFF 赋值给 x, x 的值为 0xFF, 因为数据在内存中是其补码表示的, 0xFF 的真值是 -1, 所以显示的结果是 -1, 即正确答案为 C。

(30) 已知: `int x = 1, y = -1;`, 则语句 `printf("%d\\n", (x-- & ++y));` 的输出结果是 ()。

- A. 1 B. 0 C. -1 D. 2

【答案】 B。

【解释】 $x-- \& ++y$ 是将 x 的值 1 与 $++y$ 的值 0 进行按位与操作，其结果显然是 0。所以正确答案为 B。

1.4 习题 4 参考答案和解释

1. 填空题

- (1) 在 C 语言中，格式化输入库函数为 scanf，格式化输出库函数为 printf。
- (2) printf 函数中的格式控制字符串的作用是 按指定的输出格式将信息输出到屏幕上，它包含两类字符，即 常规字符 和 格式控制符。
- (3) 格式转换符中，除了 X、E、G 以外，其他均为小写字母。
- (4) getche 函数和 getch 函数在功能上的主要区别是 getche 输入的字符回显，而 getch 输入的字符不回显。
- (5) 在输入数据类型和字符类型的两个 scanf 函数调用之间通常要使用 fflush 库函数以清除键盘缓冲区中的字符。
- (6) 算法是 解决某个问题的具体的方法和步骤。
- (7) 算法的描述方法有 自然语言描述、伪代码、流程图、N-S 图、PAD 图等。
- (8) 任何复杂的程序都可以由 顺序、分支 和 循环 这三种基本结构组成。

2. 选择题

- (1) 在 16 位 C 语言编译系统上，下列程序执行后的输出结果是 ()。

```
void main( )
{ int a = -32769; printf ("%8U\n", a); }
```

- A. 32769 B. 8U C. 32767 D. -32767

【答案】 B。

【解释】 格式字符必须小写，题中的 "%8U" 不会当作格式控制符，而是原样输出，变量 a 将没有对应的格式说明，也不会输出。所以正确答案为 B（注意：在 VC 下结果为 U）。

- (2) 下面程序段执行后的输出结果是 ()。（“□”表示一个空格）

```
int a = 3366;
printf ("%08d|", a);
```

- A. |-0003366| B. |00003366|

C. |3366□□□□|

D. 输出格式非法

【答案】 C。**【解释】** % -08d 表示输出占 8 个空格的位置, 并且左对齐, 所以正确的答案是 C。

(3) 以下程序的输出结果是 ()。

```
void main( )
{
    printf ("s1 = |%15s|    s2 = |%-5s|", "chinabeijing", "chi");
}
```

A. s1 = |chinabeijing□□□□| s2 = |chi|

B. s1 = |chinabeijing□□□□| s2 = |chi□□□|

C. s1 = |□□□chinabeijing| s2 = |□□chi|

D. s1 = |□□□chinabeijing| s2 = |chi□□□|

【答案】 D。

【解释】 %15s 表示输出占 15 个空格的位置, 并且右对齐, 左边多余的位置补空格, 因为 "chinabeijing" 包含 12 个字符, 所以输出时, 左边要补上 3 个空格, %-5s 表示输出占 5 个空格的位置, 并且左对齐, 右边多于位置补空格, 因为 "chi" 包含 3 个字符, 所以输出时, 右边要补上 2 个空格。所以正确的答案是 D。

(4) 在 16 位 C 语言编译系统上, 以下程序的输出结果是 ()。

```
void main( )
{
    long y = -43456;
    printf ("y = |%-8ld|  y = |%-08ld|  y = |%08ld|  y = |%+8ld|", y, y, y, y);
}
```

A. y = |□□-43456| y = |-□□43456| y = |-0043456| y = |-43456□□|

B. y = |□□-43456| y = |-43456□□| y = |-0043456| y = |-□□43456|

C. y = |-43456□□| y = |-43456□□| y = |-0043456| y = |□□-43456|

D. y = |-43456□□| y = |-4345600| y = |-0043456| y = |□□-43456|

【答案】 C。

【解释】 %-8ld 表示输出占 8 个空格的位置, 左对齐, 右边多余的位置补空格, 故第一个 y 的输出为: |-43456□□|。%-08ld 等价于 %-8ld, 因为不可能在右边空位上补 0, 所以第二个 y 的输出也为: |-43456□□|。%08ld 表示输出占 8 个空格的位置, 右对齐, 左边多余的位置补 0, 所以第三个 y 的输出为: |-0043456|。%+8ld 表示输出占 8 个空格的位置, 右对齐, 左边多余的位置补空格, 必须输出正负号, 所以第四个 y 的输出为: |□□-43456|。正确答案应为 C。

(5) 在 16 位 C 语言编译系统上, 以下程序的输出结果是 ()。

```
void main( )
{
```

```
int y = 2456;
printf ("y = |%3o| y = |%8o| y = |%#8o| y = |%08o|", y, y, y, y);
}
```

- A. y = |2456| y = |□□□□2456| y = |□□□02456| y = |00002456|
 B. y = |4630| y = |□□□□4630| y = |□□□04630| y = |00004630|
 C. y = |2456| y = |□□□□2456| y = |###02456| y = |00002456|
 D. y = |4630| y = |4630□□□□| y = |###04630| y = |00004630|

【答案】 B。

【解释】 因为 y 的输出是以其八进制的形式输出， $(2456)_{10}=(4630)_8$ ，所以 A 和 C 是错误的。%3o 表示以八进制数形式输出，占 3 个空格的位置，右对齐，左边多余的位置补空格，但实际数据的宽度为 4 大于规定的宽度，所以此时按实际宽度输出，故第一个 y 的输出为：|4630|。%8o 与 %3o 的差别就在于输出占 8 个空格的位置，所以右边要补 4 个空格，故第二个 y 的输出也为：|□□□□4630|。%#8o 与 %8o 的差别只是输出时必须输出八进制前导 0，所以第三个 y 的输出为：|□□□04630|。%08o 与 %8o 的差别只是输出时左边多余的位置补 0，所以第四个 y 的输出为：|00004630|。正确答案应为 B。

(6) 若有说明语句：int a; float b;，以下输入语句正确的是 ()。

- A. scanf ("%f %f", &a, &b); B. scanf ("%f %d", &a, &b);
 C. scanf ("%d,%f", &a, &b); D. scanf ("%6.2f%6.2f", &a, &b);

【答案】 C。

【解释】 因为 a 是 int 型，所以格式控制符应为 %d，而 f 是 float 型，所以格式控制符应为 %f。故正确答案应为 C。

(7) 执行下面程序段，给 x、y 赋值时，不能作为数据分隔符的是 ()。

```
int x, y;
scanf ("%d%d", &x, &y);
```

- A. Space 键 B. Tab 键 C. Enter 键 D. 逗号

【答案】 D。

【解释】 如果相邻两个格式控制符之间，不指定数据分隔符（如逗号、冒号等），则相应的两个输入数据之间，至少用 Space 键分隔，或者用 Tab 键分隔，或者输入一个数据后，按 Enter 键，然后再输入下一个数据。而逗号不能作为数据分隔符，应选择 D。

(8) 执行下面程序时，欲将 25 和 2.5 分别赋给 a 和 b，正确的输入方法是 ()。

```
int a;
float b;
scanf ("a=%d,b=%f", &a, &b);
```

- A. 25□2.5 B. 25,2.5 C. a=25,b=2.5 D. a=25□b=2.5

【答案】 C。

【解释】 格式控制字符串中出现的常规字符（包括转义字符），务必原样输入。所

以正确答案为 C。

(9) 若有说明语句: `int a, b;`, 用户的输入为 111222333, 结果 a 的值为 111, b 的值为 333, 那么以下输入正确的语句是 ()。

- A. `scanf ("%*3d%3c%3d", &a, &b);` B. `scanf ("%3d%*3c%3d", &a, &b);`
 C. `scanf ("%3d%3d%*3d", &a, &b);` D. `scanf ("%3d%*2d%3d", &a, &b);`

【答案】 B。

【解释】 当格式控制字符串中含有抑制符 “*” 时, 表示本输入项对应的数据读入后, 不赋给相应的变量 (该变量由下一个格式指示符输入)。所以 A 的结果将使 a 的值不确定 (因为 %3c 与 a 的类型不匹配), b 的值为 333; C 的结果将使 a 的值为 111, b 的值为 222; D 的结果将使 a 的值为 111, b 的值为 233; B 的结果将使 a 的值为 111, b 的值为 333, 中间输入的 222 不赋给任何变量 (%*3c 的作用)。所以正确答案是 B。

(10) 执行下面的程序时, 假设用户输入为 1□22□333, 则 ch1、ch2 和 ch3 的值为 ()。

```
char ch1, ch2, ch3;
scanf ("%1c%2c%3c", &ch1, &ch2, &ch3);
```

- A. '1'、'2'、'3' B. '1'、' '、'2' C. '1'、'2'、' ' D. '1'、' '、'3'

【答案】 B。

【解释】 当格式控制符是 %nc, 空格符和转义字符均作为有效字符被输入, 将把 n 个字符中的第一个字符赋值给相应的字符变量, 其余字符被舍弃。所以 ch1 的值为 '1', ch2 将是 □2 的第一个空格符 ' ', ch3 为 2□3 的第一个字符 '2'。所以正确答案是 B。

(11) 已知: `int x, y; double z;` 以下语句中错误的函数调用是 ()。

- A. `scanf ("%d,%1x,%1e", &x, &y, &z);`
 B. `scanf ("%2d*%d%1f", &x, &y, &z);`
 C. `scanf ("%x%*d%o", &x, &y);`
 D. `scanf ("%x%o%6.2f", &x, &y, &z);`

【答案】 D。

【解释】 scanf 中的格式控制符不能指明浮点数的精度, D 中 %6.2f 是错误的。所以应选择 D。

(12) 已有如下定义和输入语句, 若要求 a1, a2, c1, c2 的值分别为 10, 20, A 和 B, 当从第一列开始输入数据时, 正确的数据输入方式是 ()。

```
int a1, a2; char c1, c2;
scanf ("%d%c%d%c", &a1, &c1, &a2, &c2);
```

- A. 10A□20□B✓ B. 10□A□20□B✓ C. 10A20B✓ D. 10A20□B✓

【答案】 C。

【解释】 scanf 用于输入整数时, 当遇到非数字字符结束, 此时非数字字符将存入键盘缓冲区, 如果接着读入字符, 将从键盘缓冲区中读取该字符。对于 C 的输入, 10 后面

是字符'A'，则将 10 给 a1，'A'就赋给 c1，20 赋给 a2，'B'赋给 c2。所以正确的输入方法是 C。

(13) 阅读以下程序，当输入数据的形式为 25,13,10↵，正确的输出结果为（ ）。

```
void main( )
{
    int x, y, z;
    scanf ("%d%d%d", &x, &y, &z);
    printf ("x + y + z = %d\n", x + y + z);
}
```

A. $x+y+z=48$ B. $x+y+z=38$ C. $x+y+z=35$ D. 无法确定

【答案】 D。

【解释】 当一次 scanf 调用需要输入多个数据项时，如果前面数据的输入遇到非法字符，并且输入的非法字符不是格式控制字符串中的常规字符，那么，这种非法输入将影响后面数据的输入，导致数据输入失败。当输入为 25,13,10 时，a 的值是 25，y 和 z 的值将无法预测，所以 $x+y+z$ 的值是无法确定的。故正确答案是 D。

(14) 已有定义 int x; float y; 且执行 scanf("%3d%f", &x, &y); 语句时，假设输入数据为 12345□678↵，则 x、y 的值分别为（ ）。

A. 12345 678.000000

B. 123 678.000000

C. 123 45.678000

D. 123 45.000000

【答案】 D。

【解释】 因为 %3d，所以输入中的前 3 位数 123 将赋给 x，输入中后面的 45 将赋给 y，因为 45 后键入了空格符使得后面的输入无效，又因 y 是浮点数，所以 y 的值应为 45.000000。故正确答案是 D。

(15) 阅读以下程序，当输入数据的形式为 12a345b789↵，正确的输出结果为（ ）。

```
void main( )
{
    char c1, c2;
    int a1, a2;
    c1 = getchar( );
    scanf ("%2d", &a1);
    c2 = getchar( );
    scanf ("%3d", &a2);
    printf ("%d, %d, %c, %c\n", a1, a2, c1, c2);
}
```

A. 2, 345, 1, a

B. 12, 345, a, b

C. 2a, 45b, 1, 3

D. 2, 789, 1, a

【答案】 A。

【解释】 通过调用 getchar 函数将输入的第一个字符赋给 c1，所以 c1 的值是字符'1'，

接下来调用 `scanf` 函数将输入的两位数赋给整型变量 `a1`，但此时输入的两位是 `2a`，`a` 是无效的数字，所以只将 `2` 赋给变量 `a1`，字符 `'a'` 通过后续的 `getchar` 函数将其赋给变量 `c2`，再接下来调用 `scanf` 函数将输入的三位数赋给整型变量 `a2`，则 `a2` 的值为 `345`。所以正确答案是 `A`。

3. 编程题

(1) 编写一程序要求任意输入四位十六进制整数，以反序的方式输出该十六进制数。例如：输入 `9AF0`，则输出 `0FA9`。

【设计思想】 对输入的四位十六进制数，分别取其最低、次低、次高和最高的一位十六进制数，具体方法可以将该数分别与 `0X000F`、`0X00F0`、`0X0F00` 和 `0XF000` 进行按位与的运算，然后分别进行左移 12 位、左移 4 位、右移 4 位和右移 12 位，再把所得到的每一步的值进行求和即可得到该十六进制数的反序结果。例如： $(0X9AF0 \& 0X000F) \ll 12$ 得到 `0X0000`， $(0X9AF0 \& 0X00F0) \ll 4$ 得到 `0X0F00`， $(0X9AF0 \& 0X0F00) \gg 4$ 得到 `0X00A0`， $(0X9AF0 \& 0XF000) \gg 12$ 得到 `0X0009`，再把得到的结果进行求和，即 `0X0000 + 0X0F00 + 0X00A0 + 0X0009` 就得到 `0X0FA9`。

【参考答案】

```
#include <stdio.h>

void main( )
{
    unsigned short a, b;

    scanf ("%4X", &a);        //输入4位十六进制数给变量a
    b = (a & 0x000F) << 12; //取a的最低一位十六进制数并左移12位后赋值给b
    b += (a & 0x00F0) << 4; //取a的次低一位十六进制数并左移4位后与b相加再赋值给b
    b += (a & 0x0F00) >> 4; //取a的次高一位十六进制数并右移4位后与b相加再赋值给b
    b += (a & 0xF000) >> 12; //取a的最高一位十六进制数并左移12位后与b相加再赋值给b
    printf ("%4X\n", b);      //输出变换后的十六进制数
}
```

程序运行结果（假设输入为：`9AF0`✓）：

0FA9

(2) 编程从键盘输入两个整数分别给变量 `a` 和 `b`，要求在不借助于其他变量的条件下，将变量 `a` 和 `b` 的值实现交换。

【设计思想】 首先将变量 `a` 的值与变量 `b` 的值进行相加存入变量 `a` 中，然后再把此时 `a` 的值减去 `b` 的值后再赋给 `b`，`b` 中的值就是原来 `a` 的值，最后将 `a` 的值减去 `b` 的值就得到了原来 `b` 的值，存入变量 `a` 中。这样就实现了将变量 `a` 和 `b` 的值的交换。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int a, b;
    scanf ("%d%d", &a, &b);
    printf ("before swap: a = %d b = %d\n", a, b);
    a = a + b;
    b = a - b;
    a = a - b;
    printf ("after swap: a = %d b = %d\n", a, b);
}
```

程序运行结果（假设输入为：10 20✓）：

```
before swap: a = 10 b = 20
after swap: a = 20 b = 10
```

（3）编程从键盘输入圆的半径 r ，计算并输出圆的周长和面积。

【设计思想】 圆的周长 $=2\times$ 半径 $\times\pi$ ，圆的面积 $=$ 半径的平方 $\times\pi$ 。

【参考答案】

```
#include <stdio.h>

#define PI 3.14159

void main( )
{
    float r, c, s;

    scanf ("%f", &r);
    c = 2 * r * PI;
    s = PI * r * r;
    printf ("c = %f s = %f\n", c, s);
}
```

程序运行结果（假设输入为：2.5✓）：

```
c = 15.707950 s = 19.634937
```

（4）编程从键盘输入任意一个十六进制负整数，以输入的形式输出。例如：输入

-FA98, 输出-FA98。

【设计思想】 对输入的十六进制负整数, 由于内存中是以其补码的形式存放, 如果以%X 直接输出将是其补码的十六进制形式。例如, 如果输入为-FA98, 其输出将是 568, 而不是-FA98。因此可将该值求负, 输入时加入负号来实现。

【参考答案】

```
#include <stdio.h>

void main( )
{
    short int a;

    scanf ("%X", &a);
    printf ("-%hX\n", -a);
}
```

程序运行结果 (假设输入为: -FA98✓):

-FA98

(5) 已知一元二次方程 $ax^2 + bx + c = 0$, 编一程序当从键盘输入 a、b、c 的值后, 计算 x 的值。

【设计思想】 程序运行时应保证输入的 a,b,c 的值满足 $b^2 - 4ac \geq 0$ 的条件。另外, 根据一元二次方程的求根公式: $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$, $x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$, 注意写成:
 $x_1 = (-b + \text{sqrt}(b*b - 4*a*c))/(2*a)$, $x_2 = (-b - \text{sqrt}(b*b - 4*a*c))/(2*a)$ 。

【参考答案】

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

void main( )
{
    float a, b, c;
    float d;
    float x1, x2;

    printf ("input a,b,c: ");
```

```
scanf ("%f%f%f", &a, &b, &c);  
d = sqrt (b * b - 4 * a * c);  
x1 = (-b + d) / (2 * a);  
x2 = (-b - d) / (2 * a);  
printf ("x1 = %.2f x2 = %.2f\n", x1, x2);  
}
```

程序运行结果（假设输入为：1 -5 6✓）：

```
x1 = 3.00 x2 = 2.00
```

（6）假设从键盘输入从某日午夜零点到现在已经历的时间（单位：s），编一程序计算到现在为止已过了多少天，现在的时间是多少？

【设计思想】 一天总的秒数是 24×3600 ，所以计算已过天数可以用总的秒数整除 24×3600 得到。计算现在的时间，应将总的秒数减去已过天数所需的秒数后，再整除 3600 得现在时间的时数，现在的分数则是除以 3600 后的余数来整除 60 得到，秒数则是除以 3600 后的余数再除以 60 所得到的余数。

【参考答案】

```
#include <stdio.h>  
  
void main( )  
{  
    unsigned long t, r;  
    int d, h, m, s;  
  
    printf ("second = ");  
    scanf ("%d", &t);  
    d = t / (24 * 3600);          //天数  
    r = t % (24 * 3600);         //当天时间的总秒数  
    h = r / 3600;                //当天的时数  
    m = (r % 3600) / 60;         //当天的分数  
    s = (r % 3600) % 60;        //当天的秒数  
    printf ("have passed days is %d, current time is %02d:%02d:%02d\n", d,  
        h, m, s);  
}
```

程序运行结果（假设输入为：1234567✓）：

```
have passed days is 14, current time is 06:56:06
```

1.5 习题 5 参考答案和解释

1. 填空题

(1) C 语言中的语句可以分为表达式语句、函数调用语句、复合语句、空语句、控制语句 五类。

(2) C 语言用0表示假，非 0表示真。

(3) C 语言提供的三种逻辑运算符是与 (&&)，或 (||) 和 非 (!)。

(4) 关系运算符具有左结合性，相同优先级的关系运算符连用时，按照从左向右的顺序计算表达式的值。

(5) 对于一个关系表达式的值，0表示假，1表示真。

(6) 对于 C 语言运算符的优先级，单目运算符优先级最高，逗号运算符的优先级最低。

(7) C 语言中用于选择结构的控制语句有if语句和switch语句两种，前者用于两者选一的情况，而后者用于多分支选一的情形。

(8) 当 $a = 3, b = 2, c = 1$ 时，表达式 $f = a > b > c$ 的值是0。

2. 选择题

(1) 有如下程序，该程序的输出结果是 ()。

```
void main( )
{
    int x = 1, a = 0, b = 0;
    switch (x) {
        case 0: b++;
        case 1: a++;
        case 2: a++; b++;
    }
    printf ("a = %d, b = %d\n", a, b);
}
```

A. $a = 2, b = 1$ B. $a = 1, b = 1$ C. $a = 1, b = 0$ D. $a = 2, b = 2$

【答案】 A。

【解释】 因 x 的值是 1，所以 case 1 后面的语句 $a++$ 执行， a 的值增 1，但因后面无 break 子句，case 2 后的语句也要执行， a 的值再增 1， b 的值增 1，所以 a 和 b 的值最后分别为 2 和 1。故正确答案是 A。

(2) 若有如下程序，该程序的输出结果是 ()。

```
void main( )
{
```

```
float x = 2.0, y;  
if (x < 0.0) y = 0.0;  
else if (x < 10.0) y = 1.0 / x;  
else y = 1.0;  
printf ("%f\n", y);  
}
```

A. 0.000000 B. 0.250000 C. 0.500000 D. 1.000000

【答案】 C。

【解释】 因 x 的值是 2.0, 所以 $x < 0.0$ 为假, 则执行 `if(x < 0.0)` 后面的 `else` 语句, 继续判断 $x < 10.0$, 因该条件为真, 所以执行 $y = 1.0 / x$, 则 y 的值为 0.5。故正确答案为 C。

(3) 设有: `int a = 1, b = 2, c = 3, d = 4, m = 2, n = 2;` 执行 `(m=a>b)&&(n=c>d)` 后 n 的值是 ()。

A. 1 B. 2 C. 3 D. 4

【答案】 B。

【解释】 逻辑与 (`&&`) 运算具有左结合性, 所以先求 `(m=a>b)` 的值, 因关系运算符的优先级高于赋值运算符, 所以该表达式应理解为将 `a>b` 比较的结果 0 赋值给 m, 整个表达式的值也就是 0。对于逻辑与的运算特点, 左边表达式的值如果为假 (0), 则右边表达式不参与运算, 所以 n 的值不变。故正确答案是 B。

(4) 对 `if` 语句中表达式的类型, 下面正确的描述是 ()。

- A. 必须是关系表达式
- B. 必须是关系表达式或逻辑表达式
- C. 必须是关系表达式或算术表达式
- D. 可以是任意表达式

【答案】 D。

【解释】 `if` 语句中的表达式不局限于哪种类型的表达式, 只要表达式非零时, 表达式的值就为真, 否则就是假。所以正确答案是 D。

(5) 多重 `if_else` 语句嵌套使用时, 寻找与 `else` 配对的 `if` 方法是 ()。

- A. 缩排位置相同的 `if`
- B. 其上最近的 `if`
- C. 下面最近的 `if`
- D. 其上最近的未配对的 `if`

【答案】 D。

【解释】 C 语言规定, 在默认 `{ }` 时, `else` 总是和它上面离它最近的未配对的 `if` 配对。所以正确答案是 D。

(6) 以下错误的 `if` 语句是 ()。

- A. `if (x > y) z = x;`
- B. `if (x==y) z = 0;`
- C. `if (x != y) printf("%d", x) else printf("%d", y);`
- D. `if (x < y) { x++; y--; }`

【答案】 C。

【解释】 答案 C 中, 语句 `printf("%d", x)` 后面掉了分号 (;)。所以应选择 C。

(7) 以下程序的输出为 ()。

```
void main( )
{
    int a = 20, b = 30, c = 40;
    if (a > b) a = b,
    b = c; c = a;
    printf ("a = %d, b = %d, c = %d", a, b, c);
}
```

A. a = 20, b = 30, c = 20

B. a = 20, b = 40, c = 20

C. a = 30, b = 40, c = 20

D. a = 30, b = 40, c = 30

【答案】 A。

【解释】 `a = b, b = c;` 应看做是一条语句, 因 `a > b` 为假, 所以语句 `a = b, b = c;` 不执行, `c = a;` 语句则总是要执行的, 此时 `a`、`b` 的值不变 (分别为 20、30), `c` 的值为 `a` 的值 (20)。故正确答案为 A。

(8) 对于条件表达式 `(k) ? (i++) : (i--)` 来说, 其中的表达式 `k` 等价于 ()。

A. `k == 0`

B. `k == 1`

C. `k != 0`

D. `k != 1`

【答案】 C。

【解释】 对于该条件表达式应理解为: `k` 为真时返回 `i++`, 否则返回 `i--`。在 C 语言中表达式的值非 0 表示真, 为 0 表示假, 所以 `k` 为真应表示为 `k != 0`。故正确答案为 C。

(9) 下面程序运行结果为 ()。

```
void main( )
{
    char c = 'a';
    if ('a' < c <= 'z') printf ("LOW");
    else printf ("UP");
}
```

A. LOW

B. UP

C. LOWUP

D. 程序语法错误

【答案】 A。

【解释】 关系运算符具有左结合性, 所以表达式 `('a' < c <= 'z')`, 将先求 `'a' < c` 的值, 即为 0 (假), 再将求 `0 <= 'z'` 的值, 即为 1 (真), 所以执行 `if` 后面的语句 `printf ("LOW");` 故正确答案为 A。

(10) 对下述程序, 正确的判断是 ()。

```
void main( )
{
    int a, b;
    scanf ("%d, %d", &a, &b);
    if (a > b) a = b; b = a;
```



```
else a++; b++;  
printf ("%d, %d", a, b);  
}
```

- A. 有语法错误不能通过编译 B. 若输入 4, 5 则输出 5, 6
C. 若输入 5, 4 则输出 4, 5 D. 若输入 5, 4 则输出 5, 5

【答案】 A。

【解释】 在 if 和 else 之间如果有多条语句，则必须使用复合语句，否则程序语法错误。该程序 if 和 else 之间有两条语句 `a = b; b = a;`，但没有加 `{ }`，是错误的。故正确答案为 A。

(11) 逻辑运算符两侧运算对象的数据类型 ()。

- A. 只能是 0 或 1 B. 只能是 0 或非 0 正数
C. 只能是整型或字符型数据 D. 可以是任何类型的数据

【答案】 D。

【解释】 逻辑运算符两侧运算对象的值如果是 0，则表示假，非 0 就表示真，不管其类型是什么。故正确答案为 D。

(12) 以下关于运算符优先顺序的描述中正确的是 ()。

- A. 关系运算符 < 算术运算符 < 赋值运算符 < 逻辑运算符
B. 逻辑运算符 < 关系运算符 < 算术运算符 < 赋值运算符
C. 赋值运算符 < 逻辑运算符 < 关系运算符 < 算术运算符
D. 算术运算符 < 关系运算符 < 赋值运算符 < 逻辑运算符

【答案】 C。

【解释】 算术运算符的优先级高于关系运算符，关系运算符的优先级高于逻辑运算符，逻辑运算符的优先级则高于赋值运算符。例如：`a = x+y>2 && x+y<8;` 等价于 `a = ((x+y)>2) && ((x+y)<8);` 故正确答案为 C。

(13) 下列运算符中优先级最高的是 ()。

- A. < B. + C. && D. !=

【答案】 B。

【解释】 算术运算符的优先级高于关系运算符，关系运算符的优先级高于逻辑运算符。`+`是算术运算符，`<`和`!=`是关系运算符，`&&`是逻辑运算符。故正确答案为 B。

(14) 若希望当 A 的值为奇数时，表达式的值为“真”，A 的值为偶数时，表达式的值为“假”，则以下不能满足要求的表达式是 ()。

- A. `A % 2 == 1` B. `!(A % 2 == 0)` C. `!(A % 2)` D. `A % 2`

【答案】 C。

【解释】 当 A 为奇数时，`A % 2` 的值为 1 (真)，所以 `A % 2 == 1` 为真，`!(A % 2 == 0)` 也为真，而 `!(A % 2)` 得值却为假。当 A 为偶数时，`!(A % 2)` 得值却为真，其余均为假。故不能满足要求的表达式是 C。

(15) 判断 char 型变量 cl 是否为小写字母的正确表达式是 ()。

- A. `'a' <= cl <= 'z'` B. `(cl >= a) && (cl <= z)`

C. ('a' >= c1) || ('z' <= c1)

D. (c1 >= 'a') && (c1 <= 'z')

【答案】 D。

【解释】 字符的比较其实是比较它们的 ASCII 码值, 在 ASCII 码表中, 小写字母字符'a'~'z'是连续的, 其 ASCII 码值是逐渐增 1, 故判断 c1 是否为小写字母, 应判断 c1 是否大于或等于字符'a', 并且小于或等于字符'z'。故正确答案是 D。

(16) 已知 int x = 10, y = 20, z = 30; 以下语句执行后 x, y, z 的值是 ()。

```
if (x > y)
```

```
z = x; x = y; y = z;
```

A. x = 10, y = 20, z = 30

B. x = 20, y = 30, z = 30

C. x = 20, y = 30, z = 10

D. x = 20, y = 30, z = 20

【答案】 B。

【解释】 语句 z = x; 是否执行与 if 后面的条件是否为真有关, 条件为真该语句执行, 为假则不执行。而语句 x = y; y = z; 则与 if 后面的条件无关, 总要执行。因 x=10, y=20 所以 x > y 为假, 则语句 z = x; 不执行, 即 z 的值不变 (30), 语句 x = y; y = z; 执行后, x 的值为 y 的值 (20), y 的值为 z 的值 (30)。故正确答案是 B。

(17) 请阅读以下程序:

```
main( )
{
    int a = 5, b = 0, c = 0;
    if (a = b + c) printf ("***\n");
    else           printf ("$$$ \n");
}
```

以上程序 ()

A. 有语法错不能通过编译

B. 可以通过编译但不能通过连接

C. 输出***

D. 输出\$\$\$

【答案】 D。

【解释】 if 后面的表达式可以是任何类型的表达式, 当然可以是赋值表达式, 所以答案 A 和 B 是错误的。因赋值表达式的值为 0 (即为假), 所以执行 else 后的语句, 输出\$\$\$。故正确答案是 D。

(18) 请阅读以下程序, 其运行结果是 ()。

```
void main( )
{
    char c = 'A';
    if ('0' <= c <= '9') printf ("YES");
    else printf ("NO");
}
```

A. YES

B. NO

C. YESNO

D. 语句错误

【答案】 A。

【解释】 关系运算符具有左结合性，所以 $(0' \leq c \leq '9')$ 等价于 $((0' \leq c) \leq '9')$ 。因 $c = 'A'$ ，所以 $(0' \leq c)$ 的值为真，即为 1， $1 \leq '9'$ 的值为真，即为 1，所以执行 if 后面的语句，输出 YES。故正确答案是 A。

(19) 当 $a = 1, b = 3, c = 5, d = 4$ 时，执行完下面一段程序后 x 的值是 ()。

```
if (a < b)
if (c < d) x = 1;
else
    if (a < c)
        if (b < d) x = 2;
        else x = 3;
    else x = 6;
else x = 7;
```

A. 1

B. 2

C. 3

D. 6

【答案】 B。

【解释】 因 $a < b$ 为真，执行 $\text{if}(c < d)$ ，但 $c < d$ 为假，故执行 $\text{if}(a < c)$ ， $a < c$ 为真，执行 $\text{if}(b < d)$ ， $b < d$ 为真，执行 $x = 2$ ，所以 x 的值是 2。正确答案为 B。

(20) 已知 $x = 43, ch = 'A', y = 0$ ；则表达式 $(x >= y \&\& ch < 'B' \&\& !y)$ 的值是 ()。

A. 0

B. 语法错

C. 1

D. “假”

【答案】 C。

【解释】 表达式 $x >= y \&\& ch < 'B' \&\& !y$ 等价于 $(x >= y) \&\& (ch < 'B') \&\& (!y)$ ， $(x >= y)$ 的值为真， $(ch < 'B')$ 的值也为真（因 $'A' < 'B'$ ），而 $y = 0$ ，所以 $(!y)$ 的值也为真，所以整个表达式的值为真，即为 1。正确答案为 C。

(21) 有如下程序，正确的输出结果是 ()。

```
void main( )
{
    int a = 15, b = 21, m = 0;
    switch (a % 3)
    {
        case 0: m++; break;
        case 1: m++;
            switch (b % 2)
            {
                default: m++;
                case 0: m++; break;
            }
    }
    printf("%d\n", m);
}
```

- A. 1 B. 2 C. 3 D. 4

【答案】 A。

【解释】 a 的值为 15, 则 $a \% 3$ 的值为 0, 所以执行 case 0 后面的语句, 即 m 增 1 后, m 的值为 1, 执行 break 子句退出 switch。故正确答案为 A。

(22) 阅读以下程序, 如果从键盘上输入 5, 则正确的输出结果是 ()。

```
void main( )
{
    int x;
    scanf ("%d", &x);
    if (x-- < 5) printf ("%d", x);
    else printf ("%d", x++);
}
```

- A. 3 B. 4 C. 5 D. 6

【答案】 B。

【解释】 表达式 $x-- < 5$ 是先作 $x < 5$ 的比较, 然后 x 减 1, 因 x 的开始值为 5, 所以表达式的值为假, 执行 else 后的语句, 但此时 x 的值减 1 后变为 4, 执行语句 `printf ("%d", x++);` 是先输出 x 的值, 即 4, 然后将 x 增 1。故正确答案为 B。

(23) 若 a、b、c1、c2、x、y 均是整型变量, 正确的 switch 语句是 ()。

- | | |
|--|--|
| <p>A. switch (a + b);</p> <pre>{ case 1: y = a + b; break; case 0: y = a - b; break; case 3: y = b - a, break; }</pre> | <p>B. switch (a * a + b * b)</p> <pre>{ case 3: case 1: y = a + b; break; }</pre> |
| <p>C. switch a</p> <pre>{ case c1: y = a - b; break case c2: x = a * d; break default: x = a + b; }</pre> | <p>D. switch (a - b)</p> <pre>{ default: y = a * b; break case 3: case 4: x = a + b; break case 10: case 11: y = a - b; break; }</pre> |

【答案】 A。

【解释】 答案 B 中 case 3 后跟分号是错误的; 答案 C 中 switch 后面的表达式忘了小括号, 并且 break 子句后也忘了分号, 是错误的; 答案 D 中的前面两个 break 子句后忘了分号, 也是错误的。故正确答案为 A。

(24) 与 $y = (x > 0 ? 1 : x < 0 ? -1 : 0);$ 的功能相同的 if 语句是 ()。

- | | |
|---|--|
| <p>A. if (x > 0) y = 1;</p> <p>else if (x < 0) y = -1;</p> <p>else y = 0;</p> | <p>B. if (x)</p> <p>if (x > 0) y=1;</p> <p>else if (x < 0) y = -1;</p> |
|---|--|

```

C. y = -1
   if (x)
       if (x > 0) y=1;
       else if (x == 0) y = 0;
       else y = -1;

D. y = 0;
   if (x >= 0)
       if (x > 0) y = 1;
       else y = -1;

```

【答案】 A。

【解释】 表达式 $(x>0?1:x<0?-1:0)$ 的功能是：如果 x 大于 0 则返回 1，否则如果小于 0 则返回-1，再否则（即等于 0）返回 0。B、C、D 均是错误的，只有 A 正确。

(25) 若有定义：float w; int a, b; 则合法的 switch 语句是 ()。

```

A. switch (w) {
    case 1.0: printf ("*\n");
    case 2.0: printf ("**\n");
}

B. switch (a) {
    case 1 printf ("*\n");
    case 2 printf ("**\n");
}

C. switch (b) {
    case 1: printf ("*\n");
    default: printf ("\n");
    case a: printf ("**\n");
}

D. switch (a + b); {
    case 1: printf ("*\n");
    case 2: printf ("**\n");
    default: printf ("\n");
}

```

【答案】 B。

【解释】 switch 后面的“表达式”，可以是 int、char 和枚举型中的一种，不能是 float 型变量，所以 A 是非法的；case 后面必须是“常量表达式”，表达式中不能包含变量，所以 C 也是非法的；D 中 switch 后的表达式加了分号(;)是非法的。所以正确答案是 B。

3. 程序填空题

(1) 执行下面程序时，若从键盘上输入 8，则输出为 9，请填空。

```

void main( )
{
    int x;
    scanf ("%d", &x);
    if (____x++____ > 8)
        printf ("%d\n", ++x);
    else printf ("%d\n", x--);
}

```

(2) 执行下面程序时输出为 1，请填空。

```

void main( )
{
    int a = 4, b = 3, c = 2, d = 1;
    printf ("%d\n", (a < b ? a : d < c ? ____d____ : b));
}

```

(3) 下面的程序的功能是根据输入的百分制成绩 score，转换成相应的五分制成绩 grade 并打印输出。转换的标准为：当 $90 \leq \text{score} \leq 100$ 时，grade 为 A；当 $80 \leq \text{score} < 90$ 时，

grade 为 B; 当 $70 \leq \text{score} < 80$ 时, grade 为 C; 当 $60 \leq \text{score} < 70$ 时, grade 为 D; 当 $\text{score} < 60$ 时, grade 为 E; 请填空。

```
#include <stdio.h>
void main( )
{
    int score, mark;
    scanf ("%d", &score);
    mark = score / 10;
    switch (mark) {
        default: printf ("%d--E", score);
                break
        case 10:
        case 9 : printf ("%d--A", score); break;
        case 8 : printf ("%d--B", score); break;
        case 7 : printf ("%d--C", score); break;
        case 6 : printf ("%d--D", score); break;
    }
}
```

4. 编程题

(1) 编一程序判断输入整数的正负性和奇偶性。

【设计思想】 对一个整数 a 的正负性判断, 可将 a 与 0 比较来求得; 奇偶性判断, 可根据 $a \% 2$ 的值为 0 还是为 1 来求得。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int a;

    scanf ("%d", &a);
    if (a >= 0)
        printf ("the number %d is positive number\n", a);
    else
        printf ("the number %d is negative number\n", a);
    if (a % 2 == 0)
        printf ("the number %d is even number\n", a);
    else
        printf ("the number %d is odd number\n", a);
}
```

程序运行结果 (假设输入为: 9):

```
the number 9 is positive number
the number 9 is odd number
```

(2) 编程判断输入数据的符号属性。

$$\text{sign} = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

输入 x，打印出 sign 的值。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int x, sign;

    scanf ("%d", &x);
    if (x > 0)
        sign = 1;
    else if (x == 0)
        sign = 0;
    else
        sign = -1;
    printf ("sign = %d\n", sign);
}
```

程序运行结果（假设输入为：9↵）：

```
sign = 1
```

(3) 输入任意三个数 num1、num2、num3，按从小到大的顺序排序输出。

【设计思想】 使用 if-else 结构将三个数两两之间进行比较。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int num1, num2, num3;

    scanf ("%d%d%d", &num1, &num2, &num3);
```

```

if (num1 <= num2)
    if (num2 <= num3)
        printf ("%d %d %d\n", num1, num2, num3);
    else
        if (num1 >= num3)
            printf ("%d %d %d\n", num3, num1, num2);
        else
            printf ("%d %d %d\n", num1, num3, num2);
else
    if (num2 >= num3)
        printf ("%d %d %d\n", num3, num2, num1);
    else
        if (num1 >= num3)
            printf ("%d %d %d\n", num2, num3, num1);
        else
            printf ("%d %d %d\n", num2, num1, num3);
}

```

程序运行结果（假设输入为 9 5 10✓）：

5 9 10

（4）在屏幕上显示一张如下所示的时间表：

*****Time*****

1 morning

2 afternoon

3 night

Please enter your choice:

操作人员根据提示进行选择，程序根据输入的时间序号显示相应的问候信息，选择 1 时显示“Good morning”，选择 2 时显示“Good afternoon”，选择 3 时显示“Good night”，对于其他的选择显示“Selection error!”，用 switch 语句编程实现。

【设计思想】 使用 printf 函数输出一个简单的菜单语句，通过 switch 语句进行选择。需要注意的是，输入选项是字符型的，所以在 case 后的数字要加单引号。

【参考答案】

```

#include <stdio.h>

void main( )
{
    char c;

    printf ("*****Time*****\n");

```



```
printf ("1 morning\n");
printf ("2 afternoon\n");
printf ("3 night\n");
printf ("Please enter your choice: ");

c = getchar( );
switch (c)
{
    case '1': printf ("Good morning\n");
              break;
    case '2': printf ("Good afternoon\n");
              break;
    case '3': printf ("Good night\n");
              break;
    default : printf ("Selection error!\n");
}
}
```

程序运行结果:

```
*****Time*****
1 morning
2 afternoon
3 night
Please enter your choice: 1✓
Good morning
```

(5) 输入一个年份和月份, 打印出该月份有多少天 (考虑闰年), 用 `switch` 语句编程。

【设计思想】 每年中 1、3、5、7、8、10、12 月份有 31 天, 4、6、9、11 月份有 30 天, 对于 2 月份来说, 闰年有 29 天, 平年有 28 天, 所以应判断年份是否为闰年。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int year, month;

    printf ("Input year,month: ");
    scanf ("%d,%d", &year, &month);

    switch (month)
```

```
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12: printf ("31 days\n");
            break;

    case 4:
    case 6:
    case 9:
    case 11: printf ("30 days\n");
            break;
    case 2:  if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
                printf ("29 days\n"); //闰年的2月有29天
            else
                printf ("28 days\n"); //平年的2月有28天
            break;
    default : printf ("Input error!\n");
}
}
```

程序运行结果:

```
Input year,month: 2005,2✓
28 says
Input year,month: 1976,2✓
29 days
```

1.6 习题 6 参考答案和解释

1. 填空题

(1) C 语言中实现循环结构的控制语句有 for 语句、while 语句和 do-while 语句。

(2) 当循环体内遇到 break、return 或 goto 语句时，将退出循环。

(3) do-while 语句和 while 语句的区别在于 do-while 是先执行后判断，因此 do-while 至少要执行一次循环体。而 while 是先判断后执行，如果条件不满足，则循环体语句一次也不执行。

(4) break 语句在循环体中的作用是 跳出循环结构，转而执行循环语句后的第一条语句，continue 语句在循环体中的作用是 结束本次循环，继续执行下一次循环。

(5) goto 语句可与条件语句配合使用, 构成循环。

(6) 如果循环次数在执行循环体之前就已确定, 一般用for循环; 如果循环次数是由循环体的执行情况确定的, 一般用while循环或者do-while循环。当循环体至少执行一次时, 用do-while循环, 反之, 如果循环体可能一次也不执行, 选用while循环。

2. 选择题

(1) 若 i 为整型变量, 则以下循环执行次数是 ()。

```
for (i = 2; i != 0;)    printf ("%d", i--);
```

A. 无限次 B. 0 次 C. 1 次 D. 2 次

【答案】 D。

【解释】 第一次输出 2 后, i 的值减 1 变为 1, 不等于 0, 再循环, 输出 1 后, i 的值减 1 变为 0, 循环结束, 所以总共循环 2 次。正确答案为 D。

(2) 下面程序的功能是把 316 表示为两个加数的和, 使两个加数分别能被 13 和 11 整除, 请选择填空。

```
#include <stdio.h>
void main( )
{
    int i = 0, j, k;
    do { i++; k = 316 - 13*i; } while ( _____ );
    j = k / 11;
    printf ("316 = 13 * %d + 11 * %d", i, j);
}
```

A. k / 11 B. k % 11 C. k / 11 == 0 D. k % 11 == 0

【答案】 B。

【解释】 程序的设计思想是每次将 316 减去 13 的倍数, 然后判断其差值是否被 11 整除, 如果不能整除, 则继续循环, 直到能被 11 整除为止。所以正确答案为 B。

(3) 下面程序的运行结果是 ()。

```
#include <stdio.h>
void main( )
{
    int y = 10;
    do { y--; } while (--y);
    printf ("%d\n", y--);
}
```

A. -1 B. 1 C. 8 D. 0

【答案】 D。

【解释】 do-while 循环是直到 y 变为 0 时才结束, 所以输出的结果是 0, 当然, 最

后 y 的值是-1。故正确答案为 D。

(4) 若运行以下程序时, 从键盘输入 ADescriptor↵(↵表示回车), 则下面程序的运行结果是()。

```
#include <stdio.h>
void main( )
{
    char c;
    int v0 = 0, v1 = 0, v2 = 0;
    do {
        switch (c = getchar( )) {
            case 'a' : case 'A' :
            case 'e' : case 'E' :
            case 'i' : case 'I' :
            case 'o' : case 'O' :
            case 'u' : case 'U' : v1 += 1;
            default: v0 += 1; v2 += 1;
        }
    } while (c != '\n');
    printf ("v0 = %d, v1 = %d, v2 = %d\n", v0, v1, v2);
}
```

A. v0 = 7, v1 = 4, v2 = 7

B. v0 = 8, v1 = 4, v2 = 8

C. v0 = 11, v1 = 4, v2 = 11

D. v0 = 12, v1 = 4, v2 = 12

【答案】 D。

【解释】 因 case 后面都没有 break 语句, 所以进入 case 后, 除了 v1 增 1 以外, 还要执行 default 后的语句, v0 和 v2 都要增 1。输入的字符串 ADescriptor↵中 A、e、i、o 各出现一次, 所以 v1 要增加 4 次, 其值为 4, 字符串中字符总数为 12 (包含回车符), 所以 v0 和 v2 要增加 12 次, 其值分别为 12、12。故正确答案为 D。

(5) 下面程序的运行结果是()。

```
#include <stdio.h>
void main( )
{
    int a = 1, b = 10;
    do
    { b -= a; a++; } while (b-- < 0);
    printf("a = %d, b = %d\n", a, b);
}
```

A. a = 3, b = 11

B. a = 2, b = 8

C. a = 1, b = -1

D. a = 4, b = 9

【答案】 B。

【解释】 因 a、b 的初始值分别为 1 和 10, 进入循环体后, b 的值为 10-1, 即 9, a

增 1 变为 2, 再对 `while` 后面的表达式进行检测, 此时先判断 `b<0` 为假, 循环结束, 再将 `b` 减 1, 即 `b` 变为 8。故正确答案为 B。

(6) 设有程序段:

```
int k = 10;
while (k = 0) k = k - 1;
```

则下面描述中正确的是 ()。

- A. `while` 循环执行 10 次
- B. 循环是无限循环
- C. 循环体语句一次也不执行
- D. 循环体语句执行一次

【答案】 C。

【解释】 因 `while` 后面的表达式是赋值表达式, 其值为 0 (即为假), 所以循环体内的语句不可能执行。故正确答案是 C。

(7) 设有以下程序段:

```
int x = 0, s = 0;
while (!x != 0) s += ++x;
printf ("%d", s);
```

则 ()。

- A. 运行程序段后输出 0
- B. 运行程序段后输出 1
- C. 程序段中的控制表达式是非法的
- D. 程序段执行无限次

【答案】 B。

【解释】 `x` 的初始值为 0, 所以 `!x` 为 1, `!x != 0` 条件成立, 进入循环体, 先将 `x` 增 1, `x` 的值变为 1, 然后将 `s` 的值与 `x` 的值相加的结果 (即为 1) 赋值给 `s`, `s` 的值变为 1, 再对 `while` 后的条件进行判断, 此时 `x` 为 1, 所以 `!x` 为 0, 则 `!x != 0` 条件不成立, 循环结束, 输出 `s` 的值为 1。故正确答案是 B。

(8) 语句 `while (!E);` 中的表达式 `!E` 等价于 ()。

- A. `E == 0`
- B. `E != 1`
- C. `E != 0`
- D. `E == 1`

【答案】 A。

【解释】 对于 `while (!E)` 来讲, 只有 `E` 为 0, 条件才为真, 所以其等价于 `while (E == 0)`。故正确答案是 A。

(9) 下面程序段的运行结果是 ()。

```
a = 1; b = 2; c = 2;
while (a < b < c) { t = a; a = b; b = t; c--; }
printf ("%d, %d, %d", a, b, c);
```

- A. 1, 2, 0
- B. 2, 1, 0
- C. 1, 2, 1
- D. 2, 1, 1

【答案】 A。

【解释】 第一次循环时, 条件 `(a<b<c)` 相当于 `(1<2<2)`, 其值为真 (左结合性), 进入循环体, 先将 `a`、`b` 的值交换 (`t = a; a = b; b = t;`), 然后将 `c` 的值减 1, 此时 `a=2, b=1, c=1`。

第二次循环时, 条件 ($a < b < c$) 相当于 ($2 < 1 < 1$), 其值为真, 进入循环体, a 、 b 的值交换, c 减 1, 此时 $a=1, b=2, c=0$ 。第三次循环时, 条件 ($a < b < c$) 相当于 ($1 < 2 < 0$), 其值为假, 循环结束, 所以正确答案是 A。

(10) 下面程序段的运行结果是 ()。

```
x = y = 0;
while (x < 15) y++, x += ++y;
printf ("%d, %d", y, x);
```

- A. 20, 7 B. 6, 12 C. 20, 8 D. 8, 20

【答案】 D。

【解释】 第一次循环, x 为 0, $x < 15$ 为真, 进入循环体, y 的值变为 2, x 的值变为 2; 第二次循环, $x < 15$ 为真, 进入循环体, y 的值变为 4, x 的值变为 6; 第三次循环, $x < 15$ 为真, 进入循环体, y 的值变为 6, x 的值变为 12; 第四次循环, $x < 15$ 为真, 进入循环体, y 的值变为 8, x 的值变为 20; 第五次循环, $x < 15$ 为假, 退出循环。故正确答案是 D。

(11) 对 `for(表达式 1; 表达式 3)` 可理解为 ()。

- A. `for(表达式 1; 0; 表达式 3)` B. `for(表达式 1; 1; 表达式 3)`
C. `for(表达式 1; 表达式 1; 表达式 3)` D. `for(表达式 1; 表达式 3; 表达式 3)`

【答案】 B。

【解释】 表达式 2 的省略意味着其值恒为真。故正确答案是 B。

(12) 以下程序段 ()。

```
x = -1;
do { x = x * x; } while (!x);
```

- A. 是死循环 B. 循环执行二次
C. 循环执行一次 D. 有语法错误

【答案】 C。

【解释】 `do-while` 循环结构中的循环体至少要执行一次, 执行循环体中的语句后, x 的值为 1, 再对 `while` 后的表达式进行计算, 因 $!x$ 的值为 0, 所以循环结束, 即整个循环只执行了一次。故正确答案是 C。

(13) 以下 `for` 循环的执行次数是 ()。

```
for (x = 0, y = 0; (y = 123) && (x < 4); x++);
```

- A. 是无限循环 B. 循环次数不定 C. 4 次 D. 3 次

【答案】 C。

【解释】 决定 `for` 循环次数的表达式是 $(y = 123) \&\& (x < 4)$, 而 $(y = 123)$ 表达式恒为真, 所以该表达式等价于 $(x < 4)$, 即 x 的值决定循环的次数, 因每次循环后 x 增 1, 所以当 x 增到 4 时, $(x < 4)$ 就为假, 循环结束, 所以循环次数是 4 次。故正确答案是 C。

(14) 以下不是无限循环的语句为 ()。

- A. for (y = 0, x = 1; x > ++y; x = i++) i = x; B. for(;; x++ = i);
C. while (1) { x++; } D. for (i = 10; ; i--) sum += i;

【答案】 A。

【解释】 对于 A 来讲, 条件判断 $x > ++y$ 的值为假 (因 x 的值为 1, $++y$ 的值为 1), 循环结束。对于 B 和 D 来讲, 因为表达式 2 省略了, 则相当于恒为真, 所以是死循环, 对于 C 来说, while 后表达式的值为 1, 恒为真, 也是死循环。故正确答案是 A。

(15) 下面有关 for 循环的正确描述是 ()。

- A. for 循环只能用于循环次数已经确定的情况
B. for 循环是先执行循环体语句, 后判断表达式
C. 在 for 循环中, 不能用 break 语句跳出循环体
D. for 循环的循环体语句中, 可以包含多条语句, 但必须用花括号括起来

【答案】 D。

【解释】 for 循环常用于循环次数已知的循环中, 但也可以用于循环次数未知的循环中; for 循环是先判断表达式, 根据表达式的值来决定是否循环; 在 for 循环中如果要中途退出循环, 可以使用 break 语句来实现。所以 A、B、C 都是错误的, 正确答案是 D。

(16) 设有程序段:

```
t = 0;
while (printf("*")) { t++; if (t<3) break; }
```

下面描述正确的是 ()。

- A. 其中循环控制表达式与 0 等价 B. 其中循环控制表达式与 '0' 等价
C. 其中循环控制表达式是不合法的 D. 以上说法都不对

【答案】 B。

【解释】 因 printf("*") 函数调用的返回值时字符串中字符的个数, 即为 1。所以 while 后面的条件恒为真, 所以循环控制表达式与 '0' 是等价的。正确答案是 B。

(17) 以下描述中正确的是 ()。

- A. 由于 do-while 循环中循环体语句只能是一条可执行语句, 所以循环体内不能使用复合语句
B. do-while 循环由 do 开始, 用 while 结束, 在 while(表达式) 后面不能写分号
C. 在 do-while 循环体中, 一定要有能使 while 后面表达式的值变为零 (“假”) 的操作
D. do-while 循环中, 根据情况可以省略 while

【答案】 C。

【解释】 do-while 循环中循环体语句如果多余一条则可使用复合语句; do-while 循环由 do 开始, 用 while 结束, while 不能省略, 在 while(表达式) 后面必须写分号; 为了保证循环正常结束, 通常在循环体中对循环控制变量进行更改, 以使得 while 后面的值变为零。所以正确答案是 C。

(18) 有以下程序段:

```
int n = 0, p;  
do { scanf ("%d", &p); n++; } while (p != 12345 && n < 3);
```

此处 do-while 循环的结束条件是 ()。

- A. P 的值不等于 12345 并且 n 的值小于 3
- B. P 的值等于 12345 并且 n 的值大于等于 3
- C. P 的值不等于 12345 或者 n 的值小于 3
- D. P 的值等于 12345 或者 n 的值大于等于 3

【答案】 D。

【解释】 while 后面的条件为真是循环继续的条件, 要想循环结束, 则 while 后面的条件必须为假, 即原条件求反, 也就是 $(p == 12345 \parallel n \geq 3)$ 。所以正确答案是 D。

(19) 以下程序的输出结果是 ()。

```
void main( )  
{  
    int a, b;  
    for (a = 1, b = 1; a <= 100; a++)  
    {  
        if (b >= 10) break;  
        if (b % 3 == 1) { b += 3; continue; }  
    }  
    printf ("%d\n", a);  
}
```

- A. 101
- B. 6
- C. 5
- D. 4

【答案】 D。

【解释】 第一次循环: $a=1, b=1, a \leq 100$ 为真, 进入循环, 使得 $b=4$; 第二次循环: $a=2, b=4, a \leq 100$ 为真, 进入循环, 使得 $b=7$; 第三次循环: $a=3, b=7, a \leq 100$ 为真, 进入循环, 使得 $b=10$; 第四次循环: $a=4, b=10, a \leq 100$ 为真, 进入循环, 此时因 $b \geq 10$ 成立, break 退出循环执行 printf 语句, 所以显示 a 的值为 4。正确答案是 D。

(20) 以下程序中, while 循环的循环次数是 ()。

```
int i = 0;  
while (i < 10)  
{  
    if (i < 1) continue;  
    if (i == 5) break;  
    i++;  
}
```

- A. 1
- B. 10
- C. 6
- D. 死循环, 不能确定次数

【答案】 D。

【解释】 因 i 的初始值为 0，所以 while 后面的条件为真，进入循环体，if 后面的条件 i<1 成立，执行 continue 语句，继续对 while 后的条件进行判断，因为此时对变量 i 的值没有任何变化，所以条件总是成立，循环将无限进行下去。所以正确答案是 D。

3. 程序填空题

(1) 下面程序段是从键盘输入的字符中统计数字字符的个数，用换行符结束循环。请填空。

```
int n = 0, c;
c = getchar( );
while ( ____ c != '\n' ____ )
{
    if ( ____ c >= '0' && c <= '9' ____ )    n++;
    c = getchar( );
}
```

【设计思想】 首先判断输入的字符是否为回车换行符 ('\n')，如果是，则循环结束，否则进入循环体，因为是统计输入字符中数字字符的个数，数字字符必须是在 '0'~'9' 范围内。

(2) 下面程序的功能是：输出 100 以内能被 3 整除且个位数为 6 的所有整数，请填空。

```
#include <stdio.h>
void main( )
{
    int i, j;
    for (i = 0; ____ i < 10 ____ ; i++)
    {
        j = i * 10 + 6;
        if ( ____ j % 3 ____ ) continue;
        printf ("%d", j);
    }
}
```

【设计思想】 只需对 100 以内的数 6、16、26、36、46、56、66、76、86、96 进行检查，判定其是否被 3 整除即可，如果不被 3 整除，则检查下一个数，否则输出该数。

(3) 下面程序的功能是：求 Fibonacci（斐波纳契）数列的前 40 个数，并按照 4 列一行输出（Fibonacci 数列有如下特点：第一、二个数都是 1，从第三个数开始，每个数都是前面两个数的和）。请填空。

```
#include <stdio.h>
void main( )
{
```

```
long f1 = 1, f2 = 1;
int i;
for (i = 1; i <= 20; i++)
{
    printf ("%12d%-12d", f1, f2);
    if ( i % 2 == 0 ) printf ("\n");
    f1 = f1 + f2;
    f2 = f1 + f2;
}
}
```

【设计思想】i 的初始值为 1，因为每次循环打印出 2 列数据，所以打印完 4 列数据后 i 的值为偶数，这时要换行，然后将 f1 和 f2 的值用前两项的值之和进行替换。

4. 编程题

(1) 编程计算 $2+4+6+\cdots+98+100$ 的值。

【参考答案】

算法 1 利用 for 循环语句实现，在循环体外为 sum 赋初值 0。

```
#include <stdio.h>

void main( )
{
    int i, sum = 0;

    for (i = 2; i <= 100; i += 2)
        sum += i;
    printf ("sum = %d\n", sum);
}
```

算法 2 利用 while 循环语句实现，在循环体外为 i 和 sum 赋初值。

```
#include <stdio.h>

void main( )
{
    int i = 2, sum = 0;

    while (i <= 100)
    {
        sum += i;
        i += 2;
    }
}
```

```
printf ("sum = %d\n", sum);  
}
```

程序运行结果:

```
sum = 2550
```

(2) 编程计算 $1*2*3+3*4*5+\cdots+99*100*101$ 的值。

【设计思想】 用累加和算法, 通项公式为 $\text{term}=i*(i+1)*(i+2)$; $i=1,3,\cdots,99$; 或者公式为 $\text{term}=(i-1)*i*(i+1)$; $i=2,4,\cdots,100$; 步长为 2。

【参考答案】

```
#include <stdio.h>  
  
void main( )  
{  
    int i;  
    long term, sum = 0;  
  
    for (i = 1; i <= 99; i += 2)  
    {  
        term = i * (i + 1) * (i + 2);  
        sum += term;  
    }  
    printf ("sum = %ld\n", sum);  
}
```

程序运行结果:

```
sum = 13002450
```

(3) 编程计算 $1!+2!+3!+\cdots+10!$ 的值。

【参考答案】

算法 1 用累加和算法, 累加项为 $\text{term}=\text{term}*i$; $i=1,2,\cdots,10$ 。term 的初始值为 1, 使用单重循环完成。

```
#include <stdio.h>  
  
void main( )  
{  
    long term = 1, sum = 0;  
    int i;
```

```
    for (i = 1; i <= 10; i++)
    {
        term *= i;
        sum += term;
    }
    printf ("1!+2!+...+10! = %ld\n", sum);
}
```

算法 2 用内层循环求阶乘，外层循环控制累加的项数。

```
#include <stdio.h>

void main( )
{
    long term, sum = 0;
    int i, j;

    for (i = 1; i <= 10; i++)
    {
        term = 1;
        for (j = 1; j <= i; j++)
            term *= j;
        sum += term;
    }
    printf ("1!+2!+...+10! = %ld\n", sum);
}
```

程序运行结果:

1!+2!+...+10! = 4037913

(4) 编程计算 $a+aa+aaa+\cdots+aa\cdots a$ (n 个 a) 的值, n 和 a 的值由键盘输入。

【设计思想】 用累加和算法, 累加项为 $term=term*10+a$; $i=1,2,\cdots,n$; $term$ 初值为 0。

【参考答案】

```
#include <stdio.h>

void main( )
{
    long term = 0, sum = 0;
    int a, i, n;
```

```

printf ("Input a, n: ");
scanf ("%d,%d", &a, &n);      //输入a, n的值

for (i = 1; i <= n; i++)
{
    term = term * 10 + a;      //求出累加项
    sum += term;               //进行累加
}
printf ("sum = %ld\n", sum);
}

```

程序运行结果（假如 a 和 n 的值输入为 2,4✓）:

```
sum = 2468
```

(5) 利用 $\frac{\pi}{2} \approx \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \cdots$ 前 100 项之积计算 π 的值。

【设计思想】 采用累乘积算法，累乘项为 $\text{term} = n * n / ((n-1) * (n+1))$; $n=2,4,\cdots,100$; 步长为 2。

【参考答案】

```

#include <stdio.h>

void main( )
{
    float term, result = 1; //累乘项初值应为1
    int n;

    for (n = 2; n <= 100; n += 2)
    {
        term = (float)(n * n) / ((n - 1) * (n + 1)); //计算累乘项
        result *= term;
    }
    printf ("result = %f\n", 2 * result);
}

```

程序运行结果:

```
result = 3.126079
```

(6) 利用泰勒级数 $\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$ ，计算 $\sin(x)$ 的值。要求最后一

项的绝对值小于 10^{-5} ，并统计出此时累计了多少项。

【设计思想】 x 由键盘输入，采用累加和算法 $\text{sum}=\text{sum}+\text{term}$ ， sum 的初值为 x ，利用前项求后项的方法计算累加项： $\text{term}=-\text{term}*x*x/((n+1)*(n+2))$ ； term 初值为 x ， n 初值为 1， $n=n+2$ 。

【参考答案】

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

void main( )
{
    int n = 1, count = 0;
    float x;
    double sum, term;

    printf ("Input x: ");
    scanf ("%f", &x);
    sum = x;
    term = x;    //赋初始值
    do
    {
        term = -term * x * x / ((n+1) * (n+2)); //计算相应项,并改相应符号
        sum += term; //累加
        n += 2;
        count++;
    } while (fabs(term) >= 1e-5);
    printf ("sin(x) = %f, count = %d\n", sum, count);
}
```

程序运行结果:

```
Input x: 3✓
sin(x) = 0.141120, count = 8
```

(7) 打印所有的“水仙花数”。所谓“水仙花数”是指一个三位数，其各位数字的立方和等于该数本身。例如，153 是“水仙花数”，因为 $153=1^3+3^3+5^3$ 。

【设计思想】 首先确定水仙花数 n 可能存在的范围，因为 n 是一个三位数，所以范围确定为 n 从 100 变化到 999，分离出 n 的百位 i 、十位 j 、个位 k 后，只要判断 n 是否等于 $i*i*i+j*j*j+k*k*k$ 即可知道 n 是否是水仙花数。

【参考答案】

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

```
void main( )
{
    int i, j, k, n;

    printf ("result is: ");
    for (n = 100; n < 1000; n++)
    {
        i = n / 100;           //分出百位
        j = (n - i *100) / 10; //分出十位
        k = n % 10;           //分出个位
        if (n == i*i*i + j*j*j + k*k*k)
            printf ("%d ", n);
    }
    printf ("\n");
}
```

程序运行结果:

```
result is: 153 370 371 407
```

(8) 从键盘上任意输入一个整数 x , 编程计算 x 的每一位数字相加之和 (忽略整数前的正负号)。例如, 输入 x 为 1234, 则由 1234 分离出 1、2、3、4 四个数字, 然后计算 $1+2+3+4=10$, 并输出 10。

【设计思想】 对输入的整数取绝对值, 即可实现忽略整数前的正负号。

【参考答案】

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

void main( )
{
    int i1, i2, i3, i4, k, n;

    printf ("Input data is: ");
    scanf ("%d", &n);
    k = abs(n);           //求绝对值
    i1 = k / 1000;        //分离出千位
    i2 = (k - i1 * 1000) / 100; //分离出百位
    i3 = (k - i1 * 1000 - i2 * 100) / 10; //分离出十位
    i4 = k % 10;          //分离出个位

    printf ("The sum of the toal bit is %d\n", i1 + i2 + i3 +i4);
}
```

}

程序运行结果:

```
Input data is: 1234✓
The sum of the toal bit is 10
```

(9) 从键盘上输入任意正整数，编程判断该数是否为回文数。所谓回文数就是从左到右读这个数与从右到左读这个数是一样的。例如，12321、4004 都是回文数。

【设计思想】 将该整数按照从最低位到最高位进行分离，然后重新组合成一整数，再将该整数与原来的整数比较，如果相等，则为回文数，否则不是。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int n, m = 0, s, r;

    printf ("Input data is: ");
    scanf ("%d", &n);
    s = n;
    while (s != 0)
    {
        r = s % 10;           //从低位到高位逐一分离
        m = 10 * m + r;       //重新组合一整数
        s = s / 10;           //求其商
    }
    if (m == n)
        printf ("yes\n");
    else
        printf ("no\n");
}
```

程序运行结果:

```
Input data is: 12321✓
yes
```

(10) 用 1 元 5 角钱人民币兑换 5 分、2 分和 1 分的硬币（每一种都要有）共 100 枚，问共有几种兑换方案？每种方案各换多少枚？

【设计思想】 设 5 分、2 分和 1 分的硬币各换 x、y、z 枚，依题意有 $x+y+z=100$ ，

$5x+2y+z=150$ ，由于每一种硬币都要有，故 5 分硬币最多可换 28 枚，2 分硬币最多可换 73 枚，1 分硬币可换 $100-x-y$ 枚， x 、 y 、 z 只需满足第二个方程即可打印，对每一组满足条件的 x 、 y 、 z 值用计数器计数即可得到兑换方案的数目。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int x, y, z, count = 0;

    for (x = 1; x <= 28; x++)
        for (y = 1; y <= 73; y++)
        {
            z = 100 - x - y;
            if (5*x + 2*y + z == 150)
            {
                count++;
                printf ("%02d, %02d, %02d  ", x, y, z);
                if(count % 6 == 0)
                    printf ("\n");
            }
        }
    printf ("count = %d\n", count);
}
```

程序运行结果：

01,46,53	02,42,56	03,38,59	04,34,62	05,30,65	06,26,68
07,22,71	08,18,74	09,14,77	10,10,80	11,06,83	12,02,86
count = 12					

(11) 某学校有 4 位同学中的一位做了好事，不留名，表扬信来了之后，校长问这 4 位是谁做的好事。4 人的回答是：

A 说：不是我。

B 说：是 C。

C 说：是 D。

D 说：他胡说。

已知 3 个人说的是真话，一个人说的是假话。现在问做好事者到底是谁？

【设计思想】 将 A、B、C、D 4 个人的回答用一条件表达式来表示，对于 A 的回答： $thisman \neq 'A'$ ；对于 B 的回答： $thisman == 'C'$ ；对于 C 的回答： $thisman == 'D'$ ；对于 D

的回答: `thisman != 'D'`; 然后, 采用枚举的方法, 一个人一个的去试, 如果这 4 个条件表达式中有 3 个为真, 即 4 个条件表达式的值相加为 3, 则可判定是其中某人做的好事。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int k = 0, sum = 0, g = 0;
    char thisman = ' ';

    for (k = 0; k <= 3; k++)
    {
        thisman = 'A' + k;
        sum = (thisman != 'A') + (thisman == 'C') + (thisman == 'D') + (thisman != 'D');
        if (sum == 3)
        {
            printf ("This man is %c\n", thisman);
            g = 1;
        }
    }
    if (g != 1)
        printf ("Can't found\n");
}
```

程序运行结果:

```
This man is C
```

(12) 编程打印如下图案。



【参考答案】 ①

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

```

void main( )
{
    int i, j;

    for (i = 1; i <= 3; i++)                //先显示前三行
    {
        for (j = 1; j <= i - 1; j++)        //输出每行前的空格
            printf ( " ");
        for (j = 1; j <= 5-2*(i-1); j++)    //输出每行的*号
            printf ("*");
        printf ("\n");
    }
    for (i = 1; i <= 2; i++)                //后显示后两行
    {
        for (j = 1; j <= 2 - i; j++)
            printf ( " ");
        for (j = 1; j <= 2*i+1; j++)
            printf ("*");
        printf ("\n");
    }
}

```

【参考答案】 ②

```

#include <stdio.h>

void main( )
{
    int i, j, k;

    for (i = 1; i <= 5; i++)                //i控制行数
    {
        for (j = 1; j <= 5 - i; j++)        //随行数的增加，输出递减数目的空格
            printf ( " ");
        for (k = 1; k <= 5; k++)            printf ( " ");
    }
}

```

```
#include <stdio.h>

void main( )
{
    int i, j;

    for (i = 1; i <= 5; i++)
    {
        for (j = 1; j <= 2*(i-1)+1; j++)
            printf ("*");
        printf ("\n");
    }
}
```

1.7 习题 7 参考答案和解释

1. 选择题

(1) 在 C 语言中, 引用数组元素时, 其数组下标的数据类型允许是 ()。

- A. 整型常量
- B. 整型表达式
- C. 整型常量或整型表达式
- D. 任何类型的表达式

【答案】 C。

【解释】 数组元素引用时, 其数组下标的数据类型只能是整型常量或整型表达式。所以正确答案是 C。

(2) 若有说明: `int a[10];` 则对数组元素的正确引用是 ()。

- A. `a[10];`
- B. `a[3.5]`
- C. `a(5)`
- D. `a[10-10]`

【答案】 D。

【解释】 因为定义的数组包含 10 个元素, 对数组元素引用的下标是从 0 到 9。显然 `a[10]` 超出了数组的范围, A 是错误的引用。因数组下标的数据类型必须是整型, 所以 B 是错误的。数组引用应使用中括号, 小括号是非法的, 所以 C 也是错误的。故正确答案是 D, 即引用 `a[0]`。

(3) 设有数组定义: `char array [] = "China";`, 则数组 `array` 所占的空间为 ()。

- A. 4 个字节
- B. 5 个字节
- C. 6 个字节
- D. 7 个字节

【答案】 C。

【解释】 因为字符串 "China" 中有 5 个字符, 要占 5 个字节的内存单元, 再加上字符串的结尾符 '\0', 所以整个数组应占用 6 个字节的内存单元。正确答案是 C。

(4) 若二维数组 `a` 有 `m` 列, 则在 `a[i][j]` 前的元素个数为 ()。

- A. $j * m + i$
- B. $i * m + j$
- C. $i * m + j - 1$
- D. $i * m + j + 1$

【答案】 B。

【解释】 对于元素 $a[i][j]$ ，其前面有 i 行，本行前面有 j 列（注意：行、列均是从 0 开始），所以其前元素的个数为 $i*m+j$ 。正确答案是 B。

(5) 若有说明： $\text{int } a[][3] = \{1, 2, 3, 4, 5, 6, 7\};$ ，则 a 数组第一维的大小是（ ）。

- A. 2 B. 3 C. 4 D. 无确定值

【答案】 B。

【解释】 因为二维数组 a 每一行有 3 个元素，而赋初始值有 7 个元素，所以其行数应为 3。正确答案是 B。

(6) 以下不正确的定义语句是（ ）。

- A. $\text{double } x[5] = \{2.0, 4.0, 6.0, 8.0, 10.0\};$
B. $\text{int } y[5] = \{0, 1, 3, 5, 7, 9\};$
C. $\text{char } c1[] = \{'1', '2', '3', '4', '5'\};$
D. $\text{char } c2[] = \{'\x10', '\xa', '\x8'\};$

【答案】 B。

【解释】 对数组赋初值时，初值的个数不能超过数组的大小。因为 B 中定义的数组 y 其大小为 5，而赋初值时包含了 6 个值，所以 B 定义是非法的。

(7) 以下不能对二维数组 a 进行正确初始化的语句是（ ）。

- A. $\text{int } a[2][3] = \{0\};$
B. $\text{int } a[][3] = \{\{1, 2\}, \{0\}\};$
C. $\text{int } a[2][3] = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\};$
D. $\text{int } a[][3] = \{1, 2, 3, 4, 5, 6\};$

【答案】 C。

【解释】 C 中定义的二维数组是 2 行 3 列的数组，但赋初值时给了 3 行的数据，超出了数组的范围，是非法的。所以答案是 C。

(8) 以下能对二维数组 a 进行正确初始化的语句是（ ）。

- A. $\text{int } a[2][] = \{\{1, 0, 1\}, \{5, 2, 3\}\};$ B. $\text{int } a[][3] = \{\{1, 2, 3\}, \{4, 5, 6\}\};$
C. $\text{int } a[2][4] = \{\{1, 2, 3\}, \{4, 5\}, \{6\}\};$ D. $\text{int } a[][3] = \{\{1, 0, 1\}, \{\}, \{1, 1\}\};$

【答案】 B。

【解释】 定义二维数组并赋初值时，可以省略第一维的大小，但不能省略第二维的大小，所以 A 是错误的；对于 C 来说，定义的数组 a 是 2 行 4 列的数组，但赋初值却包含了 3 行，所以是错误的；D 中初值列表中有一行是空的，这在 C 语言中是不允许的，所以也是错误的；B 定义了 2 行 3 列的二维数组并对其赋初值，是正确的，所以正确答案是 B。

(9) 以下不能正确进行字符串赋初值的语句是（ ）。

- A. $\text{char } str[5] = \text{"good!"};$ B. $\text{char } str[] = \text{"good!"};$
C. $\text{char } str[8] = \text{"good!"};$ D. $\text{char } str[5] = \{'g', 'o', 'o', 'd'\};$

【答案】 A。

(10) 判断字符串 s1 是否大于字符串 s2，应当使用 ()。

- 【答案】 D。

(11) 给出以下定义, 则正确的叙述为 ()。

A. 数组 x 和数组 y 等价 B. 数组 x 和数组 y 的长度相同
C. 数组 x 的长度大于数组 y 的长度 D. 数组 x 的长度小于数组 y 的长度

【答案】 C。

(12) 以下程序的输出结果是 ()

A. 9 9 B. 5 20 C. 13 20 D. 20 20

【答案】 B。

(13) 定义如下变量和数组:

则下面语句的输出结果是 ()。

A. 357 B. 369 C. 159 D. 147

【答案】 A。

【解释】 k=0 时，输出 a[0][2]，即为 3；k=1 时，输出 a[1][1]，即为 5；k=2 时，输出 a[2][0]，即为 7。所以正确答案是 A。

(14) 当执行下面的程序时，如果输入 ABC，则输出结果是（ ）。

```
#include "stdio.h"
#include "string.h"
void main( )
{
    char ss[10]="1,2,3,4,5";
    gets (ss); strcat (ss, "6789"); printf ("%s\n", ss);
}
```

A. ABC6789 B. ABC67 C. 12345ABC6 D. ABC456789

【答案】 A。

【解释】 字符数组 ss 的大小为 10，执行 get(ss);后，等待键盘输入一字符串，由于输入的是 ABC，因此此时 ss 中存放的字符串是"ABC"，再通过 strcat 库函数将字符串"6789"连接到 ss 当前的字符串"ABC"之后，这样 ss 中保存的字符串变为"ABC6789"。所以正确答案是 A。

(15) 以下程序的输出结果是（ ）。

```
void main( )
{
    char w[ ][10] = { "ABCD", "EFGH", "IJKL", "MNOP"}, k;
    for (k = 1; k < 3; k++) printf ("%s\n", w[k]);
}
```

A. ABCD B. ABCD C. EFG D. EFGH
FGH EFG JK IJKL
KL IJ O
M

【答案】 D。

【解释】 从 for 循环的次数可知，要显示两个字符串，一个是 w[1]，即为 EFGH，另一个是 w[2]，即为 IJKL。所以正确答案是 D。

(16) 以下程序的输出结果是（ ）。

```
void main( )
{
    char arr[2][4];
    strcpy (arr[0], "you"); strcpy (arr[1], "me");
    arr[0][3] = '&';
    printf ("%s \n", arr);
}
```

A. you&me B. you C. me D. err

【答案】 A。

【解释】 因为 arr 是一个 2 行 4 列的二维数组，每一行可以存放最多 3 个字符的字符串。执行 strcpy (arr[0], "you"); 后则将字符串 "you" 放置在数组 arr 的第 1 行上，即 a[0][0]='y', a[0][1]='o', a[0][2]='u', a[0][3]='\0' (字符串结尾符)，执行 strcpy (arr[1], "me"); 后则将字符串 "me" 放置在数组 arr 的第 2 行上，即 a[1][0]='m', a[1][1]='e', a[1][2]='\0' (字符串结尾符)，由于二维数组在内存中是以行序存放的，执行 arr[0][3]='&; 后，第 1 行与第 2 行将视为一个字符串 "you&me"，而数组名则是该字符串的首地址。所以正确答案是 A。

(17) 已知: char str1[8], str2[8]= {"good"}; 则在程序中不能将字符数组 str 2 赋值给 str1 的语句是 ()。

A. str1 = str2; B. strcpy (str1, str2);
C. strncpy (str1, str2, 6) D. memcpy (str1, str2, 5);

【答案】 A。

【解释】 数组名是一地址常量，数组名之间是不能彼此赋值的，所以 A 是错误的。故选择答案是 A。

(18) 下面程序段的运行结果是 ()。 (注: □代表空格符)

```
char c[5] = {'a', 'b', '\0', 'c', '\0'};
printf ("%s", c);
```

A. 'a"b' B. ab C. ab□c D. ab□

【答案】 B。

【解释】 printf 函数是用来输出字符数组 c 所存放的字符串，输出字符串中的字符将从第一个字符开始直到遇到 '\0' 为止。所以输出的结果是 ab。故正确答案是 B。

(19) 下面程序段的运行结果是 ()。 (注: □代表空格符)

```
char a[7] = "abcde";      char b[4] = "ABC";
strcpy (a, b);      printf ("%c", a[4]);
```

A. □ B. \0 C. e D. f

【答案】 C。

【解释】 执行 strcpy(a, b) 后，将把字符串 "ABC" 复制到数组 a 中，但是要注意复制的字符串只覆盖原来数组 a 中的前 4 个元素 (字符串 "ABC" 的长度加 1)，第 5 个元素 a[4] 的值没变，仍是字符 'e'。故正确答案是 C。

(20) 下面程序的运行结果是 ()。

```
void main( )
{
    char ch[7] = {"65ab21"};
    int i, s = 0;
    for (i = 0; ch[i] >= '0' && ch[i] <= '9'; i += 2)
```



```

        s = 10 * s + ch[i] - '0';
    printf ("%d\n", s);
}

```

A. 12ba56

B. 6521

C. 6

D. 62

【答案】 C。

【解释】 进入 for 循环时, i 的值为 0, 表达式(ch[i] >= '0' && ch[i] <= '9')是用来检测 ch[i]是否为数字字符, 因为 ch[0]是'6', 所以表达式为真, 进入循环体中执行 s = 10 * s + ch[i] - '0';语句, 该语句的功能就是将数字串转换成相应的整数, 因 s 的初值为 0, 所以 s 的结果为 6。语句执行完后, 再将 i 增 2, 再判断 ch[2]是否为数字字符, 因 ch[2]是'a', 表达式为假, 循环退出。所以 s 的值输出为 6, 故正确答案是 C。

2. 程序填空题

(1) 以下是一个评分统计程序, 共有 8 个评委打分, 统计时, 去掉一个最高分和一个最低分, 其余 6 个分数的平均分即是最后得分, 程序最后应显示这个得分, 显示精度为 1 位整数, 2 位小数, 程序如下, 请将程序补充完整。

```

#include <stdio.h>
void main( )
{
    float x[8] = {9.2, 9.5, 9.8, 7.4, 8.5, 9.1, 9.3, 8.8};
    float aver, max, min;
    int i;
    for( aver = 0, i = 0; i < 8; i++)
        aver += x[i];
    max = x[0];
    min = max;
    for (i = 1; i < 8; i++)
    {
        if (max < x[i]) max = x[i];
        if (min > x[i]) min = x[i];
    }
    aver = (aver - max - min) / 6;
    printf ("Average = %4.2f\n", aver);
}

```

【设计思想】 先求 8 个评委打分的总和, 放入变量 aver 中, 然后通过遍历数组 x 求得最高分 max 和最低分 min, 则平均分就为: (aver-max-min)/6。

(2) 以下程序是实现在 M 行 N 列的二维数组中, 找出每一行上的最大值。请将程序补充完整。

```

#define M 3
#define N 4
void main( )

```

```

{
    int x[M][N] = {1, 5, 7, 4, 2, 6, 4, 3, 8, 2, 3, 1};
    int i, j, p;
    for (i = 0; i < M; i++)
    {
        p = 0;
        for (j = 1; j < N; j++)
            if (x[i][p] < x[i][j]) p = j ;
        printf ("The max value in line %d is %d\n", i, x[i][p] );
    }
}

```

【设计思想】 第一层 for 循环，用于行控制，第二层 for 循环计算每一行中最大元素的列下标 p。

(3) 下面程序的功能是在三个字符串中找出最小的。请将程序补充完整。

```

#include <stdio.h>
#include <string.h>
void main( )
{
    int i;
    char s[20], str[3][20];
    for (i = 0; i < 3; i++) gets (str[i]);
    strcpy (s, strcmp (str[0], str[1]) > 0 ? str[1] : str[0] );
    if (strcmp (s, str[2]) > 0) strcpy (s, str[2]);
    printf ("The min string is %s\n", s );
}

```

【设计思想】 对输入的三个字符串 str[0]、str[1]、str[2]，首先将 str[0]和 str[1]中的小者复制给临时串 s，然后将 s 与字符串 str[2]进行比较，如果 str[2]比 s 还小，则将 str[2]复制给 s，s 中的字符串就是最小字符串。

(4) 下面程序的功能是将键盘输入的字符串 str 中的所有 'c' 字符用 'C' 替换。请将程序补充完整。

```

#include <stdio.h>
#include <string.h>
void main( )
{
    int i;
    char str[80];
    gets ( str );
    for (i = 0; str[i] != '\0' ; i++)
    {
        if (str[i] != 'c') continue ;
    }
}

```

```
        str[i] = 'C';
    }
    printf ("%s\n", str);
}
```

【设计思想】 先通过调用 gets 函数从键盘输入一字符串到 str 字符数组中，然后对字符数组中的每个元素与字符'c'进行比较，如果不等，则进入下轮循环，否则将该字符置为字符'C'，循环结束条件是数组元素的值为'\0'（字符串结尾符）。

3. 编程题

(1) 编程实现从键盘任意输入 20 个整数，统计非负数个数，并计算非负数之和。

【设计思想】 定义一个 20 个元素的整型数组，用于存放从键盘输入的 20 个整数。然后通过 for 循环对数组元素逐一与 0 比较，如果为负数，则进入下一轮循环，否则将该数组元素累计到变量 sum 中，sum 的初始值为 0。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int i, sum = 0, a[20];

    for (i = 0; i < 20; i++)        //输入20个整数
        scanf ("%d", &a[i]);
    for (i = 0; i < 20; i++)        //对非负数进行统计
    {
        if (a[i] < 0)                //为负数,进入下一轮循环
            continue;
        sum += a[i];                //非负数累计
    }
    printf ("sum = %d\n", sum);
}
```

程序运行结果（假设输入为：1 -1 2 3 4 -5 6 7 8 9 10 -4 11 2 -20 -4 2 3 4 5 ✓）：

sum = 76

(2) 输入 10 个整数，将这 10 个整数按升序排列输出，并且奇数在前，偶数在后。比如，如果输入的 10 个数是 10 9 8 7 6 5 4 3 2 1，则输出 1 3 5 7 9 2 4 6 8 10。

【设计思想】 将输入的 10 个整数，按其奇偶性分别放在数组 a 的左部和右部。其具体方法就是设置两个整型变量 odd 和 even，分别表示奇数和偶数存放在数组 a 中元素的

下标, odd 的初始值为 0, 每存放一个奇数 odd 增 1, even 的初始值为 9, 每存放一个偶数 even 减 1, 然后通过选择排序的方法分别对数组 a 左边的奇数和右边的偶数进行排序。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int i, j, odd, even, n, t, a[10];

    odd = 0;
    even = 9;
    for (i = 0; i < 10; i++)                //将键盘输入的数据存入数组中
    {
        scanf ("%d", &n);
        if (n % 2 != 0)                    //将奇数放置在数组的左边
            a[odd++] = n;
        else
            a[even--] = n;                //将偶数放置在数组的右边
    }

    for (i = 0; i < odd - 1; i++)            //通过选择排序对奇数进行升序排列
    {
        n = i;
        for (j = i + 1; j < odd; j++)
            if (a[j] < a[n])
                n = j;
        if (n != i)
        {
            t = a[i];
            a[i] = a[n];
            a[n] = t;
        }
    }

    for (i = odd; i < 9; i++)                //通过选择排序对偶数进行升序排列
    {
        n = i;
        for (j = i + 1; j < 10; j++)
            if (a[j] < a[n])
                n = j;
        if (n != i)
```



```
a[min] = i;
for (i = 0; i < 10; i++)           //显示结果
    printf("%d ", a[i]);
printf("\n");
}
```

程序运行结果（假设输入为：1 2 3 4 5 6 7 8 9 10↵）：

10 2 3 4 5 6 7 8 9

（4）编写一程序，其功能是给一维数组 *a* 输入任意的 6 个整数，假设为：5 7 4 8 9 1，然后建立一个具有以下内容的方阵，并打印出来。

```
5 7 4 8 9 1
1 5 7 4 8 9
9 1 5 7 4 8
8 9 1 5 7 4
4 8 9 1 5 7
7 4 8 9 1 5
```

【设计思想】 将数组 *a* 的数据循环显示 6 次，每次显示后将数组元素进行循环右移。其方法是先将 *a*[5] 保存到变量 *t* 中，再将 *a*[0]~*a*[4] 往右移 1 位，即 *a*[5]=*a*[4],...,*a*[1]=*a*[0]，然后将 *a*[0]=*t*。

【参考答案】

```
#include <stdio.h>

void main( )
{
    int i, j, t, a[6];

    printf ("Input 6 integer number: "); //输入6个整数
    for (i = 0; i < 6; i++)
        scanf ("%d", &a[i]);
    printf ("\n");
    for (i = 0; i < 6; i++)
    {
        for (j = 0; j < 6; j++)           //显示结果
            printf ("%d ", a[j]);
        printf ("\n");
        t = a[5];
        for (j = 5; j > 0; j--)           //将数组元素循环右移
            a[j] = a[j-1];
    }
}
```

```

    a[0] = t;
}
}

```

程序运行结果（假设输入为 5 7 4 8 9 1↵）:

```

5  7  4  8  9  1
1  5  7  4  8  9
9  1  5  7  4  8
8  9  1  5  7  4
4  8  9  1  5  7
7  4  8  9  1  5

```

(5) 输入 5×5 阶的矩阵，编程实现：

A. 求两条对角线上的各元素之和；

B. 求两条对角线上行、列下标均为偶数的各元素之积。

【设计思想】 通过两重 for 循环将键盘输入的 5×5 的矩阵元素存放在二维数组 a 中，采用累加和算法： $\text{sum} = \text{sum} + \text{a}[\text{i}][\text{i}]$ ($\text{i} = 0, 1, 2, 3, 4, 5$)，累计左对角线上的元素之和，同时用 $\text{sum} = \text{sum} + \text{a}[\text{i}][4 - \text{i}]$ ($\text{i} = 0, 1, 3, 4, 5$)，累计右对角线上的元素之和（除去两对角线上的交叉元素 $\text{a}[2][2]$ ）。采用累乘算法： $\text{mul} = \text{mul} * \text{a}[\text{i}][\text{i}]$ ($\text{i} = 0, 2, 4$)，累乘左对角线上行、列下标均为偶数的元素之积，同时用 $\text{mul} = \text{mul} * \text{a}[\text{i}][4 - 1]$ ($\text{i} = 0, 4$)，累乘右对角线上行、列下标均为偶数的元素之积（除去两对角线上的交叉元素 $\text{a}[2][2]$ ）。

【参考答案】

```

#include <stdio.h>

void main( )
{
    int i, j, sum = 0, mul = 1, a[5][5];

    printf ("Input 5*5 array:\n"); //输入5*5的矩阵
    for (i = 0; i < 5; i++)
        for (j = 0; j < 5; j++)
            scanf ("%d", &a[i][j]);

    printf ("\n");
    for (i = 0; i < 5; i++)
    {
        sum += a[i][i];           //对左对角线元素进行累加
        if (i != 2)               //对右对角线上的元素进行累加(对角线中间元素除外)
            sum += a[i][4-i];
        if (i % 2 != 0)           //如果行下标为奇数,进入下一次循环
            continue;
    }
    mul *= sum;
}

```

```

        continue;
    mul *= a[i][i];           //对左对角线上行、列下标均为偶数的元素进行累乘
    if (i != 2)               //对右对角线上行、列下标均为偶数的元素进行累乘(中间元素除外)
        mul *= a[i][4-i];
}
printf ("sum = %d    mul = %d\n", sum, mul);
}

```

程序运行结果:

```

Input 5*5 array:
7  2  7  4  8 ✓
9  3  5  7  4 ✓
8  3  5  6  7 ✓
4  8  5  3  5 ✓
2  4  8  9  1 ✓

sum = 44    mul = 560

```

(6) 编程打印如下形式的杨辉三角形。

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1

```

【设计思想】 对于有 6 行的杨辉三角形, 可用一个 6 行 6 列的二维数组 $a[6][6]$ 来表示。对于第 i 行的元素: $a[i][0] = 1$, $a[i][i] = 1$ ($i = 0, 1, 2, \dots, 5$), $a[i][j] = a[i-1][j-1] + a[i-1][j]$ ($j = 1, 2, \dots, i-1$)。然后显示二维数组中计算的结果, 其中第 1 行显示 1 个数据, 第 2 行显示 2 个数据, ……第 6 行显示 6 个数据。

【参考答案】

```

#include <stdio.h>
#define N 6

void main( )
{
    int i, j, a[N][N];

    for (i = 0; i < N; i++)
    {

```



```

    a[i][0] = 1;
    a[i][i] = 1;
    for (j = 1; j < i; j++)
        a[i][j] = a[i-1][j-1] + a[i-1][j];
}

for (i = 0; i < N; i++) //显示结果
{
    for (j = 0; j < N-i-1; j++)
        printf (" ");
    for (j = 0; j <= i; j++)
        printf ("%2d ", a[i][j]);
    printf ("\n");
}
}

```

程序运行结果:

```

        1
      1  1
    1  2  1
  1  3  3  1
1  4  6  4  1
1  5 10 10 5  1

```

(7) 编写一程序实现将用户输入的一字符串以反向形式输出。比如, 输入的字符串是 `ancdefg`, 输出为 `gfedcba`。

【设计思想】 通过 `gets` 库函数将键盘输入的一字符串存放到一个字符数组中, 然后利用 `for` 循环将数组中的字符串从高位到低位逐个字符输出。

【参考答案】

```

#include <stdio.h>
#include <string.h>
#define N 80

void main( )
{
    char str[N];
    int i;

    printf ("Input a string: ");
    gets (str);

```

```
printf ("After reverse: ");
for (i = strlen(str)-1; i >= 0; i--)
    printf ("%c", str[i]);
printf ("\n");
}
```

程序运行结果（假设输入为：ancdefg✓）：

```
After reverse: gfedcba
```

（8）编写一程序实现将用户输入的一字符串中所有的字符'c'删除，并输出结果。

【设计思想】 将键盘输入的字符串存入一字符数组 `str` 中，然后通过 `for` 循环对数组 `str` 中的字符逐一检查，如果不是字符'c'，则将该字符存入数组临时数组 `strtemp` 中，最后将 `strtemp` 复制到数组 `str`，并输出。

【参考答案】

```
#include <stdio.h>
#include <string.h>
#define N 80

void main( )
{
    char str[N], strtemp[N];
    unsigned int i, j;

    printf ("Input a string: ");           //将输入的字符串存入str数组中
    gets (str);
    for (i = 0, j = 0; i < strlen(str); i++) //将str中的非'c'字符复制到字符
                                                //数组strtemp中
        if (str[i] != 'c')
            strtemp[j++] = str[i];
    strtemp[j] = '\0';
    strcpy (str, strtemp);                 //将删除字符'c'后的字符串
                                                //strtemp复制给字符串str

    printf ("After delete charactor \'c\': ");
    printf ("%s\n", str);
}
```

程序运行结果（假设输入为：abcdefcg✓）：

```
After delete charactor 'c': abdefg
```

(9) 编写一程序，将字符数组 s2 中的全部字符复制到字符数组 s1 中，不用 strcpy 函数。复制时，'\0'也要复制过去。'\0'后面的字符不复制。

【设计思想】 使用 while(1)循环将 s2 中的字符从左到右逐一复制到 s1 中，如果此时复制的字符是'\0'，则循环结束，输出字符串 s1。

【参考答案】

```
#include <stdio.h>
#define N 80

void main( )
{
    char s1[N], s2[N] = "abcdefg\0hijk";
    unsigned int i;

    i = 0;
    while (1)
    {
        s1[i] = s2[i];
        if (s2[i] == '\0')
            break;
        i++;
    }
    printf ("After string copy: %s\n", s1);
}
```

程序运行结果：

After string copy: abcdefg

(10) 不用 strcat 函数编程实现字符串连接函数 strcat 的功能，将字符串 dstStr 连接到字符串 srcStr 的尾部。

【设计思想】 先找到字符串 dstStr 的结尾符'\0'的位置，然后通过 for 循环将字符串 srcStr 中的字符逐一复制到 dstStr 的后面，直到字符串 srcStr 的结尾符'\0'为止，最后在 dstStr 的后面加上结尾符'\0'。

【参考答案】

```
#include <stdio.h>

void main( )
{
    char dstStr[20] = "12345", srcStr[20] = "67890";
```

```

unsigned int i, j;

printf ("Before strcat: dstStr = %s srcStr = %s\n", dstStr, srcStr);
for (i = 0; dstStr[i] != '\0'; i++) //找到dstStr串中'\0'所在的位置i
;
for (j = 0; srcStr[j] != '\0'; j++) //将srcStr串中的字符逐个复制给dstStr
//串的后面

    dstStr[i+j] = srcStr[j];
dstStr[i+j] = '\0';
printf ("After  strcat: dstStr = %s\n", dstStr);
}

```

程序运行结果:

```

Before strcat: dstStr = 12345   srcStr = 67890
After  strcat: dstStr = 1234567890

```

(11) 有一个已排好序(升序)的整型数组,要求从键盘输入一整数按原来排序的规律将它插入数组中,并输出结果。比如,原来数据为: 1 3 5 7,需插入 4,插入后为: 1 3 4 5 7。

【设计思想】 假设插入前数组元素的个数为 num,首先要找到在整型数组中要插入的位置,其具体方法是:通过循环将插入的整数 n 与数组中的元素 a[i] (i=num-1,...,0) 从右到左进行逐一比较,如果此时数组元素 a[i]>n,则插入的位置在元素 a[i]之前的某一位置,将 a[i]往右移一个位置,循环继续;如果 a[i]<=n,则插入的位置刚好就在 a[i]之后,即 a[i+1]的位置,循环结束,执行 a[i+1]=n 就实现了插入的任务。

【参考答案】

```

#include <stdio.h>

void main( )
{
    int a[10] = {1, 3, 5, 7, 9}, num = 5, i, n;

    printf ("Before insert: "); //显示插入之前数组元素的值
    for (i = 0; i < num; i++)
        printf ("%d ", a[i]);
    printf ("\n");

    printf ("Input a number: "); //输入一插入的整数n
    scanf ("%d", &n);
    for (i = num-1; i >= 0; i--) //从右到左将与n进行比较
        if (a[i] > n) //数组元素比n大

```

```
        a[i+1] = a[i];           //将该数组元素往右移一个位置
    else                         //否则,退出循环,要插入的位置就是第i个元素之后
        break;
    a[i+1] = n;

    printf ("After insert: ");   //显示插入后数组元素的值
    for (i = 0; i < num+1; i++)
        printf ("%d ", a[i]);
    printf ("\n");
}
```

程序运行结果:

```
Before insert: 1 3 5 7 9
Input a number: 6✓
After insert: 1 3 5 6 7 9
```

(12) 假设有五位同学四门功课的成绩如下, 现要求得每位同学的总分、平均分及各门功课的平均分。

姓名	语文	数学	英语	综合
张大明	120	130	110	280
李小红	110	120	105	290
王志强	108	128	120	278
汪晓成	112	135	122	286
李 丹	100	120	108	276

【设计思想】 定义一个 6 行 6 列的二维浮点型数组, 用于存放五位学生的各门课程的成绩, 其中 1~5 行的前 4 列存放每位学生四门课程的成绩, 最后 2 列用于存放每位学生的总分和平均分。第 6 行的前 4 列用于存放每门课程的平均成绩。然后对这个二维数组分别按行、列进行统计, 即可得到所需的结果, 最后输出统计结果。

【参考答案】

```
#include <stdio.h>

void main( )
{
    //多增加的最后两列分别用来表示每个学生的总分和平均分
    //多增加的最后一行用来表示每门课程的平均分
    float score[6][6] = {{120, 130, 110, 280}, {110, 120, 105, 290}, {108,
        128, 120, 278}, {112, 135, 122, 286}, {100, 120,
        108, 276}};
    char name[6][10] = {"张大明", "李小红", "王志强", "汪晓成", "李 丹", "平
        均分"};
}
```

```
int i, j;

for (i = 0; i < 5; i++)
{
    for (j = 0; j < 4; j++)
    {
        score[i][4] += score[i][j];    //求每个学生的总分
        score[5][j] += score[i][j];    //累计每门课程的总分
    }
    score[i][5] = score[i][4] / 4;    //求每个学生的平均分
}
for (j = 0; j < 4; j++)                //求每门课程的平均分
    score[5][j] /= 5;

//显示结果
printf ("姓 名  语文   数学   英语   综合   总分   平均分\n");
printf ("-----\n");
for (i = 0; i < 6; i++)
{
    if (i < 5)
        printf ("%s  %.0f   %.0f   %.0f   %.0f   %.0f   %.2f\n",
                name[i], score[i][0], score[i][1], score[i][2],
                score[i][3], score[i][4], score[i][5]);
    else
    {
        printf ("-----\n");
        printf ("%s  %.2f %.2f %.2f %.2f\n", name[i], score[i][0],
                score[i][1], score[i][2], score[i][3]);
    }
}
}
```

程序运行结果:

姓 名	语文	数学	英语	综合	总分	平均分
张大明	120	130	110	280	640	160.00
李小红	110	120	105	290	625	156.25
王志强	108	128	120	278	634	158.50
汪晓成	112	135	122	286	655	163.75
李 丹	100	120	108	276	604	151.00
平均分	110.00	126.60	113.00	282.00		

1.8 习题 8 参考答案和解释

1. 填空题

(1) 在 C 语言程序中, 功能模块是由函数来实现的。函数是一段可以重复调的、功能相对独立完整的程序段。

(2) 从函数定义的角度看, 函数可分为标准库函数和自定义函数两种。

(3) 对于有返回值的函数来说, 通常函数体内包含有return语句, 其格式为return (表达式), 用于将返回值带给调用函数。

(4) 当一个函数的返回值类型默认时, 意味着该函数返回值类型为int类型。

(5) 调用带参数的函数时, 实参列表中的实参必须与函数定义时的形参数量相同、类型相符。

(6) 对带有参数的函数进行调用时, 参数的传递方式主要有传值调用和传址

【解释】 函数的形参可以是常量、变量或表达式，但不可以是常量，所以 B 是不正确的说法。故应选择 B。

(3) 以下正确的函数定义形式是 ()。

- A. `double fun(int x, int y)`
- B. `double fun (int x; int y)`
- C. `double fun (int x, int y);`
- D. `double fun (int x, y);`

【答案】 A。

【解释】 函数定义时每个形参都必须指明数据类型符，形参之间必须以逗号隔开，所以 B 和 D 是错误的，另外，函数定义时末尾不能带分号，只有函数说明时才带分号，所以 C 也是错误的。故正确答案是 A。

(4) 以下正确的说法是 ()。

- A. 定义函数时，形参的类型说明可以放在函数体内。
- B. `return` 后边的值不能为表达式。
- C. 如果函数值的类型与返回值类型不一致，以函数值类型为准。
- D. 如果形参与实参类型不一致，以实参类型为准。

【答案】 C。

【解释】 C 语言中，定义函数时，形参的类型说明必须放在形参的前面，不能放在函数体内；`return` 后边的值可以是常量或表达式；如果形参与实参类型不一致，应以形参类型为准；所以答案 A、B、D 都是不正确的说法。正确答案是 C。

(5) 在 C 语言中，函数的隐含存储类别是 ()。

- A. `auto`
- B. `static`
- C. `extern`
- D. 无存储类别

【答案】 C。

【解释】 如果在函数定义中没有说明 `extern` 或 `static`，则隐含为 `extern`。正确答案是 C。

(6) 凡是函数中未指定存储类别的局部变量，其隐含的存储类别为 ()。

- A. 自动 (`auto`)
- B. 静态 (`static`)
- C. 外部 (`extern`)
- D. 寄存器 (`register`)

【答案】 A。

【解释】 C 语言程序中，某函数内定义的局部变量，如果未指定存储类型，则默认为自动 (`auto`) 型。所以正确答案是 A。

(7) 若使用一维数组名作函数实参，则以下正确的说法是 ()。

- A. 必须在主调函数中说明此数组的大小
- B. 实参数组类型与形参数组类型可以不匹配
- C. 在被调用函数中，不需要考虑形参数组的大小
- D. 实参数组名与形参数组名必须一致

【答案】 A。

【解释】 当用数组名作为函数的实参时，在主调函数中必须说明该数组的大小，因为数组定义时必须要指明数组的大小，以方便系统为其分配内存单元。所以正确答案是 A。

(8) 已有如下数组定义和 f 函数调用语句, 则在 f 函数的说明中, 对形参数组 array 的正确定义方式为 ()。

```
int a[3][4];  
f(a);
```

- A. f(int array[][6])
- B. f(int array[3][])
- C. f(int array[][4])
- D. f(int array[2][5])

【答案】 C。

【解释】 当用二维数组作为形参时, 第一维的大小可以省略, 但第二维的大小是不能省略的, 必须和实参数组第二维的大小一致。所以正确答案是 C。

(9) 若用数组名作为函数的实参, 传递给形参的是 ()。

- A. 数组的首地址
- B. 数组第一个元素的值
- C. 数组中全部元素的值
- D. 数组元素的个数

【答案】 A。

【解释】 因为数组名是该数组所占内存单元的首地址, 是一地址常量, 当其作为函数的实参时, 传递给形参的显然是数组的首地址。所以正确答案是 A。

(10) 函数调用不可以 ()。

- A. 出现在执行语句中
- B. 出现在一个表达式中
- C. 作为一个函数的实参
- D. 作为一个函数的形参

【答案】 D。

【解释】 函数调用可以出现在 C 语言执行语句中, 也可以出现在表达式中, 甚至还可以作为一个函数的实参, 但不可作为函数的形参。所以正确答案是 D。

(11) C 语言规定, 函数返回值的类型是由 ()。

- A. return 语句中的表达式类型所决定
- B. 调用该函数时的主调函数类型所决定
- C. 调用该函数时系统临时决定
- D. 在定义该函数时所指定的函数类型所决定

【答案】 D。

【解释】 函数的返回值的类型是由定义该函数时所指定的数据类型来决定的。所以正确答案是 D。

(12) C 语言规定: 简单变量作为实参时, 它和对应形参之间的数据传递方式是 ()。

- A. 地址传递
- B. 单向值传递
- C. 由实参传给形参, 再由形参传回给实参
- D. 由用户指定的传递方式

【答案】 B。

【解释】 简单变量作为实参时, 将该变量所占内存单元的值传递给形参, 实参和形参各占不同的内存单元, 传递完后, 实参和形参不再有任何联系, 所以这种传递方式也叫做单向值传递方式。所以正确答案是 B。

(13) 以下只有在使用时才为该类型变量分配内存的存储类说明是 ()。

- A. auto 和 static
- B. auto 和 register
- C. register 和 static
- D. extern 和 register

【答案】 B。

【解释】 C 语言中变量的存储类型分为两种：动态存储和静态存储，其存储类型说明符有 auto、static、register 和 extern，其中 auto 和 register 所说明的变量是动态存储类型的变量，该类变量只有在使用时系统才为其分配内存单元。而 static 和 extern 所说明的变量是静态存储类型的变量，程序运行时系统就会为该类变量分配存储单元。所以正确答案是 B。

(14) 以下叙述中不正确的是 ()。

- A. 在不同的函数中可以使用相同名字的变量
- B. 函数中的形式参数是局部变量
- C. 在一个函数内定义的变量只在本函数范围内有效
- D. 在一个函数内的复合语句中定义的变量在本函数范围内有效

【答案】 D。

【解释】 在一个函数内的复合语句中定义的变量是局部变量，其作用范围是从定义处到复合语句的结束，也就是只在本复合语句内有效。所以 D 的叙述是错误的。应选择 D。

(15) 有以下程序，程序运行后的输出结果是 ()。

```
float fun (int x, int y)
{ return (x + y); }
void main( )
{
    int a = 2, b = 5, c = 8;
    printf ("%3.0f\n", fun ((int)fun (a + c, b), a - c));
}
```

- A. 编译出错
- B. 9
- C. 21
- D. 9.0

【答案】 B。

【解释】 对函数调用 fun ((int)fun (a+c, b), a-c)，首先调用 fun (a + c, b)，得到的值是 15.000000，因为 fun 函数的返回值是 float 型，然后通过对 fun (a + c, b) 进行强制类型转换将得到的结果转换为整型数 15，再调用 fun(15, a-c)，则得到的值为 9.000000。因为 printf 中输出格式控制符是 %3.0f，即不输出小数位，所以输出将是 9。所以正确答案是 B。

(16) 下列程序执行后的输出结果是 ()。

```
char st[ ] = "hello,friend!";
void func1 (int i)
{
    printf ("%c", st[i]);
    if (i < 3) { i += 2; func2 (i); }
}
```

```
void func2 (int i)
{
    printf ("%c", st[i]);
    if (i < 3) { i += 2; func1 (i); }
}
void main( )
{
    int i = 0; func1(i); printf("\n");
}
```

A. hello B. hel C. hlo D. hlm

【答案】 C。

【解释】 首先从 main 函数开始， $i=0$ ，调用 func1 函数，显示字符 str[0]（即 h），因 $i<3$ 成立， $i+=2$ 后， $i=2$ ，再调用 func2，显示字符 str[2]（即 l），因 $i<3$ 成立， $i+=2$ 后， $i=4$ ，又调用 func1，显示 str[4]（即 o），此时 $i<3$ 不成立，返回到 func2，再又返回到 func1，最后返回到 main 函数。所以输出结果是 hlo，正确答案是 C。

(17) 有以下程序，程序运行后的输出结果是（ ）。

```
int f (int n)
{
    if (n == 1) return 1;
    else return f (n - 1) + 1;
}
void main( )
{
    int i, j = 0;
    for (i = 1; i < 3; i++) j += f(i);
    printf ("%d\n", j);
}
```

A. 4 B. 3 C. 2 D. 1

【答案】 B。

【解释】 首先了解函数 f 的功能： $f(1)=1$ ， $f(2)=f(1)+1=2$ ， $f(3)=f(2)+1=3$ ， \dots ， $f(n)=f(n-1)+1$ ，所以 f 的功能其实就是返回实参的值。再来执行 main 函数中的 for 循环， $i=1$ 时， $j+=f(1)$ 后，j 的值为 1， $i++$ 后， $i<3$ 成立，继续执行 $j+=f(2)$ ，则 j 的值为 $1+2=3$ 。所以正确答案为 B。

(18) 以下程序的输出结果是（ ）。

```
void incre( );
int x = 3;
void main( )
{
    int i;
    for (i = 1; i < x; i++) incre( );
```

```

}
void incre( )
{
    static int x = 1;
    x *= x + 1;
    printf ( " %d ", x);
}

```

A. 3 3 B. 2 2 C. 2 6 D. 2 5

【答案】 C。

【解释】 因为在函数 `incre` 内部定义了一静态局部变量 `x`，尽管与全局变量 `x` 同名，但它们占用各自不同的内存单元，对于静态变量 `x` 来说，因为它的生存期与程序相同，所以每次调用后其值将长期保留，其初始值是当程序运行时赋值一次，以后调用不再赋初始值。所以第一次调用输出 2，第二次调用就输出 6 ($2*3$)，故正确答案为 C。

(19) 以下程序的输出结果是 ()。

```

int a = 3;
main( )
{
    int s = 0;
    { int a = 5; s += a++; }
    s += a++; printf ("%d\n", s);
}

```

A. 8 B. 10 C. 7 D. 11

【答案】 A。

【解释】 复合语句中定义的变量如果与全局变量同名，那么在复合语句中对该变量的引用是指复合语句中定义的变量，而不是全局变量。所以，程序中复合语句内 `s+=a++`；执行的结果将使得 `s` 的值为 `s=s+5=0+5=5`，复合语句执行完后，再执行 `s+=a++`；此时的 `a` 是指全局变量，`s` 的值将是 `s=s+3=5+3=8`。所以正确答案为 A。

(20) 以下程序的输出的结果是 ()。

```

void f (int a[ ], int i, int j)
{
    int t;
    if (i < j) {t = a[i]; a[i] = a[j]; a[j] = t; f (a, i+1, j-1);}
}
void main( )
{
    int i, a[5] = {1, 2, 3, 4, 5};
    f (a, 0, 4);
    for (i = 0; i < 5; i++) printf("%d, ", a[i]);
}

```

A. 5,4,3,2,1 B. 5,2,3,4,1 C. 1,2,3,4,5 D. 1,2,3,5,4

【答案】 A。

【解释】 函数f的功能是：如果数组a的下标i比下标j小，就将a[i]与a[j]交换，然后进一步判断i+1与j-1的大小，如果i+1比j-1小，再将a[i+1]与a[j-1]交换，直到前者的下标不小于后者的下标为止，即函数f的功能就是将数组元素a[i]到a[j]之间元素进行交换。所以函数调用f(a, 0, 4)其结果就是将a[0]与a[4]交换、a[1]与a[3]交换，所以交换后数组a中的元素为：5 4 3 2 1。所以正确答案为A。

3. 编程题

(1) 设计一个函数，用来判断一个整数是否为素数。

【设计思想】 按照素数的定义，可以用2~number-1的所有整数去除number，只要有能整除的，便说明number不是素数。不过对任意值的number检验它是否是素数时，不必使用比其平方根大的整数去整除它。

【参考答案】

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

int IsPrimeNumber (int number);

void main( )
{
    int a;

    printf ("Input a integer number: ");
    scanf ("%d", &a);
    if(IsPrimeNumber(a))
        printf ("%d is prime number.\n", a);
    else
        printf ("%d isn't prime number.\n", a);
}

//函数功能：判断参数是否为素数
//函数入口参数：整型数，要求为正整数
//函数返回值：非0值表示素数，否则不是素数
int IsPrimeNumber (int number)
{
    int i;

    if (number <= 1)                //负数、0和1都不是素数
        return (0);
```

```

    for(i = 2; i < sqrt(number); i++)
        if(number % i == 0)    //被整除, 不是素数
            return (0);
    return (1);
}

```

程序运行结果（假设输入的整数为：5✓）：

```
5 is prime number
```

(2) 设计函数 MaxCommonFactor(), 计算两个正整数的最大公约数。

【设计思想】 对于 a 和 b 两个数, 当 $a > b$ 时, 如果 a 中含有与 b 相同的公约数, 则 a 中减去 b 后剩余的部分 $a-b$ 也应该含有与 b 相同的公约数, 对 $a-b$ 和 b 计算公约数就相当于对 a 和 b 计算公约数。反复使用最大公约数的 3 个性质, 直到 a 和 b 相等为止, 这时 a 或 b 就是它们的最大公约数。

【参考答案】

```

#include <stdio.h>

int MaxCommonFactor (int a, int b);

void main( )
{
    int a, b, c;

    printf ("Input two integer number: ");
    scanf ("%d%d", &a, &b);
    c = MaxCommonFactor (a, b);
    if(c!=-1)
        printf ("The biggest common factor of %d and %d is %d\n", a, b, c);
    else
        printf("The biggest common factor of %d and %d isn't exist\n", a,b);
}

//函数功能: 计算两个正整数的最大公约数
//函数入口参数: 两个整型数
//函数返回值: 最大公约数, -1表示没有最大公约数
int MaxCommonFactor (int a, int b)
{
    if(a <= 0 || b <= 0)    //保证输入的参数正确
        return (-1);

    while (a != b)

```

```
{
    if(a > b)
        a=a-b;
    else
        if(b > a)
            b = b - a;
}
return (a);
}
```

程序运行结果（假设输入的整数为：8 20✓）：

```
The biggest common factor of 8 and 20 is 4
```

（3）定义函数 `GetData()` 用于接收键盘输入的一组整型数据，并放入一数组中；另外再定义一个函数 `Sort()` 用于对输入的这一组数据按照降序排列。主函数先后调用 `GetData` 和 `Sort` 函数，输出最后的排序结果。

【设计思想】 `GetData` 函数是用于从键盘接收一组整型数据，因此该函数要带两个形参，一个是数组，一个是数组中元素的个数。`Sort` 函数是对数组元素进行降序排列，参数的形式与 `GetData` 函数相同，排序的方法可以采用冒泡排序或选择排序，本答案采用选择排序法。

【参考答案】

```
#include <stdio.h>

#define N 10

void GetData (int a[ ], int n);
void Sort(int a[ ], int n);

void main( )
{
    int i, a[N];

    GetData (a, N);
    Sort (a, N);
    printf ("After sort: ");
    for(i = 0; i < N; i++)
        printf ("%d ", a[i]);
    printf ("\n");
}
```

//通过键盘输入n个整数到数组a中

```
void GetData (int a[ ], int n)
{
    int i;

    printf ("Input number: ");
    for(i = 0; i < n; i++)
        scanf ("%d", &a[i]);
}

//对数组a中的n个元素进行降序排列(排列算法:选择排序)
void Sort(int a[ ], int n)
{
    int i, j, k, t;

    for (i = 0; i < n-1; i++)
    {
        k = i;
        for (j = i+1; j < n; j++)
            if (a[j] > a[k])
                k = j;
        if (k != i)
        {
            t = a[i];
            a[i] = a[k];
            a[k] = t;
        }
    }
}
```

程序运行结果（假设输入的整数为：0 1 2 3 4 5 6 7 8 9↵）：

```
After sort: 9 8 7 6 5 4 3 2 1 0
```

（4）请编制函数 JsSort()，其函数的功能是：对字符串变量的下标为奇数的字符按其 ASCII 值从大到小的顺序进行排序，排序后的结果仍存入字符串数组中。例如：

位置	0	1	2	3	4	5	6	7
源字符串	a	b	c	d	e	f	g	h
处理后字符串	a	h	c	f	e	d	g	b

【设计思想】 JsSort 函数要完成对字符串下标为奇数的字符排序，一方面，函数应带一形参字符串 str，另一方面排序的方法可以用选择排序法，但要注意的是控制字符串下标的变量 i 和 j 都应是奇数，所以 i 的取值是 1, 3, …，每次循环后加 2，j 的取值则是 i+2, i+4, …，每次循环后也加 2。

【参考答案】

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

void JsSort (char str[ ]);

void main( )
{
    char str[80];

    printf ("Input a string: ");
    gets (str);
    JsSort (str);
    printf ("After Sort: %s\n", str);
}

//对字符串str中下标为奇数的字符按其ASCII码值大小从大到小排序
void JsSort (char str[ ])
{
    int i, j, k, len;
    char ch;

    len = strlen (str);                                //计算字符串的长度
    for (i = 1; i < len-1; i += 2)                      //下标取奇数
    {
        k = i;
        for(j = i+2; j < len; j += 2)                  //下标取奇数
            if(str[j] > str[k])
                k = j;
        if(k != i)
        {
            ch = str[i];
            str[i] = str[k];
            str[k] = ch;
        }
    }
}
```

程序运行结果（假设输入的字符串为：abcdefgh✓）：

```
After sort: ahcfedgb
```

(5) 设有 n 个人围坐一圈并按顺时针方向从 1 到 n 编号，从第 s 个人开始进行 1 到 m 的报数，报数到第 m 个人，此人出圈，再从他的下一个人重新开始 1 到 m 的报数，如此进行下去直到所有的人都出圈为止。现要求按出圈次序，给出这 n 个人的顺序表。请编制函数 `Josegh()` 实现此功能。

【设计思想】 设 $n=30$, $s=1$, $m=10$ 。

(1) 将 $1\sim n$ 个人的序号存入一维数组 p 中;

(2) 若第 i 个人报数后出圈, 则将 $p[i]$ 置于数组的倒数第 i 个位置上, 而原来第 $i+1$ 个至倒数第 i 个元素依次向前移动一个位置;

(3) 重复步骤 (2) 直至圈中只剩下 $p[1]$ 为止。

在函数 `Josegh` 中第一个 `for` 循环是先对数组 p 赋初值。在第二个 `for` 循环中用 i 来控制没出圈的总人数, $s1=(s1+m-1)\%i$ 的作用是找出报数后出圈人的下标, 其中对 i 求余的作用是使报数按圈进行 (即报到末尾后又从头再报), 该算法在很多题目中都用到。由于求余的作用, 当报数正好到最后一个时 $s1$ 为 0 , 故要进行 `if(s1==0)` 的判断。内嵌的 `for` 循环是将出圈以后的人依次往前移。

【参考答案】

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

#define N 30

void Josegh(int p[ ], int n, int s, int m);

void main( )
{
    int i, p[N];

    Josegh(p, N, 1, 10);
    for (i = N-1; i >= 0; i--)
    {
        printf(" %4d", p[i]);
        if(i % 10 == 0)
            printf ("\n");
    }
}

void Josegh(int p[ ], int n, int s, int m)
{
    int i, j, s1, w;

    s1 = s;
    for(i = 1; i <= n; i++)
        p[i-1] = i;
    for (i = n; i >= 2; i--)
    {
        s1 = (s1 + m - 1) % i;
```

```

        if(s1 == 0)
            s1 = i;
        w = p[s1-1];
        for (j = s1; j < i; j++)
            p[j-1] = p[j];
        p[i-1] = w;
    }
}

```

程序运行结果:

10	20	30	11	22	3	15	27	9	24
7	23	8	26	14	2	21	16	6	4
1	5	13	19	12	29	18	25	17	28

(6) 编写一函数 StrLoc, 其功能是求得一字符串 str1 在另一字符串 str2 中首次出现的位置, 如果 str1 不在 str2 中, 则返回-1。例如, 假设 str1 为"do", str2 为"how do you do?", 则返回值为 4。

【设计思想】 要求得字符串 str1 在字符串 str2 中的位置, 其方法就是设置一控制变量 i, i 的取值是从 0~strlen(str2)-strlen(str1), 利用 strncmp 函数将 str1 与 str2+i 的 strlen(str1) 个字符进行比较, 如果返回值为 0, 则说明 str1 在 str2 中, 并且第一次出现的位置就是 i。否则, str1 不在 str2 中出现, 返回值为-1。当然, 开始比较前, 先判断一下 str1 的长度是否大于 str2 的长度, 如果成立, 则直接返回-1。

【参考答案】

```

#include <stdio.h>
#include <string.h>

int StrLoc(char str1[ ], char str2[ ]);

void main( )
{
    int loc;
    char str1[ ] = {"do"};
    char str2[ ] = {"how do you do?"};

    loc = StrLoc (str1, str2);
    if(loc == -1)
        printf ("The string \"%s\" not in the string \"%s\".\n", str1, str2);
    else
        printf ("Location of the string \"%s\" in the string \"%s\" is %d.\n",
            str1, str2, loc);
}

```

```

int StrLoc(char str1[ ], char str2[ ])
{
    unsigned int i, len1, len2;

    len1 = strlen (str1);
    len2 = strlen (str2);
    if(len1 > len2)
        return (-1);
    for (i = 0; i <= strlen(str2)-strlen(str1); i++)
        if(strncmp(str1, str2+i, len1) == 0)
            return (i);
    return (-1);
}

```

程序运行结果:

Location of the string "do" in the string "how do you do?" is 4.

(7) 编写一函数计算 $\sum_{x=1}^n x^k$ 。

【设计思想】 设 $n=10$, $k=2$ 。

采用两重 for 循环来进行计算, 第一重 i 循环实现累计和运算, 即 $\text{sum} = \text{sum} + i^k$, sum 的初值为 0, $i = 1, 2, \dots, n$; 第二重 j 循环计算 i^k , 其方法是用通过 item 累乘, 即 $\text{item} = \text{item} * i$, item 的初值为 1, $j = 1, 2, \dots, k$; 第二重循环结束后 item 的值就是 i^k 。

【参考答案】

```

#include <stdio.h>

long count (int n, int k);

void main( )
{
    int n, k;

    n = 10;
    k = 2;
    printf ("Result = %ld\n", count(n, k));
}

long count (int n, int k)
{
    int i, j, sum, item;
    sum = 0;

```

```
for(i = 1; i <= n; i++)
{
    item = 1;
    for(j = 1; j <= k; j++)          //计算i的k次方
        item *= i;
    sum += item;                     //累计求和
}
return (sum);
}
```

程序运行结果:

Result = 385

(8) 编写一递归函数求斐波纳契数列的前 40 项。

【设计思想】 斐波纳契数列的第一项和第二项的值都为 1，以后各项的值为其前两项值之和。所以要计算第 n 项的值 $F(n)$ ，可以列出其递归式： $F(n) = F(n-1) + F(n-2)$ ，当 $n=1$ 或 2 时其值为 1。

【参考答案】

```
#include <stdio.h>

long F (int n);

void main( )
{
    int i;

    for (i = 1; i <= 40; i++)
    {
        printf ("F(%2d) = %-9ld ", i, F(i));
        if (i % 4 == 0)
            printf ("\n");
    }
}

long F (int n)                                //求斐波纳契数列的第n项
{
    if (n <= 2)
        return (1);
    return (F(n-1) + F(n-2));
}
```

程序运行结果:

F(1) = 1	F(2) = 1	F(3) = 2	F(4) = 3
F(5) = 5	F(6) = 8	F(7) = 13	F(8) = 21
F(9) = 34	F(10) = 55	F(11) = 89	F(12) = 144
F(13) = 233	F(14) = 377	F(15) = 610	F(16) = 987
F(17) = 1597	F(18) = 2584	F(19) = 4181	F(20) = 6765
F(21) = 10946	F(22) = 17711	F(23) = 28657	F(24) = 46368
F(25) = 75025	F(26) = 121393	F(27) = 196418	F(28) = 317811
F(29) = 514229	F(30) = 832040	F(31) = 1346269	F(32) = 2178309
F(33) = 3524578	F(34) = 5702887	F(35) = 9227465	F(36) = 14930352
F(37) = 24157817	F(38) = 39088169	F(39) = 63245986	F(40) = 102334155

(9) 编写一递归函数计算组合 C_m^n 。

【设计思想】寻找递归公式: 当 $m=n$ 或 $n=0$ 时, $C_m^n=1$; 否则, $C_m^n = m/(m-n) * C_{m-1}^n$ 。

【参考答案】

```
#include <stdio.h>

long fun(int m, int n);

void main( )
{
    int m, n;
    long c;

    printf ("Input m and n(m>=n): ");
    scanf ("%d%d", &m, &n);
    c = fun (m, n);
    printf ("c = %ld\n", c);
}

long fun (int m, int n)
{
    if (m == n || n == 0)
        return (1L);
    else
        return (m * fun(m-1, n) / (m-n));
}
```

程序运行结果:

```
Input m and n(m>=n): 20 8✓
c = 125970
```

(10) 编写一递归函数将一个整数 n 转换成字符串，例如输入 483，应输出字符串 "483"。 n 的位数不确定，可以是任意位数的整数。

【设计思想】 设置一字符型数组 str ，用于存放转换后的字符串，因为在递归函数中要反复使用它，所以将其设置成全局数组变量。要实现整数 n 转换成字符串的递归转换，首先将 n 的个位数 $n\%10$ 转换成字符，插入到 str 字符串的最前面，然后将除个位后的高位部分 $n/10$ 按照相同的方式来转换，直到要转换的数为 0 结束。

【参考答案】

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

void IntToStr (int n);

char str[80] = {0};

void main( )
{
    int num;

    printf ("Input an Integer number: ");
    scanf ("%d", &num);
    IntToStr (num);
    printf ("The string is : %s\n", str);
}

//将一整型数n转换成字符串存放到字符型数组str中
void IntToStr (int n)
{
    int i;

    if (n == 0)                                //整数为0时，递归结束
        return;
    for (i = strlen(str)-1; i >= 0; i--) //将字符串整个往后移一个字符
        str[i+1] = str[i];
    str[0] = n % 10 + 0x30;    //将该数的个位数转换成字符放在字符数组的第一个元素
                                //的位置
    IntToStr (n / 10);        //将该数的商，即除个位外的其他数通过递归转换
}
```

程序运行结果（假设输入的整数为： 12345✓）：

The string is : 12345

1.9 习题 9 参考答案和解释

1. 选择题

(1) 下面能正确进行字符串赋值操作的是 ()。

- A. `char s[5] = {"ABCDE"};` B. `char s[5] = {'A', 'B', 'C', 'D', 'E'};`
C. `char *s; s = "ABCDE";` D. `char *s; scanf ("%s", s);`

【答案】 C。

【解释】 C 语言中，字符串必须是以 '\0' 结尾。在答案 A 中，s 定义为 5 个元素的字符型数组，即最多能存放 5 个字符，字符串结尾符 '\0' 无法存放，故 A 是错误的字符串赋值；答案 B 对数组赋值没错，但同 A 一样，没有字符串结尾符 '\0'，所以也是错误的；答案 D 是“野指针”赋值，是错误的；而答案 C 先定义字符指针 s，而后将字符串 "ABCDE" 的首地址赋值给指针变量 s，是正确的赋值。所以正确答案是 C。

(2) 对于基类型相同的两个指针变量之间，不能进行的运算是 ()。

- A. < B. = C. + D. -

【答案】 C。

【解释】 C 语言中，对于基类型相同的两个指针变量之间可以进行大小比较、赋值及减法运算，但加法运算没有意义。所以不能进行的运算应选择 C。

(3) 下面说明不正确的是 ()。

- A. `char a[10] = "china";` B. `char a[10], *p = a; p = "china";`
C. `char *a; a = "china";` D. `char a[10], *p; p = a = "china";`

【答案】 C。

【解释】 C 语言中，在定义字符型数组时，可以直接用字符串对数组赋初值，只要数组定义的大小能容纳字符串中的字符即可。另外，对定义的字符型指针变量可以将字符串常量对其赋值，但数组名是地址常量，是不能用字符串对其赋值的。所以正确答案是 C。

(4) 若有下面的程序段，则下列叙述正确的是 ()。

`char s[] = "china"; char *p; p = s;`

- A. s 和 p 完全相同
B. 数组 s 中的内容和指针变量 p 中的内容相等
C. s 数组长度和 p 所指向的字符串长度相等
D. *p 与 s[0] 相等

【答案】 D。

【解释】 s 是数组名，是一地址常量，对 s 不能进行运算，而 p 是指针变量，可进行赋值等运算，所以答案 A 是错误的；数组 s 中的内容是一字符串，而指针变量 p 中的内容是地址，两者不可能相等，故答案 B 是错误的；数组 s 的大小为 6 (包含 '\0')，而 p 所指向的字符串长度是 5 (不包含 '\0')，所以 C 也是错误的。p 指向数组 s 的第一个元素，

所以 *p 与 s[0] 的值都为字符 'c'，所以正确答案是 D。

(5) 若有说明：int *p, m = 5, n; 以下正确的程序段是 ()。

- A. p = &n; scanf ("%d", &p); B. p = &n; scanf ("%d", *p)
C. scanf ("%d", &n); *p = n; D. p = &n; *p = m;

【答案】 D。

【解释】 因 p 是指针变量，在 scanf 中不能再使用 &p 或 *p 来接收键盘的输入值，应直接用 p 即可，所以答案 A 和 B 都是错误的；而 C 中 *p=n; 是“野指针”赋值，因 p 未指向某一内存单元，所以答案 C 也是错误的；答案 D 中是将变量 n 的地址给指针变量 p，然后再把 m 的值赋值给 p 所指向的内存单元，也就是赋值给 n，是正确的。所以正确答案是 D。

(6) 若有语句 int *point, a = 4; 和 point = &a; 下面均代表地址的一组选项是 ()。

- A. a, point, *&a B. &*a, &a, *point
C. *&point, *point, &a D. &a, &*point, point

【答案】 D。

【解释】 a 是整型变量不是地址，所以答案 A 是错误的；p 是指针变量，其所指向内存单元的内容 *point 是整型值，也不是地址，所以答案 B 和 C 也是错误的；对于答案 D 来说，&a 指变量 a 的地址，&*point 表示 point 所指向内存单元值的地址，也就是 point。所以正确答案是 D。

(7) 变量的指针，其含义是指该变量的 ()。

- A. 值 B. 地址 C. 名 D. 一个标志

【答案】 B。

【解释】 指针即地址，变量的指针也就是该变量在内存中的地址。所以正确答案是 B。

(8) 已有定义 int k = 2; int *ptr1, *ptr2; 且 ptr1 和 ptr2 均已指向变量 k，下面不能正确执行的赋值语句是 ()。

- A. k = *ptr1 + *ptr2; B. ptr2 = k;
C. ptr1 = ptr2; D. k = *ptr1 * (*ptr2);

【答案】 B。

【解释】 因 ptr1 和 ptr2 均是指向变量 k 的指针变量，指针变量只能存放地址，所以 ptr2=k; 这种将变量 k 的值（整型值 2）赋值给指针变量显然是错误的。所以选择答案是 B。

(9) 若有说明：int i, j = 2, *p = &i; 则能完成 i = j 赋值功能的语句是 ()。

- A. i = *p; B. *p = *&j; C. i = &j; D. i = **p;

【答案】 B。

【解释】 因为指针变量 p 指向了变量 i，所以 *p 就是变量 i，而 *&j 表示变量 j 的地址所指向内存单元的内容，故 *p = *&j; 完成的功能就是 i = j;。所以正确答案是 B。

(10) 若有定义：int a[8]; 则以下表达式中不能代表数组元素 a[1] 的地址的是 ()。

- A. &a[0] + 1 B. &a[1] C. &a[0]++ D. a + 1

【答案】 C。

【解释】 &a[0]+1 表示元素 a[0] 的地址加 1，也就是 a[1] 的地址；&a[1] 当然表示元

素 $a[1]$ 的地址; $a+1$ 表示数组 a 的首地址(即 $a[0]$ 的地址)加 1, 也是 $a[1]$ 的地址; 而 $\&a[0]++$ 包含有对 $\&a[0]$ 的赋值运算, 本身是非法的。所以选择答案是 C。

(11) 若有以下语句且 $0 \leq k < 6$, 则正确表示数组元素地址的语句是 ()。

```
static int x[ ] = {1, 3, 5, 7, 9, 11}, *ptr = x, k;
```

- A. $x++$ B. $\&ptr$ C. $\&ptr[k]$ D. $\&(x+1)$

【答案】 C。

【解释】 x 是数组名, 是一地址常量, 不能赋值, 所以答案 A 是错误的; ptr 是指针变量, $\&ptr$ 表示该指针变量所占内存单元的地址, 所以答案 B 也是错误的; $\&(x+1)$ 是非法的表达式, 故 D 也是错误的; $\&ptr[k]$ 表示数组元素 $x[k]$ 的地址, 它等价于 $\&*(ptr+k)$ 。所以正确答案是 C。

(12) 设已有定义: $\text{char } *st = \text{"how are you"};$ 下列程序段中正确的是 ()。

- A. $\text{char } a[11], *p; \text{strcpy}(p = a + 1, \&st[4]);$
B. $\text{char } a[11]; \text{strcpy}(++a, st);$
C. $\text{char } a[11]; \text{strcpy}(a, st);$
D. $\text{char } a[], *p; \text{strcpy}(p = a[1], st + 2);$

【答案】 A。

【解释】 a 是数组名, 是一地址常量, $++a$ 是非法的, 所以答案 B 错误; 因为数组 a 的大小为 11, 执行 $\text{strcpy}(a, st);$, st 字符串的结尾符 $\backslash 0$ 无法存放到数组 a 中, 所以答案 C 也是错误的; 答案 D 中定义数组 a 时未指定大小, 所以也是错误的。而答案 A 执行 $\text{strcpy}(p=a+1, \&st[4]);$ 首先将数组 a 的第二个元素 $a[1]$ 的地址赋给指针变量 p , 然后将 st 第 5 个元素开始直到字符串结束为止的字符复制到 p 所指向的内存单元中都是正确的。所以选择答案是 A。

(13) 下面程序段的运行结果是 ()。

```
char s[ ] = "abcdefgh", *p = s;  
p += 3;  
printf ("%d\n", strlen(strcpy(p, "ABCD")));
```

- A. 8 B. 12 C. 4 D. 7

【答案】 C。

【解释】 $\text{strcpy}(p, \text{"ABCD"})$ 的功能是将字符串 "ABCD" 复制到指针变量 p 所指向的内存单元中, 其返回值是指针 p 所指内存单元的地址, 然后计算 p 所指字符串的长度, 显然是 4。所以正确答案是 C。

(14) 下面程序段中, for 循环的执行次数是 ()。

```
char *s = "\ta\018bc";  
for (; *s != '\0'; s++) printf ("*");
```

- A. 9 B. 5 C. 6 D. 7

【答案】 C。

【解释】for 循环的执行次数其实就是字符串 s 的长度，在 s 所指向的字符串中，'\t' 是转义字符，'\01' 也是一个转义字符，这样字符串的长度就是 6。所以正确答案是 C。

(15) 以下程序的输出结果是 ()。

```
void main( )
{
    char *p = "abcdefgh", *r;
    long *q;
    q = (long*)p;
    q++;
    r = (char*)q;
    printf ("%s\n", r);
}
```

A. bcdefgh B. cdefgh C. defgh D. efgh

【答案】D。

【解释】执行 `q = (long*)p;` 后，q 指向字符串的第一个元素'a'；`q++;` 执行后，q 指向字符串的第 5 个元素'e'。因为 q 是 long 型指针变量，增 1 相当于将 q 往后移动 4 个字符的位置。然后执行 `r = (char*)q;` 则 r 指向字符'e'，这时再输出 r 所指向的字符串就是"efgh"了。所以正确答案是 D。

(16) 设有语句 `int array[3][4];` 则在下面几种引用下标为 i 和 j 的数组元素的方法中，不正确的引用方式是 ()。

A. `array[i][j]` B. `*(*(array + i) + j)`
C. `*(array[i] + j)` D. `*(array + i * 4 + j)`

【答案】D。

【解释】因为 array 是一个二维数组，`*(array + i)` 是行地址，等价于 `array[i]`，而 `*(array[i] + j)` 则是第 i 行中的第 j 个元素，即 `array[i][j]`，所以答案 A、B、C 都是正确的；而答案 D 只是对行地址的引用，不是对数组元素的引用。所以选择答案是 D。

(17) 以下程序的运行结果是 ()。

```
void sub (int x, int y, int *z)
{ *z = y - x ; }
void main( )
{
    int a, b, c;
    sub (10, 5, &a);
    sub (7, a, &b);
    sub (a, b, &c);
    printf ("%d, %d, %d\n", a, b, c);
}
```

A. 5, 2, 3 B. -5, -12, -7 C. -5, -12, -17 D. 5, -2, -7

【答案】B。

【解释】 第一次调用函数 `sub (10, 5, &a);` 后, `a` 的值为 $5-10=-5$, 第二次调用函数 `sub (7, a, &b);` 后, `b` 的值为 $a-7=-5-7=-12$, 第三次调用函数 `sub (a, b, &c);` 后, `c` 的值为 $b-a=-12-(-5)=-7$ 。所以选择答案是 B。

(18) 在说明语句 `int *f();` 中, 标识符 `f` 代表的是 ()。

- A. 一个用于指向整型数据的指针变量
- B. 一个用于指向一维数组的行指针
- C. 一个用于指向函数的指针变量
- D. 一个返回值为指针型的函数名

【答案】 D。

【解释】 指向整型数据的指针变量应说明为: `int *f;` 指向一维数组的行指针应说明为: `int (*f)[5];` 指向函数的指针变量应说明为: `int (*f)();` 所以选择答案是 D。

(19) 若有以下说明和定义, 在必要的赋值之后, 对 `fun` 函数的正确调用语句是 ()。

```
int fun (int *c) { ... }  
void main( )  
{  
    int (*a)(int * ) = fun, *b( ), w[10], c;  
    ...  
}
```

- A. `a = a(w);` B. `(*a)(&c);` C. `b = *b(w);` D. `fun (b);`

【答案】 B。

【解释】 在答案 A 中, `a(w)` 调用正确, 但将其返回值赋值给 `a` 是错误的, 因为返回值是整型数, 而 `a` 是一函数指针, 所以答案 A 是错误的。在答案 C 中, 因 `b` 是一个返回值为整型数指针的函数, `*b(w)` 本身就是错误的; 在答案 D 中, `b` 后面省掉了括号, 因为 `b` 是函数, 故也是错误的; 而对于答案 B 来说, 因为 `a` 指向了函数 `fun`, 所以它等价于 `func(&c)` 的调用, 是正确的。所以正确答案是 B。

(20) 若指针 `p` 已正确定义, 要使 `p` 指向两个连续的短整型动态存储单元, 不正确的语句是 ()。

- A. `p = 2*(short *) malloc (sizeof(short));`
- B. `p = (short *) malloc (2*sizeof(short));`
- C. `p = (short *) malloc (2*2);`
- D. `p = (short *) calloc(2, sizeof(short));`

【答案】 A。

【解释】 因为短整型数据在内存中占 2 字节, 而函数 `calloc` 也是动态内存分配的函数, 第一个参数表示分配的单元个数, 第二个参数表示每个分配单元的大小, 所以答案 B、C、D 都是正确的; 而答案 A 中, `2*` 应放在 `malloc` 后面的括号里面, 放在外面是错误的。所以选择答案是 A。

(21) 若有以下定义和语句, 则对 `s` 数组元素的正确引用形式是 ()。

```
int s[4][5], (*ps)[5];
ps = s;
```

A. ps+1 B. *(ps+3) C. ps[0][2] D. *(ps+1)+3

【答案】 C。

【解释】 ps 是一个指向二维数组 s 的指针，ps+1 表示指向数组 s 第二行首地址的指针；*(ps+3)表示数组 s 第四行的首地址；*(ps+1)+3 表示数组 s 第二行第四列元素的地址，答案 A、B、D 都不是对数组元素的正确引用。所以正确答案是 C。

(22) 有以下程序，若从键盘输入：abc def↵，则输出结果是（ ）。

```
void main( )
{
    char *p, *q;
    p = (char *) malloc(sizeof(char)*20); q = p;
    scanf ("%s%s", p, q); printf ("%s %s\n", p, q);
}
```

A. def def B. abc def C. abc abc D. def abc

【答案】 A。

【解释】 因为指针 p 和 q 都是指向动态分配的 20 字节的内存单元，当使用 scanf 来读取输入的字符串到该内存块时，首先将 abc 存入到指针 p 所指的内存块，输入中的空格符，表示字符串输入结束，下一字符串 def 将存入到 q 所指的内存块，即原先的字符串 abc 将被覆盖掉。所以输出的结构就应是 def def。即正确答案是 A。

(23) 有以下程序，程序运行后的输出结果是（ ）。

```
void ss (char *s, char t)
{
    while (*s)
    {
        if (*s == t) *s = t - 'a' + 'A';
        s++;
    }
}
void main( )
{
    char str1[100] = "abcddfefdbd", c = 'd';
    ss (str1, c); printf ("%s\n", str1);
}
```

A. ABCDDEFEDBD B. abcDDfefDbD
C. abcAAfefAbA D. Abcddfefdbd

【答案】 B。

【解释】 函数 ss 的功能就是将字符串 s 中的所有字符为 t 的字符替换为大写字符。对于主函数中的函数 ss 的调用，就是将字符串 str1 中的小写字符 d 替换成大写字符 D。

(24) 若定义了以下函数:

p 是该函数的形参，要求通过 p 把动态分配存储单元的地址传回主调函数，则形参 p 的正确定义应当是（ ）。

【答案】 C。

(25) 阅读程序，下面程序的输出结果是（ ）。

A. -386 net B. prog.exe -386 net
C. prog.exe - 386 net D. prog -386 net

【答案】 D。

2. 程序填空题

```
int mystrlen(char *str)
{
    int i;
    for (i = 0; *(str + i) != '\0' ; i++);
    return ( i );
}
```

【设计思想】 计算字符串的长度可以通过对字符串中的字符从左到右逐一与'\0'（字符串结尾标志）进行比较，直到等于'\0'为止。比较时通过变量 i（初值为 0）来控制比较的字符，循环结束后，变量 i 的值就是字符串的长度。

(2) 函数 sstrcmp()的功能是对两个字符串进行比较。当 s 所指字符串和 t 所指字符串相等时，返回值为 0；当 s 所指字符串大于 t 所指字符串时，返回值大于 0；当 s 所指字符串小于 t 所指字符串时，返回值小于 0（功能等同于库函数 strcmp()）。请填空。

```
#include <stdio.h>
int sstrcmp (char *s, char *t)
{
    while (*s && *t && *s == ____ t ____ )
    { s++; t++; }
    return ____ ( s == t ) ? 0 : ( s > t ? 1 : -1 ) ____ ;
}
```

【设计思想】 比较两个字符串的大小的方法是：对两个字符串从左到右逐个字符进行比较，如果两字符串均未遇到结尾符，并且 s 与 t 所指字符相等，则将指针 s 和 t 分别指向各自字符串中的下一字符，再进行比较，直到 s 或 t 指向字符串结尾符（此时*s 或*t 为 0）或 s 与 t 所指指向的字符不等才结束循环。循环结束后，若*s 等于*t（此时只有 s 和 t 都指向结尾符），则表明两字符串相等；否则，若*s 大于*t，则表明 s 所指字符串大于 t 所指的字符串；若*s 小于*t，则表明 s 所指字符串小于 t 所指字符串。

(3) 以下函数把 b 字符串连接到 a 字符串的后面，并返回 a 中新字符串的长度。请填空。

```
int Strcen (char a[ ], char b[ ])
{
    int num = 0, n = 0;
    while (*(a+num) != ____ '\0' ____ ) num++;
    while (b[n]) { *(a+num) = b[n]; num++; ____ n++ ____ ; }
    ____ (a+num) = '\0'; ____
    return (num);
}
```

【设计思想】 先计算字符串 a 的长度，其方法就是将 a 中的每个字符从左到右与'\0'进行比较，字符串长度由 num 来存放，然后通过 while 循环将字符串 b 中的每个字符逐一连入到字符串 a 的后面，直到 b 字符串结束。最后不要忘记在字符串 a 的最后加上'\0'。

(4) 函数 expand (char *s, *t) 在将字符串 s 复制到字符串 t 时，将其中的换行符和制表符转换为可见的转义字符，即用'\n'表示换行符，用'\t'表示制表符。请填空。

```
void expand ( char *s, char *t)
{
    int i, j;
    for (i = j = 0; s[i] != '\0'; i++)
```

```

switch (s[i])
{
    case '\n': t[ __j++__ ] = __'\\'__ ;
                t[j++] = 'n';
                __break__ ;
    case '\t': t[ __j++__ ] = __'\\'__ ;
                t[j++] = 't';
                break;
    default: t[ __j++__ ] = s[i];
                break;
}
t[j] = __ '\0'__ ;
}

```

【设计思想】 对字符串 *s* 中的每个字符进行判断, 如果是 '\n', 则首先将 \ 字符 (即 '\\') 赋值给 *t*, 同时 *t* 的下标变量 *j* 增 1, 然后将字符 'n' 赋值给 *t*, *j* 同样要增 1; 如果是 '\t', 则将 \ 字符 (即 '\\') 赋值给 *t*, 同时 *t* 的下标变量 *j* 增 1, 然后将字符 't' 赋值给 *t*; 如果都不是, 则直接将 *s* 中的字符赋值给 *t*, 下标变量 *j* 增 1。最后要在字符串 *t* 的最后加上 '\0'。

3. 编程题

(1) 编写一个交换变量值的函数, 利用该函数交换数组 *a* 和数组 *b* 中的对应元素值。

【设计思想】 用指针变量作为函数参数实现两数交换。定义两个指针变量作为形参, 在调用时分别指向两个数组的相应位置。

【参考答案】

```

#include <stdio.h>
#define SIZE 10
void swap (int *x, int *y);

void main( )
{
    int a[SIZE], b[SIZE], i, n;

    printf ("Input array length n<=%d: ", SIZE);
    scanf ("%d", &n);
    printf ("Input array a: ");
    for(i = 0; i < n; i++)
        scanf ("%d", &a[i]);
    printf ("Input array b: ");
    for(i = 0; i < n; i++)
        scanf ("%d", &b[i]);

    for(i = 0; i < n; i++)                //调用交换函数, 实现两数组元素互换
        swap (&a[i], &b[i]);
}

```



```
printf ("After swap:\n");
printf ("Array a: ");
for (i = 0; i < n; i++)
    printf ("%d ", a[i]);
printf ("\n");
printf ("Array b: ");
for (i = 0; i < n; i++)
    printf ("%d ", b[i]);
printf ("\n");
}
```

//交换整型指针x和y所指向的两个整数

```
void swap (int *x, int *y)
{
    int temp;

    temp = *x;
    *x = *y;
    *y = temp;
}
```

程序运行结果:

```
Input array length n<=10: 5✓
Input array a: 1 3 5 7 9✓
Input array b: 0 2 4 6 8✓
After swap:
Array a: 0 2 4 6 8
Array b: 1 3 5 7 9
```

(2) 不用 `strcat` 函数, 编写实现字符串连接函数 `strcat` 的功能, 将字符串 `t` 连接到字符串 `s` 的尾部。

【设计思想】 用 `i` 和 `j` 分别作为字符数组 `dstStr` 和字符数组 `srcStr` 的下标, 先将 `i` 和 `j` 同时初始化为 0, 然后移动 `i` 使其位于字符 `dstStr` 的尾部, 即字符串结束标志处, 再将字符数组 `srcStr` 中的字符依次复制到字符数组 `dstStr` 中。

【参考答案】 函数 `MyStrcat` 用下面两种方法实现。

方法一: 用字符数组编程实现函数 `MyStrcat`。

```
void MyStrcat (char dstStr[ ], char srcStr[ ])
{
    int i = 0, j = 0;                //数组下标初始化为0
```

```
while(dstStr[i] != '\0')
    i++;
while (srcStr[j] != '\0')
{
    dstStr[i] = srcStr[j];
    i++;
    j++;
}
dstStr[i] = '\0';           //在字符串dstStr的末尾添加一个字符串结束标志
}
```

方法二：用字符指针编程实现函数 MyStrcat。

```
void MyStrcat (char *dstStr, char *srcStr)
{
    while(*dstStr != '\0')//若srcStr所指字符不是字符串结束标志
        dstStr++;
    while (*srcStr != '\0')
    {
        *dstStr = *srcStr;    //将srcStr所指字符复制到dstStr所指的存储单元中
        dstStr++;            //将dstStr指向下一个字符
        srcStr++;            //将srcStr指向下一个字符
    }
    *dstStr = '\0';          //在字符串dstStr的末尾添加一个字符串结束标志
}
```

主函数程序如下：

```
#include <stdio.h>

void main( )
{
    char s[80], t[80];

    printf ("Please enter the first string: ");
    gets(s);
    printf ("Please enter the second string: ");
    gets(t);
    MyStrcat (s, t);
    printf ("The concat is: ");
    puts(s);
}
```

程序运行结果：

```
Please enter the first string: abcd✓  
Please enter the second string: efgh✓  
The concat is: abcdefgh
```

(3) 编程判断输入的一串字符是否为“回文”。所谓“回文”是指顺读和倒读都一样的字符串。如, "level"、"ABCCBA"都是回文。

【设计思想】 由题意可知, 回文就是一个对称的字符串, 利用这一特点可采用如下算法:

- 设置两个指针 pStart 和 pEnd, 让 pStart 指向字符串首部, 让 pEnd 指向字符串尾部。
- 利用循环, 从字符串两边对指针所指字符进行比较, 当对应的两字符相等且两指针未超越对方时, 使指针 pStart 向后移动一个字符位置 (即加 1), 使指针 pEnd 向前移动一个字符位置 (即减 1), 一旦发现对应的两个字符不等或两指针已相互超越 (不可能是回文), 则立即停止循环。
- 根据退出循环时两指针的位置, 判断字符串是否为回文。

【参考答案】

```
#include <stdio.h>  
#include <string.h>  
  
void main( )  
{  
    char str[80], *pStart, *pEnd;  
    int len;  
  
    printf ("Input String: ");  
    gets (str);  
    len = strlen (str);  
    pStart = str;  
    pEnd = str + len - 1;  
    while (*pStart == *pEnd && pStart <= pEnd)  
    {  
        pStart++;  
        pEnd--;  
    }  
    if(pStart < pEnd)  
        printf ("No!\n");  
    else  
        printf ("Yes!\n");  
}
```

程序运行结果:

```
Input string: abcba✓
Yes!
```

(4) 编写一取某字符串子串的函数 `char *substr (char *s, int startloc, int len)`, 其中 `s` 为字符串, `startloc` 为起始位置 (0 表示第一个字符的位置), `len` 为子串的长度。要求返回值为求得的子串。例如, 调用 `substr ("12345678", 0, 4)`, 求得的子串为 "1234"。

【设计思想】 因为该函数返回指向子串的字串型指针, 所以在函数中可定义一静态字串型数组 `sstr`, 用于存放所取得的子串。取子串的方法可以从字符串 `s` 的第 `startloc` 个字符开始复制到静态字串数组 `sstr` 中, 直到已复制了 `len` 个字符或遇到了字符串结尾标志。然后将 `sstr` 返回即可。

【参考答案】

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

char *substr (char *s, int startloc, int len);

void main( )
{
    char str[80], s, l;

    printf ("Input string: ");
    gets (str);
    printf ("Start location: ");
    scanf ("%d", &s);
    printf ("Substring length: ");
    scanf ("%d", &l);
    printf ("The substring is: %s\n", substr(str, s, l));
}

char *substr (char *s, int startloc, int len)
{
    static char sstr[80];    //用于存放所取得子串
    int i;

    if((startloc < 0) || (startloc >= strlen(s)) || (len < 0))
        //判断起始位置和子串长度是否合法
        return (NULL);
    for (i = 0; i < len && s[startloc+i] != '\0'; i++)
        sstr[i] = s[startloc+i];
```

```
sstr[i] = '\\0';  
return (sstr);  
}
```

程序运行结果:

```
Input string: 12345678✓  
Start location: 0✓  
Substring length: 4✓  
The substring is: 1234
```

(5) 编写一函数 `strlshif (char *s, int n)`, 其功能是把字符串 `s` 中的所有字符左移 `n` 个位置, 串中的前 `n` 个字符移到最后。

【设计思想】 要实现将字符串 `s` 循环左移 `n` 个字符, 可通过 `n` 次循环。首先将字符串的第一个字符 `s[0]` 保存到临时字符变量 `ch` 中, 然后利用 `strncpy` 函数将字符串 `s` 中第二个字符开始共 `strlen(s)-1` 个字符复制到 `s` 的第一个字符的位置, 最后将 `ch` 放到 `s` 的最后一个字符的位置。整个 `n` 次循环结束就实现了字符串循环左移 `n` 位的操作。

【参考答案】

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

```
void strlshif (char *s, int n);
```

```
void main( )  
{  
    char str[ ] = "0123456789";  
  
    strlshif (str, 3);  
    printf ("%s\n", str);  
}
```

//将字符串s循环左移n个字符

```
void strlshif (char *s, int n)
```

```
{  
    int i, len;  
    char ch;  
  
    len = strlen (s);  
    for (i = 0; i < n; i++)  
    {  
        ch = s[0];
```

//将字符串的第一个字符保存在变量ch中

```

    strncpy (s, s+1, len-1);    //将第二个字符开始的len-1个字符左移
    s[len-1] = ch;             //将原先第一个字符置于字符串的末尾
}
}

```

程序运行结果:

```
3456789012
```

(6) 编写一个函数 fun，它的功能是：删除字符串中的数字字符。例如输入字符串 48CTYP9E6，则输出 CTYPE。

【设计思想】 要删除字符串 s 中的数字字符，可预先设置一字符指针变量 pstr，pstr 首先指向 s 的第一个字符，然后通过循环使用 pstr 来遍历该字符串的每个字符。如果 pstr 所指字符是数字字符，则利用 strcpy 函数将 pstr+1 所指向的后续字符串复制到 pstr 所指向的字符串，也就是将原来的数字字符覆盖掉，直到 pstr 所指向的字符是非数字字符，接着 pstr 加 1，即右移一个字符位置。重复上述过程，直到 pstr 所指字符为字符串结尾符'\0'为止。

【参考答案】

```

#include <stdio.h>
#include <string.h>

void func(char *s);

void main( )
{
    char str[80];

    printf ("Input string: ");
    gets (str);
    func (str);
    printf ("After delete digital char: %s\n", str);
}

//删除字符串s中的数字字符
void func(char *s)
{
    char *pstr;

    for(pstr = s; *pstr != '\0'; pstr++)
    {
        while(*pstr >= '0' && *pstr <= '9')
            strcpy (pstr, pstr+1);
    }
}

```

```
}  
}
```

程序运行结果:

```
Input string: 48CTYP9E6✓  
After delete digital char: CTYPE
```

(7) 编写一个函数 `totsubstrnum(char *str, char *substr)`, 它的功能是: 统计子字符串 `substr` 在字符串 `str` 中出现的次数。

【设计思想】 要统计子串 `substr` 在源字符串 `str` 中出现的次数, 可设置 `str` 的一下标变量 `i`, 其初始值为 0, 并计算源串 `str` 的长度 `len1` 和子串 `substr` 的长度 `len2`, 通过循环, 使用 `strncmp` 函数将 `str+i` 开始的 `len2` 个字符与子串 `substr` 进行比较。如果相等, 表明找到了子串, 即子串出现次数 `count` 加 1, 此时 `i` 加 `len2`; 如果不等, 则 `i` 增 1, 然后继续循环, 直到 `i` 大于 `len1-len2` 为止。

【参考答案】

```
#include <stdio.h>  
#include <string.h>  
  
int totsubstrnum (char *str, char *substr);  
  
void main( )  
{  
    char str[80], substr[80];  
  
    printf ("Input string: ");  
    gets (str);  
    printf ("Input substring: ");  
    gets (substr);  
    printf ("count = %d\n", totsubstrnum(str, substr));  
}  
  
//统计子串substr在字符串str中出现的次数  
int totsubstrnum (char *str, char *substr)  
{  
    int i = 0, count = 0, len1, len2;  
  
    len1 = strlen (str);  
    len2 = strlen (substr);  
    while (i <= len1-len2)  
    {  
        if (strncmp (str+i, substr, len2) == 0)
```

```
{
    count++;
    i += len2;
}
else
    i++;
}
return (count);
}
```

程序运行结果:

```
Input string:  abc12ab3abcdef✓
Input substring:  abc✓
count = 2
```

(8) 编写一带命令行参数的程序输出星期一到星期日的英文名。

【设计思想】 定义一字符串指针数组 `week_str`, 并将星期一到星期日的英文名以字符串的形式作为初始值赋值给 `week_str`。命令行参数的第一个参数是执行文件名, 第二个参数才是星期几的值 (0 表示星期日, 1~6 表示星期一到星期六), 程序的关键首先要判断命令行参数是否在 0~6 之间, 即是否合法? 如果非法, 则要给出提示信息, 否则将其转换成整型数值, 程序中的自定义函数 `GetWeek` 就是完成该项功能。得到正确的星期值后, 利用指针数组 `week_str` 引用其相应的元素就可得到对应的星期英文名。

【参考答案】

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

int GetWeek(char *str);

char *week_str[ ] = { "sunday", "monday", "tuesday", "wednesday",
                      "tuesday", "friday", "saturday" };

void main (int argc, char *argv[ ])
{
    int k;

    if ( argc != 2)
    {
        printf("Use this program like this: example9-8 3");
        exit(0);
    }
}
```



```
k = GetWeek(argv[1]);
if (k == -1)
{
    printf("command line argument must be in [0..6]\n");
    exit(0);
}
printf("%s\n", week_str[k]);
}
```

//功 能：判断命令行参数中的星期数是否合法，
//返回值：合法返回星期的整型值，不合法则返回-1

```
int GetWeek(char *str)
{
    int k, i = 0;

    while(str[i] != 0)
    {
        if (str[i] < '0' || str[i] > '9')
            return(-1);
        else
            i++;
    }
    k = atoi(str);
    if (k < 0 || k > 6)
        return(-1);
    return(k);
}
```

程序运行结果：

```
example9-8 3✓
wednesday
```

(9) 编写一程序用于计算任意大的两整数之差（提示：大整数用字符串来表示）。

【设计思想】 将两个大整数分别用两个字符型数组 a 和 b 来表示，每个数字以其字符形式存放在数组中，进行求差之前，先判断两数的大小，保证被减数的值大于减数的值，然后对两数按照从右向左的顺序进行数字字符的相减，求得其差值放在字符数组 c 中。

【参考答案】

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

```

#define N 20 //数字字符的个数
void beep( );
void GetNumberStr (char *s);
void SubNumberStr (char *a, char *b, char *c);
char SubChar (char ch1, char ch2);
void LeftTrim (char *str, char sign);

int tag = 0; //借位标志

void main( )
{
    char a[N+1] = {0}, b[N+1] = {0}, c[N+2];

    //输入被减数a、减数b
    printf ("a = ");
    while (strlen(a) == 0)
        GetNumberStr (a);
    printf ("\nb = ");
    while (strlen(b) == 0)
        GetNumberStr (b);

    //计算差数c
    SubNumberStr (a, b, c);

    //显示计算结果
    printf ("\na - b = %s \n", c);
}

void GetNumberStr (char *s) //读取数字字符串
{
    int i = 0;
    char ch;

    while (1)
    {
        ch = getch( );
        if(ch == '\r') //回车符, 退出
            break;
        if(ch == '\b') //退格符
        {
            if(i > 0)
            {
                printf ("%c %c", ch, ch);

```

```

        i--;
    }
    else
        beep( );
    continue;
}
if(ch < '0' || ch > '9')           //非数字字符
{
    beep( );
    continue;
}
if (i < N)                          //数字字符
{
    printf ("%c", ch);
    s[i++] = ch;
}
else
    beep( );
}
s[i] = '\0';                       //置字符串结束标志
}

void beep( )                        //响铃
{
    printf("\07");
}

//计算两个数字字符串之差，放在c中
void SubNumberStr (char *a, char *b, char *c)
{
    int i, j, k;
    char t[N+1], sign = ' ';

    //将大数放在a中,小数放在b中
    if(strlen(a)<strlen(b)||strlen(a) == strlen(b) && strcmp(a, b)<0)
    {
        sign = '-';
        strcpy (t, a);
        strcpy (a, b);
        strcpy (b, t);
    }

    memset (c, ' ', N+2);           //将c全部清空

```

```

    i = strlen (a) - 1;
    j = strlen (b) - 1;
    k = N;
    while (i >= 0 && j >= 0)          //将被减数与减数按照从右向左的顺序相减
        c[k--] = SubChar (a[i--], b[j--]);
    while (i >= 0)                    //被减数没有减完
        c[k--] = SubChar (a[i--], '0');
    c[N+1] = '\0';                    //置字符串结束标志
    LeftTrim (c, sign);               //去掉字符串c左边的空格和0
}

//计算两个数字字符之差
char SubChar (char ch1, char ch2)
{
    char ch;

    ch = ch1 - ch2 - tag;             //两数字字符所对应的数字与借位相减
    if (ch >= 0)                      //结果大于或等于0
    {
        tag = 0;                     //没借位
        return (ch + 0x30);           //将差数加上0x30转换成其数字字符
    }
    else                              //结果小于0
    {
        tag = 1;                     //有进位
        return (ch + 10 + 0x30);      //将差数加上10再加0x30转换成其数字字符
    }
}

//去掉字符串左边的空格和0
void LeftTrim (char *str, char sign)
{
    int i;

    for(i=0; str[i] == ' ' || str[i] == '0'; i++)
        ;                            //查找第一个既非空格字符也非0字符的位置
    if(str[i] == '\0')                 //如果差值为0
        i--;
    if(sign == '-')                    //如果为负数,添加负号
        str[--i] = sign;
    strcpy (str, str+i);
}

```

有参数的宏时，参数类型可以与宏定义时的类型不一致，因为宏替换时是纯文本替换；对于宏定义：`#define C R 045`，C 是宏名，而不是 C R；所以答案 A、B、D 都是错误的。正确答案是 C，因为宏替换是在编译时进行的，不是在执行时进行的。

(4) 以下程序运行的结果是 ()。

```
#define ADD(x) x+x
void main( )
{
    int m = 1, n = 2, k = 3;
    int sum = ADD(m+n) * k;
    printf ("sum = %d", sum);
}
```

A. sum = 9 B. sum = 10 C. sum = 12 D. sum = 18

【答案】 B。

【解释】 语句 `sum = ADD(m+n) * k`；在预编译时将进行宏替换为：`sum = m+n+m+n * k`；因 `m = 1`，`n = 2`，`k = 3`，所以语句等价于 `sum = 1+2+1+2*3`；因此 sum 的值将是 10。故正确答案是 B。

(5) 程序中头文件 `typel.h` 的内容是：

```
#define N 5
#define M1 N*3
```

程序如下：

```
#define "typel.h"
#define M2 N*2
void main( )
{
    int i;
    i = M1 + M2;
    printf ("%d\n", i);
}
```

程序编译后运行的输出结果是 ()。

A. 10 B. 20 C. 25 D. 30

【答案】 C。

【解释】 语句 `i = M1 + M2`；在预编译时将进行宏替换为：`i = 5*3 + 5*2`；因此 i 的值将是 25。故正确答案是 C。

(6) 以下叙述正确的是 ()。

- A. 可以把 `define` 和 `if` 定义为用户标识符
- B. 可以把 `define` 定义为用户标识符，但不能把 `if` 定义为用户标识符
- C. 可以把 `if` 定义为用户标识符，但不能把 `define` 定义为用户标识符

D. `define` 和 `if` 都不能定义为用户标识符

【答案】 B。

【解释】 C 语言中，除了关键字以外的单词串都可以定义成标识符。`define` 不是关键字，所以可以把 `define` 定义成标识符，但 `if` 是关键字，所以不能定义成标识符。故正确答案是 B。

(7) 若有以下说明和定义，则叙述正确的是 ()。

```
typedef int *INTEGER;
INTEGER p, *q;
```

- A. `p` 是 `int` 型变量
- B. `p` 是基类型为 `int` 的指针变量
- C. `q` 是基类型为 `int` 的指针变量
- D. 程序中可用 `INTEGER` 代替 `int` 类型名

【答案】 B。

【解释】 进行类型定义以后，`INTEGER` 其实就是基类型为 `int` 的指针类型，然后由 `INTEGER` 来定义两个变量 `p` 和 `*q`，则 `p` 是基类型为 `int` 型的一级指针变量，而 `q` 是基类型为 `int` 型的二级指针变量。故正确答案是 B。

(8) 从下列选项中选择不会引起二义性的宏定义是 ()。

- A. `#define POWER(x) x * x` B. `#define POWER(x) (x) * (x)`
- C. `#define POWER(x) (x * x)` D. `#define POWER(x) ((x) * (x))`

【答案】 D。

【解释】 要想让宏定义不产生二义性，就是将参数及宏体都加上小括号即可。故正确答案是 D。

(9) 以下程序运行的结果是 ()。

```
#define X 5
#define Y X + 1
#define Z Y * X / 2
void main( )
{
    int a = Y;
    printf ("%d, %d", Z, --a);
}
```

- A. 7, 6 B. 12, 6 C. 12, 5 D. 7, 5

【答案】 D。

【解释】 先来计算 `Z` 的值，`Z` 将替换成 `Y * X / 2`，再把 `X + 1` 替换 `Y`，得 `Z = X + 1 * X / 2`，再用 5 替换 `X`，得 `Z = 5 + 1 * 5 / 2`，所以 `Z` 的值是 7（注意整除）。`a = Y` 宏替换后变成 `a = X + 1`，再将 5 替换 `X`，得 `a = 5 + 1`，即 `a` 的值是 6，但输出是 `--a`，`a` 先减 1 即变为 5，再输出。故正确答案是 D。

1.11 习题 11 参考答案和解释

1. 选择题

(1) 当说明一个结构体变量时系统分配给它的内存是 ()。

- A. 各成员所需内存量的总和
- B. 结构中第一个成员所需内存量
- C. 成员中占内存量最大者所需的容量
- D. 结构中最后一个成员所需内存量

【答案】 A。

【解释】 C 语言中, 结构体变量所占内存的大小是其所包含的各个成员所需内存大小之和。故正确答案是 A。

(2) 设有以下说明语句, 则下面的叙述不正确的是 ()。

```
struct stu
{
    int a;
    float b;
} stutype;
```

- A. struct 是结构体类型的关键字
- B. struct stu 是用户定义的结构体类型
- C. stutype 是用户定义的结构体类型名
- D. a 和 b 都是结构体成员名

【答案】 C。

【解释】 stutype 是类型为 struct stu 的结构体变量, 不是结构体类型名。故选择答案是 C。

(3) C 语言结构体类型变量在程序执行期间 ()。

- A. 所有成员一直驻留在内存中
- B. 只有一个成员驻留在内存中
- C. 部分成员驻留在内存中
- D. 没有成员驻留在内存中

【答案】 A。

【解释】 结构体变量不管其包含有多少个成员, 都应当看成是一个整体。在程序运行期间, 只要在变量的生存期内, 所有成员一直驻留在内存中, 不可能出现有的成员驻留内存, 有的成员不驻留内存的情况。故选择答案是 A。

(4) 在 16 位 IBM-PC 机上使用 C 语言, 若有如下定义:

```
struct data
{
    int i;
    char ch;
```



```
double f;
} b;
```

则结构变量 **b** 占用内存的字节数是 ()。

- A. 1 B. 2 C. 8 D. 11

【答案】 D。

【解释】 结构体变量 **b** 所占内存的字节数是其成员所占内存字节数之和。在 16 位机中, 成员 **i** 是整型变量, 占 2 字节, **ch** 是字符型变量, 占 1 字节, **f** 是双精度实型变量, 占 8 字节, 总共占 11 字节。所以正确答案是 D。

(5) 根据下面的定义, 能打印出字母 **M** 的语句是 ()。

```
struct person
{
    char name[9];
    int age;
};
struct person class[10]={"John", 17, "Paul", 19, "Mary", 18, "adam", 16 };
```

- A. printf ("%c\n", class[3].name); B. printf ("%c\n", class[3].name[1]);
C. printf ("%c\n", class[2].name[1]); D. printf ("%c\n", class[2].name[0]);

【答案】 D。

【解释】 包含大写字母 **M** 的结构体, 只有结构体数组的第 3 个元素 **class[2]**, **class[2]** 包含两个成员: 一个姓名 **name** 字符数组, 一个是年龄 **age**, **name** 的值是一字符串 "Mary", 其第一个字符 **name[0]** 就是字符 **M**, 所以通过结构体数组来输出字母 **M** 的引用是 **class[2].name[0]**。故正确答案是 D。

(6) 以下对结构体类型变量的定义中, 不正确的是 ()。

- A. typedef struct aa B. #define AA struct aa
 { AA {
 int n; int n;
 float m; float m;
 } AA; } td1;
AA td1;
C. struct D. struct
 { {
 int n; int n;
 float m; float m;
 } aa; } td1;
stuct aa td1;

【答案】 C。

【解释】 在答案 C 中, **aa** 是结构体变量, 不是结构体类型名, 不能再利用 **aa** 来定

义别的结构体变量，而且 `struct` 也是非法的。所以应选择 C。

(7) 设有如下定义：

```
struct sk
{
    int a;
    float b;
} data;
int *p;
```

若要使 `p` 指向 `data` 中的 `a` 域，正确的赋值语句是 ()。

A. `p = &a;` B. `p = data.a;` C. `p = &data.a;` D. `*p = data.a;`

【答案】 C。

【解释】 要使 `p` 指向 `data` 中的 `a` 成员，则只需将 `data` 中成员 `a` 的地址，即 `&data.a` 赋值给指针变量 `p` 即可。故正确答案是 C。答案 A 中没有指明结构体变量，直接对成员引用是非法的；答案 B 中不能将 `data` 的成员 `a` 的值（整型数）赋给一指针变量；而答案 D 中，只是将 `data` 的成员 `a` 的值赋值给指针 `p` 所指向的内存单元，不符合题目的要求。

(8) 有以下结构体说明和变量的定义，且指针 `p` 指向变量 `a`，指针 `q` 指向变量 `b`。则不能把结点 `b` 连接到结点 `a` 之后的语句是 ()。

```
struct node
{
    char data;
    struct node *next;
} a, b, *p = &a, *q = &b;
```

A. `a.next = q;` B. `p.next = &b;` C. `p->next = &b;` D. `(*p).next = q;`

【答案】 B。

【解释】 `p` 是一指向结构体类型的指针变量，其初始值是让其指向结构体变量 `a`，要通过指针变量 `p` 来引用结构体变量的成员 `next`，通常可使用两种形式：`p->next` 或 `(*p).next`，而 `p.next` 是非法的引用。故选择答案是 B。

(9) 有以下程序，程序运行后的输出结果是 ()。

```
struct STU
{ char num[10]; float score[3]; };
void main( )
{
    struct STU s[3]={{"20021", 90, 95, 85},
                      {"20022", 95, 80, 75},
                      {"20023", 100, 95, 90}}, *p = s;

    int i; float sum = 0;
    for(i = 0; i < 3, i++)
        sum = sum + p->score[i];
```

```
printf ("%6.2f\n", sum);
}
```

A. 260.00 B. 270.00 C. 280.00 D. 285.00

【答案】 B。

【解释】 s 是具有 3 个元素的结构体数组，p 为一指向结构体的指针变量，开始时将数组的首地址赋给 p，即 p 指向了数组 s 的第一个元素 s[0]，所以通过 p 对结构体成员 score 的引用将是 s[0] 中的成员 score，也就是说，程序中的 for 循环其实是统计 s[0] 中的成员数组 score 的 3 个元素值之和，则 sum 的值就是 $90+95+85=270$ 。故正确答案是 B。

(10) 在 16 位 IBM-PC 机上，若有下面的说明和定义：

```
struct test
{
    int m1; char m2; float m3;
    union uu {char u1[5]; int u2[2];} ua;
} myaa;
```

则 sizeof(struct test) 的值是 ()。

A. 12 B. 16 C. 14 D. 9

【答案】 A。

【解释】 sizeof(struct test) 其实就是计算结构体变量 myaa 所占内存的字节数，它应当等于该结构体变量所包含的各个成员所占内存字节数之和，该结构体变量共包含 4 个成员：m1 整型变量在 16 位机中占 2 字节，m2 字符型变量占 1 字节，m3 浮点型变量占 4 字节，ua 联合体变量，其所占字节数由联合体中所占字节数最多成员来决定，在联合体 uu 中 u1 是大小为 5 的字符数组，所占字节数最多，因此 ua 占用内存字节数为 5，这样加起来就是 $2+1+4+5=12$ 字节。故正确答案是 A。

(11) 以下程序的输出是 ()。

```
struct st { int x; int *y;} *p;
int dt[4] = {10, 20, 30, 40 };
struct st aa[4]={ 50, &dt[0], 60, &dt[0], 60, &dt[0], 60, &dt[0]};
void main( )
{
    p = aa;
    printf ("%d\n", ++(p->x));
}
```

A. 10 B. 11 C. 51 D. 60

【答案】 C。

【解释】 aa 是大小为 4 的结构体数组，p 是一指向该结构体的指针，将数组名 aa 赋给 p，p 将指向数组 aa 的第一个元素 aa[0]，p->x 其实就是利用指针 p 来引用 aa[0] 的成员 x，其值为 50，加 1 后输出的结果为 51。故正确答案是 C。

(12) 以下程序的输出是 ()。

```

union myun
{
    struct { int x, y, z; } u;
    int k;
} a;
void main( )
{
    a.u.x = 4; a.u.y = 5; a.u.z = 6;
    a.k = 0;
    printf ("%d\n", a.u.x);
}

```

A. 4 B. 5 C. 6 D. 0

【答案】 D。

【解释】 变量 a 是一联合体变量，它包含两个成员：一个是结构体类型成员 u，一个是整型变量 k，联合体变量 a 所占内存大小由结构体成员 u 来决定，k 与结构体成员 u 的成员 x 共用同一内存单元，所以执行 a.k=0 相当于执行 a.u.x=0。故正确答案是 D。

(13) 以下程序的输出是 ()。

```

struct HAR { int x, y; struct HAR *p; } h[2];
void main( )
{
    h[0].x = 1; h[0].y = 2;
    h[1].x = 3; h[1].y = 4;
    h[0].p = &h[1]; h[1].p = h;
    printf ("%d, %d \n", (h[0].p)->x, (h[1].p)->y);
}

```

A. 1,2 B. 2,3 C. 1,4 D. 3,2

【答案】 D。

【解释】 h 是大小为 2 的结构体数组，对其两个元素进行赋值后得内存映像如图 1-1 所示，其输出结果应为 3, 2。故正确答案是 D。

(14) 以下程序的输出是 ()。

```

struct NODE { int num; struct NODE *next; };
void main( )
{
    struct NODE *p, *q, *r;
    p = (struct NODE *) malloc(sizeof(struct NODE));
    q = (struct NODE *) malloc(sizeof(struct NODE));
    r = (struct NODE *) malloc(sizeof(struct NODE));
    p->num = 10; q->num = 20; r->num = 30;
}

```

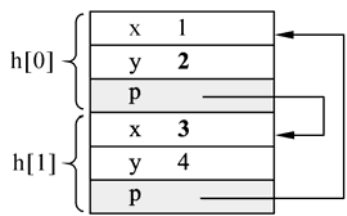


图 1-1 结构体数组 h 的内存映像

```

p->next = q; q->next = r;
printf ("%d\n", p->num + q->next->num);
}

```

A. 10 B. 20 C. 30 D. 40

【答案】 D。

【解释】 p、q、r 是 3 个指向结构体的指针变量，通过动态内存分配分别为它们分配所对应的内存块，即 3 个结点，然后对其成员赋值，这样 3 个结点就构成了一单链表。其内存映像如图 1-2 所示，p->num 的值为 10，q->next->num 其实就是 r.num，其值为 30，所以输出结果为 40。故正确答案是 D。

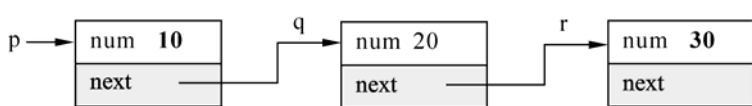


图 1-2 p、q、r 三个结构体指针变量内存映像

(15) 执行以下语句后的输出结果是 ()。

```

enum weekday {sun, mon = 3, tue, wed, thu};
enum weekday workday;
workday = wed;
printf ("%d\n", workday);

```

A. 5 B. 3 C. 4 D. 编译错

【答案】 A。

【解释】 在枚举类型 weekday 中，因 mon 的值为 3，则 tue 的值为 4，wed 的值就为 5，所以将 wed 赋值给枚举类型变量 workday，其值就是 5。故正确答案是 A。

2. 程序填空题

(1) 从键盘上顺序输入整数，直到输入的整数小于 0 时才停止输入。然后反序输出这些整数。请填空。

```

#include <stdio.h>
struct data { int x; struct data *link; } *p;
void input( )
{
    int num;
    struct data *q;
    printf ("Enter data: ");
    scanf ("%d", &num);
    if (num < 0) _____ return _____ ;
    q = _____(struct data *) malloc (sizeof(struct data))_____ ;
    q->x = num; q->link = p;
}

```

```

    p = q;      input( ) ;
}
void main( )
{
    printf ("Enter data until data < 0: \n");
    p = NULL;
    input( );
    printf ("Output: ");
    while ( p != NULL )
    {
        printf ("%d\n", p->x);
        p = p->link ;
    }
}

```

【设计思想】 以链表的形式来存储每次输入的整数。每次输入整数后，如果大于或等于 0，通过动态内存分配来产生链表的结点。然后把输入的值保存到结点中，再把结点插入到先前建立的链表之前，这样建立的链表，最早输入的数所在的结点将在最后，而最后输入的数的结点则在最前。注意指针 p 总是指向链表的首结点。链表建立后，通过头指针 p 从左到右来遍历链表，就实现了反序输出。

(2) 下面的程序从终端上输入 5 个人的年龄、性别和姓名，然后输出。请填空。

```

#include <stdio.h>
struct man { char name[20]; unsigned age; char sex[7]; } ;
void data_in (struct man *p, int n);
void data_out (struct man *p, int n);
void main( )
{
    struct man person[5];
    data_in (person, 5);
    data_out (person, 5);
}
void data_in (struct man *p, int n)
{
    struct man *q = p + n ;
    for (; p < q; p++)
    {
        printf ("age:sex:name");
        scanf ("%u%s", &p->age, p->sex);
        gets (p->name) ;
    }
}
void data_out (struct man *p, int n)
{

```

```
struct man *q = p + n ;  
for(; p < q; p++)  
    printf ("%s;%u%s\n", p->name, p->age, p->sex);  
}
```

【设计思想】 结构体数组名是该结构体数组的首地址。通过定义两个函数 `data_in` 和 `data_out` 来实现对结构体数组每个元素的赋值和输出，函数的形参分别是指向结构体的指针及数组元素的大小，在实现过程中定义一结构体指针 `q`，将 `q` 指向最后一个数组元素的下一个元素，用于循环控制，循环时将 `p` 与 `q` 进行反复比较，直到 `p` 等于 `q` 时结束。

3. 编程题

(1) 已知学生的记录由学号和学习成绩构成，输入 `n` 个学生的学号和成绩。找出成绩最低的学生记录（假定最低成绩的记录是惟一的），通过函数返回。请编写程序实现该要求。

【设计思想】 定义一个学生信息的结构体类型 `StuMsg`，其包含两个成员：一个是学号 `no`，一个是成绩 `score`。首先输入学生实际人数 `num`，然后调用函数 `StuMsg_In` 来输入学生记录，输入之前，通过动态内存分配函数 `malloc` 建立具有 `num` 个学生记录信息的内存块，其首地址赋给指针变量 `p`，学生信息输入完后，将指针 `p` 返回。再通过调用函数 `GetMinStuMsg` 来获得成绩最低学生的记录信息，然后输出结果。

【参考答案】

```
#include <stdio.h>  
#include <malloc.h>  
#include <stdlib.h>  
  
struct StuMsg  
{  
    char no[10];  
    float score;  
};  
struct StuMsg *StuMsg_In (int num);  
struct StuMsg GetMinStuMsg (struct StuMsg *pstu, int num);  
  
void main( )  
{  
    int num;  
    struct StuMsg *pstu, minstu;  
  
    printf ("Input student number: "); //输入学生实际人数  
    scanf ("%d", &num);  
    pstu = StuMsg_In (num);  
    if (pstu == NULL)  
        exit(0);
```

```
    minstu = GetMinStuMsg (pstu, num);
    printf ("The lowest score student record is: ");
    printf ("No = %s Score = %.2f\n", minstu.no, minstu.score);
    free (pstu); //释放动态内存分配
}

//建立num个学生的成绩记录
struct StuMsg *StuMsg_In (int num)
{
    int i;
    struct StuMsg *p;

    p = (struct StuMsg *) malloc (num * sizeof(struct StuMsg));
    if (p == NULL)
    {
        printf ("Not enough memory!");
        return (NULL);
    }
    for (i = 0; i < num; i++)
    {
        printf ("Student %d: ", i+1);
        scanf ("%s%f", p[i].no, &p[i].score);
    }
    return (p);
}

//查找学生成绩最低的记录
struct StuMsg GetMinStuMsg (struct StuMsg *pstu, int num)
{
    int i, k;

    k = 0;
    for(i = 1; i < num; i++)
        if(pstu[i].score < pstu[k].score)
            k = i;
    return (pstu[k]);
}
```

程序运行结果:

```
Input student number: 3✓
Student 1: 0001 90✓
Student 2: 0001 80✓
Student 3: 0001 95✓
The lowest score student record is: No = 0001 Score = 80.00
```


(2) 每个产品销售记录由产品代码 dm(字符型 4 位), 产品名称 mc(字符型 10 位), 单价 dj(整型), 数量 sl(整型), 金额 je(长整型)四部分组成。其中: 金额=单价*数量计算得出。请编制函数 SortDat(), 其功能要求: 按产品代码从大到小进行排列, 若产品代码相同, 则按金额从大到小进行排列。

【设计思想】 定义一个产品信息的结构体类型 PRO, 其包含题目所要求的 5 个成员。首先输入产品实际销售记录数 n, 然后调用函数 Data_In 来输入产品销售记录, 输入之前, 通过动态内存分配函数 malloc 建立具有 n 个产品记录信息的内存块, 其首地址赋给指针变量 p, 产品信息输入完后, 将指针 p 返回。再通过调用函数 Data_Sort 来对产品信息按代码从小到大排序, 代码相同时按金额从大到小来排序, 最后调用函数 Data_Out 输出排序后的结果。

【参考答案】

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

typedef struct
{
    char dm[5];           //产品代码
    char mc[11];          //产品名称
    int  dj;              //单价
    int  sl;              //数量
    long je;              //金额
} PRO;

PRO *Data_In (int n);
void Data_Out (PRO *p, int n);
void Data_Sort(PRO *p, int n);

void main( )
{
    int n;
    PRO *p;

    printf ("Input the product number: ");
    scanf ("%d", &n);
    p = Data_In (n);
    if(p==NULL)
        exit(0);
    Data_Sort (p, n);
    Data_Out (p, n);
```

```
    free (p);
}

//产品信息输入
PRO *Data_In (int n)
{
    PRO *p;
    int i;

    p = (PRO *) malloc (n * sizeof(PRO));
    if(p == NULL)
    {
        printf ("Not enough Memory!");
        return (NULL);
    }
    printf ("Input product message(dm,mc,dj,sl):\n");
    for(i = 0; i < n; i++)
    {
        printf ("Product %d: ", i+1);
        scanf ("%4s%10s%d%d", p[i].dm, p[i].mc, &p[i].dj, &p[i].sl);
        p[i].je = p[i].dj * p[i].sl;
    }
    return (p);
}

//产品信息输出
void Data_Out (PRO *p, int n)
{
    int i;

    printf ("After sort:\n");
    for(i = 0; i < n; i++)
        printf ("%4s %10s %4d %5d %10Ld\n", p[i].dm, p[i].mc, p[i].dj, p[i].sl,
            p[i].je);
}

//产品信息排序(代码从大到小, 代码相同时金额从大到小)
void Data_Sort (PRO *p, int n)
{
    int i, j;
    PRO swap;

    for(i = 0; i < n-1; i++)
        for(j = i+1; j < n; j++)
```

```

        if (strcmp(p[j].dm,p[i].dm) > 0 || strcmp(p[j].dm,p[i].dm) == 0
            && p[j].je > p[i].je)
        {
            swap = p[i];
            p[i] = p[j];
            p[j] = swap;
        }
    }
}

```

(3) 编写一个程序，该程序可以完成个人财务管理。每个人的财务项目应当包括姓名、年度、收入、支出等。为了叙述简单，以一个财政年度为统计单位，程序中可以计算每个人的每个财政年度的收入总额、支出总额、存款总额等，并能够打印出来。需要注意的是，收入总额不可能只输入一次，而可能是多次收入的和；同样地，支出总额也不可能只是一次支出，应是多次支出的总和。

【设计思想】 本程序是一个财务管理程序，涉及收入和支出，虽然是个人财务管理程序，但最好能够按照一种标准的财务管理软件来考虑，所以在程序设计时，需要考虑如下几个因素：

- 每一笔收入或支出都可以理解为一笔交易，那么程序最多可以容纳多少笔交易决定数组的元素个数，必须有一个预测，不妨先假定为 50 笔。
- 确定结构体形式时需要认真考虑，它关系到程序实现的思路 and 方式。一个人的收入和支出显然是多次输入的，而每一笔交易必须记录交易的日期和姓名，所以结构体应包含日期和姓名信息。从银行存款单我们可以知道，每一次存款和取款的数目必须记录，而且是分别记录，所以结构体考虑增加收入和支出两个元素，这样就形成了如下描述每一笔交易的结构体。

```

struct Date
{
    int year, month, day;
};

struct deal
{
    struct Date dt;           //每笔交易的日期
    char name[12];           //每笔交易的姓名
    double earning;           //每笔交易的收入额度
    double payout;           //每笔交易的支出额度
};

```

- 这样的设计能够满足题目要求吗？首先，每笔交易的记录可以存储起来，而且记录了交易的日期和金额，但是计算机如何知道是收入还是支出？此处必须设计一种输入方式，需要用户在输入每一笔交易金额的前面添加“+”或“-”，“+”代表收入，“-”代表支出，程序根据金额前面的符号判断存储在结构体

的哪个元素中。其次，题目要求计算每个人的每个财政年度的收入总额、支出总额、存款总额等，并能够打印出来，因此必须对所有输入的交易按姓名和年度进行统计，为了统计的实现，可首先对原交易记录按姓名和年度建立索引，然后再对每个人的收支情况按年度进行统计，余额为每人当年收入总额减去支出总额。

【参考答案】

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>

#define DEAL 50

struct Date
{
    int year, month, day;
};

struct deal
{
    struct Date dt;
    char name[12];
    double earning;
    double payout;
};
typedef struct deal FINANCE;

void Menu( );
void InputOneDeal (FINANCE *per);
void PrintOneYear (FINANCE *per);
void SortPerson (FINANCE *per, FINANCE **per_sort);

void main( )
{
    FINANCE person[DEAL] = {0};
    char key;
    int i = 0;

    while(1)
    {
        Menu( );
        key = getche( );
```

```
printf ("\n");
switch (key)
{
    case '1': InputOneDeal (person+i);
              i++;
              break;
    case '2': PrintOneYear (person);
              break;
    case '3': exit (0);
    default : break;
}
}
}

//菜单模块
void Menu( )
{
    printf ("1. Input one deal\n");
    printf ("2. Print one year Balance\n");
    printf ("3. Exit\n");
    printf ("Please select(1-3): ");
}

//每一笔交易输入模块
void InputOneDeal (FINANCE *per)
{
    char string[10];
    printf ("Date(YYYY-MM-DD): ");
    scanf ("%d-%d-%d", &per->dt.year, &per->dt.month, &per->dt.day);
    printf ("Name: ");
    scanf ("%11s", per->name);
    printf ("Deal(+/-): ");
    scanf ("%9s", string);
    if(string[0] == '-')
        per->payout = atof (string);
    else
        per->earning = atof (string);
}

//打印每个人的每个财政年度的收入总额、支出总额、存款总额模块
void PrintOneYear (FINANCE *per)
{
    FINANCE *per_sort[DEAL] = {NULL};
    int i, year, tag;
```

```

char name[12];
double earning, payout;

SortPerson (per, per_sort);
printf ("Name-----Year-----Earning-----Payout-----Balance\n");
i = 0;
while ( per_sort[i] != NULL )
{
    strcpy (name, per_sort[i]->name);
    printf ("%12s ", name);
    tag = 1;
    while ( per_sort[i] != NULL && strcmp(name, per_sort[i]->name) == 0 )
    {
        year = per_sort[i]->dt.year;
        earning = 0;
        payout = 0;
        while(per_sort[i]!=NULL && strcmp(name, per_sort[i]->name)==0 &&
            year == per_sort[i]->dt.year )
        {
            earning += per_sort[i]->earning;
            payout += per_sort[i]->payout;
            i++;
        }
        if (tag == 1)
        {
            printf("%-6d %11.2lf %11.2lf %11.2lf\n", year, earning, payout,
                earning+payout);
            tag = 0;
        }
        else
            printf ("%12s %-6d %11.2lf %11.2lf %11.2lf\n", " ", year, earning,
                payout, earning + payout);
    }
}

//按姓名和年份升序排列
void SortPerson (FINANCE *per, FINANCE **per_sort)
{
    int i, j, k;
    FINANCE *t;

    i = 0;
    while ((per+i)->dt.year != 0)

```

```
{
    per_sort[i] = per + i;
    i++;
}

for (i = 0; per_sort[i] != NULL; i++)
{
    k = i;
    for(j = i + 1; per_sort[j] != NULL; j++)
        if(strcmp(per_sort[j]->name, per_sort[k]->name) < 0 ||
            strcmp(per_sort[j]->name, per_sort[k]->name) == 0 &&
            per_sort[j]->dt.year < per_sort[k]->dt.year )
            k = j;
    if (k != i)
    {
        t = per_sort[i];
        per_sort[i] = per_sort[k];
        per_sort[k] = t;
    }
}
}
```

(4) 从键盘上输入 10 个学生的成绩, 编写一程序对学生的成绩按从高到低输出, 要求用链表来实现。

【设计思想】 为了建立链表, 首先为链表中的结点定义一个结构体类型 `NODE`, 它包含两个成员: 一个是成绩 `score`, 一个是指向 `NODE` 数据的指针 `next`。然后为链表建立头结点 `head`, 再根据每次输入的成绩建立新的结点, 将该结点插入到相应的位置, 使得每次插入后链表中的结点保持成绩从大到小的顺序不变, 最后从头结点开始遍历整个链表, 输出结果。当然, 最后不要忘了将链表销毁。

【参考答案】

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

struct Node_tag
{
    int score;
    struct Node_tag *next;
};
typedef struct Node_tag NODE;

void main( )
```

```
{
    int k, a;
    NODE *head, *p, *q;

    //创建头结点head
    head = (NODE *) malloc (sizeof (NODE));
    if (head == NULL)
    {
        printf ("no enough memory!\n");
        return;
    }
    head->next = NULL;

    for (k = 0; k < 10; k++)
    {
        //创建一个新结点p
        p = (NODE *) malloc (sizeof (NODE));
        if (head == NULL)
        {
            printf ("no enough memory!\n");
            return;
        }
        printf ("Input the %dth student's score: ", k+1);
        scanf ("%d", &a);
        p->score = a;

        //将结点p插入到链表中
        q = head;
        while (q->next != NULL)
        {
            //查找插入的位置
            if(q->next->score < p->score)
                break;
            q = q->next;
        }
        //结点p应该插入到q结点之后
        p->next = q->next;
        q->next = p;
    }

    //输出成绩
    printf ("After sorted: \n");
    p = head->next;
    while (p != NULL)
```



```

{
    printf ("%d ", p->score);
    p = p->next;
}
printf ("\n");

//链表销毁
p = head;
while (p->next != NULL)
{
    q = p->next;
    p->next = q->next;
    free (q);
}
free (head);
}

```

程序运行结果（假设输入的成绩为：70 75 80 85 86 90 88 96 95 98）：

```

After sorted:
98 96 95 90 88 86 85 80 75 70

```

(5) 口袋中有若干红、黄、蓝、白、黑 5 种颜色的球，每次从口袋中取出 3 个球，编程打印出得到 3 种不同颜色的球的所有可能取法。

【设计思想】 利用三重循环分别模拟取球过程，但每次取出的球需要与前面的球比较颜色，颜色相同的球要抛弃。

【参考答案】

```

#include <stdio.h>

void main( )
{
    char *ballcolor[ ] = {"RED", "YELLOW", "BLUE", "WHITE", "BALCK"};
    int i, j, k, m = 0;

    for(i=0; i < 5; i++)
        for(j = i+1; j < 5; j++)
            for (k = j+1; k < 5; k++)
            {
                m++;
                printf ("%d:  %s,  %s,  %s\n", m, ballcolor[i], ballcolor[j],
                    ballcolor[k]);
            }
}

```

(6) 25 个人围成一个圈，从第 1 个人开始顺序报号，凡报号为 3 和 3 的倍数者退出圈子，找出最后留在圈子中的人原来的序号。

【设计思想】 可以用链表方式实现。首先按照 1~25 号的顺序建立链表，1~25 号存放在链表的数据区，从链表头开始数，数到 3 或 3 的倍数的结点，删除该结点，一直到链表结束；重复上述过程一直到链表中只剩下两个结点，读出最后两个结点的数据区的数据就是最后的答案。

编程中遇到如下几个问题：

- 在第一次从链表头开始数每个结点时，可以很方便地确定每个结点的顺序，当到该链表的结尾时，结尾结点的顺序号应该是 25 号，下面应该数哪个结点？显然是头结点，也就是说，需要将该链表的首尾相接形成一个环形，如何形成一个环形？程序怎么做？
- 删除结点操作中包含三种不同的结点删除方式：中间结点、头结点、结尾结点。每一种删除方式都会不同，在访问和删除过程中必须分清要删除结点的类型。
- 必须考虑循环何时结束。

【参考答案】

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

struct Link
{
    int data;
    struct Link *next;
};

struct Link *head;           //链表头指针

struct Link *CreateNode (int nodeNumbers);
void DispLink (struct Link *head);
struct Link *DelNode (struct Link *p, struct Link *ptr);

void main( )
{
    int i = 0, nodenum = 25;
    struct Link *p, *pr;

    head = NULL;
    for (i = 0; i < 25; i++)
    {
        if (i == 0)
        {
```

```

        //如果是第一个结点,在head中保留该结点的首地址
        head = CreateNode (i);
        pr = head;
    }
    else
    {
        //如果不是第一个结点,将新建结点连到链表的结尾处
        pr->next = CreateNode (i);
        pr = pr->next;
    }
}
DispLink (head);

i = 1;
p = head;
for (; )
{
    if (i % 3 == 0)                //如果是3的倍数
    {
        p = DelNode (p, pr);
        i++;
        nodenum--;                //结点数减1
        if(nodenum < 3)            //如果结点数<3, 退出循环
            break;
    }
    else
    {
        pr = p;                    //链表走到下一个结点
        p=p->next;
        if(p==NULL)                //如果到尾结点, 下一个结点连接到头结点
            p = head;
        i++;                        //计数器加1
    }
}
printf ("The last man is :\n");
DispLink (head);
}

```

//函数功能: 建立一个新结点, 并为该结点赋值

//函数参数: 整型变量nodeNumbers, 表示建立的结点数

//返回 值: 返回该结点的指针

```

struct Link *CreateNode (int nodeNumbers)
{
    struct Link *p;

```

```

    p = (struct Link *) malloc (sizeof (struct Link));
    if (p == NULL)
    {
        printf ("No enough memory!");
        exit (0);
    }
    p->next = NULL;
    p->data = nodeNumbers + 1;
    return (p);
}

```

//函数功能: 显示所有已建立的结点的结点号和该结点中数据项内容

//函数参数: 结构体指针变量head, 表示指向链表的头指针

//返回 值: 无

```

void DispLink (struct Link *head)
{
    struct Link *p;
    int j = 1;

    p = head;
    do
    {
        printf ("%5d%10d\n", j, p->data);
        p = p->next;
        j++;
    } while (p != NULL);
}

```

//函数功能: 删除结点, 释放内存

//函数参数: 结构体指针变量p, 表示指向链表的当前结点的指针

// 结构体指针变量pr, 表示指向链表的当前结点的前一个结点的指针

//返回 值: 返回指向当前结点的指针

```

struct Link *DelNode (struct Link *p, struct Link *pr)
{
    if (p == head)
    {
        head = p->next;           //头结点的删除
        free (p);
        return (head);
    }
    if (p->next == NULL)         //尾结点的删除
    {
        pr->next = NULL;
        free (p);
    }
}

```

```
        return (head);
    }
    else
    {
        pr->next = p->next;
        free (p);
        return (pr->next);
    }
}
```

1.12 习题 12 参考答案和解释

1. 选择题

(1) fscanf 函数的正确调用形式是 ()。

- A. fscanf(fp, 格式字符串, 输出表列);
- B. fscanf(格式字符串, 输出表列, fp);
- C. fscanf(格式字符串, 文件指针, 输出表列);
- D. fscanf(文件指针, 格式字符串, 输入表列);

【答案】 D。

【解释】 fscanf 函数与 scanf 函数的主要区别就是在 fscanf 函数中第一个形参是文件指针, 其他参数与 scanf 完全相同。故正确答案是 D。

(2) 函数 rewind 的作用是 ()。

- A. 使位置指针重新返回文件的开头
- B. 将位置指针指向文件中所要求的特定位置
- C. 使位置指针指向文件的末尾
- D. 使位置指针自动移至下一个字符位置

【答案】 A。

【解释】 rewind 函数的作用就是将文件位置指针重新指向文件的开始位置。故正确答案是 A。

(3) fseek 函数的正确调用形式是 ()。

- A. fseek(文件类型指针,起始点,位移量)
- B. fseek(fp,位移量,起始点)
- C. fseek(位移量,起始点,fp)
- D. fseek(起始点,位移量,文件类型指针)

【答案】 B。

【解释】 fseek 函数的第一个参数是文件指针, 第二个参数应是位移量, 第三个参数是起始点。故正确答案是 B。

(4) 利用 fseek 函数可以实现的操作是 ()。

- A. 改变文件的位置指针
- B. 文件的顺序读写
- C. 文件的随机读写
- D. 以上答案均正确

【答案】 D。

【解释】 fseek 函数的功能就是能将文件的位置指针进行定位, 以实现文件的随机读写, 当然文件的顺序读写就更不成问题。故正确答案是 D。

(5) 函数调用语句: fseek(fp, -20L, 2); 的含义是 ()。

- A. 将文件位置指针移到距离文件头 20 个字节处
- B. 将文件位置指针从当前位置向后移动 20 个字节
- C. 将文件位置指针从文件末尾处退后 20 个字节
- D. 将文件位置指针移到离当前位置 20 个字节处

【答案】 C。

【解释】 fseek 函数的第三个参数有三个值可选: 0 表示文件的开始处, 1 表示当前位置, 2 表示文件尾; 位移量为负, 表示将文件位置指针往前移动。故正确答案是 C。

(6) 设有以下结构体类型:

```
struct st { char name[8]; int num; float s[4]; } student[50];
```

并且结构体数组 student 中的元素都已有值, 若要将这些元素写到硬盘文件 fp 中, 以下不正确的形式是 ()。

- A. fwrite (student, sizeof(struct st), 50, fp);
- B. fwrite (student, 50*sizeof(struct st), 1, fp);
- C. fwirte (student, 25*sizeof(struct st), 25, fp);
- D. for (i = 0; i < 50; i++)

```
    fwrite (student+i, sizeof(struct st), 1, fp);
```

【答案】 C。

【解释】 要实现将结构体数组 student 的所有元素写到文件 fp 中, 只需要保证写到文件中的总字节数为 50*sizeof(struct st)即可, 显然答案 A、B 都满足, 是正确的; 而在答案 C 中, 总共写到文件中的字节数为 25*25*sizeof(struct st), 是不正确的; 答案 D 是通过循环分别将 50 个元素逐一写到文件中, 所以也是正确的。故选择答案是 C。

(7) 若调用 fputc 函数输出字符成功, 则其返回值是 ()。

- A. EOF
- B. 1
- C. 0
- D. 输出的字符

【答案】 D。

【解释】 fputc 函数输出字符成功时, 其返回值就是输出的字符。故选择答案是 D。

(8) 系统的标准输入文件是指 ()。

- A. 键盘
- B. 显示器
- C. 软盘
- D. 硬盘

【答案】 A。

【解释】 C 语言中, 系统的标准输入文件就是指键盘。故选择答案是 A。

(9) 若有以下定义和说明:

```
struct std { char num[6]; char name[8]; float mark[4]; } a[30];
FILE *fp;
```

设文件中以二进制形式存有 10 个班的学生数据，且已正确打开，文件指针定位于文件开头。若要从文件中读出 30 个学生的数据放入 a 数组中，以下不能实现此功能的语句是（ ）。

- A. for (i = 0; i < 30; i++) fread (&a[i], sizeof(struct std), 1L, fp);
- B. for (i = 0; i < 30; i++) fread (a+i, sizeof(struct std), 1L, fp);
- C. fread (a, sizeof(struct std), 30L, fp);
- D. for (i = 0; i < 30; i++) fread (a[i], sizeof(struct std), 1L, fp);

【答案】 D。

【解释】 fread 函数的功能是从已打开的文件中读取若干字节的数据信息存放到以某变量为首地址的内存区域中，也就是要求 fread 函数的第一个参数必须是内存地址。故选择答案是 D。

(10) 若执行 fopen 函数时发生错误，则函数的返回值是（ ）。

- A. 地址值 B. 0 C. 1 D. EOF

【答案】 B。

【解释】 执行 fopen 函数时，如果文件打开成功，则返回该文件结构体的指针，如果打开失败（如：读打开时文件不存在，写打开时文件不能创建），则返回 NULL（即 0）。故选择答案是 B。

(11) 当顺利执行了文件关闭操作时，fclose 函数的返回值是（ ）。

- A. -1 B. TRUE C. 0 D. 1

【答案】 C。

【解释】 如果正常关闭了文件，则函数返回值为 0；否则，返回值为非 0。故选择答案是 C。

(12) 若以"a+"方式打开一个已存在的文件，则以下叙述正确的是（ ）。

- A. 文件打开时，原有文件内容不被删除，位置指针移到文件末尾，可进行添加和读操作
- B. 文件打开时，原有文件内容不被删除，位置指针移到文件开头，可进行重写和读操作
- C. 文件打开时，原有文件内容被删除，只可进行写操作
- D. 以上各种说法皆不正确

【答案】 A。

【解释】 若以"a+"方式打开一个已存在的文件，原文件如果不存在，则创建，如果存在则打开，并且将文件位置指针移到文件尾，可以在文件尾进行数据添加，也可以从文件开头处进行读数，但文件位置指针不可移到文件头，不能作重写操作。故选择答案是 A。

(13) 若要用 fopen 函数打开一个新的二进制文件，该文件既能读也能写，则文件方式字符串应是（ ）。

- A. "ab++" B. "wb+" C. "rb+" D. "ab"

【答案】 B。

【解释】 因为要打开一个新的二进制文件,则表示应具备文件创建的功能,所以文件方式字符串中应包含 **w** 和 **b**,又因为要对打开的文件既能读又能写,所以文件方式字符串中还应包含 **+**。故选择答案是 **B**。

(14) 以下可作为函数 **fopen** 中第一个参数的正确格式是 ()。

- A. `c:user\text.txt` B. `c:\user\text.txt`
- C. `"c:\user\text.txt"` D. `"c:\\user\\text.txt"`

【答案】 D。

【解释】 函数 **fopen** 的第一个参数是指要打开的文件名的字符串,所以答案 **A**、**B** 是错误的;文件路径分隔符必须使用转义字符,即双反斜杠,所以答案 **C** 也是错误的。故选择答案是 **D**。

(15) **fgetc** 函数的作用是从指定文件读入一个字符,该文件的打开方式必须是 ()。

- A. 只写 B. 追加 C. 读或读写 D. 答案 **B** 和 **C** 都正确

【答案】 D。

【解释】 **fgetc** 函数要求文件的打开方式必须是以读或读写的方式或者追加的方式,只写方式是不能读的。故选择答案是 **D**。

(16) 若 **fp** 是指向某文件的指针,且已读到此文件末尾,则库函数 **feof(fp)** 的返回值是 ()。

- A. **EOF** B. **0** C. 非零值 D. **NULL**

【答案】 C。

【解释】 在执行读文件操作时,如果遇到文件尾,则函数 **feof(fp)** 返回逻辑真 (非 0 值);否则,返回逻辑假 (0)。故选择答案是 **D**。

(17) 若要打开 **A** 盘上 **user** 子目录下名为 **abc.txt** 的文本文件进行读、写操作,下面符合此要求的函数调用是 ()。

- A. `fopen("A:\user\abc.txt", "r")` B. `fopen("A:\\user\\abc.txt", "r+")`
- C. `fopen("A:\user\abc.txt", "rb")` D. `fopen("A:\\user\\abc.txt", "w")`

【答案】 B。

【解释】 如果要对文件进行读、写操作,则文件打开方式字符串中必须有 **+**。答案 **A** 和 **C** 都只能对文件进行读操作,答案 **D** 则只能对文件进行写操作。故选择答案是 **B**。

(18) 下面的程序执行后,文件 **test.txt** 中的内容是 ()。

```
#include <stdio.h>
void fun (char *fname, char *st)
{
    FILE *myf; int i;
    myf = fopen (fname, "w" );
    for (i = 0; i < strlen(st); i++) fputc (st[i], myf);
    fclose (myf);
}
```



```
void main( )
{
    fun("test.txt", "new world");
    fun("test.txt", "hello,");
}
```

```

    fprintf (fp, "%d\n", i); fprintf (fp, "%d\n", j);
    fclose (fp);
    fp = fopen ("d1.dat", "r");
    fscanf (fp, "%d%d", &k, &n); printf ("%d, %d\n", k, n);
    fclose (fp);
}

```

A. 20, 30 B. 20, 50 C. 30, 50 D. 30, 20

【答案】 A。

【解释】 该程序的功能就是将两个整型数 i 和 j 的值通过 fprintf 函数将它们转换成字符串的形式（包含\n）写入到文件 d1.dat 中，然后通过 fscanf 函数从文件 d1.dat 中格式化读出到变量 k 和 n 中，所以 k 的值将是 20，n 的值将是 30。故选择答案是 A。

(23) 已知函数的调用形式：fread (buffer, size, count, fp); 其中 buffer 代表的是 ()。

- A. 一个整数，代表要读入的数据项总数
- B. 一个文件指针，指向要读的文件
- C. 一个指针，指向要读入数据的存放地址
- D. 一个存储区，存放要读的数据项

【答案】 C。

【解释】 fread 函数的功能就是从 fp 所代表的文件中读取 count*size 个字节的数据存放到指针 buffer 所指向的内存块中。故选择答案是 C。

2. 程序填空题

(1) 下面程序的功能是将一个磁盘中的二进制文件复制到另一个磁盘中，两个文件名随命令行一起输入，输入时原有文件的文件名在前，新复制文件的文件名在后，请在_____处填入适当内容。

```

#include <stdio.h>
void main (int argc, char *argv[ ])
{
    FILE *old, *new;
    char ch;
    if (argc != 3)
    {
        printf ("You forgot to enter a filename\n");
        exit (0);
    }
    if ( (old = fopen( __argv[1], "rb"__ )) == NULL )
    {
        printf ("cannot open infile\n");
        exit (0);
    }
    if ( (new = fopen( __argv[2], "wb"__ )) == NULL)

```

```

{
    printf ("cannot open outfile\n");
    exit (0);
}
while (!feof(old)) fputc ( __fgetc(old)__ , new);
fclose (old);
fclose (new);
}

```

【设计思想】 原有文件的文件名和新复制文件的文件名通过命令行参数带入，其中原有文件的文件名是由参数 `argv[1]` 来表示，新复制文件的文件名是由参数 `argv[2]` 来表示，原有文件应以读方式打开，而新复制文件则要以写方式打开。其复制的方法就是通过 `fgetc` 函数从原有文件中逐个字节地读出，然后通过 `fputc` 函数向新复制文件中逐个字节地写入，一直到原有文件结束为止。

(2) 以下程序将从终端上读入的 10 个整数以二进制方式写入一个名为“bi.dat”的新文件中。请在_____处填入适当内容。

```

#include <stdio.h>
void main( )
{
    int i, j;
    if ((fp = fopen( __"bi.dat"__ , "wb")) == NULL)
        exit (0);
    for (i = 0 ; i < 10; i++)
    {
        scanf ("%d", &j);
        fwrite ( __&j__ , sizeof(int), 1, __ fp __ );
    }
    fclose (fp);
}

```

【设计思想】 先以写的方式打开文件 `bi.dat`，然后通过循环将每次从键盘上输入的整数写入到文件 `bi.dat` 中。`fwrite` 函数的第一个参数是要写入的数据指针，第四个参数为文件指针。

3. 编程题

(1) 编写一程序，其功能是显示指定的文本文件，在显示文件内容时同时加上行号。

【设计思想】 命令行的第二个参数 `argv[1]` 为待显示的文件名，显示前以读方式打开，然后通过 `fgets` 函数从文件中读出一行信息到临时字符串 `string` 中，再利用 `printf` 函数显示该行内容（通过变量 `i` 来显示行号），直到文件结束，即读出的字符串指针为 `NULL` 为止。

【参考答案】

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

#define MAX 256

void main (int argc, char *argv[ ])
{
    FILE *fp;                                //fp: 待显示文件指针
    char string[MAX];
    int i;

    if(argc != 2)                            //参数个数不对
    {
        printf("the number of arguments not correct\n");
        printf("\n Usage: 可执行文件名 dispfilename");
        exit(0);
    }

    if ((fp = fopen(argv[1], "r")) == NULL)    //打开显示文件失败
    {
        printf("can not open file\n");
        exit(0);
    }

    //显示文件内容
    i = 1;
    while (fgets(string, MAX, fp) != NULL)
        printf ("%d %s", i++, string);

    fclose(fp);                               //关闭文件
}
```

(2) 编写一程序, 其功能是将两个文件的内容合并到一个文件中, 并显示合并后的文件内容。三个文件名随命令行一起输入, 输入时原有两文件的文件名在前, 合并文件的文件名在后。

【设计思想】 命令行的第二个参数 `argv[1]` 为源文件名 1, 第三个参数 `argv[2]` 为源文件名 2, 第四个参数 `argv[3]` 为目标文件名, 即合并文件的文件名。前两个文件以读方式打开, 合并文件则以写方式打开, 然后将源文件 1 中的信息复制到目标文件中, 其具体方法是通过 `fgets` 函数从源文件 1 中逐一读出信息到临时字符串 `string` 中, 再利用 `fputs` 函数将 `string` 写入到目标文件中, 直到源文 1 结束, 即读出的字符串指针为 `NULL` 为止。按同样的方法将源文件 2 中的信息复制到目标文件中, 这样就完成了两个文件内容的

合并。

【参考答案】

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

void main (int argc, char *argv[ ])
{
    FILE *input1, *input2, *output;
                                //input1,input2: 源文件指针, output: 目标文件指针
    char string[81];

    if(argc != 4)                //参数个数不对
    {
        printf ("the number of arguments not correct\n");
        printf ("\n Usage:  可执行文件名  source-file1 source-file2 dest-file");
        exit (0);
    }

    if ((input1 = fopen(argv[1],"r")) == NULL)        //打开源文件1失败
    {
        printf ("can not open source file1\n");
        exit (0);
    }

    if ((input2 = fopen(argv[2],"r")) == NULL)        //打开源文件2失败
    {
        printf ("can not open source file2\n");
        exit (0);
    }

    if ((output = fopen(argv[3],"w")) == NULL)        //创建目标文件失败
    {
        printf ("can not create destination file\n");
        exit (0);
    }

    //复制源文件1到目标文件中
    while (fgets (string, 81, input1) != NULL)
        fputs (string, output);
    //复制源文件2到目标文件中
```

```

while (fgets (string, 81, input2) != NULL)
    fputs (string, output);

fclose (input1);          //关闭源文件1
fclose (input2);          //关闭源文件2
fclose (output);          //关闭目标文件
}

```

(3) 编写文本文件显示程序，在命令行中指定文本文件显示的范围。如下所示：

```
type filename m n✓
```

其中 `type` 为执行文件名，`filename` 是要显示的文本文件名，`m` 和 `n` 指定了显示的范围，即显示从 `m` 行到 `n` 行的内容，当 `m` 和 `n` 不指定时，则显示文件的全部内容。

【设计思想】 命令行参数的个数应当为 2 或者 4，如果为 2，则表示显示全部内容，其显示的方法可采用第(1)或(2)题的方法来完成，如果为 4，则表示显示从第 `m` 行到第 `n` 行的内容，其方法可以先从文件中读取 `m-1` 行不显示，然后再从第 `m` 行开始读并显示，直到第 `n` 行读出并显示或遇到文件尾为止。

【参考答案】

```

#include <stdio.h>
#include <stdlib.h>

#define MAX 250

void main (int argc, char *argv[ ])
{
    FILE *fp;
    char string[MAX];
    int i, m, n, tag = 0;

    if (argc != 2 && argc != 4)                //参数个数不对
    {
        printf("the number of arguments not correct\n");
        printf("\n Usage: 可执行文件名 filename m n\n");
        printf("or\n Usage: 可执行文件名 filename\n");
        exit(0);
    }

    if (argc == 4)
    {
        m = atoi (argv[2]);

```

```
n = atoi (argv[3]);
if (m > n)
{
    printf ("arguments error!\n");
    exit (0);
}

if ((fp = fopen(argv[1], "r")) == NULL)    //打开文件失败
{
    printf("can not open source file\n");
    exit(0);
}

if (argc == 4)                            //有起止行号, 显示从第m行到第n行的内容
{
    for (i = 1; i < m; i++)                //读取m-1行不显示
        if (fgets (string, MAX, fp) == NULL)
        {
            tag = 1;
            break;
        }
    if (tag == 0)                          //未遇到文件尾, 显示从第m行到第n行的内容
    {
        for(; i <= n && fgets(string, MAX, fp) != NULL; i++)
            printf ("%s", string);
    }
}
else                                       //无起止行号, 显示全部内容
{
    while (fgets(string, MAX, fp) != NULL)
        printf ("%s", string);
}

fclose(fp);                              //关闭文件
}
```

(4) 有一张成绩表, 表名为 `score_tab.txt`, 文件类型为文本文件, 其结构为: 学号 `no` (10 位数字)、姓名 `name` (20 个字符以内, 中间没有空格)、分数 `score` (3 位数以内)、所用时间 `time` (3 位数以内), 彼此之间以一个空格分隔。要求分数按降序排列, 分数相同则按所用时间升序排列, 并输出名次。如果分数、所用时间均相同, 则以相同名次输出。

样例：成绩表（文件为 score_tab.txt）				输出结果：				
学 号	姓 名	分数	时间	名次	学 号	姓 名	分数	时间
1234567890	Tom	90	10	1	3243424324	WangJingHua	123	15
2342534524	John	100	10	2	3612536273	ZhangSen	123	59
3243424324	WangJingHua	123	15	3	2342534524	John	100	10
5656565656	ZhangSan	100	26	3	2757577577	TanSheng	100	10
3612536273	ZhangSen	123	59	4	5656565656	ZhangSan	100	26
7352735255	GuoJia	30	60	5	1234567890	Tom	90	10
2757577577	TanSheng	100	10	6	9734289772	LiuLiang	67	26
9734289772	LiuLiang	67	26	7	7352735255	GuoJia	30	60

【设计思想】 首先定义一个有关成绩表的结构体数组 score，用来存放从文件 score_tab.txt 读出的成绩信息，然后对结构体数组 score 按照成绩降序排列，成绩相同按时间升序排列，再对排序后的结构体数组 score 进行排名，最后输出排名的结果。

【参考答案】

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

#define NUM 100

struct Score_Tab
{
    int rank;
    char no[11];
    char name[21];
    int score;
    int time;
};

void main( )
{
    struct Score_Tab score[100], temp;
    FILE *fp;
    int i, j, k, n, s, t;

    if ((fp = fopen("score_tab.txt","r")) == NULL) //打开文件失败
    {
        printf("can not open source file: score_tab.txt\n");
        exit(0);
    }

    //从文件中读取数据到结构体数组score中
```



```
n = 0;
while(fscanf(fp, "%s%d%d", score[n].no, score[n].name,
            &score[n].score, &score[n].time) != EOF)
    n++;

//对结构体数组score按分数降序排列, 分数相同则按所用时间升序排列
for (i = 0; i < n-1; i++)
{
    k = i;
    for(j = i+1; j < n; j++)
        if(score[j].score > score[k].score ||
            score[j].score==score[k].score && score[j].time < score[k].
                time)
            k = j;
    if (k != i)
    {
        temp = score[i];
        score[i] = score[k];
        score[k] = temp;
    }
}

//排名次
j = 0;
s = 0;
t = 0;
for (i = 0; i < n; i++)
{
    if (score[i].score != s || score[i].time != t)
    {
        j++;
        score[i].rank = j;
        s = score[i].score;
        t = score[i].time;
    }
    score[i].rank = j;
}

//显示排名结果
for(i = 0; i < n; i++)
    printf("%4d %-10s %-20s %3d %3d\n", score[i].rank, score[i].no, score[i].
name, score[i].score, score[i].time);

fclose(fp); //关闭文件
}
```

第2章

思考题解答

◇ 学习要点

- 完成主教材中的全部思考题。
- 通过思考题练习加深对主教材中有关知识的理解。
- 加强对 C 语言基本语法规则的掌握。

2.1 第 3 章思考题参考答案和解释

思考题 3-1: 在 C 语言中, 变量名 `total` 与变量名 `TOTAL`、`ToTaL`、`tOtAl` 等是同一个变量吗?

【参考答案】 不是同一个变量。因为 C 语言对英文字母的大小敏感, 即同一字母的大小写, 被认为是两个不同的字符。

思考题 3-2: 判断下列标识符号合法性

`sum Sum M.D.John day Date 3days student_name #33 lotus_1_2_3 char a>b _above $123`

【参考答案】 `sum` (✓) `Sum` (✓) `M.D.John` (×, 含有.) `day` (✓) `Date` (✓) `3days` (×, 首字符不能是数字) `student_name` (✓) `#33` (×, 含有#) `lotus_1_2_3` (✓) `char` (×, `char` 是关键字) `a>b` (×, 含有>) `_above` (✓) `$123` (×, \$不能打头)

思考题 3-3: 下列那些整型常量是合法的? `012`, `oX7A`, `00`, `078`, `0x5Ac`, `-0xFFFF`, `0034`, `7B`。

【参考答案】 `012` (✓), `oX7A` (×, 首字符不能是字母 o), `00` (✓), `078` (×, 八进制数中不能有数字 8), `0x5Ac` (✓), `-0xFFFF` (✓), `0034` (✓), `7B` (×, 十进制数中不能有字母 B)。

思考题 3-4: 请问 `-0` 和 `0` 在内存中如何表示? 如果一个整数用两个字节的存储单元来存放, 问 `-1` 和 `65535` 在内存中表示的是同一个数吗? `-65535` 在内存中又如何表示呢?

【参考答案】 `-0` 和 `0` 在内存中都是 0; `-1` 和 `65535` 在内存中表示的是同一个数; `-65535` 在内存中的表示为十进制数 1。因为数据在计算机中是以其补码的形式存放的, `-0`

和 0 的补码都是 0; $(-1)_{\text{补}} = 1111\ 1111\ 1111\ 1111$, 与 $(65536)_{\text{补}} = 1111\ 1111\ 1111\ 1111$ 一致, 故 -1 和 65535 表示的是同一个数; $(-65535)_{\text{补}} = 0000\ 0000\ 0000\ 0001$, 即表示十进制数 1。

思考题 3-5: 请问字符常量'A'与字符串常量"A"在内存中的表示有何不同? 只包含一个空格字符的字符串常量" "与不包含任何字符的字符串常量""又有何不同?

【参考答案】 字符常量'A'占一个字节, 其值为 A 字符的 ASCII 码 65, 而"A"表示的是包含一个字符 A 的字符串, 占两个字节, 一个是 A 字符, 另一个是字符串结尾符'\0'。只包含一个空格字符的字符串常量" "占有两个字节, 一个是空格字符, 另一个是'\0', 字符串长度为 1, 而不包含任何字符的字符串常量""只占一个字节, 就是'\0', 字符串长度为 0。

思考题 3-6: 如果将【例 3-5】中的第 21 行中的 %ld 变为 %u, 问在 BC 下和在 VC 下 L 的输出又是多少呢?

【参考答案】 在 BC 下 L 的输出是 65535, 而在 VC 下 L 的输出是 4294967295。因为在 BC 下 -1 用两个字节来表示, 其无符号数为 $(1111\ 1111\ 1111\ 1111)_2$, 即 $(65535)_{10}$; 而在 VC 下 -1 用 4 个字节来表示, 其无符号数为 $(1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111)_2$, 即 $(4294967295)_{10}$ 。

思考题 3-7: 如果将 ch 的值赋为 -4, 问 b 的值又是多少呢?

【参考答案】 b 的值为 0x00fc。因为 $(-4)_{\text{补}} = (1111\ 1100)_2 = (\text{fc})_{16}$, ch 是无符号字符型变量, 将其值赋给有符号整型变量 b, 则只需将 ch 赋给 b 的低位部分, 而高位部分补 0, 所以 b 的值为 0x00fc。

思考题 3-8: 执行语句 `a = (int)(2.5 * 4) + 5` 后, 问 a 的值又是多少呢?

【参考答案】 a 的值是 15。因为括号中的乘法先计算得 10.000000, 然后再通过强制类型转换成整型数 10, 最后加上 5, 这样就得到了 a 的值为 15。

思考题 3-9: `p = ++i(++i)` 能否写成 `p = ++i+++i`? 能否写成 `p = ++i+++i? p = ++i-++i` 和 `p = ++i+-i` 合法吗?

【参考答案】 `p = ++i(++i)` 不能写成 `p = ++i+++i`, 因为 `p = ++i+++i` 是非法的; `p = ++i(++i)` 能写成 `p = ++i+++i`; `p = ++i-++i` 和 `p = ++i+-i` 都是合法的。

2.2 第4章思考题参考答案和解释

思考题 4-1: 假设有一整数 888, 如何使用三种方法使其输出结果为 000888?

【参考答案】 方法一: `printf ("%06d", 888);` 方法二: `printf ("%6.6d", 888);` 方法三: `printf ("%6d", 888);`

思考题 4-2: 假设有整型变量 a 和浮点型变量 f, 用户的输入为 12345678765.43, 现要将输入中的 123 赋给变量 a, 8765.43 赋给 f, 问 scanf 函数调用中的格式控制字符串如何组织?

【参考答案】 `scanf ("%3d%*3d%f ", &a, &f);` 利用 %*3d 可舍掉 456。

2.3 第 6 章思考题参考答案和解释

思考题 6-1: 请问上例中 for 语句后面的分号如果省掉, 那么 for 循环的循环体还是空语句吗? 如果不是, 那循环体语句是什么? 程序的输出将是什么?

【参考答案】 循环体语句不再是空语句, 而是 `printf("%d",n);` 程序的输出将是:

012……

思考题 6-2: 将【例 6-5】程序中内层循环的控制变量 `j` 改成 `i` 后, 运行结果将是什么?

【参考答案】 输出的结果将是: `i = 1: i = 1 i = 2 i = 3 i = 4`

2.4 第 7 章思考题参考答案和解释

思考题 7-1: 试比较以下几种数组初始化之间的差异:

(1) `int a[5] = {1,2,3,4,5};`

(2) `int a[] = {1,2,3,4,5};`

(3) `int a[5] = {1,2,3};`

(4) `int a[] = {1,2,3};`

【参考答案】 (1) `int a[5] = {1,2,3,4,5};`

定义了含有 5 个元素的整型数组 `a`, 并将 `a[0]~a[4]` 分别赋初值为 1、2、3、4、5。

(2) `int a[] = {1,2,3,4,5};`

与 (1) 效果相同。

(3) `int a[5] = {1,2,3};`

定义了含有 5 个元素的整型数组 `a`, 并将 `a[0]~a[2]` 分别赋初值为 1、2、3, `a[3]` 和 `a[4]` 的值默认为 0。

(4) `int a[] = {1,2,3};`

定义了含有 3 个元素的整型数组 `a`, 并将 `a[0]~a[2]` 分别赋初值为 1、2、3, 其等价于 `int a[3] = {1,2,3};`

思考题 7-2: 请问下面的程序是否正确? `int a[3]; scanf ("%d", a);` 如果正确, 输入的值将赋给数组 `a` 的哪些元素?

【参考答案】 正确。键盘输入的值将赋值给 `a[0]`。因为 `a` 是数组在内存中的首地址, 也是数组第一个元素 `a[0]` 的内存地址, 所以输入的值给 `a[0]`。

思考题 7-3: 请问下面的程序执行后数组 `a` 的各元素的值是多少? `short a[10]; memset(a, 2, 10*sizeof(short));`

【参考答案】 数组 `a` 各数据单元的值均为 514。因为是将 2 赋值给数组 `a` 的所占存储块的每个字节单元中, 而数组 `a` 的每个元素占 2 字节单元, 即每个元素的值为 $(00000010\ 00000010)_2 = (514)_{10}$ 。

思考题 7-4: 请问下面定义的二维数组 a 包含有多少个元素? 各元素的值是多少?

```
int a[ ][3] = { 1, 2 };
```

【参考答案】 数组 a 包含 3 个元素。各元素的值为: $a[0][0] = 1$, $a[0][1] = 2$, $a[0][2] = 0$ 。

思考题 7-5: 请问字符串 "0123\0a456789" 和 "0123\04a56789" 的长度各是多少?

【参考答案】 字符串 "0123\0a456789" 的长度为 4, 因为第 5 个字符为 '\0', 即为字符串结尾符; "0123\04a56789" 的长度为 11, 因为这里应将 '\04' 看成是 ASCII 码值为 4 的一个字符。

2.5 第 8 章思考题参考答案和解释

思考题 8-1: 如果将【例 8-3】中的第 8 行改为 `compare (++i, i++)`; 那么输出结果是什么?

【参考答案】

在 TC 或 BC 下:

```
a = 4  b = 2
a > b
i = 4
```

在 VC 下:

```
a = 3  b = 2
a > b
i = 4
```

2.6 第 9 章思考题参考答案和解释

思考题 9-1: 如果要将【例 9-1】中的变量 a 的高字节的值改变为 0XFF 该如何操作?

【参考答案】 可使用语句 `*(pc+1) = 0XFF`; 即可。

思考题 9-2: 方法三中的 `*p++ = 'A' + k`; 语句能否改为 `*str++ = 'A' + k`? 能否改为 `(*p)++ = 'A' + k`? 能否改为 `*++p = 'A' + k`?

【参考答案】 不可改为 `*str++ = 'A' + k`; 因为 str 是地址常量不可进行 ++ 运算; 不可改为 `(*p)++ = 'A' + k`; 因为执行的结果是先把 'A'+k 赋给 *p, 再把 *p 的值增 1, 这与程序的功能不一致; 不可改为 `*++p = 'A' + k`; 因为执行的结果是先把 p 增 1, p 指向了下一单元, 再把 'A'+k 赋给 *p, 这与程序的功能也是不一致的。

思考题 9-3: 假设定义了一指针变量: `int *q`; 现将【例 9-5】中的第 13 行~第 15 行换成: `q = p[i]; p[i] = p[j]; p[j] = q`; 是否可行? 结果是否相同? 试比较两种方法对变量和指针数组有何变化?

【参考答案】 可行，结果相同。对于【例 9-5】执行以后，数据按从小到大的顺序分别存放在变量 a、b、c、d、e 的存储单元中，变量原来的值发生了调整，而指针数组中每个指针元素所指向的变量存储单元并未发生改变，p[0]...p[4]还是分别指向 a、b、c、d、e。当用 q = p[i]; p[i] = p[j]; p[j] = q; 来代替 13 行~15 行后，变量 a、b、c、d、e 的值没有发生任何改变，但指针数组中的指针元素所指向的内存单元将发生了改变，p[0]指向了其值最小的变量内存单元，p[4] 指向了其值最大的变量内存单元。

2.7 第 11 章思考题参考答案和解释

思考题 11-1: 问用 struct Student_Info 所定义的变量所占用的内存大小是多少字节？

【参考答案】 在 BC 或 TC 下占 38 个字节，在 VC 下占 44 字节。

思考题 11-2: 如果将【例 11-11】中的第 6 行改为 unsigned int b : 5;，问程序运行的结果在 BC 和 VC 下有何不同？写出运行结果。

【参考答案】

在 BC 或 TC 下：m.y.a = 1 m.y.b = 24 m.y.c = 1023 m.x = FFC1

在 VC 下：m.y.a = 1 m.y.b = 0 m.y.c = 0 m.x = FFF1

2.8 第 12 章思考题参考答案和解释

思考题 12-1: 【例 12-2】中的第 28、29 行如果不用 feof 函数来判断源文件结束，是否有其他的方法来代替？如果有，如何改写？

【参考答案】 可用其他方法来代替。其方法是用下列语句代替第 28、29 行语句：

```
char ch;
for (; (ch = fgetc (input)) != EOF; )
    fputc (ch, output);
```

思考题 12-2: 【例 12-3】中的第 23 行如果改为 fgets (str, 3, fp2);，则程序的输出是否发生变化？写出程序的运行结果。

【参考答案】 程序的输出将发生变化，其输出将变为：

```
1234567
89
```

思考题 12-3: 【例 12-7】中的第 27 行如果改为 fscanf (fp, "%d%f ", &i, &f);，则程序的输出是否发生变化？写出程序的运行结果。

【参考答案】 程序的输出将发生变化，其输出将变为：

```
i= 3, f= 0.00
```

因为文件中存放的数据形式是： 3, 9.80，其中有','号，而格式控制符中没有','号，故只读取了 i 的值 3，f 的值没有读取，为 0。

C 语言上机开发环境介绍

◇ 学习要点

- 掌握 Borland C++ 3.1 IDE 的使用方法;
- 掌握 Borland C++ 3.1 环境下程序调试方法;
- 掌握 Visual C++ 6.0 环境下 C 程序调试、运行方法。

3.1 Borland C++ 3.1 开发环境

Borland C++ 3.1 是 Borland 公司开发的 C/C++ 语言集成开发环境 (IDE)，它集程序编辑、编译、链接、调试与运行于一体，是当今许多 C 语言爱好者学习 C 语言上机编程的开发工具。本节主要介绍 Borland C++ 3.1 开发环境下的程序编辑、编译、链接及调试的有关方法，以及工程文件的使用方法和带参数的 main 函数的运行方法。

3.1.1 Borland C++ 的启动及准备工作

在启动 BC 的 IDE 环境之前，先要进行一些必要的准备工作：

(1) 弄清楚 Borland C++ 3.1 被安装在硬盘的哪个目录（文件夹）下，假设为 C:\Borlandc。

(2) 在存放数据的硬盘中建立自己的目录，以便将所编写的程序存放在此目录中，假设为 C:\Cppprg。

按照下面的方法之一启动 BC 的 IDE 环境：

- 如果桌面上有“Bolandc C++ for DOS”图标存在，使用鼠标左键进行双击，即可打开 BC 环境。
- 通过鼠标双击“我的电脑”→“本地磁盘(C:)”→BORLANDC→BIN，再找到执行文件 BC.exe，用鼠标双击，即可打开 BC 环境。
- 用鼠标单击桌面左下角的“开始”菜单，然后将鼠标指针移到“运行(R)”处，单击会出现一个“运行”对话框，再在弹出的对话框中输入 c:\borlandc\bin\bc.exe，如图 3-1 所示，然后单击“确认”按钮即可打开 BC 环境。



图 3-1 “运行”弹出的对话框

- 进入 MS-DOS 环境（其方法：在图 3-1 中的对话框中输入 `cmd`，然后单击“确定”按钮；或单击“开始”→“程序”→“附件”→“命令提示符”），然后输入 `c:\borlandc\bin\bc.exe`↵，即可打开 BC 环境（↵表示回车）。进入 BC 环境以后，系统将出现如图 3-2 所示的 BC 编程环境。

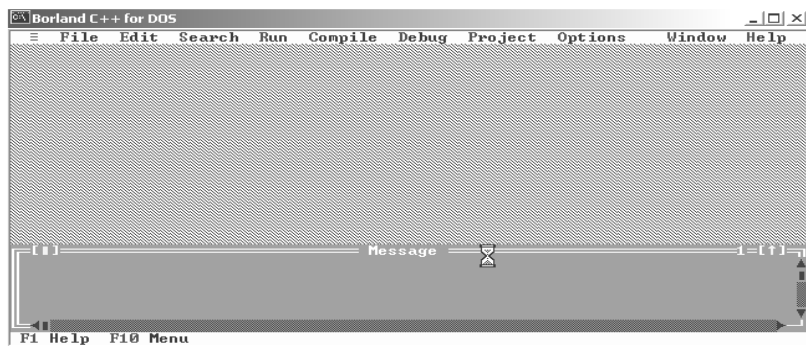


图 3-2 Borland C++ 3.1 环境窗口

设定工作目录：

进入 BC 的 IDE 环境以后，需要进行工作目录的设定。其具体方法就是按 ALT+O 键激活 Options 菜单，选择 Directories...，如图 3-3 所示，打开 Directories 对话框，如图 3-4 所示。



图 3-3 Options 菜单图



图 3-4 工作目录对话框

Include Directories: 用于设定 BC 中包含文件（如 `stdio.h` 文件等）所在的目录，它位于 Borlandc 下的 INCLUDE 子目录下，可根据需要进行设置。

Library Directores: 用于设定 BC 中库文件所在的目录，它位于 Borlandc 下的 LIB 子目录下，可根据需要进行设置。

Output Directores: 用于设定源程序经编译和链接以后生成的文件（如目标文件.obj、执行文件.exe）所存放的目录。如图 3-4 所示，设定输出文件的目录为 C:\CPPPRG，也就是说在 IDE 环境下编译所生成的执行文件将存放在 C:\CPPPRG 目录下。

Source Directores: 用于设定源程序文件目录。

以上目录设定好以后，单击 OK 按钮保存即可。一旦设定完毕，如果以后不需要更改，就不必每次都进行设置。

3.1.2 Borland C++编辑器的使用

进入 BC 的 IDE 环境以后，如果是第一次启动 IDE，系统会打开一个新的窗口，等待用户输入源程序。这时要按 ALT+F3 键关闭该窗口，我们要重新建立一个完整的程序输入流程。

按 ALT+F 键激活 File（文件）菜单，选择 Open...命令，如图 3-5 所示或直接按 F3 键，打开 Open a File（打开文件）对话框，如图 3-6 所示。在打开文件对话框的 Name 文本框中输入 C:\cppprg\test.cpp 后，按回车键返回编辑状态，如图 3-7 所示。



图 3-5 File 菜单图

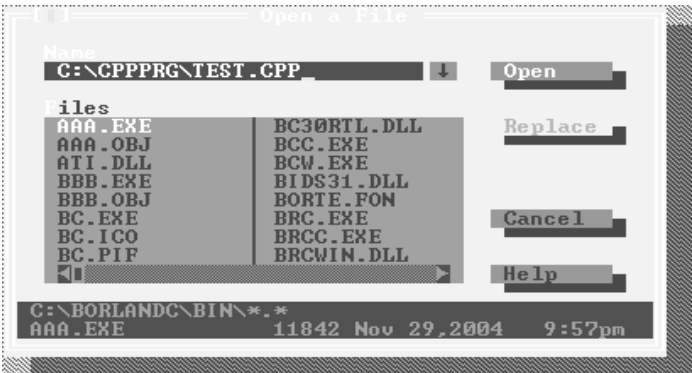


图 3-6 打开文件对话框

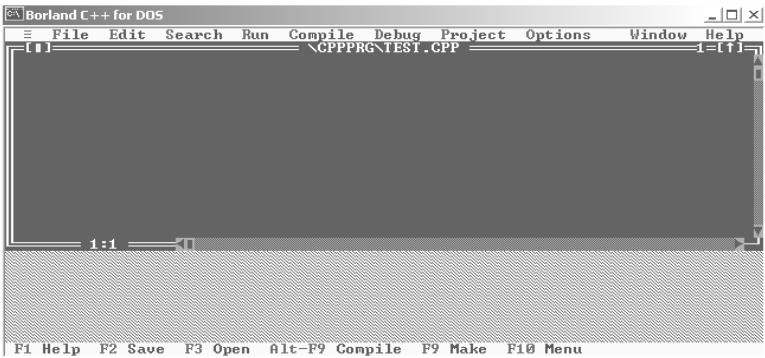


图 3-7 空的 test.cpp

在图 3-7 中, TEST.CPP 是窗口的标题 (Borland C++ 用打开的文件名作为窗口的标题), 具有蓝色背景的区域就是窗口的编辑区域, 可以看到位于窗口左上角的光标正在闪烁。与窗口标题 TEST.CPP 位于同一行的有一个数字 1, 它表示窗口的编号, 因为 Borland C++ IDE 可同时打开多个窗口。窗口的左下角的 1:1 表示光标的位置 (第几行、第几列)。这时可以在窗口中输入任何文本, 也可以移动光标。具体键盘命令见表 3-1。可以随便输入一些文本, 再逐个试一下表 3-1 列出的命令。

表 3-1 键盘命令及其功能对照表

键 盘 命 令		功 能
移动光标	左方向键	向左移动一个字符
	右方向键	向右移动一个字符
	Ctrl+左方向键	向左移动一个单词
	Ctrl+右方向键	向右移动一个单词
	上方向键	向上移动一行
	下方向键	向下移动一行
	Ctrl+W	光标位置不变, 窗口文本向下移动一行 (窗口向上滚动一行)
	Ctrl+Z	光标位置不变, 窗口文本向上移动一行 (窗口向下滚动一行)
	PgUp	向上翻页
	PgDn	向下翻页
	Home	将光标移动到光标所在行的行首
	End	将光标移动到光标所在行的行尾
	Ctrl+Q E	将光标移动到窗口的顶部 (注意: 不是文件的顶部)
编辑文本	Del	删除所在处的字符, 如果光标位于行尾, 下一行文本会续接到光标所在行上
	Backspace	删除光标左边的字符, 如果光标位于行首, 光标所在行会续接到上一行
	Ctrl+Y	删除光标所在行的全部文本
	Ctrl+Q Y	删除从光标处到行尾的全部字符
	Ctrl+T	删除光标所在处的一个单词, 但光标左边的字符不删除
	Ctrl+N	在光标处插入新行
	Alt+Backspace	恢复前一次的操作
选择文本块	Ctrl+K B	将光标的位置设置为被选择文本的首位置, 如果被选择文本的首位置位于尾位置的前面, 首位置和尾位置之间的文本将被高亮度显示
	Ctrl+K K	将光标的位置设置为被选择文本的尾位置, 如果被选择文本的首位置位于尾位置的前面, 首位置和尾位置之间的文本将被高亮度显示
	Ctrl+K H	隐藏或高亮度显示被选择的文本
	Ctrl+K C	将高亮度显示的被选择文本复制到光标处
	Ctrl+K V	将高亮度显示的被选择文本粘贴到光标处
	Shift+左方向键	将被选择文本块向左扩展或压缩一个字符
	Shift+右方向键	将被选择文本块向右扩展或压缩一个字符
	Shift+End	将被选择文本块扩展或压缩到行尾

续表

键 盘 命 令		功 能
选择文本块	Shift+Home	将被选择文本块扩展或压缩到行首
	Shift+下方向键	将被选择文本块向下扩展或压缩一行
	Shift+上方向键	将被选择文本块向上扩展或压缩一行
	Shift+PaDn	将被选择文本块向下扩展或压缩一页
	Shift+PaUp	将被选择文本块向上扩展或压缩一页
	Shift+Ctrl+左方向键	将被选择文本块向左扩展或压缩一个单词
	Shift+Ctrl+右方向键	将被选择文本块向右扩展或压缩一个单词
	Shift+Ctrl+End	将被选择文本块扩展或压缩至文件尾
	Shift+Ctrl+Home	将被选择文本块扩展或压缩至文件首
	Ctrl+Q B	将光标移动到被选择文本的首位置
	Ctrl+Q K	将光标移动到被选择文本的尾位置
	Ctrl+K P	打印被选择的文本
	Ctrl+K Y	删除被选择的文本
	Ctrl+Ins	将被选择的文本复制到粘贴板上
	Shift+Del	将被选择的文本剪切到粘贴板上
	Ctrl+Del	删除被选择的文本
	Ctrl+K I	将被选择的文本向右缩进一个字符
	Shift+Ins	将粘贴板上的文本复制到光标处
	Ctrl+K R	将一个文件的内容读入并插入到光标处
	Ctrl+K U	将被选择的文本向左移动一个字符
	Ctrl+K W	将被选择的文本写到一个文件中
常规命令	F3	打开一个文件
	F2	保存文件
	F5	将窗口最大化或恢复原来大小
	Ctrl+Q F	打开查找字符串的对话框
	Ctrl+L	继续上一次的字符串查找或替换
	Ctrl+Q A	打开查找替换字符串的对话框
	F1	打开求助窗口
	Shift+F1	打开求助索引窗口，这时输入要求助的函数名，就可以找到相关的帮助文档
	Ctrl+F1	打开光标所在处单词的帮助文档
	Ins	插入模式的开/关
	Alt+X	退出 Borland C 的集成开发环境

注意：

Ctrl+K C 是指按住 Ctrl 键不放，先后按一下 K 键和 C 键。

Shift+F1 是指按住 Shift 后再按一下 F1 键。

其他以此类推。

下面将正式输入一个程序，这个程序的内容如图 3-8 所示。这时窗口的左下角出现了一个*，它表明 TEST.C 文件被修改但尚未存盘。这时如果按 F2 键保存文件，此符号就会消失。

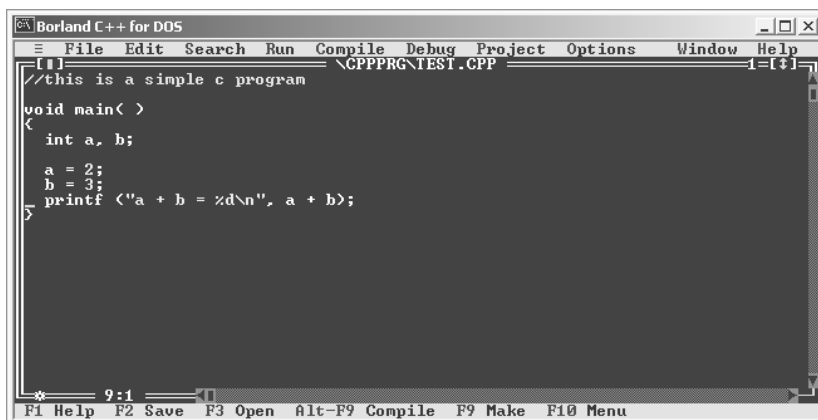


图 3-8 编辑 test.cpp

当用户打开多个窗口时，这些窗口中只有一个是活动的。这个活动的窗口显示在最前面，并且窗口的边框是白色双线，窗口的标题也是白色的。而其他窗口是不活动的，它们的窗口边框是灰色单线，窗口的标题也是灰色的。在如图 3-9 所示的编辑窗口中，新打开的窗口 **TEST2.CPP** 是活动的，用户可以编辑 **TEST2.CPP** 文件，但不能编辑 **TEST.CPP** 文件，因为窗口 **TEST.CPP** 是不活动的。用户可以激活某个窗口来编辑相应的程序。当按住 **ALT** 键不放，再输入窗口的编号，就可以将该编号代表的窗口激活。当然，也可以按 **F6** 键在各个窗口间切换。



图 3-9 两个编辑窗口

这时，用户可以开始使用 Borland C++ IDE 了。

3.1.3 程序的编译、链接、运行

如果读者已经熟悉了 Borland C++ 3.1 集成环境中的编辑器的使用。那么从现在开始，读者需要了解编程的完整流程以及程序编译、链接和运行的方法。基本操作步骤如下：

(1) 启动 BC 后，按 **F3** 键激活打开文件对话框，在 Name 文本框中输入一个你想要的文件名。这个文件名不能超过 8 个字符，并且后缀是 .cpp(假设文件名输入为: test.cpp)。

按回车键开始编辑程序。

(2) 输入程序。建议事先在纸上写好程序，再输入到计算机中。程序可以是完全新编的，也可以直接打开原先已编写好的存放在磁盘上的源程序，然后根据情况进行相应的修改。

(3) 程序输入完后，仔细检查程序中是否存在语法错误或输入错误，在运行程序之前一定要按 F2 键，保存程序。

(4) 按 F9 键，系统将对源程序进行编译和链接，如果在编译和链接过程中没有任何错误，系统将生成 test.exe 可执行文件，并弹出链接成功对话框信息，如图 3-10 所示。如果要运行程序，按 Ctrl+F9 键，BC 将会执行这个程序，这个过程可能会很快，如果不注意，还会以为什么都没有发生，其实这期间进行了编译、链接和运行。要想查看程序运行的结果，可以按 ALT+F5 键，这时会看到程序运行结果，按任意键可返回到 BC 状态。

(5) 如果程序存在着语法错，BC 会提示编译链接不成功。这时，需要根据错误提示（可能不只一条）修改程序的错误。

(6) 如果编译、链接不成功，BC 会提示程序有错误，如图 3-11 所示。这时要按回车键，BC 会显示一个错误信息窗口，显示出所有的错误信息，如图 3-12 所示。

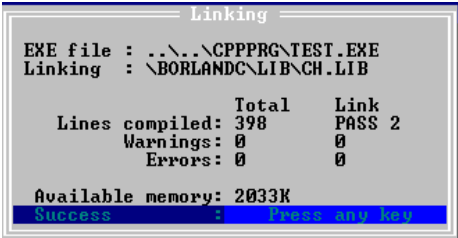


图 3-10 编译链接成功

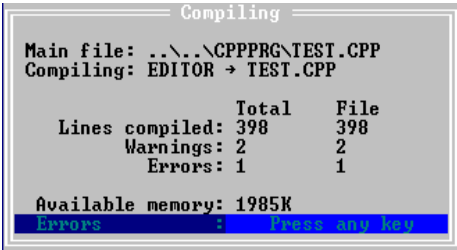


图 3-11 编译链接不成功



图 3-12 错误信息窗口

(7) 在错误信息窗口中，利用上下光标键移动亮度条，被高亮度显示的错误信息所对应的程序语句所在行也被高亮度显示（如图 3-12 所示）。这时，可以按回车键进入编辑状态，根据错误提示信息修改程序。

(8) 通常情况下，必须先改掉第一个错误，才能准确修改后续错误。有时仅仅是一个地方错了，就会引起多条错误信息。

(9) 修改完程序后回到步骤 (4)，直到程序能正常执行。这时，可以按 ALT+F5 键查看运行结果。

3.1.4 程序调试方法

任何一个程序员都无法保证所编写的程序没有错误，语法错在编译阶段就会被发现，链接错在链接时会被发现，但程序内部的逻辑错只能由程序员自己发现并定位。下面介绍的调试手段将有助于程序员快速发现程序的逻辑错。

1. 程序的单步执行

单步执行是指让程序一次只执行一行语句。单步执行程序时，程序员可以观察某些变量或表达式的值，由此来判断哪里出了问题。要想单步执行，必须使程序进入调试状态而不是运行状态。

程序只有在下面几种情况下才能进入调试状态：

- 启动 BC，打开某源程序文件，按 F7 键，进入调试状态。
- 启动 BC，打开某源程序文件，将光标移到某条语句处，按 F4 键程序执行到光标所在行的语句处便停下来，进入调试状态。
- 启动 BC，打开某源程序文件，在某条语句处设置断点，按 Ctrl+F9，程序开始执行，执行到断点处便停下来，进入调试状态。

进入调试状态后，程序的下一条要执行的语句将被用青色亮度条高亮度显示。这时，程序员可以按 F7 键让程序一次执行一条语句，也可以将光标移到下面的任何语句处，按 F4 键让程序直接执行到光标所在行；可以按 Ctrl+F2 键终止调试；可以按 Ctrl+F9 键直接运行程序。

下面的步骤演示了 F7 键和 F4 键的作用。

- 启动 BC，输入程序。将光标移动到第 10 行，按下 F4 键，程序执行到第 10 行处停下来（如图 3-13 所示）。按 ALT+F5 键，会发现第 9 行的 printf 函数已经执行完毕。
- 按 F7 键，程序开始执行 scanf 函数调用，输入 1, 2, 3↵，程序在第 11 行处停下来（如图 3-14 所示）。
- 接着按 F7 键，直到程序执行完毕。这一步读者可自己操作。

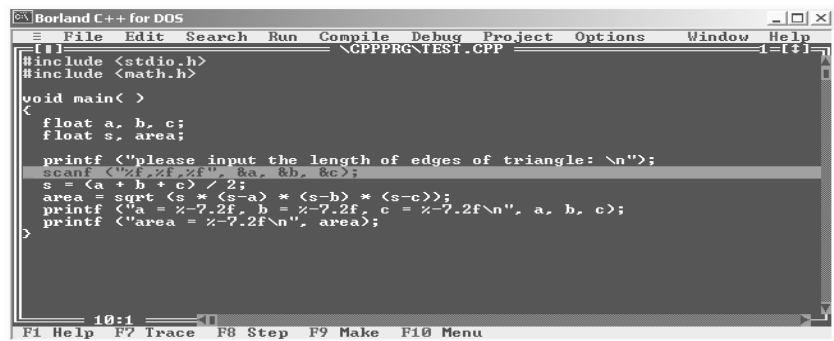


图 3-13 F4 的作用

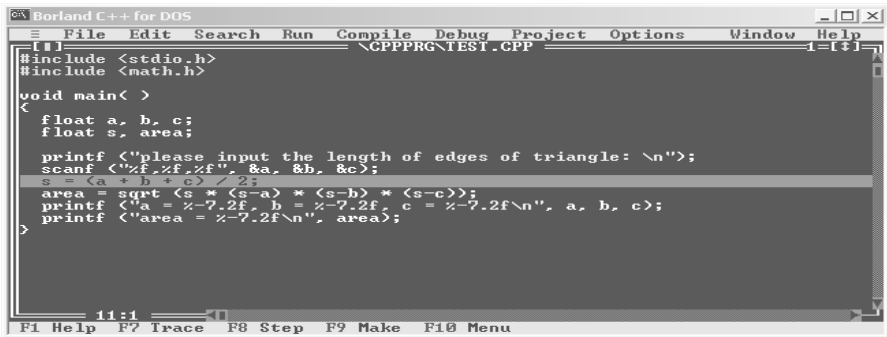


图 3-14 F7 的作用

2. 在程序中设置断点

断点是程序中的一个标记，当程序执行到断点时，程序会暂停下来。

设置断点的命令是 Ctrl+F8 键。将光标移到某条语句处，按 Ctrl+F8 键，这行文本将被用红色亮度条高亮度显示。将光标移到断点处，按 Ctrl+F8 键可以取消这个断点。

将光标移到第 14 行，按下 Ctrl+F8 键就可以设置断点，这时，按 Ctrl+F9 键继续执行，执行到第 14 行会停下来（如图 3-15 所示）。

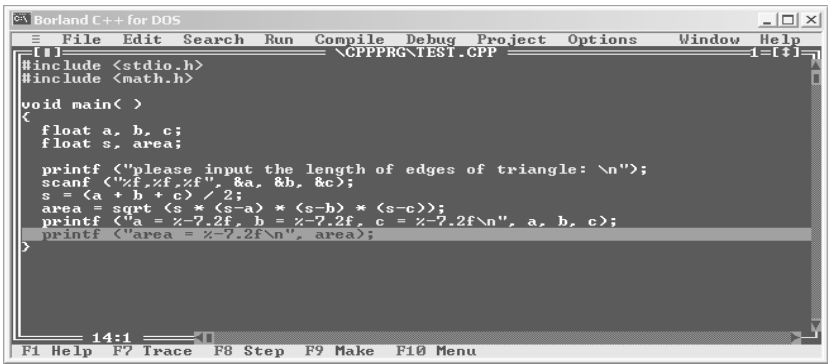


图 3-15 执行到断点

3. 在程序调试过程中观察变量和表达式的值

程序员调试程序的任务有二：一是观察语句执行的顺序是否正确，二是观察变量或表达式的值是否正确。程序员可以从不正常的语句执行顺序和不正确的变量或表达式的值来判断程序的逻辑错出现在何处。

要想观察变量或表达式的值，首先应打开 Watch 窗口。BC 的 IDE 环境中 Watch 窗口专门用来观察变量或表达式的值。选择 Window 菜单的 Watch 命令（按键顺序是：Alt+W, W），Watch 窗口将显示出来，如图 3-16 所示。

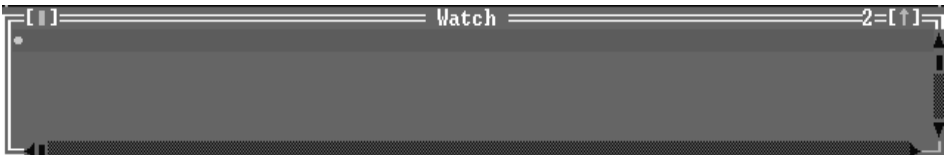


图 3-16 Watch 窗口

在 Watch 窗口中可以同时观察多个变量或表达式的值。当 Watch 窗口处于活动状态时,程序员可以按上下光标键,选择不同的行。如果被选择的是空行,按回车键,可以输入要观察的变量名或表达式(如图 3-17 所示)。如果被选择的行包含有变量或表达式,按回车键,可以修改变量名或表达式(如图 3-18 所示)。输入完后,按回车键即可。



图 3-17 增加观察项



图 3-18 修改观察项

如果被选择的行是非空行,按 Del 键可删除该行。

Watch 窗口中的变量和表达式的值会随着程序的执行而变化。图 3-19 演示了上述例子 test.cpp 执行到第 14 行时,Watch 窗口中的内容。



图 3-19 观察结果

4. 终止调试

当程序处于调试状态时,按 Ctrl+F2 键就会终止调试,程序也将终止运行。

5. 终止程序的运行

当程序正常执行时,按 Ctrl+Break 键并按回车键会终止程序运行。有时需要按两次 Ctrl+Break 键。

3.1.5 工程文件的使用方法

为了使软件更容易维护，程序员通常将程序的不同模块分别用不同的源程序文件来实现，然后将这些源程序文件组合在一起构成一个完整的程序。如何将多个文件组合在一起构成一个程序呢？BC++ 3.1 采用的方法是利用工程（project）文件来管理构成一个程序的多个文件。例如，假设一个程序由两个文件构成：prg1.cpp 和 prg2.cpp，它们的内容分别如下：

prg1.cpp:

```
#include <stdio.h>

int getmax (int x, int y);

void main( )
{
    int a, b, c;

    printf ("input a, b: ");
    scanf ("%d%d", &a, &b);
    c = getmax (a, b);
    printf ("the max of a and b is: %d\n", c);
}
```

prg2.cpp:

```
int getmax (int x, int y)
{
    if (x > y)
        return (x);
    else
        return (y);
}
```

并且这两个程序文件就存放在你的程序目录下，按照下面的操作就可以将它们组合到工程中，构成一个完整的程序。

(1) 启动 BC，如果有窗口被打开，请按 ALT+F3 键关闭它们。

(2) 按 ALT+P 键，激活 Project 菜单（如图 3-20 所示），选择 Open Project 命令，将会看到打开工程文件对话框（如图 3-21 所示）。

(3) 如果在打开工程文件对话框的 Files 列表中选择一个文件，则会打开这个工程文件（工程文件的后缀是.prj），如果在 Open Project File 文本框中输入一个新工程文件的名



图 3-20 Project 菜单

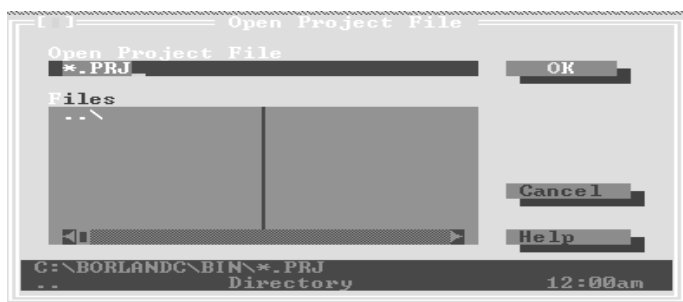


图 3-21 打开工程文件对话框

字，比如输入 `pro`，然后按回车键，BC 将创建一个新工程文件 `pro.prj`，读者会看到在 BC 环境中打开一个工程窗口，工程窗口的标题是工程文件的名字，内容是空的，因为创建的工程还是一个空工程（如图 3-22 所示）。

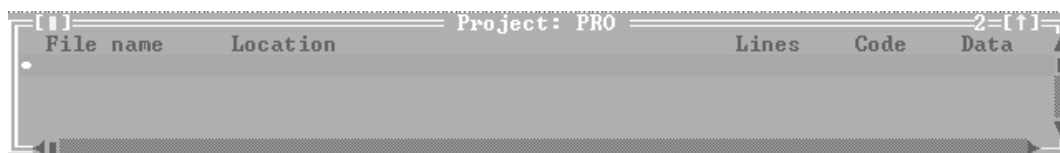


图 3-22 空工程 pro 的工程窗口

（4）现在要向创建的工程文件 `pro.prj` 中增加工程项目。将前面编写的各种 `.cpp` 文件作为工程项目插入到工程文件中，才可以将它们组合成程序。BC 可以同时打开多个 `.cpp` 文件，但一次只能打开一个工程文件，工程文件的内容显示在工程窗口中。激活工程窗口后，按 `Insert` 键，可激活增加工程项目对话框（如图 3-23 所示）。



图 3-23 增加工程项目对话框

（5）在增加项目对话框的 `Name` 输入框中输入 `*.cpp` 后按回车键，在 `Files` 列表中选择 `prg1.cpp` 文件，按回车键，就可以将 `prg1.cpp` 文件插入到工程文件中。在增加项目对话框中，一次可以向工程文件增加多个项目。在 `Files` 列表中选择 `prg2.cpp` 文件，按回车键将 `prg2.cpp` 文件也插入到工程文件中。最后按 `ESC` 键返回工程窗口。读者会看到，`pro.prj` 文件的工程窗口中包含了 `prg1.cpp` 和 `prg2.cpp` 文件（如图 3-24 所示）。这时，读者可以按 `Ctrl+F9` 键编译运行整个程序。



图 3-24 工程 pro 的工程窗口

工程文件所包含的每个文件都代表一个程序模块，程序模块可以是.cpp 的源程序文件也可以是.obj 文件，也可以是.lib 文件。比如，可以将 prg2.cpp 编译生成 prg2.obj 文件，然后将工程文件 pro.prj 中的 prg2.cpp 用 prg2.obj 文件来代替，也可以实现对工程文件 pro.prj 的编译运行。包含 prg1.cpp 和 prg2.obj 文件的工程窗口如图 3-25 所示。

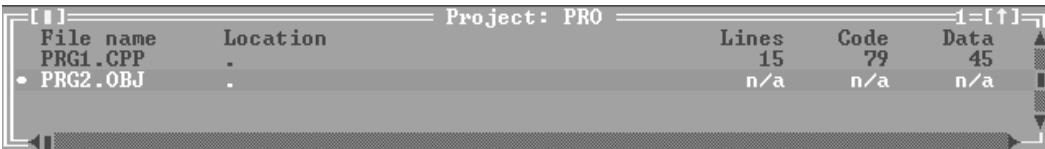


图 3-25 包含 prg2.obj 的工程 pro 的工程窗口

程序员可以在工程窗口中选择一个源程序文件，按回车键打开这个源程序文件进行编辑，不需要再利用打开文件对话框来打开工程中的源程序文件。

- 注意：
- 由于工程中的文件都属于同一个程序，因此只允许一个文件包含 main 函数。
 - 只能用 Project 菜单中的 Open Project 命令打开工程文件，不能按 F3 键或用 File 菜单中的 Open 命令打开工程文件。
 - 工程窗口所包含的文件中，只有源程序文件是可编辑的。

3.1.6 带参数的 main 函数的运行方法

在 BC 环境下可以运行带命令行参数的程序，而不用退到 DOS 下运行。具体方法是：按 ALT+R 键，激活 Run 菜单，选择 Arguments...命令（如图 3-26 所示），激活程序参数输入框，如图 3-27 所示。在程序参数输入框中，输入程序的命令行参数，即 argv[1]以及后面的字符串，按回车键后，就可以按 Ctrl+F9 运行程序了。比如，如果在程序参数输入框中输入了 12，并运行【例 9-16】的程序 example9-16，则相当于在 DOS 状态下执行命令：example9-16 12。

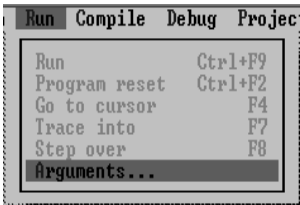


图 3-26 Run 菜单

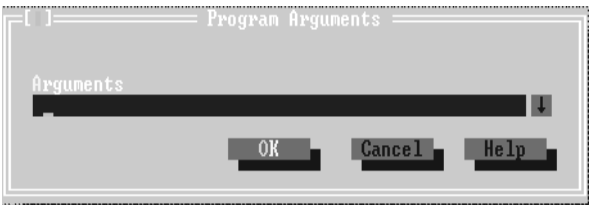


图 3-27 程序参数输入框

3.2 Visual C++ 6.0 开发环境

美国微软公司出品的 Visual C++(以下简称 VC)是 Windows 平台上最流行的 C/C++ 集成开发环境。从 1993 年发行 1.0 版本开始,于 2003 年推出了最新的 7.0 版本 (Visual C++ .NET 2003)。本书将介绍如何在流行更广泛的 6.0 版本下运行 C 语言程序。

3.2.1 启动 Visual C++ 6.0 环境

启动 VC 环境常用的方法有 3 种:双击桌面图标;从桌面左下角的“开始”菜单进入;从桌面左下角的“运行”功能中进入。

1. 通过双击桌面图标直接启动 VC 环境

在桌面上找到 VC 的图标,如图 3-28 所示,使用鼠标左键进行双击,即可打开 VC 环境。



图 3-28 通过鼠标左键双击桌面图标直接启动 VC 环境

2. 从桌面左下角的“开始”菜单进入 VC 环境

- (1) 用鼠标单击桌面左下角的“开始”菜单 (见图 3-29)。
- (2) 将鼠标指针移至“程序(P)”处。
- (3) 将鼠标指针再水平右移,在下一级菜单中移至 Microsoft Visual Studio 6.0 处。
- (4) 将鼠标指针右移至下一级菜单上,并将鼠标指针移动到 Microsoft Visual C++ 6.0 处,单击会出现如图 3-31 所示的情况,这就是编程时要用到的 VC 环境。

3. 从桌面左下角的“运行(R)”功能中进入 VC 环境

- (1) 用鼠标单击桌面左下角的“开始”菜单。
- (2) 将鼠标指针移到“运行(R)”处,单击会出现一个“运行”对话框。
- (3) 在弹出的对话框中输入 msdev,如图 3-30 所示,然后单击“确定”按钮,即会出现如图 3-31 所示的 VC 编程环境。

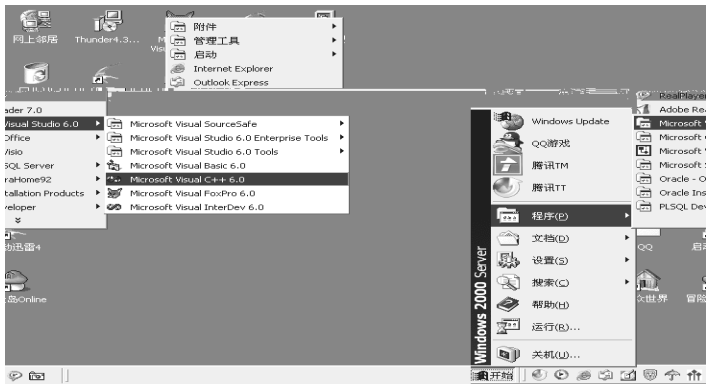


图 3-29 从桌面左下角的“开始”菜单进入 VC 环境



图 3-30 “运行”弹出的对话框

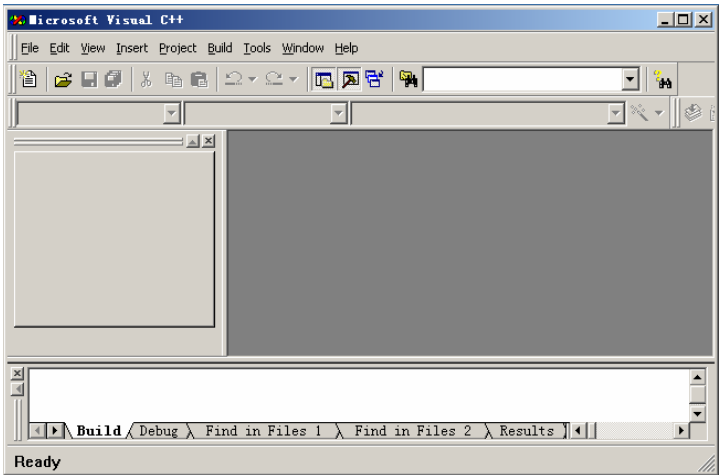


图 3-31 VC 环境窗口

3.2.2 建立或打开源程序文件

1. 建立新的源程序文件

进入 VC 环境以后，如果需要创建一个新的源程序文件（.cpp 的文件），可以单击

File 菜单项，然后选择 New 命令（或直接按 Ctrl+N 键），就会弹出 New 对话框，初始出现的是 Projects 选项卡，用鼠标单击 Files 标签，进入 Files 选项卡（如图 3-32 所示）。

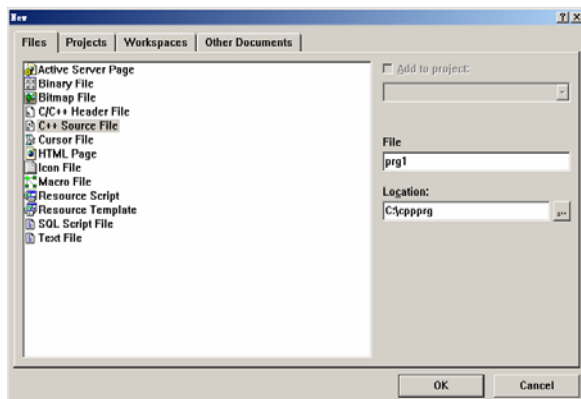


图 3-32 新建文件对话框

在如图 3-32 所示的对话框中选择文件类型。因为是要建立一个 .cpp 的源程序文件，所以就应当选择 C++ Source File 项目。然后需要在 File 的文本框中输入要创建的源程序文件名（这里假设是 prg1），再在 Location 文本框中输入或选择文件保存的路径（假设为 c:\cppprg），最后单击 OK 按钮。这样就成功新建了一个名为 prg1.cpp 的源程序文件，如图 3-33 所示。

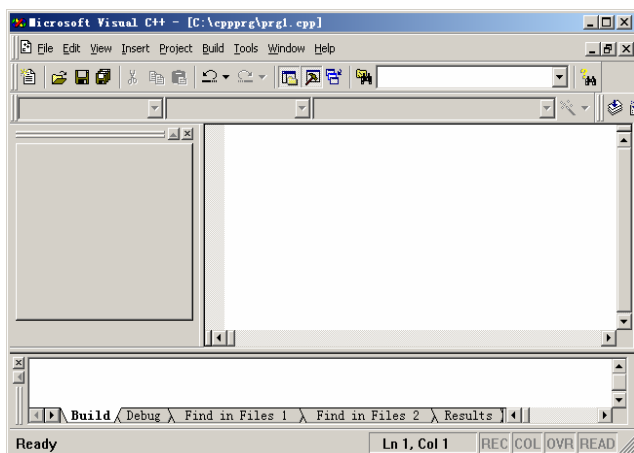


图 3-33 新建文件 prg1.cpp 后的 VC 环境

在图 3-33 中，标题栏上的 [c:\cppprg\prg1.cpp] 表示刚刚所新建的源程序文件名，窗口的右面大片空白区域就是程序编辑区，程序员可以在此区域编写程序，编写好的程序可以按 Ctrl+F2 键或单击工具栏上的磁盘图标来保存。

2. 打开已保存的源程序文件

进入 VC 环境后，如果想打开以前保存了的源程序文件，可以单击 File 菜单项，然

程序编写完毕，单击 **Build** 菜单项，然后选择 **Build** 命令或直接按 **F7** 键，开始编译和链接。但在正式编译之前，VC 先会弹出如图 3-36 所示的对话框，询问是否建立一个默认的项目工作区。VC 必须有项目才能编译，所以这里必须回答 **Yes**。然后在 **.cpp** 文件的目录里会生成与 **cpp** 源程序文件同名的 **.dsw** 和 **.dsp** 等文件。以后可以直接打开这些文件继续编写程序，不必再重复上面的过程。

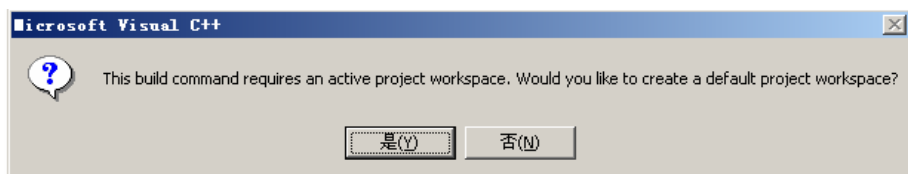


图 3-36 询问是否建立项目工作区

如果修改完代码后没有保存，这时还会提示是否保存。保存后，随着硬盘清脆的响声，VC 下方的白色消息区会显示如图 3-37 所示的内容。它表明编译和链接过程中没有任何错误信息。

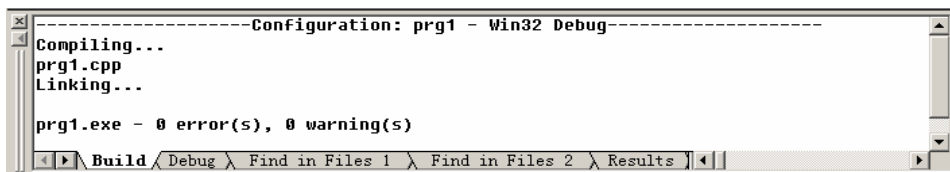


图 3-37 编译信息

如果没有错误，VC 将生成与 **cpp** 源程序文件同名的执行文件（**.exe** 文件），该执行文件存放在与源程序文件同一文件夹下的 **Debug** 文件夹下。此时就可以运行了，按 **Ctrl+F5** 键或点击工具栏上的“!”图标，程序将在一个新的 DOS 窗口中运行。窗口的最后会显示一行“**Press any key to continue**”，这是 VC 加上的提示，并不是程序的输出。看到此条提示时，说明程序已经运行完毕，按照提示按任意键关闭窗口。

如果编译出错，在 VC 环境的下面窗口中会列出错误的位置与内容，并统计错误和警告的个数。在错误信息窗口中，利用鼠标双击错误信息，该错误信息所对应的程序语句所在行将会出现一箭头标识（如图 3-38 所示）。这时，根据错误提示信息修改你的程序。通常情况下，必须先改掉第一个错误，才能准确修改后续错误。有时仅仅是一个地方错了，就会引起多条错误信息。

3.2.4 程序调试方法

VC 的调试功能极其强大，熟练使用后将如虎添翼。

要进入程序调试，可首先将程序进行编译、链接，当没有错误后，单击 **Build** 菜单项，然后选择 **Start Debug** 中的 **Step Into**，或直接按 **F11** 键就进入调试状态，此时会弹出一个 **Debug** 的窗口，窗口中有许多基本调试命令和各个调试窗口开关。表 3-2 是基本的调试命令及其图标和快捷键对照表。

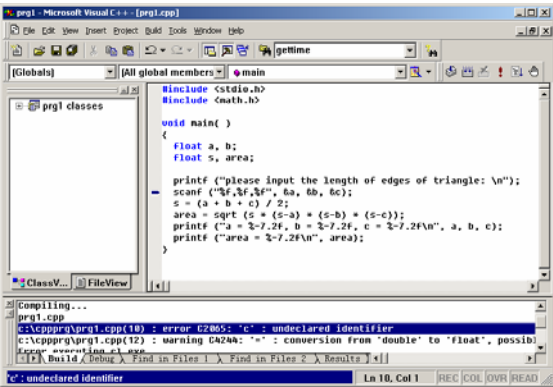


图 3-38 错误信息窗口

表 3-2 VC 基本调试命令表

命 令	图标	快捷键	说 明
Go		F5	开始或继续在调试状态下运行程序
Run to Cursor		Ctrl+F10	运行到光标所在行
Stop Debugging		Shift+F5	停止调试程序
Insert/Remove Breakpoint		F9	插入或删除断点
Step Into		F11	进入函数内部单步执行
Step Over		F10	执行下一条语句，不进入函数
Step Out		Shift+F11	跳出当前函数

调试基本功能与 BC 基本相同，请参考 2.1 节。但很多高级功能是 VC 特有的，如图 3-39 所示。

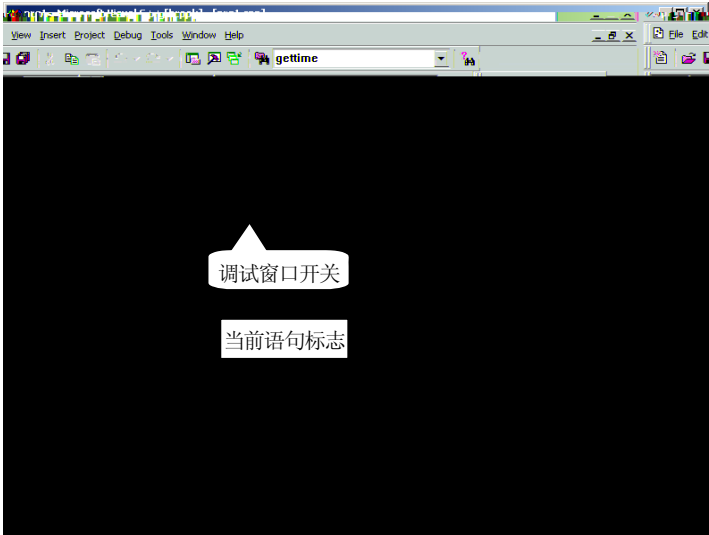


图 3-39 VC 调试界面

图 3-39 中的各个调试窗口可以改变位置，也可以随意开关。左下方的 **Variables** 窗口随着程序的执行动态更新，显示选定的上下文（Context）中的局部变量和函数调用的返回值等信息。右下方的 **Watch** 窗口有很强的定制性，用户可以自行在里面填写想要监视的变量，任意修改变量的值，也可以直接把源代码中的变量拖放到此窗口中。

Memory 窗口是直接的内存映像，在这里能直接查询和修改任意地址的数据。对初学者来说，看此窗口能更深刻地理解各种变量、数组和结构等是如何占用内存的。**Call Stack** 窗口显示了函数调用的嵌套情况。

此外，还可以在汇编代码级进行调试，以及修改代码后，不需重新编译运行就能继续调试。因为这些功能并不常用，仅在比较特别的情况下才能发挥巨大作用，所以本书不再详细介绍。

3.2.5 建立工程

像 BC 通过建立工程文件来管理多个源程序文件一样，在 VC 中也可以建立相应的工程文件。其实在 VC 中必须有工程（或项目），程序才能编译。所以学会建立工程对于在 VC 环境下编程就更为重要。

建立一个 VC 工程（Project）的步骤如下：

（1）在如图 3-31 所示的 VC 环境窗口中单击 **File** 菜单项，然后选择 **New** 命令，就会出现如图 3-40 所示的对话框。

（2）此时编程者需要在工程类别列表中选择合适的工程类别。“C 语言程序设计”课程主要涉及到的是 C 语言的语法规则及数据类型，所以一般选择 **Win32 Console Application** 类型的工程就可以了。

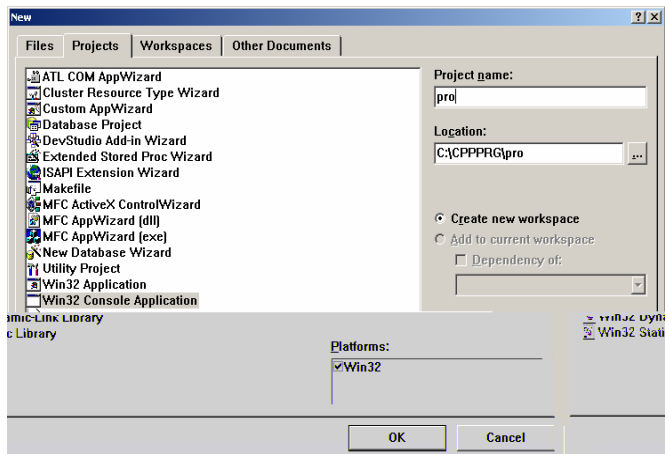


图 3-40 新建工程的对话框

（3）在选择了合适的工程类型后，就要填写工程的名称。具体方法是在如图 3-40 所示对话框中的 **Project name** 编辑框中输入工程名称，这里输入的工程名称是 **pro**，编程者可以根据自己的需要和喜好输入其他的名称。

（4）再在 **Location** 编辑框中选择工程保存的目录，并单击 **OK** 按钮，就会弹出如图

3-41 所示的对话框。

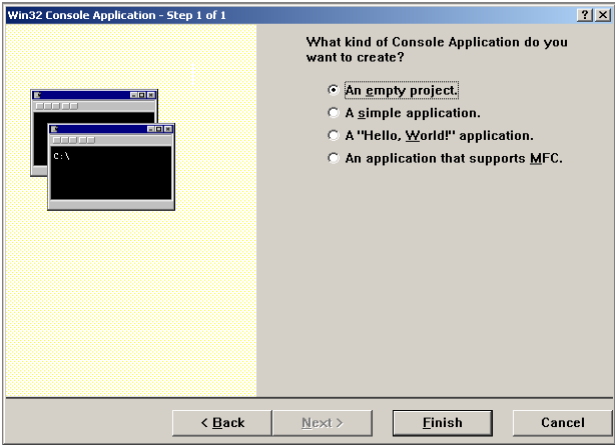


图 3-41 选择 Win32 Console Application 后的模板

(5) 在如图 3-41 所示的对话框中，列出了 4 种预设的工程模板，选择第 1 项 An empty project，然后单击 Finish 按钮，会弹出如图 3-42 所示的新工程报告对话框。

(6) 从图 3-42 所示的对话框中可以看到新建工程的一些简要信息，确定信息准确无误后，单击 OK 按钮，这样就成功地新建了一个 VC 工程。

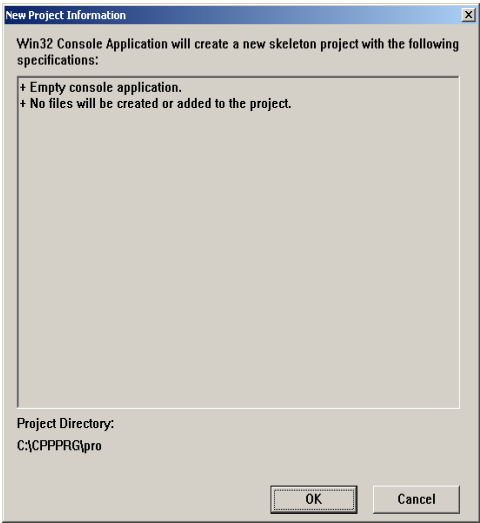


图 3-42 新建工程报告对话框

(7) 现在观察一下 VC 的 (New) Projects 做了什么工作。图 3-43 显示了刚刚新建的 pro 工程在 VC 环境中的情况。窗口的左边部分是 Workspace，它显示了有关工程的信息，包括类信息、资源信息、源文件信息等。单击下部的 FileView，在 Workspace 中可以看到 3 个目录：一般.cpp 文件放在 Source Files 中；头文件.h 或.hpp 放在 Header Files 中；资源文件放在 Resource Files 中；还可以建立自己的目录。

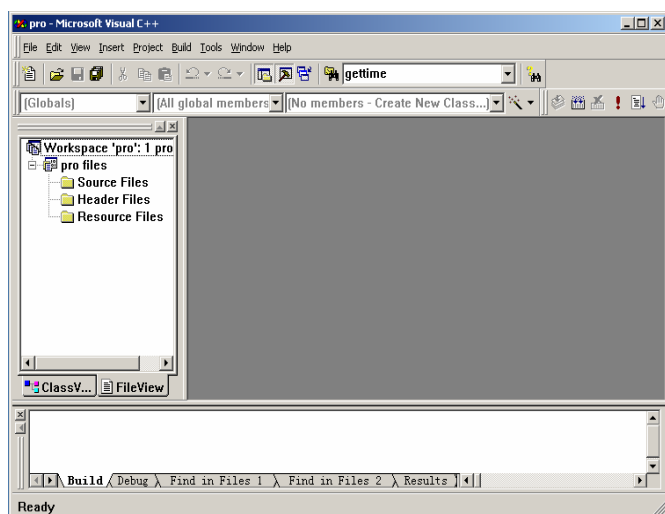


图 3-43 新建 pro 工程的展开全貌

3.2.6 向已有工程中加入新文件

(1) 如果希望在上面已建立的工程 `pro` 之中新建一个文件，可以单击 **File** 菜单项，然后单击 **New** 子菜单项，就会弹出如图 3-44 所示的对话框。

(2) 在如图 3-44 所示的对话框中选择文件类型。例如：要想建立一个 `.cpp` 文件，那么就可以选择 **C++ Source File** 项目。如果想要建立一个 `.h` 文件，那么可以选择 **C/C++ Header File** 项目。

(3) 需要输入文件名称，这里输入 `cpp1`。再选择文件保存的路径，最后确认选中 **Add to project** 复选框，并单击 **OK** 按钮。这样就成功地新建了一个名为 `cpp1.cpp` 的文件，并已加入到了前面创建的 `pro` 工程之中。

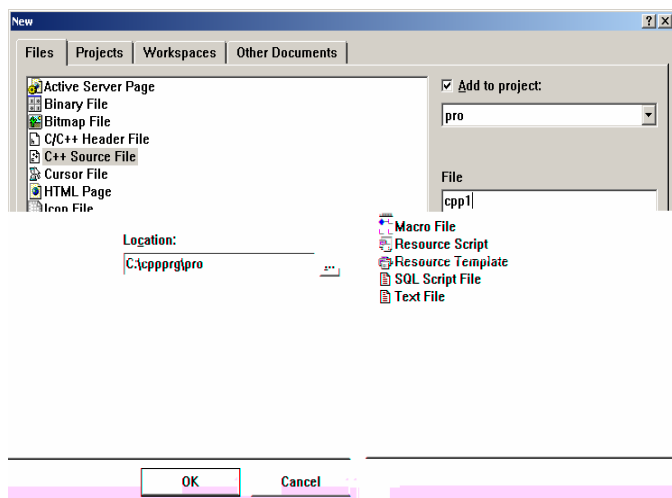


图 3-44 新建文件对话框

C 语言上机实验题

☆ 学习要点

- 通过上机编程进一步熟悉和掌握 C 语言集成开发环境。
- 熟练掌握上机程序调试的方法。
- 掌握模块化程序设计的编程方法。
- 独立完成平时上机实验题目。
- 完成期末上机实验考试题目的准备工作。

4.1 平时上机实验题目

4.1.1 实验 1 熟悉上机环境和基本数据类型编程练习

1. 简单程序练习

下面的程序是通过 `printf` 函数直接输出一个由 “*” 号组成的一个三角形。

```
#include <stdio.h>

void main( )
{
    printf ("*\n");
    printf ("* *\n");
    printf ("* * *\n");
    printf ("* * * *\n");
}
```

现要求编写一程序，输出的结果也是一个三角形，只不过组成三角形的符号不一定是 “*” 号，而是由用户任意输入的符号构成的。

[提示：定义一字符型变量，通过 `scanf` 函数接收用户输入的字符，然后通过 `printf`

函数将其输出。]

2. 基本数据类型表示范围练习

假设有如下程序：

```
#include <stdio.h>

void main( )
{
    char ch1, ch2, ch;
    unsigned char c;
    int a;

    ch1 = 80;
    ch2 = 60;
    ch = ch1 + ch2;
    c = ch1 + ch2;
    a = ch1 + ch2;

    printf ("ch1 + ch2 = %d\n", ch1+ch2);
    printf ("ch = %d\n", ch);
    printf ("c = %d\n", c);
    printf ("a = %d\n", a);
}
```

问题 1 运行该程序，写出输出结果。并说明为什么存在输出结果的差异？

问题 2 如果要求四行输出的结果均为 140，在不改变变量的数据类型的基础上，则应修改程序中的哪条语句？怎样修改？

问题 3 编一程序，要求两个负整数相加，输出的结果为一正整数。

4.1.2 实验 2 输入/输出与顺序结构编程练习

1. 键盘输入与屏幕输出练习

用下面的 scanf 函数输入数据，使 a = 3, b = 7, x = 8.5, y = 71.82, c1 = 'A', c2 = 'a', 问在键盘上如何输入？要求把下列程序补充完整。

```
#include <stdio.h>
void main( )
{
    int a, b;
    float x, y;
    char c1, c2;
    scanf ("a = %d b = %d", &a, &b);
    scanf ("x = %f y = %e", &x, &y);
```

```
scanf ("c1 = %c c2 = %c", &c1, &c2);
}
```

[提示: 在“格式控制”字符串中除了格式说明以外还有其他字符, 在输入数据时应输入与这些字符相同的字符。为了验证输入格式的正确与否, 应在程序中补充调用标准的输出函数 printf。]

2. 按输出结果要求编写程序

若 $a=3$, $b=4$, $c=5$, $x=1.2$, $y=2.4$, $z=-3.6$, $u=51274$, $n=128765$, $c1='a'$, $c2='b'$, 想得到以下输出格式和结果, 请写出完整的程序。要求输出结果如下:

```
a = 3□□□b = 4□□□c = 5
x = 1.20000, y = 2.400000, z = -3.600000
x + y = □3.60□□y + z = -1.20□□z + x = -2.40
u = 51274□□□n = □□□128765
c1 = 'a'□or□97(ASCII)
c2 = 'b'□or□98(ASCII)
```

[提示: 根据题目中变量的值定义合适的变量类型。正确定义和使用 printf 函数中的格式控制字符串。□表示空格。]

3. 计算定期存款本利之和

设银行定期存款的年利率 rate 为 2.25%, 并已知存款期为 n 年, 存款本金为 capital 元, 试编程计算 n 年后本利之和 deposit。要求定期存款的年利率 rate、存款期 n 和存款本金 capital 均由键盘输入。

4.1.3 实验3 选择结构编程练习

1. 多分支编程练习

根据以下函数关系, 对输入每个 x 值, 计算出相应的 y 值。

$$y = \begin{cases} 0 & \text{当 } x < 0 \\ x & \text{当 } 0 \leq x < 10 \\ 10 & \text{当 } 10 \leq x < 20 \\ -0.5x + 20 & \text{当 } 20 \leq x < 40 \end{cases}$$

问题 1 如何用 if 语句来编程实现上述功能?

问题 2 如何用 switch 语句来编程实现上述功能?

[提示: 使用一临时变量 c, 当 $x < 0$ 时, $c = -1$; 否则 $c = x/10$, 再根据 c 的值用 switch 语句来实现多分支。]

2. 数据整除判断

编程实现：输入一个整数，判断它能否被 3，5，7 整除，并输出以下信息之一：

- ① 能同时被 3，5，7 整除；② 能被其中两数（要指出哪两个）整除；③ 能被其中一个数（要指出哪一个）整除；④ 不能被 3，5，7 任一个整除。

[提示：(1) 判断能否被某一个数整除应采用求余运算。(2) 在一个程序中利用选择结构把四种可能的信息全部表示出来。]

3. 计算圆的面积或周长

程序功能：输入圆的半径 r 和运算标志符 m ，按照运算标志符进行指定计算。当 m 等于's'时，只计算圆的面积；当 m 等于'c'时，只计算圆的周长；当 m 等于'a'时，则圆的面积和周长均计算。

4.1.4 实验 4 循环结构编程练习

1. 猜数游戏

编程先由计算机“想”一个 1~100 之间的数请人猜，如果猜对了，在屏幕上输出人猜了多少次才猜对此数，以此来反映猜数者“猜”的水平，则游戏结束；否则计算机给出提示，告诉人所猜的数是太大还是太小，最多可以猜 10 次，如果猜了 10 次仍未猜中的话，则停止本次猜数，然后继续猜下一个数。每次运行程序可以反复猜多个数，直到操作者想停止时才结束。

2. 彩票选号

某市体育彩票采用整数 1、2、3、…、36 表示 36 种体育运动，一张彩票可选择 7 种运动。编写程序，选择一张彩票的号码，使这张彩票的 7 个号码之和是 105 且相邻两个号码之差按顺序依次是 1、2、3、4、5、6。例如第一个号码是 1，则后续号码应是 2、4、7、11、16、22。

[提示：(1) 若已知 7 个号码中的第一个号码是 k_0 ，则后续号码之间的关系是： $k_i - k_{i-1} = i$ ，其中 $i = 1, 2, 3, 4, 5, 6$ 。(2) $k_0 + k_1 + k_2 + k_3 + k_4 + k_5 + k_6$ 之和必须是 105。(3) $k_i \leq 36$ ，其中 $i = 0, 1, 2, 3, 4, 5, 6$ 。]

4.1.5 实验 5 数组编程练习

1. 数字检查

通过键盘输入 n ($n < 50$) 个 4 位数（输入 -1 时结束），统计这 n 个 4 位数中各位上的数字均是偶数的数的个数，并把这些 4 位数按从大到小的顺序进行输出。

[提示：定义两个一维数组 a 和 b ，数组 a 用于存放输入 4 位数，数组 b 用于存放满足条件的 4 位数；判断 4 位数上的各个数是否为偶数的方法可将先将该数的每一位数拆

分出来，然后进行判断即可。]

2. 成绩排名

假设有五位同学四门功课的成绩如下，现要求得每位同学的总分，并按总分从高到低的顺序进行排序，输出其名次，如果总分相同，则按语文和数学分数之和从高者排前，低者排后，但名次一样。

姓 名	语文	数学	英语	综合
张大明	120	130	110	280
李小红	110	120	105	290
王志强	108	128	126	278
汪晓成	112	135	122	286
李 丹	100	120	108	276

[提示：用二维数组 score 存放每个学生各门课程的成绩，二维数组定义时最后多定义一行，用于存放每个学生的总分；用一字符串数组 name 存放学生的姓名。]

4.1.6 实验 6 函数编程练习

1. 去掉字符串前面的*号

规定输入的字符串中只包含字母和*号。请编写函数 fun，它的功能是将字符串前面连续的 *号全部删除，中间和尾部的*号不删除。例如，字符串中的内容为：*****A*B C*D E F*G****，删除后，字符串中的内容应当是： A*B C*D E F*G****，在编写函数时不得使用 C 语言提供的字符串函数。注意：部分源程序给出如下。请勿改动主函数 main 和其他函数中的任何内容，仅在函数 fun 的花括号中填入编写的若干语句。

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

void fun (char a[ ])
{

}

void main( )
{
    char s[81];
    printf ("Input a string: ");
    gets (s);
    fun (s);
    printf ("The string after deleted: ");
    puts (s);
}
```

2. 去掉字符串后面的*号

如果要去掉字符串后面的*号，上述程序如何改动？

3. 完全平方数

编写一函数，其功能是：在 3 位整数（100~999）中寻找既是完全平方数（某个数的平方），又有两位数字相同的整数，例如 144，676 等，并依次从小到大存入数组 b 中；满足该条件的整数的个数通过所编制的函数返回。

4.1.7 实验 7 指针编程练习

1. 字符串合并

使用指针参数编写字符串 s 和 t 进行并置的程序，`stradd(s, t, f)`，其中 s 和 t 为字符串，f 为标志。当 `f=0` 时，字符串 s 并置到字符串 t 后，当 `f=1` 时，字符串 t 并置到字符串 s 后。要求不能使用 `strcpy` 和 `strcat` 函数。

[提示：利用两个临时指针分别指向目的串的尾部和源串的头部，然后一个字符一个字符地进行赋值。]

2. 数据交换

从键盘输入 3 个整数，将 3 个数中的最大数与最小数交换，要求用指针作为函数参数的方法处理。

[提示：在子函数中先要求出最大数与最小数所在的位置，然后进行交换。]

4.1.8 实验 8 数组、指针和函数综合编程练习

打印最高分和学号

假设每班人数不超过 40 人，具体人数由键盘输入，试编程打印最高分及其学号。

程序 1 用一维数组和指针变量作为函数参数，编程打印某班一门课成绩的最高分及其学号。

程序 2 用二维数组和指针变量作为函数参数，编程打印 3 个班学生（假设每班 4 个学生）的某门课成绩的最高分，并指出具有该最高分成绩的学生是第几个班的第几个学生。

程序 3 用指向二维数组第 0 行第 0 列元素的指针作为函数参数，编写一个计算任意 m 行 n 列二维数组中元素的最大值，并指出其所在的行列下标值的函数，利用该函数计算 3 个班学生（假设每班 4 个学生）的某门课成绩的最高分，并指出具有该最高分成绩的学生是第几个班的第几个学生。

程序 4 编写一个计算任意 m 行 n 列二维数组中元素的最大值，并指出其所在的行

列下标值的函数，利用该函数和动态内存分配方法，计算任意 m 个班、每班 n 个学生的某门课成绩的最高分，并指出具有该最高分成绩的学生是第几个班的第几个学生。

4.1.9 实验 9 复杂数据类型编程练习

1. 计算天数

编写一函数 `Day_Of_Year (nowdate)`，该函数的功能是根据日期参数 `nowdate`，返回这一日期是当年的第几天。

[提示：函数参数用一包含年、月、日的结构体。]

2. 学生成绩

有 n 个学生 (n 由键盘输入)，每个学生的信息包括：学号、姓名及语文、数学、英语三门课的成绩。要求用结构体指针编程序，从键盘上输入学生人数和每个学生的数据，打印出每个学生的学号、姓名、总成绩和平均成绩，并按总成绩从高到低排序。

4.1.10 实验 10 文件编程练习

1. 选票统计问题

对 10 个候选人进行选举，现有一个不超过 100 条记录的选票数据文件 `IN.DAT`，其数据存放的格式是每条记录的长度均为 10 位，第一位表示第一个人的选中情况，第二位表示第二个人的选中情况，以此类推。每一位内容均为字符 0 或 1，1 表示此人被选中，0 表示此人未被选中，若一张选票选中人数小于等于 5 个人时则被认为是无效的选票。请编制一程序来统计每个人的选票数，并把每个人选票数输出到文件 `OUT.DAT` 中。

2. 文件中大小写英文字符的转换

编写一程序，将某文本文件中的大写英文字符转换成小写字符，小写英文字符转换成大写字符，其他字符不变。要求转换的文件名通过命令行参数提供。

4.2 平时上机实验题目参考答案

4.2.1 实验 1 熟悉上机环境和基本数据类型编程练习

1. 简单程序练习

【参考答案】

```
#include <stdio.h>

void main( )
{
```

```
char ch;

scanf ("%c", &ch);
printf ("%c\n", ch);
printf ("%c %c\n", ch, ch);
printf ("%c %c %c\n", ch, ch, ch);
printf ("%c %c %c %c\n", ch, ch, ch, ch);
}
```

程序运行结果（假设输入的字符为：A↵）：

2. 基本数据类型表示范围练习

【参考答案】 问题 1

```
ch1 + ch2 = 140
ch = -116
c = 140
a = 140
```

因为 ch1、ch2 和 ch 都是 char 型变量，其存放数据的表示范围是-128～127，所以所求范围是-128～127。

```
ch = ch1 + ch2;
c = ch1 + ch2;
a = ch1 + ch2;

printf ("ch1 + ch2 = %d\n",ch1+ch2);
printf ("ch = %u\n", (unsigned char)ch);
printf ("c = %d\n", c);
printf ("a = %d\n", a);
}
```

程序运行结果:

```
ch1 + ch2 = 140
ch = 140
c = 140
a = 140
```

【参考答案】 问题3

```
#include <stdio.h>

void main( )
{
    char ch1, ch2, ch;

    ch1 = -80;
    ch2 = -60;
    ch = ch1 + ch2;
    printf ("ch = %d\n", ch);
}
```

程序运行结果:

```
ch = 116
```

4.2.2 实验2 输入/输出与顺序结构编程练习

1. 键盘输入与屏幕输出练习

【参考答案】

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

```
#include <stdlib.h>

void main( )
{
    int a, b;
    float x, y;
    char c1, c2;

    scanf ("a = %d b = %d", &a, &b);
    fflush(stdin);
    scanf ("x = %f y = %e", &x, &y);
    fflush(stdin);
    scanf ("c1 = %c c2 = %c", &c1, &c2);

    printf ("a = %d b = %d\n", a, b);
```

```
printf ("x + y = %5.2f y + z = %.2f z + x = %.2f\n", x+y, y+z, z+x);
printf ("u = %-8ldn = %9ld\n", u, n);
printf ("c1 = \'%c\' or %d(ASCII)\n", c1, c1);
printf ("c2 = \'%c\' or %d(ASCII)\n", c2, c2);
}
```

3. 计算定期存款本利之和

【参考答案】

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

void main( )
{
    int n;           //存款年数
    double rate;     //年存款利率
    double capital;  //存款本金
    double deposit;  //本利之和

    printf ("Please enter rate, year, capital: ");
    scanf ("%lf,%d,%lf", &rate, &n, &capital); //输入数据
    deposit = capital * pow (rate+1, n);        //计算本利之和, pow为幂函数
    printf ("deposit = %.2f\n", deposit);        //打印存款利率之和
}
```

程序运行结果:

```
Please enter rate, year, capital: 0.0225,1,10000✓
deposit = 10225.00
```

4.2.3 实验3 选择结构编程练习

1. 多分支编程练习

【参考答案】 问题1

```
#include <stdio.h>

void main( )
{
    int x;
    float y;
```

```
printf ("x = ");
scanf ("%d", &x);
if (x < 0)
    y = 0;
else if (x >= 0 && x < 10)
    y = x;
else if (x >= 10 && x < 20)
    y = 10;
else if (x >= 20 && x < 40)
    y = -0.5 * x + 20;
else
    y = -2;

if (y != -2)
    printf ("y = %f\n", y);
else
    printf ("error\n");
}
```

程序运行结果:

```
x = 30✓
y = 5.000000
```

【参考答案】 问题 2

```
#include <stdio.h>

void main( )
{
    int x, c;
    float y;

    printf ("x = ");
    scanf ("%d", &x);

    c = x < 0 ? -1 : x / 10;
    switch (c)
    {
        case -1: y = 0;
                 break;
        case 0: y = x;
                break;
```



```
    case 1: y = 10;
           break;
    case 2:
    case 3: y = -0.5 * x + 20;
           break;
    default: y = -2;
}
if (y != -2)
    printf ("y = %f\n", y);
else
    printf ("error\n");
}
```

2. 数据整除判断

【参考答案】

```
#include <stdio.h>

void main( )
{
    int x;
    char tag;

    printf ("Input a integer number: ");
    scanf ("%d", &x);

    tag = 0;
    if (x % 3 == 0)
        tag = tag | 4;
    if (x % 5 == 0)
        tag = tag | 2;
    if (x % 7 == 0)
        tag = tag | 1;
    switch (tag)
    {
        case 0: printf ("%d can not be divided by 3,5,7 exactly\n", x);
                break;
        case 1: printf ("%d can be divided by 7 exactly\n", x);
                break;
        case 2: printf ("%d can be divided by 5 exactly\n", x);
                break;
        case 3: printf ("%d can be divided by 5,7 exactly\n", x);
    }
```

```
        break;
    case 4: printf ("%d can be divided by 3 exactly\n", x);
        break;
    case 5: printf ("%d can be divided by 3,7 exactly\n", x);
        break;
    case 6: printf ("%d can be divided by 3,5 exactly\n", x);
        break;
    case 7: printf ("%d can be divided by 3,5,7 exactly\n", x);
        break;
}
}
```

程序运行结果:

```
Input a integer number: 15✓
15 can be divided by 3,5 exactly
```

3. 计算圆的面积或周长

【参考答案】

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

#define PI 3.14159

void main( )
{
    float r, s, c;
    char m;

    printf ("Input r = ");
    scanf ("%f", &r);
    fflush (stdin);
    printf ("Input m = ");
    scanf ("%c", &m);

    switch (m)
    {
        case 's': s = PI * r * r;
                printf ("s = %f\n", s);
                break;
        case 'c': c = PI * r * 2;
                printf ("c = %f\n", c);
```

```
        break;

    case 'a':    s = PI * r * r;
                c = PI * r * 2;
                printf ("s = %f\n", s);
                printf ("c = %f\n", c);
                break;
    default :   printf ("error!\n");
}
}
```

程序运行结果:

```
Input r = 4✓
Input m = a✓
s = 50.265442
c = 25.132721
```

4.2.4 实验4 循环结构编程练习

1. 猜数游戏

【参考答案】

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>

void main( )
{
    int magic;           //计算机“想”的数
    int guess;           //人猜的数
    int counter;         //记录人猜的次数
    char reply;          //用户键入的回答

    srand((unsigned)time(NULL));
    do
    {
        magic = rand () % 100 + 1;
        counter = 0;
        do
        {
            printf ("Please guess a magic number: ");
```

```
scanf ("%d", &guess);
counter++;
if (guess > magic)
    printf ("Wrong! Too high!\n");
else if (guess < magic)
    printf ("Wrong! Too low!\n");
else
    printf ("Right!\n");
} while (guess != magic && counter < 10); //猜不对且未超过10次时继续猜

printf ("counter = %d\n", counter);
printf ("Do you want to continue(Y/N or y/n)? ");
reply = getche ();
printf ("\n");
} while (reply == 'y' || reply == 'Y');

printf ("The game is over!\n");
}
```

程序运行结果:

```
Please guess a magic number: 50✓
Wrong! Too low!
Please guess a magic number: 80✓
Wrong! Too low!
Please guess a magic number: 90✓
Wrong! Too high!
Please guess a magic number: 85✓
Right!
counter = 4
Do you want to continue(Y/N or y/n)? n✓
The game is over!
```

2. 彩票选号

【参考答案】

```
#include <stdio.h>

void main( )
{
    int i, j, k, sum;

    sum = 0;
```

```
for (i = 1; i <= 36; i++)
{
    k = i;                //彩票起始号
    sum = i;
    for (j = 1; j <= 6; j++)
    {
        k = k + j;        //第j+1张彩票号
        if (k > 36)        //彩票号超过36退出该循环
            break;
        sum = sum + k;    //彩票号累计
    }

    if (sum != 105)        //和数不为105继续下一轮循环
        continue;

    k = i;
    for (j = 0; j <= 6; j++)    //显示彩票号码
        printf ("%d ", k = k + j);
    printf ("\n");
}
}
```

程序运行结果:

7 8 10 13 17 22 28

4.2.5 实验5 数组编程练习

1. 数字检查

【参考答案】

```
#include <stdio.h>
#define MAX 50

void main( )
{
    int a[MAX], b[MAX], c[4];
    int count, n, x, i, j, k, flag;

    //输入数据到数组a中, n为实际输入的个数
    printf ("Input number: ");
    for (n = 0, i = 0; i < MAX; i++)
```

```
{
    scanf ("%d", &x);
    if (x == -1)
        break;
    a[i] = x;
    n++;
}

count = 0;
for (i = 0; i < n; i++)
{
    c[0] = a[i] / 1000;           //求四位数的千位数字
    c[1] = a[i] % 1000 / 100;   //求四位数的百位数字
    c[2] = a[i] % 100 / 10;     //求四位数的十位数字
    c[3] = a[i] % 10;           //求四位数的个位数字
    for (j = 0; j < 4; j++)
    {
        if (c[j] % 2 == 0)       //如果各位上的数字均是偶数
            flag = 1;           //则置标志位flag为1
        else                     //否则置flag为0，退出循环
        {
            flag = 0;
            break;
        }
    }
    if (flag == 1)               //将满足条件的数存入数组b中，并统计满足条件的个数count
    {
        b[count] = a[i];
        count++;
    }
}

//把数组b中的数按从大到小的顺序排序
for (i = 0; i < count-1; i++)
    for(j = i + 1; j < count; j++)
        if (b[i] < b[j])
        {
            k = b[i];
            b[i] = b[j];
            b[j] = k;
        }

//输出结果
printf ("count = %d\n", count);
```

```
printf ("after sorted\n");  
for (i = 0; i < count; i++)  
    printf ("%d ", b[i]);  
printf ("\n");  
}
```

程序运行结果:

```
Input number: 1234 2008 4268 8856 8648 8024 7890 4862 6082 -1✓  
count = 6  
after sorted  
8856 8648 6082 4862 4268 2008
```

2. 成绩排名

【参考答案】

```
#include <stdio.h>  
#include <string.h>  
  
void main( )  
{  
    //多增加的最后一列用来表示每个学生的总分  
    //多增加的最后一行用来在排序时数据交换用  
    float score[6][5] = {{120, 130, 110, 280}, {110, 120, 105, 290},  
        {108, 128, 126, 278}, {112, 135, 122, 286}, {100, 120, 108, 276}};  
    char name[6][10] = {"张大明", "李小红", "王志强", "汪晓成", "李 丹", " "};  
    int i, j, k;  
    float mark;  
  
    //求每个学生的总分  
    for (i = 0; i < 5; i++)  
        for (j = 0; j < 4; j++)  
            score[i][4] += score[i][j];  
  
    //按总分进行从高到低排序  
    for (i = 0; i < 4; i++)  
    {  
        k = i;  
        for (j = i+1; j < 5; j++)  
            if (score[j][4] > score[k][4] || score[j][4] == score[k][4] &&  
                score[j][0]+score[j][1] > score[k][0]+score[k][1])  
                k = j;  
        if (k != i)
```

```

    {
        memcpy (score[5], score[i], 5 * sizeof(float));
        memcpy (score[i], score[k], 5 * sizeof(float));
        memcpy (score[k], score[5], 5 * sizeof(float));
        strcpy (name[5], name[i]);
        strcpy (name[i], name[k]);
        strcpy (name[k], name[5]);
    }
}

//显示排名
printf ("名 次 姓 名 语文 数学 英语 综合 总分\n");
printf ("-----\n");
k = 1;
mark = score[0][4];
for (i = 0; i < 5; i++)
{
    if (score[i][4] != mark)
    {
        k++;
        mark = score[i][4];
    }
    printf(" %d    %s  %.0f    %.0f    %.0f    %.0f    %.0f\n",
           k, name[i], score[i][0], score[i][1], score[i][2], score[i][3],
           score[i][4]);
}
}

```

程序运行结果:

名 次	姓 名	语文	数学	英语	综合	总分
1	汪晓成	112	135	122	286	655
2	张大明	120	130	110	280	640
2	王志强	108	128	126	278	640
3	李小红	110	120	105	290	625
4	李 丹	100	120	108	276	604

4.2.6 实验 6 函数编程练习

1. 去掉字符串前面的*号

【参考答案】

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



```
#include <conio.h>

void fun (char a[ ])
{
    int i, j;

    for (i = 0; a[i] == '*'; i++) //找到第一个非*号字符
        ;
    for (j = 0; a[i] != '\0'; i++, j++)
        a[j] = a[i];
    a[j] = '\0';
}

void main( )
{
    char s[81];
    printf ("Input a string: ");
    gets (s);
    fun (s);
    printf ("The string after deleted: ");
    puts (s);
}
```

程序运行结果:

```
Input a string: ****A*B*C*D****✓
The string after deleted: A*B*C*D****
```

2. 去掉字符串后面的*号

【参考答案】

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

void fun (char a[ ])
{
    int i, j;

    for (i = 0; a[i] != '\0'; i++)
        ;
    for (j = i-1; j >= 0 && a[j] == '*'; j--)
        ;
    a[j+1] = '\0';
}
```

```
void main( )
{
    char s[81];
    printf ("Input a string: ");
    gets (s);
    fun (s);
    printf ("The string after deleted: ");
    puts (s);
}
```

程序运行结果:

```
Input a string: ****A*B*C*D****✓
The string after deleted: ****A*B*C*D
```

3. 完全平方数

【参考答案】

```
#include <stdio.h>

int jsValue (int b[ ]);

void main( )
{
    int i, n, b[20];

    n = jsValue (b);
    printf ("count = %d\n", n);
    printf ("the number is: ");
    for (i = 0; i < n; i++)
        printf ("%d ", b[i]);
    printf ("\n");
}

int jsValue (int b[ ])
{
    int i, j, k = 0;
    int hun, ten, data;

    for (i = 100; i <= 999; i++)
    {
        j = 10;
```

```

while (j*j <= i)
{
    if (i == j*j)                //如果该数是完全平方数
    {
        hun = i / 100;           //求该数的百位数字
        data = i - hun*100;       //得到后两位数
        ten = data / 10;         //求该数的十位数字
        data = data - ten * 10;   //求该数的个位数字
        if (hun == ten || hun == data || ten == data) //有两位数字相同
        {
            b[k] = i;             //则把该数存入数组b中
            k++;                  //统计满足条件的数的个数
        }
    }
    j++;
} //while
} //for
return k;                        //返回满足该条件的整数的个数
}

```

程序运行结果:

```

count = 9
the number is: 100 121 144 225 400 441 484 676 900

```

4.2.7 实验7 指针编程练习

1. 字符串合并

【参考答案】

```

#include <stdio.h>
#include <string.h>

void stradd (char *s, char *t, int flag);

void main( )
{
    char s[80], t[80];
    int flag;

    printf ("Input flag: ");
    scanf ("%d", &flag );
    fflush (stdin);

```

```
printf ("Input string s: ");
gets (s);
fflush (stdin);
printf ("Input string t: ");
gets (t);
stradd (s, t, flag);
if (flag)
    printf ("%s\n", s);
else
    printf ("%s\n", t );
}

void stradd (char *s, char *t, int flag)
{
    char *ptr, *ptr1;

    if (flag)
    {
        ptr = s + strlen (s);
        ptr1 = t;
    }
    else
    {
        ptr = t + strlen (t);
        ptr1 = s;
    }
    for ( ; *ptr1 != '\0'; ptr1++, ptr++)
        *ptr = *ptr1;
    *ptr = '\0';
}
```

程序运行结果:

```
Input flag: 1✓
Input string s: 12345✓
Input string t: 67890✓
1234567890
```

2. 数据交换

【参考答案】

```
#include <stdio.h>

void exchange (int *pa, int *pb, int *pc);
```

```
void main( )
{
    int a, b, c;

    printf ("Input a,b,c: ");
    scanf ("%d%d%d", &a, &b, &c);
    exchange (&a, &b, &c);
    printf ("After exchange: %d %d %d\n", a, b, c);
}

void exchange (int *pa, int *pb, int *pc)
{
    int t, *pmax, *pmin;

    if (*pa < *pb)
    {
        pmax = pb;
        pmin = pa;
    }
    else
    {
        pmax = pa;
        pmin = pb;
    }

    if (*pmax < *pc)
        pmax = pc;
    if (*pmin > *pc)
        pmin = pc;

    t = *pmax;
    *pmax = *pmin;
    *pmin = t;
}
```

程序运行结果:

```
Input a,b,c: 5 8 9✓
After exchange: 9 8 5
```

4.2.8 实验8 数组、指针和函数综合编程练习

【参考答案】 打印最高分和学号 程序1

```
#include <stdio.h>

#define ARR_SIZE 40
```

```
int FindMax (int score[ ], long num[ ], int n, long *pMaxNum);

void main( )
{
    int score[ARR_SIZE], maxScore, n, i;
    long num[ARR_SIZE], maxNum;

    printf ("Please enter total number: ");
    scanf ("%d", &n);                                //输入学生人数
    printf ("Please enter the number and score:\n");
    for (i = 0; i < n; i++)
        scanf ("%ld%d", &num[i], &score[i]);
    maxScore = FindMax (score, num, n, &maxNum);        //计算最高分及学生学号
    printf ("maxScore = %d, maxNum = %ld\n", maxScore, maxNum);
}

//计算最高分及最高分学生的学号
int FindMax (int score[ ], long num[ ], int n, long *pMaxNum)
{
    int i, maxScore;

    maxScore = score[0];
    *pMaxNum = num[0];                                //假设score[0]为最高分
    for (i = 1; i < n; i++)
        if (score[i] > maxScore)
        {
            maxScore = score[i];                      //记录最高分
            *pMaxNum = num[i];                          //记录最高分学生的学号
        }
    return (maxScore);                                //返回最高分
}
```

程序运行结果:

```
Please enter total number: 5✓
Please enter the number and score:
2005010 90✓
2005011 89✓
2005012 86✓
2005013 95✓
2005014 88✓
Maxscore = 95, maxNum = 2005013
```

【参考答案】 打印最高分和学号 程序2

```
#include <stdio.h>

#define CLASS 3
#define STU 4

int FindMax (int score[ ][STU], int m, int *pRow, int *pCol);

void main( )
{
    int score[CLASS][STU], i, j, maxScore, row, col;

    printf ("Please enter score:\n");
    for (i = 0; i < CLASS; i++)
        for (j = 0; j < STU; j++)
            scanf ("%d", &score[i][j]);           //输入学生成绩

    maxScore = FindMax (score, CLASS, &row, &col);    //计算最高分及其学生所
                                                    //在班号和学号
    printf ("maxScore = %d, class = %d, number = %d\n", maxScore, row+1,
        col+1);
}

//计算任意m行STU列二维数组中元素的最大值, 并指出其所在行列下标值
int FindMax (int score[ ][STU], int m, int *pRow, int *pCol)
{
    int i, j, maxScore;

    maxScore = score[0][0];           //置初值, 假设第一个元素值最大
    *pRow = 0;
    *pCol = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < STU; j++)
            if (score[i][j] > maxScore)
            {
                maxScore = score[i][j];    //记录当前最大值
                *pRow = i;                  //记录行下标
                *pCol = j;                  //记录列下标
            }
    return (maxScore);                  //返回最大值
}
```

程序运行结果:

```
Please enter score:
90 89 87 78✓
78 98 90 94✓
80 87 89 92✓
maxScore = 98, class = 2, number = 2
```

【参考答案】 打印最高分和学号 程序 3

```
#include <stdio.h>

#define CLASS 3
#define STU 4

int FindMax (int *p, int m, int n, int *pRow, int *pCol);

void main( )
{
    int score[CLASS][STU], i, j, maxScore, row, col;

    printf ("Please enter score:\n");
    for (i = 0; i < CLASS; i++)
        for (j = 0; j < STU; j++)
            scanf ("%d", &score[i][j]);           //输入学生成绩

    //计算最高分及其学生所在班号和学号
    maxScore = FindMax (*score, CLASS, STU, &row, &col);
    printf ("maxScore = %d, class = %d, number = %d\n", maxScore, row+1,
            col+1);
}

//计算任意m行STU列二维数组中元素的最大值, 并指出其所在行列下标值
int FindMax (int *p, int m, int n, int *pRow, int *pCol)
{
    int i, j, maxScore;

    maxScore = p[0];           //置初值, 假设第一个元素值最大
    *pRow = 0;
    *pCol = 0;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (p[i*n+j] > maxScore)
```



```
        {
            maxScore = p[i*n+j];           //记录当前最大值
            *pRow = i;                     //记录行下标
            *pCol = j;                     //记录列下标
        }
    return (maxScore);                     //返回最大值
}
```

【参考答案】 打印最高分和学号 程序4

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

int FindMax (int *p, int m, int n, int *pRow, int *pCol);

void main( )
{
    int *pScore, i, j, m, n, maxScore, row, col;

    printf ("Please enter array size m, n: ");
    scanf ("%d%d", &m, &n);                //输入班级数m和学生人数n

    //动态申请m*n个sizeof(int)字节的存储空间
    pScore = (int *) malloc (m * n * sizeof(int));
    if (pScore == NULL)
    {
        printf ("No enough memory!\n");
        exit (0);
    }

    printf ("Please enter the score:\n");
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            scanf ("%d", &pScore[i*n+j]);    //输入学生成绩

    maxScore = FindMax (pScore, m, n, &row, &col);    //计算最高分及其学生所
                                                    //在班号和学号
    printf ("maxScore = %d, class = %d, number = %d\n", maxScore, row+1,
            col+1);
    free (pScore);                            //释放内存
}

//计算任意m行n列二维数组中元素的最大值,并指出其所在行列下标值
int FindMax (int *p, int m, int n, int *pRow, int *pCol)
{
    int i, j, maxScore;
```

```
maxScore = p[0];                //置初值, 假设第一个元素值最大
*pRow = 0;
*pCol = 0;
for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
        if (p[i*n+j] > maxScore)
        {
            maxScore = p[i*n+j];    //记录当前最大值
            *pRow = i;              //记录行下标
            *pCol = j;              //记录列下标
        }
return (maxScore);              //返回最大值
}
```

4.2.9 实验 9 复杂数据类型编程练习

1. 计算天数

【参考答案】

```
#include <stdio.h>

struct date
{
    int year, month, day;
};

static int day_tab[][13] = { {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
                               30, 31}, {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31,
                               30, 31, 30, 31} };

int Day_Of_Year (struct date nowdate);

void main( )
{
    struct date nowdate;
    int days;
    printf ("Input date(YYYY-MM-DD): ");
    scanf ("%4d-%2d-%2d", &nowdate.year, &nowdate.month, &nowdate.day);
    days = Day_Of_Year (nowdate);
    printf ("The year passed days = %d\n", days);
}
```

```
int Day_Of_Year (struct date nowdate)
{
    int i, leap, days;

    //是否闰年
    leap = nowdate.year % 4 == 0 && nowdate.year % 100 != 0 || nowdate.year
        % 400 == 0;

    days = nowdate.day;
    for (i = 0; i < nowdate.month; i++)
        days += day_tab[leap][i];
    return (days);
}
```

程序运行结果:

```
Input date(YYYY-MM-DD): 2000-03-05✓
The year passed days = 65
```

2. 学生成绩

【参考答案】

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

struct stu_msg
{
    char no[11];
    char name[16];
    int score[3];
    int totscore;
    float avscore;
};

void Sort (struct stu_msg *pstu, int n);

void main( )
{
    struct stu_msg *pstu;
    int n, i;

    printf ("Input the number of student: ");
    scanf ("%d", &n);

    //动态内存分配
    pstu = (struct stu_msg *) malloc (n * sizeof(struct stu_msg));
```

```
if (pstu == NULL)
{
    printf ("No enough memory!");
    exit (0);
}

//输入学生信息
printf ("Input the message of students:\n");
for (i = 0; i < n; i++)
{
    scanf ("%s%s%d%d%d", pstu[i].no, pstu[i].name,
            &pstu[i].score[0], &pstu[i].score[1], &pstu[i].score[2]);
    pstu[i].totalscore = pstu[i].score[0]+pstu[i].score[1]+pstu[i].score[2];
    pstu[i].avescore = pstu[i].totalscore / 3.0;
}

Sort (pstu, n); //按总分从高到低排序

printf ("\nNo      Name      TotalScore  AverageScore\n");
for (i = 0; i < n; i++)
    printf ("%11s %-16s %5d   %8.2f\n", pstu[i].no, pstu[i].name,
            pstu[i].totalscore, pstu[i].avescore);

free (pstu);
}

//按总分从高到低排序
void Sort (struct stu_msg *pstu, int n)
{
    int i, j, k;
    struct stu_msg stu;

    for (i = 0; i < n-1; i++)
    {
        k = i;
        for (j = i+1; j < n; j++)
            if (pstu[j].totalscore > pstu[k].totalscore)
                k = j;
        if (k != i)
        {
            stu = pstu[i];
            pstu[i] = pstu[k];
            pstu[k] = stu;
        }
    }
}
```

4.2.10 实验10 文件编程练习

1. 选票统计问题

【参考答案】

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

#define Max 100

void main( )
{
    FILE *fp;
    int i, n, count[10];
    char ticket[13];

    for (i = 0; i < 10; i++)
        count[i] = 0;

    //从文件中读取选票信息, 并统计结果
    if ((fp = fopen ("IN.DAT", "r")) == NULL)
    {
        printf ("Cann't open file IN.DAT!");
        exit (0);
    }
    while (fgets (ticket, 13, fp) != NULL)
    {
        for (n = 0, i = 0; ticket[i] != '\0'; i++)
            if (ticket[i] == '1')
                n++;
        if (n <= 5)
            continue;
        for (i = 0; ticket[i] != '\0'; i++)
            if (ticket[i] == '1')
                count[i]++;
    }
    fclose (fp);

    //将统计结果写入文件OUT.DAT中, 并在屏幕上显示
    if ((fp = fopen ("OUT.DAT", "w")) == NULL)
    {
        printf ("Cann't create file OUT.DAT!");
        exit (0);
    }
    for (i = 0; i < 10; i++)
```

```
{
    fprintf (fp, "%d\n", count[i]);
    printf ("第%d个人的选票数 = %d\n", i+1, count[i]);
}
fclose (fp);
}
```

2. 文件中大小写英文字符的转换

【参考答案】

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

void main (int argc, char *argv[ ])
{
    FILE *fpr, *fpw;
    char ch;

    if (argc != 2)                                //命令行参数有误
    {
        printf ("Useage: 执行文件名 待转换文件名 \n");
        exit (0);
    }

    fpr = fopen (argv[1], "rb");                    //以读的方式打开文件
    if (fpr == NULL)                                //打开文件失败
    {
        printf ("file: %s not found!\n", argv[1]);
        exit (0);
    }

    fpw = fopen (argv[1], "rb+");                   //以读写的方式打开文件
    if (fpw == NULL)                                //打开文件失败
    {
        printf ("file: %s not found!\n", argv[1]);
        exit (0);
    }

    while ((ch = fgetc (fpr)) != EOF)
    {
        if (ch >= 'a' && ch <= 'z')
            ch = ch - 'a' + 'A';
        else if (ch >= 'A' && ch <= 'Z')
            ch = ch - 'A' + 'a';
        fputc (ch, fpw);
    }
}
```

```
    }  
  
    fclose(fpr);           //关闭文件  
    fclose(fpw);           //关闭文件  
}
```

4.3 期末上机实验考试题目

1. 编制万年历。程序名为 Calendar.cpp, 要求带命令行参数, 即四位年份。如程序执行: Calendar 2003, 即可显示 2003 年 1 月份到 12 月份的日历。输出格式为:

```
=====JANUARY=====  
Sun      Mon      Tues     Wed      Thur     Fri      Sat  
          1        2        3        4  
5         6        7        8        9       10      11  
12        13       14       15       16      17      18  
19        20       21       22       23      24      25  
26        27       28       29       30      31
```

[提示: 要计算该年份 1 月 1 日是星期几, 然后顺推出该年中每一天的星期数。注意: 要判断是否是闰年。]

2. 对数组 A 中的 N ($0 < N < 100$) 个整数从小到大进行连续编号, 输出各个元素的编号。要求不能改变数组 A 中元素的顺序, 且相同的整数要具有相同的编号。例如数组是: A=(5, 3, 4, 7, 3, 5, 6), 则输出为: (3, 1, 2, 5, 1, 3, 4)。

[提示: 定义一个 2 行 100 列的数组, 第一行存放输入的数据, 第二行存放每个数的顺序编号 (初始值均为 0)。每次从第一行数据中查找其对应编号为 0 的最小数, 然后将其编号写入到第 2 行对应位置上。]

3. 使用数组完成两个超长 (长度小于 100) 正整数的乘法。

[提示: 定义三个整型数组, 一个存放被乘数, 一个存放乘数, 另外一个存放积。]

4. 现将不超过 2000 的所有素数从小到大排成第一行, 第二行上的每个数都等于它“右肩”上的素数与“左肩”上的素数之差。请编程求出: 第二行数中是否存在这样的若干个连续的整数, 它们的和恰好是 1898? 假如存在的话, 又有几种这样的情况?

```
第一行: 2  3  5  7  11  13  17 ... 1979  1987  1993  
第二行:  1  2  2  4  2  4  ...   8    6
```

[提示：用一数组存放 2000 以内的所有素数，再用一数组存放相邻素数的差值，然后对该数组从第一个元素到最后一个元素分别计算连续的数之和是否为 1898。]

5. 八皇后问题：在一个 8×8 的国际象棋盘，有 8 个皇后，每个皇后占一格；要求棋盘上放上 8 个皇后时不会出现相互“攻击”的现象，即不能有两个皇后在同一行、列或对角线上。问共有多少种不同的方法。

[提示：采用试探法求解。用 I、J 表示行、列坐标。开始棋盘为空，对于第 1 个皇后先占用第一行即 $I=1$ ，先试探它占用第一列 $J=1$ 位置，则它所在的行、列和斜线方向都不可再放其他皇后了，用线将它们划掉。第 2 个皇后不能放在 $J=1, 2$ 的位置，试 $J=3$ 。第 2 个皇后占用 $[2, 3]$ 后，它的行列和斜线方向也不可再放其他皇后。第 3 个皇后不能放在 $J=1, 2, 3, 4$ 的位置，试 $J=5$ 。第 4 个皇后可以试位置 $[4, 2]$ ，第 5 个皇后试位置 $[5, 4]$ 。第 6 个皇后已经没有可放的位置（棋盘上所有格子都已占满），说明前面所放位置不对。退回到前一个皇后 5，释放它原来占用的位置 $[5, 4]$ ，改试空位置 $[5, 8]$ 。然后再前进到第 6 个皇后，此时仍无位置可放，退回到第 5 个皇后，它已没有其他位置可选择。进一步退回到第 4 个皇后释放位置 $[4, 2]$ 改试位置 $[4, 7]$ ，再前进到第 5 个皇后进行试探，如此继续，直到所有 8 个皇后都选择一个合适的位置，即可打印一个方案。然后从第 8 个皇后开始，改试其他空位置，若没有可改选的空位置，则退回到第 7 个皇后改试其他位置，若也没有空位置可改，继续退，直到有另外的空位置可选的皇后。将它原来占用的位置释放，改占其他新位置，然后前进到下一个皇后进行试探，直到所有 8 个皇后都找到合适位置，又求出一个解，打印输出新方案。按此方法可得到 92 个方案。]

6. 利用字符串指针编写一程序求所有不超过 200 的 N 值，N 的平方是具有对称性质的回文数。

[提示：将 N 的平方这个数可通过 `sprintf` 或 `ltoa` 函数转换成字符串，然后通过设置头、尾指针对字符串进行比较。]

7. 编写一函数 `char *Replace (char *str, char *substr, char *newstr)`，该函数的功能为：将 str 字符串中所包含的所有子串 substr 用新串 newstr 替代，并返回替换后的字符串。如：`Replace ("qweabcdefabcgh", "abc", "123")`，则执行后返回的字符串为：`"qwe123def123gh"`。

[提示：先查找子串位置，然后再替换，字符指针应移到子串后的第一个字符的位置，再进行后面的字符串替换。]

8. 编一程序，输入下列学生成绩表中的数据（具体人数由键盘输入），并用链表的形式存放。然后统计出每位学生的总分，并按总分从高到低的顺序显示学生成绩信息。

Name	Maths	English	C	Total
ZhangSan	90.5	80.0	88.5	
LiHao	96.0	89.5	95.0	
WangFei	90.0	88.5	80.0	
LiMing	70.0	89.0	92.0	
LiuQiang	90.0	85.5	86.0	

[提示：每输入一个学生成绩，就计算其总分，然后按序插入到链表中。]

9. 设有学生信息如下：学号（长整型）、姓名（字符型数组）、出生日期（含年、月、日，均为整型）。从键盘上输入 10 个学生信息，写入硬盘的某个目录下名为 student.dat 的二进制数据文件中。然后再读取该文件中的前 5 名学生的信息，输出每个学生的学号、姓名及年龄（当前计算机的年份减去出生年份）。

[提示：定义一结构体数组，待信息输入完后，利用 fwrite 函数写入到文件中，然后利用 fread 函数读出前 5 名学生的信息。计算年龄之前要取系统年份。]

10. 编一文件复制程序。文件名为 MyCopy.cpp，要求源文件名和目的文件名通过命令行参数给出，目的文件名在前，源文件名在后，其功能是将源文件的内容添加到目的文件的后面。文件名可带其路径。如：Mycopy c:\bak\prg1.txt d:\bak\prg2.txt，执行完后将把文件 prg2.txt 的内容追加到文件 prg1.txt 的后面。

[提示：目的文件以追加的方式打开，源文件以读方式打开。]

11. 编一程序，它读入一行正文，统计在正文中出现的各个字的次数，并按字典顺序显示结果。例如，如果输入的正文为：how do you do，则输出结果为：

```
do      2
how     1
you     1
```

[提示：用链表的方式来存储每个单词和出现的次数。从文本中每次分离出一个字，就遍历链表是否存在该字，如果存在，计数增 1，否则按序添加到链表中。]

12. 编一程序，要求对某文本文件以行为单位对字符按从小到大的顺序进行排序，排序后的结果仍按行重新写入到该文件中。

例如，原文：dAe, BfC

CCbbAA

结果：, ABCdef

AACCbb

原始数据文件存放的格式是：每行的宽度均小于 80 个字符，含标点符号和空格。
[提示：先以读的方式打开该文本文件，再将该文件中的信息读入到字符串数组中，

然后对字符串数组按行从小到大进行排序，排序后以写方式打开文件，将字符串数组的内容写入到文件中。]

13. 用递归方法计算两个整数的最大公约数。

[提示：假设两个整数分别为 x 和 y ，如果 $x > y$ ，则 x 和 y 的最大公约数与 $x-y$ 和 y 的公约数相同；如果 $x < y$ ，则 x 和 y 的最大公约数与 x 和 $y-x$ 的公约数相同；如果 $x=y$ ，则 x 或 y 就是最大公约数。]

14. 编一程序，可以将英语规则名词由单数变成复数。已知规则如下：

- (1) 以辅音字母 y 结尾，则将 y 改为 i ，再加 es ；
- (2) 以 s , x , ch , sh 结尾，则加 es ；
- (3) 以元音 o 结尾，则加 es ；
- (4) 其他情况直接加 s 。

要求从键盘输入英语规则名词，屏幕输出该名词的复数形式。

15. 求数字的乘积根。

定义：正整数中非 0 数字的乘积称为该数数字成绩。如 1620 的数字成绩为 $1*6*2=12$ ，12 的数字乘积为 $1*2=2$ 。

定义：正整数的数字乘积根为反复取该整数的数字乘积，直到最后的数字乘积为一位数字，这个一位数字就叫该正整数的数字乘积根。

例如：1620 的数字乘积为 $1*6*2=12$ ，12 的数字乘积为 $1*2=2$ ，因此 2 为 1620 的数字乘积根。

编程要求：统计 10000 以内，其数字乘积根分别为 1~9 的正整数的个数。

[提示：在计算数字乘积时用 `itoa` 函数将该数转换成字符串，然后再计算数字乘积。统计结果可以定义 10 个元素的整型数组来存放。]