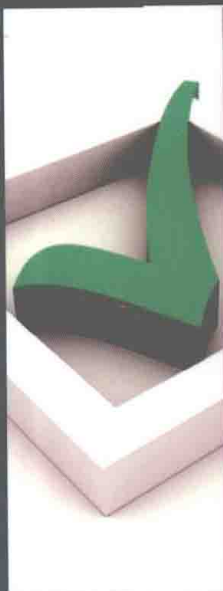


21世纪高等院校计算机专业规划教材



C语言程序设计 上机指导与习题解答

(第三版)

夏宽理 赵子正 编著

21世纪高等院校计算机专业规划教材

- 计算机导论 (第二版)
- 计算机科学与技术导论
- 计算机英语 (第二版)
- 数字逻辑 (第二版)
- 数字逻辑习题解答与实验指导
- C语言程序设计 (第三版)
- C语言程序设计上机指导与习题解答 (第三版)
- C/C++程序设计
- C/C++程序设计学习及应试指导
- Java 2面向对象程序设计
- Java网络程序设计
- 计算机网络
- 计算机网络习题解答与实验指导
- Oracle数据库开发——SQL&PL/SQL
- 网络与信息安全 (第二版)
- 操作系统 (第三版)
- 操作系统习题解答与实验指导 (第三版)
- 数据结构 (第三版)
- 数据结构习题解答与实验指导 (第三版)
- 计算机组成原理 (第二版)
- 计算机组成原理学习指导与实验教程
- 软件工程
- 实用软件工程教程
- 离散数学
- 编译原理 (第二版)
- 数据库原理及应用
- Linux网络服务器应用教程

责任编辑：孟欣 杜鹃 封面设计：付巍 封面制作：白雪



中国铁道出版社 教材研究开发中心
地址：北京市西城区右安门西街8号
邮编：100054
网址：<http://www.51eds.com>
读者热线：400-668-0820

ISBN 978-7-113-16965-7

定价：24.00元

ISBN 978-7-113-16965-7



9 787113 169657 >

21 世纪高等院校计算机专业规划教材

C 语言程序设计上机指导 与习题解答（第三版）

夏宽理 赵子正 编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书是主教材《C 语言程序设计（第三版）》（夏宽理、赵子正编著）的配套教材，主要包括四部分内容。第一部分为上机指导，根据主教材章节安排了 10 个实验，共 67 个实验小题，涉及的内容有基本数据类型、结构化程序设计、数组、函数、指针和引用、结构和链表、文件等，在综合实验中给出的是 3 个非常有趣的游戏程序。第一部分对前 8 个实验中的习题给出了较为详细的分析与解题思路，同时还提供了参考程序及运行后的结果，能让读者在实验环节中进一步分析、领会和拓展所学内容。第二部分为习题与分析解答，此部分对读者准确理解 C 语言的基本功能有很大的帮助。第三部分为 Visual C++ 6.0 快速入门，简述 C 源程序的编写、编译、连接、运行以及程序调试的方法与技巧。第四部分为 Visual Studio 2010 快速入门，介绍在 Windows 7 及以上操作系统环境下编写、编译、运行与调试 C 源程序的方法与技巧。

本书适合作为高等院校各类专业计算机程序设计的配套实验教材，也可作为参加“计算机软件著作权资格和水平考试”人员学习程序设计的配套实验教材，还可作为各类进修班、培训班讲授程序设计的参考书。

图书在版编目（CIP）数据

C 语言程序设计上机指导与习题解答/夏宽理，赵子正编著. —3 版. —北京：中国铁道出版社，2013.8
21 世纪高等院校计算机专业规划教材
ISBN 978-7-113-16965-7

I. ①C… II. ①夏…②赵… III. ①
C 语言—程序设计—高等学校—教学参考资料 IV.
①TP312

中国版本图书馆 CIP 数据核字（2013）第 155832 号

书 名：C 语言程序设计上机指导与习题解答（第三版）
作 者：夏宽理 赵子正 编著

策划编辑：孟 欣

读者热线电话：（010）63583215

责任编辑：孟 欣 杜 鹃

封面设计：付 巍

封面制作：白 雪

责任印制：李 佳

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）

网 址：<http://www.51eds.com>

印 刷：北京鑫正大印刷有限公司

版 次：2006 年 6 月第 1 版 2011 年 8 月第 2 版 2013 年 8 月第 3 版 2013 年 8 月第 1 次印刷

开 本：787mm×1092mm 1/16 印张：11 字数：257 千

印 数：3 000 册

书 号：ISBN 978-7-113-16965-7

定 价：24.00 元

版权所有 侵权必究

凡购买铁道版图书，如有印制质量问题，请与本社教材图书营销部联系调换。电话：（010）63550836

打击盗版举报电话：（010）63549504

第三版前言

FOREWORD >>>

目前, C 语言仍然是国内、外广泛使用的计算机程序设计语言, 这主要得益于它是一种具有功能丰富、表达能力强、使用灵活方便、便于程序开发、可移植性好等特点的程序设计语言。

本书是主教材《C 语言程序设计(第三版)》(夏宽理、赵子正编著)的配套教材, 主要包括以下四个部分。

第一部分为上机指导。此部分包含了 10 个实验, 前 9 个实验分别对应主教材第 1~9 章的内容。在每个实验中安排了多个实验小题目, 目的是给读者提供更多的学习、思考和实践机会。通过编写、编译、连接与运行, 读者可逐步做到发现问题, 修改程序后再编译运行, 直至程序正常运行, 并给出正确的解答。实验 10 为综合实验, 便于读者综合应用所学的程序设计知识, 独立完成相对比较完整的课题, 以进一步提高程序设计能力。

为了帮助读者尽快掌握实验环节, 在第一部分中, 前 8 个实验中的实验题目中给出了较为详细的分析与解题思路, 同时还提供了参考程序以及运行后的结果。在参考程序中, 尽可能地给出较多的注释, 以便于读者理解与阅读。当然, 本书中提供的参考程序并非是唯一正确答案, 也不一定是最佳方案, 而仅仅是给读者参考和启发, 其目的是使读者可以举一反三, 掌握编程技巧, 以便以后编写出更好的程序。在参考程序之后附有程序运行后的结果, 供读者对照分析。在每一个实验内容的最后还提出了一些思考题, 让读者在实验环节中进一步分析、领会和拓展所学内容, 以达到巩固知识点、提高编程能力的目的。

第二部分为练习题与分析解答。此部分共精选了选择题和填空题各 100 道供读者学习参考, 并对每道题给出了详细的分析与解答, 目的是使读者进一步准确掌握 C 语言的基本概念。此部分对读者正确理解 C 语言的基本功能有很大的帮助。

第三部分为 Visual C++ 6.0 快速入门。此内容简明扼要地介绍了目前非常流行的 Visual C++ 6.0 的上机环境, 使完全不熟悉 Visual C++ 6.0 的读者能快速上手。从如何新建与编写一个 C 源程序开始, 到源程序的编译、连接和运行这一完整的上机过程都在这一部分有所介绍。在此部分的最后, 还介绍了如何查错、修正错误等调试手段, 相信这一部分内容对读者上机实际操作会有一定的帮助。

第四部分为 Visual Studio 2010 快速入门。由于在新的操作系统环境下不能安装和使用 Visual C++ 6.0, 这一部分内容特意为在 Windows 7 及以上操作系统平台下使用者打造, 主要介绍如何新建一个项目与 C 源程序、源程序的编译、运行和调试这一完整的过程。

本书由夏宽理和赵子正编著。在本书编写过程中, 王春森、杭必政、李应华、金旭东、王德新、陈学青、张向东、王宗彩等对实验内容的选择与审定给予了很大的帮助, 提出了一些非常宝贵的意见与建议, 在此表示衷心的感谢。

由于时间仓促、编者水平有限, 本书难免有疏漏和不足之处, 敬请读者不吝指正。

编 者

2013 年 6 月

第二版前言

FOREWORD >>>

目前, C 语言仍然是国内外广泛使用的计算机程序设计语言, 这主要得益于它是一种具有功能丰富、表达能力强、使用灵活方便、便于程序开发、可移植性好等特点的程序设计语言。

本书是《C 语言程序设计(第二版)》(夏宽理、赵子正编著)的配套教材, 主要包括以下 3 个部分。

第一部分为上机指导。与教材再版时删除第 10 章内容相对应, 本书再版时也删除了原有实验 10 的全部内容。现在, 此部分包含的 10 个实验, 前 9 个实验分别对应教材第 1~9 章的内容。在每个实验中安排了多道实验小题, 目的是给读者提供更多的学习、思考和实践机会。通过编写、编译、连接与运行, 读者可逐步做到发现问题, 修改程序后再编译运行, 直至程序正常运行, 并给出正确的解答。实验 10 为综合作业, 便于读者综合应用所学程序设计的知识, 独立完成相对比较完整的课题, 以进一步提高程序设计能力。

为了帮助读者尽快掌握实验环节, 在第一部分中, 前 8 个实验中的实验小题给出了较为详细的分析与解题思路, 同时还提供了参考程序以及运行后的结果。在参考程序中, 我们尽可能地给出较多的注释, 以便于读者理解与阅读。当然, 我们提供的参考程序并非是唯一正确答案, 也不一定是最佳方案, 而仅仅是给读者参考和启发, 其目的是使读者可以举一反三, 掌握编程技巧, 以便以后编出更好的程序。在参考程序之后附有程序运行后的结果, 供读者对照分析。在每一个习题的最后我们还提出了一些思考题, 能让读者在实验环节中进一步分析、领会和拓展所学内容, 以达到巩固知识点、提高编程能力的目的。

第二部分为习题与分析解答。此部分共精选了选择题和填空题各 100 题供读者学习参考, 并对每个题给出了详细的分析与解答, 目的是使读者进一步准确掌握 C 语言的基本概念。此部分对读者正确理解 C 语言的基本功能有很大的帮助。

第三部分为 Visual C++ 6.0 快速入门。此内容简明扼要地介绍了目前非常流行的 Visual C++ 6.0 的上机环境, 使完全不熟悉 Visual C++ 6.0 的读者能快速上手。从如何新建与编写一个 C 源程序开始, 到源程序的编译、连接和运行这一完整的上机过程都在这一部分有所介绍。在此部分的最后, 还介绍了如何查错、修正错误等调试手段, 相信这一部分内容对读者上机实际操作会有一定的帮助。

本书由夏宽理和赵子正编著。在本书编写过程中, 王春森、杭必政、李应华、金旭东、王德新、陈学青、张向东、王宗彩等对实验内容的选择与审定给予了很大的帮助, 提出了一些非常宝贵的意见与建议, 在此表示衷心的感谢。

由于时间仓促、编者水平有限, 本书难免有疏漏和不足之处, 敬请读者不吝指正。

编 者

2009 年 1 月

第一版前言

FOREWORD

C 语言仍然是目前国内外广泛使用的计算机程序设计语言,这主要得益于 C 语言是一种具有功能丰富、表达能力强、使用灵活方便、便于程序开发、可移植性好等特点的程序设计语言。

本书是《C 语言程序设计》的配套教材,主要包括以下 3 个部分。

第一部分为上机指导。此部分包含了 11 个实验,每个实验分别对应教材一章的内容。在每章的实验内容中安排了多道习题,目的是提供给学生更多的学习、思考和实践机会。通过编写、编译、连接与运行,读者可逐步做到发现问题,修改程序后再编译运行,直至程序正常运行,并给出正确的解答。实验十一为综合作业,便于读者综合应用所学程序设计的知识,独立完成相对比较完整的课题,以进一步提高程序设计能力。

为了帮助读者尽快掌握实验环节,在第一部分中,对前 8 个实验中的每个习题都给出了较为详细的分析与解题思路,同时还提供了参考程序以及运行后的结果。在参考程序中,我们尽可能地给出较多的注释,便于读者理解与阅读。当然,我们提供的参考程序并非是唯一正确答案,也不一定是最佳方案,而仅仅是给读者参考和启发,其目的是使读者可以举一反三,掌握编程技巧,以便以后编出更好的程序。在参考程序之后附有程序运行后的结果,供读者对照分析。在每一个习题的最后我们还提出了一些思考题,能让读者在实验环节中进一步分析、领会和拓展,以达到巩固知识点、提高编程能力的目的。

第二部分为习题与分析解答。此部分共精选了选择题和填空题各 100 题供读者学习参考,并对每个题给出了详细的分析与解答,目的是使读者进一步准确掌握 C 语言的基本概念。此部分对读者正确理解 C 语言的基本功能有很大的帮助。

第三部分为 Visual C++ 6.0 快速入门。这一部分简明扼要地介绍了目前非常流行的 Visual C++ 6.0 的上机环境,使完全不熟悉 Visual C++ 6.0 的读者能快速上手。从如何新建与编写一个 C 源程序开始,到源程序的编译、连接和运行这一完整的上机过程都在这一部分有所介绍。在这一部分的最后,还介绍了如何查错、修正错误等调试手段,相信这一部分内容对读者上机实际操作会有一定的帮助。

本书由夏宽理和赵子正两位老师编著。在本书编写过程中,王春森、李应华、金旭东、王德新、陈学青、张向东、王宗彩等老师对实验内容的选择与审定给予了很大的帮助,提出了一些非常宝贵的意见与建议,在此表示衷心的感谢。由于时间仓促、编者水平有限,本书难免有疏漏和不足之处,敬请读者不吝指正。

编 者

2006 年 4 月

目 录

CONTENTS >>>

第一部分 上机指导

实验 1 程序设计基础	1
实验 2 基本数据类型及其运算	5
实验 3 结构化程序设计	8
实验 4 数组	18
实验 5 函数	29
实验 6 指针和引用	37
实验 7 结构和链表	46
实验 8 数据文件处理技术	58
实验 9 算法设计技术基础	64
实验 10 综合实验	76

第二部分 练习题与分析解答

一、选择题	98
二、填空题	124

第三部分 Visual C++ 6.0 快速入门

一、启动 Visual C++ 6.0 集成环境	149
二、新建一个 C 源程序	149
三、打开一个已有的 C 源程序	151
四、源程序编译	152
五、程序连接	153
六、程序执行	154
七、修正编译指出的程序错误	154
八、程序调试	156

第四部分 Visual Studio 2010 快速入门

一、启动 Visual Studio 2010 中文版	158
二、新建一个项目及 C 源程序	158
三、打开一个项目及项目中的 C 源文件	162
四、源程序仅编译不运行	163
五、源程序编译与运行	164
六、程序调试	164

第一部分 上机指导

实验 1 | 程序设计基础

一、实验目的

1. 了解所用计算机系统的基本操作方法，学会独立使用该系统。
2. 熟悉 C 程序的基本组成。
3. 熟悉在 Visual C++6.0 或 Visual Studio 2010 中文版环境中，C 程序的编写、编译、连接、运行等操作方法。
4. 通过运行简单的 C 程序，初步了解 C 程序的特点。

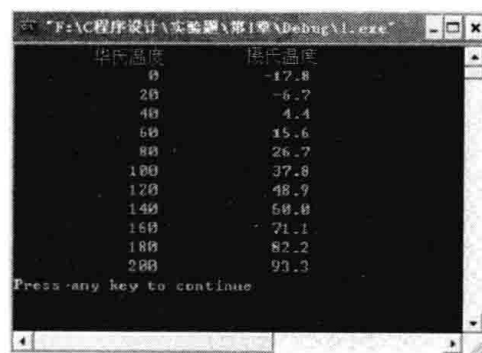
二、实验内容

1. 输入并运行主教材（以下简称教材）中例 1.3 的程序。

教材中例 1.3 利用公式 $C=(5/9)(F-32)$ 输出华氏温度与摄氏温度对照表，设华氏温度分别取 0, 20, ..., 200。

【参考程序】

```
#include<stdio.h>
void main()
{ double f,c;          /*变量定义*/
  int lower,upper,step;
  lower=0;upper=200;step=20;f=lower;
  printf("\t 华氏温度\t 摄氏温度\n");
  while(f<=upper)
  { /*循环计算*/
    c=5.0/9.0*(f-32.0);
    printf("\t%8.0f\t%8.1f\n",f,c);
    f=f+step;
  }
}
```



程序运行结果如图 1-1-1 所示。

图 1-1-1 例 1.3 的运行结果

注意：图 1-1-1 标题中“F:\C 程序设计\实验题\第 1 章\Debug\1.exe”为本书统一选择的路径，读者可根据自己的习惯选择其他不同的路径，以后类似问题不再重复说明。

2. 输入并运行教材中例 1.4 的程序。

教材中例 1.4 输入两个实数, 输出它们中较小的数。

【参考程序】

```
#include<stdio.h>
/*以下定义函数 min()*/
float min(float a,float b)
{ float temp;                                /*函数使用的变量的定义*/
  if(a<b)                                    /*这是 if 条件选择结构*/
    temp=a;
  else
    temp=b;
  return temp;                                /*返回 temp, 让控制返回到调用 min() 函数处*/
}
void main()                                  /*主函数*/
{ float x,y,c;                                /*变量定义*/
  printf("Input x and y.\n");
  scanf("%f%f",&x,&y);
  c=min(x,y);                                /*调用函数 min()*/
  printf("MIN(%.2f,%.2f)=%.2f\n",x,y,c);
}
```

程序运行结果如图 1-1-2 所示。

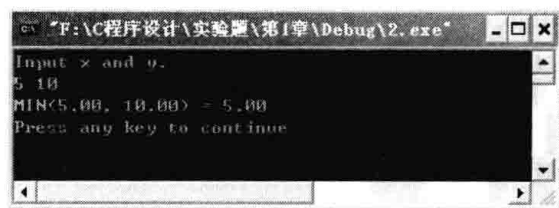


图 1-1-2 例 1.4 的运行结果

3. 编写输入华氏温度, 输出摄氏温度的转换程序。

【解题思路】

华氏温度与摄氏温度的转换公式如下:

$$C=(5/9)(F-32)$$

编写该程序时, 应考虑以下几个步骤:

- (1) 定义 C 和 F 两个变量, 其中 C 表示摄氏温度, F 表示华氏温度。
- (2) 输入华氏温度, 存入到变量 F 中。
- (3) 计算摄氏温度, 存入到变量 C 中。
- (4) 输出对应摄氏温度变量 C 中的值。

【参考程序】

```
#include<stdio.h>
void main()
{ float f,c;                                /*变量定义*/
  printf(" 请输入华氏温度: ");
  scanf("%f",&f);                            /*输入华氏温度*/
  c=5.0/9.0*(f-32.0);                        /*计算华氏温度转换为摄氏温度*/
  printf(" 对应的摄氏温度: %.2f\n",c);
}
```

程序运行结果如图 1-1-3 所示。

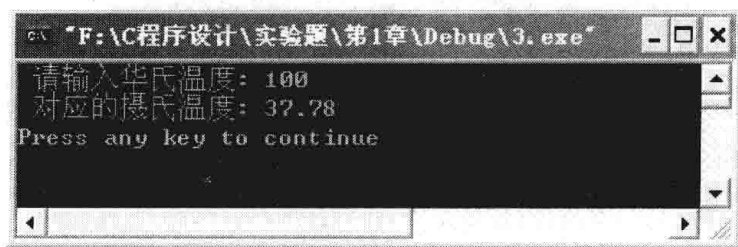


图 1-1-3 华氏温度转为摄氏温度的运行结果

【思考题】

在程序的第 6 行，计算摄氏温度是否可改成以下语句形式？给出行或不行的理由。

```
c=5/9*(f-32);          /*计算华氏温度转换为摄氏温度*/
```

4. 模仿教材中例 1.4 编写程序，要求输入两个实数，输出它们中较大的数。

【解题思路】

按教材中例 1.4，将程序分为主函数 main()和求较大值函数 max()。

(1) 主函数 main()的功能：

- ① 输入两个实数 x 和 y。
- ② 调用 max(x, y)函数，得到 x 与 y 中较大的数。
- ③ 输出结果。

(2) 求较大值函数 max()的功能：

- ① 分别将实参 x 和 y 的值传递给形参 a 和 b。
- ② 将 a 和 b 中较大的值赋给变量 temp。
- ③ 将变量 temp 的值返回给主函数。

【参考程序】

```
#include<stdio.h>
float max(float a,float b)          /*求较大值函数*/
{ float temp;                      /*函数使用的变量的定义*/
  if(a>b)                           /*这是 if 条件选择结构*/
    temp=a;
  else
    temp=b;
  return temp;                     /*返回 temp, 让控制返回到调用 max() 函数处*/
}
void main()                        /*主函数*/
{ float x,y,c;                     /*变量定义*/
  printf("Input x and y.\n");
  scanf("%f%f",&x,&y);
  c=max(x,y);                      /*调用函数 max()*/
  printf("MAX(%.2f, %.2f) = %.2f\n",x,y,c);
}
```

程序运行结果如图 1-1-4 所示。

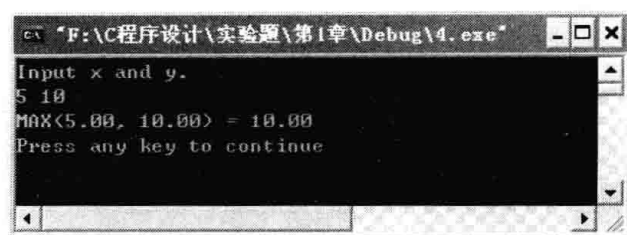


图 1-1-4 输出较大实数的运行结果

【思考题】

为什么主函数 `main()` 放在求较大值函数 `max()` 之后?

三、实验小结

运用 Visual C++ 6.0 或 Visual Studio 2010 中文版集成环境是学习 C 语言程序设计的第 1 步, 希望结合教材、上机指导等相关资料, 积极思考, 不断上机实践, 培养良好的编程风格。

实验 2 | 基本数据类型及其运算

一、实验目的

1. 掌握 C 语言数据类型，熟悉整型、字符型、浮点型变量的定义与赋值方法。
2. 掌握基本数据运算，特别是自增（++）、自减（--）和复合赋值运算。
3. 掌握不同数据类型之间进行混合运算的规律。
4. 熟悉 ASCII 码中字符的表达方法。
5. 掌握各种基本数据类型的输入/输出方法，能正确使用基本的格式转换符。

二、实验内容

通过上机实践，分析并输出下列程序的结果。

1. 程序代码如下：

```
#include<stdio.h>
void main()
{ char ch='a';
  printf(" %c对应的ASCII码是: %d\n",ch,ch);
}
```

程序运行结果如图 1-2-1 所示。

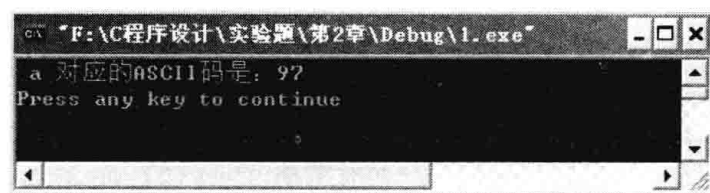


图 1-2-1 求 ASCII 码的运行结果

2. 程序代码如下：

```
#include<stdio.h>
void main()
{ int a=168;
  float b=123.456;
  float x=35.567,y;
  printf(" a = %5d\n",a); /*右对齐方式输出 5 个字符宽度,不足 5 位前补空格*/
  printf(" a = %-5d\n",a); /*左对齐方式输出 5 个字符宽度,不足 5 位后补空格*/
  printf(" b = %6.2f\n",b); /*共输出 6 个字符宽度,保留 2 位小数*/
  printf(" b = %E\n",b); /*以指数形式输出*/
  y=(int)(x*100+0.5)/100.0; /*实现 x 小数后第 3 位四舍五入,并保存到变量 y 中*/
}
```



```
printf(" y = %f\n",y);
}
```

程序运行结果如图 1-2-2 所示。

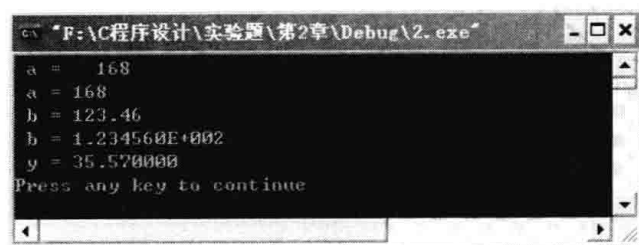


图 1-2-2 不同数据类型的运行结果

3. 程序代码如下:

```
#include<stdio.h>
void main()
{ int x=168;
  printf(" 十进制数 %d 对应的八进制数是: %o\n",x,x);
  printf(" 十进制数 %d 对应的十六进制数是: %x\n",x,x);
}
```

程序运行结果如图 1-2-3 所示。



图 1-2-3 数制转换的运行结果

4. 程序代码如下:

```
#include<stdio.h>
void main()
{ int m,n,p,x=8,y=18;
  printf(" x = %d\t y = %d\n",x,y);
  m=(x++)+(++y);          /*执行顺序: y=y+1; m=x+y; x=x+1;*/
  printf(" x = %d\t y = %d\t m = %d\n",x,y,m);
  n=(y--)-(--x);          /*执行顺序: x=x-1; n=y-x; y=y-1;*/
  printf(" x = %d\t y = %d\t n = %d\n",x,y,n);
  p=y%x;                  /*变量 p 取 y 除以 x 的余数*/
  printf(" x = %d\t y = %d\t p = %d\n",x,y,p);
  x=3;
  x+=x/=x-=x*x;           /*执行顺序: x=x-x*x; x=x/x; x=x+x;*/
  printf(" x = %d\n",x);
}
```

程序运行结果如图 1-2-4 所示。



图 1-2-4 自增自减的运行结果

5. 程序代码如下:

```
#include<stdio.h>
void main()
{ unsigned a=0x9d,b=0xa5; /*a 对应二进制数为: 1001 1101; b 为: 1010 0101*/
  int x,y,z;
  printf(" a & b = %x\n",a&b);
  /*a&b 是位与运算, 结果为 1000 0101, 对应十六进制数为 85*/
  printf(" a | b = %x\n",a|b);
  /*a|b 是位或运算, 结果为 1011 1101, 对应十六进制数为 bd*/
  printf(" a ^ b = %x\n",a^b);
  /*a^b 是位异或运算, 结果为 0011 1000, 对应十六进制数为 38*/
  x=a>>3;          /*将 a 右移 3 位, 结果为 10011, 对应八进制数为 23*/
  printf(" 0x9d >> 3 = %o\n",x);
  y=~(~0<<4);      /*执行顺序为: 将 0 取反为 1111 1111, 左移 4 位结果为 1111 0000,
  再取反结果为 0000 1111, 对应十进制数为 15*/
  printf(" ~ ( ~ 0 << 4) = %d\n",y);
  z=2<<4;          /*将 2 左移 4 位, 结果为 0010 0000, 对应十进制数为 32*/
  printf(" 2 << 4 = %d\n",z);
}
```

程序运行结果如图 1-2-5 所示。

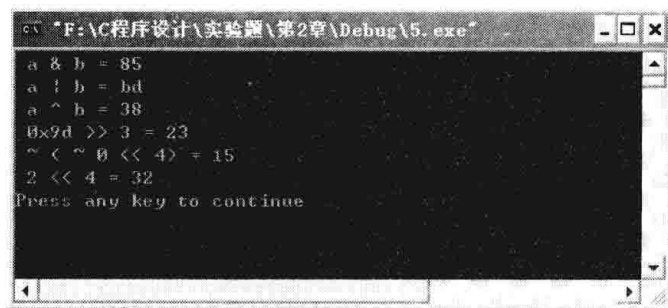


图 1-2-5 混合运算的运行结果

三、实验小结

通过上机实践, 掌握基本数据类型、基本数据的运算规律和输入/输出函数的使用方法。通过上机操作和编程锻炼, 进一步掌握 Visual C++ 6.0 或 Visual Studio 2010 中文版的基本操作, 并熟悉教学内容, 为以后各实验的学习奠定坚实的基础。

实验 3 | 结构化程序设计

一、实验目的

1. 熟练掌握顺序结构、选择结构 (if、switch)、循环结构 (while、do...while、for) 以及 continue、break 等控制语句。
2. 掌握一些应用选择结构和循环结构实现的常用算法。
3. 学会 Visual C++ 6.0 或 Visual Studio 2010 中文版中的程序调试方法。

二、实验内容

1. 输入一个字符, 若为小写字母, 则转换为对应的大写字母输出; 若为大写字母, 则转换为对应的小写字母输出。

【解题思路】

从 ASCII 码表中可以得知, 小写字母的 ASCII 码与大写字母的 ASCII 码之差为 32, 即大写字母 A 的 ASCII 码加上 32 得到小写字母 a 的 ASCII 码。

【参考程序】

```
#include<stdio.h>
void main()
{ char c1,c2;                                /*变量定义*/
  printf(" 请输入大写字母或小写字母: ");
  scanf("%c",&c1);                            /*输入字母*/
  if(c1>='A'&&c1<='Z')
  { c2=c1+'a'-'A';                            /*大写字母转换为小写字母, 或 c2=c1+32*/
    printf(" %c 转换为小写字母是 %c\n",c1,c2);
  }
  else if(c1>='a'&&c1<='z')
  { c2=c1+'A'-'a';                            /*小写字母转换为大写字母, 或 c2=c1-32*/
    printf(" %c 转换为大写字母是 %c\n",c1,c2);
  }
  else printf(" 输入的不是字母!\n");
}
```

程序运行结果如图 1-3-1 所示。

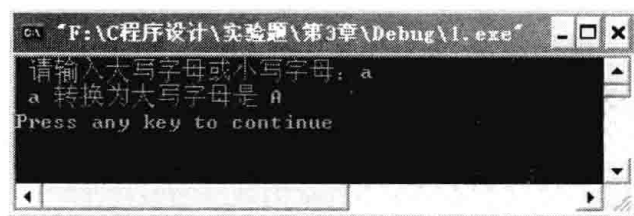


图 1-3-1 大小写字母转换的运行结果

【思考题】

为什么输入存储在 `c1` 中的大写字母，用表达式 `c1+'a'-'A'` 可以转换为对应小写字母的 ASCII 码？输入存储在 `c1` 中的小写字母，能用表达式 `c1+'A'-'a'` 转换为对应的大写字母？

2. 输入 3 个实数 a 、 b 、 c ，若能构成三角形，则输出三角形的面积。

【解题思路】

设三角形的三条边分别为 a 、 b 、 c 。为了判定方便，程序先对输入的数据作调整，让 c 为最大， b 其次， a 最小。当满足两边之和大于第 3 边时，则能构成三角形。

计算三角形的面积公式如下：

$$p = \frac{a+b+c}{2}$$
$$\text{area} = \sqrt{p(p-a)(p-b)(p-c)}$$

【参考程序】

```
#include<stdio.h>
#include<math.h>
void main()
{ double a,b,c,area,p;
  printf(" 请输入三角形的三条边 a, b, c : ");
  scanf("%lf%lf%lf",&a,&b,&c);      /*输入三角形的三条边*/
  if(c<b) {p=c;c=b;b=p;}
  if(c<a) {p=a;a=c;c=p;}
  if(b<a) {p=b;b=a;a=p;}
  /*满足 c>=b>=a*/
  if(a+b<=c)
  { printf(" 错误，不能构成三角形\n");
    return;
  }
  /*计算三角形的面积*/
  p=(a+b+c)/2.0; area=sqrt(p*(p-a)*(p-b)*(p-c));
  printf(" 三角形的面积 = %.2f\n",area);
}
```

程序运行结果如图 1-3-2 所示。

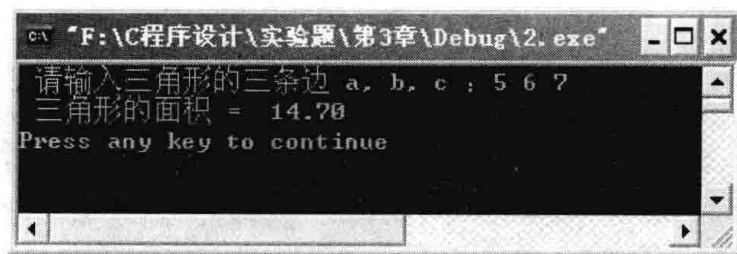


图 1-3-2 判定是否为三角形的运行结果

【思考题】

试修改程序，若能构成三角形，判定这个三角形是直角三角形、等边三角形、等腰三角形、普通的锐角三角形还是普通的钝角三角形。

3. 编制一个“判断成年人体重是否达标”的程序。

【解题思路】

判断体重标准方法是: 体重指数 $BMI = \text{体重} \div (\text{身高} \times \text{身高})$

正常体重: $18.5 \leq BMI < 24$; 偏瘦: $BMI < 18.5$; 超重: $24 \leq BMI < 27$;

轻度肥胖: $27 \leq BMI < 30$; 中度肥胖: $30 \leq BMI < 35$; 重度肥胖: $BMI \geq 35$ 。

【参考程序】

```
#include<stdio.h>
void main()
{ char s; /*s 表示性别*/
  float h,w; /*h 表示身高、w 表示体重*/
  float bw,bmi; /*bw 表示标准体重、bmi 表示体重指数*/
  printf("\t***** 这是一个测试成人体重标准的程序 *****\n\n");
  printf("请输入您的性别(M表示男性, F表示女性): ");
  scanf("%c",&s); /* 输入性别 */
  printf("请输入您的身高(CM): ");
  scanf("%f",&h); /*输入身高*/
  printf("请输入您的体重(KG): ");
  scanf("%f",&w); /*输入体重*/
  bmi=w/(h*h)*10000; /*体重指数 BMI=体重/身高的平方*/
  if(s=='M' || s=='m')
    bw=(h-100)*0.95; /*男性的标准体重=(身高-100)×0.95*/
  else bw=(h-100)*0.9; /*女性的标准体重=(身高-100)×0.9*/
  printf("标准体重为: %0.2f\t 体重指数 BMI= %0.2f\n",bw, bmi);
  if(bmi<18.5) printf("体重偏瘦\n");
  if(bmi>=18.5&&bmi<24) printf("体重正常\n");
  if(bmi>=24 &&bmi<27) printf("体重超重\n");
  if(bmi>=27 &&bmi<30) printf("轻度肥胖\n");
  if(bmi>=30 &&bmi<35) printf("中度肥胖\n");
  if(bmi>=35) printf("重度肥胖\n");
}
```

程序运行结果如图 1-3-3 所示。

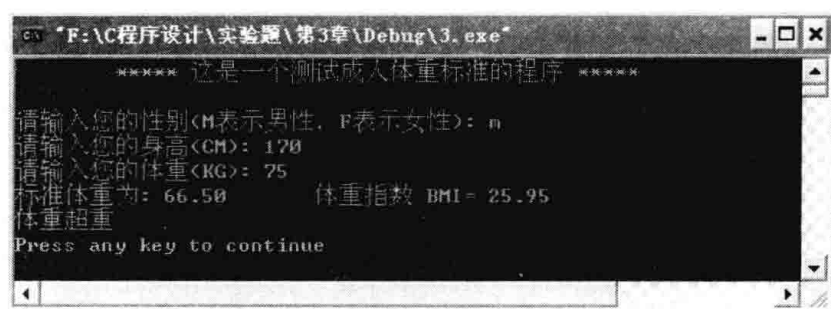


图 1-3-3 判定体重达标的运行结果

【思考题】

程序中判定体重是否达标连续使用了若干个 if 语句, 为什么可以没有 else 语句?

4. 给出一个百分制成绩, 要求输出对应成绩等级 A、B、C、D、F。例如, 90 分以上为 A, 80~89 分为 B, 70~79 分为 C, 60~69 分为 D, 60 分以下为 F。

【解题思路】

由于此题涉及多分支, 在程序中使用 switch 语句显然比 if 语句简单。设分数用 score 变量存储, 则用 score 除以 10 取商作为 switch 语句判定的条件。须要注意的是, 当输入的成绩小于 0 或大于 100 时, 程序应输出“输入数据错误!”的提示信息。

【参考程序】

```
#include<stdio.h>
void main()
{ float score;
  char degree;
  printf(" 请输入百分制的成绩: ");
  scanf("%f",&score);
  if(score<0||score>100)
  { printf(" 输入数据错误!\n");
    return;
  }
  switch((int)score/10)
  { /*取成绩除以 10 的商*/
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
    case 5: degree='F'; break; /*0~59 分*/
    case 6: degree='D'; break; /*60~69 分*/
    case 7: degree='C'; break; /*70~79 分*/
    case 8: degree='B'; break; /*80~89 分*/
    case 9:
    case 10: degree='A'; break; /*90~100 分*/
  }
  printf(" 成绩 %.0f 分, 对应的等级为 %c\n",score,degree);
}
```

程序运行结果如图 1-3-4 所示。

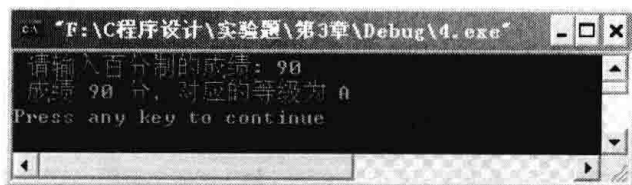


图 1-3-4 判定成绩等级的运行结果

【思考题】

在 switch 语句的判定条件中, 为什么要先将输入的成绩 score 强制转换为整型后再除以 10?

5. 编制一个“计算个人所得税”的程序。

【解题思路】

个人所得税 = 应税金额 × 适用税率 - 速算扣除数。

税率如表 1-3-1 所示。

表 1-3-1 税率一览表

应 税 金 额	适 用 税 率	速算扣除数
≤1 500	3%	0
(1 500, 4 500]	10%	105
(4 500, 9 000]	20%	555
>9 000	25%	1 005

【参考程序】

```
#include<stdio.h>
void main()
{ float x,y;
  printf("\t***** 个人所得税计算器 *****\n\n");
  printf("请输入您的薪酬总金额(其中免税金额为 3500 元: ");
  scanf("%f",&x);
  if(x<=3500)                                /*低于 3500 免税*/
  { printf("您输入的薪酬金额低于 3500 元, 不需要加个人所得税!\n");
    return;
  }
  x=x-3500;                                  /*x 为应税金额*/
  switch((int)(x/1500))
  { case 0: y=0.03*x; break;                  /*应税金额小于 1500*/
    case 1:
    case 2:
    case 3: y=x*0.10-105; break;              /*应税金额在 1500~4500*/
    case 4:
    case 5:
    case 6: y=x*0.20-555; break;              /*应税金额在 4500~9000*/
    default: y=x*0.25-1005;                   /*应税金额大于 9000*/
  }
  printf("您的应税金额为: %.2f, 应交个人所得税为: %.2f\n",x,y);
}
```

程序运行结果如图 1-3-5 所示。

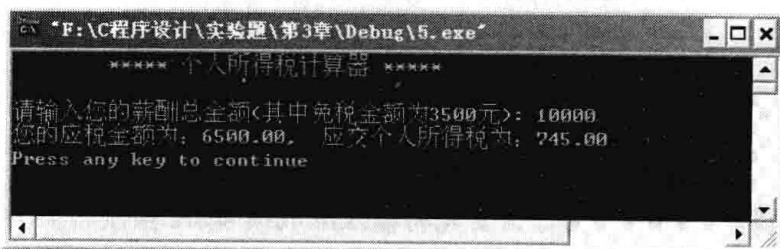


图 1-3-5 计算个人所得税的运行结果

【思考题】

试使用 if 语句代替 switch 语句，实现上述程序的功能。

6. 水仙花数是一个 n ($n \geq 3$) 位数字的数，它等于每个数字的 n 次幂之和。例如，153 是一个水仙花数，因为 $153=1^3+5^3+3^3$ 。试编写一个程序，求小于 999 的所有水仙花数。

【解题思路】

编写计算水仙花数的程序显然要从 100 循环到 999。假设要考察的 3 位数为 n ，程序先要分别求出 n 的个位数 k 、十位数 j 和百位数 i ，然后判别 k 的 3 次方、 j 的 3 次方与 i 的 3 次方之和是否等于 n ，若相等，则 n 是一个水仙花数，将 n 输出即可。

【参考程序】

```
#include<stdio.h>
void main()
{ int i,j,k,n;
  printf(" 水仙花数:");
  for(n=100;n<=999;n++)
  { i=n/100;                                /*取 n 的百位数*/
    j=(n-i*100)/10;                        /*取 n 的十位数*/
    k=n%10;                                /*取 n 的个位数*/
    if(n==i*i*i+j*j*j+k*k*k)
      printf("%6d",n);
  }
  printf("\n");
}
```

程序运行结果如图 1-3-6 所示。

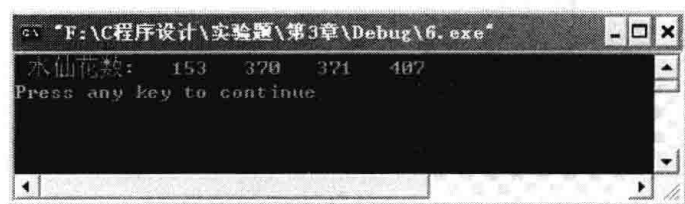


图 1-3-6 水仙花数的运行结果

【思考题】

使用 while 语句或 do...while 语句代替上述程序的 for 循环语句，实现上述程序的功能。

7. 编写程序解百鸡问题：鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。问百钱买百鸡时，鸡翁、鸡母和鸡雏各几何？

【解题思路】

此题的意思是，每只公鸡价值是 5 钱，每只母鸡价值是 3 钱，每 3 只小鸡价值是 1 钱。如果有 100 钱想买 100 只鸡，问公鸡、母鸡和小鸡可以各买几只？

设 x 、 y 、 z 分别代表鸡翁、鸡母、鸡雏的数目，则得到以下方程式：

$$\begin{cases} 5x+3y+\frac{z}{3}=100 & (1) \\ x+y+z=100 & (2) \end{cases}$$

这实际上是一个三元一次不定式方程组, 显然 x 、 y 和 z 是 $0 \sim 100$ 范围内的整数, 并由方程 (1) 知, z 是 3 的倍数。

程序要寻找满足上述方程 (1) 和方程 (2) 的正整数解, 可以采用穷举法, 让 x 和 y 分别穷举 $0 \sim 100$ 的整数。对确定的 x 和 y 由方程式 (2) 求出 z , 测试 z 是否是 3 的倍数以及它们是否满足方程式 (1)。

【参考程序】

```
#include<stdio.h>
void main()
{ int x,y,z;
  printf(" 百鸡问题解:\n");
  for(x=0;x<=100;x++)
    for(y=0;y<=100-x;y++)
    { /*必须满足以下两个方程, 并且 z 是 3 的倍数, 才能得到方程的解*/
      z=100-x-y; /*满足方程 (2) */
      if(z%3!=0) continue; /*并且 z 必须是 3 的倍数*/
      if(5*x+3*y+z/3==100) /*满足方程 (1) */
        printf("\tx = %3d  y = %3d  z = %3d\n",x,y,z);
    }
  printf("\n");
}
```

程序运行结果如图 1-3-7 所示。

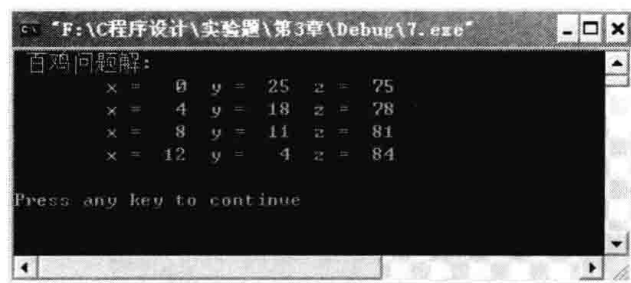


图 1-3-7 百鸡的运行结果

【思考题】

由 (1) 式和 (2) 式得到以下方程式 (3):

$$7x+4y=100 \quad (3)$$

是否可以利用方程 (3) 求解百鸡问题? 如果可以, 程序应如何修改?

8. 输入正整数 n , 输出 n 行 n 列的空心四边形图案。以下是 n 等于 5 的图案。

```
* * * * *
*           *
*           *
*           *
*           *
* * * * *
```

【解题思路】

可用两重循环实现图案的输出要求。外循环 i 从 1 变化到 n , 即要输出 n 行; 内循环 j 也是从 1 变化到 n , 即从第 1 列输出到第 n 列。注意: 第 1 行与最后 1 行要全部输出星号, 而第 $2 \sim n-1$ 行仅第 1

列和最后 1 列才要输出星号。因此,当 $j=1$ (表示第 1 列) 或 $j=n$ (表示最后 1 列) 时要输出星号;另外还要考虑,当 $i=1$ (表示第 1 行) 或 $i=n$ (表示最后 1 行) 时,也要输出星号,其余位置输出空白。

【参考程序】

```
#include<stdio.h>
void main()
{ int i,j,n;
  printf(" 输入 n 的值: ");
  scanf("%d",&n);
  for(i=1;i<=n;i++)          /*外循环,控制行的变化*/
  { for(j=1;j<=n;j++)          /*内循环,控制列的变化*/
    if(i==1||i==n||j==1||j==n)
      printf(" *");           /*第 1 行和最后 1 行、第 1 列和最后 1 列须输出星号*/
    else printf(" ");          /*否则输出一个空白字符*/
    printf("\n");              /*换行,准备输出下一行内容*/
  }
}
```

程序运行结果如图 1-3-8 所示。

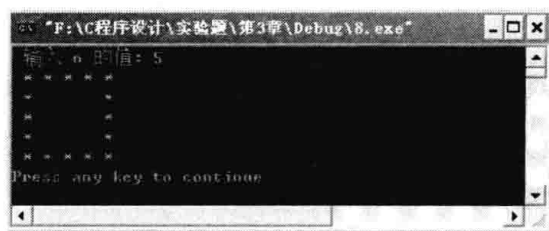


图 1-3-8 空心四边形的运行结果

【思考题】

如果要输出 n 行 n 列的实心四边形图案,如何修改程序?

9. 编写程序,按下面的公式计算自然对数 e 的近似值。

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} + \cdots$$

【解题思路】

按 e 的幂级数展开式计算 e 的近似值,这是典型的求级数的计算问题,通常采用逐项计算并累计的方法。计算新的项可采用递推法,即用上一轮循环计算出的项除以 i 得到,这样能提高程序的效率。循环的次数取决于当前项的值是否满足精度的要求。

【参考程序】

```
#include<stdio.h>
void main()
{ double e,t,i;
  e=0.0;t=1.0;i=1.0;
  while(t>=1.0e-6)
  { e+=t;t/=i;i+=1.0;          /*t 表示第 i 项的值, t 除以 i 得到下一项的值*/
  }
  printf("e = %.6f\n",e);
}
```

程序运行结果如图 1-3-9 所示。

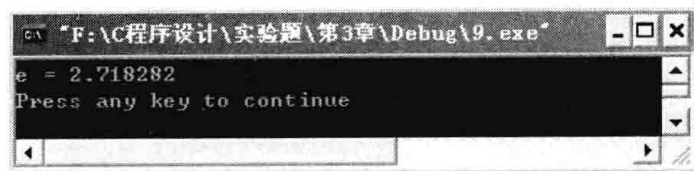


图 1-3-9 自然对数 e 的近似值的运行结果

【思考题】

计算 e 的公式中, 第 i 项的分母是 i!, 为什么在程序中没有直接看到计算求阶乘的算式, 计算的结果却是正确的?

10. 编写程序, 用如下公式计算圆周率 π 的近似值。

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \dots$$

【解题思路】

此题拟采用逐项计算并累计的方法实现, 设存储累计结果的变量为 pi。从公式中看, 每项的分子固定为 4, 每项的分母是在前一项分母的基础上加 2。另外, 奇数项为正值, 偶数项为负值。

在编程时, 可以考虑用变量 j 作为分母 (初始值为 1, 计算下一项时每次加 2), 用变量 s 表示每项是正值或负值 (初始值为 1, 计算下一项时取反, 可实现奇数项为 1, 偶数项为 -1), 用变量 t 作为每一项的计算值 (初始值为 4, 第 1 项的值)。

累计过程是一个循环, 当某一项的绝对值小于 10^{-6} (即 0.000 001) 时, 即求出 π 的近似值。

【参考程序】

```
#include<stdio.h>
#include<math.h>
void main()
{ long i,j=1,s=1;
  double pi=4,t=4;                /*t 作为每一项的计算值*/
  for(i=1;fabs(t)>1e-6;i++)        /*当|t|>0.000 001 时继续计算下一项*/
  { s=-s;                          /*s 表示每项是正值或负值*/
    t=s*4.0/(j+=2); pi+=t;         /*j 作为分母, 每项的值累计到 pi*/
  }
  printf(" PI = %f\n",pi);
}
```

程序运行结果如图 1-3-10 所示。

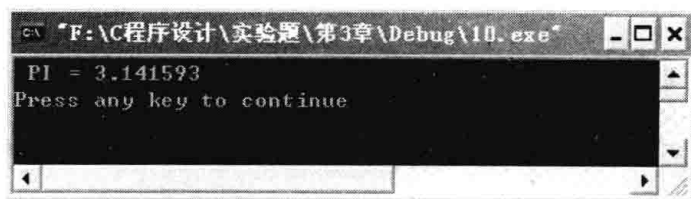


图 1-3-10 圆周率的近似值的运行结果

【思考题】

如果还要考虑计算多少项后才能得到 π 的近似值，可以在程序中设置一个项数计数器。要求程序能输出要计算多少项才能得到数值 3.14、3.141、3.141 5、3.141 59？

提示：累计直至得到某个数值的条件由两部分组成，首先是前几位确实与某个指定的数值相同，其次是与指定数值差的绝对值应小于指定数值最后一位的 0.5 单位，如 π 累计要达到 3.14 的条件是 `((int)(pi*100-314) == 0)&&fabs(pi-3.14)<0.005`。

三、实验小结

本实验主要涉及运用 3 种控制结构进行结构化程序设计，学习了不少新的语句形式。要求在学习过程中结合上机实验，扎实掌握结构化程序设计的基本方法。

实验 4 | 数 组

一、实验目的

1. 掌握一维数组和二维数组的定义、赋值和输入/输出方法。
2. 掌握与数组有关的常用的统计、整理和排序的基本算法。
3. 掌握字符数组、字符串的处理方法，能正确使用字符串库函数。

二、实验内容

1. 统计一维数组中小于 0、等于 0 和大于 0 的元素个数。

【解题思路】

设数组为 $x[]$ ，元素个数为 n 。引入大于 0、等于 0 和小于 0 的计数器变量 $great$ 、 $equal$ 和 $less$ 。在统计之前，计数器变量必须预置初值为 0。用循环遍历数组，顺序考察数组的每个元素，当发现正在考察的元素满足某个条件时，相应的计数器变量加 1。

【参考程序】

```
#include<stdio.h>
void main()
{ int i,less,equal,great,n=10;
  /*设置一维数组 x 并赋初始值*/
  int x[]={1,5,-4,0,-2,0,4,-10,9,100};
  great=equal=less=0;
  for(i=0;i<n;i++)
    if(x[i]>0) great++;
    else if(x[i]==0) equal++;
    else less++;
  printf(" x[] =");
  for(i=0;i<n;i++) printf("%4d",x[i]);
  printf("\n 数组中小于 0 的个数: %d\n",less);
  printf(" 数组中等于 0 的个数: %d\n",equal);
  printf(" 数组中大于 0 的个数: %d\n",great);
}
```

程序运行结果如图 1-4-1 所示。

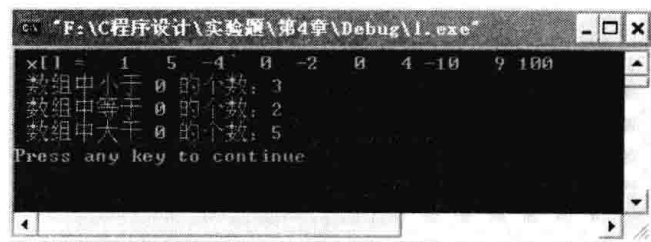


图 1-4-1 统计元素个数的运行结果

【思考题】

上述程序中，一维数组 $x[]$ 是在程序中赋初始值的，如何改成从键盘上输入 n 个数据？

2. 将一维数组 a 中具有相同值的元素仅保留一个，然后按值从小到大的顺序输出。

【解题思路】

在一维数组中，将具有相同值的元素仅保留一个的算法思想是：将 $a[i]$ 依次与其后每个元素进行比较，若值相同则其后所有元素往前移一个位置，即删除了 $a[i]$ 元素，同时元素总个数减少一个；排序一般使用冒泡法，即所有相邻两元素进行比较，将小的排在前面，大的排在后面即可。

【参考程序】

```
#include<stdio.h>
void main()
{ int i,j,k,n,temp;
  int a[100];                /*设置一维数组 a*/
  printf("请输入数组 a 中每个元素值(最多不超过 20 个)\n");
  for(n=0;n<20;n++){        /*输入数据，送到一维数组 a 中*/
    printf("请输入第 %d 个元素的值(输入值为-9999 表示结束): ",n+1);
    scanf("%d",&a[n]);
    if(a[n]==-9999) break;    /*输入-9999，表示输入结束*/
  }
  printf("\n 原数组 a[] =");  /*输出原数组内容*/
  for(i=0;i<n;i++)printf("%4d",a[i]);
  for(i=0;i<n;i++)           /*第 i 个元素依次和后面元素比较*/
    for(j=i+1;j<n;j++)
      if(a[i]==a[j])         /*找到相同的元素，将该元素删除*/
      { for(k=j+1;k<n;k++) a[k-1]=a[k];
        j--;                /*移上来的元素仍要比较*/
        n--;                /*总元素减少 1*/
      }
  for(i=0;i<n-1;i++)         /*对数组进行小到大排序*/
    for(j=0;j<n-1-i;j++)
      if(a[j]>a[j+1])
        {temp=a[j];a[j]=a[j+1];a[j+1]=temp;}
  printf("\n 结果数组 a[] ="); /*输出排序后数组内容*/
  for(i=0;i<n;i++) printf("%4d",a[i]);
  printf("\n");
}
```

程序运行结果如图 1-4-2 所示。

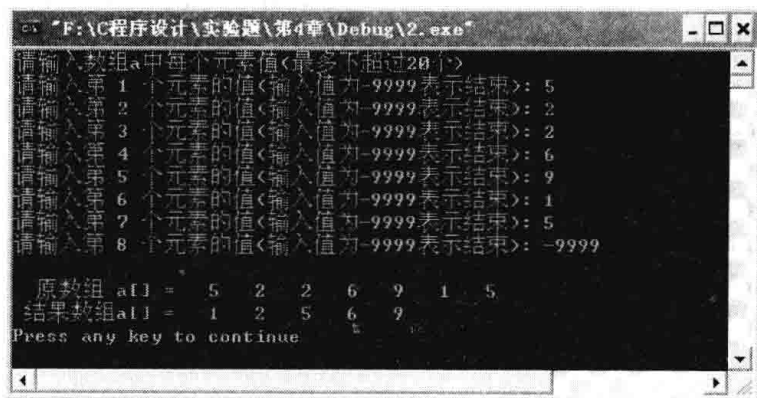


图 1-4-2 一维数组中按值从小到大输出的运行结果

【思考题】

若上述程序最后不要排序,改为按一维数组中原先存储的顺序反向重新存储,如何修改程序?

例如,一维数组 $a[] = \{2, 6, 9, 1, 5\}$, 反向存储后应变为 $a[] = \{5, 1, 9, 6, 2\}$ 。

3. 设有 n 个元素存放在一个数组 $a[]$ 中,并按升序排列。输入一个数存储到变量 key 中,要求用二分法查找出该数是数组中的第几个元素,如果该数不在数组中,则输出“无此数!”。

【解题思路】

二分法查找的算法思想如下:

首先将 key 与一维数组中的元素 $a[m]$ 进行比较 ($m = (i+j)/2$, 变量 i 的初始值为第 1 个元素的下标 0, 变量 j 的初始值为最后 1 个元素的下标 $n-1$)。 key 与 $a[m]$ 比较有 3 种可能结果,分别采取 3 种不同的对策:

$key = a[m]$, 找到, 结束查找。

$key > a[m]$, 下一轮的查找区间为 $[m+1, j]$ 。

$key < a[m]$, 下一轮的查找区间为 $[i, m-1]$ 。

由于每次循环查找使查找区间减半,因此称此查找方法为二分查找法。当循环条件 $j < i$ 时, 区间 $[i, j]$ 变为一个空区间,即表示在数组 $a[]$ 中没有值为 key 的元素,此时应显示“无此数!”。

在下面程序中,一次查找结束后,程序询问是否要继续作下一轮查找。

【参考程序】

```
#include<stdio.h>
#define n 10                                /*定义数组元素的个数*/
void main()
{ int a[n], i, j, m, key;
  char ans;
  for(i=0; i<n; i++) a[i]=i*2;             /*对一维数组 a[] 赋初始值*/
  printf("数组 a[] =");                    /*输出数组的每个元素*/
  for(i=0; i<n; i++) printf("%4d", a[i]);
  printf("\n");
  while(1)
  { printf("\n 请输入要查找的数: ");
    scanf("%d", &key);
    i=0; j=n-1;                             /*设定初始查找区间*/
    while(i<=j)
    { m=(i+j)/2;                             /*取中间元素的下标*/
      if(key==a[m]) break;                   /*找到, 跳出循环*/
      if(key>a[m]) i=m+1;                   /*舍弃前面一半, 下次从后面一半查找*/
      else j=m-1;                           /*舍弃后面一半, 下次从前面一半查找*/
    }
    if(i<=j)                                /*找到时, i<=j, a[m]==key; 没有找到时, i>j*/
      printf("找到, 对应的下标是 %d\n\n", m);
    else printf("无此数!\n\n");
    printf("是否继续做查找操作(Y/N)?");
    fflush(stdin);                          /*清除输入缓冲区中数据*/
    scanf("%c", &ans);
    if(ans=='N' || ans=='n') break;         /*结束程序运行*/
  }
}
```

程序运行结果如图 1-4-3 所示。

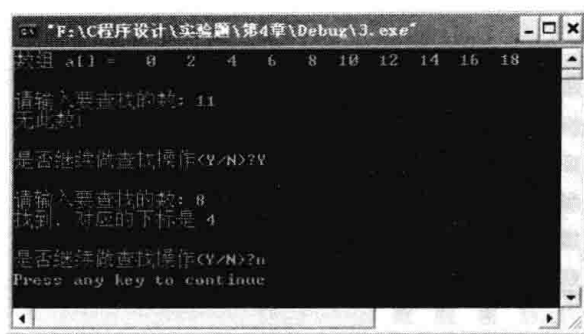


图 1-4-3 二分法查找的运行结果

【思考题】

上述程序中, 一维数组 $a[]$ 的元素是通过赋初始值设定的, 请改成从键盘上输入 n 个数据并对数组进行排序, 再用二分法进行查找。

4. 求前 50 个质数。例如, 前 10 个质数依次为 2、3、5、7、11、13、17、19、23、29。

【解题思路】

对任意整数 m , 如果它不能被小于它的质数整除, 则 m 是质数。在程序中使用一维数组 `primes[]` 存储质数, 质数的个数用变量 `count` 表示。要求确定一个整数 m 是否是质数, 用已求出的质数对 m 的整除性来确定。当 m 能被某个质数整除时, 则 m 加 2, 并重新用第 2 个质数开始逐一检验它们对 m 的整除性。如果 m 不能被平方不超过 m 的所有质数整除, 则 m 是一个质数。

【参考程序】

```
#include<stdio.h>
#define N 50
void main()
{ int primes[N],m,k,count;
  primes[0]=2;count=1;          /*2 是第一个质数, 已有一个质数*/
  m=3;                          /*被测试的数从 3 开始*/
  k=0;
  while(count<N)
  { while(primes[k]*primes[k]<=m)
    { if(m%primes[k]==0)
      {                          /*m 除得尽已求出的质数, 则 m 是合数*/
        m+=2;                  /*让 m 取下一个奇数*/
        k=1;                   /*不必用 primes[0]=2 去测试 m, 所以 k 从 1 开始*/
      }
      else k++;                /*继续用下一个质数去测试*/
      primes[count++]=m;        /*m 是质数*/
      m+=2;k=1;                /*除 2 外, 其余质数均为奇数*/
    }
    printf(" 前 %d 个质数如下: ",count);
    for(k=0;k<count;k++)
    { if(k%10==0) printf("\n"); /*每行输出 10 个元素*/
      printf("%4d",primes[k]);
    }
    printf("\n");
  }
}
```

程序运行结果如图 1-4-4 所示。

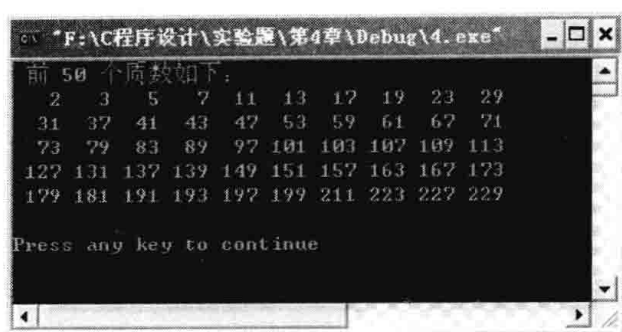


图 1-4-4 前 50 个质数的运行结果

【思考题】

求质数有许多种方法, 试用其他方法来判别一个整数 m 是合数还是质数。

5. 编写一个程序, 输入一个整型数, 输出整型数的每位数字, 在每位数字之间用逗号分隔。

例如, 输入的整型数为 12345, 则输出应为 1, 2, 3, 4, 5。

【解题思路】

程序的主要工作是将从整数拆分出来的各位数字存于一维数组中, 然后将这些数字按自高位到低位的顺序逐位输出。拆分一个整数要用一个循环。每次循环所做的工作是: 将拆分的整数除以 10 取余数求出它的个位数, 同时被拆分的整数除以 10, 即缩小为原来的 $1/10$, 去掉个位数。当被拆分的整数为 0 时, 循环结束。

【参考程序】

```
#include<stdio.h>
void main()
{ int a[20],i,n,base=10;
  printf("请输入一个整数: "); scanf("%d",&n); /*整数输入*/
  i=0;
  do
  { /*对 n 进行拆分, 将各位数字自低到高存于数组 a*/
    a[i++]=n%base;    n/=base;
  } while(n);
  for(i--;i>=0;i--)
    printf("%d%c",a[i],i?',':'\n'); /*自高位到低位输出*/
}
```

程序运行结果如图 1-4-5 所示。

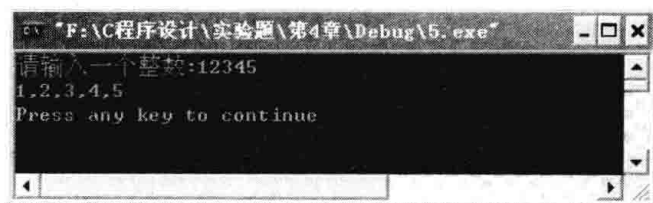


图 1-4-5 整数拆分的运行结果

【思考题】

(1) 在程序的第 3 行, 为什么要定义整型变量 `base`, 并设置其值为 10?

(2) 修改程序, 将输入的正整数逆序输出。例如, 输入的正整数为 12345, 输出的结果为 54321。

6. 输入一个 $n \times m$ 的整数矩阵, 输出矩阵中的最大值、最小值以及对应的下标。

【解题思路】

矩阵是一个二维数组。寻找二维数组的最大值和最小值应使用两重循环来实现, 即按行(外循环)和按列(内循环)来考察二维数组的每一个元素。首先, 用二维数组的第 1 个元素, 即 $a[0][0]$ 假设为最小值及最大值元素(用代码 $\text{min}=\text{max}=a[0][0]$ 实现), 然后依次与数组中的其他元素进行比较, 较大值元素和较小值元素分别存于变量 max 与变量 min 中。

【参考程序】

```
#include<stdio.h>
#define N 4
#define M 5
void main()
{ int a[N][M], i, j;
  int max, maxrow, maxcol, min, minrow, mincol;
  printf("请输入一个 %d×%d 的矩阵\n", N, M);
  for(i=0; i<N; i++)
    for(j=0; j<M; j++) scanf("%d", &a[i][j]); /* 整数输入 */
  min=max=a[0][0]; /* 假设第 1 个元素为最小值和最大值 */
  /* 最小值和最大值元素的下标行、下标列均为 0 */
  minrow=mincol=maxrow=maxcol=0;
  for(i=0; i<N; i++)
    for(j=0; j<M; j++)
      { if(a[i][j]>max)
        { max=a[i][j]; maxrow=i; maxcol=j; }
        else if(a[i][j]<min){ min=a[i][j]; minrow=i; mincol=j; }
      }
  printf(" MAX = a[%d][%d] = %d  MIN = a[%d][%d] = %d\n",
    maxrow, maxcol, max, minrow, mincol, min);
}
```

程序运行结果如图 1-4-6 所示。

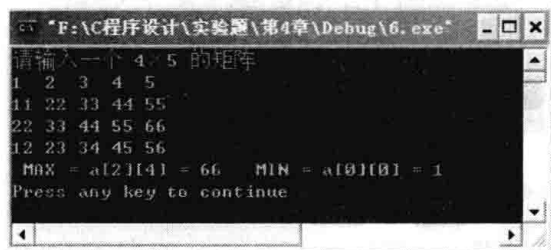


图 1-4-6 整数矩阵最大值、最小值的运行结果

【思考题】

修改程序, 使之能输出矩阵中每行的最大值、最小值以及它们的下标。

7. 设一个班有 n 名学生, 每一位学生有 3 门课的成绩, 如表 1-4-1 所示。

表 1-4-1 成绩表

课程 1	课程 2	课程 3
75	80	85
80	85	90
...	...	...

编写程序, 输入 n 个 (假设 $n=5$) 学生的 3 门课成绩, 输出每门课成绩的平均分。

【解题思路】

程序定义一个 n 行 3 列的二维数组, 用于存储全班学生的成绩。按顺序依次输入每个学生的 3 门课成绩, 然后按列的顺序累计各门课的总分, 并计算每门课的平均分。

【参考程序】

```
#include<stdio.h>
#define N 5
#define M 3
void main()
{ int score[N][M],i,j;
  float s; /*存放平均成绩*/
  printf("请输入每位学生的三门课成绩\n");
  for(i=0;i<N;i++)
    for(j=0;j<M;j++)
      scanf("%d",&score[i][j]); /*成绩输入*/
  for(j=0;j<M;j++)
  { for(s=0.0,i=0;i<N;i++)
    s+=score[i][j]; /*每门课的成绩的累计*/
    printf("课程 %d 的平均分是 %.2f\n",j+1,s/N);
  }
}
```

程序运行结果如图 1-4-7 所示。

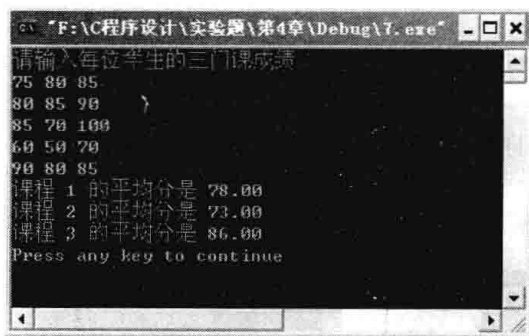


图 1-4-7 成绩平均分的运行结果

【思考题】

修改程序, 使之能输出每位学生的平均成绩。

8. 输出 8 行以内的杨辉三角形, 如图 1-4-8 所示。

```

      1
    1  1
  1  2  1
1  3  3  1
1  4  6  4  1
```

图 1-4-8 杨辉三角形

【解题思路】

杨辉三角形是 $(a+b)^n$ 展开式的各项系数。根据上述图形可以看到, 杨辉三角形具有以下一

些特征:

(1) 第 1 列数 (列下标为 0) 都是 1。

(2) 主对角线 (行下标=列下标) 都是 1。

(3) 从第 3 行起 (行下标 $i=2$), 除第 1 列与主对角线外, 每个数都是上一行同列的数与左边的数之和, 即满足下列公式:

$$a[i][j] = a[i-1][j-1] + a[i-1][j]$$

【参考程序】

```
#include<stdio.h>
#define N 8
void main()
{ int i,j,n,a[N][N];
  while(1)
  { printf("请输入行数: ");
    scanf("%d",&n);
    if(n>=1&&n<=N) break;
    printf("输入的行数必须在 1~%d 之间, 请重新输入行数\n", N);
  }
  for(i=0;i<n;i++) a[i][i]=a[i][0]=1; /*对角线和第 1 列元素均为 1*/
  /*第 3 行起, 每个元素是上一行同列与上一行前一列两数之和*/
  for(i=2;i<n;i++)
    for(j=1;j<=i-1;j++)
      a[i][j]=a[i-1][j-1]+a[i-1][j];
  for(i=0;i<n;i++)
  { /*输出结果*/
    printf("%*c",20-2*i,' ');
    for(j=0;j<=i;j++) printf("%4d",a[i][j]);
    printf("\n");
  }
  printf("\n");
}
```

程序运行结果如图 1-4-9 所示。

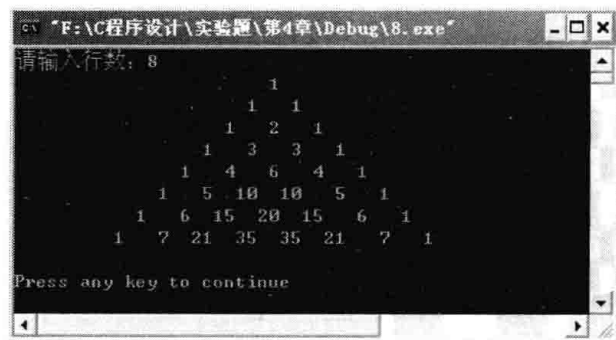


图 1-4-9 杨辉三角形的运行结果

【思考题】

如何将上述杨辉三角形输出成图 1-4-10 所示的三角形?



图 1-4-10 思考题图

9. 有两个字符串 s_1 和 s_2 , 编写程序实现将 s_1 和 s_2 连接后输出, 不能使用字符串连接函数 `strcat()`。

【解题思路】

在程序中设有两个能存放字符串的数组, 先输入两个字符串, 然后找到第 1 个字符串 s_1 的末尾, 接着将第 2 个字符串 s_2 复制到第 1 个字符串 s_1 的后面, 然后输出。

【参考程序】

```
#include<stdio.h>
#define N 120
void main()
{ int i,j;
  char s1[N+N],s2[N];
  printf("请输入第 1 个字符串: ");
  scanf("%s",s1);
  printf("请输入第 2 个字符串: ");
  scanf("%s",s2);
  for(i=0;s1[i]!='\0';i++);          /*找 s1 字符串末尾*/
  for(j=0;s2[j]!='\0';j++)
    s1[i++]=s2[j];                  /*将 s2 字符串依次复制到 s1 的末尾*/
  s1[i]='\0';                        /*字符串以 '\0' 结尾*/
  printf("新字符串为: %s\n",s1);
}
```

程序运行结果如图 1-4-11 所示。



图 1-4-11 字符串连接的运行结果

【思考题】

分别编程实现求字符串长度函数 `strlen()`、字符串复制函数 `strcpy()` 和字符串比较函数 `strcmp()` 的功能。

10. 输入字符串, 分别统计其中大写字母、小写字母、数字及其他字符出现的次数。

【解题思路】

程序首先设置 4 个计数器, 分别表示大写字母、小写字母、数字及其他字符出现的次数, 然后使用循环顺序考察字符串当前的字符, 对应的计数器加 1, 直至遇到字符串结束符 ('\0') 结束。

【参考程序】

```
#include<stdio.h>
#define N 120
void main()
{ int i,upper,lower,digital,others;
  char s[N];
  upper=lower=digital=others=0;
  printf("请输入字符串: ");
  gets(s);
  for(i=0;s[i]!='\0';i++)
    if(s[i]>='A'&&s[i]<='Z')
      upper++;                      /*大写字母*/
    else if(s[i]>='a'&&s[i]<='z')
      lower++;                      /*小写字母*/
    else if(s[i]>='0'&&s[i]<='9')
      digital++;                    /*数字*/
    else others++;                  /*其他字符*/
  printf("大写字母个数: %d\n",upper);
  printf("小写字母个数: %d\n",lower);
  printf("数字个数      : %d\n",digital);
  printf("其他字符个数: %d\n",others);
}
```

程序运行结果如图 1-4-12 所示。

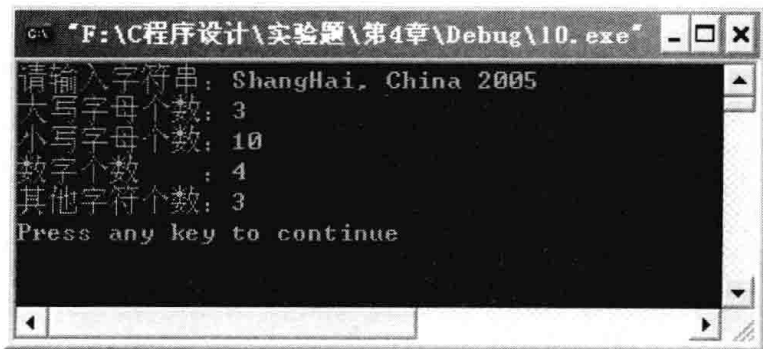


图 1-4-12 统计字母出现次数的运行结果

【思考题】

修改上述程序, 使之能够同时输入多行字符串, 再分别统计大写字母、小写字母、数字及其他字符出现的次数。

三、实验小结

数组是一种非常重要的数据类型,在定义数组时,系统将自动为它分配一块连续的存储区域,数组名指向这块区域的开始位置,即首地址。数组名是一个常量,也就是说,在程序运行过程中不能改变数组名的值。在 C 语言中,数组与指针之间有较为密切的联系,通过学习本实验,能为以后学习指针打下坚实的基础。本实验学习的重点是一维数组和二维数组的应用及字符串处理的基本方法。

实验 5 | 函 数

一、实验目的

1. 掌握 C 语言函数定义及调用的规则。
2. 了解函数调用的执行过程。
3. 理解并掌握实参与形参之间传递数据信息的过程。
4. 能为简单的函数设计算法和编写函数。
5. 了解并掌握宏定义的应用。

二、实验内容

1. 编写判别一个正整数是否是质数的函数。

【解题思路】

判别一个正整数 n 是否为质数的步骤如下：

- (1) $n < 2$ ，则 n 不是质数。
- (2) $n = 2$ ，则 n 是质数。
- (3) n 为大于 2 的偶数，则 n 不是质数。
- (4) n 为大于 2 的奇数，进行测试：若能被任何小于 $n/2$ 的奇数整除，则 n 为合数，否则为质数。

【参考程序】

```
#include<stdio.h>
void main()
{ long n;
  int isPrime(long n);          /*函数声明*/
  char ans;
  while(1)
  { printf("请输入一个正整数: ");
    scanf("%d", &n);
    if(n<=0)
    { printf("输入的数据不能≤0, 请重新输入\n\n");
      continue;
    }
    if(isPrime(n)) printf("%d 是一个质数\n\n", n);
    else printf("%d 不是一个质数\n\n", n);
    printf("是否继续做判别质数的操作 (Y/N)?");
    fflush(stdin);              /*清除输入缓冲区中数据*/
    scanf("%c", &ans);
    if(ans=='N' || ans=='n') break; /*结束程序运行*/
  }
```

```

}
int isPrime(long n)                /*判别是否为质数的函数*/
{ long k;
  if(n==1) return 0;               /*1 不是质数*/
  if(n==2) return 1;               /*2 是质数*/
  if(n%2==0) return 0;             /*大于 2 的偶数不是质数*/
  for(k=3;k<n/2;k+=2)
    if(n%k==0) return 0;           /*能被任何小于 n/2 的奇数整除, 则 n 不是质数*/
  return 1;                        /*不能被任何小于 n/2 的奇数整除, 则 n 是质数*/
}

```

程序运行结果如图 1-5-1 所示。

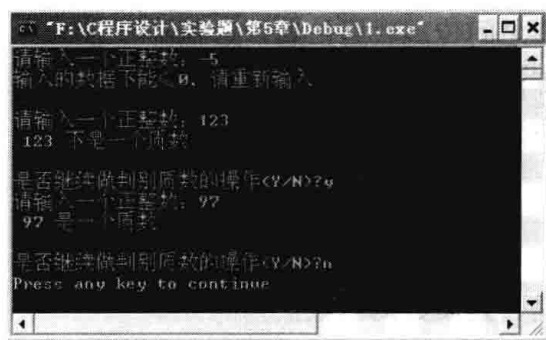


图 1-5-1 判断是否为质数的运行结果

【思考题】

(1) 上述判别是否是质数的方法非常简单, 但循环次数较多, 效率不高。如何修改 `isPrime(long n)` 函数, 使之提高判别质数的效率?

(2) 修改主函数, 当输入一正整数 n 时, 输出 n 以内的所有质数, 且每行显示 10 个质数。

2. 编写一个数制转换函数, 要求输入一个十进制数, 输出相应的二进制数、八进制数和十六进制数。

【解题思路】

数制转换函数 `convert()` 应设置两个形参 n 和 k , 其中, n 表示十进制数, k 表示要转换的是二进制数、八进制数或十六进制数。将一个正整数转换为 k 进制数, 只须不断地除以 k 取余, 直至 n 为 0 结束即可。

【参考程序】

```

#include<stdio.h>
void convert(int n,int k)
{ char a[100];                    /*存放转换后每个字符*/
  char b[]={'A','B','C','D','E','F'}; /*用于十六进制数*/
  int i;
  for(i=0;n;i++)
  { a[i]=n%k;                     /*取余*/
    n/=k;                         /*取商*/
  }
  printf("对应的%d进制数为: ",k);
  for(i--;i>=0;i--)
    if(a[i]<10)printf("%2d",a[i]); /*<10 按原样显示*/
}

```

```

        else printf("%2c",b[a[i]-10]);          /*>=10, 转换为字母显示*/
    printf("\n");
}
void main()
{ int n;
  printf("请输入一个十进制整数: ");
  scanf("%d",&n);
  convert(n,2);                               /*转换为二进制数*/
  convert(n,8);                               /*转换为八进制数*/
  convert(n,16);                              /*转换为十六进制数*/
}

```

程序运行结果如图 1-5-2 所示。

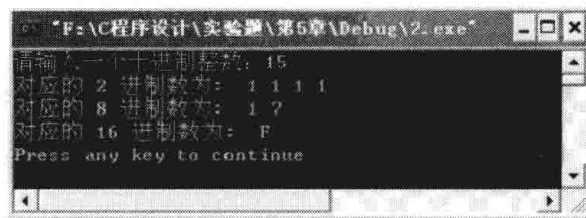


图 1-5-2 数制转换的运行结果

【思考题】

如何将一个二进制数、八进制数或十六进制数转换成等价的十进制数？

3. 编写判别是否为闰年的函数。

【解题思路】

判别是否为闰年的方法是：每 4 年有 1 个闰年，但每 100 年少 1 个闰年，每 400 年又增加 1 个闰年。也就是说，能被 4 整除，不能被 100 整除，是闰年；能被 4 和 100 整除，又能被 400 整除，也是闰年。

【参考程序】

```

#include<stdio.h>
int isLeap(int y)
{ return(y%4==0&& y%100!=0 || y%400==0);
}
void main()
{ int year;
  char ans;
  while(1)
  { printf("请输入一个年份: ");
    scanf("%d",&year);
    if(isLeap(year)) printf("%d 年是一个闰年\n",year);
    else printf("%d 年是一个平年\n",year);
    printf("是否继续做判别闰年的操作(Y/N)?");
    fflush(stdin);                               /*清除输入缓冲区中数据*/
    scanf("%c",&ans);
    if(ans=='N' || ans=='n') break;              /*结束程序运行*/
  }
}

```

程序运行结果如图 1-5-3 所示。

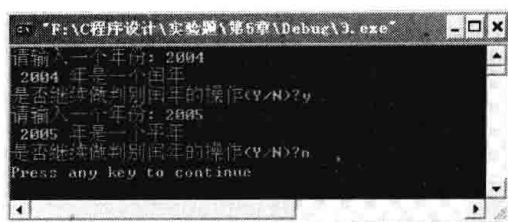


图 1-5-3 判断是否为闰年的运行结果

【思考题】

输入年、月、日，如何计算该日是该年的第几天？

4. 为低年级小学生编写一位数相乘的测验程序。当回答正确时，显示“回答得太棒了”，并产生下一个题目；当回答错误时，则显示“认真思考一下，答案到底多少？”，并要求重新回答；当回答为 0 时表示程序结束。

【解题思路】

程序利用随机函数产生两个 1~9 的整数，并给出算式让小学生输入回答。为使程序每次运行时产生的随机数不一样，在程序开始处的代码 `srand(time(NULL))` 为随机函数初始化。

【参考程序】

```
#include<stdio.h>
#include<stdlib.h>                /*有关随机库函数*/
#include<time.h>                  /*有关时间库函数*/
void main()
{ int x,y,answer;
  srand(time(NULL));              /*随机函数初始化*/
  printf("这是一个 1 位数乘法的小游戏，回答为 0 表示程序结束。\\n");
  while(1)
  {
    x=1+rand()%9;                 /*随机产生 1~9 的整数*/
    y=1+rand()%9;                 /*随机产生 1~9 的整数*/
    printf("\\n%d × %d = ",x,y);
    scanf("%d",&answer);
    if(answer==0) return;         /*程序运行结束*/
    while(1)
    { if(x*y==answer)
      { /*回答正确*/
        printf("回答得太棒了\\n");
        break;
      }
      else
      { /*回答错误*/
        printf("认真思考一下，答案到底多少？ ");
        scanf("%d",&answer);
        if(answer==0) return;    /*程序运行结束*/
      }
    }
  }
}
```

程序运行结果如图 1-5-4 所示。

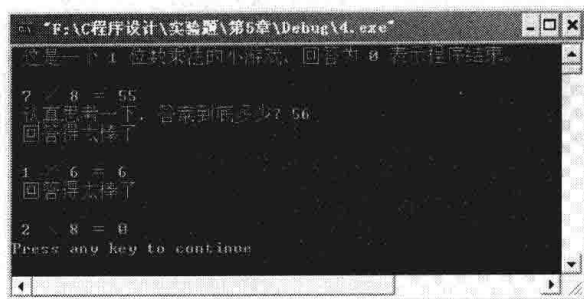


图 1-5-4 测验程序的运行结果

【思考题】

修改程序，使之可以随机测试加法、减法、乘法或除法运算，并可指定已知整数的范围。

5. 使用递归方法求解 Fibonacci 数列的前 n 项。

【解题思路】

Fibonacci 数列有以下公式定义：

$$\text{Fib}(n) = \begin{cases} 0 & \text{当 } n=0 \\ 1 & \text{当 } n=1 \\ \text{Fib}(n-2)+\text{Fib}(n-1) & \text{当 } n>1 \end{cases}$$

从上述公式可以看出：Fibonacci 数列的第 1 个数是 0，第 2 个数是 1，从第 3 个数开始，以后每一个数是前两个数之和。

【参考程序】

```
#include<stdio.h>

int fib(long n)                /*计算 Fibonacci 数列的递归函数*/
{ if(n==0) return 0;          /*第 1 个数为 0*/
  if(n==1) return 1;          /*第 2 个数为 0*/
  return fib(n-1)+fib(n-2);    /*递归调用*/
}

void main()                    /*主函数*/
{ int i,n;
  printf("请输入一个正整数: ");
  scanf("%d",&n);
  printf("前%d个 Fibonacci 数列如下:\n",n);
  for(i=0;i<n;i++)
  { if(i%5==0)printf("\n");    /*每行输出 5 个数*/
    printf("%10d",fib(i));      /*调用计算 Fibonacci 数列的递归函数*/
  }
  printf("\n");
}
```

程序运行结果如图 1-5-5 所示。

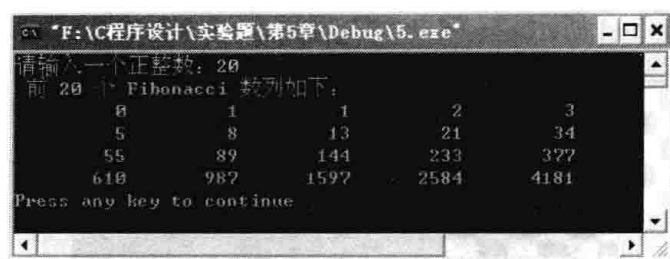


图 1-5-5 Fibonacci 数列的运行结果

【思考题】

不用递归方法, 使用递推法如何计算 Fibonacci 数列?

6. 使用递归方法实现奇数之和 $1+3+5+\dots$ 或 $2+4+6+\dots$; 如果从键盘上输入的 n 是奇数则计算奇数之和, 输入的 n 是偶数则计算偶数之和。

【解题思路】

为计算奇数或偶数之和, 在递归函数中要不断地调用自己直到 $n=1$ (奇数之和递归) 或 $n=2$ (偶数之和递归) 为止, 然后在返回中得到 $1+3+5+\dots$ 或偶数之和 $2+4+6+\dots$ (在递归函数中依次得到), 最后才将得到的结果返回到主函数。

【参考程序】

```
#include<stdio.h>

int f1(int n)                /*实现奇数之和的递归函数*/
{ if(n==1) return 1;
  else return n+f1(n-2);
}

int f2(int n)                /*实现偶数之和的递归函数*/
{ if(n==2) return 2;
  else return n+f2(n-2);
}

void main()
{ int n;
  printf("请输入一个大于 0 的正整数, 计算奇数之和或偶数之和: ");
  scanf("%d",&n);           /*正整数输入*/
  while(n<=0){
    printf("输入的正整数不能≤0, 请重新输入: ");
    scanf("%d",&n);         /*输入正整数小于 0, 重新输入*/
  }
  if(n%2==1)                 /*判定 n 是奇数还是偶数*/
    printf("1 加到%d 的奇数之和为: %d\n",n,f1(n));
  else printf("2 加到%d 的偶数之和为: %d\n",n,f2(n));
}
```

程序运行结果如图 1-5-6 所示。

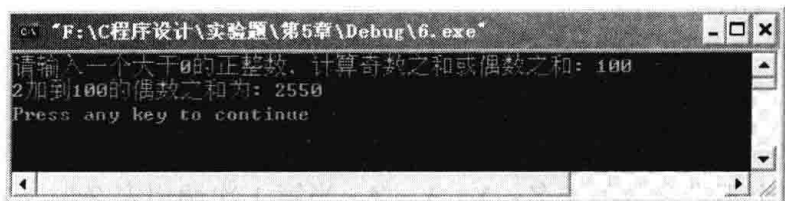


图 1-5-6 递归法运用的运行结果

【思考题】

使用递归实现求 $1+2+3+\cdots+n$, 如何修改上述程序来实现?

7. 编写从 3 个整数中求出最大数的宏定义。

【解题思路】

此题的解答显然是要用带参数的宏定义。在宏定义中, 字符序列中出现的形参应该用圆括号括起来, 否则, 实参为表达式时返回结果可能不正确。

程序可以先定义一个从两个数中求最大值的宏定义 MAX2, 然后, 利用这个宏定义再定义从 3 个数中求最大值的宏定义 MAX3。

【参考程序】

```
#include<stdio.h>
#define MAX2(A,B) ((A)>(B)?(A):(B)) /*从两个数中求最大值的宏定义*/
#define MAX3(A,B,C) MAX2(MAX2((A),(B)),(C)) /*从三个数中求最大值的宏定义*/
void main()
{ int x, y, z;
  printf("请输入三个整数: ");
  scanf("%d%d%d",&x,&y,&z);
  printf("最大值为: %d\n",MAX3(x,y,z));
}
```

程序运行结果如图 1-5-7 所示。

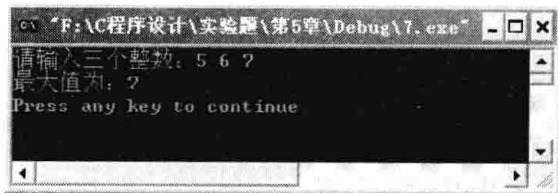


图 1-5-7 求最大数宏定义的运行结果

【思考题】

(1) 在宏定义中, 为什么要将参数 A 和参数 B 加括号?

(2) 修改宏定义, 使宏定义 MAX 有 3 个形参, 而不用带两个形参的宏定义, 直接求出 3 个整数中最大值。

8. 分析并写出下列程序运行后的结果。

```
#include<stdio.h>
#define X 5
#define Y X+1
#define Z Y*Y/2
void main()
```

```
{ printf("Z = %d\n", Z);  
  printf("Z*Z= %d\n", Z*Z);  
}
```

【解题思路】

程序中定义了 3 个不带参数的宏名: X, Y, Z。当第 1 次调用宏 Z 时, Z 代换成 $Y \times Y/2$, 再将 Y 代换成 $X+1$ 变为: $X+1 \times X+1/2$, 进一步将 X 代换成 5 又变成: $5+1 \times 5+1/2$, 因此, 第 1 个输出结果是 10。同样, 第 2 次调用宏 Z、Y 和 X 时, 被展开为: $5+1 \times 5+1/2 \times 5+1 \times 5+1/2$, 因此, 第 2 个输出结果是 15。

程序运行结果如图 1-5-8 所示。

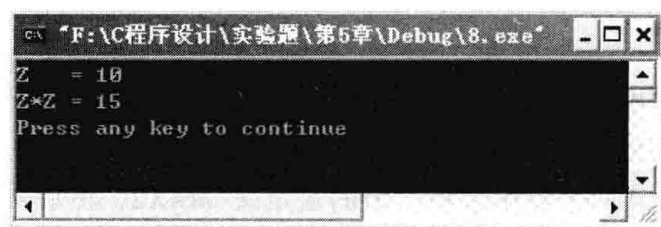


图 1-5-8 宏运用的运行结果

【思考题】

如果将上述宏定义 `#define Y X+1` 改为 `#define Y (X+1)`, 程序运行结果会改变吗? 请说出具体原因。

三、实验小结

函数设计能力是程序设计能力的一个重要方面。函数设计又包含两个方面, 即合理设定函数的形参和为函数设计有效的算法。通过实例可以提高这两个方面的能力。宏定义是 C 语言为简化程序而提供的一种编写程序的措施, 合理应用宏定义对编写应用程序会有很大帮助。

实验 6 | 指针和引用

一、实验目的

1. 掌握地址和指针的基本概念，掌握一级指针的定义和使用。
2. 熟悉指针与数组之间的关系。
3. 掌握字符数组、字符指针的处理方法。
4. 了解数组指针、指针数组以及二级指针的概念和应用。
5. 掌握指针形参的作用以及能够熟练编写指针形参的函数。
6. 了解引用的概念、引用与指针的区别及引用形参的用法。

二、实验内容

1. 编写程序，将已知数组内容复制到新数组，使复制产生的新数组中包含已知数组全部出现过的值，而又不重复。例如，已知数组中的内容为 6、5、8、7、5、6、4、3，复制到新数组中后变为 6、5、8、7、4、3，因为 5 和 6 都重复出现过，有相同的只复制一次。

【解题思路】

程序首先定义两个数组 `a[]` 和 `b[]`，其中 `a[]` 是已知数组，`b[]` 是存放数组 `a[]` 中不重复数据的新数组，然后再分别定义指向 `a[]` 和 `b[]` 两个数组的指针变量 `pa` 和 `pb`（当然本题直接可以使用数组元素进行比较来实现，但实验 6 中都要求用指针实现）以及临时指针变量 `pc`。编写程序的思路如下：

（1）外循环依次取已知数组 `a[]` 中的每一个元素，直至数组结束。

（2）内循环用于考察数组 `a[]` 的当前元素是否曾在新数组 `b[]` 中出现过（即该值已经复制了）。考察的方法是：先将当前考察的元素存于新数组当前最后位置，然后再依次与新数组 `b[]` 的每个元素进行比较，有以下两种情况：

① 若当前元素与数组 `b[]` 当前最后一个元素相等，用代码 `pc==pb` 判别成立（即自己与自己相等），则表示未重复，将 `pb` 增 1，表示确认开始时的复制。

② 若当前元素与之前的某一个元素相等，用代码 `pc==pb` 判别不成立，则表示之前已有相同的数值复制过，当然不要再将当前数组元素复制到新数组 `b[]` 中，`pb` 不增 1，即开始时的复制不被确认。

【参考程序】

```
#include<stdio.h>
#define N 100
void main()
{ int a[]={6,5,8,7,5,6,4,3},b[N];
  int n=8;                               /*n 表示数组 a[] 中有效元素个数*/
  int *pa,*pb,*pc;                       /*定义指针变量 pa、pb 和 pc*/
```

```

for (pa=a, pb=b; pa-a<n; pa++)
{
    *pb=*pa;                                /*先复制, 存于新数组的当前最后位置*/
    for (pc=b; *pc!=*pb; pc++);             /*判别数组 a[] 当前元素之前是否出现过*/
    if (pc==pb)                             /*出现的就是当前元素, 表示之前未出现过*/
        pb++;                             /*确认开始时的复制*/
    }                                       /*否则, 不确认早先的复制*/
printf("已知数组中的数据有:");           /*显示已知数组内容*/
for (pa=a; pa-a<n; pa++) printf("%5d", *pa);
printf("\n\n");
printf("复制到新数组数据有:");           /*显示新数组中内容*/
for (pc=b; pc<pb; pc++) printf("%5d", *pc);
printf("\n");
}

```

程序运行结果如图 1-6-1 所示。

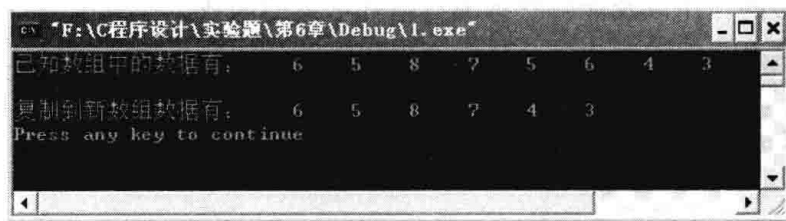


图 1-6-1 复制数据的运行结果

【思考题】

(1) 若将上述题目修改为: 将已知数组内容复制到新数组, 使复制产生的新数组中包含已知数组全部出现过的值, 而又不重复, 并使之成为有序数组, 如何修改上述程序?

(2) 将实现上述要求的复制编写成函数, 函数返回新数组的元素个数。

2. 不使用字符串连接函数 `strcat()`, 将两个字符串连接后输出。

【解题思路】

程序设有两个存储字符串的数组 `s1[]` 和数组 `s2[]`, 先从键盘上分别为两个字符数组输入字符串, 然后找到第 1 个字符串 `s1` 的末尾, 接着将第 2 个字符串 `s2` 复制在第 1 个字符串 `s1` 的后面, 最后输出。

【参考程序】

```

#include<stdio.h>
#define N 120
void main()
{
    char s1[N+N], s2[N], *p, *q;           /*定义字符数组及指针变量*/
    printf("请输入第 1 个字符串: ");
    gets(s1);
    printf("请输入第 2 个字符串: ");
    gets(s2);
    for (p=s1; *p!='\0'; p++);             /*找到字符串 s1 的末尾*/
    for (q=s2; *p++=*q++);                 /*将字符串 s2 复制 s1 的后面*/
    printf("两字符串连接后结果: %s\n", s1);
}

```

程序运行结果如图 1-6-2 所示。

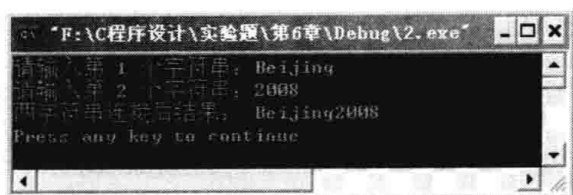


图 1-6-2 连接两个字符串的运行结果

【思考题】

(1) 修改上述程序, 将两字符串连接编写成一个函数, 函数的返回值是连接后字符串的长度。主函数通过调用函数实现两字符串的连接, 并输出结果。

(2) 编写删除字符串中重复出现字符的函数。例如, 输入的字符串为"ABCB CD", 输出的结果应为"ABCD"。

3. 编写字符串加密的函数。加密规则如下:

(1) 若当前字符不是字母则不加密, 直接复制该字符到新字符串中。

(2) 若当前字符是一个字母, 则按表 1-6-1 所示转换。

表 1-6-1 字母密码转换表

A ↔ Z	H ↔ S	a ↔ z	h ↔ s
B ↔ Y	I ↔ R	b ↔ y	i ↔ r
C ↔ X	J ↔ Q	c ↔ x	j ↔ q
D ↔ W	K ↔ P	d ↔ w	k ↔ p
E ↔ V	L ↔ O	e ↔ v	l ↔ o
F ↔ U	M ↔ N	f ↔ u	m ↔ n
G ↔ T		g ↔ t	

例如, 输入的字符串为 BeiJing 2008, 加密后变为 YvrQrmt 2008。

【解题思路】

此题编程有多种方法。最简单的方法是设置两个字符数组 str1[]和 str2[]。str1[]存放加密的大写字母, str2[]存放加密的小写字母。当循环考察字符串时, 有以下 3 种情况:

(1) 若当前的字符是大写字母, 只要将这个大写字母减去'A'或 65 (字母 A 的 ASCII 码是十进制数 65) 作为 str1[]字符数组的下标或位移量, 然后再取其值即可。

(2) 若当前的字符是小写字母, 只要将这个 小写字母减去'a'或 97 (字母 a 的 ASCII 码是十进制数 97) 作为 str2[]字符数组的下标或位移量, 然后再取其值。

(3) 若当前的字符不是字母, 不需要加密转换。

【参考程序】

```
#include<stdio.h>
#define N 1000
/*数组 str1[], str2[] 分别存放加密的大、小写字母*/
char str1[]="ZYXWVUTSRQPONMLKJIHGFEDCBA";
char str2[]="zyxwvutsrqponmlkjihgfedcba";
void encode(char *ip, char *op)          /*加密函数*/
{ char ch;
  while(ch=*ip++)                        /*当前字符不是结束符时循环*/
    if(ch>='A'&&ch<='Z')
```

```

        *op++=str1[ch-'A'];           /*当前字符是大写字母*/
    else if(ch>='a'&&ch<='z')
        *op++=str2[ch-'a'];         /*当前字符是小写字母*/
    else *op++=ch;                   /*当前字符不是字母, 不需要加密*/
    *op='\0';                         /*新数组结束加结束符*/
}
void main()
{ char in[N],out[N];
  printf("请输入一个字符串: ");
  gets(in);
  encode(in,out);                    /*调用加密函数*/
  printf("加密后的字符串为: %s\n",out);
}

```

程序运行结果如图 1-6-3 所示。

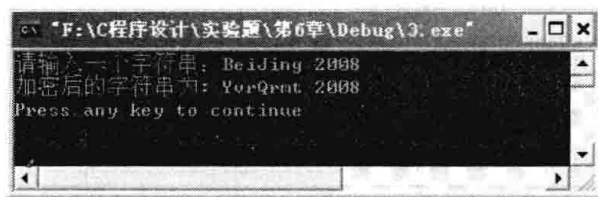


图 1-6-3 加密字符串的运行结果

【思考题】

(1) 在上述程序中增加解密函数。

(2) 不使用字符数组 `str1[]` 和 `str2[]`, 如何使用 ASCII 编码技术实现上述功能?

4. 输入一个字符串要求, 输出字符串中的每个字符, 并在每个字符之间用逗号加空格分隔。例如, 输入的字符串为 "abcd", 则输出应为 a, b, c, d。要求字符之间加逗号和空格, 而最后一个字符后面不应有逗号和空格, 并要求用函数实现。

【解题思路】

主函数 `main()` 十分简单, 先定义两个字符串数组: `str1[]` 存放输入的字符串, `str2[]` 存放结果的字符串, 然后调用字符之间加逗号和空格函数 `insComma()` 即可。在 `insComma()` 函数中, 用两个指针形参变量 `p1` 和 `p2` 指向两个实参字符串数组。循环考察指针变量 `p1` 是否遇到字符串结束符 `'\0'`, 若还未遇到, 则将当前字符加上逗号和空格送入新字符串数组中。在程序中须注意的是, 循环结束最后一个字符后也加上了一个逗号和空格, 应删除。

【参考程序】

```

#include<stdio.h>
#include<string.h>
#define N 80
void insComma(char *p1,char *p2)      /*字符间加逗号和空格函数*/
{ char *p;                             /*定义临时指针变量*/
  for(p=p1;*p;p++)
  { /*反复循环直至遇到结束符'\0'*/
    *p2++=*p;*p2++=', '*p2++=' ';
  }
  *(p2-2)='\0';                        /*设置字符串结束符*/
}
void main()
{ char str1[N],str2[N*3];              /*定义字符数组*/

```

```

printf("输入字符串: ");
gets(str1);
insComma(str1, str2);          /*调用插逗号和空格函数*/
printf("字符间插入逗号和空格后结果为: %s\n", str2);
}

```

程序运行结果如图 1-6-4 所示。

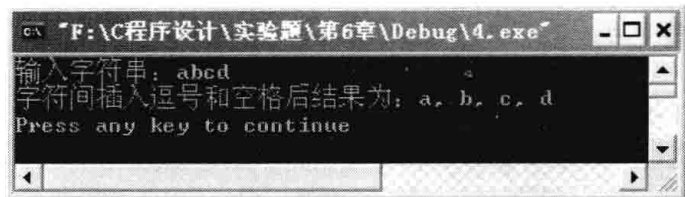


图 1-6-4 字符间插入逗号的运行结果

【思考题】

如何按输入字符串的逆序输出到新字符串数组中，字符之间仍须要加逗号和空格？例如，输入的字符串为"abcd"，则输出应为 d, c, b, a。

5. 整理字符串，将字符串中前导空白符和随后的空白符删除，中间连续的多个空白符只保留一个，去掉多余的空白符。

【解题思路】

程序将所有字符分成两类：空白类字符和非空白类字符。为了区别不同的字符作不同的处理，程序引入状态变量 s ，用于记录程序正在分析中的状态。在初始状态， s 为 0，遇到空白类字符，继续保持状态为 0；遇到非空白类字符，程序复制字符，并转入状态 1。在状态 1 情况下（上一个字符为非空白字符），遇到空白类字符，转入状态 2；遇到非空白类字符则复制字符，并继续保持状态 1。在状态 2 情况下（上一个字符为空白字符），遇空白类字符，继续保持状态 2；遇非空白类字符，转入状态 1，并插入一个空白符和复制字符。

综合以上 3 种情况，对于非空白类字符，在状态为 2 情况下须插入复制一个空白符，还须要复制当前字符，并置状态为 1。对于空白类字符，如果程序状态为 1，则要置程序状态为 2，其余状态继续保持原来状态，并忽略这个空白字符。

程序另设置两个指针变量 p 和 q ，初始状态都是指向字符串的开始位置。其中， q 指向要考察的字符，每考察完后，指向下一个字符； p 指向结果字符串。

【参考程序】

```

#include<stdio.h>
char *parse(char *str)          /*删除多余空白字符函数*/
{ int s=0, isSpace;              /*s 为状态变量*/
  char *p=str, *q=str;          /*指针变量 p 和 q 指向字符串*/
  while(*q)
  { /*当字符串未处理完，反复循环*/
    isSpace=*q==' '||*q=='\t'||*q=='\n';
    if(!isSpace)
    { /*当前字符不是空白字符*/
      if(s==2) *p++=' ';
      s=1; *p++=*q;
    }
    else if(s==1) s=2;          /*当前字符是空白字符*/
  }
}

```

```

        q++;
    }
    *p='\0';
    return str;
}
void main()
{
    char s[120]=" Welcome to C World!";
    printf("原始字符串为: %s\n",s);
    printf("结果字符串为: %s\n",parse(s));
}

```

程序运行结果如图 1-6-5 所示。

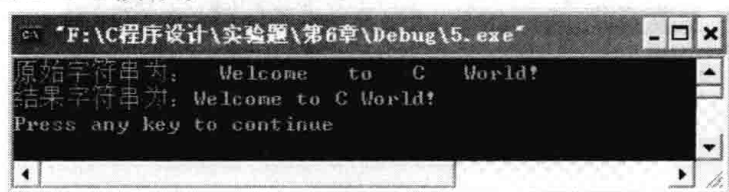


图 1-6-5 整理字符串的运行结果

【思考题】

改写 parse()函数, 将其中的指针形参改为引用形参。

6. 已知有一个一维数组 a[], 其初值为: 1、2、3、...、12。使用指针数组将这个一维数组 a[] 转换按 3×4 的二维格式输出。

【解题思路】

将一维数组按 3×4 的二维格式输出, 就是要找到“二维”数组每一行的首地址。程序设置指针数组 p, 使其每一个元素指向“二维”数组每一行的首元素, 即 p[0]指向 a[0], p[1]指向 a[4], p[2]指向 a[8]。当然, 在程序中使用循环来实现: p[i]=&a[N*i];(其中, i 表示第 i 行, N 代表列数即有 4 列), 最后依次输出 p[i][j]即可 (其中, i 表示第 i 行, j 表示第 j 列)。

【参考程序】

```

#include<stdio.h>
#define M 3
#define N 4
void main()
{
    int i,j,a[M*N], *p[M];
    for(i=0;i<M*N;i++) a[i]=i+1;
    printf("一维数组如下:\n");
    for(i=0;i<M*N;i++)
        printf("%4d",a[i]);
    for(i=0;i<M;i++)
        p[i]=&a[N*i];
    printf("\n 按 %d×%d 输出如下:\n",M,N);
    for(i=0;i<M;i++)
    {
        for(j=0;j<N;j++)
            printf("%4d",p[i][j]);
        printf("\n");
    }
}

```

程序运行结果如图 1-6-6 所示。

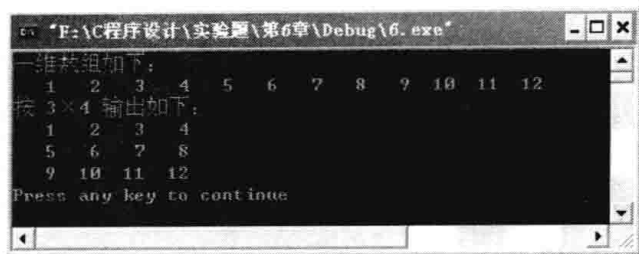


图 1-6-6 一维数组转换的运行结果

【思考题】

按 3×4 的二维格式输出后，如何再将矩阵转置输出（见图 1-6-7）？



图 1-6-7 改后运行结果

7. 编写一个函数，按所给的百分制的成绩分数，返回与该分数对应的等级代号字符。百分制的成绩分数与等级代号字符的对照如表 1-6-2 所示。

表 1-6-2 成绩分数与等级代号对照表

百分制成绩	90~100	85~89	82~84	78~81	75~77
等级	A	A-	B+	B	B-
百分制成绩	71~74	66~70	62~65	60~61	60 以下
等级	C+	C	C-	D	F

【解题思路】

引入两个数组：一个是字符指针数组 `order[]`，指针数组 `order[]` 的每个元素也是指针，指向表示等级的字符串；另一个是整型数组 `mark[]`，整型数组 `mark[]` 存放各等级对应分数线的底线。等级底线的分数与对应的等级字符串在数组下标的位置应一致，这样，函数只要检查分数等级表，确定是属于第几个等级，然后取对应等级的字符串指针即可。

【参考程序】

```
#include<stdio.h>
#include<string.h>
#define N 80
char *markToOrder(int m) /*百分制成绩转换为等级函数*/
{ int k;
  int mark[]={90,85,82,78,75,71,66,62,60,-1};
  char *order[]={ "A", "A-", "B+", "B", "B-", "C+", "C", "C-", "D", "F"};
  for(k=0;mark[k]>=0;k++)
    if(m>=mark[k]) return order[k];
}
```



```

    return order[9];
}
void main()
{ int score;                                /*定义字符数组*/
  while(1)
  { printf("输入百分制成绩: ");
    scanf("%d",&score);
    if(score<0) printf("输入的成绩不能<0, 请重新输入\n");
    else break;
  }
  printf("对应的等级为:  %s\n",markToOrder(score));
}

```

程序运行结果如图 1-6-8 所示。

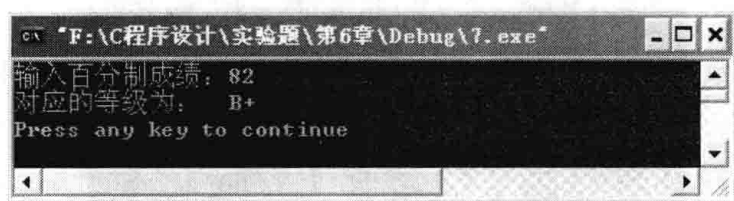


图 1-6-8 成绩分等级的运行结果

【思考题】

百分制成绩转换为等级函数*markToOrder(int m)前面为什么要加星号*, 如果没有星号是否可以实现?

三、实验小结

通过以上实验说明这样的事实: 在一般情况下, 利用指针间接引用变量, 或用指针处理数组并没有比直接引用变量, 或用下标处理数组有更多的方便之处。指针真正的应用是在函数中设置指针形参, 一般的指针形参能将计算结果存于实参所指定的变量中; 在函数中设置数组形参, 能使函数处理实参指定的数组; 在函数中设置函数指针形参, 函数进一步要调用的函数也能由实参指定。上述 3 种应用几乎会出现在每个应用程序中, 所以, 要用 C 语言编写程序, 必须掌握指针的这 3 种应用。为了克服指针传递地址和间接引用的复杂性, C++重新引入引用的概念, 在很多情况下, 引用形参能实现指针形参的功能, 但比指针形参应用更简便。

指针在 C 语言的使用中非常灵活、方便, 但比较难掌握。本实验作如下的小结:

1. 指向变量的指针。例如:

```

int x,*p;
p=&x;                                /*指针变量 p 指向变量 x*/

```

2. 指向一维数组元素的指针变量。例如:

```

int a[100],*p;
p=a; 或 p=&a[0];                    /*p 指向数组首地址*/

```

说明: $p+i$ 或 $a+i$ 都表示 $\&a[i]$ 。

$*(p+i)$ 或 $p[i]$ 或 $*(a+i)$ 都表示 $a[i]$ 。

3. 指向二维数组的指针变量。例如：

```
int a[3][4], (*p)[4];  
p=a;                      /*p 指向二维数组首地址*/
```

说明：*(p+i)+j 或 p[i]+j 或 a[i]+j 或 *(a+i)+j 都表示 &a[i][j]。

((p+i)+j)或*(p[i]+j)或*(a[i]+j)或*(*(a+i)+j)都表示 a[i][j]。

4. 指向字符串的指针变量。例如：

```
char *p;  
p="I am a string.";       /*p 指向字符串的首地址*/
```

5. 指针数组。例如：

```
int a,b,c;  
int *p[]={&a,&b,&c};      /*p 有 3 个元素，每个元素是一个指针*/
```

6. 多级指针。例如：

```
int x,*p,**pp;  
p=&x;                      /*p 指向变量 x*/  
pp=&p;                     /*pp 指向指针变量 p*/
```

说明：*p 或 **pp 都是访问变量 x。

实验 7 | 结构和链表

一、实验目的

1. 掌握结构数据类型、结构变量和结构指针的定义以及结构成分的引用方法。
2. 掌握结构数组的定义和使用方法。
3. 掌握定义带结构形参、结构指针形参、返回结构的函数的方法。
4. 掌握链表的基本操作，并能熟练编写链表应用程序。
5. 了解联合、枚举数据类型的概念和使用方法。
6. 了解类型定义和变量定义的方法。

二、实验内容

1. 利用结构数组，输入 3 个学生的姓名、语文成绩和数学成绩，然后计算 3 个学生的语文平均成绩和数学平均成绩并输出。

【解题思路】

首先要定义一个结构类型，例如 `stuScore`，该类型有 3 个成员：字符数组 `name[20]` 表示学生的姓名，整型变量 `chinese` 和 `math` 分别表示学生的语文成绩和数学成绩，然后定义一个结构数组，用于存放 3 个学生的信息。通过循环语句分别输入 3 个学生的姓名、语文成绩和数学成绩就可以统计出这 3 个学生的语文平均成绩和数学平均成绩了。为了不限定具体的学生人数，在程序前增加了一条宏定义命令 `#define N 3`，方便以后修改学生数。

【参考程序】

```
#include<stdio.h>
#define N 3
struct stuScore
{ char name[20];           /*姓名*/
  int chinese;             /*语文成绩*/
  int math;                /*数学成绩*/
};
void main()
{ int i,ywsum=0,sxsum=0;
  struct stuScore st[N];   /*定义结构数组，含 N 个元素*/
  for(i=0;i<N;i++)
  { /*依次输入每位学生的信息*/
    printf("请输入第 %d 位学生的姓名、语文成绩、数学成绩\n",i+1);
    scanf("%s%d%d",st[i].name,&st[i].chinese,&st[i].math);
  }
  for(i=0;i<N;i++)
  { ywsum+=st[i].chinese;   /*累加语文成绩*/
    sxsum+=st[i].math;      /*累加数学成绩*/
  }
```

```
}  
printf("\n 语文平均成绩\t数学平均成绩\n");  
printf("%8.2f\t%8.2f\n\n", (float)ywsum/N, (float)sxsum/N);  
}
```

程序运行结果如图 1-7-1 所示。

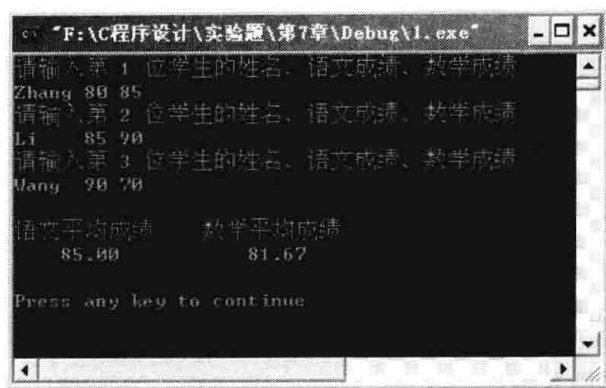


图 1-7-1 计算平均成绩的运行结果

【思考题】

- (1) 结构数组的引用有几种方法？请使用不同的结构数组的引用方法实现上述程序的功能。
- (2) 如果要计算每位学生的平均成绩，如何修改程序？

2. 建立一个学生的链表，每个表元包括：学号、姓名、成绩。要求制作一个菜单，可以在链表末尾添加表元、删除指定表元、显示链表中每个表元的程序。

【解题思路】

在程序开始处首先要定义一个结构类型，例如 `stuS`，该类型有 4 个成员：整型变量 `number` 表示学生的学号，字符数组 `name[20]` 表示学生的姓名，整型变量 `score` 表示学生的成绩，指针变量 `next` 指向下一个表元。

(1) 主函数 `main()` 的设计：

- ① 设置空链表。
- ② 设置一个指针数组 `menu[]`，用于存放菜单项命令。
- ③ 循环显示菜单选项命令，让用户选择要做的操作。

(2) 在链表末尾添加表元函数 `append()` 的设计。

设要插入的新表元由指针 `p` 所指，将 `p` 所指表元添加到链表末尾的操作可由以下 3 个操作步骤完成：

- ① 从头指针开始依次找到最后的表元，由指针 `w` 指向最后的表元。
- ② 使指针 `p` 所指表元成为指针 `w` 所指表元的后继表元，用代码 `w->next=p` 实现。
- ③ 将新表元的后继表元设置为空，用代码 `p->next=NULL` 实现。

(3) 删除指定表元函数 `delStu()` 的设计。

首先要在链表中查找要删除的表元：如果删除的是首表元，则要修改链表头指针；如果删除的表元不是链表的首表元，则要更改删除表元的前驱表元的后继指针。

假设指针变量 `p` 的初值指向首表元，通过循环，`p` 依次指向下一个表元（用 `p=p->next` 实现）；指针变量 `w` 总是指向 `p` 的前驱表元（在 `p` 指向下一个表元之前，先将 `p` 保存在指针变量 `w` 中，用 `w=p` 实现）。

(4) 显示链表中每个表元函数 `travelStuLink()` 的设计。

从链表的首表元开始, 沿着表元的链接顺序逐一考察链表的每个表元, 直至链表结束。设置指针 `p` 的初值指向链表的头指针, 循环条件是指针 `p` 还指向某个表元, 即 `p` 不等于 `NULL`。循环要做的工作包括输出 `p` 所指向的表元的值和准备考察下一个表元 (用 `p=p->next` 实现)。

【参考程序】

```
#include<stdio.h>
#include<string.h>
#include<malloc.h>
struct stuS
{ int number;           /*学号*/
  char name[20];        /*姓名*/
  int score;            /*成绩*/
  struct stuS *next;
};                       /*其他函数应插在主函数之前*/
/*在链表末尾添加新表元函数*/
struct stuS *append(struct stuS *head)
{ struct stuS *w=head, *p;
  p=(struct stuS *)malloc(sizeof(struct stuS)); /*p 指向新申请的表元*/
  printf("请输入学号、姓名、成绩: ");
  scanf("%d%s%d", &p->number, p->name, &p->score);
  p->next=NULL;          /*p 将成为链表的最后一个表元*/
  if(head==NULL) return p; /*原链表是空链表, 直接返回新表元指针*/
  while(w->next!=NULL)    /*非空链表, 顺序向后找尾表元*/
    w=w->next;           /*后移一个表元, 直到 w 指向尾表元*/
  w->next=p;             /*p 作为 *w 的后继表元*/
  return head;           /*返回头指针*/
}

/*删除指定表元函数*/
struct stuS *delStu(struct stuS *head, int num) /*num 为要删除学生的学号*/
{ struct stuS *w, *p;
  if(head==NULL) return NULL;
  p=head;               /*让 p 指向链表的首表元*/
  while(p!=NULL&& p->number!=num)
  { w=p; p=p->next;     /*w 指向前驱表元, p 指向下一个表元*/
  }
  if(p!=NULL)
  { /*找到*/
    if(p==head) head=p->next; /*删除的是首表元, 修改链表头指针*/
    else w->next=p->next;      /*修改前导表元的后继指针*/
    printf("学号为 %d 的学生记录已被删除\n", num);
    free(p);                  /*释放被删表元的存储空间*/
  }
  else printf("学号 %d 找不到! \n", num);
  return head;               /*返回头指针*/
}

/*显示链表中每个表元函数*/
void travelStuLink(struct stuS *head)
{ struct stuS *p=head;
  if(head==NULL)
  { printf("\n 目前在链表中没有学生\n");
    return;
  }
```

```

    }
    printf("\n 目前在链表中的学生如下\n");
    while(p!=NULL)
    { printf("%d\t%s\t%d\n",p->number,p->name,p->score);
      p=p->next;                      /*准备访问下一个表元*/
    }
}
/*主函数*/
void main()
{ struct stuS *head=NULL;
  char *menu[]={"0 退出","1 在链表末尾添加新表元",
               "2 显示链表中所有表元","3 删除链表中指定表元",""};
  int i,ans,number;
  while(1)
  { printf("\n 请选择下列菜单命令:\n");
    for(i=0;menu[i][0]!='\0';i++)      /*显示在链表上操作的菜单*/
      printf("\t%s\n",menu[i]);
    printf("    请选择 0~3: ");
    scanf("%d",&ans);
    switch(ans)
    { case 1:head=append(head); break;
      case 2:travelStuLink(head); break;
      case 3:printf("请输入要删除记录的学生学号: ");
              scanf("%d",&number);
              head=delStu(head,number);break;
      default:return;
    }
  }
}

```

程序运行结果如图 1-7-2 所示。

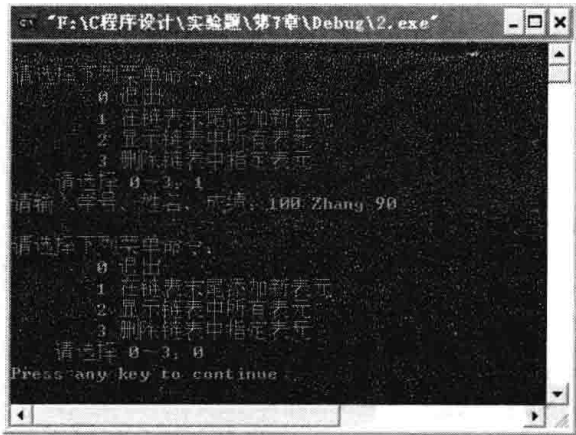


图 1-7-2 建立链表的运行结果

【思考题】

增加一个函数，其功能是在指定的表元之后插入一个新表元。

3. 编写一个函数，实现将一个已知链表的表元链接顺序颠倒，即使链表的第 1 个表元变为最后一个表元，第 2 个表元变为倒数第 2 个表元，……，最后一个表元变为第 1 个表元。

【解题思路】

颠倒操作是一个循环过程, 即从第 1 个表元开始颠倒, 直到最后一个表元颠倒完成。每循环一次颠倒一个表元的链接关系, 在程序开始处首先要定义一个结构类型。例如, `intNode` 有 2 个成员: 整型变量 `value` 用于存放整型值, 指针变量 `next` 用于指向下一个表元。本程序设置了 3 个函数: 主函数 `main()`、链表颠倒函数 `reverse()` 和输出链表每个表元函数 `output()`。

(1) 主函数 `main()` 的设计:

① 循环输入整型数据, 每输入一个整型数据产生一个表元, 并将输入的值存放到该表元中。若输入的是非数字, 如 `end`, 则输入结束。

② 调用输出函数 `output()`, 显示新建立链表中每个表元的值。

③ 调用链表颠倒函数 `reverse()` 和输出函数 `output()`, 颠倒链表并显示链表颠倒后的每个表元的数值。

(2) 输出链表每个表元函数 `output()` 的设计。

从链表首表元开始, 循环显示每个表元的数值, 直到最后一个表元为止。

(3) 链表颠倒函数 `reverse()` 的设计:

① 设置 3 个结构指针变量: `v2` 指向还未完成颠倒的部分链表的首表元, 它所指的正是当前要颠倒的表元(初值指向首表元, 用代码 `v2=h` 实现), `v1` 指向已完成颠倒的部分链表的首表元(初值为 `NULL`), `p` 是工作指针, 用于指向 `v2` 所指表元的后继表元(用代码 `p=v2->next` 实现)。

② 循环处理每一个表元, 当前表元由 `v2` 指向, `v2` 的后继表元由 `p` 指向。颠倒要使 `v2` 所指表元指向 `v1` 所指向的表元。颠倒当前表元的操作是: 将 `*v2` 原指向后继表元的指针改为指向 `*v1` (用代码 `v2->next=v1` 实现); `v1` 指向 `v2` 所指表元 (用代码 `v1=v2` 实现); `v2` 指向 `p` 所指表元 (用代码 `v2=p` 实现); 直到最后一个表元颠倒完成 (`v2==NULL`)。

③ 循环结束后, 让链表的头指针指向原最后一个表元, 即原最后一个表元应变为首表元, 用代码 `h=v1` 实现。

【参考程序】

```
#include<stdio.h>
#include<malloc.h>
struct intNode
{ int value;
  struct intNode *next;
};

/*链表颠倒函数*/
struct intNode *reverse(struct intNode *h)
{ struct intNode *p,*v1,*v2;
  v2=h; /*v2 指向当前要颠倒的表元, 初值指向首表元*/
  v1=NULL; /*v1 指向 v2 前一个表元, 初值指向 NULL*/
  while(v2!=NULL)
  { p=v2->next; /*p 指向 v2 的后继表元*/
    v2->next=v1; /*v2 所指表元的后继表元是 v1 所指表元*/
    v1=v2; /*调整 v1 的指向, 即指向下一个表元*/
    v2=p; /*调整 v2 的指向, 即指向下一个表元*/
  }
}
```

```

    h=v1;                                /*原链表的最后一个表元变为首表元*/
    return h;
}
/*输出链表中表元函数*/
void output(struct intNode *h)
{ struct intNode *p=h;
  while(p!=NULL)
  { printf(" %d\t",p->value);
    p=p->next;                            /*准备访问下一个表元*/
  }
}
/*主函数*/
void main()
{ struct intNode *h=NULL,*p,*w;
  int n;
  printf("请输入若干个整型数据(按 end 结束输入):\n");
  while(scanf("%d",&n))
  { p=(struct intNode *)malloc(sizeof(struct intNode));
    p->value=n;
    p->next=NULL;
    if(h==NULL) h=p;
    else
    { w=h;
      while(w->next!=NULL)                /*非空链表,顺序向后找尾表元*/
        w=w->next;                        /*后移一个表元,直到w指向尾表元*/
      w->next=p;                          /***p 作为*w 的后继表元*/
    }
  }
  printf("\n 颠倒前的链表:\n");
  output(h);
  h=reverse(h);
  printf("\n 颠倒后的链表:\n");
  output(h);
  printf("\n");
}

```

程序运行结果如图 1-7-3 所示。

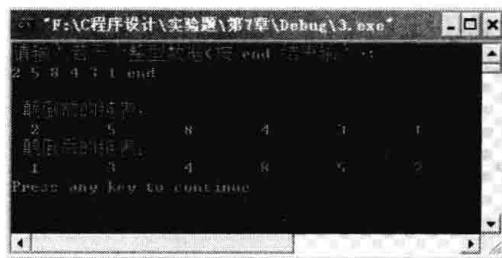


图 1-7-3 链表数据颠倒的运行结果

【思考题】

如何实现链表的排序?

4. 编写实现将两个已知的有序链表合并成一个有序链表的函数。

【解题思路】

将两个有序链表合并成一个有序链表有多种方法。以下程序设置了 4 个函数: 主函数 main()、建立有序链表函数 makeLink()、链表合并函数 copyLink()、输出链表每个表元函数 output()。

(1) 主函数 main()的设计:

① 设置3个链表头指针:第1个有序链表头指针 h1、第2个有序链表头指针 h2、合并后的新链表头指针 newh。

② 调用建立有序链表函数 makeLink(),产生两个有序链表。

③ 调用两个链表合并函数 copyLink(),合并链表并显示合并后新链表中每个表元的数值。

(2) 建立有序链表函数 makeLink()的设计:

① 设置3个结构指针变量:p指向新生成的表元,u指向新表元要插入的位置(即p指向表元插在u指向表元之前),w指向*u之前的表元。

② 循环查找插入点。循环条件是:有当前考察的表元且当前表元的值小于要插入表元的值。

③ 循环结束后,要考虑是否插在首表元之前,或插在某表元之前。

(3) 输出链表每个表元函数 output()的设计。

从链表首表元开始,循环显示每个表元的数值直到最后一个表元为止。

(4) 链表合并函数 copyLink()的设计:

① 设置新链表的头指针 newh 和尾指针 t,新链表初始时空链表,用代码 newh=t=NULL 实现。

② 在两链表还未全结束情况下循环。每次循环取两个链表中当前表元值较小的那个表元作为值要被复制的表元,并由它复制出一个新表元,将新表元添加到新链表末尾。

③ 循环结束后,完成了两链表的合并,并将新链表头指针返回给主函数。

【参考程序】

```
#include<stdio.h>
#include<malloc.h>
struct intNode
{ int value;
  struct intNode *next;
};
/*建立有序链表函数*/
struct intNode *makeLink(struct intNode *h)
{ struct intNode *p,*w,*u;
  int n;
  while(scanf("%d",&n))
  { /*还能输入整数,继续循环*/
    p=(struct intNode *)malloc(sizeof(struct intNode));
    p->value=n;
    u=h;
    /*寻找插入点*/
    while(u!=NULL&&u->value<n)
    { w=u;
      u=u->next;
    }
    if(u==h) h=p;
    else w->next=p;
    p->next=u;
  }
  return h;
}
/*输出链表中表元函数*/
void output(struct intNode *h)
{ struct intNode*p=h;
  while(p!=NULL)
```



```

    { printf("%6d",p->value);
      p=p->next;          /*准备访问下一个表元*/
    }
}
/*两链表合并函数*/
struct intNode *copyLink(struct intNode *h1,struct intNode *h2)
{ struct intNode *newh,*t,*p,*ph1=h1,*ph2=h2;
  newh=t=NULL;          /*新链表初始时空链表, newh 为头指针, t 为尾指针*/
  while(ph1||ph2)
  { p=(struct intNode *)malloc(sizeof(struct intNode));
    if(ph2==NULL||ph1!=NULL&&ph1->value<ph2->value)
    { p->value=ph1->value;
      ph1=ph1->next;
    }
    else
    { p->value=ph2->value;
      ph2=ph2->next;
    }
    p->next=NULL;
    if(t==NULL) newh=t=p;
    else t=t->next=p;
  }
  return newh;
}
/*主函数*/
void main()
{ struct intNode *h1,*h2,*newh;
  newh=h1=h2=NULL;
  printf("请输入第 1 个链表的整型数据(按 end 结束输入):\n");
  h1=makeLink(h1);      /*建立第 1 个有序链表, 头指针为 h1*/
  fflush(stdin);         /*清空输入缓冲区*/
  printf("请输入第 2 个链表的整型数据(按 end 结束输入):\n");
  h2=makeLink(h2);      /*建立第 2 个有序链表, 头指针为 h2*/
  newh=copyLink(h1, h2); /*将 h1 和 h2 两链表合并, 新链表头指针为 newh*/
  printf("\n 两链表合并后的结果: \n");
  output(newh);
  printf("\n");
}

```

程序运行结果如图 1-7-4 所示。

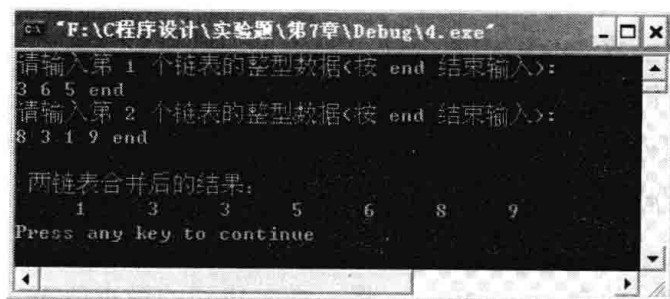


图 1-7-4 合并两个链表的运行结果

【思考题】

(1) 在上述主函数中, 在输入完第 1 个链表数据后, 为什么要加一条清空输入缓冲区函数语句 fflush(stdin);?

(2) 当实现两链表合并后, 如果出现相同的值, 如何只取一个? 例如, 上述程序运行的结果改为: 1 3 5 6 8 9。

5. 编写从无序的整数链表中找出最小表元, 并将它从链表中删除的函数。

【解题思路】

本程序设置了 5 个函数: 主函数 `main()`、建立链表函数 `makeLink()`、查找最小值表元函数 `searchMin()`、删除最小值表元函数 `deleMin()` 和输出链表每个表元函数 `output()`。

(1) 主函数 `main()` 的设计:

- ① 调用建立链表函数 `makeLink()`, 建立一个整数的链表。
- ② 调用输出链表函数 `output()`, 显示已建链表中每个表元的数值。
- ③ 调用删除最小值表元函数 `deleMin()`。
- ④ 调用输出链表函数 `output()`, 显示最小值表元被删除后链表的结果。

(2) 建立链表函数 `makeLink()` 的设计:

① 设置 3 个结构指针变量: `h` 为链表的头指针, `tail` 为链表的尾指针, `p` 指向新生成的表元; 新链表初始状态为空链表。

② 循环输入整型数据, 每输入一个整型数据产生一个表元 (由指针 `p` 指向), 并将输入的值存放到该表元中。若输入非数字字符, 如 `End`, 则输入结束。

(3) 输出链表每个表元函数 `output()` 的设计:

从链表首表元开始, 循环显示每个表元的数值直到最后一个表元为止。

(4) 查找最小值表元函数 `searchMin()` 的设计:

① 设置 4 个结构指针变量: `minp` 指向最小值的表元, 初始时指向第一个表元; `mpre` 指向最小值表元的前驱表元, 初始时为 `NULL`; `p` 指向要循环考察的表元, 从第一个表元一直考察到最后一个表元; `q` 指向要考察表元的前驱表元, 初始时为 `NULL`。

② 查找过程如下:

- 假设第 1 个表元为最小值表元, 先由 `minp` 指向。
- 由 `minp` 指向的表元与 `p` 指向的表元进行循环比较。若前者大于后者, 则后者指向的表元由 `minp` 指向, 而 `mpre` 始终指向 `minp` 指向表元的前一个表元。
- 循环结束后, 返回 `minp`, 即指向已找到的最小值表元, 并保存最小值表元的前驱表元的指针在 `*pre` 中。

(5) 删除最小值表元函数 `deleMin()` 的设计:

① 设置 2 个结构指针变量: `p` 指向要删除的表元, 即最小值的表元, 通过调用查找最小值表元函数 `searchMin()` 的返回值得到; `q` 指向要删除表元的前驱表元, 通过调用查找最小值表元函数 `searchMin()`, 作为实参地址得到。

② 删除过程如下:

- 调用查找最小值表元函数 `searchMin()`, 将 `p` 指向找到的最小值表元位置。
- 如果为空链表, 不做删除操作并返回到主函数。
- 如果删除的表元是首表元, 则修改链表头指针, 用 `*hpt=p->next` 实现; 否则, 直接删除, 用 `q->next=p->next` 实现, 并释放被删除表元的存储空间。

【参考程序】

```

#include<stdio.h>
#include<stdlib.h>
struct intNode
{ int value;
  struct intNode *next;
};
/*建立链表函数*/
struct intNode *makeLink()
{ struct intNode *h,*tail,*p;
  int n;
  h=tail=NULL;          /*h 是头指针, tail 是尾指针*/
  printf("请输入链表的整型数据(按 end 结束输入):\n");
  while(scanf("%d",&n)>0)
  {                      /*还能输入整数, 继续循环*/
    p=(struct intNode *)malloc(sizeof(struct intNode));
    p->value=n; p->next=NULL;
    if(h==NULL) h=tail=p;
    else tail=tail->next=p;
  }
  return h;
}
/*输出链表中表元函数*/
void output(struct intNode *h)
{ struct intNode *p=h;
  while(p!=NULL)
  { printf("%6d",p->value);
    p=p->next;          /*准备访问下一个表元*/
  }
}
/*在链表中最小值的表元的函数*/
struct intNode *searchMin(struct intNode *h,struct intNode **pre)
{ struct intNode *minp,*mpre,*p,*q;
  for(mpre=NULL,minp=h,q=NULL,p=h;p;p=q,p=p->next)
    if(minp->value>p->value)
    { mpre=q;minp=p;
    }
  *pre=mpre;
  return minp;
}
/*删除表元的函数*/
int deleMin(struct intNode **hpt)
{ struct intNode *p,*q;
  p=searchMin(*hpt,&q);
  if(p==NULL) return 0;
  if(q==NULL) *hpt=p->next;
  else q->next=p->next;
  free(p);
}

```

```

    return 1;
}
/*主函数*/
void main()
{ struct intNode *head;
  head=makeLink();
  printf("\n 删除前, 链表中每个表元的整型值为:\n");
  output(head);deleMin(&head);
  printf("\n 删除后, 链表中每个表元的整型值为:\n");
  output(head);printf("\n\n");
}

```

程序运行结果如图 1-7-5 所示。

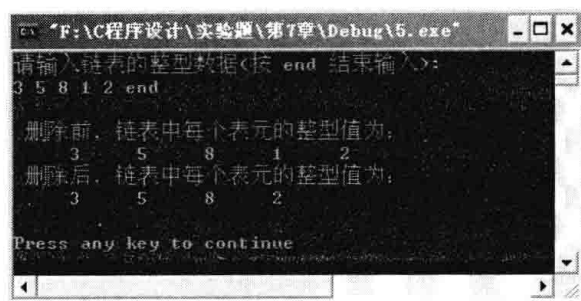


图 1-7-5 删除最小表元的运行结果

【思考题】

修改上述程序, 使之既可以删除链表中的最小表元, 也可以删除链表中的最大表元。

6. 分析并写出下列程序运行后的结果。

```

#include<stdio.h>
union          /*定义联合类型*/
{ short int i;
  char c[2];
} u;
void main()
{ u.c[0]='A';
  u.c[1]='a';
  printf(" u.i    = %d\n",u.i);
  printf(" u.c[0] = %c\n",u.c[0]);
  printf(" u.c[1] = %c\n",u.c[1]);
}

```

【解题思路】

程序中在定义一个联合类型的同时, 定义了一个联合变量 u 。它包含了两个成员: 短整型变量 i 和有两个元素的字符型数组 $c[]$ 。短整型变量 i 占两个字节, 两个元素的字符型数组 $c[]$ 也占两个字节, 它们共用相同的两字节的存储单元。当字符数组 $c[]$ 的两个元素被赋值后, 如下所示:

$u.c[1]='a'$	$u.c[0]='A'$
01100001	01000001

大写字母 A 的 ASCII 码是 41H(十进制为 65),小写字母 a 的 ASCII 码是 61H(十进制为 97)。由于短整型变量 i 与字符数组 c[]共用两个字节,i 的值十六进制数是 6141,转换成十进制为 24 897。程序运行结果如图 1-7-6 所示。

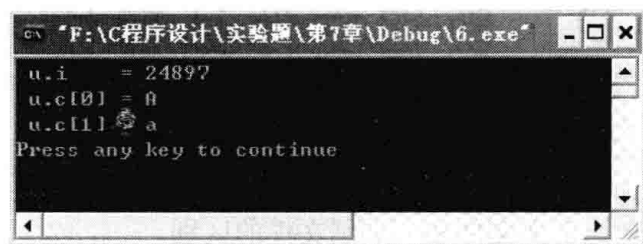


图 1-7-6 定义联合类型的运行结果

【思考题】

如果将上述定义的联合类型改为结构类型定义,即将 union 改为 struct,程序运行结果会改变吗?请说出具体理由。

三、实验小结

结构是常用的数据结构之一,要求能定义结构变量,能熟练编写应用结构实现的程序。链表是最简单的动态数据结构,学会编写链表应用程序,为以后学习数据结构打下坚实的基础。

实验 8 | 数据文件处理技术

一、实验目的

1. 了解文件的基本概念、正文文件和二进制文件的区别。
2. 了解文件的使用方法及文件生成、文件读入处理程序的基本程序结构。
3. 掌握常用的文件库函数。
4. 能熟练编写简单的文件应用程序。

二、实验内容

1. 从键盘上输入若干行字符串，存入名为 f1.txt 的磁盘文件中，直到最后一行的最后一个字符为'#'为止。要求程序最后显示共输入了多少个字符（不计换行符和结束符）及多少行。

【解题思路】

要将键盘上输入的字符依次写入到指定的文件，程序首先定义一个文件指针变量，并以写方式打开指定的文件。通过循环，反复从键盘上输入字符，将读入的字符写入到文件中。在程序中使用 getchar()函数从键盘上一个一个读入字符，并写入文件。当输入的字符为换行符时，行计数器加 1，为其他字符时，字符计数器加 1。

【参考程序】

```
#include<stdio.h>
void main()
{ FILE *fp;                                /*定义文件指针变量*/
  char ch;
  int c1=0;                                /*统计输入的字符数*/
  int c2=1;                                /*统计输入的行数*/
  if((fp=fopen("f1.txt","w"))==NULL)
  {                                         /*以写方式打开文件*/
    printf("不能打开文件 f1.txt\n");      /*打开文件错误*/
    return;
  }
  printf("请输入若干行字符串:\n");
  ch=getchar();                            /*从键盘上读入一个字符*/
  while(ch!='#')
  {                                         /*当输入的字符不是约定的结束符，循环*/
    fputc(ch,fp);                          /*读入的一个字符依次写入文件*/
    if(ch=='\n') c2++;                     /*行计数器加 1*/
    else c1++;                             /*字符计数器加 1*/
    ch=getchar();                          /*从键盘上读入一个字符*/
  }
  fclose(fp);                             /*关闭文件*/
  printf("共输入 %d 行, %d 个字符\n",c2,c1);
}
```

程序运行结果如图 1-8-1 所示。

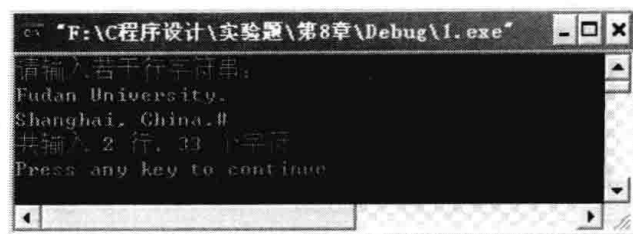


图 1-8-1 计算字符的运行结果

【思考题】

修改上述程序，当输入的字符全部写入文件后，再从文件中依次读出并显示在屏幕上。

2. 将一个已知文件中的数据依次读出，并复制到另一个文件中。

【解题思路】

这里涉及两个文件：一个是读出数据的源文件，当然必须是已存在的，文件打开的方式是“r”；另一个是写入数据的目标文件，文件打开的方式是“w”。程序首先定义两个文件指针变量，分别以读和写方式打开两个指定的文件。通过循环，使用 `fgetc()` 函数从源文件读出一个字符，将已读出的字符用 `fputc()` 函数写入到目标文件中，直至从源文件中读出文件结束符 EOF，结束循环。

【参考程序】

```
#include<stdio.h>
void main()
{ FILE *in,*out;                                /*定义文件指针变量*/
  char ch,infile[40],outfile[40];
  printf("输入源文件名: ");
  gets(infile);                                  /*或 scanf("%s%c",infile);*/
  if((in=fopen(infile,"r"))==NULL)
  { /*以读方式打开文件*/
    printf("不能打开源文件 %s\n",infile);
    scanf("%c");                                /*起暂停作用*/
    return;
  }
  printf("输入目标文件名: ");
  gets(outfile);
  if((out=fopen(outfile,"w"))==NULL)
  { /*以写方式打开文件*/
    printf("不能打开目标文件%s\n",outfile);
    fclose(in);
    scanf("%c");                                /*关闭前面打开的文件，并暂停*/
    return;
  }
  while((ch=fgetc(in))!=EOF)                    /*从源文件读出一个字符*/
  { fputc(ch,out);                               /*写入目标文件*/
    /*关闭文件*/
    fclose(in);
    fclose(out);
    printf("文件读写正常结束\n");
  }
}
```

程序运行结果如图 1-8-2 所示。

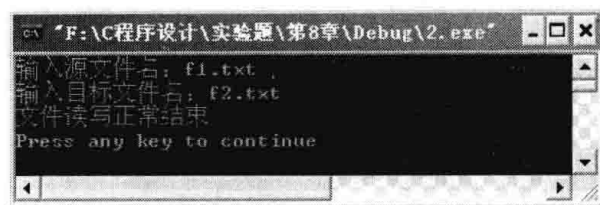


图 1-8-2 读文件中数据的运行结果

【思考题】

- (1) 要求程序最后能显示共输入了多少个字符 (不计换行符和文件结束符) 及多少行。
- (2) 如何将源文件中的所有数据追加到目标文件的末尾 (即目标文件中原有的数据不能被破坏)?
3. 从键盘输入通讯录信息并写到文件中, 然后从文件中读出并显示。通讯录中信息有: 姓名、地址、电话号码、邮政编码。要求通讯录使用结构来实现。

【解题思路】

程序应首先定义一个结构类型, 其成员包括: name (姓名)、addr (地址)、phone (电话号码)、zip (邮政编码)。然后, 定义一个文件指针变量, 并以写方式打开指定的文件。通过循环, 反复从键盘上以字符串方式输入通讯录上每个人的信息, 将读入的信息写入到文件中。当输入的姓名为空 (调用字符串长度函数 strlen() 判别是否为 0) 时, 则输入结束并跳出循环。最后, 将写入文件的数据一块块读出, 依次显示到屏幕上。

【参考程序】

```
#include<stdio.h>
#include<string.h>
char fname[40];
typedef struct                                /*使用类型定义或 struct student*/
{ char name[15];                             /*姓名*/
  char addr[20];                             /*地址*/
  char phone[15];                           /*电话号码*/
  char zip[10];                             /*邮政编码*/
} STU_TYPE;
STU_TYPE stud;                               /*或 struct student stud*/
FILE *fp;                                    /*定义文件指针变量*/
void main()
{ int count;
  printf("输入文件名: ");
  gets(fname);                               /*输入文件名*/
  if((fp=fopen(fname,"wb"))==NULL)
  { /*以写方式打开二进制文件*/
    printf("不能打开目标文件%s\n", fname);
    scanf("%*c");                            /*起暂停作用*/
    return;
  }
  for(count=1;;count++)
  { /*依次输入每个人的信息*/
```



```

printf("\n 请输入第 %d 个人的通讯录信息, 按回车键结束\n", count);
printf("姓名: ");
gets(stud.name);
if(strlen(stud.name)==0)break;          /*姓名为空, 表示输入结束*/
printf("地址: ");
gets(stud.addr);
printf("电话号码: ");
gets(stud.phone);
printf("邮政编码: ");
gets(stud.zip);
if(fwrite(&stud, sizeof(STU_TYPE), 1, fp) != 1)
{ /*stud 中数据写入文件*/
    fclose(fp);
    printf("写文件错误!\n");
    return;
}
}
fclose(fp);                             /*关闭文件*/
fp=fopen(fname, "rb");                   /*以读方式打开二进制文件*/
printf("\n 已写入文件的通讯录信息如下:\n");
printf("姓名      地址      电话号码      邮政编码\n");
while(fread(&stud, sizeof(STU_TYPE), 1, fp) == 1)
    printf("%-15s%-20s%-15s%-10s\n", stud.name,
        stud.addr, stud.phone, stud.zip); /*输出时左对齐*/
fclose(fp);                             /*关闭文件*/
}

```

程序运行结果如图 1-8-3 所示。

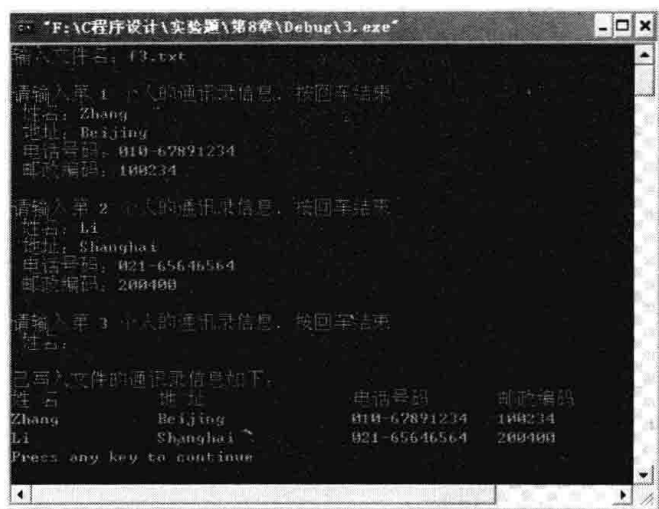


图 1-8-3 通讯录的运行结果

【思考题】

什么是正文文件? 什么是二进制文件? 这两种文件是否可以通用?

4. 编写程序, 从键盘上输入整数序列, 并按从小到大的顺序写到指定文件, 然后再从文件中依次读出并显示在屏幕上, 显示时要求每行显示 5 个整型数据。

【解题思路】

将输入的整数序列按从小到大的顺序写到指定文件, 这要进行排序处理。程序首先可以考虑设置一个整型数组, 用于存放来自键盘上输入每个整数, 然后对这个整型数组进行排序, 最后将有序数组写到指定文件就可以了。这里须要注意的是, 指定文件的打开方式必须选择既可以读也可以写的方式。

【参考程序】

```
#include<stdio.h>
#define N 1000
void main()
{ FILE *fp;                                /*定义文件指针变量*/
  int i,j,count,a[N];                      /*定义整型数组 a*/
  char fname[40];
  printf("输入文件名: ");
  gets(fname);                             /*或 scanf("%s%c",fname);*/
  if((fp=fopen(fname,"w+"))==NULL)
  { /*以读/写方式打开文件*/
    printf("不能打开文件 %s\n",fname);
    scanf("%*c");                          /*起暂停作用*/
    return;
  }
  count=0;                                /*统计输入的整型数的个数*/
  printf("请输入要写入到文件 %s 整型数据(按 end 结束):\n",fname);
  while(scanf("%d",&a[count++])!=1);      /*能正确输入一个整数循环*/
  count--;                                 /*减去最后输入一个非整型数, 即 end*/
  /*对整型数组 a 排序*/
  for(i=0;i<count;i++)
    for(j=i+1;j<count;j++)
      if(a[i]>a[j])
        {int temp=a[i];a[i]=a[j];a[j]=temp;} /*a[i]与 a[j]交换*/
  /*将整型数组 a 写到指定文件*/
  for(i=0; i<count; i++)
    fprintf(fp,"%d\t",a[i]);
  rewind(fp);                             /*文件当前读/写位置回到首部*/
  /*从文件依次读出数据, 每行显示 5 个整型数*/
  count=1;
  printf("\n 从文件 %s 读出的数据如下:\n",fname);
  while(fscanf(fp,"%d",&i)!=1)
  { /*能正确读出一个整数*/
    printf("\t%d",i);count++;
    if(count%5==0) printf("\n"); /*每行输出 5 个数据*/
  }
  printf("\n");
  fclose(fp);                             /*关闭文件*/
}
```

程序运行结果如图 1-8-4 所示。

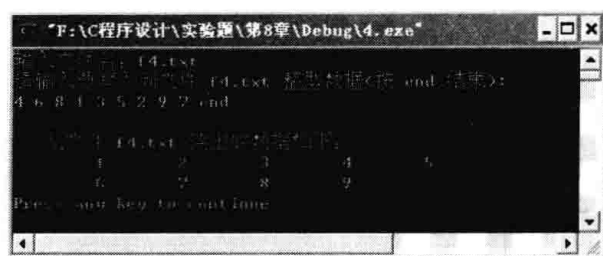


图 1-8-4 整数序列排列的运行结果

【思考题】

- (1) 修改程序，当输入的整数已出现过时无效（即重复的仅取一个），然后排序后存入到指定文件中。
- (2) 修改程序，将输入的整数序列按逆序方法存入到指定的文件中。例如，输入的整数序列是：2 4 6 1 3 5，按 5 3 1 6 4 2 的顺序存入到指定文件中。

三、实验小结

数据文件是应用程序之间实现数据交换的主要手段。通过本实验，能熟悉编写文件应用程序的基本结构和一般方法。

实验 9 | 算法设计技术基础

一、实验目的

1. 掌握迭代算法的结构，了解迭代算法须要注意的问题。
2. 掌握递推算法的基本原理，能用递推算法求解具有递推特性的问题。
3. 掌握回溯算法的基本原理，对于扩展、调整回溯相对简单明确的问题能用回溯法求解。
4. 了解贪婪算法的基本原理，在贪婪规则已知的情况下，能用贪婪算法求解问题。
5. 了解动态规划法的基本原理。

二、实验内容

1. 求方程 $x^3 - x^2 - 1 = 0$ 在 1.5 附近的一个根。

假定将方程写成以下等价形式：

$$x = 1 + 1/x^2$$

【参考程序】

```
#include<stdio.h>
#include<math.h>
typedef double(*fpt)(double);
int rootfx(fpt g, double initRoot, double Epsilon, double *root, int maxn)
{ double x1, x0=initRoot; int n=0;
  do
  { printf("x%d=%.7f\n", n, x0);
    if(n==maxn)
    { /*测试迭代次数是否超过限制*/
      printf("已迭代了%d次，还未找到解！程序终止迭代!\n", n);
      break;
    }
    x1=x0;
    x0=(*g)(x1);          /*按指定的公式计算*/
    n++;                  /*迭代次数增1*/
  } while(fabs(x0-x1)>=Epsilon);
  *root=x0;
  printf("x%d=%.7f\n", n, x0);
  return n;
}

double f(double x)
{ return 1.0+1.0/x/x;
}
```

```

void main()
{ double x;
  int m=rootfx(f,1.5,0.000005,&x,30);
  if(m==30) printf("迭代%d次, 还未找到根\n",m);
  else printf("迭代%d次, 根为%.7f\n",m,x);
}

```

2. 产生 2^m 个由 0 和 1 数字组成的数字串。将串看做是首尾相连的环, 则所有由连续 m 个数字组成的序列共有 2^m 个, 将 m 个 0 和 1 数字序列看做是一个 m 位的二进制数, 要求这个数字串满足以下要求: 所有 m 位二进制数都互不相同。例如, $m=3$, 在数字串 11101000 首尾相连构成的环中, 由 3 个数字组成的序列共有 8 个, 每个可能的序列都在环中出现一次, 它们依次是 000、001、010、101、011、111、110、100 (见图 1-9-1), 它们的值互不相同。

【参考程序】

程序采用试探法, 为了便于处理环上的 2^m 个连续 m 个 0 和 1 数字序列, 将数字序列延伸至 2^m+m 个数字。

```

#include<stdio.h>
#define M 10
#define N 1024
int b[N+M-1];
int equal(int k,int j,int m)/*比较分别自 k 和 j 位开始的连续 m 位 0、1 串是否相同*/
{ int n=0x01<<m;
  for(int i=0;i<m;i++)
    if(b[(k+i)%n]!=b[(j+i)%n])
      return 0;
  return 1;
}
/*调整回溯, 当自 k 位开始的第 m 位为 1 时, k 回溯; 为 0 时, 置 1。返回回溯后的开始位*/
int exchange(int k,int m)
{ int n=0x01<<m;
  while(b[(k+m-1)%n]==1){ b[(k+m-1)%n]=0;k--;}
  b[(k+m-1)%n]=1;
  return k;
}
void main()
{ int m,k,n,j;
  printf("Enter m(1<m<10)\n"); scanf("%d",&m);
  n=0x01<<m; /*n=2^m*/
  for(k=0;k<n;k++) b[k]=0;
  for(k=1,j=0;k<n;)
  { /*从第 2 个子序列开始, 检查它与前面的所有子序列是否相等*/
    if(equal(k,j,m))
    { /*第 k 个子序列与前面某个 j 子序列相等*/
      k=exchange(k,m); /*回溯或调整*/
      if(k+m>=n||k==0) /*回溯引起前面检查过的位也调整, 或回溯至第 1 个*/
        k=1; /*从第 2 个序列开始重新检查*/
    }
    j++;
  }
}

```

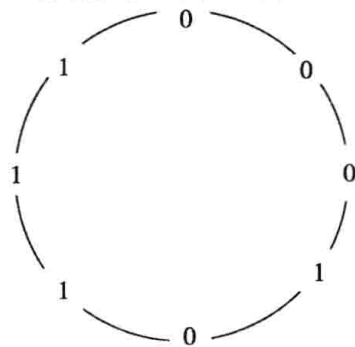


图 1-9-1 题 2 图

【思考题】将检查、扩展、调整分别编写成函数，采用传统的试探法找所有的解。

例如，设数组 $d[]$ 中存储以下整数：

则有

$$71 = 49 + 9 + 9 + 4$$

采用递归方法找解，设递归函数设有以下 5 个参数：被分解的数为 n ，最多还可有 m 个和数，和数取自数组 $d[]$ ， $d[]$ 中还有 c 个供选的和数，分解出来的下一个和数表指针 a 所指的元素中。函数返回求得和数个数，如果不能分解，函数返回 0。算法须考虑以下多种情况：

- (1) n 为 0, 不要继续分解。
- (2) m 为 0, 或 c 为 0, 不能再有别的和数, 或不再有新的和数, 不能再分解。
- (3) 当前和数比被分解的数 n 更大, 用后一个和数再分解。
- (4) 当前和数不比被分解的数 n 更大, 试验用当前和数分解。

【参考程序】

```
#include<stdio.h>
int parse(int n,int m,int *d,int c,int *a)
{ int r;
  if(n==0) return 0; /*被分解的数为 0，不要再分解*/
  if(m==0||c==0) /*不能再有和数，或没有更多的和数，不能分解*/
    return 0;
  if(n< *d) /*当前和数比被分解的数大，试用下一个和数*/
    return parse(n,m,d+1,c-1,a); /*试用下一个和数分解*/
  else
  { /*试用当前和数分解*/
```

```

    *a=*d;
    if(n-*d==0)
        return 1; /*分解完成*/
    if((r=parse(n-*d,m-1,d,c,a+1))>0) /*对剩余的数分解, 并且分解成功*/
        return r+1;
    return parse(n,m,d+1,c-1,a); /*用下一个和数对 n 进行分解*/
}
}
#define N 20
void main()
{ int n,m,p[20],*a,k;
  int d[]={100,81,64,49,36,25,16,9,4,1};
  printf("Enter n&m\n");
  scanf("%d%d",&n,&m);
  a=p; k=parse(n,m,d,10,a);
  if(k==0) printf("不能分解!\n");
  else
  { printf("分解如下:\n%d=%d",n,*a++);
    while(--k)printf("+%d",*a++);
  }
  printf("\n");
}

```

4. 找出从自然数 1、2、 $\dots$ 、 n 中任取 r 个数的所有组合。例如, $n=5$, $r=3$, 所有组合为:

```

1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5

```

【算法 1】

分析例示所列的 10 个组合, 可用这样的递归思想来考虑求组合函数的算法: 设函数为 `void comb(int m, int k)`, 即找出从自然数 1、2、 $\dots$ 、 m 中任取 k 个数的所有组合。当组合的第 1 个数字选定时, 其后的数字是从余下的数字中取 $k-1$ 个数的组合, 这就将求 m 个数中取 k 个数的组合问题转化成在较小数字范围内取 $k-1$ 个数的组合问题。

设函数引入工作数组 `a[]` 存放求出的组合的数字, 约定函数将确定的 k 个数字组合的第 1 个数字放在 `a[k-1]` 中, 当一个组合的全部元素都求出后, 才将 `a[]` 中的一个组合输出。第 1 个数可以是 m 、 $m-1$ 、 $\dots$ 、 k , 函数将确定的组合的第 1 个数字放入数组后, 有两种可能的选择, 因还未确定组合的其余元素, 继续递归去确定组合的其余元素; 或因已确定了组合的全部元素, 输出这个组合。

【参考程序】

```

#include<stdio.h>
#define MAXN 100

```

```

int r=3;
int a[MAXN];
void comb(int m,int k)
{ int i,j;
  for(i=m;i>=k;i--)
  { a[k-1]=i;
    if(k>1) /*组合中还有其他元素*/
      comb(i-1,k-1);
    else
    { /*找到了一个组合, 将组合的元素输出*/
      for(j=r-1;j>=0;j--)printf("%4d",a[j]);
      printf("\n");
    }
  }
}
void main()
{ comb(5,3); }

```

【算法 2】

采用回溯法找问题的解, 将找到的组合以从小到大顺序存于 $a[0]$ 、 $a[1]$ 、 $\dots$ 、 $a[r-1]$ 中, 组合的元素满足以下性质:

- ① $a[i+1] > a[i]$, 后一个数字比前一个大。
- ② $a[i] - i \leq n - r + 1$ 。

找解过程可以叙述如下: 首先放弃组合数个数为 r 的条件, 候选组合从只有一个数字 1 开始。因该候选解满足除问题规模之外的全部条件, 扩大其规模, 并使其满足上述条件①, 候选组合改为 1、2。继续这一过程, 得到候选组合 1、2、3。该候选解满足包括问题规模在内的全部要求, 因而是个解。在该解基础上, 选下一个候选解, 因 $a[2]$ 上的 3 调整为 4 以及继续调整为 5, 得到解 1、2、4 和 1、2、5。由于对 5 不能再作调整, 就要从 $a[2]$ 回溯至 $a[1]$, 这时的 $a[1]=2$, 可以调整为 3, 并向前试探, 得到解 1、3、4。重复上述向前试探和向后回溯, 直至要从 $a[0]$ 再回溯时, 说明已找完问题的全部解。

【参考程序】

```

#define MAXN 100
int a[MAXN];
void comb_back(int m,int r)
{ int i,j;
  i=0;
  a[i]=1;
  do
  { if(a[i]-i<=m-r+1)
    { /*还可以向前试探*/
      if(i==r-1)
      { /*已找到一个组合*/
        for(j=0;j<r;j++) printf("%4d",a[j]);
        printf("\n");
        a[i]++;
        continue;
      }
    }
  }
}

```



```

    }
    i++;
    a[i]=a[i-1]+1;
}
else
{ /*回溯*/
    if(i==0) return;
    a[--i]++;
}
} while(1);
}
void main()
{ comb_back(5,3);
}

```

5. 从有 n 个元素的整数数组中删除 i 个元素，使剩下的 $n-i$ 个元素构成一个不减的序列。试设计算法和程序找出数组最长的不减序列。

【算法】

设 n 个整数存于数组 $b[0\dots n-1]$ 中，当前已考察了 $b[0\dots i-1]$ ，并求得最长不减子序列的长为 k ，则继续考察 $b[i]$ 是否会引起 k 值的增加，依赖于 $b[i]$ 是否大于或等于 $b[0\dots i-1]$ 中某个最长（长为 k ）不减子序列的终元素。当然，可能有多个长为 k 的不减子序列。很显然，程序只要保留其中序列终元素最小的那一个不减子序列就足够了。为此，要有变量记录长为 k 的不减子序列的终元素。这样，有长为 $k+1$ 的不减子序列，当且仅当 $b[i]$ 不小于这个终元素。但当 $b[i]$ 小于这个终元素时，如果 $b[0\dots i-1]$ 含有一个长为 $k-1$ 的不减子序列，且它的终元素值不大于 $b[i]$ ，这时，因 $b[i]$ 小于这个终元素，就得到一个终元素值更小的长为 k 的不减子序列。为了实现这样的原判断，就得保留长为 $k-1$ 的不减子序列的终元素。依此类推，为了考虑长为 $k-2$ 、 $k-3$ 、 $\dots$ 、 1 的不减子序列，相应地，都得保留它们的最小终元素的值。为此，要另外引入数组 $a[]$ ， $a[j]$ 是 $b[0\dots i-1]$ 中长为 j 的不减子序列的最小终元素的值。

显然， $a[]$ 有性质：

$a[1]<a[2]<\dots<a[k]$

利用这个性质，就能知道考察 $b[i]$ 时， $a[]$ 中哪些元素须要改变。

当 $a[1]\leq b[i]<a[k]$ 时，则存在某个 j ，使

$a[j-1]\leq b[i]<a[j]$

显然， $a[1\dots j-1]$ 的元素不应改变。由于 $a[j-1]$ 终止 $b[0\dots i-1]$ 的一个长为 $j-1$ 的不减子序列，所以， $b[i]$ 必定终止 $b[0\dots i]$ 中的一个长为 j 的不减子序列。这样， $a[j]$ 应改变为 $b[i]$ 。最后， $a[j+1\dots k]$ 中的元素也不应改变。得到以下算法：

```

{ a[1]=b[0]; k=1;
  for(i=1;i<n;i++)
    if(b[i]>=a[k]) a[++k]=b[i];
    else if(b[i]<a[1]) a[1]=b[i];
    else
    { /*寻找满足条件 a[j-1]<=b[i]<a[j] 的 j;*/
      a[j]=b[i];
    }
}

```

```

    }
    return k;
}

```

【参考程序】

```

#include<iostream.h>
#define N 100
int a[N],b[N];
int maxAdLen(int b[],int a[],int n)
{ int k,i,j;
  a[1]=b[0]; k=1;
  for(i=1;i<n;i++)
    if(b[i]>=a[k])a[++k]=b[i];
    else if(b[i]<a[1])a[1]=b[i];
    else
      { for(j=1;j<=k;j++) /*寻找满足条件 a[j-1]<=b[i]<a[j]的 j;*/
        if(b[i]<a[j])
          break;
        a[j]=b[i];
      }
  return k;
}

void main()
{ int n,k,len;
  cout<<"Enter n\n";
  cin>>n;
  cout<<"输入"<<n<<"个整数\n";
  for(k=0;k<n;k++)
    cin>>b[k];
  len=maxAdLen(b,a,n);
  cout<<"递增序列的长度是"<<len<<endl;
}

```

【思考题】

如果要求程序同时求出递增序列,如何修改程序?

6. 有一个工程承包人需要确定每月的工人人数,工程承包人知道工程需要工作的月数,每个月需要的最少工人人数以及每一个工人的雇用手续费、解雇手续费和每月工资。试为工程承包人确定工程的最低费用和每月实际雇用的工人人数。

【算法】

假设整个工程要持续 m 个月,每个月需要的最少工人人数存于数组 $n[]$,雇用手续费为 $hires$,解雇手续费为 $fires$,工人每月工资额为 $salary$ 。采用试探法寻找最低费用,并用递归实现。另引入以下工作变量:

$minV$: 已找到的最优解的最低费用。

$minn[]$: 已找到的最优解的每月工人人数。

$curn[]$: 当前正在形成的解的每月工人人数。

以月份的递增顺序对费用找最优解。对于 i 月须要考虑以下多种情况。

$i=0$: 承包人按第 1 月需要的最少工人数雇用工人 ($curn[0]=n[0]$), 须支付全部雇用工人的雇用手续费和第 1 个月的工资。

$i=m$: 已找到了 1 个可能解, 若 v 与解雇全部工人所需费用比 $minV$ 还要少, 则找到一个更优解, 用当前形成的解替代早先找到的最优解。

$n[i]=curn[i-1]$: 当前月需要工人数与上月雇用的工人数相同, 则这些工人应继续雇用, 即继续支付这些工人的工资。

$n[i]>curn[i-1]$: 当前月需要工人数比上月雇用的工人数多, 则要求按需要工人数再雇用 ($n[i]-curn[i-1]$) 个工人, 承包人须支付这些工人的手续费和 $n[i]$ 个工人的工资。

$n[i]<curn[i-1]$: 在不考虑 $i+1$ 月需要的工人数情况下, 以解雇 0 个工人、解雇 1 个工人、直至解雇 ($curn[i-1]-n[i]$) 个工人的每一种情况作试探递归找解。承包人须支付解雇工人的手续费和留下的工人的工资。

search(考虑 i 号月, 当前方案累计费用 v)

```
{ if(要考虑的月已全部考虑完) 正要考虑的月  $i$  是最后月  $m$ 
  if(累计费月加上解聘全部工人的费用小于早先最佳方案的费用)
  { /*保存当前解为目前最优解*/
    minV=v;
    for(j=0; j<m; j++)
      minn[j]=curn[j];
    return;
  }
  else return;
  if(i==0)
  { curn[0]=n[0];
    search(i+1, (hires+salary)*curn[0]);
  }
  else if(n[i]==curn[i-1])
  { /*当前月需要工人数与上月雇用的工人数相同*/
    curn[i]=n[i];
    search(i+1, v+salary*curn[i]);
  }
  else if(n[i]>curn[i-1])
  { /*当前月需要工人数比上月雇用的工人数多*/
    curn[i]=n[i];
    search(i+1, v+hires*(curn[i]-curn[i-1])+salary*curn[i]);
  }
  else
  for(j=curn[i-1]; j>=n[i]; j--)
  { curn[i]=j;
    search(i+1, v+fires*(curn[i-1]-curn[i])+salary*curn[i]);
  }
}
```

【参考程序】

```

#include<stdio.h>
#define M 20
int minV,m,n[M],minn[M],curn[M],hires,salary,fires;
void search(int i,int v)
{ int j;
  if(i==m) /*如果是最后一个月*/
    if((v+=fires*curn[i-1])<minV)
    { /*找到了一个费用更少的方案*/
      minV=v;
      for(j=0;j<m;j++) minn[j]=curn[j];
      return;
    }
  else return; /*方案没有更好*/
  if(i==0)
  { /*如果是第 1 个月*/
    curn[0]=n[0]; search(i+1,(hires+salary)*curn[0]);
  }
  else if(n[i]==curn[i-1])
  { curn[i]=n[i]; search(i+1,v+salary*curn[i]);
  }
  else if(n[i]>curn[i-1])
  { curn[i]=n[i]; search(i+1,v+hires*(curn[i]-curn[i-1])+salary*curn[i]);
  }
  else for(j=curn[i-1];j>=n[i];j--){
    curn[i]=j; search(i+1,v+fires*(curn[i-1]-curn[i])+salary*curn[i]);
  }
}

void main()
{ int j;
  printf("Enter m,hires,salary,fires,n[]\n");
  scanf("%d%d%d%d",&m,&hires,&salary,&fires);
  minV=0;
  for(j=0;j<m;j++)
  { scanf("%d",&n[j]);
    minV+=n[j]*(hires+salary+fires);
  }
  search(0,0);
  printf("MIN Value=%d\n",minV);
  for(j=0;j<m;j++) printf("%4d",minn[j]);
  printf("\n");
}

```

7. 一场激烈的足球比赛开始前,售票工作正在紧张进行,每张球票 50 元。现有 $2n$ 人排队等待购票,其中有 n 个人手持 50 元,另外 n 个人手持 100 元。假设开始售票时售票处没有零钱,问这 $2n$ 个人有多少种排队方式,使售票处不至于出现找不开钱的局面。

【算法 1】

采用深度优先搜索算法。令变量 k 记录售票处有 50 元钞票的张数。初始时, k 为 0, 若当前购票人手持 100 元, 且 $k=0$, 则回溯; 否则, 继续考虑下一个人。若第 $2n$ 人能正常购票, 则排队种数计数器加 1。待递归结束, 计数器即为排队总数。

【参考程序】

```
#include<stdio.h>
int dps(int i,int k,int m,int n)
/*第 i 人, 当前有 k 张 50 元, 前面有 m 人手持 50 元购票*/
{ int t=0;
  /*若第 i 人手持 50 元*/
  if(m+1==n) t+=1; /*已共有 n 人手持 50 元购票, 找到一个解*/
  else t+=dps(i+1,k+1,m+1,n);
  /*若第 i 人手持 100 元, 将要找回一张 50 元, 仅当还有 50 元才是可行的*/
  if(k>0) t+=dps(i+1,k-1,m,n);
  return t;
}
void main()
{ int n;
  printf("Enter n!\n");scanf("%d",&n);
  printf("Total=%d\n",dps(1,0,0,n));
}
```

若要求输出每种排队购票的情况, 程序须保存找到的解, 并输出。引入数组 $p[]$ 存储解, 程序如下:

【参考程序】

```
#include<stdio.h>
#define N 100
int p[N];
int dps(int i,int k,int m,int n)
/*第 i 人, 当前有 k 张 50 元, 前面有 m 人手持 50 元购票*/
{ int j,t=0;
  /*若第 i 人手持 50 元*/
  p[i]=50;
  if(m+1==n)
  { t+=1;
    for(j=i+1;j<=2*n;j++) p[j]=100;
    for(j=1;j<=2*n;j++) printf("%4d",p[j]);
    printf("\n");
  }
  else t+=dps(i+1,k+1,m+1,n);
  p[i]=100;
  /*若第 i 人手持 100 元*/
  if(k>0) t+=dps(i+1,k-1,m,n);
  return t;
}
void main()
```

```

{ int n;
  printf("Enter n!\n"); scanf("%d", &n);
  printf("Total=%d\n", dps(1, 0, 0, n));
}

```

【算法2】

采用动态规划法, 令 $f(m, n)$ 表示有 m 个人持 50 元的钞票, n 个人持 100 元的钞票时, 方案总数分以下多种情况:

(1) $n=0$: 排队购票者都持 50 元钞票, 则 m 个人排队总数为 1, 即 $f(m, 0)=1$ 。

(2) $m < n$: 因有人不能得到零钱, 排队总数为 0, 即 $f(m, n)=0$ 。

(3) 其他情况: 有 $m+n$ 人排队购票, 第 $(m+n)$ 个人站在第 $(m+n-1)$ 个人的后面, 则第 $(m+n)$ 个人的排队方式可由下列两种情况获得:

a. 第 $(m+n)$ 个人持 100 元的钞票, 则在他之前的 $(m+n-1)$ 个人中有 m 个人持 50 元钞票, 有 $(n-1)$ 个人持 100 元钞票, 此种情况共有 $f(m, n-1)$ 种。

b. 第 $(m+n)$ 个人持 50 元的钞票, 则在他之前的 $(m+n-1)$ 个人中有 $(m-1)$ 个人持 50 元钞票, 有 n 个人持 100 元钞票, 此种情况共有 $f(m-1, n)$ 种。

$m+n$ 个人的排队方案应是上述两种情况之和, 即 $f(m, n)=f(m-1, n)+f(m, n-1)$ 。

于是得到 $f(m, n)$ 的计算公式:

$$f(m, n) = \begin{cases} 0 & (m < n) \\ 1 & (n = 0) \\ f(m, n-1) + f(m-1, n) & (\text{其他}) \end{cases}$$

【参考程序】

```

#include<stdio.h>
#define M 40
int f[M+1][M+1];
int arithmetic(int m)
{ int i, j;
  for(i=1; i<=m; i++) f[i][0]=1;
  for(j=0; j<m; j++) f[j][j+1]=0;
  for(i=1; i<=m; i++)
    for(j=1; j<=i; j++)
      f[i][j]=f[i-1][j]+f[i][j-1];
  return f[m][m];
}
void main()
{ int n;
  printf("Enter n!\n"); scanf("%d", &n);
  printf("Total=%d\n", arithmetic(n));
}

```

8. 一场激烈的足球比赛之后, 场地中被一群足球流氓丢了许多杂物。假定南北向的足球场如一个二维数组, 被分成若干行和列, 每个行列位置有多少不等的杂物。现在准备用一个清除杂物的机器人清扫足球场。机器人从足球场的 0 行 0 列位置出发, 只能按列或者按行增加的方向前进, 不能后退。机器人经过位置的杂物将被机器人清除。试编写程序, 在杂物分布给定的情况下, 为机器人确定一个行进路线, 使机器人清除的杂物最多。

【算法】

采用动态规划法, 设足球场被分割成 n 行 m 列, 第 i 行 j 列位置有杂物 $a[i][j]$ 。令 $f(m,n)$ 表示在 i 行 j 列位置上机器人能清除的最多杂物数量, 有以下多种情况:

(1) $f(0, j) = a[0][0] + a[0][1] + \cdots + a[0][j]$ 。

(2) $f(i, 0) = a[0][0] + a[1][0] + \cdots + a[i][0]$ 。

(3) $f(i, j) = \max(f(i-1, j), f(i, j-1)) + a[i][j]$ 。

【参考程序】

```
#include<stdio.h>
#define M 40
int f[M][M], a[M][M];
int scavenge(int n, int m)
{ int i, j;
  for(f[0][0]=a[0][0], i=1; i<n; i++) f[i][0]=f[i-1][0]+a[i][0];
  for(j=1; j<m; j++) f[0][j]=f[0][j-1]+a[0][j];
  for(i=1; i<n; i++)
    for(j=1; j<m; j++)
      f[i][j]=(f[i-1][j]>f[i][j-1]?f[i-1][j]:f[i][j-1])+a[i][j];
  return f[n-1][m-1];
}

void main()
{ int n, m, i, j;
  printf("Enter n!\n");
  scanf("%d", &n);
  printf("Enter m!\n");
  scanf("%d", &m);
  printf("Enter data!\n");
  for(i=0; i<n; i++)
    for(j=0; j<m; j++)
      scanf("%d", &a[i][j]);
  printf("MAX=%d\n", scavenge(n, m));
}
```

【思考题】

修改上述程序, 求出机器人的最佳行走路线。

三、实验小结

为了求解问题, 必须先设计一个算法。算法的设计须要根据问题的性质, 确定求解算法的类型, 一般的问题就可由算法的类型得到算法的基本结构, 然后寻求一个可行的算法, 最后按算法编写程序。掌握了本章的内容, 并能独立完成相应实验, 说明具有了一定的程序开发能力。

实验10 | 综合实验

一、实验目的

通过综合实验，综合应用所学程序设计的知识，独立完成相对比较完整的课题，进一步提高程序设计能力。

二、实验内容

1. 钓鱼。利用 52 张扑克牌进行“钓鱼”游戏。

游戏规则如下：

(1) 游戏开始时，先随机发牌，将牌分成相同数目的两堆。

(2) 双方轮流从牌堆顶端出牌，所出的牌互相交叉，连成长龙。

(3) 若某一方出的牌与长龙中前面的某张牌的面值相同，则这两张牌连同当中的牌全部归这一方所有，并按原来的顺序放到这一家牌堆的底部；然后，再从牌堆顶部出一张牌放到长龙尾部，并检查前述情况是否发生。如果再次发生前述情况，则继续作前述那样的处理。

(4) 游戏以某一方无牌结束。

为了便于取走连续的一叠牌和一叠牌的移动，游戏两方的牌和长龙分别用链表存储。一张牌的结构以及一叠牌的结构见梭哈问题。

【参考程序】

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include<conio.h>
/*有人胜利的条件*/
#define win(player[0].num==0||player[1].num==0)
typedef struct card                                /*一张牌*/
{
    int suit;                                       /*花色*/
    int face;                                       /*面值*/
} CARD;
typedef struct deck                                /*一叠牌链表中的一张牌表元*/
{
    CARD Card;                                     /*一张牌*/
    struct deck *next;                             /*下一张牌*/
} DECK;
typedef struct head                                /*一叠牌*/
{
    int num;                                       /*牌的张数*/
    DECK *first;                                   /*第一张牌*/
    DECK *last;                                   /*最后一张牌*/
} HEAD;
char *suit[4]={"梅花","方块","红桃","黑桃"};    /*牌的四种花色*/
HEAD *newdeck()
{
    /*按顺序生成一副牌*/
```



```

int i, j;
HEAD *Head=(HEAD *)malloc(sizeof(HEAD));
DECK *p, *q;
Head->num=52;
for(i=0; i<4; i++)
    for(j=1; j<14; j++)
    { q=(DECK *)malloc(sizeof(DECK));
      q->Card.face=j; q->Card.suit=i;
      if(i==0&&j==1)
      { Head->first=q; p=q;
      }
      else
      { p->next=q; p=q;
      }
    }
p->next=NULL;
Head->last=p;
return Head;
}

void freedeck(HEAD Head)
{ /*释放一叠牌空间*/
  DECK *p, *q;
  for(p=Head.first; p; p=q)
  { q=p->next; free(p);
  }
}

CARD pick(HEAD *Head)
{ /*随机抽取一张牌*/
  int i, j=rand()%Head->num;
  CARD temp={0,0};
  if(Head->num==0) return temp;
  DECK *p=Head->first, *q;
  if(j==0) Head->first=p->next;
  else
  { for(i=0; i<j-1; i++) p=p->next;
    q=p; p=p->next; q->next=p->next;
    if(p->next==NULL) Head->last=q;
  }
  Head->num--;
  return p->Card;
}

void addcard(HEAD *Head, CARD Card)
{ /*在牌叠的末尾添加一张牌*/
  DECK *p=(DECK *)malloc(sizeof(DECK));
  p->Card=Card; /*复制一张牌*/
  p->next=NULL;
  Head->num++;
  if(Head->last) Head->last=Head->last->next=p; /*添加在牌叠的末尾*/
  else Head->last=Head->first=p; /*原牌堆空, 增加的牌在牌堆之首*/
}

void push(HEAD *player, HEAD *desk)
{ /*玩家打出一张牌, 放到台面*/
  CARD Card=player->first->Card;
  addcard(desk, Card);
  player->num--;
  player->first=player->first->next;
}

```

```

    if(player->first==NULL)player->last=NULL;
}
bool pull(HEAD *player,HEAD *desk)
{ /*判断是否可以收取牌, 可以则收取*/
    DECK *p=desk->first, *q=desk->last,*w;
    if(p==q)return false; /*没牌, 或只有一张牌, 立即返回*/
    if(p->Card.face==q->Card.face)
    { while(p)
        { /*一堆牌全部收取*/
            addcard(player,p->Card);
            q=p; p=p->next; free(q);
        }
        desk->num=0;
        desk->first=NULL;
        desk->last=NULL;
        return true;
    }
    while(p->next!=q)
    { /*判断是否有面值相同的牌*/
        if(p->next->Card.face==q->Card.face)
        { /*如果找到, 则收取*/
            w=p;p=p->next;
            while(p)
            { addcard(player,p->Card);
                q=p; p=p->next;free(q);
                desk->num--;
            }
            w->next=NULL; desk->last=w;
            return true;
        }
        else p=p->next;
    }
    return false;
}
void printcard(CARD Card)
{ /*输出一张牌*/
    if(Card.face<10)
        printf("%5s %-2d ",suit[Card.suit],Card.face+1);
    else switch(Card.face)
    { case 10:printf("%5s J ",suit[Card.suit]);break;
      case 11:printf("%5s Q ",suit[Card.suit]);break;
      case 12:printf("%5s K ",suit[Card.suit]);break;
      case 13:printf("%5s A ",suit[Card.suit]);break;
    }
}
void print(HEAD Head)
{ /*输出一叠牌*/
    int i;
    DECK *p=Head.first;
    for(i=0;i<Head.num;i++)
    { if(i&&5==0) putchar('\n');
        printcard(p->Card);
        p=p->next;
    }
    putchar('\n');
}

```

```

void printAll(char*s, HEAD h0, HEAD h1, HEAD h2)
{
    system("cls");
    printf("%s",s);
    printf("台面的牌:\n");
    print(h0);    printf("玩家 1 有牌%d 张:\n",h1.num);
    print(h1);    printf("玩家 2 有牌%d 张\n", h2.num);
    print(h2);
    printf("\n\n");
}

void main(void)
{
    srand((unsigned)time(NULL));                /*初始化随机数发生器的种子*/
    int i,oneOrTwo;                             /*玩家标志*/
    char c;
    bool skip,restart;                          /*忽略和重新开始标志*/
    HEAD Head,player[2],desk;
    Head.first=NULL;
    do
    {
        if(!Head.first)freedeck(Head);
        system("cls");
        printf("游戏开始,按任意键发牌...");
        getch();
        Head=*newdeck();                       /*生成一副新牌*/
        skip=false;                            /*清除忽略标志*/
        restart=false;                         /*清除重新开始标志*/
        for(oneOrTwo=0; oneOrTwo<2; oneOrTwo++)
        {
            player[oneOrTwo].num=0;            /*玩家牌堆清空*/
            player[oneOrTwo].first=NULL;
            player[oneOrTwo].last=NULL;
            for(i=0;i<26;i++)                  /*为玩家随机抽取 26 张牌*/
                addcard(&player[oneOrTwo],pick(&Head));
        }
        desk.num=0;                            /*桌面牌堆清空*/
        desk.first=NULL;
        desk.last=NULL;
        oneOrTwo=2;                            /*用玩家 2 作初值*/
        while(!win)
        {
            /*没有一方获胜,循环*/
            /*玩家轮换*/
            oneOrTwo=3-oneOrTwo ;
            do
            {
                printAll("玩家出牌前\n", desk, player[0], player[1]); /*出牌前*/
                if(!skip)
                    printf("玩家%d 出牌,按任意键继续...\n(
                        按 i 键进行到结束, r 键重新开始, q 键退出)", oneOrTwo);
                if((!skip)|| (kbhit()))c=getch(); /*当忽略状态中按键时*/
                switch(c)
                {
                    case 'q':putchar('\n');    return; /*退出*/
                    case 'r':restart=true;     /*重新开始*/
                        player[oneOrTwo-1].num=0;
                        break;
                    case 'i':skip=true;        /*忽略*/
                        break;
                    default:skip=false;        /*其他键退出忽略*/
                }
            }
            push(&player[oneOrTwo-1],&desk); /*玩家出牌前*/
            if(win) break;                  /*有人获胜,终止循环*/
            printAll("玩家出牌后\n", desk, player[0], player[1]);
        }
    }
}

```

```

    if(!skip)
        printf("按任意键继续...\n(按 i 键进行到结束, r 键重新开始, q 键退出)");
    else
        printf("按任意键暂停...\n(按 i 键进行到结束, r 键重新开始, q 键退出)");
    if((!skip)|| (kbhit())) c=getch();
    switch(c)
    { case 'q': putchar('\n'); return;
      case 'r': restart=true;
        player[oneOrTwo-1].num=0;
        break;
      case 'i': skip=true;
        break;
      default: skip=false;
    }
    }while(pull(&player[oneOrTwo-1], &desk)); /*当有收取牌时, 继续由玩家出牌*/
    if(win) break;
}
if(restart) continue; /*如果按过 r 键, 重新开始*/
system("cls");
printf("玩家 1:%d 张 玩家 2:%d 张\n\n", player[0].num, player[1].num);
print(desk);
printf("\n\n");
if(player[0].num==0) printf("玩家 2 获胜");
else printf("玩家 1 获胜");
while(kbhit()) getch(); /*先清空输入缓冲区*/
getch(); /*暂停, 等待用户输入*/
system("cls");
printf("按 q 键退出, 按其他任意键重新开始...");
if(getch()!='q') restart=true;
putchar('\n');
}while(restart);
}

```

2. 小霸王学习机为低年级小学生编写一个“学习机”模拟程序, 程序利用随机函数产生两个整数, 并给出算式请小学生输入答案。为了提高程序的实用性, 运算范围和模拟方式先由用户指定。

要求至少具有加法、减法、乘法、除法 4 种运算, 并提供以下两种模拟方式:

- (1) 练习模式, 不计分, 连续出题, 直至用户回答结束。
- (2) 测验模式, 计分, 连续出题数由用户指定, 学习机给出评分。

也可以有其他好玩的设置, 全由设计者自己设定。

用链表记录答题过程, 一道题的解答记录在以下结构中:

```

typedef struct score /*存储计分信息的结构*/
{ int op1; /*第 1 个整数*/
  int op2; /*第 2 个整数*/
  char opp; /*运算符*/
  int right; /*正确答案*/
  int answer; /*用户解答*/
  struct score *next; /*指向下一结构*/
  struct score *pre; /*指向上一结构*/
} SCORE;

```

采用以上结构, 以便能浏览一次解题过程。

【参考程序】

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
#include<math.h>
typedef struct score
{ int op1;
  int op2;
  char opp;
  int right;
  int answer;
  struct score *next;
  struct score *pre;
}SCORE;
void practice(void);
void test(void);
void aSample(SCORE *,int);
void waitTime(long n)
{ for(long now=0;now<n;now++);
}
void main()
{ long now;
  printf("\n\n\n\t\t\t 欢迎使用小霸王学习机\n");
  waitTime(90000000);
  srand(time(&now)%53);
  while(1)
  { int choice;
    system("cls");
    printf("\n\n\n");
    printf("\t\t 选择练习模式请按键 1\n");
    printf("\t\t 选择测验模式请按键 2\n");
    printf("\t\t\t\t 退出请按键 Esc\n");
    while(1)
    { choice=getch();
      if(choice==27) return;
      if(choice==49) break;
      if(choice==50) break;
    }
    if(choice==49) practice();
    else if(choice==50) test();
  }
}
void viewHistory(SCORE *t)
{ int i,choice;
  do
  { printf("\n\t\t 你是否想查看本次答题历史? 是(Y)/否(N):");
    choice=getche();
    printf("\n\n");
  }while(choice!='y'&&choice!='n'&&choice!='Y'&&choice!='N');
  if(choice=='n' || choice=='N' || t==NULL) return;
  system("cls");
  i=1;
  while(1)
  { printf("第%d题:\t%d%c%d=?\n",i,t->op1,t->opp,t->op2);
    printf("\t\t 你的回答:%d\t\t 正确答案: %d\n",t->answer,t->right);
```

```

printf("\t\t 你答%s 了! \n",t->answer==t->right?"对":"错");
if(t->next)    printf("pgDn");
if(t->pre)    printf("pgUp");
printf("Esc\n");
while(1)
{ choice=getch();
  if(choice==27)  return;
  if(choice==224)
  { choice=getch();
    if(choice==73&&t->pre)
    { t=t->pre;i--;
      break;
    }
    else if(choice==81&&t->next)
    { t=t->next;i++;
      break;
    }
  }
}
}
}

void freeList(SCORE *h)
{ SCORE *p;
  while(h)
  { p=h;
    h=h->next;
    free(p);
  }
}

void practice()
{ /*连续出题, 不计分, 直至用户回答结束*/
  void viewHistory(SCORE*);
  void freeList(SCORE*);

  SCORE *u, *v, *head;
  int range;
  int choice;
  system("cls");
  printf("\n\n\n\t 进入练习模式, 你将连续答题, 不计分. 想退出时输入 Esc\n");
  printf("\n\n\n\t\t\t 输入计算范围 0~");
  while(1)
  { scanf("%d",&range);
    if(range>=2) break;
    system("cls");
    printf("\n\n\n\t\t\t 范围不能为 0--%d, 请输入一个大于 1 的正数\n",range);
  }
  system("cls");
  head=NULL;
  for(;;)
  { u=(SCORE*)malloc(sizeof(SCORE));
    u->next=NULL;
    if(head==NULL)
    { head=v=u;
      u->pre=NULL;
    }
  }
}

```

```

else
{ u->pre=v;
  v=v->next=u;
}
aSample(u, range);
if(u->answer==u->right)
{ printf("\t\t\t 正确!\n");
}
else
{ printf("\t\t\t 错了!\n");
  printf("\t\t\t 正确的答案是:%d\n", u->right);
}
printf("\n\n\t\t\t 任意键继续\n");
printf("\t\t\t 返回主菜单请按键 Esc\n");
choice=getch();
system("cls");
if(choice==27) break;
}
viewHistory(head);
freeList(head);          /*clean up*/
}
void test(void)
{ SCORE *head, *u, *v;
  long range, num;
  int i, t, err;
  system("cls");
  printf("\n\n\n\t\t 进入测试模式, 你的回答将计分. \n");
  printf("\n\n\n\t\t\t 输入计算范围 0 ~");
  while(1)
  { scanf("%d*c", &range);
    if(range>=2) break;
    system("cls");
    printf("\n\n\n\t\t\t 范围不能为 0 ~ %d, 请输入一个大于 1 的正数\n", range);
  }
  printf("\n\t\t\t 输入解题的道数");
  while(1)
  { scanf("%d*c", &num);
    if(num>=2) break;
    system("cls");
    printf("\n\n\t\t\t 解题道数不能为 0 ~ %d, 请输入一个大于 1 的正数\n", num);
  }
  system("cls");
  head=NULL;
  for(i=0; i<num; i++)
  { u=(SCORE*)malloc(sizeof(SCORE));
    u->next=NULL;
    if(head==NULL)
    { head=v=u;
      u->pre=NULL;
    }
    else
    { u->pre=v;
      v=v->next=u;
    }
    aSample(u, range);
  }
}

```

```
err=0;
system("cls");
printf("\t\t\t\t\t 测验完毕,结果如下:\n");
for(t=1,u=head;u!=NULL;u=u->next,t++)
{ if(u->answer!=u->right)
    { err++;
      printf("\t\t\t\t\t 第%d题错误,正确的解答是:\n",t);
      printf("\t\t\t\t\t %d%c%d=%d\n\n",u->op1,u->opp,u->op2,u->right);
      waitTime(200000000);
    }
}
printf("\n\t\t\t\t\t 得%d分.\n", (num-err)*100/num);
printf("\t\t\t\t\t 按任意键继续      ");
getch();
system("cls");
viewHistory(head);
freeList(head);                                /*clean up*/
}

void aSample(SCORE * u,int range )
{ int opp0, t;
  opp0=rand()%4;                               /*产生运算符*/
  if(opp0==0)
  { u->opp='+';
    /*产生整数*/
    u->op1=rand()%(range-1)+1;
    u->op2=rand()%(range-1)+1;
    if(u->op1+u->op2>range)
        u->op2=range-u->op1;
    u->right=u->op1+u->op2;
  }
  else if(opp0==1)
  { u->opp='-';
    /*产生整数*/
    u->op1=rand()%range;
    u->op2=rand()%range;
    if(u->op1<u->op2)
    { int temp=u->op1;
      u->op1=u->op2;
      u->op2=temp;
    }
    u->right=u->op1-u->op2;
  }
  else if(opp0==2)
  { u->opp='*';
    /*产生整数*/
    u->op1=rand()%range;
    u->op2=rand()%range;
    if(u->op1*u->op2>range)
        u->op2=range/u->op1;
    u->right=u->op1*u->op2;
  }
  else if(opp0==3)
  { u->opp='/';
    u->op1=rand()%(range-1)+1;
    u->op2=rand()%(range-1)+1;
    if(u->op2==0 || (u->op1)%(u->op2)!=0)
```



```

    { t=(int)sqrt(range);
      if(t<2)t=2;
      u->right=rand()%(t-1);
      u->op2=rand()%(t-1)+1;
      u->op1=(u->right)*(u->op2);
    }
    else u->right=u->op1/u->op2;
  }
  printf("\n\n\n\t\t\t%d %c %d=",u->op1,u->opp,u->op2);
  scanf("%d%c",&u->answer);
}

```

3. 五子棋是一种棋类游戏，棋盘大小为 15×15，即 15 行 15 列。

黑白双方轮流下子，黑子先行，由首先连成五子一条线的一方获胜。五子一条线可以是横向、纵向或对角方向。

要求实现双人对战、人机对战两种方式。

为了记录下棋结果，表示棋盘的二维数组的元素可用以下结构表示：

```

typedef struct
{ /*棋盘信息*/
    int step; /*步数，步数为 0 表示该位置为空*/
    int color; /*棋子颜色*/
}Ele;

```

【参考程序】

/*以下程序采用图示的方法实现双方轮换下棋，光标停于最近下棋处，下棋者通过光标移动键(↑、↓、←、→)，将光标移至下一步下棋位置，然后按【Enter】键，则光标所在位置将显示当前方所下的棋*/

```

#include<stdio.h>
#include<conio.h>
#include<windows.h>
#include<time.h>

#define TEXTS 7 /*文本颜色*/
#define CURSOR 48 /*光标颜色*/
#define CHESSBOARD 352 /*棋盘颜色*/
#define WHITECHESS 103 /*白棋颜色*/
#define SELECTEDWHITE 55 /*白棋被选中时的颜色*/
#define BLACKCHESS 96 /*黑棋颜色*/
#define SELECTEDBLACK 48 /*黑棋被选中时的颜色*/
#define qx1_num 27 /*防御棋形的数量*/
#define qx2_num 26 /*攻击棋形的数量*/

typedef struct node
{ /*棋盘信息*/
    int step; /*步数，步数为 0 表示该位置为空*/
    int color; /*棋子的颜色*/
} NODE;

typedef struct point
{ /*点*/
    int x;

```

```

    int y;
}_POINT;

typedef struct qixing                                /*棋形信息*/
{ char qx[8];                                        /*棋形*/
  int value;                                         /*相应的权值*/
}QIXING;

HANDLE hOutput=GetStdHandle(STD_OUTPUT_HANDLE); /*得到标准输出的句柄*/
_POINT cursor;                                    /*游戏中, 光标所在的当前位置*/
int
direction[8][2]={ {0,-1},{0,1},{-1,0},{1,0},{-1,-1},{1,1},{-1,1},{1,-1}};
/*向量数组, 依次为左、右、上、下、左上、右下、右上、左下*/

QIXING qx1[qx1_num]={ {"x1111",200000}, {"1x111",200000},
{"11x11",200000}, /*连五型*/
{"0x1110",6000}, {"01x110",6000}, {"101x101",6000}, /*活四型*/
{"0x111",1100}, {"x0111",1100}, {"0x1011",1100}, {"0x1101",1100},
{"01x11",1100}, /*冲四型*/
{"011x1",1100}, {"1x011",1100}, {"10x11",1100}, {"11x01",1100},
{"1x101",1100}, /*冲四型*/
{"x011102",250}, {"0x110",250}, {"01x10",250}, {"0x01102",240},
{"0x101",240}, /*活三型*/
{"0x112",20}, {"01x12",10}, {"011x2",20}, {"1x12",10},
{"0x10",20}, {"0x010",5}}; /*死三活二*/
/*防御的基本棋形及权值, 0 为空, 1 为对手, 2 为自己, x 为下棋位置*/

QIXING qx2[qx2_num]={ {"x1111",2000000}, {"1x111",2000000},
{"11x11",2000000}, /*连五型*/
{"0x1110",24000}, {"01x110",24000}, {"101x101",24000}, /*活四型*/
{"0x111",2000}, {"x0111",1900}, {"0x1011",1900}, {"0x1101",2000},
{"01x11",2000}, /*冲四型*/
{"011x1",2000}, {"1x011",1900}, {"10x11",2000}, {"1x101",2000},
{"x01112",2000}, /*冲四型*/
{"0x110",850}, {"01x10",850}, {"0x0110",840}, {"0x101",840}, /*活三型*/
{"0x112",125}, {"01x12",125}, {"011x2",115}, {"1x12",115},
{"0x10",125}, {"0x010",110}}; /*死三活二*/
/*攻击的基本棋形及权值, 0 为空, 1 为自己, 2 为对手, x 为下棋位置*/

/*-----界面部分-----*/

void textcolor(int color)
/*更改字体颜色*/
{
    SetConsoleTextAttribute(hOutput,color);
}

void gotoxy(int x, int y)

```

```
{ /*将光标移动到指定位置*/
COORD coordScreen={0,0};
coordScreen.X=x;
coordScreen.Y=y;
SetConsoleCursorPosition(hOutput,coordScreen);
}

void printnode(NODE chessboard[][15],int x,int y)
/*打印棋盘上的一个点*/
{ textcolor(CHESSBOARD);
if(chessboard[x][y].step==0)
{ if(x==cursor.x && y==cursor.y)
    textcolor(CURSOR); /*如果光标在这个点,改成光标颜色*/
    switch(x)
    {case 0:
        if(y==0)printf("┐"); /*左上角*/
        else if(y==14)printf("┌"); /*右上角*/
        else printf("└"); /*上边线*/
        break;
    case 3:
        if(y==0)printf("├"); /*左边线*/
        else if(y==3 || y==11)printf("+"); /*星位*/
        else if(y==14)printf("┤"); /*右边线*/
        else printf("⦶"); /*交叉点*/
        break;
    case 7:
        if(y==0)printf("├"); /*左边线*/
        else if(y==7)printf("+"); /*星位*/
        else if(y==14)printf("┤"); /*右边线*/
        else printf("⦶"); /*交叉点*/
        break;
    case 11:
        if(y==0)printf("├"); /*左边线*/
        else if(y==3 || y==11)printf("+"); /*星位*/
        else if(y==14)printf("┤"); /*右边线*/
        else printf("⦶"); /*交叉点*/
        break;
    case 14:
        if(y==0)printf("└"); /*左下角*/
        else if(y==14)printf("┘"); /*右下角*/
        else printf("┬"); /*下边线*/
        break;
    default:
        if(y==0)printf("├"); /*左边线*/
        else if(y==14)printf("┤"); /*右边线*/
        else printf("⦶"); /*交叉点*/
```

```

    }
}
else if(chessboard[x][y].color==0)
{ /*如果是白棋*/
    if(x==cursor.x&&y==cursor.y)
        textcolor(SELECTEDWHITE); /*被选中的白棋*/
    else textcolor(WHITECHESS); /*未被选中的白棋*/
    printf("●");
} /*打印棋子*/
else
{ if(x==cursor.x&&y==cursor.y)
    textcolor(SELECTEDBLACK); /*被选中的黑子*/
    else textcolor(BLACKCHESS); /*未被选中的黑子*/
    printf("●");
} /*打印棋子*/
}

void printchessboard(NODE chessboard[][15])
{ /*输出整个棋盘*/
    int i,j;
    char letter[]={"A B C D E F G H I J K L M N O\n"};

    for(i=0;i<15;i++)
    { /*行*/
        textcolor(TEXTS); /*改为文本颜色*/
        printf("%2d",15-i); /*打印行坐标*/
        for(j=0;j<15;j++) /*列*/
            printnode(chessboard,i,j); /*打印棋盘的每一块*/
        textcolor(TEXTS);
        printf("\n");
    }
    textcolor(TEXTS); /*改为文本颜色*/
    printf("%s",letter); /*打印列坐标*/
    printf("移动:方向键 下棋:Enter 悔棋:U 退出:F12");
}

void renew(NODE chessboard[][15], int x, int y)
{ /*更新棋盘指定位置的图像*/
    COORD coordScreen; /*系统提示符位置*/
    CONSOLE_SCREEN_BUFFER_INFO csbi; /*屏幕信息*/
    if(x<0||x>14||y<0||y>14)return; /*如果不在棋盘上直接返回*/
    if(!GetConsoleScreenBufferInfo(hOutput,&csbi)) /*获取屏幕信息*/
        return; /*不成功则返回*/
    coordScreen=csbi.dwCursorPosition; /*获取系统提示符位置*/
    gotoxy((y-1)*2+4,x+1); /*将系统提示符移动到棋盘的(x,y)所在位置*/
    printnode(chessboard,x,y); /*重新打印这一块*/
    SetConsoleCursorPosition(hOutput,coordScreen); /*系统提示符回复到原来位置*/
}

```

```

}
void showmenu()
{ /*输出主菜单*/
    textcolor(TEXTS);
    system("cls");
    printf("1.人机对战\n");
    printf("2.双人对战\n");
    printf("3.退出\n");
    printf("\n 请选择[1~3]:");
}
void showsubmenu()
{ /*打印子菜单*/
    textcolor(TEXTS);
    system("cls");
    printf("1.你先手\n");
    printf("2.电脑先手\n");
    printf("3.返回上级菜单\n");
    printf("\n 请选择[1~3]:");
}

int getchoose(int min, int max)
{ /*获取选项*/
    int choose;
    do
    { choose=getch()-48;
      }while(choose<min||choose>max);          /*过滤不在min到max之间的字符*/
    printf("%d",choose);                      /*屏幕回显*/
    return choose;
}

/*-----控制部分-----*/

bool quit;                                  /*是否按下了退出热键*/
bool regret;                               /*是否按下了悔棋热键*/
bool getmove(NODE chessboard[][15])
{ /*获取光标移动,并响应*/
    /*当按下悔棋、下子、退出热键时,返回true*/
    char c;
    for(;;)
    { c=getch();
      if(c==32)
          switch(getch())
          { case 72:                          /*上*/
              cursor.x--;
              if(cursor.x<0)cursor.x=0;
              renew(chessboard,cursor.x+1,cursor.y);
          }
      }
    }
}

```

```

        renew(chessboard, cursor.x, cursor.y);
        break;
    case 80:                                /*下*/
        cursor.x++;
        if(cursor.x>14) cursor.x=14;
        renew(chessboard, cursor.x-1, cursor.y);
        renew(chessboard, cursor.x, cursor.y);
        break;
    case 75:                                /*左*/
        cursor.y--;
        if(cursor.y<0) cursor.y=0;
        renew(chessboard, cursor.x, cursor.y+1);
        renew(chessboard, cursor.x, cursor.y);
        break;
    case 77:                                /*右*/
        cursor.y++;
        if(cursor.y>14) cursor.y=14;
        renew(chessboard, cursor.x, cursor.y-1);
        renew(chessboard, cursor.x, cursor.y);
        break;
    case 134:                               /*退出*/
        quit=true;
        return true;
    }
    else if(c==13&&chessboard[cursor.x][cursor.y].step==0)
        return true;                                /*下子*/
    else if(c=='U' || c=='u')
    {                                                  /*悔棋*/
        regret=true;
        return true;
    }
}

}

void beback(NODE chessboard[][15], int step)
{ /*悔棋*/
    int i, j, temp_x, temp_y;
    if(step==1) return;                                /*如果才开始, 直接返回*/
    if(step>2)
    { /*如果下了多于两步*/
        for(i=0; i<15; i++)                            /*搜索前两步所下的位置*/
            for(j=0; j<15; j++)
            { if(chessboard[i][j].step==step-1)
                { /*找到上一步*/
                    chessboard[i][j].step=0;            /*清空棋子标志*/
                    renew(chessboard, i, j);              /*重绘棋盘*/
                }
            }
        }
    }
}

```

```

    }
    else if(chessboard[i][j].step==step-2)
    { /*找到上两步*/
        chessboard[i][j].step=0; /*清空棋子标志*/
        tempx=cursor.x; /*记录光标位置*/
        tempy=cursor.y;
        cursor.x=i; /*将光标回复到两步前的位置*/
        cursor.y=j;
        renew(chessboard,i,j); /*重绘棋盘*/
        renew(chessboard,tempx,tempy); /*重绘光标原本所在处*/
    }
}
else if(step==2)
{ /*如果下了一步*/
    for(i=0;i<15;i++) /*搜索上一步所下的位置*/
    for(j=0;j<15;j++)
        if(chessboard[i][j].step==step-1)
        { /*找到上一步*/
            chessboard[i][j].step=0; /*清空棋子标志*/
            renew(chessboard,i,j);
        } /*重绘棋盘*/
    tempx=cursor.x; /*记录光标位置*/
    tempy=cursor.y;
    cursor.x=7; /*将光标移回棋盘中央*/
    cursor.y=7;
    renew(chessboard,i,j); /*重绘棋盘*/
    renew(chessboard,tempx,tempy); /*重绘光标原本所在处*/
}
}

/*-----判断部分-----*/

bool inside(int x, int y)
{ /*如果不在棋盘内返回 false, 否则返回 true*/
    if(x<0||x>14||y<0||y>14) return false;
    return(true);
}

int line(NODE chessboard[][15],int dirt,int x,int y,int color)
{ /*判断颜色为 color 的点(x,y), 在 dirt 方向上有多少相连的同色棋子*/
    int i;
    for(i=0;chessboard[x+direction[dirt][0]][y+direction[dirt][1]].step>0&&
        chessboard[x+direction[dirt][0]][y+direction[dirt][1]].color==color;i++)
    { x=x+direction[dirt][0];
      y=y+direction[dirt][1];
    }
}

```

```

        if(!inside(x,y)) return i;
    }
    return i;
}

bool win(NODE chessboard[][15],int x,int y,int color)
{ /*判断是否有人赢棋*/
    if(line(chessboard,0,x,y,color)+line(chessboard,1,x,y,color)>3)
        return true;
    if(line(chessboard,2,x,y,color)+line(chessboard,3,x,y,color)>3)
        return true;
    if(line(chessboard,4,x,y,color)+line(chessboard,5,x,y,color)>3)
        return true;
    if(line(chessboard,6,x,y,color)+line(chessboard,7,x,y,color)>3)
        return true;
    return false;
}

/*-----AI 部分-----*/

int attacktrend,defenttrend; /*攻击防御平衡权值*/

bool macthl(NODE chessboard[][15], int x, int y, int dirt, int kind, int color)
{ /*匹配在颜色为 color 的点(x,y), 在 dirt 方向上的第 kind 种防守棋形, 成功返回 true, 否则返回 false*/
    int k;
    char c;
    char *p;
    p=strchr(qx1[kind].qx,'x');
    for(k=0;k<=p-qx1[kind].qx;k++)
    { x-=direction[dirt][0];
      y-=direction[dirt][1];
    }
    for(k=0;(unsigned)k<strlen(qx1[kind].qx);k++)
    { x+=direction[dirt][0];
      y+=direction[dirt][1];
      if(!inside(x,y)) return(false);
      if(chessboard[x][y].step>0&&chessboard[x][y].color==color) c='2';
      else if(chessboard[x][y].step>0) c='1';
      else c='0';
      if(c=='0'&&qx1[kind].qx[k]=='x') continue;
      if(c!=qx1[kind].qx[k]) return(false);
    }
    return true;
}

int value_qx1(NODE chessboard[][15],int x,int y,int dirt,int color)

```



```

{ /*计算颜色为 color 的点 8 个方向上的防守权值之和*/
    int i;
    for(i=0;i<qx1_num;i++)
        if(macth1(chessboard,x,y,dirt,i,color))
            return qx1[i].value;
    return 0;
}

bool macth2(NODE chessboard[][15],int x,int y,int dirt,int kind,int color)
{ /*匹配在颜色为 color 的点(x,y), 在 dirt 方向上的第 kind 种进攻棋形, 成功返回 true, 否则返回 false*/
    int k;
    char c;
    char *p;
    p=strchr(qx2[kind].qx,'x');
    for(k=0;k<=p-qx2[kind].qx;k++)
    { x-=direction[dirt][0];
      y-=direction[dirt][1];
    }
    for(k=0;(unsigned)k<strlen(qx2[kind].qx);k++)
    { x+=direction[dirt][0];
      y+=direction[dirt][1];
      if(!inside(x,y))return false;
      if(chessboard[x][y].step>0&&chessboard[x][y].color==color) c='2';
      else if(chessboard[x][y].step>0) c='1';
      else c='0';
      if(c=='0'&&qx2[kind].qx[k]=='x') continue;
      if(c!=qx2[kind].qx[k])return false;
    }
    return true;
}

int value_qx2(NODE chessboard[][15],int x,int y,int dirt,int color)
{ /*计算颜色为 color 的点 8 个方向上的进攻权值之和*/
    int i;
    for(i=0;i<qx2_num;i++)
        if(macth2(chessboard,x,y,dirt,i,color))
            return qx2[i].value;
    return 0;
}

void AI(NODE chessboard[][15],int *x,int *y,int color)
{ /*AI 的主要函数, 将思考后的结果返回给*x 和*y*/
    int max=0; /*价值的最大值*/
    int maxi,maxj; /*获得最大值时所对应的坐标*/
    int i,j,k; /*循环控制变量*/
    int probability=1; /*几率参数*/

```

```

int value[15][15]={0};           /*棋盘上各点的价值*/
int valueattack[15][15]={0};     /*棋盘上各点的进攻价值*/
int valuedefent[15][15]={0};     /*棋盘上各点的防守价值*/
for(i=0;i<15;i++)               /*计算棋盘上各点的防守价值*/
for(j=0;j<15;j++)
{ if(chessboard[i][j].step>0)continue;
  for(k=0;k<8;k++)
    valuedefent[i][j]+=value_qx1(chessboard,i,j,k,color);
  if(maxi<valuedefent[i][j])
    maxi=valuedefent[i][j];      /*记录防守价值的最大值*/
}
for(i=0;i<15;i++)               /*计算棋盘上各点的进攻价值*/
for(j=0;j<15;j++)
{ if(chessboard[i][j].step>0) continue;
  for(k=0;k<8;k++)
    valueattack[i][j]+=value_qx2(chessboard,i,j,k,(color+1)% 2);
  if(maxj<valuedefent[i][j])
    maxj=valuedefent[i][j];      /*记录进攻价值的最大值*/
}
if(rand()%(maxi+maxj+1)>maxi)
{ /*如果防守价值的最大值比较低*/
  attacktrend=1;                 /*AI 优先进攻*/
  defenttrend=1;
}
else
{ /*相反*/
  attacktrend=1;                 /*AI 优先防守*/
  defenttrend=2;
}
for(i=0;i<15;i++)               /*根据各点的进攻和防守价值*/
for(j=0;j<15;j++)
{ /*计算最终价值*/
  value[i][j]=valuedefent[i][j]*defenttrend+valueattack[i][j]*attacktrend;
  if(max<value[i][j])
  { max=value[i][j];             /*记录价值的最大值*/
    maxi=i;                     /*以及相应的坐标*/
    maxj=j;
    probability=1;
  }
  else if(max==value[i][j])
  { /*如果出现相同价值的最大点*/
    if(rand()%(probability+1)<probability) /*随机决定选取哪一个*/
      probability++; /*由于前面的点容易被淘汰,所以相应提高前面的点的被选择的概率*/
    else
    { probability=1;             /*选择后面的点,则概率权值回复*/
      max=value[i][j];           /*记录*/
    }
  }
}

```

```

        maxi=i;
        maxj=j;}
    }
}
*x=maxi;          /*返回价值最大的点*/
*y=maxj;
}

/*-----主要部分-----*/

bool vshuman;      /*对手是否是人*/

void Vs(bool human)
{   /*对局主要函数*/
    int i,j;
    int color=1;    /*黑棋先走*/
    int lastx,lasty; /*光标的上一个位置*/
    int computer;   /*计算机执黑还是执白*/
    NODE chessboard[15][15]={0,0}; /*棋盘*/
    if(!human)
    {   /*对手是计算机*/
        showsubmenu(); /*选择谁是先手*/
        switch(getchoose(1,3))
        { case 1:
            computer=0;
            attacktrend=1; /*计算机先手则优先进攻*/
            defenttrend=1;
            break;
        case 2:
            computer=1; /*计算机后手则优先防御*/
            attacktrend=1;
            defenttrend=2;
            break;
        case 3: return; /*返回上级菜单*/
        }
    }
    for(i=0;i<15;i++) /*清空棋盘*/
        for(j=0;j<15;j++)
            chessboard[i][j].step=0;
    cursor.x=7; /*光标居中*/
    cursor.y=7;
    quit=false; /*清空退出标志*/
    system("cls"); /*清屏*/
    printf("\n");
    printchessboard(chessboard); /*打印棋盘*/
    for(i=1;i<=225;)

```

```

{ /*行棋主循环*/
    gotoxy(0,0); /*系统光标移到屏幕左上角*/
    textcolor(TEXTS);
    printf("      第%03d手, ",i);
    if(color==1)printf("黑棋下");
    else printf("白棋下");
    regret=false; /*清空悔棋标志*/
    if(i>1)
    { /*第一子必须下在棋盘中央*/
        if(color!=computer||human) getmove(chessboard); /*该人走棋*/
        else
        { /*该计算机走棋*/
            lastx=cursor.x; /*记录光标位置*/
            lasty=cursor.y;
            AI(chessboard,&cursor.x,&cursor.y,color); /*计算机走棋*/
            renew(chessboard,lastx,lasty); /*更新棋盘*/
        }
    }
    if(quit)return; /*如果按了退出热键则返回*/
    if(regret)
    { /*如果按了悔棋热键则调用悔棋函数*/
        beback(chessboard,i);
        if(i>2) i-=2;
        else if(i==2)
        { i=1;
            color=(color+1)%2;
        }
    }
    else
    { /*如果没有按热键,则在光标位置下子*/
        chessboard[cursor.x][cursor.y].step=i++;
        chessboard[cursor.x][cursor.y].color=color;
        renew(chessboard,cursor.x,cursor.y);
        color=(color+1)%2;
    }
    if(win(chessboard,cursor.x,cursor.y,(color+1)%2)&&!regret)
    { /*有人赢*/
        textcolor(TEXTS);
        gotoxy(0,0);
        printf("      ");
        gotoxy(0,0);
        if(color==1) printf(" 白棋赢了!");
        else printf("      黑棋赢了!");
        getch();
        return;
    }
}

```

```
    }
    gotoxy(0,0);
    printf("                                ");
    gotoxy(0,0);
    printf("                                平局!");
}

void main(void)
{ /*主函数*/
    srand((unsigned)time(NULL)); /*初始化随机种子*/
    for(;;)
    { showmenu(); /*输出主菜单*/
      switch(getchoose(1,3))
      {
        case 1:Vs(false);break;
        case 2:Vs(true);break;
        case 3: printf("\n");return;
      }
    }
}
```

第二部分 练习题与分析解答

为了切实掌握 C 语言的基本概念,本部分精选了选择题和填空题各 100 题供读者学习参考。通过对一些不熟悉的概念或理解不够准确的问题的解答,对读者准确理解 C 语言的基本概念会有很大的帮助。希望读者在学习本部分内容时,先自己给出解答,然后再阅读习题的参考解答。

一、选择题

1. 以下选项中,能作为 C 程序自定义标识符的是 ()。

- A. num(10) B. file_bak C. break D. class+2

【答案与解析】本题答案是 B。

C 语言规定,标识符要由英文字母或下画线字符开头,由英文字母、下画线和数字构成的字符序列,并且不可以与 C 语言规定的关键字相同。在供选择的答案中,选项 A 含有括号字符,不是标识符,类似于函数调用表达式;选项 B 符合 C 语言关于标识符的命名规则,所以它可以作为标识符;选项 C 是 C 语言的保留字,不可以再被程序设定为标识符;选项 D 是一个加法运算算术表达式,也不可以作为程序的标识符。所以答案是 B。

2. 下列选项中,不是 C 语言提供的合法关键字的是 ()。

- A. switch B. begin C. case D. default

【答案与解析】本题答案是 B。

因为 C 语言的关键字表中没有 begin,它不是 C 语言的关键字。所以答案是 B。

3. 键盘符号是指如其标记所示能显示的字符。下列字符中属于键盘符号的是 ()。

- A. \ B. \n C. \t D. \b

【答案与解析】本题答案是 A。

键盘符号是指键盘上有标记,并能在显示屏幕上直接以其标记字样显示的字符。有许多键盘上有标记的符号,它们不是用于直接显示的,输入这种字符用于表示特定的意义,如常用的回车符。为了能让 C 程序标记这种符号,程序采用转义字符的方式书写这种字符。例如, '\n'、'\t'、'\b' 都不是键盘字符,在 C 语言中,它们都得用转义字符来表达。只有字符 '\ ' 才是键盘字符。所以答案是 A。但在 C 程序中,反斜杠字符 '\ ' 已特别用做转义字符的引导符,它也得用转义字符的表达形式书写,将其写成 '\\ '。

4. 下列选项中,不是用来表达转义字符的是 ()。

- A. \\ B. \ ' C. 074 D. \0

【答案与解析】本题答案是 C。

转义字符有 3 种书写形式:反斜杠字符后接上某个规定的字符;反斜杠字符后接上 1~3 个八进制数字;反斜杠字符和字符 x 之后接上 1~2 个十六进制数字,后两种分别以八进制数和十六进制数直接给出字符的 ASCII 代码值,而 074 是八进制整数,不是转义字符。所以答案是 C。

5. 下列选项中, 可以作为“字符串常量”的是 ()。

- A. ABC B. "ABC" C. 'abc' D. 'a'

【答案与解析】本题答案是 B。

C 程序中, 一个字符串常量是表示一个字符序列, 书写时, 用双引号字符前后括住这个字符序列。所以只有"ABC"是一个正确的字符串常量, 其余都不是。其中, ABC 可作为标识符, 'abc'不能出现在 C 程序中, 'a'是一个字符常量。所以答案是 B。

6. 在用 1 个字节存储一个字符的系统中, '\n'在内存占用的字节数是 ()。

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

【答案与解析】本题答案是 A。

一般来说, 一个字符在内存中只占 1 个字节, '\n'是转义字符, 其意义是换行符, 它作为一个字符存储, 在内存也只占 1 个字节。所以答案是 A。

7. 字符串 "xyz"在内存占用的字节数是 ()。

- A. 3 B. 4 C. 6 D. 8

【答案与解析】本题答案是 B。

字符串中的每个字符占 1 个字节, 但 C 程序在存储字符串时, 要在最后一个有效字符后面接上 1 个字符串结束的标记符'\0'。这样, 存储字符串常量 "xyz" 需要 4 个字节。所以答案是 B。

8. 在以下选项中, 合法的长整型常量是 ()。

- A. 0L B. 4 962 710 C. 0.054 838 743 D. 2.186 9e10

【答案与解析】本题答案是 A。

为了表示不同范围的整数, 整型数据分短整型、基本型和长整型, 并根据对 3 种整型数据内部表示的最高位的不同理解, 又分别分成无符号和带符号两种。若要明确指明一个整数是长整型的, 必须在整数之后接上字符'L'。所以, 0L 是一个长整型的整型常量, 而 4 962 710 是基本整型数据, 而对于用 2 个字节表示一个基本整型数据的系统来说, 该整数将因超出范围而是一个错误的整数; 0.054 839 743 和 2.186 9e10 都是 double 型的实数。所以答案是 A。

9. 一个 char 型数据, 它的值在内存中存放的是 ()。

- A. ASCII 码值 B. BCD 码值 C. 内码值 D. 十进制码值

【答案与解析】本题答案是 A。

计算机存储字符通常是存储字符的某种代码值。有许多种字符编码的方法, 在 C 语言中, char 型数据在内存中存放的是 ASCII 码。所以答案是 A。

10. 设变量 m、n、a、b、c、d 均为 0, 执行(m=a==b)||(n=c==d)后, m、n 的值是 ()。

- A. 0、0 B. 0、1 C. 1、0 D. 1、1

【答案与解析】本题答案是 C。

计算(m=a==b)||(n=c==d)的过程是先计算逻辑或的左运算分量(m=a==b)的值, 由于赋值运算符的优先级比关系运算符的优先级低, 先计算 a==b。因 a、b 均为 0, 所以比较结果值为 1, 将 1 赋给变量 m, 使变量 m 的值变为 1, 同时, 这个赋值运算的结果也是 1, 这样, 逻辑运算的左运算分量的值为 1。由于逻辑或运算的左运算分量值为 1, 按照 C 语言对逻辑或运算优化计算的规定, 不再计算逻辑或的右运算分量, 而直接得到逻辑或运算的结果为 1。由于逻辑或的右运算分量不曾计算过, 所以变量 n 的值也没有变化。这样, 上述表达式计算后, 变量 m 的值为 1, 变量 n 的值为 0。所以答案是 C。

11. 设 a 为 5, 执行下列语句后, b 的值不为 2 的是 ()。

- A. $b=a/2$ B. $b=6-(-a)$ C. $b=a\%2$ D. $b=a<3?3:2$

【答案与解析】本题答案是 C。

因为两个整型数据相除结果是整数, 当 a 的值为 5 时, 计算表达式 $b=a/2$ 后, b 的值为 2。计算表达式 $b=6-(-a)$ 是先计算子表达式 $(-a)$, 先让 a 的值减 1 变成 4, 最后 b 的值为 2。在计算表达式 $b=a\%2$ 时, 求余运算 $a\%2$ 的结果为 1, 最后变量 b 的值为 1。计算表达式 $b=a<3?3:2$ 时, 先计算条件表达式 $a<3?3:2$, 结果为 2, 最后变量 b 的值为 2。所以答案是 C。

12. 执行语句 $x=(a=3, b=a--);$ 后, x 、 a 、 b 的值依次为 ()。

- A. 3、3、2 B. 2、3、2 C. 3、2、3 D. 2、3、3

【答案与解析】本题答案是 C。

计算表达式 $x=(a=3, b=a--)$ 时, 先计算赋值号右端圆括号内的逗号表达式。逗号表达式要求按各子表达式顺序计算, 并以最后子表达式的值为整个逗号表达式的结果。所以该表达式计算时, 先让变量 a 的值为 3, $a--$ 的值是 a 当时的值, 以该值为结果赋给变量 b , 并以该值为逗号表达式的值赋给变量 x , 然后 a 的值减少 1。计算该表达式后, 变量 x 、 a 、 b 的值依次为 3、2、3。所以答案是 C。

13. 设整型变量 m 、 n 、 a 、 b 、 c 、 d 均为 1, 执行表达式 $(m=a>b)\&\&(n=a>b)$ 后, m 、 n 的值是 ()。

- A. 0、0 B. 0、1 C. 1、0 D. 1、1

【答案与解析】本题答案是 B。

表达式 $(m=a>b)\&\&(n=a>b)$ 是一个逻辑表达式, 它的计算过程是先计算逻辑与的左分量 $(m=a>b)$, 其中, 先计算 $a>b$ 不成立, 结果为 0, 将 0 赋给变量 m , 最后逻辑与的左分量结果为 0。由于逻辑与运算采用特别的优化计算规则, 当逻辑与的左分量结果为 0 时, 不再计算逻辑与的右分量, 而直接以 0 作为逻辑与的结果。所以, 上述表达式计算使 m 的值变为 0, 而 n 的值没有变, 依旧为 1。所以答案是 B。

14. 设有语句 $\text{int } a=3;$, 则执行了语句 $a+=a-=a*a;$ 后, 变量 a 的值是 ()。

- A. 3 B. 0 C. 9 D. -12

【答案与解析】本题答案是 D。

由于赋值运算符的结合性自右至左, 语句 $a+=a-=a*a;$ 的执行是先计算 $a*a$, 得到值 9, 再计算 $a-=a*a$, 使 a 的值为 -6, 接着计算 $a+=a$, 使 a 的值为 -12。所以答案是 D。

15. 在以下运算符中, 优先级最高的运算符是 ()。

- A. $<=$ B. $=$ C. $\%$ D. $\&\&$

【答案与解析】本题答案是 C。

常规运算符的优先级自高到低的排列顺序是算术运算符、移位运算符、关系运算符、按位运算符、逻辑运算符、条件运算符、赋值运算符、逗号运算符。所以问题所给的 4 个运算符按优先级自高到低的排列顺序是 $\%$ 、 $<=$ 、 $\&\&$ 、 $=$ 。所以答案是 C。

16. 设整型变量 i 的值为 3, 则计算表达式 $i---i$ 后表达式的值为 ()。

- A. 0 B. 1 C. 2 D. 表达式出错

【答案与解析】本题答案是 A。

有些运算符只有一个字符, 也有许多运算符由两个字符组成, 特别是有些字符既可以是单字

符的运算符,又能组成双字符的运算符。编译系统在识别源程序时,通常是尽量多地读入字符,能组成多字符单词的先尽量组成多字符单词。所以表达式 $i---i$ 被编译器理解为 $(i---)i$ 。另外,当表达式中某变量有后缀自增和自减运算时,先按变量的原值计算表达式,然后再对变量进行自增和自减运算。同样,当表达式中某变量有前缀自增和自减运算时,先对变量进行自增和自减运算,然后再按变量的新值计算表达式。按这个约定,求表达式 $(i---)i$ 是计算 $i-i$ 的值为 0,然后再对 i 作自减运算。这样,表达式 $(i---)i$ 的值为 0。所以答案是 A。

17. 设整型变量 a 、 b 、 c 均为 2,表达式 $a+++b+++c++$ 的结果是 ()。

- A. 6 B. 9 C. 8 D. 表达式出错

【答案与解析】本题答案是 A。

与上一小题的原因相同,表达式 $a+++b+++c++$ 被系统理解成: $(a++)+(b++)+(c++)$ 。表达式“变量++”的值是变量的原来值,所以在变量 a 、 b 、 c 均为 2 的前提下,执行上述表达式,得到结果为 6。所以答案是 A。

18. 若已定义 x 和 y 为 `double` 类型,则表达式 $x=1, y=x+3/2$ 的结果是 ()。

- A. 1 B. 2 C. 2.0L D. 2.5L

【答案与解析】本题答案是 C。

由于变量 x 和 y 都是 `double` 类型,表达式 $x=1, y=x+3/2$ 是先计算 $x=1$,由于 1 是整型的, x 是 `double` 类型的,该计算先将整数 1 转换成 `double` 类型的 1.0L,并将 1.0L 赋给变量 x 。计算 $y=x+3/2$ 是先求 $3/2$ 的值,两整数相除,结果是整型值 1,然后在计算 $x+1$ 时,也先将 1 转换成 1.0L,再求得 $x+1$ 的值为 2.0L,最后将该值赋给变量 y ,并得到整个表达式的值为 2.0L。所以答案是 C。

19. 设 $a=1, b=2, c=3, d=4$,则表达式 $a<b?a:c<d?a:d$ 的结果为 ()。

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

【答案与解析】本题答案是 D。

条件运算符的优先级比关系运算符的优先级低,并且它的结合性是自右向左的,所以表达式 $a<b?a:c<d?a:d$ 可用圆括号等价地写成 $(a<b)?a:((c<d)?a:d)$ 。因 $a<b$ 成立,计算结果 a 的值为 1。所以答案是 D。

20. 设 a 为整型变量,下列 C 表达式中,不能表达数学关系式 $10<a<15$ 的是 ()。

- A. $10<a<15$ B. $a==11 \parallel a==12 \parallel a==13 \parallel a==14$
C. $a>10\&\&a<15$ D. $!(a<=10)\&\&!(a>=15)$

【答案与解析】本题答案是 A。

数学关系式 $10<a<15$ 表示 a 只能是 11、12、13、14 四个值之一。用 C 表达式表示这个条件,可写成 $a>10\&\&a<15$,或 $!(a<=10)\&\&!(a>=15)$,或 $a==11\parallel a==12\parallel a==13\parallel a==14$ 等。若写成 $10<a<15$,该算式的计算顺序是先求出 $10<a$ 的结果 0 或 1,然后以这个结果与 15 比较,是否小于 15,显然这与数学式子的意义是不一致的。所以答案是 A。

21. 下列表达式中,正确的赋值表达式是 ()。

- A. $a=7+b+c=a+7$ B. $a=7+b++=a+7$
C. $a=(7+b,b++,a+7)$ D. $a=7+b,c=a+7$

【答案与解析】本题答案是 C。

赋值运算符是运算符之一, 赋值运算也可以出现在表达式中。赋值运算时, 出现在赋值运算符左边的表达式应该是引用一个存储对象, 例如变量, 不能是一个数值计算表达式。表达式 $a=7+b+c=a+7$, 因算术运算符比赋值运算符的优先级高, 要把 $a+7$ 的结果赋给 $7+b+c$, 这是错误的。同样, 表达式 $a=7+b++=a+7$ 也是错误的, 而表达式 $a=(7+b, b++, a+7)$ 就是正确的, 它是先计算赋值号右边的括号表达式, 括号表达式内是一个逗号表达式, 逗号表达式要求按顺序逐一计算各子表达式, 并以最后一个子表达式的值作为逗号表达式的结果, 最后将该结果赋给变量 a , 而表达式 $a=7+b, c=a+7$ 则不是一个赋值表达式, 而是一个逗号表达式, 它由两个赋值子表达式组成。所以答案是 C。

22. 若有以下定义: `char a; int b; float c; double d;`, 则表达式 $a*b+d-c$ 值的类型为 ()。

- A. float B. int C. char D. double

【答案与解析】本题答案是 D。

基本类型的数据可以混合运算, 但运算时, 对要求运算分量类型相同的运算来说, 如果运算分量的类型不相同, 则会自动转换类型。类型自动转换规则规定类型低 (简单) 的向类型高 (复杂) 的转换。各类型由高到低的排列顺序是 long double、double、float、unsigned long、long、unsigned int、int。

这样, 若有定义 `char a; int b; float c; double d;`, 则表达式 $a*b+d-c$ 的值的类型是 double 的。首先计算 $a*b$, 得到 int 类型的值, 接着计算 $a*b+d$, 得到 double 类型的值, 最后计算 $a*b+d-c$, 得到 double 类型的值。所以答案是 D。

23. 表达式 $9!=7$ 的值是 ()。

- A. true B. 非 0 值 C. 0 D. 1

【答案与解析】本题答案是 D。

关系运算的结果只有 0 和 1, 若关系成立, 则结果为 1; 若关系不成立, 则结果为 0。因关系表达式 $9!=7$ 成立, 所以表达式的结果为 1。虽然 1 也是非 0 值, 但在供选择的答案中有 1, 所以最确切的答案是 D。

24. 表达式 $12 \mid 012$ 的运算结果是 (用十进制数表示) ()。

- A. 1 B. 0 C. 14 D. 16

【答案与解析】本题答案是 C。

按位运算将运算分量视做二进制的位串信息, 对两个运算分量的对应位进行运算, 产生二进制的位串信息结果。整数 12 的低 8 位二进制位是 00001100, 整数 012 是八进制数, 它的低 8 位二进制位是 00001010, 这两个二进制位串信息按位或运算, 得到结果是 00001110。所以答案是 C。

25. 设字符型变量 $a=3$, $b=6$, 计算表达式 $c=(a^b)<<2$ 后 c 的二进制值是 ()。

- A. 00011100 B. 00000111 C. 00000001 D. 00010100

【答案与解析】本题答案是 D。

a 的值为 3, 写成 8 位二进制数形式是 00000011, b 的值为 6, 写成 8 位二进制数形式是 00000110。表达式 $c=(a^b)<<2$, 先计算 a 与 b 的按位异或, 得到结果 00000101, 将这个结果向左移 2 位, 得到二进制的位串信息是 00010100。所以答案是 D。

26. 设 a 、 b 和 c 都是 `int` 型变量, 且 $a=3$, $b=4$, $c=5$, 则下面的表达式中值为 0 的是 ()。

A. `'a' && 'b'`

B. `a<=b`

C. `allb+c&&b-c`

D. `!((a<b)&&!c||1)`

【答案与解析】本题答案是 D。

对于表达式 `'a' && 'b'`, 两个非 0 运算分量的逻辑与结果为 1。对于表达式 `a<=b`, 由于关系表达式成立, 结果也为 1。又由于变量 a 为非 0, 表达式 `allb+c&&b-c` 的结果也为 1。对于表达式 `!((a<b)&&!c||1)`, 先是计算 `(a<b)` 得到结果为 1; 计算 `!c` 得到结果为 0; 再计算它们的逻辑与, 结果为 0。继续与 1 求逻辑或, 得到结果为 1。最后, 对 1 求逻辑非, 得到结果为 0。所以答案是 D。

27. 设 `ch` 是 `char` 型变量, 其值为 'A', 则下面表达式的值是 ()。

`ch=(ch>='A'&&ch<='Z')?(ch-'A'+'a'):ch`

A. A

B. a

C. Z

D. z

【答案与解析】本题答案是 B。

由于字符型变量 `ch` 的值为 'A', 计算表达式 `ch=(ch>='A'&&ch<='Z')?(ch-'A'+'a'):ch` 时, 先计算其中的条件表达式, 由于条件 `(ch>='A'&&ch<='Z')` 成立, 该条件表达式以 `ch-'A'+'a'` 为结果, 即将字符 'a' 赋给变量 `ch`。所以答案是 B。

28. 设有无符号整型变量 i 、 j 、 k , i 的值为 013, j 的值为 0x13, 计算表达式 `k=~i|j>>3` 后, k 的值是 ()。

A. 06

B. 0177776

C. 066

D. 0177766

【答案与解析】本题答案是 D。

将变量 i 的八进制数值 013 写成二进制形式为 00001011, 将变量 j 的十六进制值 0x13 写成二进制数形式为 00010011。表达式 `k=~i|j>>3` 的计算过程是先求 `~i`, 结果用八进制代码表示为 0177764; 接着计算 `j>>3`, 结果用八进制代码表示为 02。对它们进行按位或运算结果为 0177766。所以答案是 D。

29. 设 $a=3$, $b=4$, 执行 `printf("%d,%d",(a,b),(b,a));` 语句后的输出是 ()。

A. 3,4

B. 4,3

C. 3,3

D. 4,4

【答案与解析】本题答案是 B。

在调用格式输出函数的语句中, 其中每个格式符对应一个输出项, 格式符 `d` 要求输出项内容以十进制整数形式输出。语句中的第 1 个格式符对应输出项 `(a,b)` 是一个逗号表达式, 该表达式的值是 b 的值, 所以先输出 4, 接着输出字符逗号, 同样输出项 `(b,a)` 的值是 a 的值, 输出 3。所以语句执行将输出 4, 3。所以答案是 B。

30. 使用 `scanf("x=%f,y=%f",&x,&y)` 函数调用, 要使 x 、 y 均为 1.25, 正确的输入是 ()。

A. 1.25,1.25

B. 1.25 1.25

C. `x=1.25,y=1.25`

D. `x=1.25 y=1.25`

【答案与解析】本题答案是 C。

格式输入函数的格式字符串中的字符可以分成 3 类: 空格类字符、其他普通字符和格式转换说明。其中, 空格类字符用来自动跳过空格类字符, 直至下一个非空格类字符。普通字符要求输入字符与其完全相同。格式转换说明对输入字符列按格式转换说明进行转换, 得到内部值存储到对应输入项所指定的存储位置中。格式输入函数调用 `scanf("x=%f,y=%f",&x,&y)` 以普通字符 `x=` 开头, 输入时也要先输入 `x=`。然后是一个浮点数输入格式, 所有能构成一个浮点数 1.25 的字符序列都能

满足要求。接着是普通字符序列,y=, 在输入的浮点数之后也要有字符序列,y=。最后又是浮点数输入格式, 同样, 所有能构成一个浮点数 1.25 的字符序列都能满足要求。问题给出的供选择答案中只有 x=1.25,y=1.25 是正确的输入。所以答案是 C。

31. 设 a、b 为字符型变量, 执行 scanf("a=%c,b=%c",&a,&b)函数调用后使 a 为'A', b 为'B', 正确的输入是 ()。

- A. 'A' 'B' B. 'A', 'B' C. A=A,B=B D. a=A,b=B

【答案与解析】本题答案是 D。

函数调用 scanf("a=%c,b=%c",&a,&b)中, 普通字符必须按格式字符串要求按照原样输入, c 格式符对紧接的任何字符都输入。所以实现问题的要求, 输入字符序列应为 a=A,b=B。另外, 要特别指出, 在程序中, 为表示字符常量, 字符前后需加单引号, 但用字符格式输入字符时, 在要输入字符前后不必另输入单引号。若输入单引号, 则这个单引号也将作为字符被输入。所以答案是 D。

32. 设有 int i=010, j=10; 则执行 printf("%d, %d\n", ++i, j--);语句后的输出是 ()。

- A. 11,10 B. 9,10 C. 010,9 D. 10,9

【答案与解析】本题答案是 B。

变量 i 和 j 的初值分别为八进制数 010 和十进制数 10, 格式输出函数调用 printf("%d, %d\n", ++i, j--)中, ++i 的值是变量 i 增 1 后的值, 原值是八进制数 010, 等于十进制数 8, 所以输出 9。j--的值是变量 j 减 1 之前的值, 输出 10。格式字符串中的逗号是普通字符照原样输出。所以问题给出的格式输出函数调用将输出 9、10。所以答案是 B。

33. 设 x、y、z、t 均为 int 型变量, 则执行以下语句后, t 的值为 ()。

```
x=y=z=1;
t=++x || ++y && ++z;
```

- A. 不定值 B. 2 C. 1 D. 0

【答案与解析】本题答案是 C。

语句 x=y=z=1;t=++x||++y&&++z;使变量 x 的值为 1, 变量 t 是赋一个逻辑表达式的值。在 C 语言中, 由逻辑与和逻辑或构成的逻辑表达式有特别的计算规则, 顺序计算各逻辑子表达式, 并且一旦能得到逻辑表达式的结果时, 就立即结束逻辑表达式的计算, 不再计算后继逻辑子表达式。本题的逻辑表达式是两个运算分量的逻辑或, 而且左运算分量++x 的值非 0, 立即得到逻辑或运算的结果为 1, 并将 1 赋给变量 t, 所以变量 t 的值为 1。所以答案是 C。

34. 设 x、y、z 均为 int 型变量, 则执行以下语句后, x、y、z 的值为 ()。

```
x=1;y=0;z=2;
y++&&++z || ++x;
```

- A. 2、1、3 B. 2、0、3 C. 1、0、3 D. 2、1、2

【答案与解析】本题答案是 D。

语句 x=1;y=0;z=2; y++&&++z||++x;先使变量 x 的值为 1, 变量 y 的值为 0, 变量 z 的值为 2。由逻辑与和逻辑或构成的逻辑表达式有特别的计算规则, 顺序计算各逻辑子表达式, 并且一旦能得到逻辑表达式的结果, 就立即结束逻辑表达式的计算, 不再计算后继逻辑子表达式。计算 y++, 因 y 的值为 0, y++&&++z 的结果为 0, 这次计算变量 y 的值变成 1, 而变量 z 的值未改变。接着计算++x, 使 x 的值变为 2。所以, 经上述语句执行后, x、y、z 的值依次为 2、1、2。所以答案是 D。

35. 假定所有变量均已正确定义, 下列程序段运行后 x 的值是 ()。

```
a=b=c=0; x=35;
if(!a)x--;else if(b);if(c)x=3;else x=4;
```

A. 34 B. 4 C. 35 D. 3

【答案与解析】本题答案是 B。

变量 a、b、c 的值均为 0, 变量 x 的值为 35, 语句:

```
if(!a)x--;else if(b);if(c)x=3;else x=4;
```

由两个 if 语句组成。首先执行前一个 if 语句 if(!a)x--;else if(b);, 因变量 a 的值为 0, 条件!a 成立, 执行 x--使 x 的值变为 34。接着执行后继的 if 语句 if(c)x=3;else x=4;, 因变量 c 的值为 0, 条件不成立而执行 x=4, 最终使变量 x 的值为 4。所以答案是 B。注意: 前一个 if 语句的 else 部分的成分语句只有“if(b);”, 这是一个单分支 if 语句, 且其成分语句为空语句。

36. 下面的程序段所表示的数学函数关系是 ()。

```
y=-1;
if(x!=0)if(x>0)y=1; else y=0;
```

$$A. y = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

$$B. y = \begin{cases} 1 & x < 0 \\ -1 & x = 0 \\ 0 & x > 0 \end{cases}$$

$$C. y = \begin{cases} 0 & x < 0 \\ -1 & x = 0 \\ 1 & x > 0 \end{cases}$$

$$D. y = \begin{cases} -1 & x < 0 \\ 1 & x = 0 \\ 0 & x > 0 \end{cases}$$

【答案与解析】本题答案是 C。

程序段首先置变量 y 的值为-1, 接着按变量 x 值的不同情况重置变量 y 的值。重置的首要条件是 x!=0, 若变量 x 的值为 0, 则不再重置变量 y 的值, 所以在 x 值为 0 的情况下, y 的值是-1。在变量 x 的值不等于 0 的条件下, 若 x 的值大于 0, 重置变量 y 的值为 1; 若变量 x 的值小于 0, 重置变量 y 的值为 0。所以程序段实现当变量 x 的值为 0 时, 变量 y 的值为-1; 当变量 x 的值大于 0 时, 置变量 y 的值为 1; 当变量 x 的值小于 0 时, 置变量 y 的值为 0。所以答案是 C。

37. 设 int a=8,b=7,c=6,x=1;, 则执行语句

```
if(a>6)if(b>7)if(c>8)x=2;else x=3;
```

后 x 的值是 ()。

A. 0 B. 1 C. 2 D. 3

【答案与解析】本题答案是 B。

将上述语句写成易读的结构化形式:

```
if(a>6)
    if(b>7)
        if(c>8)
            x=2;
        else
            x=3;
```

该语句的执行过程是: 首先判定是否 a>6, 因 a 的值是 8, 条件成立; 接着判定是否 b>7, 因 b 的值是 7, 条件不成立。在上述语句中, 没有对应 if(b>7)的 else 分支, 上述语句就因 b>7 的条

件不成立而不执行任何有意义的操作, 结束该语句的执行。这样, 变量 a、b、c 和 x 的值都不会因执行上述语句而改变, 所以变量 x 的值依旧保持 1。所以答案是 B。

38. 执行下列程序代码:

```
x=9;
while(x>7)
{ printf("*");x--;}
```

后输出的结果是 ()。

A. ****

B. ***

C. **

D. *

【答案与解析】本题答案是 C。

上述代码以 x 的初值为 9, 在 $x > 7$ 成立的情况下循环, 每次循环输出一个 * 字符, 并让 x 的值减 1, 共执行 2 次循环, 也就共输出了 2 个 * 字符。所以答案是 C。

39. 下列语句中错误的是 ()。

A. while(x=y)5;

B. do x++ while(x==10);

C. while(0);

D. do 2; while(a==b);

【答案与解析】本题答案是 B。

while 语句的一般形式是:

while(表达式) 语句;

这里的表达式可以是任何合理的表达式, 语句可以是任何语句, 包括空语句或表达式语句。可能书写的 while 语句没有任何意义也是允许的。选项 A 和选项 C 的语句的句法没有任何错误。

do...while 语句的一般形式是:

do 语句 while(表达式);

其中的成分语句也可以是任何语句。选项 B 中的代码 x++ 是一个表达式, 不是语句, 所以是错误的。所以答案是 B。

40. 循环语句 for(x=0, y=0; (y!=123) || (x<4); x++); 的循环执行次数是 ()。

A. 无限次

B. 不确定次

C. 4 次

D. 3 次

【答案与解析】本题答案是 A。

for 循环语句的初始化部分置变量 x 和 y 的初值为 0, 循环条件是 $(y \neq 123) \text{ || } (x < 4)$, 每次循环后变量 x 的值增 1。由于循环过程中变量 y 的值未被修改过, 循环条件又是一个逻辑或, 其左分量 $(y \neq 123)$ 永远成立, 也就是循环条件永远成立, 该循环语句将循环执行无限次。所以答案是 A。

41. 若 i、j 已定义为 int 类型, 则以下程序代码中的内循环体的执行次数是 ()。

```
for(i=5; i; i--)
    for(j=0; j<4; j++){ ... }
```

A. 20

B. 24

C. 25

D. 30

【答案与解析】本题答案是 A。

问题所给的程序代码的外循环是一个 for 循环语句, 它的初始化部分置变量 i 的初值为 5, 循环条件简写成 i, 即 $i \neq 0$, 每次循环后变量 i 的值减 1, 所以外循环共控制 5 次循环。内循环也是一个 for 循环语句, 它的初始化部分置变量 j 的初值为 0, 循环条件是 $j < 4$, 每次循环后变量 j 的值增 1, 所以内循环共控制 4 次循环。这样, 如果内循环的循环体内未曾对变量 i 和 j 有修改, 在内、外循环一起控制下, 内循环的循环体共被重复执行 20 次。所以答案是 A。

42. 假定 a 和 b 为 int 型变量, 则执行以下语句后 b 的值为 ()。

```
a=1; b=10;
do
{ b-=a; a++;
} while(b--<0);
```

A. 9 B. -2 C. -1 D. 8

【答案与解析】本题答案是 D。

在这段程序代码中, 循环开始前变量 a 的值为 1, b 的值为 10, 每次循环从变量 b 减去 a, 并让 a 增 1, 并在循环条件判定时, 又让 b 减去 1。第 1 次循环后, 变量 b 的值变成 9, 变量 a 的值变为 2, 循环判断时, 因 b 的值大于 0, 循环条件不成立, 结束循环, 但在循环判断时, 让 b 减去了 1, 循环结束时, 变量 b 的值为 8。所以答案是 D。

43. 设 x 和 y 为 int 型变量, 则执行下面的循环后, y 的值为 ()。

```
for(y=1, x=1; y<=50; y++){
    if(x>=10) break;
    if(x%2==1) {x+=5; continue;}
    x-=3;
}
```

A. 2 B. 4 C. 6 D. 8

【答案与解析】本题答案是 C。

for 循环语句的初始化部分置变量 x 和 y 的初值为 1, 循环条件是 $y \leq 50$, 每次循环后变量 y 的值增 1, 控制循环最多执行 50 次。循环体有 3 条语句: 首先在发现变量 x 的值大于等于 10 时, 结束循环; 接着是当变量 x 除以 2 的余数为 1 (即变量 x 是奇数) 时, 让变量 x 的值增加 5, 让 x 变成偶数, 并直接进入下一轮循环; 如果变量 x 是偶数, 则从变量 x 减去 3, 让变量 x 变成奇数。由上述分析可知, 每两次循环使变量 x 的值增加 2。第 1 次循环后, 变量 x 的值变成 6。第 2 次循环后, 变量 x 的值变成 3。第 3 次循环后, 变量 x 的值变成 8。第 4 次循环后, 变量 x 的值变成 5。第 5 次循环后, 变量 x 的值变成 10。第 6 次循环时, 因变量 x 的值大于等于 10, 直接跳出循环, 这次循环是非正常结束, 对变量 y 的修正只执行了 5 次。循环结束后, 变量 y 的值增至 6。所以答案是 C。

44. 在 C 语言中, 下列说法中正确的是 ()。

- A. 编写程序时尽量不要使用“do 语句 while(条件);”的循环
- B. “do 语句 while(条件);”的循环中必须使用“break”语句退出循环
- C. “do 语句 while(条件);”的循环中, 当条件非 0 时将结束循环
- D. “do 语句 while(条件);”的循环中, 当条件为 0 时将结束循环

【答案与解析】本题答案是 D。

do...while 语句的一般形式是:

```
do 语句
while(表达式);
```

其语义是重复执行其成分语句, 直至表示条件的表达式值为 0 时结束。do...while 语句是正常使用的循环结构之一。do...while 语句的循环结束条件由 while 后的表达式值为 0 所控制, 并不一定要有 break 语句跳出循环来结束循环。do...while 语句在条件值非 0 时, 将继续循环, 而不是结束循环。条件值为 0 时, 才结束循环。所以答案是 D。

45. 若有以下程序代码, w 和 k 都是整型变量, 则不能与该程序代码等价的循环语句是 ()。

```

...
w=k;
LB: if (w==0) goto LE;
    w--;
    printf("*");
    goto LB;
LE:
...

```

A. for(w=k;w!=0;w--)printf("*");

B. w=k;

while(w--!=0)printf("*");

C. w=k;

D. for(w=k; w; --w)printf("*");

```

do { w--; printf("*");
    } while(w != 0);

```

【答案与解析】本题答案是 C。

问题所给出的程序段用 goto 语句构造一个循环控制结构,该循环结构的初值是让变量 w 的值为 k 的值,循环结束条件是 w 的值等于 0,循环体是让变量 w 的值减 1,并输出一个字符*。上述循环控制过程用 for 循环可描述如下:

```
for (w=k;w!=0;w--) printf("*");
```

或写成:

```
for (w=k;w;--w) printf("*");
```

若用 while 语句,可描述成如下形式:

```
w=k;
```

```
while (w--!=0) printf("*");
```

w++; /*在 w 等于 0 时,也执行了一次 w--,这里是对 w 最后减 1 的补偿*/

或更直观地写成:

```
w=k;
```

```
while (w!=0) {w--; printf("*"); }
```

若用 do...while 语句,可描述成如下形式:

```
w=k;
```

```
if (w)
```

```
do
```

```
{ w--;printf("*");
```

```
}while (w);
```

若写成:

```
w=k;
```

```
do
```

```
{ w--; printf("*");
```

```
}while (w!=0);
```

是不正确的,因为原代码当 k 的值为 0 时,不输出字符*,而上面的改写,当 k 的值为 0 时,将输出许多的字符*。所以答案是 C。

46. 若有下列说明,则数值为 4 的表达式是 ()。

```
int a[12]={1,2,3,4,5,6,7,8,9,10,11,12};
```

```
char c='a',d,g;
```

A. a[g-c]

B. a[4]

C. a['d'-'c']

D. a['d'-c]

【答案与解析】本题答案是 D。

数组元素的下标自 0 开始顺序编号, 值为 4 的元素是 `a[3]`。所以只有下标表达式的值为 3 才是正确的。下标表达式 `g-c` 中的变量 `g` 的值还未被设定, 其值不确定。`a[4]` 的值为 5, 下标为 4 是不对的。`'d'-'c'` 的值为 1, `a[1]` 的值是 2, 也是不对的。变量 `c` 的值为 `'a'`, `'d'-c=3`。所以答案是 D。

47. 设有定义: `char s[12]={ "string" };`, 则 `printf("%d\n",strlen(s));` 的输出是 ()。

- A. 6 B. 7 C. 11 D. 12

【答案与解析】本题答案是 A。

在 C 语言中, 字符串是指在有效字符之后有字符串结束标记符的字符序列, 并约定字符串的长度是指字符串中有效字符的个数, 不包括字符串的结束标记符。存于字符数组 `s[]` 中的字符串是 `"string"`, 该字符串的长度为 6, 输出该字符串的长度应是 6。所以答案是 A。

48. 下列 C 代码中正确的是 ()。

- A. `char a[3][ ]={'abc','1'};` B. `char a[ ][3]={'abc','1'};`
C. `char a[3][ ]={'a',"1"};` D. `char a[ ][3]={"a","1"};`

【答案与解析】本题答案是 D。

如定义数组时有初始化, 其最高维的长度可以省略, 由系统根据初始化时的初值个数确定, 而其他维的长度是不可以省略的。对二维字符数组的初始化, 可以按行用字符串对其初始化, 也可以按二维数组元素的存储顺序逐一用字符对其元素初始化。在选项 A 和 C 中, 有不是最高维的长度被省略的, 所以是错误的。在选项 A 和 B 中, 将字符串常量写成 `'abc'`, 这也是错误的。只有选项 D, 用字符串按行给二维字符数组初始化, 才是正确的。所以答案是 D。

49. 下列 C 代码中, 合法的数组定义是 ()。

- A. `int a[ ]={"string"};` B. `int a[5]={0,1,2,3,4,5};`
C. `char a={"string"};` D. `char a[ ]={0,1,2,3,4,5};`

【答案与解析】本题答案是 D。

选项 A 错误的原因是整数数组不可以用字符串对其初始化。选项 B 错误的原因是数组初始化时, 初始化指定的值的个数不能多于数组的元素个数。选项 C 错误的原因是能用字符串初始化的只有字符指针变量或字符数组, 字符串不能对其他类型的变量初始化, 包括字符变量。选项 D 是正确的, 因为字符数组可以用小整数 (作为字符的 ASCII 代码值) 对其元素初始化。所以答案是 D。

50. 字符串 `"m\x43\\\np\102q"` 的长度是 ()。

- A. 7 B. 9 C. 15 D. 16

【答案与解析】本题答案是 A。

字符串的长度是指字符串中有效字符的个数。在 C 程序中, 组成字符串的字符可以是一般的普通字符, 也可以是转义字符。其中, 转义字符又可以用多种不同形式来表达: 反斜杠字符之后接上预先规定的字符, 用来表示特殊字符或特定意义的控制字符, 如单引号字符、双引号字符、反斜杠字符等特殊字符, 换行、回车、退格等控制字符。反斜杠字符之后接上 1~3 位八进制数字, 这是直接用八进制数字表示字符的 ASCII 码的方式。反斜杠字符之后接上字符 `x`, 再接 1~2 个十六进制数字, 这是直接用十六进制数字表示字符的 ASCII 码的方式。后两种形式能表示 C 语言允许的任何字符。在本题给出的字符串中, 字符 `'\x43'` 是用十六进制数表示的一个字符, 字符 `'\\'` 表示一个反斜杠字符, 字符 `'\n'` 表示一个换行字符, 字符 `'\102'` 是用八进制数表示的一个字符。这样, 所给字符串只有 7 个有效字符, 该字符串的长度是 7。所以答案是 A。

51. 函数调用 `strcat(strcpy(str1, str2), str3)` 的功能是 ()。

- A. 将字符串 `str1` 复制到字符串 `str2` 中后再连接到字符串 `str3` 之后
- B. 将字符串 `str1` 连接到字符串 `str2` 之后再复制到字符串 `str3` 中
- C. 将字符串 `str2` 复制到字符串 `str1` 中后再将字符串 `str3` 连接到字符串 `str1` 之后
- D. 将字符串 `str2` 连接到字符串 `str1` 之后再将字符串 `str1` 复制到字符串 `str3` 中

【答案与解析】本题答案是 C。

函数调用 `strcat(s1, s2)` 是将 `s2` 字符串复制连接到 `s1` 字符串之后, 使得 `s1` 字符串更长。函数调用 `strcpy(s1, s2)` 是将 `s2` 字符串复制到 `s1`, 使 `s1` 字符串的内容与 `s2` 字符串的内容相同。函数调用 `strcat(strcpy(str1, str2), str3)` 是先执行 `strcpy(str1, str2)`, 然后再执行 `strcat(str1, str3)`, 所以其功能是将字符串 `str2` 复制到字符串 `str1` 中, 然后再将字符串 `str3` 复制连接到字符串 `str1` 之后。所以答案是 C。

52. 设有如下定义, 则正确的叙述为 ()。

```
char x[]={"abcdefg"}, y[]={'a','b','c','d','e','f','g'};
```

- A. 数组 `x[]` 和数组 `y[]` 等价
- B. 数组 `x[]` 和数组 `y[]` 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

【答案与解析】本题答案是 C。

不指定数组长度的字符数组定义, 其所需要的字节数可由初始化中给出的值的个数确定。字符数组可以用字符串初始化, 也可用字符逐个给字符数组的元素初始化。尽管用字符初始化时给出了字符串中的全部字符, 但字符串自动会含有字符串的结束标记符, 字符串初始化所需要的字节个数会比用同样多的字符初始化多 1 个字节。所以答案是 C。

53. 设有定义 `int a=3, b, *p=&a;`, 则下列语句中不能使 `b` 的值为 3 的语句是 ()。

- A. `b=&a;`
- B. `b=*p;`
- C. `b=a;`
- D. `b=*a;`

【答案与解析】本题答案是 D。

定义有 `int a=3, b, *p=&a;`, 对 `b` 赋值的表达式有 `*&a`、`*p`、`a`、`*a`。引用变量 `a` 的值有两种方法, 一是直接引用 `a`, 二是通过 `a` 的指针间接引用 `a`。对于后者, 又有多种表达方法, 通过指向 `a` 的指针 `p`, 间接引用 `a` 的内容, 如 `*p`, 或通过求地址运算符由变量 `a` 得到其指针 `&a`, 再由这指针表达式间接引用 `a` 的内容, 如 `*&a`。所以表达式 `*&a`、`*p` 和 `a` 都能引用变量 `a` 的值, 而使 `b` 的值为 3。表达式 `*a` 是错误的, 因为 `a` 不是指针变量, 不能对它运用取内容运算符 `*`。所以答案是 D。

54. 定义一个具有 10 个元素的整型数组, 应当使用的语句是 ()。

- A. `int a[10];`
- B. `int a[2,5];`
- C. `int a[];`
- D. `int *a[10];`

【答案与解析】本题答案是 A。

一个具有 10 个元素的整型数组可以是一维数组, 定义一维数组常用的一般形式是:
数组元素的类型 数组名 [数组元素个数常量表达式];

选项 A 是满足上述要求的代码形式; 选项 B 的方括号中有两个表达式, 用逗号分隔, 不符合上述一般形式, 所以是不正确的代码; 选项 C 未指定数组的元素个数, 它是对变量 `a` 的说明, 不是指定数组元素个数的数组定义; 选项 D 的代码所定义的数组元素的类型是指向整型数据对象的指针类型, 其元素不是整型的, 不符合本题要求。所以答案是 A。

55. 设 `char str1[10]="ABCDE", str2[10]="xyz";`, 则执行语句 `printf("%d", strlen(strcpy(str1, str2)))`; 后的输出结果是 ()。

- A. 9
- B. 8
- C. 5
- D. 3

【答案与解析】本题答案是 D。

上述代码是输出表达式 `strlen(strcpy(str1, str2))` 的值, 该表达式的求值顺序是先执行函数调用 `strcpy(str1, str2)`, 该函数将 `str2` 所指字符串内容复制到 `str1` 所指的字符数组中, 并返回存放于 `str1` 中的字符串的首字符的指针。经函数调用 `strcpy(str1, str2)` 后, `str1` 中的内容变成字符串 "xyz", 接着求更新成字符串 "xyz" 的 `str1` 中的字符串的长度并输出。这样, 结果将输出整数 3。所以答案是 D。

56. 设 `int x[]={1, 3, 5, 7, 9, 11, 13}`, `*p=x;` 则不能正确引用数组元素的表达式是 ()。

- A. `*(p--)` B. `*(--p)` C. `*(p++)` D. `*(++p)`

【答案与解析】本题答案是 B。

根据 C 语言关于数组变量名表达式就是数组首元素的指针的约定, 上述指针变量 `p` 的初值是指向数组 `x[]` 的首元素 `x[0]` 的。表达式 `*(p--)` 的求值顺序是先求 `p--`, 它的值是 `p` 的原先值, 即 `x[0]` 的指针 `&x[0]`。这样, 表达式 `*(p--)` 引用的是数组 `x[]` 的首元素 `x[0]`, 是对数组 `x[]` 的一个有意义元素的引用。这里要指出的是在该表达式求值后, 变量 `p` 的值将要减去一个单位, 使 `p` 指向 `x[0]` 的前一个位置, 这是一个在程序中没有确定意义的位置。表达式 `*(--p)` 的求值顺序是先求 `--p`, 它首先让变量 `p` 的值减去一个单位, 使 `p` 指向 `x[0]` 的前一个位置, 这是一个在程序中没有确定意义的位置。表达式 `*(--p)` 引用的不再是数组 `x` 的一个有意义的元素。表达式 `*(p++)` 的求值顺序是先求 `p++`, 它的值是 `p` 的原先值 `&x[0]`。这样, 表达式 `*(p++)` 引用的是数组 `x[]` 的首元素 `x[0]`, 是对数组 `x[]` 的一个有意义元素的引用。在该表达式求值后, 变量 `p` 的值将要增加一个单位, 使 `p` 是 `x[0]` 的后一个元素的指针, 即 `&x[1]`。表达式 `*(++p)` 的求值顺序是先求 `++p`, 它首先让变量 `p` 的值增加一个单位, 使 `p` 的值是 `x[0]` 的后一个元素的指针, 即 `&x[1]`。这样, 表达式 `*(++p)` 引用的是数组 `x` 的元素 `x[1]`, 也是一个有意义的引用。所以答案是 B。

57. 设 `int x[]={1, 2, 3, 4}`, `y, *p=&x[1];`, 则执行语句 `y=(*--p)++`; 后变量 `y` 的值为 ()。

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

【答案与解析】本题答案是 A。

上述指针变量 `p` 的初值是指向数组 `x[]` 的元素 `x[1]`。表达式 `y=(*--p)++` 的求值顺序是先求 `--p`, 它首先让变量 `p` 的值减去一个单位, 使 `p` 指向 `x[0]`。接着, 求表达式 `*--p`, 这是对 `x[0]` 的引用。表达式 `(*--p)++` 也就是 `x[0]++`, 它的值是 `x[0]` 的原来值 1。最后, 表达式 `y=(*--p)++` 将 1 赋给变量 `y`, 使 `y` 的值为 1。经上述表达式求值后, `x[0]` 的值变成 2。所以答案是 A。

58. 在 C 语言中, 下列说法中错误的是 ()。

- A. 函数定义可以分为两个部分: 函数说明部分和函数体
B. 主函数可以调用任何非主函数
C. 任何非主函数可以调用其他任何非主函数
D. 程序可以从任何函数开始执行

【答案与解析】本题答案是 D。

每个 C 函数的定义分两部分, 函数说明部分和函数体, 所以选项 A 是正确的叙述。在 C 语言中, 函数可以递归调用, 主函数可以调用程序中的任何函数, 当然可以调用任何非主函数, 所以选项 B 是一个正确的叙述。同样, 叙述 C 也是正确的。C 语言规定, C 程序只有一个主函数, 并总是从主函数开始执行, 不能从非主函数开始执行, 所以选项 D 的叙述是错误的。所以答案是 D。

59. 函数调用时, 下列说法中不正确的是()。

- A. 若用值传递方式, 则形式参数不予分配内存
- B. 实在参数和形式参数可以同名
- C. 主调函数和被调用函数可以不在同一个文件中
- D. 函数间传递数据可以使用全局变量

【答案与解析】本题答案是 A。

在 C 语言中, 函数的形式参数是函数的一种局部变量, 函数调用时, 不管哪种形式的参数传递方式, 系统都要为被调用函数的形式参数分配内存, 所以选项 A 是不正确的。因函数的形式参数当作函数的局部变量, 形式参数与实在参数是否同名, 系统是不予区别的, 所以选项 B 是正确的。为了能正确识别和翻译调用函数的代码, 在函数调用处, 有关被调用函数的函数名、返回结果以及形式参数等详细信息是应该知道的。如果被调用函数与调用代码在同一源程序文件中, 并且被调用函数的定义出现在调用代码之前, 在调用处能知道全部被调用函数的调用信息, 但如果被调用函数的定义出现在调用代码之后, 或被调用函数的定义在别的源程序文件中, 为了能知道如何正确调用被调用函数, 只要对被调用函数给出它的函数说明即可, 所以选项 C 是正确的, 主调函数和被调用函数不在同一个源程序文件中也是可以的。函数之间传递信息除通过函数的形式参数外, 也可以通过全局变量, 即调用处将有关被调用函数需要的信息预放在某些全局变量中, 被调用函数执行时, 引用这些全局变量, 并将执行结果也放在这些全局变量中, 函数调用返回后, 调用代码之后就可引用调用函数留在这些全局变量中的信息, 所以选项 D 也是正确的。所以答案是 A。

60. 执行下列程序:

```
#include<stdio.h>
int fun(int x1,int x2)
{ int x;
  x1>x2?(x=3):(x=4);
  return x+x1;
}
void main()
{ printf("%d\n",fun(7,8));
}
```

输出结果是()。

- A. 9
- B. 10
- C. 11
- D. 12

【答案与解析】本题答案是 C。

上述程序中, 主函数以实在参数 7 和 8 调用函数 fun(), 并输出函数的返回值。函数 fun() 被调用时, 形式参数 x1 的初值是 7, x2 的初值是 8。函数执行表达式语句 `x1>x2?(x=3):(x=4);` 是一个条件表达式, 因条件 `x1>x2` 不成立, 条件表达式求 `x=4`, 该表达式的求值使函数的局部变量 x 的值为 4。最后, 函数返回 `x+x1` 的值 11。所以答案是 C。

61. 设指针 x 指向的整型变量值为 25, 则 `printf("%d\n", ++*x);` 的输出是()。

- A. 23
- B. 24
- C. 25
- D. 26

【答案与解析】本题答案是 D。

若指针变量 x 指向某个变量, 如指向变量 v, 并且变量 v 的值是 25, 则表达式 `++*x` 的值是 26, 这是因为表达式的计算顺序可加圆括号表达成 `(++(*x))`, 首先是 `*x`, 是对 x 所指变量 v 的引用, 所

以++*x 就是++v。++v 是先让变量 v 增 1, 并以增 1 后的 v 的值为表达式++v 的结果, 其值是 26。所以答案是 D。

62. 若有说明 `int i, j=7, *p=&i;`, 则与 `i=j;` 等价的语句是 ()。

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

【答案与解析】本题答案是 B。

指针变量 p 指向变量 i 时, 表达式 `i=*p` 等价于 `i=i`; 表达式 `*p=*&j` 等价于 `i=j`; 而表达式 `i=&j` 企图将整型变量的指针赋给整型变量, 这是错误的; 表达式 `i=**p` 也是一种错误的表达式。p 是指针, *p 是 p 所指变量 j, **p 是企图将整型变量 j 当作指针, 并想通过 j 间接引用某个变量。所以答案是 B。

63. 若有说明语句 `int a[10], *p=a;`, 则对数组元素的正确引用是 ()。

- A. `a[p]` B. `p[a]` C. `*(p+2)` D. `p+2`

【答案与解析】本题答案是 C。

在 C 语言中, 约定数组名单独出现在表达式中时, 它表示数组首元素的指针。有 `int a[10]`, 则 a 可以作为 `&a[0]` 使用。另有整型指针变量 p, 代码 `p=a` 实现 p 指向数组 a[] 的首元素, 则表达式 `*(p+2)` 是引用数组元素 `a[2]`。表达式 `a[p]` 和 `p[a]` 都是不正确的, 下标必须是整型表达式, 不可以是指针表达式。表达式 `p+2` 是指针表达式, 它的值是 `&p[2]`。只有表达式 `*(p+2)` 引用数组 a[] 的元素 `a[2]`。所以答案是 C。

64. 下面各语句中, 能正确进行赋字符串操作的语句是 ()。

- 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);`

【答案与解析】本题答案是 C。

字符串最终存储于字符数组中, 存储字符串的字符数组可以是程序主动引入的 (定义或动态分配), 也可以是由系统分配的。其中, 字符数组用字符串常量初始化就是字符串存储于由程序引入的字符数组的例子。给字符指针赋字符串则是系统自动分配字符串存储空间例子。给字符指针赋字符串常量并不是将一个字符串常量存于字符指针变量中, 而是将字符串常量存储于常量区, 并将存储这个字符串常量的首字节地址赋给指针变量, 让指针变量指向字符串常量的首字符。对于以字符串常量作为字符数组初值的情况, 要求字符数组足够大, 能存得下字符串常量。这里有一个特别的规定: 若数组的大小少于存储字符串有效字符的字节个数, 系统将报告错误; 当字符数组的大小只能存储字符串的有效字符, 而不能存储字符串结束标记符时, 则存储于字符数组中的内容是字符序列, 因没有存储字符串结束标记符, 存储的内容就不是字符串, 如代码 `char a[5]="ABCDE"`。另外, 给字符数组元素逐一赋字符初值, 并在字符初值中没有字符串结束标记符, 则存于字符数组中的内容也不是字符串, 如代码 `char s[5]={'A','B','C','D','E'}`。特别要注意当字符指针还未指向某个字符数组的元素时, 不可以通过字符指针输入字符串, 如代码 `char *s; scanf("%s", s)`, 若写成 `char *str; scanf("%s", &str)` 则是错误的。由于 C 语言规定数组不能相互赋值, 所以只能将字符串常量赋给某字符指针, 如代码 `char *s; s="ABCDE"` 是正确的。实际上, 字符串 "ABCDE" 被存储于常量区中, 向指针变量赋的是字符指针, 让 s 指向其中的字符 'A'。所以答案是 C。

65. 若有以下定义, 则不能表示数组 a[] 的元素的表达式是 ()。

`int a[10]={1,2,3,4,5,6,7,8,9,10}, *p=a;`

- A. `*p` B. `a[10]` C. `*a` D. `a[p-a]`

【答案与解析】本题答案是 B。

上述代码定义了有 10 个元素的整型数组 `a[]` 和指针变量 `p`, 并让 `p` 指向数组元素 `a[0]`, 所以代码 `*p` 是引用 `a[0]`。由于数组 `a` 只有 10 个元素, 最后一个元素是 `a[9]`, 表达式 `a[10]` 是错误的。数组名 `a` 可以作为 `a[]` 的首元素的指针, 表达式 `*a` 就是 `a[0]`, 是对数组 `a[]` 的首元素 `a[0]` 的引用。指针 `p` 的值是 `a`, 表达式 `p-a` 的值是 0, 所以 `a[p-a]` 就是 `a[0]`。所以答案是 B。

66. 若有以下的定义, 则值为 3 的表达式是 ()。

```
int a[]={1,2,3,4,5,6,7,8,9,10}, *p=a;
```

A. `p+=2, *(p++)`

B. `p+=2, *++p`

C. `p+=3, *p++`

D. `p+=2, ++*p`

【答案与解析】本题答案是 A。

数组 `a` 有 10 个元素, 分别有值 1~10, 指针变量 `p` 指向 `a[0]`。选项 A 的逗号表达式 `p+=2, *(p++)`, 先是 `p+=2` 使 `p` 指向 `a[2]`, 接着是 `*(p++)`, 以当时 `p` 所指变量 `a[2]` 取内容 3 为表达式的值, 同时使 `p` 指向 `a[3]`。选项 B 的逗号表达式 `p+=2, *++p`, 先是 `p+=2` 使 `p` 指向 `a[2]`, 以后是 `*++p`, 又使 `p` 增 1, 让它指向 `a[3]`, 并取指针 `p` 所指变量 `a[3]` 的内容 4 作为表达式的值。选项 C 的逗号表达式 `p+=3, *p++`, 先是 `p+=3` 使 `p` 指向 `a[3]`, 以后是 `*p++`, 表达式的值是 `a[3]` 为 4, 而使 `p` 指向 `a[4]`。选项 D 的逗号表达式 `p+=2, ++*p`, 先是 `p+=2`, 使 `p` 指向 `a[2]`, 以后是 `++*p`, 因为当时的 `*p` 就是 `a[2]`, `++a[2]` 使 `a[2]` 增 1, 变成 4, 并以 4 为表达式的值。只有 `p+=2, *(p++)` 的值是 3。所以答案是 A。

67. 执行语句 `char a[10]="abcd", *p=a;` 后, `*(p+4)` 的值是 ()。

A. "abcd"

B. 'd'

C. '\0'

D. 不能确定

【答案与解析】本题答案是 C。

若有 `char a[10]="abcd", *p=a`, 则指针变量 `p` 指向 `a[0]`。在表达式 `*(p+4)` 中, `p+4` 指向 `a[4]`, `*(p+4)` 就是 `a[4]`。由于用字符串 "abcd" 给字符数组 `a[]` 赋初值, `a[4]` 的值是字符串结束标记符 '\0'。所以答案是 C。

68. 若有以下定义, 则数值为 4 的表达式是 ()。

```
int w[3][4]={{0,1},{2,4},{5,8}}, (*p)[4]=w;
```

A. `*w[1]+1`

B. `p++, *(p+1)`

C. `w[2][2]`

D. `p[1][1]`

【答案与解析】本题答案是 D。

二维数组定义时有多种赋初值的办法, 问题给出的代码是按行给数组的部分元素赋初值。它们分别是 `w[0][0]=0`、`w[0][1]=1`、`w[1][0]=2`、`w[1][1]=4`、`w[2][0]=5`、`w[2][1]=8`。根据约定, 未指定初值的元素自动全置为 0。指针变量 `p` 是一个指向具有 4 个 `int` 型元素的一维数组的指针, 定义时的初值使它指向数组 `w` 的第 1 行。选项 A 的代码 `*w[1]+1` 中的 `w[1]` 是指向 `w[1][0]` 的指针, `*w[1]` 就是 `w[1][0]`, 其值是 2, `*w[1]+1` 的值是 3。选项 B 的代码是逗号表达式 `p++, *(p+1)`, 先使 `p` 指向数组 `w` 的第 2 行, `*(p+1)` 中的 `p+1` 是指向数组 `w` 的第 3 行, `*(p+1)` 是指针值, 指向数组 `w` 的第 3 行的第 1 个元素, 即 `&w[2][0]`。选项 C 的代码 `w[2][2]` 引用数组 `w` 第 3 行的第 3 列元素, 其值是 0。选项 D 的代码 `p[1][1]` 引用数组 `w` 第 2 行的第 2 列元素 `w[1][1]`, 其值是 4。所以答案是 D。

69. 若有下面的程序代码, 则对数组元素的错误引用是 ()。

```
int a[12]={0}, *p[3], **pp, i;
for(i=0; i<3; i++) p[i]=&a[i+4];
pp=p;
```

- A. `pp[0][1]` B. `a[10]` C. `p[3][1]` D. `*(*(p+2)+2)`

【答案与解析】本题答案是 C。

上述代码定义变量 `a` 是有 12 个整型元素的数组，它的初值全是 0。`p` 是有 3 个元素的指针数组，每个指针都指向整型变量。`pp` 是指针的指针，它能指向一个指向整型变量的指针，`i` 是一个整型变量。执行代码 `for(i=0; i<3; i++)p[i]=&a[i+4]` 使指针数组 `p` 的元素 `p[0]` 指向 `a[4]`，`p[1]` 指向 `a[5]`，`p[2]` 指向 `a[6]`。代码 `pp=p` 使指针变量 `pp` 指向 `p[0]`。选项 A，代码 `pp[0][1]` 用指针表达可等价地写成 `*(pp+1)`。其中，`*pp` 就是 `p[0]`，`p[0]` 的值是 `&a[4]`，`*pp+1` 的值是 `&a[4]+1=&a[5]`，`*(pp+1)` 就是 `a[5]`。选项 B，代码 `a[10]` 当然是引用数组 `a` 的元素。选项 C，代码 `p[3][1]` 因为数组 `p` 只有 3 个元素，没有 `p[3]`，所以是错误的。选项 D，代码 `*(*(p+2)+2)` 中的 `(p+2)` 是 `&p[2]`，`*(p+2)` 是 `p[2]`，其值是 `&a[6]`，`*(p+2)+2` 的值是 `&a[8]`，`*(*(p+2)+2)` 引用的是 `a[8]`。所以答案是 C。

70. 若有如下定义和语句，则输出结果是 ()。

```
int **pp, *p, a=10, b=20;
pp=&p; p=&a; p=&b; printf("%d,%d\n", *p, **pp);
```

- A. 10, 20 B. 10, 10 C. 20, 10 D. 20, 20

【答案与解析】本题答案是 D。

上述代码定义变量 `pp` 是指针的指针，它能指向一个指向整型变量的指针。定义变量 `p` 是指针，它能指向一个整型变量。`a` 是一个整型变量，它有初值 10。`b` 也是一个整型变量，它有初值 20。代码 `pp=&p` 使 `pp` 指向 `p`，`p=&a` 使 `p` 指向 `a`，`p=&b` 又使 `p` 指向 `b`，不再指向 `a`。所以 `*p` 是引用 `b`，其值为 20。`**pp` 是通过 `pp` 间接引用 `p`，再通过 `p` 间接引用 `b`，所以也是 20。所以答案是 D。

71. 若有以下定义和语句，则对 `w` 数组元素的非法引用是 ()。

```
int w[2][3], (*pw)[3]; pw=w;
```

- A. `*(w[0]+2)` B. `*pw[2]` C. `pw[0][0]` D. `*(pw[1]+2)`

【答案与解析】本题答案是 B。

上述代码定义 2 行 3 列的二维数组 `w`，定义能指向有 3 个整型元素一维数组的指针 `pw`，并让 `pw` 指向数组 `w` 的第 1 行。选项 A，代码 `*(w[0]+2)` 中的 `w[0]` 是 `&w[0][0]`，`w[0]+2` 是 `&w[0][2]`，`*(w[0]+2)` 就是 `w[0][2]`。选项 B，代码 `*pw[2]` 中的 `pw[2]` 是 `&w[2][0]`，该数组 `w` 只有 2 行，没有 `w[2][0]`，代码 `*pw[2]` 是对数组 `w` 元素的非法引用。选项 C，代码 `pw[0][0]` 就是 `w[0][0]`。选项 D，代码 `*(pw[1]+2)` 中的 `pw[1]` 就是 `*(pw+1)`，即 `&w[1][0]`，`pw[1]+2` 就是 `&w[1][2]`，`*(pw[1]+2)` 就是 `w[1][2]`。所以答案是 B。

72. 函数调用语句 `f((e1, e2), (e3, e4, e5));` 中参数的个数是 ()。

- A. 1 B. 2 C. 4 D. 5

【答案与解析】本题答案是 B。

上述函数调用中，`(e1, e2)` 和 `(e3, e4, e5)` 是两个带括号的表达式，所以函数调用只提供两个实参，其中，第 1 个实参先计算 `e1` 的值，然后计算 `e2`，并以 `e2` 的值为实参，第 2 个实参顺序计算 `e3`、`e4`、`e5`，并以 `e5` 的值为实参。所以答案是 B。

73. 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

【答案与解析】本题答案是 C。

由于 C 函数内不能再定义函数, C 函数的存储类别只能是静态的或外部的。若定义函数时不指定存储类别是静态的, 则它的存储类别就是外部的 (extern), 即函数允许被程序的其他函数调用。所以答案是 C。

74. 以下对 C 语言函数的有关描述中, 正确的是 ()。

- A. 在 C 语言中, 调用函数时, 只能把实参的值传递给形参, 形参的值不能传递给实参
- B. C 函数既可以嵌套定义, 又可以递归调用
- C. 函数必须有返回值, 否则不能使用函数
- D. C 程序中, 有调用关系的所有函数必须放在同一个源程序文件中

【答案与解析】本题答案是 A。

C 语言规定, 调用函数时, 只能把实参的值传递给函数的形参。函数调用时, 形参从实参表达式得到初值, 形参也是函数的一种局部变量, 其值可以改变, 但形参的值不能传回给对应的实参。当函数设置非指针类型的形参时, 实参可以是同类型的一般表达式; 当函数设置指针类型的形参时, 对应的实参也必须是同类型的指针表达式, 所以叙述 A 是正确的。C 函数虽可以递归调用, 但同时规定, 在函数内不能再定义函数, 所以叙述 B 是错误的。通常 C 函数会有返回值, 但也可以没有返回值, 有许多情况, 函数的执行是完成某种预定的工作, 并没有返回值, 所以叙述 C 是不正确的。在 C 程序中, 如果函数未指定是静态的, 则就是外部的, 能让别的文件中的函数调用, 但函数要调用别的文件中的函数, 在调用代码之前, 须对它作说明, 所以叙述 D 也是不正确的。所以答案是 A。

75. 设函数的功能是交换两个变量的值, 且通过正确调用完成两个变量的值的交换。能正确执行此功能的函数是 ()。

A. funa(int *x, int *y)

```
{ int *p;  
  *p=*x; *x=*y; *y=*p;  
}
```

B. funb(int x, int y)

```
{ int t;  
  t=x; x=y; y=t;  
}
```

C. func(int *x, int *y)

```
{ int p;  
  p=*x; *x=*y; *y=p;  
}
```

D. fund(int x, int y)

```
{ int *p;  
  p=*x; *x=*y; *y=p;  
}
```

【答案与解析】本题答案是 C。

设计实现交换两个变量值的函数, 希望函数调用能改变调用环境中的变量, 方法之一是让函数设置指针类型的形参, 实参传递给形参的是希望交换值的两个变量的指针。函数的两个形参得到这两个变量的指针后, 函数就能通过形参间接引用要交换值的变量, 或引用它们的值, 或改变它们的值, 从而实现从函数传回信息存于调用环境的变量中。以上所述是设计交换变量值函数的一般原则, 具体编写函数时还须注意对上述原则的正确应用。

对于函数 funa(), 该函数的定义满足设置指针形参, 并通过指针形参间接引用环境变量的要求。但是在函数体中定义了指针变量 p, 并在 p 未明确设定它所指向的变量情况下, 代码 *p=*x 企图将值存入它所指的变量中, 这是初学者经常出现的一种严重的错误。该函数因为有错, 不能实现交换变量值的功能。

对于函数 funb(), 该函数不定义指针形参, 函数体也没有可用的指针间接引用调用环境中的

变量, 所以该函数也不能实现交换变量值的功能。

对于函数 `func()`, 该函数正确定义了指针形参, 函数体也正确通过指针形参间接引用环境中的变量, 并正确定义了自己的局部变量, 该函数能实现交换变量值的功能。

对于函数 `fund()`, 该函数没有定义指针形参, 而函数体将它们作为指针变量使用显然是错误的。

综上所述, 能正确执行交换变量值的函数是 `func()`, 所以答案是 C。

76. 要求函数的功能是在一维数组 `a[]` 中查找 `x` 值。若找到, 则返回所在的下标值; 否则, 返回 0。设数据放在数组元素的 `a[1]~a[n]` 中。以下给出的函数中, 不能正确执行此功能的函数是 ()。

A. `int funa(int *a, int n, int x)`

```
{ *a=x;
  while(a[n] != x)n--;
  return n;
}
```

B. `int funb(int *a, int n, int x)`

```
{ int k;
  for(k=1; k <= n; k++)
    if(a[k]==x)return k;
  return 0;
}
```

C. `int func(int a[], int n, int x)`

```
{ int *k;
  a[0]=x; k=a+n;
  while(*k!=x)k--;
  return k-n;
}
```

D. `int fund(int a[],int n,int x)`

```
{ int k=0;
  do k++;
  while((k<n+1)&&(a[k]!=x));
  if((k<n+1)&&a[k]==x)return k;
  else return 0;
}
```

【答案与解析】本题答案是 C。

在数组中找指定值是经常遇到的计算要求, 有多种编程方法。在这里, 数据预放在数组下标 `1~n` 的元素中, 下标为 0 的元素没有放数据, 程序可以利用这个位置简化查找函数。函数 `funa()` 先将要查找的值放入 `a[0]`, 从数据表的最后一个元素开始逆序向前查找。这样做的好处是循环条件不必担心因数组中原先没有值为 `x` 的元素而一直向前, 访问不是数表的元素, 须插入条件 `n>0`。在 `a[0]` 处放入 `x` 后, 这个条件就不必要了, 循环至少在访问了 `a[0]` 后终止, 并返回 0 值。所以该函数能完成指定的功能。函数 `funb()` 采用常规的办法编写, 循环在 `a[1]` 与 `a[n]` 之间顺序寻找, 一旦找到立即返回找到处的下标, 直至查找循环结束, 查不到指定的值时, 返回 0 值。函数 `func()` 采用与函数 `funa()` 相同的方法, 不过是另引入一个指针变量, 但是该函数 `return` 语句后的表达式有严重的错误, 应返回 `k-a`, 两指针的差是一个整数, 在这里是找到元素的下标。表达式 `k-n` 是指针 `k` 向前移 `n` 个位置的指针值。函数 `fund()` 预置 `k` 为 0, 循环让 `k` 加 1, 并在 `k` 在界内和 `a[k]` 不等于 `x` 的情况下继续循环。循环结束有两种情况, 或 `k` 已不在界内, 或 `k` 在界内且 `a[k]` 等于 `x`。若是后者, 函数返回 `k`, 而若前者, 函数返回 0。该函数也能正确完成查找工作。这样, 不能正确完成查找工作的函数是函数 `func()`。所以答案是 C。

77. 以下程序的输出结果是 ()。

```
#include<stdio.h>
sub1(char a,char b){ char c;c=a;a=b;b=c;}
sub2(char *a,char b){ char c;c=*a;*a=b;b=c;}
sub3(char *a,char*b){ char c;c=*a;*a=*b;*b=c;}
main()
```

```

{ char a,b;
  a='A';b='B';sub3(&a,&b);putchar(a);putchar(b);
  a='A';b='B';sub2(&a,b);putchar(a);putchar(b);
  a='A';b='B';sub1(a,b);putchar(a);putchar(b);
}

```

A. BABBAB

B. ABBBBBA

C. BABABA

D. BAABBA

【答案与解析】本题答案是 A。

在上述程序中, 函数 sub1() 完成两形参值的交换, 与实参提供的变量无关, 这是一个没有意义的函数。函数 sub2() 将第 2 个形参的值置入由第 1 个指针形参所指的变量中, 指针形参所指的变量由调用时的实参提供。函数 sub3() 完成将两个形参所指的变量的值交换的工作。程序调用 sub3(), 使变量 a 和 b 的值交换输出 BA; 调用 sub2(), 使变量 b 的值传送到 a, 输出 BB; 调用 sub1(), 变量 a 和 b 的值不改变, 输出 AB, 所以程序输出 BABBAB。所以答案是 A。

78. 设有定义语句 `int(*ptr)[10];`, 其中的 ptr 是 ()。

A. 10 个指向整型变量的指针

B. 指向 10 个整型变量的函数指针

C. 一个指向具有 10 个元素的一维数组的指针

D. 具有 10 个指针元素的一维数组

【答案与解析】本题答案是 C。

代码 `int(*ptr)[10];` 的分析过程是: 括号内的 ptr 先与字符 * 结合, 字符 * 修饰标识符 ptr, 使它成为一种指针; 接着与后面的一对方括号结合, 表示是指向一维数组的指针; 再有方括号中的 10, 说明这种数组有 10 个元素; 至此, ptr 是指向含 10 个元素的一维数组的指针; 最后, 是最前面的 int, 表示数组元素是 int 类型的。所以, ptr 是一个指向具有 10 个 int 型元素的一维数组的指针。所以答案是 C。另外, 选项 A, 10 个指向整型变量的指针, 就是一个指针数组, 上述代码应写成 `int *ptr[10];`, 即 ptr 是一个有 10 个指向整型变量的指针数组。选项 B, 返回值是指向 10 个整型变量的函数的指针, 上述代码应写成 `int(*(*ptr)())[10];`, 即 ptr 是指针, 指向一种函数, 函数的返回值又是一种指针, 指向 10 个元素的数组, 数组的元素是整型的。下面的代码是这样的函数指针和函数的例子:

```

#include<stdio.h>
int a[][10]={1,2,3,4,5,6,7,8,9,0},{0,1,2,3,4,5,6,7,8,9}};
int(*(*ptr)(int))[10];
int(*f(int n))[10]
{   return a+n;
}
void main()
{   int(*p)[10],*q;
    ptr=f;                               /*让 ptr 指向函数 f*/
    p=(*ptr)(0);q=*p;printf("%d\n",*q);   /*输出 a[0][0]*/
    p=(*ptr)(1);q=*p;printf("%d\n",*q);   /*输出 a[1][0]*/
}

```

在上述代码中, 函数有一个 int 型的形参。选项 D, 其意义与选项 A 相同, 上述代码应写成 `int *ptr[10];`, 即 ptr 是一个有 10 个元素的数组, 数组元素是指向整型变量的指针。

79. 计算平方数时, 不可能引起二义性的宏定义是 ()。

A. #define SQR(x) x*x

B. #define SQR(x) (x)*(x)

C. #define SQR(x) (x*x)

D. #define SQR(x) ((x)*(x))

【答案与解析】本题答案是 D。

宏调用是字符序列的替换，一个宏调用被替换后，替换后的字符序列就与它前后的字符序列一起构成新的源程序。常有这样的情况，因宏定义对宏参数的独立性注意不够，替换后的字符序列会与其前后字符序列结合，产生意想不到的结果；另外，因宏调用本身独立性注意不够，宏展开的内容会被分裂，部分内容会与其前或后的其他部分先结合。如果一个宏定义有可能发生上述情况，它就是一个有可能引起二义性的宏定义。对于 A，若希望描述计算 $u+v$ 的平方，用宏调用 $SQR(u+v)$ ，产生的代码将是 $u+v*u+v$ ，这是因为宏定义对参数的独立性注意不够，产生不是希望的展开代码。对于 B，若希望描述计算 $1.0/x^2$ ，用宏调用 $1.0/SQR(x)$ ，产生的代码将是 $1.0/(x)*(x)$ ，这是因为宏调用本身独立性注意不够而引起的。对于选项 C，与选项 A 的情况一样，宏调用 $SQR(u+v)$ ，不能正确描述 $u+v$ 的平方。对于选项 D，在宏定义中，给参数 x 加上括号能保证参数的独立性，定义给全部内容也加上括号，这也保证了整个宏调用的独立性。所以，选项 D 是计算平方数时，不可能引起二义性的宏定义。所以答案是 D。

80. 执行下列程序代码：

```
#define M(a,b) (a)>(b)?(a):(b)
main()
{ int i=10,j=15; printf("%d\n",10*M(i,j));}
```

输出的结果是 ()。

A. 10

B. 15

C. 100

D. 150

【答案与解析】本题答案是 A。

上述程序输出表达式 $10*M(i, j)$ 的值，该表达式中有宏调用，宏展开后的表达式为 $10*(i > j)?(i):(j)$ ，该表达式是一个条件表达式，它的求值顺序如下：先求 $10*(i > j)$ ，因变量 i 和 j 的值分别为 10 和 15， $10*10$ 大于 15，结果为 1。因为条件表达式中的条件子表达式为 1，以表达式 (i) 的结果为条件表达式的值。 (i) 的值是 10，条件表达式的结果也为 10。最后，程序输出 10。所以答案是 A。

81. 设有定义语句

```
struct { int x; int y; } d[2]={1,3},{2,7};,
```

则

```
printf("%d\n",d[0].y/d[0].x*d[1].x);
```

的输出是 ()。

A. 0

B. 1

C. 3

D. 6

【答案与解析】本题答案是 D。

结构数组 $d[]$ 有两个结构元素，每个结构又由两个整型成分组成，初始化后， $d[0].x=1, d[0].y=3, d[1].x=2$ ，所以表达式 $d[0].y/d[0].x*d[1].x$ 的值是 6。所以答案是 D。

82. 设有定义语句

```
enum term { my,your=4,his,her=his+10};,
```

则

```
printf("%d,%d,%d,%d\n",my,your,his,her);
```

的输出是 ()。

A. 0, 1, 2, 3

B. 0, 4, 0, 10

C. 0, 4, 5, 15

D. 1, 4, 5, 15

【答案与解析】本题答案是 C。

按枚举类型定义的规定，枚举类型 $team$ 的枚举常量对应的整数分别是 $my=0, your=4, his=5,$

her=15。所以语句输出 0,4,5,15。所以答案是 C。

83. 以下对枚举类型名的定义中, 正确的是 ()。

A. enum a={one, two, three};

B. enum a {a1, a2, a3};

C. enum a={'1', '2', '3'};

D. enum a {"one", "two", "three"};

【答案与解析】本题答案是 B。

枚举类型定义的句法是

```
enum 枚举类型名 {枚举类型常量 1, 枚举类型常量 2, ...};
```

其中, 枚举常量都是互不相同的标识符, 所以定义:

```
enum a{a1, a2, a3};
```

是一个正确的定义。任何在枚举类型名之后接上=, 或用字符常量, 或用字符串常量作为枚举常量都是不正确的定义。所以答案是 B。

84. 若有如下定义,

```
typedef union{long x[2];int y[4];char z[8]; } MYTYPE;
```

```
MYTYPE them;
```

则代码

```
printf("%d\n", sizeof(them));
```

的输出是 ()。

A. 32

B. 16

C. 8

D. 24

【答案与解析】本题答案是 C。

共用型变量的成分是覆盖存储, 为共用型变量分配存储时, 以其中需要最多存储字节的成分进行分配。设 long 型数据占 4 个字节、int 型数据占 2 个字节。共用型 MYTYPE 有 3 个成分, 第 1 个成分是有两个 long 型元素的数组, 它需要 8 个字节; 第 2 个成分是有 4 个 int 型元素的数组, 它也需要 8 个字节; 第 3 个成分是有 8 个字符元素的数组, 也是 8 个字节。所以 MYTYPE 类型的共用型变量只需要 8 个字节。所以答案是 C。

85. 设有以下说明和定义:

```
typedef union {long i;int k[5];char c;} DATE;
```

```
struct date{ int cat;DATE cow;double dog;} too;
```

```
DATE max;
```

则下列语句的执行结果是 ()。

```
printf("%d", sizeof(struct date)+sizeof(max));
```

A. 26

B. 30

C. 18

D. 8

【答案与解析】本题答案是 B。

设 long 型数据占 4 个字节, int 型数据占 2 个字节, double 数据占 8 个字节。共用型 DATE 有 3 个成分: long 型成分 i, 有 5 个 int 型元素的数组 k, char 型成分 c。所以 DATE 类型的变量需要 10 个字节。结构类型 date 有 3 个成分: int 型成分 cat, DATE 共用类型 cow, double 型成分 dog。所以结构类型 date 需要 20 个字节。所以表达式 sizeof(struct date)+sizeof(max)的值是 30。所以答案是 B。

86. 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 c[10]={{"John",17}, {"Paul",19}, {"Mary",18}, {"Adam",16}};
```

A. printf("%c", c[3].name);

B. printf("%c", c[3].name[1]);

C. printf("%c", c[2].name[1]);

D. printf("%c", c[2].name[0]);

节,成员z需要10个字节,这样共用型变量u1就需要40个字节,输出共用型变量u1占用的字节数应是40。所以答案是C。

90. 下列数据类型中不属于构造类型的是()。

- A. 数组型 B. 构造型 C. 共用型 D. 枚举型

【答案与解析】本题答案是D。

在C语言中,构造型数据是指由若干成分数据组成,并且成分数据是程序可直接访问的。数组型、结构型和共用型都由若干成员(元素)组成,并且它们的成员都是程序可直接访问的,而枚举型列出一些可能值,限定变量取其中的某个值,枚举型变量是没有成员的,所以枚举型不属于构造类型。所以答案是D。

91. 存储整型数据-7856时,在二进制文件和文本文件中占用的字节数分别是()。

- A. 2, 2 B. 2, 5 C. 5, 5 D. 5, 2

【答案与解析】本题答案是B。

二进制文件中的数据信息与数据在内存中的存储形式完全相同,并且数据之间也不须要另外存储分割符。例如,系统将一个int型数据用2个字节存储,则任何int型数据在二进制文件中只需2个字节。若把一个int型数据存到文本文件中,则须将它翻译成十进制数字的字符序列,表示整数-7856需要5个字符。所以答案是B。

92. 以只写方式打开一个二进制文件,应选择的使用方式是()。

- A. "a+" B. "w+" C. "rb" D. "wb"

【答案与解析】本题答案是D。

程序使用文件可以有多种不同方式。在C系统中,使用方式由文件打开方式指定。首先是文件信息的存储形式,有以内存表示形式存储的二进制文件和以字符形式存储的文本文件。要特别指定为二进制形式存储,在打开方式中要有字符b。其次是文件的读/写方式。其中,只读一个已存在的文件,在打开方式中用字符r;若只写建立一个新文件,或只写重建一个文件,在打开方式中用字符w;向已存在的文件只写追加信息,在打开方式中用字符a;若有读有写方式使用文件,在打开方式中有字符+。特别地,对已存在文件的有读有写,用双字符r+;在文件尾部追加信息的有读有写,用双字符a+;新建或重建文件的有读有写,用双字符w+。由以上说明知,以只写方式打开一个二进制文件,在打开方式中有字符b和有字符w。所以答案是D。

93. 若文件型指针fp所指向的文件的当前位置已经到达文件的末尾,则函数feof(fp)的返回值是()。

- A. 0 B. -1 C. 非零值 D. NULL

【答案与解析】本题答案是C。

当调用fopen()函数时,系统将为实参文件设定一个存放文件读/写控制信息的控制块。该控制块记录有关文件特征、使用方式及当前读/写位置等文件状态,并将该控制块的指针返回作为文件指针变量的值,以后程序通过该指针调用函数时,系统能访问到这个控制块,利用该文件控制块中的控制信息,正确完成文件的读或写。所以文件指针并不真正指向文件当前读/写位置,但通过文件指针,系统能在文件控制块中访问文件的当前读/写位置。若某文件与文件指针fp相关联着,该文件的当前读/写位置在文件的末尾时,函数调用feof(fp)的返回值是一个非零值。所以正确答案是C。

94. 下列语句中, 将 c 定义为文件型指针的是 ()。

- A. FILE c; B. FILE *c; C. file c; D. file *c;

【答案与解析】本题答案是 B。

如上题所述, 文件控制块的类型已由系统命名为 FILE。所以定义文件指针变量是 c, 就得用代码 FILE *c;。所以答案是 B。

95. 标准库函数 fputs(p1, p2)的功能是 ()。

- A. 从 p1 指向的文件中读一个字符串存入 p2 指向的内存
B. 从 p2 指向的文件中读一个字符串存入 p1 指向的内存
C. 从 p1 指向的内存中的一个字符串输出到 p2 指向的文件
D. 从 p2 指向的内存中的一个字符串输出到 p1 指向的文件

【答案与解析】本题答案是 C。

标准函数调用 fputs(p1, p2)完成的功能是将由字符指针 p1 所指的字符串内容输出到与文件指针 p2 相关联的文件中, 即将 p1 所指的内存中的字符串内容复制存入到 p2 所指向的文件中。所以答案是 C。

96. 下列 C 代码中, 正确的是 ()。

- A. #define MYNAME= "ABC" B. #include string.h
C. for(i=0; i<10; i++); D. struct int stu {int name};

【答案与解析】本题答案是 C。

不带参数宏定义的一般定义格式是:

#define 宏名标识符 字符列

选项 A 是关于宏定义的, 在宏的定义形式中, 宏名与字符列之间至少有一个空白符, 并在宏名希望代表的字符序列之间也不能有任何非空格类字符, 若有也被当作是字符序列的一部分。有了宏定义, 以后程序中若要有宏名后的字符序列就可简写成宏名。选项 A 在宏名之后紧接字符=。

选项 B 是关于包含命令的, 在上述包含命令的格式中, 被包含文件应放在一对尖括号之间, 或一对双引号之间, 不可在文件名字符列的首末没有这种符号。

选项 C 是 for 语句, for 语句的循环体可以用空语句, 从语言的句法规则来说, 该语句是正确的, 但通常循环是为了完成某种重复计算操作, 单一地让一个变量循环加 1 到某个值。

选项 D 是想定义结构类型, 定义结构时, struct 后接结构类型名, 不可在结构名之前插入其他类型符。所以答案是 C。

97. 下列 C 代码中, 错误的是 ()。

- A. #define PI 3.14159 B. #include "stdio.h";
C. if(2); D. for(;;)if(1)break;

【答案与解析】本题答案是 B。

选项 A 是正确的宏定义。选项 B 的文件包含预处理命令行不可以有多余的分号字符。选项 C 是 if 语句, 虽没有任何意义, 但就句法来说是正确的。选项 D 是 for 语句, 其成分语句是 if 语句, 从 for 语句来说, 是一个无穷循环的控制结构, 但它的成分语句因条件成立, 执行 break 语句, 立即退出循环, 该代码虽然也是一条没有任何实际意义的语句, 但从句法规则来说, 还是正确的。所以答案是 B。

98. 设有以下 C 代码, 则执行语句 $z=2*(N+Y(5+1));$ 后, z 的值为 ()。

```
#define N 3
#define Y(n) ((N+1)*n)
```

A. 出错 B. 42 C. 48 D. 54

【答案与解析】本题答案是 C。

在新的宏定义中, 可以使用前面已定义的宏名。另外, 宏调用是这样被替换展开的, 分别用宏调用中的实参字符序列替换宏定义字符序列中对应所有出现的形式参数, 而宏定义字符序列中的不是形式参数的其他字符则保留。这样形成的字符序列, 即为宏调用的展开替换结果。所以代码 $z=2*(N+Y(5+1));$ 被替换成:

```
z=2*(3+((3+1)*5+1));
```

执行该代码, 将使 z 的值为 48。所以答案是 C。

99. 以下程序中的 for 循环执行的次数是 ()。

```
#include<stdio.h>
#define N 2
#define M N+1
#define NUM (M+1)*M/2
main()
{ int i,n=0;
  for(i=1;i<=NUM;i++){ n++;printf("%d",n); }
  printf("\n");
}
```

A. 5 B. 6 C. 8 D. 9

【答案与解析】本题答案是 C。

代码 NUM 被替换成 $(2+1+1)*2+1/2$, 其值是 8, 程序中的 for 循环执行的次数是 8 次。所以答案是 C。

100. 设有如下结构类型:

```
typedef struct ele
{ int no;
  char name[20];
} Ele;
```

另有如下变量定义:

```
Ele a[20],x;
```

以下语句希望实现向与文件指针变量 fp 关联的文件输出一个结构信息, 错误的代码是 ()。

A. $fwrite(a, sizeof(Ele), 1, fp);$ B. $fwrite(&a[2], sizeof(Ele), 1, fp);$
C. $fwrite(&x, sizeof(Ele), 1, fp);$ D. $fwrite(x, sizeof(Ele), 1, fp);$

【答案与解析】本题答案是 D。

函数 $fwrite(char *buf, int size, int count, FILE *fp)$ 实现将从地址 buf 开始存储的, 每块信息有 $size$ 个字节, 连续的 $count$ 块信息复制存入与文件指针 fp 关联的文件中。这里要求调用时的第 1 个实参是指针。在选项 A 中因 a 是数组, 单一的 a 是数组 $a[]$ 的首元素的指针, 所以是正确的。选项 B 的 $\&a[2]$ 是数组元素 $a[2]$ 的指针, 也是正确的。C 中的 $\&x$ 是取结构变量 x 的指针, 所以也是正确的。选项 D 中的 x 应表示对结构变量 x 的值的引用, 不是指针, 所以是错误的。所以答案是 D。

二、填空题

1. 用高级语言编写的程序称为 _____, 它可以通过 _____ 翻译一句执行一句的方式执

行,也可以通过_____一次翻译产生_____,然后执行。

【答案与解析】用高级语言编写的程序称为源程序,源程序不能在计算机上直接运行,运行源程序有两种方式:一种是通过解释程序,对源程序逐句解释执行;另一种是先让编译程序将源程序一次翻译产生目标程序(目标程序是计算机可直接执行的机器语言程序,它是一种二进制代码程序),然后执行目标程序。

2. 转义字符是由_____开始,后接单个字符或若干字符组成。

【答案与解析】在C语言中,转义字符由反斜杠字符开始,后接单个字符或若干个字符组成。

3. C语言词类主要分为_____,_____、_____和_____等。

【答案与解析】语言的基本词汇是指直接由字符序列组成,有确定意义的最基本单位,所以C语言的词汇有:字面形式常量、特殊符号(主要是运算符)、保留字和标识符等4类,而表达式、函数调用等是更高级的语言成分,如表达式中还可分运算分量和运算符等;函数调用也是一种表达式,它有函数名标识符、圆括号和实在参数表组成。利用基本词汇,按照给定的C语言的句法规则,就可命名程序对象,描述表达式计算、构造语句、函数,直至整个程序。

4. C程序是由函数构成的。其中有并且只能有_____个主函数。C语言程序的执行总是由_____开始,并且在_____中结束(如果别的函数未调用终止程序执行的库函数)。

【答案与解析】C程序是由若干函数构成的。其中有并且只能有1个主函数(用main命名)。C程序的执行总是由主函数开始执行,通常在主函数中结束(如果没有在别的函数中调用终止程序执行的库函数的话)。

5. C语言程序的注释可以出现在程序中的任何地方,一个注释以双符号_____作为开始标记,以双符号_____作为结束标记。

【答案与解析】C语言程序的注释可以出现在程序基本单词之间的任何地方,C语言程序的注释以双符号 /* 作为开始标记,并以双符号 */ 作为结束标记。

6. 在内存中,存储字符串"X"要占用_____个字节,存储字符'x'要占用_____个字节。

【答案与解析】计算机存储一个字符用1个字节,存储字符串时,每个字符也占用1个字节,另外在字符串的有效字符之后存储1个字符串的结束标记符。所以存储字符串"X"要占用2个字节,存储字符'x'只要1个字节。

7. 在C程序中,判断逻辑值时,用_____表示逻辑值“真”,又用_____表示逻辑值“假”。在求逻辑值时,用_____表示逻辑表达式值为“真”,又用_____表示逻辑表达式值为“假”。

【答案与解析】在C程序中,判断逻辑值时,用非0表示真,用数0表示假。逻辑表达式计算时,逻辑表达式值为真是用1表示的,若逻辑表达式的值为假,则用0表示。

8. 定义符号常量的一般形式是_____。

【答案与解析】定义符号常量用预处理命令的宏定义,其定义的一般形式是:

#define 符号常量名 常量

9. 在函数内,说明变量时,若省略存储类别符,系统默认其为_____,该存储类别的标记符为:_____。

【答案与解析】在函数内,说明变量时,若省略存储类别符,系统就默认其为自动存储类别,该存储类别用 auto 标记。

10. 无符号基本整型的数据类型符为 _____, 双精度实型数据类型符为 _____, 字符型数据类型符为 _____。

【答案与解析】无符号的基本整型的类型符为 unsigned int, 或简写成 unsigned。双精度实型数据类型符为 double, 字符型数据类型符为 char。

11. 定义指定存储类别和数据类型的变量时, 较完整的一般形式是 _____。

【答案与解析】定义时, 要指定变量的存储类别和数据类型, 其一般形式是:

存储类别 数据类型符 变量名 1, 变量名 2, ...;

12. 设有下列运算符: $<<$ 、 $+$ 、 $++$ 、 $\&\&$ 、 $<=$, 其中优先级最高的是 _____, 优先级最低的是 _____。

【答案与解析】对运算符 $<<$ 、 $+$ 、 $++$ 、 $\&\&$ 、 $<=$, 按它们的优先级自高到低的顺序排列为:

$++$ 、 $+$ 、 $<<$ 、 $<=$ 、 $\&\&$

所以, 优先级最高的是 $++$, 优先级最低的是 $\&\&$ 。

13. 设二进制数 A 是 00101101, 若想通过异或运算 $A \wedge B$ 使 A 的高 4 位取反, 低 4 位不变, 则二进制数 B 应是 _____。

【答案与解析】按位异或运算的一个重要应用是让某个整型变量的二进制位串信息的某些位信息反向, 0 变成 1, 而 1 变成 0, 这只要设计这样一个位串信息, 让要变反的位为 1, 不要改变的位为 0, 用这个位串信息与整型变量按位异或就能得到希望的结果。要使字节的高 4 位取反, 低 4 位不变, 则需要的位串信息是 11110000, 写成八进制数是 0360, 写成十六进制数为 0xF0。

14. 设 $a=3$, $b=2$, $c=1$, 则 $a>b$ 的值为 _____, $a>b>c$ 的值为 _____。

【答案与解析】计算 $a>b$, 因 a 的值为 3, b 的值是 2, 条件 $a>b$ 为真, 其值为 1。表达式 $a>b>c$ 的求值顺序是 $(a>b)>c$ 。先计算 $a>b$, 结果为 1, 接着计算 $1>c$, 因 c 的值为 1, 条件 $1>c$ 为假, 结果为 0。

15. 若已知 $a=10$, $b=20$, 则表达式 $!a<b$ 的值为 _____。

【答案与解析】计算表达式 $!a<b$, 先计算 $!a$, 因 a 的值为 10, $!a$ 的值为 0。关系表达式 $0<20$ 为真, 所以表达式 $!a<b$ 的值为 1。

16. 设 x 和 y 均为 int 型变量, 且 $x=1$, $y=2$, 则表达式 $1.0+x/y$ 的值为 _____。

【答案与解析】计算表达式 $1.0+x/y$, 先求 x/y , 因 x 和 y 是整型变量, 其中, 除运算是整除, $1/2$ 的结果为 0, 接着计算 $1.0+0$, 计算时, 先将右分量转换成 0.0, 最后得到结果 1.0。

17. 设整型变量 x 、 y 、 z 均为 5:

(1) 执行 $x-=y-z$ 后, $x=$ _____,

(2) 执行 $x\%=y+z$ 后, $x=$ _____,

(3) 执行 $x=(y>z)?x+2:x-2, 3, 2$ 后, $x=$ _____。

【答案与解析】在变量 x 、 y 、 z 的值均为 5 的情况下, 计算各表达式。由于表达式 $x-=y-z$ 等价于表达式 $x=x-(y-z)$, 所以计算后 x 的值为 5。表达式 $x\%=y+z$ 等价于表达式 $x=x\%(y+z)$, 所以计算后 x 的值也为 5。表达式 $x=(y>z)?x+2:x-2, 3, 2$ 的计算过程用圆括号描述其计算顺序是:

$((x=(y>z)?x+2:x-2), 3), 2$

即这是一个逗号运算表达式, 由一个赋值表达式和两个数值构成, 逗号表达式要求顺序求各子表达式的值。表达式 $x=(y>z)?x+2:x-2$ 的计算是先求赋值号右边的条件表达式, 因条件 $(y>z)$ 为假, 求出 $x-2$ 的值为 3, 将该值赋给变量 x , 使 x 的值为 3。

18. 能表述“ $20 < x < 30$ 或 $x < -100$ ”的 C 语言表达式是_____。

【答案与解析】首先表述 $20 < x < 30$ 的 C 表达式可写成 $20 < x \ \&\& \ x < 30$ 。所以表述“ $20 < x < 30$ 或 $x < -100$ ”的 C 表达式为 $20 < x \ \&\& \ x < 30 \ || \ x < -100$ 。

19. 写出数学式 $\frac{x}{yz}$ 的 C 语言表达式_____。

【答案与解析】数学式子表述的是 x 除以 y 乘 z 的积, 写成 C 表达式可以写为 $x/(y*z)$, 或可等价地写成 $x/y/z$ 。

20. C 语言中运算结果可以当成逻辑值使用的表达式有: _____、_____、_____、_____、_____。

【答案与解析】在 C 语言中, 逻辑判断以非 0 值为真, 以 0 值为假。所以几乎所有的表达式的运算结果都可以当成逻辑值使用, 如算术表达式、关系表达式、逻辑表达式、赋值表达式、逗号表达式、条件表达式等。

21. 结构化程序设计规定的 3 种基本结构是_____, 选择结构和_____。

【答案与解析】结构化程序设计的 3 种基本控制结构是顺序结构、选择结构和循环结构。

22. 若有定义语句 $\text{int } a=25, b=14, c=19;$, 以下语句的执行结果是_____。

```
if (a++<=25&&b--<=2&&c++)
    printf("***a=%d,b=%d,c=%d\n", a, b, c);
else printf("###a=%d,b=%d,c=%d\n", a, b, c);
```

【答案与解析】问题所给的 if 语句中, 条件 $a++<=25 \ \&\& \ b--<=2 \ \&\& \ c++$ 是先求逻辑与的第 1 个运算分量, 它是一个关系式, 关系成立, 接着判定第 2 个逻辑与运算分量, 又是一个关系式, 由于变量 b 的值是 14, b 不小于等于 2, 运算分量的关系式不成立, 导致 if 语句的条件为假, 执行 else 部分。在求 if 语句的条件时, 计算了两个逻辑与分量, 使变量 a 的值增了 1, 变量 b 的值被减了 1。所以输出内容是: ###a=26,b=13,c=19。

23. 以下两条 if 语句可合并成一条 if 语句为_____。

```
if (a<=b) x=1;
else y=2;
if (a>b) printf("****y=%d\n", y);
else printf("####x=%d\n", x);
```

【答案与解析】在以上两条 if 语句中, 两个条件刚巧相反。若将前一个 if 语句的第 1 个成分语句与第 2 个 if 语句的第 2 个成分语句合并; 第 1 个 if 语句的第 2 个成分语句与第 2 个 if 语句的第 1 个成分语句合并, 写成一条 if 语句如下:

```
if (a<=b) {x=1; printf("####x=%d\n", x); }
else {y=2; printf("****y=%d\n", y); }
```

24. 设 i, j, k 均为 int 型变量, 则执行完下面的 for 语句后, k 的值为_____。

```
for (i=0, j=10; i<=j; i++, j--) k=i+j;
```

【答案与解析】该 for 语句以 i 为 0、 j 为 10 初始化, 循环条件是 $i \leq j$, 每次循环后 i 增 1、 j 减 1, 循环体是将 i 与 j 的和赋给 k 。这样, 变量 k 将保存的是最后一次赋给它的值。第 1 次循环后 i 为 1、 j 为 9, 第 2 次循环后 i 为 2、 j 为 8, ……第 5 次循环后 i 为 5、 j 为 5, 继续第 6 次循环, 将 i 与 j 的和 10 存于 k 后, i 为 6、 j 为 4, 结束循环。所以循环执行后 k 为 10。

实际上, 每次循环使 i 增 1, j 减 1, $i+j$ 的值在循环过程中一直没有改变。

25. 下列程序的功能是输入一个整数, 判断是否为素数, 若为素数输出 1, 否则输出 0, 请为程序填空。

```
#include<stdio.h>
void main()
{ int i,x,y=1;
  scanf("%d",&x);
  for(i=2;i<=x/2;i++)
    if _____ { y=0;break;}
  printf("%d\n",y);
}
```

【答案与解析】为判断 x 是否是素数, 预置变量 y 的值为 1 (x 可能是素数), 然后用循环, 用 $2 \sim x/2$ 的整数 i 测试对 x 的整除性, 若能被其中的某个整数整除, 则 x 是合数, 置 y 的值为 0 (x 不是素数), 结束测试循环。若都不能整除, 则 x 为素数。 i 能否整除 x , 可用求余运算 $x\%i$ 是否等于 0 来判定。所以程序中的空框处可填 $x\%i==0$ 。测试 x 为素数的程序代码可写成:

```
y=1;
for(i=2;i<=x/2;i++)
  if(x%i==0){ y=0;break; }
```

26. 根据变量定义 `static int b[5], a[][3]={1,2,3,4,5,6};`, $b[4]=$ _____, $a[1][2]=$ _____。

【答案与解析】系统规定, 静态变量定义时, 若未初始化, 系统自动给定二进制代码全 0 的值。所以静态数组 $b[]$ 的元素全为 0, 当然包括 $b[4]$ 也为 0。静态数组 $a[][]$ 是二维数组, 每行有 3 个元素, 在初始化中给出了 6 个初值, 所以数组 $a[][]$ 有 2 行。由于数组的元素按行顺序存储, 并按行顺序初始化。前 3 个值是第 1 行, 后 3 个值是第 2 行。 $a[1][2]$ 是第 2 行第 3 列元素, 也是 $a[][]$ 的最后一个元素, 所以其值是 6。

27. 设有代码 `static int a[3][4]={{1},{2},{3}};`, 则 $a[1][1]$ 值为 _____, $a[2][1]$ 的值为 _____。

【答案与解析】给二维数组初始化时, 可按行给出初值, 并且每行可以只给前面部分的元素赋初值。系统规定, 对那些未给出初值的部分元素, 系统自动置二进制代码全 0 的值。由于定义语句只给 $a[0][0]$ 给定初值 1、 $a[1][0]$ 给定初值 2、 $a[2][0]$ 给定初值 3, 这样, 其他元素都为 0 值。所以, $a[1][1]=$ 0, $a[2][1]=$ 0。

28. 若在程序中用到 `putchar()` 函数时, 应在程序开头写上包含文件预处理命令 _____, 若在程序中用到 `strlen()` 函数时, 应在程序开头写上包含文件预处理命令 _____。

【答案与解析】`putchar()` 是系统提供的头文件 `stdio.h` 中定义的宏, 所以程序要使用它, 必须写上包含文件预处理命令: `#include<stdio.h>`。函数 `strlen()` 是系统提供的字符串处理函数之一, 字符串处理函数的函数原型说明和它们用到的有关常量、数据结构等, 在系统提供的头文件 `string.h` 中定义, 所以使用该函数的程序应在程序开头写上包含文件预处理命令 `#include<string.h>`。

29. 下面程序的功能是输出数组 $s[]$ 中最大元素的下标, 试完成以下程序:

```
#include<stdio.h>
void main()
{ int k,p,s[]={1,-9,7,2,-10,3};
  for(p=0,k=p;p<6;p++)
    if(s[p]>s[k]) _____
  printf("%d\n",k);
}
```

【答案与解析】为要寻找数组中的最大元素的下标,须要先预设一个临时最大元素的下标,并顺序逐一考察数组的元素,当发现当前元素比临时最大元素更大时,就用当前元素的下标更新临时最大元素下标,直至考察了数组的全部元素后,这临时最大元素下标就是数组的最大元素下标。通常,预设的最大元素下标是数组的首元素下标,考察是从首元素开始顺序向后继元素考察。程序中,存储临时最大元素下标的变量是 k ,变量 p 是顺序考察的循环控制变量。当发现当前元素 $s[p]$ 比临时最大元素 $s[k]$ 更大时,应该用 p 更新 k 。所以在空框处应填入代码 $k=p$ 。

30. 下面程序的功能是将一个字符串 str 的内容颠倒过来,试完成以下程序:

```
#include<stdio.h>
#include<string.h>
void main()
{ int i,j,_____; char str[]={"1234567"};
  for(i=0,j=strlen(str)_____; i<j;i++,j--)
  { k=str[i];str[i]=str[j];str[j]=k;}
  printf("%s\n",str);
}
```

【答案与解析】颠倒一个字符串中的字符,就是首尾对应的元素两两交换。简单地可用两个下标变量 i 和 j , i 是前端元素的下标, j 是后端元素的下标,交换以这两个变量值为下标的元素 $str[i]$ 和 $str[j]$ 。开始时, i 的值为 0, j 的值为字符串末元素的下标(字符串长度减 1)。字符串末元素的下标是它的长度减 1,所以在第二个空框处应填入 -1 。每次交换后, i 增 1, j 减 1,继续交换的条件是 $str[i]$ 位于 $str[j]$ 的前面,即 $i < j$ 。程序为了交换 $str[i]$ 和 $str[j]$,使用了变量 k ,该变量应在程序的变量定义部分中一起定义,所以在第一个空框处应填入 k 。

31. 以下程序可把输入的十进制数以十六进制数的形式输出,试完成以下程序:

```
#include<stdio.h>
void main()
{ char b[17]={"0123456789ABCDEF"};
  int c[64],d,i=0,base=16;
  long n;
  printf("Enter a number:\n"); scanf("%ld",&n);
  do { c[i]=_____; i++; n=n/base; }
  while(n!=0);
  printf("%d 的 %d 进制值为\n",n,base);
  for(--i;i>=0;--i)
  { d=c[i]; printf("%c",b_____) };
  printf("\n");
}
```

【答案与解析】程序中,字符数组 $b[]$ 存储十六进制的 16 个数字符,整型数组 $c[]$ 存储译出的十六进制数的各位数值。从整数 n 译出它的各位十六进制数值,须采用除 16 取余的方法,即求 n 除以 16 的余数,得到它的十六进制的个位数,接着将 n 除以 16。在 $n \neq 0$ 的情况下循环,能顺序求出 n 的十六进制的个位数、十位数、百位数等。程序中变量 $base$ 已预置 16,所以在第 1 个空框处可填代码 $n \% base$ 或 $n \% 16$ 。当 n 的十六进制数的个位、十位、百位等依次存放于数组 $c[]$ 中后,就从最高位至最低位,参照数组 $c[i]$ 的内容 d ,取十六进制数字符表中的字符 $b[d]$ 输出。所以在第 2 个空框处可填入代码 $[d]$ 。

32. 单目运算符“*”称为_____运算符, 单目运算符“&”称为_____运算符。

【答案与解析】单目运算符“*”称为取内容运算符, 单目运算符“&”称为取地址运算符。

33. 若两个指针变量指向同一个数组的元素, 可以进行减法运算和_____运算。

【答案与解析】若两个指针变量指向同一个数组的元素, 可以进行减法运算求它们所指元素相差多少个元素。进行关系运算, 判定它们所指元素在数组中的前后或是否指向同一个元素等。

34. 设 `int a[10], *p=a;`, 则对 `a[3]` 的引用可以是 `p[_____]` 和 `*(p_____)`。

【答案与解析】若指针 `p` 指向数组 `a[]` 的首元素, 则引用 `a[3]` 用指针 `p` 可以写成 `p[3]`, 和 `*(p+3)`。

35. 若 `d` 是已定义的双精度变量, 再定义一个指向 `d` 的指针变量 `p` 的代码是_____。

【答案与解析】若 `d` 是一个双精度变量, 定义能指向它的指针变量 `p` 可以用以下代码 `double *p=&d`。

36. 单目运算符 `&` 后跟变量名, 表示求该变量的_____; 单目运算符 `*` 后跟指针变量名, 表示该指针变量_____; `&` 后跟的是指针变量名, 表示该指针变量_____。

【答案与解析】单目运算符 `&` 是取地址运算符, `&` 后跟变量名, 表示求该变量的地址。单目运算符 `*` 是取内容运算符, `*` 后跟指针变量名, 表示该指针变量所指变量。若 `&` 后跟的是指针变量名, 表示该指针变量的地址。若知道指针变量的地址, 欲通过指针变量的地址引用指针变量所指变量的内容, 须要连续两次取内容运算。

37. 设有 `char *a="ABCD"`, 则 `printf("%s",a)` 的输出是_____; 而 `printf("%c",*a)` 的输出是_____。

【答案与解析】若给字符指针变量 `a` 赋一个常量字符串 `"ABCD"`, 实际上是给 `a` 赋指向字符串常量首字符 `'A'` 的指针。程序通过 `a` 能访问字符串中的各字符。例如, 能用代码 `printf("%s",a)` 输出这个字符串常量 `"ABCD"` 的字符列 ABCD, 用代码 `printf("%c",*a)` 输出 `a` 所指的字符 A。

38. 设有以下定义和语句, 则 `*(p+2)+1` 的值为_____。

```
int a[3][2]={10,20,30,40,50,60}, (*p)[2];
p=a;
```

【答案与解析】上述代码定义 3 行 2 列的二维数组 `a[]`, 定义指向两个元素的一维数组指针 `p`, 并让 `p` 指向二维数组 `a[]` 的首行, 则代码 `*(p+2)+1` 中的 `p+2` 指向二维数组 `a[]` 的第 3 行 `a[2]`, `*(p+2)` 指向 `a[2][0]`, `*(p+2)+1` 指向 `a[2][1]`, `*(p+2)+1` 是引用 `a[2][1]`, 其值是 60。

39. 以下程序的功能是从键盘上输入若干个字符 (以回车符作为结束) 组成一个字符串存入一个字符数组, 然后输出该字符数组中的字符串, 试完成以下程序:

```
#include<stdio.h>
#include<ctype.h>
void main()
{ char str[81],*sptr;
  int i;
  for(i=0;i<80;i++)
  { str[i]=getchar();if(str[i]=='\n')break; }
  str[i]=_____; sptr=str;
  while(*sptr)putchar(*sptr_____);
}
```

【答案与解析】从键盘输入字符序列, 通过循环逐一输入字符, 当发现输入的字符是换行符时, 结束循环。为了使输入的字符序列变成字符串, 必须在输入字符序列之后, 原存储换行符的元素

改为存储字符串的结束标记符, 须用代码 `str[i]='\0'`, 所以在第 1 个空框填入代码 `'\0'`。当要将存于字符数组 `str[]` 中的字符串通过循环逐一输出字符串的字符时, 可以用字符指针 `sptr`, 让 `sptr` 遍历整个字符串, 每次循环时, 输出 `sptr` 所指字符, 并让 `sptr` 加 1, 即可用代码 `*sptr++` 实现, 所以在第 2 个空框填入代码 `++`。

40. 静态存储类别的内部变量的作用域是_____。

【答案与解析】静态存储类别的主要特性是永久性和专用性。静态局部变量提供永久存储, 但因它是局部变量, 也得遵守局部变量的作用域规则: 定义它的函数或复合语句。

41. 函数中的形参和调用时的实参都是数组名时, 传递方式为_____, 都是变量时, 传递方式为_____。

【答案与解析】在 C 语言中, 实参向形参传递方式只有值传递, 但传递的内容可能是非指针类型的值, 如基本类型值, 或结构类型值, 也可以是指针类型值。通常传递指针类型值特称为传地址方式。其中, 实参向指针形参传递数组名, 就是一种传地址方式。若形参不是指针类型的, 形参是一般的变量, 实参可以是变量或表达式, 就是值传递方式。

42. 函数的形式参数的作用域为_____, 全局的外部变量和函数体内定义的局部变量重名时, _____变量优先。

【答案与解析】函数将它的形参当做它的局部变量, 函数形参的作用域为定义它的函数。当函数的形参或函数内的局部变量与函数外的全局变量同名时, 函数的局部变量优先。

43. 若自定义函数要求返回一个值, 则应在该函数体中有一条_____语句; 若自定义函数要求不返回一个值, 则应在该函数说明时加一个类型符_____。

【答案与解析】若自定义的函数有返回值, 则函数返回时应执行带表达式的 return 语句返回, 该表达式的值将作为函数调用的结果。为了强调函数是一个不返回值的函数, 应在该函数说明时, 加上一个 `void` 类型说明符。

44. 若给 `fun()` 函数的形参 `s` 传送字符串: `"U U 6354abcc"` (其中 `U` 表示空格字符), 则函数的返回值是_____。

```
#include<ctype.h>
long fun(char s[])
{ long n;int sign;
  for(;isspace(*s);s++);
  sign=(*s=='-')?-1:1;
  if(*s=='+'||*s=='-')s++;
  for(n=0;isdigit(*s);s++)n=10*n+(*s-'0');
  return sign*n;
}
```

【答案与解析】函数 `fun()` 的第 1 个 `for` 循环跳过字符串的前导空白符, 接着分析第 1 个非空白符是否是负号, 若是置变量 `sign` 为 -1; 否则, 置变量 `sign` 为 1。接着的 `if` 语句在第 1 个非空白字符是负号或正号时, 跳过该字符, 以后的 `for` 循环将后面的数字当作十进制数的各位数字译出一个长整数 `n`。遇到字符 `a` 时, 因它是非数字字符, 结束循环。最后, 函数返回 `sign` 与 `n` 的乘积。所以若调用函数时提供的实参是 `"U U 6354abc"`, 函数返回值是 6354。

45. 下面函数要求计算两个整数 `x`、`y` 之和, 并通过形参 `z` 传回该和值, 试完成以下函数:

```
void add(int x, int y, _____z)
```

```
{ _____=x+y; return; }
```

【答案与解析】函数希望通过形参 z 带回结果, 形参 z 必须是指针类型的, 由于返回整型结果, 所以 z 的说明是 `int *z`。函数返回前必须先把结果通过形参 z 间接引用 (写成 `*z`) 传回。

46. 以下函数用来在数组 `w[]` 中插入 `x`。`n` 所指向的存储单元中存放数组 `w[]` 中的字符个数。数组 `w[]` 中的字符已按从小到大的顺序排列, 插入后数组 `w[]` 中的字符仍有序, 试完成以下函数:

```
void fun(char *w, char x, int *n)
{ int i, p;
  p=0;
  w[*n]=x;
  while(x>w[p])p++;
  for(i=*n;i>p;i--)w[i]=_____;
  w[p]=x;
  ++*n;
}
```

【答案与解析】用数组实现在有序序列中插入内容, 必须先寻找插入位置, 插入位置找到后, 须将插入位置之后的元素向后移动一个位置, 留出插入点, 然后将元素插入, 并增加序列的元素个数。函数为了简化寻找插入位置循环的循环条件, 预先在序列的最后置入插入值。插入位置之后的元素向后移动一个位置必须从最后一个元素开始往前逆序移动, 是将前一个元素向后移, 所以在空框处应填入 `w[i-1]`。

47. 函数 `myCmp()` 的功能是比较字符串 `s` 和 `t` 的大小, 当 `s` 等于 `t` 时返回 0, 否则, 返回 `s` 和 `t` 的第 1 个不同字符的 ASCII 码差值, 即 `s > t` 时返回正值, `s < t` 时返回负值, 试完成以下函数:

```
int myCmp(char *s, char *t)
{ while(*s==*t)
  { if(*s=='\0')return 0;
    ++s; ++t;
  }
  return _____;
}
```

【答案与解析】两字符串大小比较必须从它们的首字符开始, 在对应字符相等的情况下循环, 直至不相等结束。相等时, 若字符串已到了字符串的结束标记符, 则两字符串相同, 函数返回 0 值; 如果还有后继字符, 则准备比较下一对字符。对应字符不相同, 循环结束。循环结束时, 就以两个当前字符的差返回。所以在空框处应填入 `*s-*t`, 保证在 `s > t` 时返回正值, `s < t` 时返回负值。

48. 运算符 “.” 称为 _____, 运算符 “->” 称为 _____。

【答案与解析】引用结构变量或共用型变量的成分用 成员运算符 “.”, 若某指针指向一个结构变量或共用型变量, 通过指针引用它所指变量的成分用 指向成员运算符 “->”。

49. 设有定义 `struct {int a; float b; char c;} abc, *p_abc=&abc;`, 则对结构体成员 `a` 的引用方法可以是 `abc._____a` 和 `p_abc._____a`。

【答案与解析】若有结构变量 `abc` 有成分 `a`, 并有指针 `p_abc` 指向结构变量 `abc`, 则引用变量 `abc` 成分 `a` 的标记形式有 `abc._____a` 和 `p_abc->a`。

50. 若有以下说明和定义语句, 则变量 `w` 在内存中所占的字节数是 _____。

```
union aa {float x; float y; char c[6];};
```



```
struct st{ union aa v;float w[5];double ave; } w;
```

【答案与解析】联合类型 aa 有 3 个成分，一个是 float 型成分 x，一个是 float 型成分 y，再一个是有 6 个元素的字符数组。若 float 型数据占 4 个字节，则类型为 aa 的数据要占用 6 个字节。结构变量 w 有 3 个成分，一个是共用型成分 aa，要占用 6 个字节；一个是有 5 个 float 型元素的数组 w，要占用 20 个字节；一个是 double 型的 ave。若 double 型数据要 8 个字节，则结构变量 w 要占用 34 个字节。

51. FILE *p 的作用是定义一个_____，其中，FILE 是在_____头文件中定义的。

【答案与解析】代码 FILE *p 的作用是定义一个文件指针变量 p，其中，FILE 是在标准输入输出头文件 `stdio.h` 中定义的。

52. 在对文件进行操作的过程中，若要求文件的现行位置回到文件的开头，应当调用的函数是_____函数。

【答案与解析】若要让文件从某个当前位置回到文件的开头，可调用库函数 `rewind()`。

53. 下面的程序用来统计文件中字符的个数，试完成以下程序：

```
#include<stdio.h>
void main()
{ FILE *fp;
  long num=0;
  if((fp=fopen("fname.dat", "r"))==NULL)
  { printf("Can't open file!\n"); exit(0); }
  while_____ { fgetc(fp); num++; }
  printf("num=%ld\n", num);
  fclose(fp);
}
```

【答案与解析】若要统计文件中的字符个数，可以设计一个计数器，并预置该计数器为 0 值，并打开文件。在正确打开的情况下，接着是一个循环。循环在文件未结束的情况下，反复读入字符，并增加字符计数器，所以循环条件可写成：`!feof(fp)`。要统计文件的长度，最好用二进制方式打开，通过读入的字节数求出文件的长。这是因为正文文件方式打开时，文件中的双字符（回车字符和换行字符）读入后将被当作一个字符存储，二进制方式不进行这样的转换。

54. 设有定义 `#define F(N) 2*N`，则表达式 `F(2+3)` 的值是_____。

【答案与解析】`F(N)` 被定义成 `2*N`，表达式 `F(2+3)` 被替换成 `2*2+3`，其值是 7。

55. 设有下列函数定义和变量定义的代码：

```
void m(int c,char *v[ ])
{ while(--c>0)printf("%s",v[c]);
  printf("\n");
}
char *s[]={"p","123","AAA","BBB"};
```

并假定调用函数 `m()` 的代码为 `m(4,s)`，则函数的输出结果是_____。

【答案与解析】由于循环条件是 `--c>0`，是先减 1 后判断大于 0，循环共进行 3 次，顺序输出 `v[3]`，`v[2]` 和 `v[1]`，即输出 `BBBAAA123`。

56. 以下分别是算术、位、关系和逻辑运算符：`+`，`&`，`<`，`!`。其优先级最低的是_____。

【答案与解析】C 语言的运算符十分丰富, 并且运算符的优先级也很多。各运算符的优先级大致是按运算符的意义分类的, 优先级自高到低分别是:

圆括号、数组元素运算符[]、结构或共用体成员运算符.和->

单目运算符 (如否定, 求反、自增自减、取地址、取内容、取正、取负、数据长度)

算术运算符 (先乘、除、求余, 后加、减)

移位运算符

关系运算符 (先大小, 后相等和不等)

双目位运算符 (依次为按位与、按位加、按位或)

双目逻辑运算符 (依次为逻辑与、逻辑或)

三目条件运算符

赋值运算符 (先简单赋值, 后自反赋值)

逗号运算符

本题给出的运算符+、&、<、! 分别是算术运算符的加、双目逻辑运算符的逻辑与、关系运算符的小于和单目运算符的逻辑否定, 在这 4 种运算符中, 优先级最低的是双目逻辑运算符的逻辑与&。所以答案是&。

57. 设 `int x=4;` 表达式

`x+=x*=x+1`

的值是_____。

【答案与解析】赋值运算结合性是自右至左的, 上述表达式的计算顺序可用圆括号等价表示成
(`x+=(x*=(x+1))`)

先计算(`x+1`)得 5; 接着求赋值表达式(`x*=5`), 它等价于表达式 `x=x*(5)`, 该表达式的值为 20, 并使 `x` 的值为 20; 最后计算赋值表达式(`x+=20`), 该表达式的值为 40, 并使 `x` 的值也为 40。所以答案是 40。

58. 设 `int a, b=10;`, 执行 `a=b<<2+1;`后 `a` 的值是_____。

【答案与解析】用圆括号表示上述表达式的计算顺序, 可写成

`a=(b<<(2+1))`

即将 `b` 的值左移 3 位后存于变量 `a`。一般地, 在左移没有使数值二进制位丢失的情况下, 每左移一位, 相当于乘 2, 这样, 10 左移 3 位后的值是 80, 上述表达式使 `a` 的值为 80。所以答案是 80。

59. 设 `short k=-1;` 执行 `printf("%o", k);`后输出是_____。

【答案与解析】设 `short` 型整数占用 2 个字节, 因计算机的整数用补码形式表示, 整数-1 的内部表示是 16 位全 1, 用八进制数表示 16 位全为 1 的数据, 应写成 0177777。将这个数用八进制形式输出, 即为 177777。所以答案是 177777。

60. 设 `int j, k;`, 执行 `for(j=k=0; j<11; k+=j++);`后 `k` 的值是_____。

【答案与解析】上述 `for` 语句功能的直观叙述为: 变量 `j` 和 `k` 均以 0 为初值, 在表达式 `j<11` 不为 0 的情况下循环, 每次循环将变量 `j` 的值累计到变量 `k`, 并让变量 `j` 自增 1。循环开始时变量 `j` 为 0, 每次循环 `j` 增 1, 直至 `j` 等于 11 结束, 共进行 11 次循环。循环计算的 `k` 值可用以下算式表示:

$k=0+1+2+3+4+5+6+7+8+9+10$

经上述 `for` 语句计算后, 变量 `k` 的值是 55。所以答案是 55。

61. 函数调用时, 若形参和实参均为数组, 则其传递的数据的类型是_____。

【答案与解析】在 C 语言中, 函数调用时, 实参向形参传递数据, 传递的数据类型可以是基本类型数据、结构类型数据或指针类型的数据。

强调传递指针类型数据的特殊性, 习惯称传递非指针数据为值传递方式, 传递指针数据为地址传递方式。

(1) 值传递方式: 例如, 传递基本类型的数据, 或传递结构类型数据等。在这种传递方式中, 函数可把实参的值当作形参的初值。在函数执行过程中, 形参变量的值可以改变, 但不会影响原先给它初值的实参。

(2) 地址传递方式: 实参向形参传递地址信息有两种, 传递某个独立变量的地址信息, 传递数组某元素的地址信息。前者, 函数设置指针形参, 实参提供某变量的地址。函数执行时, 函数可利用形参中的某变量的地址间接引用该变量, 不仅可取该变量的内容, 还可改变这个变量的值。后者, 函数设置指针形参或数组形参, 实参提供数组某元素的地址。函数可利用形参中的数组某元素的地址遍历整个数组, 也可取数组元素的内容, 还可改变数组元素的值。

所以, 函数调用时, 若形参和实参均为数组时, 则其传递的数据是指针类型的, 传递的方式是地址传递方式。

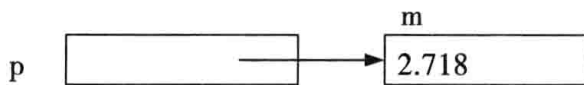
62. 在 C 语言中, _____ 类型变量的所有成员以覆盖方式共享存储单元。

【答案与解析】在 C 语言中, 只有共用型变量的所有成员以覆盖方式共享存储单元。共用型变量也称为共用体、联合体, 或 union 类型变量。

63. 设 `FILE *fp`; 关闭 `fp` 对应文件的操作语句是_____。

【答案与解析】在 C 程序中, 文件使用之前要打开, 文件使用结束后都应及时关闭。关闭文件需调用库函数 `fclose()`, 并指定与文件相对应的文件指针变量。所以, 关闭 `fp` 对应文件的操作语句是 `fclose(fp)`。

64. 设 `float *p, m=2.718`; 根据下列图示, 建立指针变量 `p` 和变量 `m` 之间的关系的赋值语句是_____。



【答案与解析】让一个指针变量指向它允许指向的变量, 只要取被指向变量的地址赋给这个指针变量即可。所以, 实现问题要求的代码为 `p=&m`。

65. 设 `int a[3][4], *p=a[1]`; 该指针变量 `p` 指向的元素是_____。

【答案与解析】在 C 语言中, 一个二维数组可以看做元素是一维数组的一维数组。例如, 问题所给的二维数组 `a[]`, 可以看做由 3 个元素 `a[0]`、`a[1]`、`a[2]` 组成的一维数组, 其中, `a[0]`、`a[1]`、`a[2]` 又各自是由 4 个元素组成的一维数组, 如 `a[1]` 表示由 `a[1][0]`、`a[1][1]`、`a[1][2]`、`a[1][3]` 这 4 个元素组成的数组, 按照这一观点, 表达式 `a[1]` 就是这个一维数组首元素 `a[1][0]` 的指针。所以, 问题中的指针变量 `p` 指向的是 `a[1][0]`。按数组元素的不同标记形式, `a[1][0]` 也可写成 `*(a+1)+0`, 或简写成 `**a+1`。

66. 下面函数 `pi()` 的功能是根据以下公式求 π 值 (直到最后一项小于 10^{-6} 为止)。

$$\frac{\pi}{2} = 1 + \frac{1}{3} + \frac{1}{3} \times \frac{2}{5} + \frac{1}{3} \times \frac{2}{5} \times \frac{3}{7} + \frac{1}{3} \times \frac{2}{5} \times \frac{3}{7} \times \frac{4}{9} + \cdots$$

```
double pi()
{ double s=0.0,t=1.0;int n;
  for(_____; t>_____; n++)
  { s+=t; t=n*t/(2*n+1);
  }
  return 2.0*_____;
}
```

【答案与解析】这是一个典型求级数部分和的程序，用一个循环累计级数的各项之和。循环之前设定当前项的初值，在当前项的值还大于指定值的情况下循环。循环时，将当前项累计到部分和，并利用递推公式由当前项的值计算出下一项的值。记级数为：

$$s(x)=t_1+t_2+\dots+t_k$$

求级数部分和的算法可描述如下：

```
{ s=0.0; /*级数的部分和变量 s 置初值 0*/
  t=首项值; /*置通项变量 t 为级数的首项值*/
  for(n=1; fabs(t)>=Epsilon;n++) /*当前项值还足够大循环*/
  { s+=t; /*累计当前项 tn 到部分和*/
    t=f(t,n); /*由当前项 t 和 n 计算下一个当前项的值*/
  }
}
```

将上述算法应用到函数 pi，不难找出解答。空框 1 填入 n=1，空框 2 填入数 1e-6。在本题给出的函数中，按公式求出的是 $s=\pi/2$ ，即 $\pi=2*s$ 。所以空框 3 的答案是 s。

67. 将一字符串中的小写英文字母改成大写英文字母，试完成以下程序：

```
#include<stdio.h>
void main()
{ int i=0;char str[80];
  scanf("%s",str);
  while(_____)
  { if(_____)str[i]='a'-'A';
    _____;
  }
  printf("%s\n",str);
}
```

【答案与解析】为了改写字符串中的小写英文字母，需要顺序地考察字符串的每一个字符。当发现当前元素 str[i] 是小写英文字母时，就改写 str[i]，减去小写字母与对应大写字母 ASCII 码的差。由于要顺序考察字符串的每个字符，所以横线 1 的循环条件是当前字符 str[i] 不是字符串结束符，写成 str[i] != '\0' 或简写成 str[i]，仅当 str[i] 是小写英文字母时才要改写，表示 str[i] 是小写英文字母的横线 2 可写成 str[i]>='a' && str[i]<='z'。每考察了一个字符后，就要准备考察下一个字符，横线 3 处的语句是让变量 i 增 1，可写成 i++，或 ++i，或 i=i+1，或 i+=1 等。

68. 使用结构型计算复数 x 和 y 的和，试完成以下程序：

```
#include<stdio.h>
void main()
{ struct comp
  { float re,im;
  };
  _____x,y,z;
```

```
scanf("%f%f%f%f",&x.re,&x.im,&y.re,&y.im);
z.re=_____; z.im=_____;
printf("%.2f,%.2f\n",z.re,z.im);
}
```

【答案与解析】用结构型计算复数，程序要定义 3 个 struct comp 结构型变量。横线 1 处的代码就是定义 struct comp 结构型变量 x、y、z 的。所以，在横线 1 处应填入代码 struct comp。两个复数相加，是两个复数的实部和虚部分别相加。所以，在横线 2 处填入代码 x.re+y.re，横线 3 处填入代码 x.im+y.im。

69. 以下程序的输出结果是_____。

```
#include<stdio.h>
void main()
{ int a=10,b=4,c=3;
  if(a<b)a=b;   if(a<c)a=c;
  printf("%d,%d,%d\n",a,b,c);
}
```

【答案与解析】程序中，第 1 个 if 语句 if(a<b)a=b，当 a<b 时，让 b 值置换 a，该语句确保 a 的值不比 b 小。同样，第 2 个 if 语句 if(a<c)a=c，当 a<c 时，让 c 值置换 a，使 a 的值不比 c 小。实际情况是 a 本来就比 b 和 c 都大，所以 a 的值没有变，程序输出 a、b 和 c 的原来值：10,4,3。

70. 以下程序的输出结果是_____。

```
#include<stdio.h>
void main()
{ int x=100,a=10,b=20,ok1=5,ok2=0;
  if(a<b)if(b!=15)if(!ok1)x=1;
  else if(ok2)x=10;
  else x=-1;
  printf("%d\n",x);
}
```

【答案与解析】程序预置变量 x 的值为 100，a 的值为 10，b 的值为 20，ok1 为 5，ok2 为 0。程序中给出的 if 语句可以写成更结构化的形式如下：

```
if(a<b)
  if(b!=15)
    if(!ok1)x=1;
  else if(ok2)x=10;else x=-1;
```

由于条件 a<b 成立，并且条件 b!=15 也成立，而 ok1 不等于 0，执行 else 后的语句，又因 ok2 等于 0，执行 else 后的语句赋 -1 给 x。所以程序输出-1。

71. 以下程序的输出结果是_____。

```
#include<stdio.h>
void main()
{ int y=9;
  for(;y>0;y--)
    if(y%3==0){ printf("%d",--y); continue;}
}
```

【答案与解析】循环前，变量 y 的值为 9，其中，循环语句在 y 大于 0 情况下循环，每次循环

后 y 的值减 1。循环体是当 y 能被 3 整除时输出表达式 $--y$, 输出的是减 1 后的 y 值。这样, 第 1 次循环因 y 为 9, 能被 3 整除, 输出 8, y 也变成 8。又经两次循环, y 的值变为 6, 又让 y 减 1 变成 5, 并输出 5; 又经两次循环, y 的值变成 3, 让 y 减 1 变成 2, 输出 2; 再经两次循环后, y 的值变成 0, 结束循环。所以程序输出 852。

72. 以下程序的输出结果是_____。

```
#include<stdio.h>
void main()
{ static int a[][3]={9,7,5,3,1,2,4,6,8};
  int i,j,s1=0,s2=0;
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      { if(i==j)s1=s1+a[i][j];
        if(i+j==2)s2=s2+a[i][j];
      }
  printf("%d\n%d\n",s1,s2);
}
```

【答案与解析】已知数组 $a[]$ 共有 3 行 3 列, 第 1 行依次是 9, 7, 5; 第 2 行是 3, 1, 2; 第 3 行是 4, 6, 8。程序引用数组元素的下标变量是 i 和 j , 外循环控制变量 i 是数组的行下标, 内循环控制变量 j 是数组的列下标。循环体的工作是将数组的某些元素累计到 $s1$, 某些元素累计到 $s2$ 。将行下标 i 和列下标 j 相同的元素累计到 $s1$, $s1=a[0][0]+a[1][1]+a[2][2]$ 。将行下标 i 与列下标 j 的和为 2 的元素累计到 $s2$, $s2=a[0][2]+a[1][1]+a[2][0]$ 。所以 $s1$ 是 18, $s2$ 是 10。程序输出:

18
10

73. 以下程序的功能是_____。

```
#include<stdio.h>
void main()
{ int i,j;
  float a[3][3],b[3][3],x;
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      { scanf("%f",&x);a[i][j]=x;
        }
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      b[j][i]=a[i][j];
  for(i=0;i<3;i++)
  { printf("\n");
    for(j=0;j<3;j++)
      printf("%f",b[i][j]);
  }
  printf("\n");
}
```

【答案与解析】程序第 1 段两重循环, 实现按行顺序输入数组 $a[]$ 的全部元素的值。程序第 2 段两重循环是将数组 $a[]$ 转置复制到数组 $b[]$ 。程序第 3 段两重循环实现按行输出数组 $b[]$ 的全部元素。所以, 程序的功能是按行顺序输入 3×3 二维数组的全部元素, 然后按列顺序输出。

74. 以下程序的输出结果是_____。

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

```

void main()
{ static char a[]={'*', '*', '*', '*', '*'};
  int i,j,k;
  for(i=0;i<5;i++)
  { printf("\n");
    for(j=0;j<i;j++)printf("%c",' ');
    for(j=0;j<5;j++)printf("%c",a[k]);
  }
  printf("\n");
}

```

【答案与解析】程序的外循环控制循环5次，每次开始先输出一个换行符，共输出5行信息。对于*i*（自0开始计算）行，先输出*i*个空白符，接着输出字符数组*a*中的5个字符，由于*a*中的5个字符都是字符*，所以输出5个*字符。这样，程序输出以下形式的图案：

```

*****
*****
*****
*****
*****

```

75. 以下程序的功能是_____。

```

#include<stdio.h>
void main()
{ int i,j;
  float a[3][3],b[3][3],c[3][3],x;
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      { scanf("%f",&x);a[i][j]=x;
        }
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      { scanf("%f",&x);b[i][j]=x;
        }
  for(i=0;i<3;i++)
    for(j=0;j<3;j++)
      c[i][j]=a[i][j]+b[i][j];
  for(i=0;i<3;i++)
  { printf("\n");
    for(j=0;j<3;j++)
      printf("%f",c[i][j]);
  }
  printf("\n");
}

```

【答案与解析】程序第1段的功能是按行顺序输入数组*a*[][]的元素。第2段的功能是按行顺序输入数组*b*[][]的元素。第3段的功能是顺序累加数组*a*[][]和数组*b*[][]的对应元素，并将和存于数组*c*[][]的对应元素中。最后，按行顺序输出数组*c*[][]的全部元素。

76. 以下程序的输出结果是_____。

```

#include<stdio.h>

```



```

void main()
{ char *a[6]={"AB", "CD", "EF", "GH", "U", "KL"};
  int i;
  for(i=0;i<4;i++)printf("%s",a[i]);
  printf("\n");
}

```

【答案与解析】程序定义了有 6 个元素的字符指针数组 a[], 指针数组 a[] 的各元素指向字符串常量。程序的 for 循环遍历了指针数组 a[] 的前 4 个元素, 用字符串输出格式, 输出指针数组 a[] 前 4 个元素所指的字符串。所以程序输出: ABCDEFGH。

77. 以下程序的功能是_____。

```

#include<stdio.h>
void main()
{ int i,a[10],*p=&a[9];
  for(i=0;i<10;i++)scanf("%d",&a[i]);
  for(;p>=a;p--)printf("%d\n",*p);
}

```

【答案与解析】程序定义了有 10 个元素的整型数组 a[], 另定义了指针变量 p, 并让它指向数组 a[] 的最后一个元素 a[9]。执行代码的第 1 个循环是顺序输入数组 a[] 的 10 个元素。第 2 个循环利用指针 p 逆序遍历数组 a[], 将数组 a[] 的各元素逆序输出。所以程序的功能是输入 10 个整数, 并逆序输出这 10 个整数。

78. 以下程序的输出结果是_____。

```

#include<stdio.h>
char s[]="ABCD";
void main()
{ char *p;
  for(p=s;p<s+4;p++)printf("%s\n",p);
}

```

【答案与解析】程序定义了一个字符数组 s[], 由给它所赋的字符串初值知, 该数组只有 5 个元素。程序另定义了字符指针变量 p。循环开始时, 先让 p 指向数组的首元素。循环每次输出以 p 所指字符为首字符的字符串, 并让它指向下一个字符。如此反复循环 4 次, 所以程序输出以下内容:

```

ABCD
BCD
CD
D

```

79. 以下程序的输出结果是_____。

```

#include<stdio.h>
void main()
{ int i,b,c,a[]={1,10,-3,-21,7,13},*pB,*pC;
  b=c=1;pB=pC=a;
  for(i=0;i<6;i++)
  { if(b<*(a+i)){ b=*(a+i);pB=&a[i];}
    if(c>*(a+i)){ c=*(a+i);pC=&a[i];}
  }
  i=*a;*a=*pB;*pB=i;i=*(a+5);*(a+5)=*pC;*pC=i;
  printf("%d,%d,%d,%d,%d,%d\n",a[0],a[1],a[2],a[3],a[4],a[5]);
}

```


【答案与解析】程序定义了一个整型数组 `a[]`，由给它所赋的初值可知，该数组有 6 个元素。程序另定义了 3 个整型变量 `i`、`b`、`c` 和两个指针变量 `pB`、`pC`。程序首先置变量 `b` 和 `c` 都为 1，`pB` 和 `pC` 都指向数组 `a[]` 的首元素 `a[0]`。接着执行 6 次循环，循环体由两个 `if` 语句。第 1 个 `if` 语句是当 `b` 的值小于 `*(a+i)`（就是 `a[i]`）时，将 `*(a+i)` 的值复制到 `b`，并让 `pB` 指向 `a[i]`，这是在数组 `a[]` 中找最大值，将最大值存于 `b`，最大值元素的指针存于指针变量 `pB`。第 2 个 `if` 语句是当 `c` 的值大于 `*(a+i)`（就是 `a[i]`）时，将 `*(a+i)` 的值复制到 `c`，并让 `pC` 指向 `a[i]`，这是在数组 `a[]` 中找最小值，将最小值存于 `c`，最小值元素的指针存于指针变量 `pC`。循环结束后的前 3 个语句实现最大值与 `a[0]` 交换，接着 3 个语句是在已经过前面交换的基础上，将最小值与 `a[5]` 交换。最后程序顺序输出数组 `a[]` 的 6 个元素。所以程序输出内容是：

13, 10, -3, 1, 7, -21。

若数组 `a[]` 的初值由输入任意指定，上述程序对一种特殊情况不能完成功能的要求，即最小值若是 `a[0]`，`pC` 所指元素已被移到 `pB` 指针所指位置，实际与 `a[5]` 交换的不再是数组中的最小值，而是最大值。

80. 以下函数的功能是_____。

```
ch(int *p1, int *p2)
{ int p;
  if(*p1 > *p2){ p=*p1; *p1=*p2; *p2=p; }
}
```

【答案与解析】函数 `ch()` 有两个指针形参 `p1` 和 `p2`，函数体通过 `*p1` 和 `*p2` 间接引用它们所指的变量。当 `p1` 所指变量的值大于 `p2` 所指变量的值时，交换两形参所指变量的值。所以，函数的功能是当形参 1 所指变量比形参 2 所指变量的值大时，交换这两个变量的值。调用该函数后，能保证实参 1 所指变量的值不大于实参 2 所指变量的值。

81. 以下函数的功能是_____。

```
float av(float a[], int n)
{ int i; float s;
  for(i=0, s=0; i<n; i++) s=s+a[i];
  return s/n;
}
```

【答案与解析】函数有一个数组形参 `a` 和一个整型形参 `n`。函数首先用循环求数组 `a[]` 的前 `n` 个元素和，然后将和除以 `n`，即求数组元素的平均值。

82. 以下程序的输出结果是_____。

```
#include<stdio.h>
void main()
{ char str[]={ 'A', 'B', 'C', '\0', 'D', 'E', 'F', '\0' }, *p=str;
  for(; p-str<sizeof(str); p++)
    printf("%s\n", p);
}
```

【答案与解析】程序用字符指针，顺序遍历字符数组，该字符数组中存放着两个字符串。当指针所指的字符还在第 1 个字符串中时，输出第 1 个字符串中的部分内容。当指针所指的字符是第 2 个字符串中的字符时，输出第 2 个字符串中的部分内容。所以程序输出：

ABC
BC
C

DEF
EF
F

83. 以下程序的输出结果是_____。

```
#include<stdio.h>
void prs(int a[],int n)
{ int max,i,j;
  for(max=a[0],i=1;i<n;i++)
    if(max<a[i])max=a[i];
  for(j=max;j>0;j--)
    { for(i=0;i<n;i++)
      printf("%s",a[i]>=j?"**":"");
      printf("\n");
    }
}
void main()
{ int a[]={1,6,2,5,4};
  prs(a,5);
}
```

【答案与解析】函数先找出数组中的最大值,然后变量j从最大值到1循环,对j的每一个值,顺序考察数组的每个元素,输出一行字符信息。当元素的值比j大时输出双字符**,否则输出两个空格字符。所以,函数的功能是以字符直方图形式输出数组的值。程序的输出结果是:

```

**
**      **
**      **  **
**      **  **
**  **  **  **
**  **  **  **  **

```

84. 以下程序的输出结果是_____。

```
#include<stdio.h>
struct stu
{ int num; char *name; int score; };
struct stu *py(struct stu *p)
{ struct stu s=p;
  for(p++;p->no>0;p++)
    if(s->score<p->score) s=p;
  return s;
}
void main()
{ struct stu student[]={1001,"Sun",76},{1002,"Ling",88},
                        {1003,"Shen",76},{0,"",0}};
  struct stu *p;p=py(student);
  printf("NO:%d NAME:%s SCORE:%d\n",p->num,p->name,p->score);
}
```

【答案与解析】函数py()的功能是在给定的结构数组中找出成绩最高的结构元素,并返回该结构的指针,主函数利用该指针输出这个结构的内容。所以程序输出:

NO: 1002 NAME:Ling SCORE:88

85. 以下程序的输出结果是_____。

```
#include<stdio.h>
unsigned fun6(unsigned num)
```

```

{ unsigned k=1;
  do {k*=num%10;num/=10;}
  while(num);
  return k;
}
void main()
{ unsigned n=26;
  printf("%d\n",fun6(n));
}

```

【答案与解析】函数有一个形参 `num`，循环前变量 `k` 置 1，循环中，表达式 `num%10` 是取 `num` 的个位，将 `num` 的个位值累乘于 `k` 中，然后 `num` 除以 10。循环直至 `num` 为 0 结束，这是逐一求出 `num` 十进制表示的各位数字，并累乘于变量 `k` 的循环。函数最后返回 `k`。函数 `fun6()` 的功能是求整数的各位十进制数字积。所以程序输出 12。

86. 试指出以下函数的功能。

```

printStr(int c,char *v[ ])
{ int i;
  printf("%d\n",c);
  for(i=0;i<c;i++)printf("%s\n",v[i]);
  printf("\n");
}

```

【答案与解析】形参 `char *v[]` 是一个字符串指针数组，每个元素是一个字符串首字符的指针。所以，`v` 是一个字符串表。函数首先输出参数 `c` 的值，然后循环 `c` 次，顺序输出字符指针数组各元素所指的字符串，即输出字符串表。

87. 以下函数的功能是_____。

```

#include<stdio.h>
#include<stdlib.h>
void copyFile(int argc,char *v[ ])
{ FILE *f1,*f2;
  char ch;
  if(argc<2) {printf("字符串表的字符串个数不合理!\n");exit(0);}
  if(((f1=fopen(v[0],"r"))==NULL)||
      ((f2=fopen(argv[1],"w"))==NULL)){printf("不能打开文件!\n");exit(0);}
  while(!feof(f1)) fputc(fgetc(f1),f2);
  fclose(f1);
  fclose(f2);
}

```

【答案与解析】程序中有调用函数 `exit()`，它是一个库函数，该函数调用将结束程序的运行。形参 `v` 是一个字符串表，在这里，字符串是文件的名。函数在 `argc < 2` 时，报告出错信息结束，`argc > 2` 时，以参数指定的第 1 参数和第 2 参数为文件名打开文件，并用循环将第 1 参数的文件内容复制到第 2 参数的文件。复制结束，关闭文件。

88. 以下程序的功能是_____。(字符 0 的 ASCII 码为 16 进制的 30)

```

#include<stdio.h>
void main()
{ union{char c;char i[4];}z;

```

```

    z.i[0]=0x39;z.i[1]=0x36;
    printf("%c\n",z.c);
}

```

【答案与解析】共用型变量 z 有两种可能的成分：一个是字符型的 c ，另一个是有 4 个字符的数组 i 。由于共用型的成分是复盖存储，成分 c 的内容也是成分 i 的首元素。程序给 z 的 i 成分的首元素置十六进制代码 $0x39$ ，即十进制数 57。数 57 是字符 '9' 的 ASCII 码。用字符输出格式，输出该变量的值，将输出字符 '9'。

89. 以下程序的功能是_____。

```

#include<stdio.h>
void main()
{ struct student
  { char name[10];
    float k1;
    float k2;
  } a[2]={{"zhang",100,70},{"wang",70,80}},*p=a;
  int i;
  printf("\nname:%s total=%f",p->name,p->k1+p->k2);
  printf("\nname:%s total=%f",a[1].name,a[1].k1+a[1].k2);
}

```

【答案与解析】结构数组 $a[]$ 有两个元素，每个元素有 3 个成分：10 个元素的字符数组和两个类型为 float 的成分 $k1$ 、 $k2$ 。结构指针变量 p 指向结构数组 $a[]$ 的首元素。其中，初始化使结构数组 $a[]$ 的两个元素分别有值如下： $a[0].name="zhang"$ ， $a[0].k1=100.0$ ， $a[0].k2=70.0$ ， $a[1].name="wang"$ ， $a[1].k1=70.0$ ， $a[1].k2=80.0$ 。所以程序运行后输出：

```

name: zhang total=170.000000
name: wang total=150.000000

```

90. 以下程序的输出结果是_____。

```

#include<stdio.h>
void main()
{ enum em{ em1=3,em2=1,em3};
  char *aa[]={"AA","BB","CC","DD"};
  printf("%s%s%s\n",aa[em1],aa[em2],aa[em3]);
}

```

【答案与解析】由枚举类型 em 的定义知，枚举常量 em1 的值是 3，枚举常量 em2 的值是 1，枚举常量 em3 的值是 2。所以程序运行后输出：

```

DD BB CC

```

91. 以下程序的功能是_____。

```

#include<stdio.h>
void main()
{ FILE *f1,*f2;
  int k;
  if((f1=fopen("c:\\tc\\pl.c","r"))==NULL)
  { printf("Can not open file!\n");
    exit(0);
  }
}

```

```

if((f2=fopen("a:\\p1.c", "w"))==NULL)
{ printf("Can not open file!\n");
  exit(0);
}
for(k=1;k<=1000;k++)
{ if(feof(f1))break;
  fputc(fgetc(f1),f2);
}
fclose(f1);
fclose(f2);
}

```

【答案与解析】程序首先以读方式打开文件 p1.c，以写方式打开文件 p2.c。程序在文件 p1.c 未结束之前读入 1 000 个字符复制到文件 p2.c，然后关闭这两个文件。所以程序的功能是将文件 p1.c 的前 1 000 个字符复制到文件 p2.c。子目录字符'\'在 C 语言中作为转义字符的引导字符，若在字符串中要用这个字符，必须双写该字符。例如，C 盘根目录 tc 子目录文件 p1.c，在 C 程序中应写成字符串"e:\\tc\\p1.c"。

92. 假定在当前盘当前目录下有两个文本文件，其名称和内容如下：

文件名	a1.txt	a2.txt
内容	121314#	252627#

以下程序的输出结果是_____。

```

#include<stdio.h>
void main()
{ FILE *fp;
  if((fp=fopen("a1.txt", "r"))==NULL)
  { printf("Can not open file!\n");    exit(1);
  }
  else { fc(fp);fclose(fp); }
  if((fp=fopen("a2.txt", "r"))==NULL)
  { printf("Can not open file!\n");    exit(1);
  }
  else { fc(fp);fclose(fp); }
}
void fc(fpl)
FILE *fpl;
{ char c;
  while((c=fgetc(fpl))!='#')putchar(c);
}

```

【答案与解析】程序以读方式打开文件 a1.txt，若能正常打开，则调用函数 fc()，接着以读方式打开文件 a2.txt，若能正常打开，也调用函数 fc()。函数 fc()从指定的文件读入字符，在读入字符不是字符'#'情况下循环，将读入字符输出。对于文件 a1.txt，程序输出 121314，对于文件 a2.txt，程序输出 262627。所以程序输出 121314252627。

93. 以下程序的输出结果是_____。

```

#include<stdio.h>
#define PR(ar) printf("%d",ar)
void main()

```

```

{ int j,a[]={1,3,5,7,9,11,15},*p=a+5;
  for(j=3;j;j--)
  switch(j)
  { case 1:
    case 2:PR(*p++);break;
    case 3:PR(*(--p));
  }
  printf("\n");
}

```

【答案与解析】程序的宏定义 PR 以 d 格式输出参数的值和逗号字符。程序给数组 a[] 赋初值,并由指针 p 指向元素 a[5]。程序让 j 从 3 至 1,共循环 3 次。当 j 是 3 时,让 p 减 1,并输出 p 所指元素的值 9;当 j 是 2 和 1 时,输出 p 所指的数组元素的值,并让 p 增 1,顺序输出 9 和 11。所以程序输出 9,9,11。

94. 以下程序的输出结果是_____。

```

#include<stdio.h>
void main()
{ char str[]="1234567"; int i;
  for(i=0;i<7;i+=3)
    printf("%s\n",str+i);
}

```

【答案与解析】程序的字符数组 str[] 中存有字符串 "1234567"。循环语句由变量 i 控制, i 的初值为 0,循环条件是 i<7,每次循环后 i 增 3,所以, i 的值依次为 0、3、6,共进行了 3 次循环。每次循环输出数组 str[] 中从 str+i 位置开始的部分字符串内容。第 1 次循环, i 的值为 0,输出内容是 1234567;第 2 次循环, i 的值为 3,输出内容是 4567;第 3 次循环, i 的值为 6,输出内容是 7。所以程序输出:

```

1234567
4567
7

```

95. 以下程序的输出结果是_____。

```

#include<stdio.h>
void main()
{ int i,a[5];
  for(i=0;i<5;i++)
    a[i]=9*(i-2+4*(i>2))%5;
  for(i=4;i>=0;i--)
    printf("%3d",a[i]);
}

```

【答案与解析】程序的第 1 个循环对于 i 取 0~4 为数组 a[] 设置值。表达式 $9*(i-2+4*(i>2))\%5$ 中的子表达式 (i>2),在 i 等于 0、1、2 时为 0;在 i 等于 3、4 时为 1。所以数组 a[] 元素的值依次是 -3、-4、0、0、4。程序中的第 2 个循环是逆序输出数组 a[] 中各元素值。所以程序输出 4 0 0 -4 -3。

96. 以下程序的输出结果是_____。

```

#include<stdio.h>
struct stu
{ int num;char name[10];int age;};
void py(struct stu *p)

```

```

{ printf("%s\n", (*p).name); }
void main()
{ struct stu student[3]=
    {{1001, "Sun", 25}, {1002, "Ling", 23}, {1003, "Shen", 22}};
  py(student+2);
}

```

【答案与解析】程序定义的结构类型 struct stu 有 3 个成分，其中，成分 name[] 是一个字符数组。主函数中用结构类型 struct stu 定义了结构数组变量 student[]，并对其 3 个结构元素分别赋了结构初值。主函数调用函数 py() 时，提供的实参是 student+2，这是结构数组 student[] 的第 3 个结构元素的指针。函数 py() 以结构 struct stu 的指针类型为形参，函数输出形参所指结构的成分 name[] 中的字符串。所以程序将输出 Shen。注意，函数 py() 中的代码 (*p).name 也可等价地写成 p->name，且后者是习惯写法。

97. 以下程序的输出结果是_____。

```

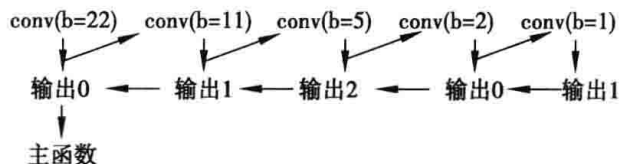
#include<stdio.h>
void conv(int b)
{ if(b>=2) conv(b/2);
  printf("%d", b%2);
}
void main()
{ int d;
  scanf("%d", &d); conv(d);
}

```

输入：

22✓

【答案与解析】程序中的函数 conv() 有一个整数形参 b，该函数是一个递归函数。函数的递归算法是当形参 $b \geq 2$ 时就继续以 $b/2$ 为实参递归。递归调用返回时，输出 $b\%2$ 。即当 b 是偶数时，输出 0；b 是奇数时，输出 1，或者说，输出 b 的二进制表示的个位。主函数以输入值 22 为实参调用函数 conv()，使函数 conv() 引起以下一系列的递归调用和输出：



由上述递归调用的执行过程知，对于输入 22，程序的这次执行将输出 10110。

98. 以下程序的输出结果是_____。

```

#include<stdio.h>
void main()
{ char c;
  while((c=getchar())!='\n')
    switch(c)
    { case '0' :
      case '1': putchar(c);
      case '2': putchar(c); break;
      case '3': putchar(c);
      default : putchar(c+1); break;
    }
}

```

```

    }
    printf("\n");
}
输入:
1357✓

```

【答案与解析】主函数是一个输入字符行的循环，循环体是一个 switch 语句，实现对输入的每个字符分别进行不同的处理。若输入字符是'0'或'1'，重复输出该字符两次；若输入字符是'2'，则输出该字符；若输入字符是'3'，则输出字符'3'和字符'4'；若是其他字符，则一律输出输入字符的后继字符。由以上分析，该程序对于输入的字符序列 1357，程序将输出字符列 113468。

99. 以下程序的功能是_____。

```

#include<stdio.h>
#define N 40
void main()
{ char s1[N],s2[N];
  int i,j;
  printf("Enter string1!\n");
  scanf("%s",s1);
  printf("Enter string2!\n");
  scanf("%s",s2);
  for(i=0;s1[i]!='\0';i++);
  for(j=0;s2[j]!='\0';i++,j++)
    s1[i]=s2[j];
  s1[i]='\0';
  printf("s1=%s\n",s1);
}

```

【答案与解析】程序输入两个字符串，接着找到字符串 s1 的末尾，然后参照字符串 s2，将字符串 s2 的全部有效字符逐一复制到字符串 s1，再在字符串 s1 的末尾接上字符串结束标记符，最后输出字符串 s1。程序的功能是输入两个字符串，将它们连接后输出。

100. 以下程序的功能是_____。

```

#include<stdio.h>
void main()
{ int a[20],i,base=10;
  int n;
  printf("Enter a number!\n");scanf("%d",&n); /*整数输入*/
  i=0; /*对 n 进行分拆，各位数字自低位到高位存于数组 a*/
  do
  { a[i++]=n%base;
    n/=base;
  } while(n);
  for(i--;i>=0;i--) /*自高位到低位输出*/
    printf("%d%c",a[i],i?',' ':' '\n');
}

```

【答案与解析】程序的主要工作是从输入整数分拆出它的十进制整数的各位数字，将分拆出来的各位数字存于数组中，然后将这些数字自高位到低位逐位输出。程序用一个循环将一个整数按十进制数的要求分拆，每次循环用求出被分拆数除以 10 的余数的办法求出它的个位，每分拆出一位就将被分拆的数除以 10，直至被分拆的数为 0 循环过程结束。程序的功能是输入一个整数，将它按十进制数按位分拆输出，输出时在数字之间插入逗号。

第三部分 Visual C++ 6.0 快速入门

Microsoft 公司推出的 Visual C++ 6.0 集成环境，由于是在 Windows 环境下工作，有良好的界面，并支持汉字系统，还全面支持 C 程序的编译、连接、运行与调试，所以笔者建议用 Visual C++ 6.0 集成环境作为学习 C 程序设计的环境。

下面简要介绍 Visual C++ 6.0 的上机操作方法。

一、启动 Visual C++ 6.0 集成环境

单击桌面上的“开始”按钮，依次选择“程序”→Microsoft Visual Studio 6.0→Microsoft Visual C++ 6.0 命令，屏幕上出现 Microsoft Visual C++ 集成环境窗口(以下简称 VC 窗口)，如图 3-1-1 所示。

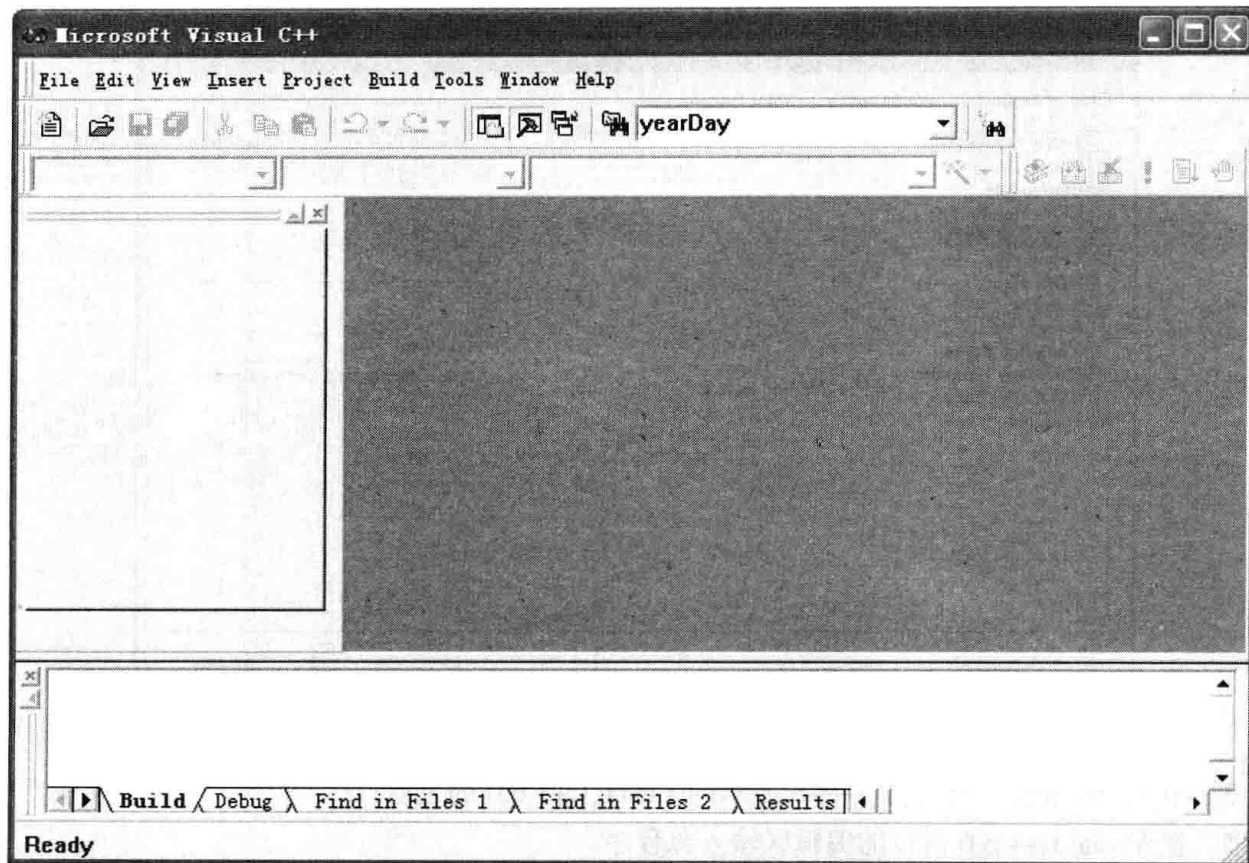


图 3-1-1 Microsoft Visual C++ 6.0 集成环境主窗口

二、新建一个 C 源程序

1. 在 Microsoft Visual C++ 6.0 主窗口中，选择 File (文件)→New (新建) 命令，在屏幕上弹出一个 New 对话框，如图 3-2-1 所示。

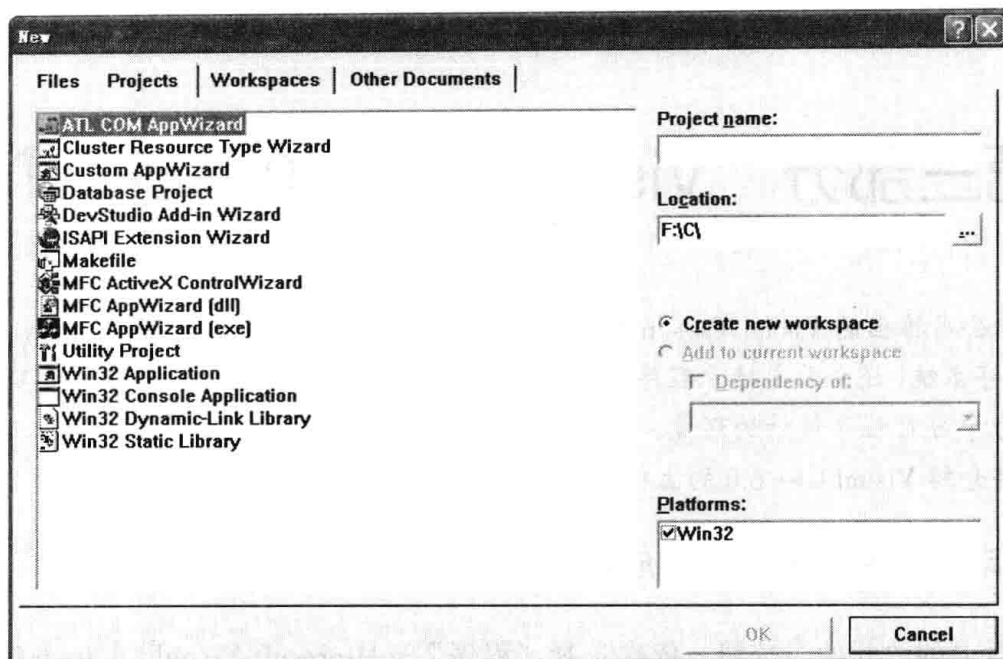


图 3-2-1 New 对话框中的 Projects 选项卡

2. 选中 New 对话框上 Files 选项卡, 并选择 C++ Source File 选项, 如图 3-2-2 所示。

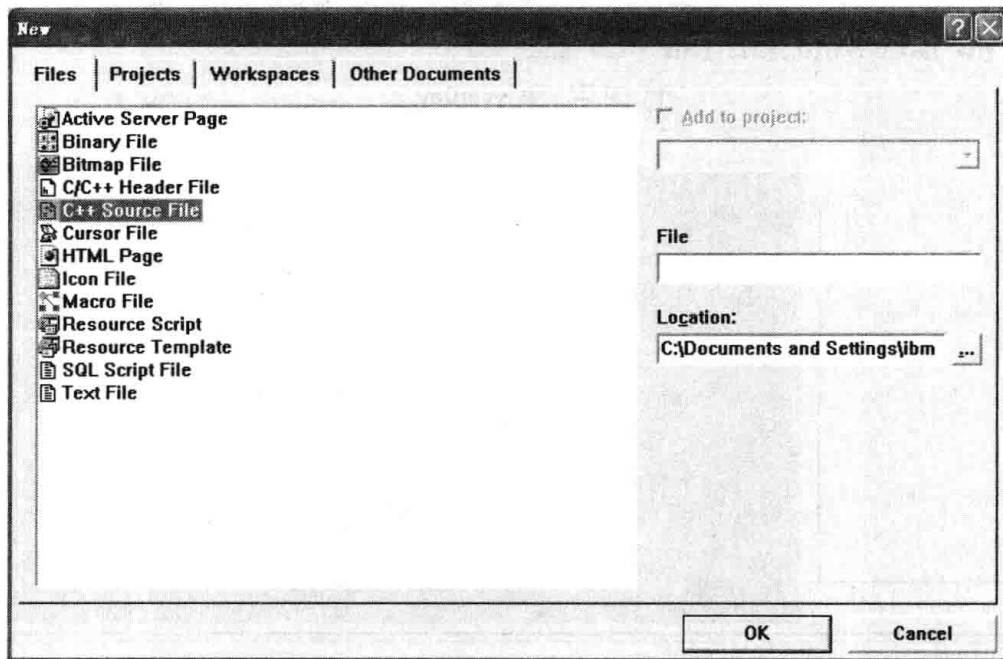


图 3-2-2 New 对话框中的 Files 选项卡

3. 单击 OK 按钮, 打开图 3-2-3 所示的 Visual C++ 6.0 的主窗口。
4. 在 Visual C++ 6.0 窗口的编辑区输入源程序。
5. 选择 File (文件) → Save As (保存为) 命令, 在弹出的“保存为”对话框中选择文件保存的路径及文件名, 如图 3-2-4 所示。

这里须注意的是, 如果给文件加上扩展名.c, 则表示这是 C 源程序, 如果用默认的扩展名.cpp, 则表示这是 C++源程序。

在这里须要注意的是,如果打开了多个源程序,如分别打开了 Ex1.c 和 Ex2.c 两个源程序,而且 Ex1.c 程序已编译并运行过,此时可以编译 Ex2.c 源程序但不能运行。也就是说,若单击运行按钮 !,仍运行 Ex1.c 程序。此时,可以采取如下的解决方法:

1. 选择 File (文件) → Close Workspace (关闭工作区) 命令,弹出图 3-3-2 所示的关闭文档窗口对话框。

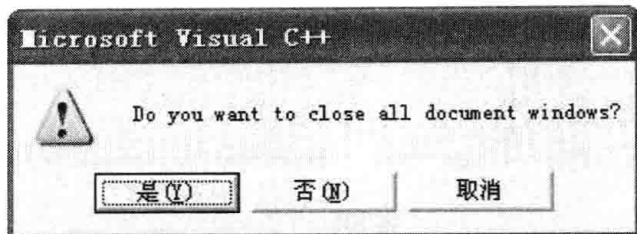


图 3-3-2 关闭文档窗口对话框

2. 单击“是”按钮,表示关闭所有文档窗口及其工作区;单击“否”按钮,表示仅仅关闭已使用的工作区,在这里应单击“否”按钮。

3. 单击“否”按钮后,原 Ex1.c 的工作区被关闭,此时可以对 Ex2.c 源程序进行编译、连接和运行。

四、源程序编译

当编辑和保存了源程序后,可以对源程序进行编译(快捷键【Ctrl+F7】)。编译的首要目标是检查源程序是否存在语法错误。其次,对于没有语法错误的源程序,产生与源程序相对应的目标文件。

源程序编译的方法如下:

1. 单击工具栏上的 Compile 按钮  或选择 Build(构建) → Compile Example.c(编译 Example.c) 命令,弹出图 3-4-1 所示的对话框,提示是否同意建立一个默认的项目工作区。

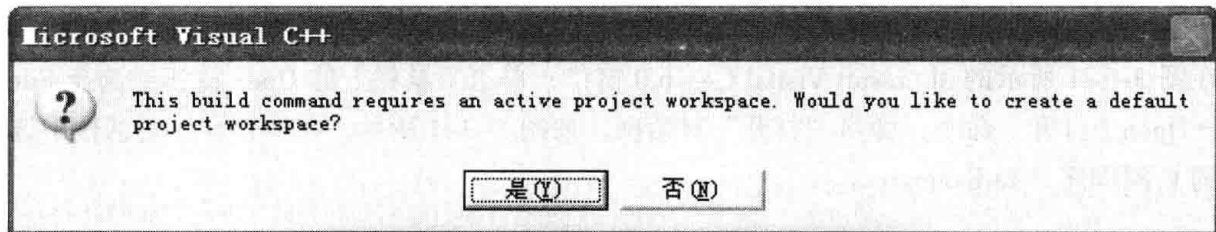


图 3-4-1 “是否同意建立一个默认的项目工作区”对话框

2. 单击“是”按钮,表示同意由编译系统建立一个默认的项目工作区,然后开始对源程序进行编译;单击“否”按钮,将不会对源程序进行编译。

3. 若编译成功,在 VC 窗口的输出区显示“0 error(s), 0 warning(s)”,如图 3-4-2 所示。“0 error(s), 0 warning(s)”表示既没有致命错误(error)也没有警告错误(warning)。这时已产生了一个目标文件 Example.obj,可以进入程序的连接与运行。

4. 若编译有错,双击提示的错误信息,在源程序错误行前会出现“→”标记。此时,应检查标记所在行(可能会是标记前一行)的程序代码,找出错误的原因并改正。

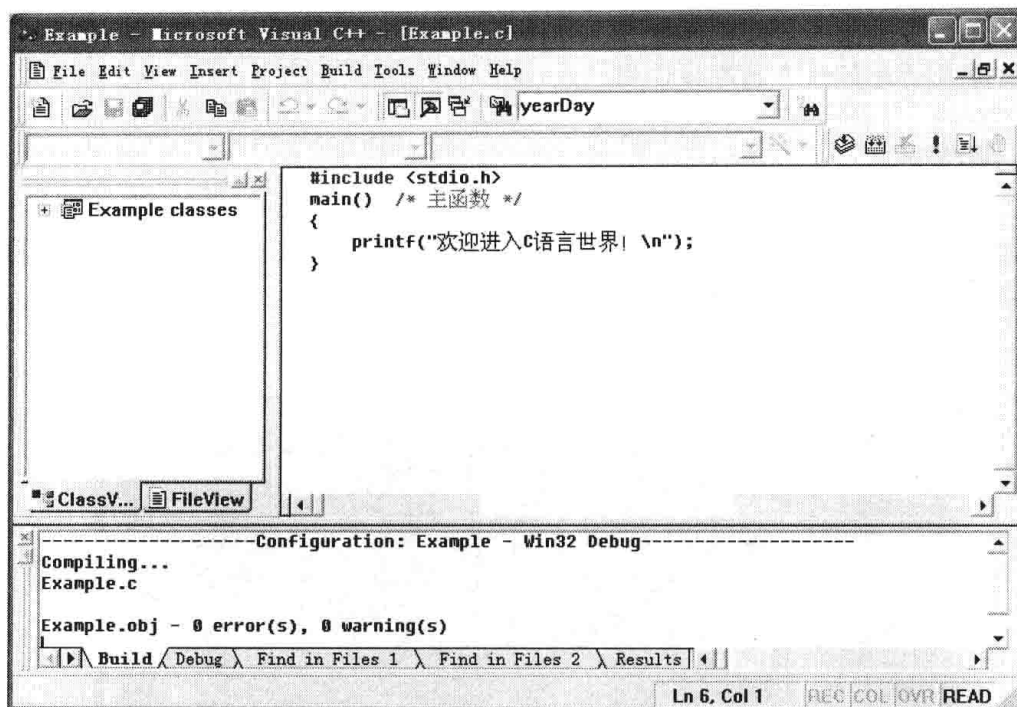


图 3-4-2 程序编译后的 VC 主窗口

五、程序连接

当源程序编译成功并产生目标文件.obj 后, 就可以对程序进行连接 (快捷键【F7】)。程序连接的方法是单击工具栏上 Build 按钮  或选择 Build (构建) → Build Example.exe (构建 Example.exe) 命令, 表示对目标文件 Example.obj 进行连接。若连接成功将生成一个可执行文件 Example.exe, 如图 3-5-1 所示。

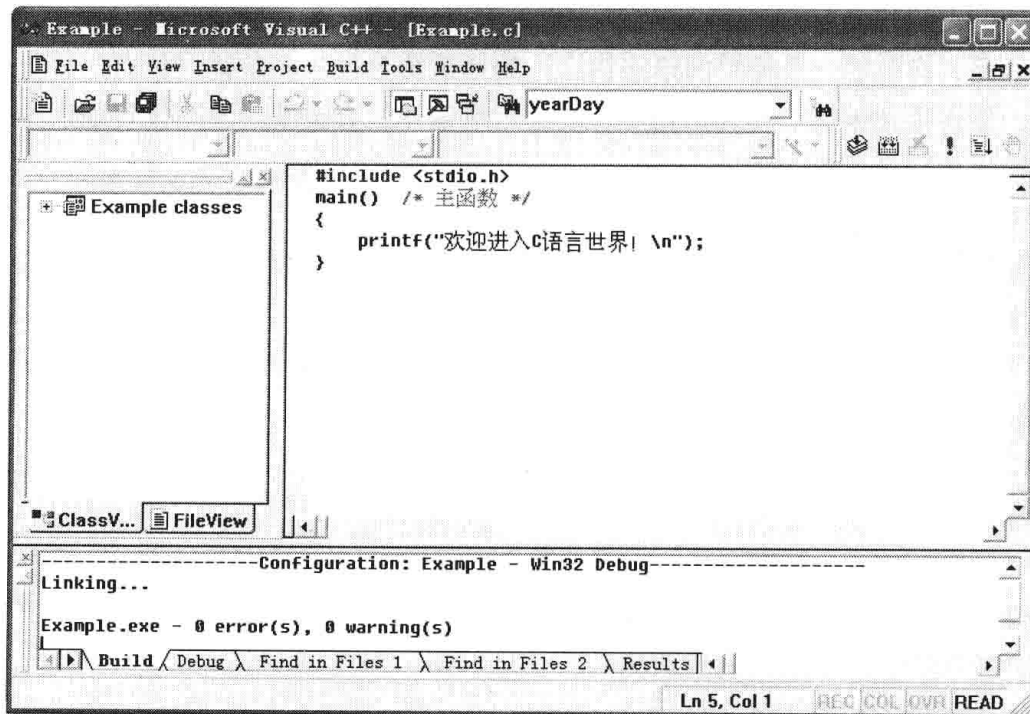


图 3-5-1 程序连接后的 VC 窗口

说明：当源程序编译成功并产生目标文件.obj 后，可以直接进入程序的执行。也就是说，在程序执行之前会自动进行程序的连接过程。

六、程序执行

当程序连接成功并产生可执行文件.exe 后，就可以运行程序（快捷键【Ctrl+F5】）。程序运行的方法是单击工具栏上的 Execute 按钮  或选择 Build 构建 →  Execute Example.exe（执行 Example.exe）命令。执行 Example.exe 产生图 3-6-1 所示的结果窗口。



图 3-6-1 程序运行后的结果窗口

七、修正编译指出的程序错误

编译程序能发现的源程序中的错误可分成两类：一类是源程序中的语法错误，编译程序用 error 表示。源程序的语法错误是致命的，编译通不过，编译系统不能为源程序产生目标文件。另一类是警告错误，如数据类型不匹配等，用 warning 表示。此时，程序可能会在连接时继续报错，不能产生可执行文件，也可能产生可执行文件，但可能会影响程序执行的结果。

须注意的是，有时候一个简单的语法错误，编译系统会报告多条错误信息。首先要找出第 1 个出错信息，然后重新编译，再找出错误，改正后再编译，直至编译没有发现程序中有错误和警告，为程序产生目标文件为止。

按编译报错信息修改程序错误的方法如下：

例如，编制一个计算两数之和的程序，在程序中有 3 处故意打错，如图 3-7-1 所示。

其中，第 5 行应改为：

```
printf("Input x and y\n"); /*提示信息*/
```

第 7 行应改为：

```
sum=x+y; /*运算*/
```

第 8 行应改为：

```
printf("x+y=%d\n",sum); /*输出结果*/
```

当程序编译时，在 VC 窗口的输出区显示“Ex1.obj-3 error(s), 1 warning(s)”，表示有 3 个致命的错误，1 个警告错误，如图 3-7-1 所示。

(1) Ex1.c(5): error C2001: newline in constant。

错误在第 5 行，表示本行未结束。

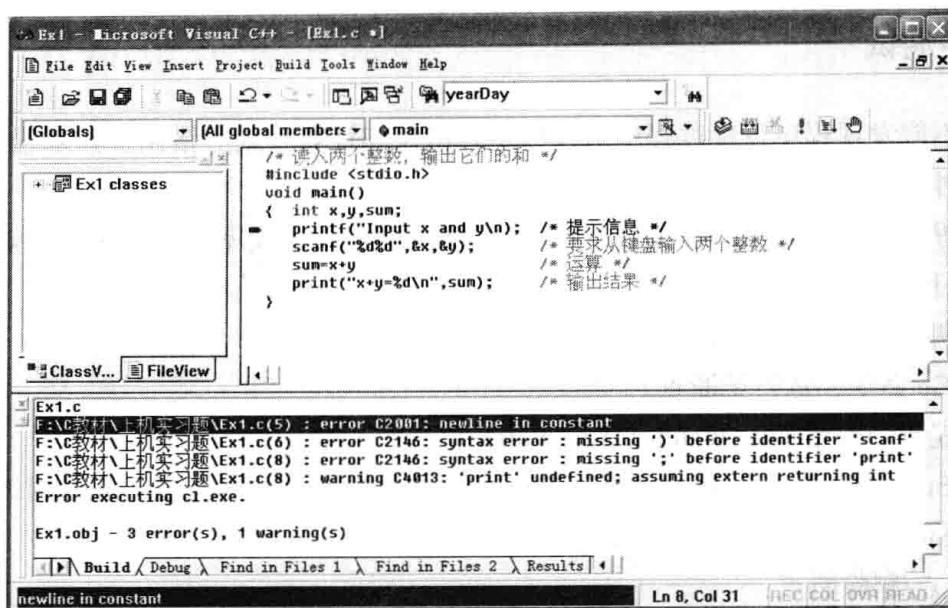


图 3-7-1 编译出错后的 VC 窗口

(2) Ex1.c(6): error C2146: syntax error : missing ')' before identifier 'scanf'.

错误在第 6 行，表示在标识符 scanf 之前遗漏了右括号“)”。由于第 5 行结尾漏了双引号“”，导致误报第 6 行有错。

(3) Ex1.c(8): error C2146: syntax error : missing ';' before identifier 'print'.

错误在第 8 行，表示在标识符 print 之前遗漏了分号“;”。

(4) Ex1.c(8): warning C4013: 'print' undefined;。

错误在第 8 行，表示 print 没有定义过。这是一个警告错误，尽管不会影响产生目标文件，但由于 print 没有被定义，将会在程序连接时导致失败。

当更正了上述错误后重新编译，此时，在输出区显示“Ex1.obj-0 error(s), 0 warning(s)”，表示既无致命错误 (error)，也无警告错误 (warning)，编译成功并生成一个目标文件 Ex1.obj，如图 3-7-2 所示。

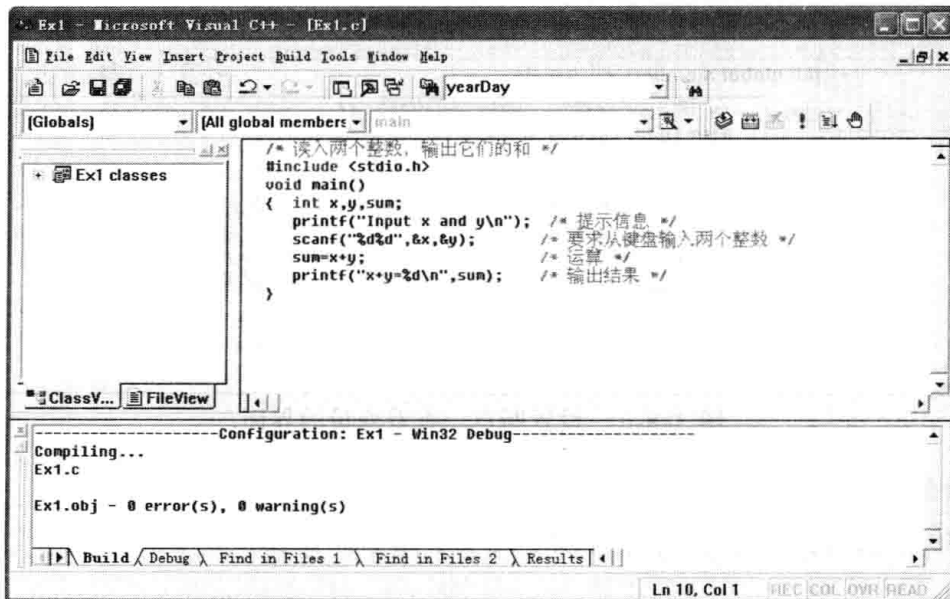


图 3-7-2 编译成功后的 VC 窗口

八、程序调试

编译系统不能发现程序设计过程中引入的逻辑错误和计算公式等错误。如果程序运行结果有错误, 只能通过程序调试去发现错误和改正错误。

为了便于程序调试, VC 集成环境中有一个程序调试工具, 程序调试方法简述如下:

1. 调试过程:

- (1) 在要调试的程序段的关键行上加上断点。
- (2) 选择 Go 命令, 运行至断点。
- (3) 将要观测的变量加入 Watch (Add Watch)。
- (4) 选择 Step Over 或 Step Into 命令, 单步执行, 观察 Watch 的变化, 进行调试。
- (5) 选择 Stop Debugging 命令, 退出调试会话。

2. 设置断点/删除断点:

在程序中可以设置若干个断点, 当程序执行到断点位置时暂停, 用户可以检查该断点前有关变量的值, 用于分段调试程序的正确性。其方法是: 将光标移到要查看的变量上, 就会显示该变量的值。

设置断点/删除断点的方法如下:

方法一: 将源程序的编辑插入点移到要设置或删除断点的行, 并单击工具栏上的 Insert/Remove Breakpoint 按钮 。

方法二: 右击要设置或删除断点的行, 选择 Insert (插入) → Remove Breakpoint (删除断点) 命令。

方法三: 将源程序的编辑插入点移到要设置或删除断点的行, 按【F9】键 (快捷键是【F9】)。当某一行设置断点后, 该行左边将出现一个红色的●。

图 3-8-1 所示显示了设置断点, 使用 Go 命令执行到断点, 并查看变量 sum 的值为 8。

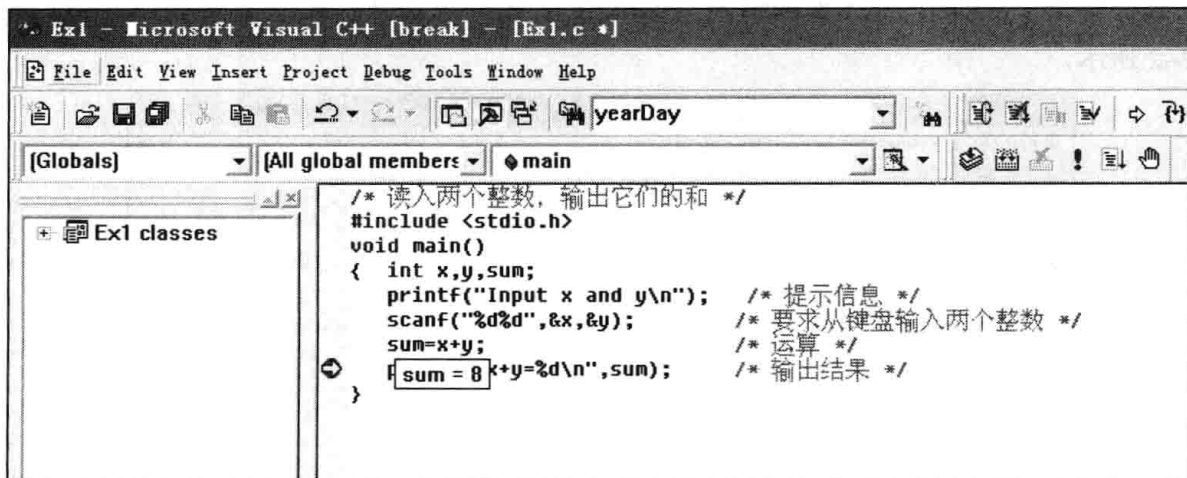


图 3-8-1 设置断点、查看变量的界面

3. 调试命令:

图 3-8-1 中的 Debug (调试) 菜单有以下几个子命令, 如表 3-8-1 所示。

表 3-8-1 Debug 菜单下的子命令功能表

子 命 令	快 捷 键	功 能
Go	F5	运行程序至断点，或程序结束
Restart	Ctrl+Shift+F5	重新载入程序，并启动执行
Stop Debugging	Shift+F5	关闭调试会话
Break		从当前位置退出
Step Into	F11	单步执行，并进入调用函数
Step Over	F10	单步执行，但不进入调用函数
Step Out	Shift+F11	跳出当前函数，回到调用处
Run to Cursor	Ctrl+F10	运行至当前光标处

4. 调试快捷对话框：

在调试过程中，会打开一个调试快捷对话框，集中了所有的调试指令，如图 3-8-2 所示。该对话框也可在快捷菜单栏处右击，在弹出的快捷菜单中选择 Debug 命令调出。

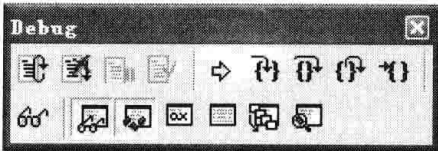


图 3-8-2 调试快捷对话框

第四部分 Visual Studio 2010 快速入门

随着操作系统的版本越来越高，在新的操作系统环境下已不能使用 Visual C++ 6.0 来编译和运行 C 或 C++ 程序。由此 Microsoft 公司推出了全新的 Visual Studio 2010 中文版，全面支持 C/C++、C#、VB 等程序的编译、运行与调试。下面我们一起学习在 Windows 7 或 Windows 8 操作系统环境下运行 C 程序的操作方法。

一、启动 Visual Studio 2010 中文版

单击桌面上的“开始”按钮，依次选择“所有程序”→Microsoft Visual Studio 2010→Microsoft Visual Studio 2010 命令，屏幕上出现 Microsoft Visual Studio 2010 起始页窗口（以下简称 VS 2010 主窗口），如图 4-1-1 所示。

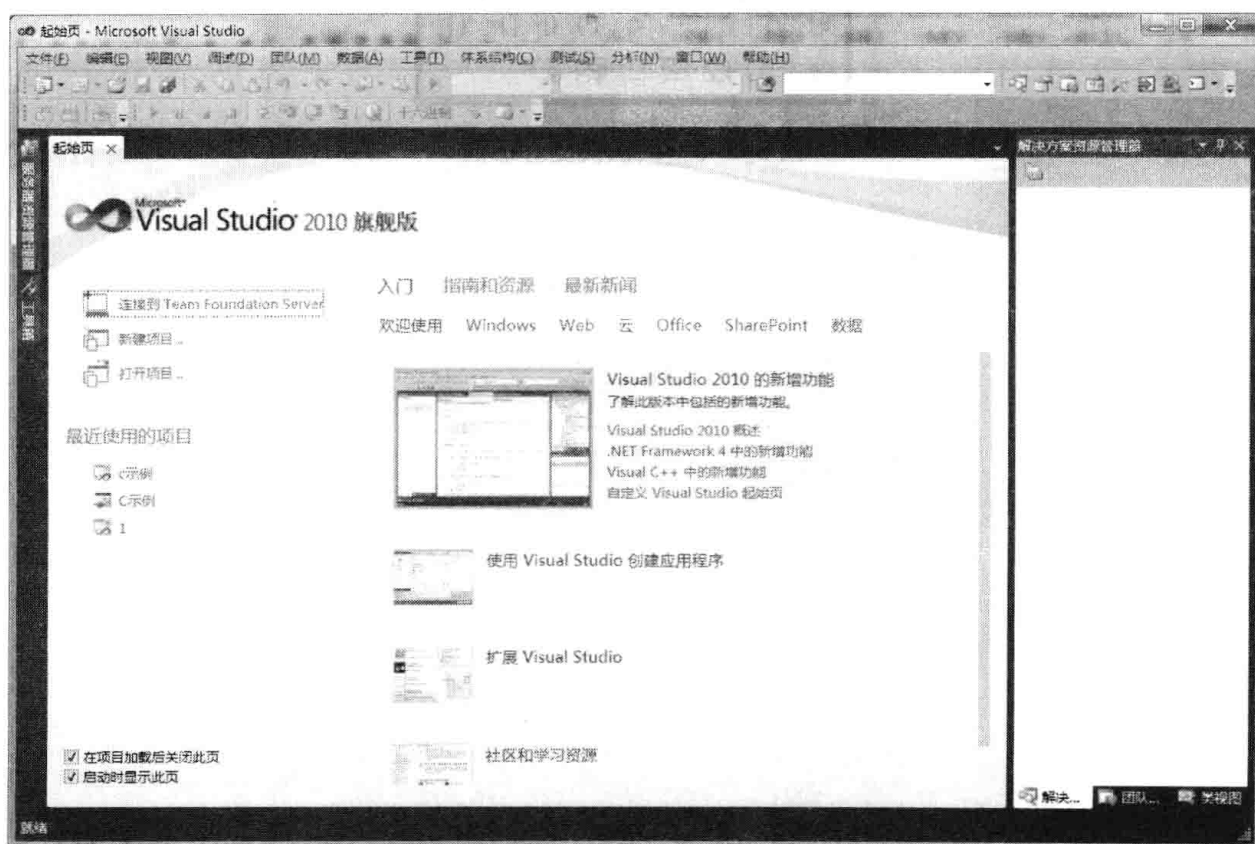


图 4-1-1 Microsoft Visual Studio 2010 起始页窗口

二、新建一个项目及 C 源程序

1. 在 Microsoft Visual Studio 2010 起始页窗口中，选择“文件”→“新建”→“项目”命令，在屏幕上弹出一个“新建项目”对话框，如图 4-2-1 所示。

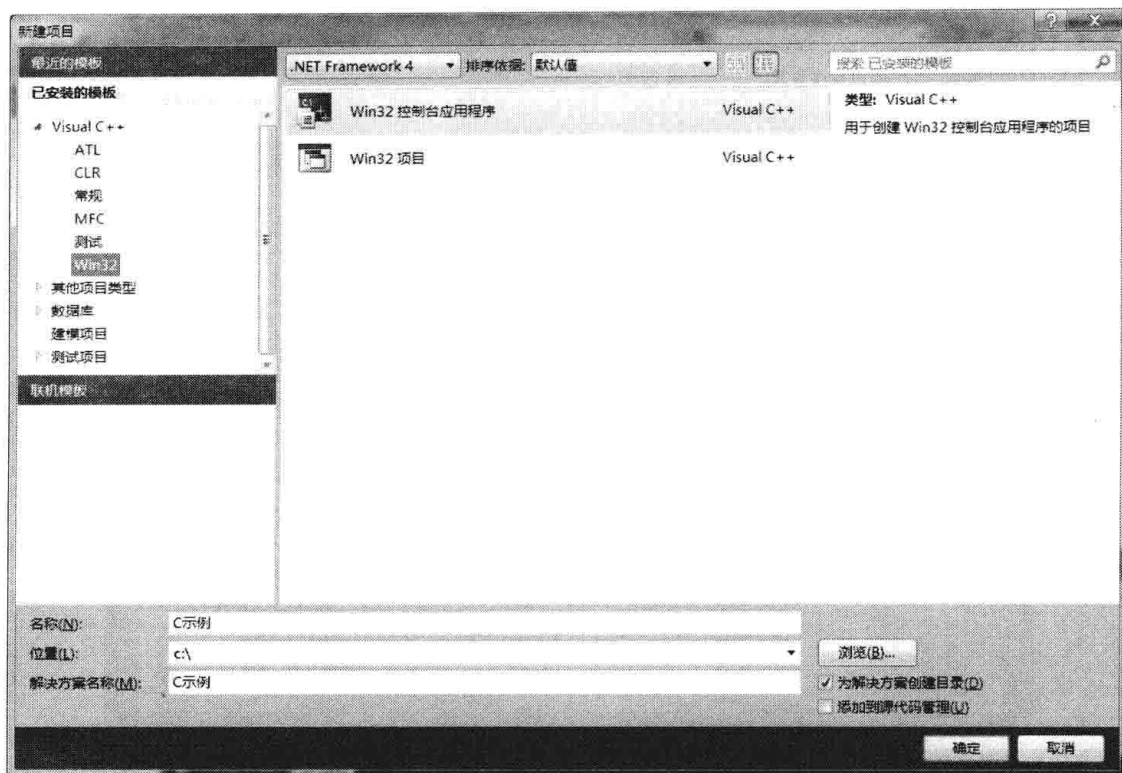


图 4-2-1 “新建项目”对话框

2. 在左窗格中选择 Visual C++ 下的“Win32”模板，在中间窗格中选择“Win32 控制台应用程序”选项，在底部“名称”文本框中输入一个项目名称如“C 示例”，在“位置”文本框中选择存放“C 示例”项目名称存放的位置如 C 盘根目录，如图 4-2-1 所示。单击“确定”按钮，弹出“Win32 应用程序向导”对话框 1，如图 4-2-2 所示。

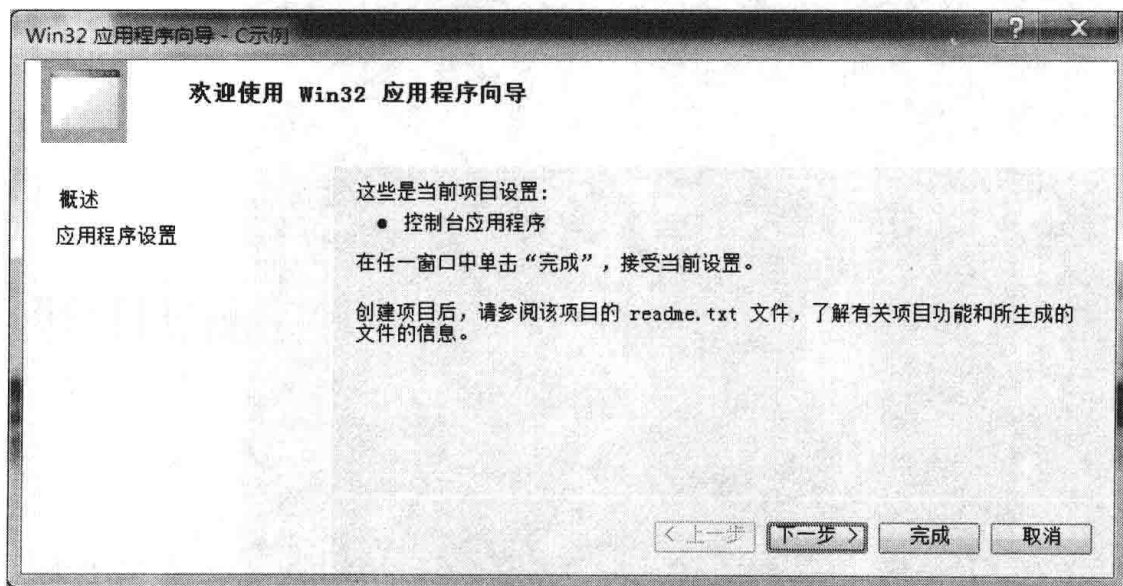


图 4-2-2 “Win32 应用程序向导”对话框 1

3. 单击“下一步”按钮，打开图 4-2-3 所示的“Win32 应用程序向导”对话框 2。

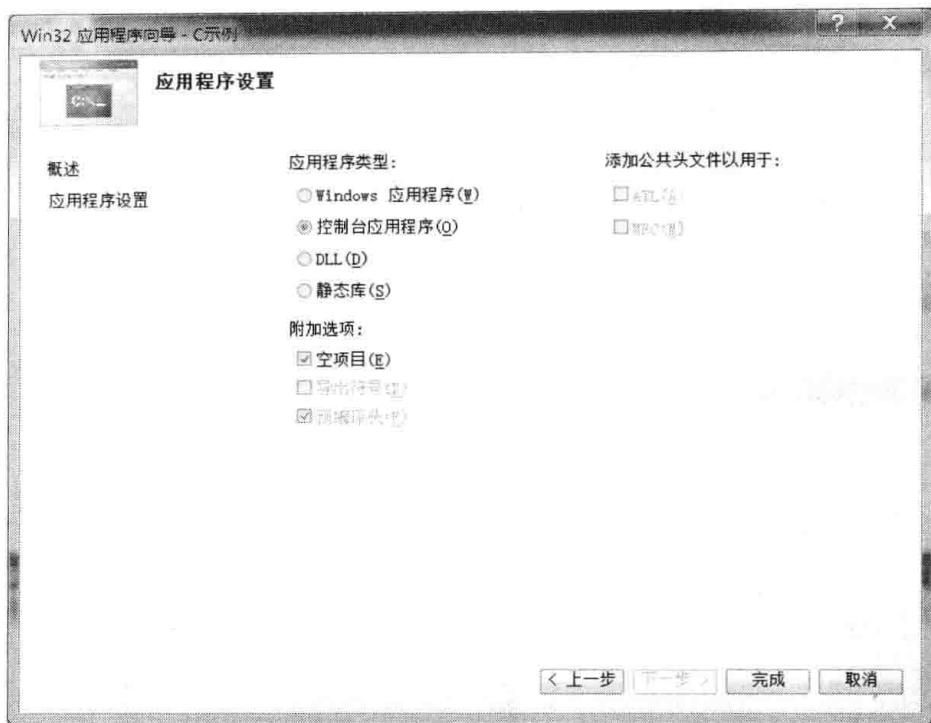


图 4-2-3 “Win32 应用程序向导”对话框 2

4. 在“附加选项”选项区域选中“空项目”复选框，单击“完成”按钮，返回到“VS2010 主窗口”，如图 4-2-4 所示。

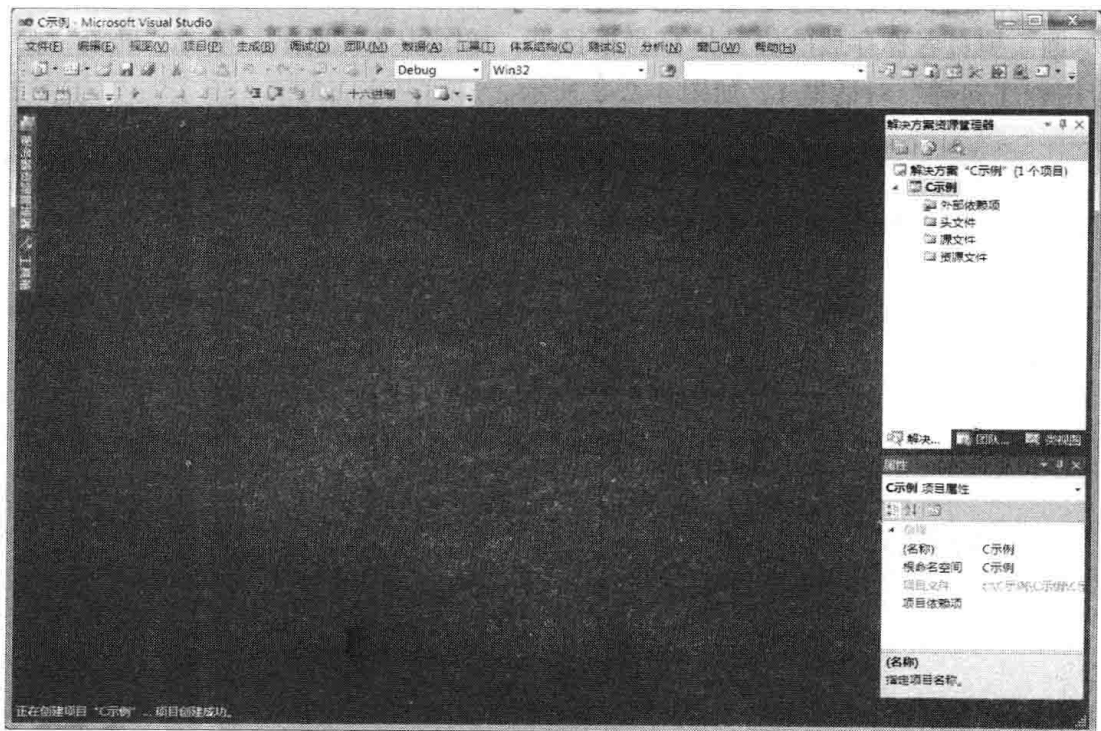


图 4-2-4 VS2010 主窗口

5. 在右窗格“解决方案资源管理器”中右击“源文件”，在弹出的快捷菜单中选择“添加”→“新建项”命令，弹出如图 4-2-5 所示的“添加新项”对话框。

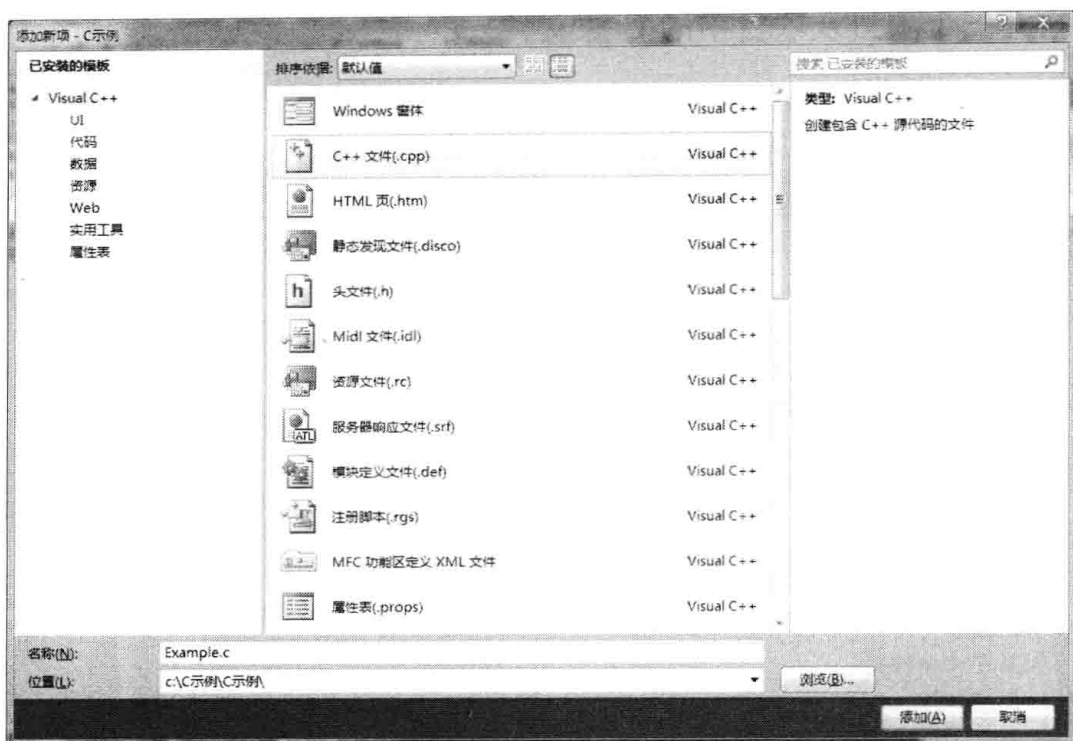


图 4-2-5 “添加新项”对话框

6. 在中间的窗格中选择“C++文件(.cpp)”，在底部“名称”文本框中输入要创建的 C 源程序文件名如 Example.c（给文件加上扩展名.c，则表示这是 C 源程序，否则使用默认的扩展名.cpp，表示这是 C++源程序）。单击“添加”按钮，返回到“VS2010 主窗口”。这时在左边的区域就可以编写 C 源程序了，如图 4-2-6 所示。

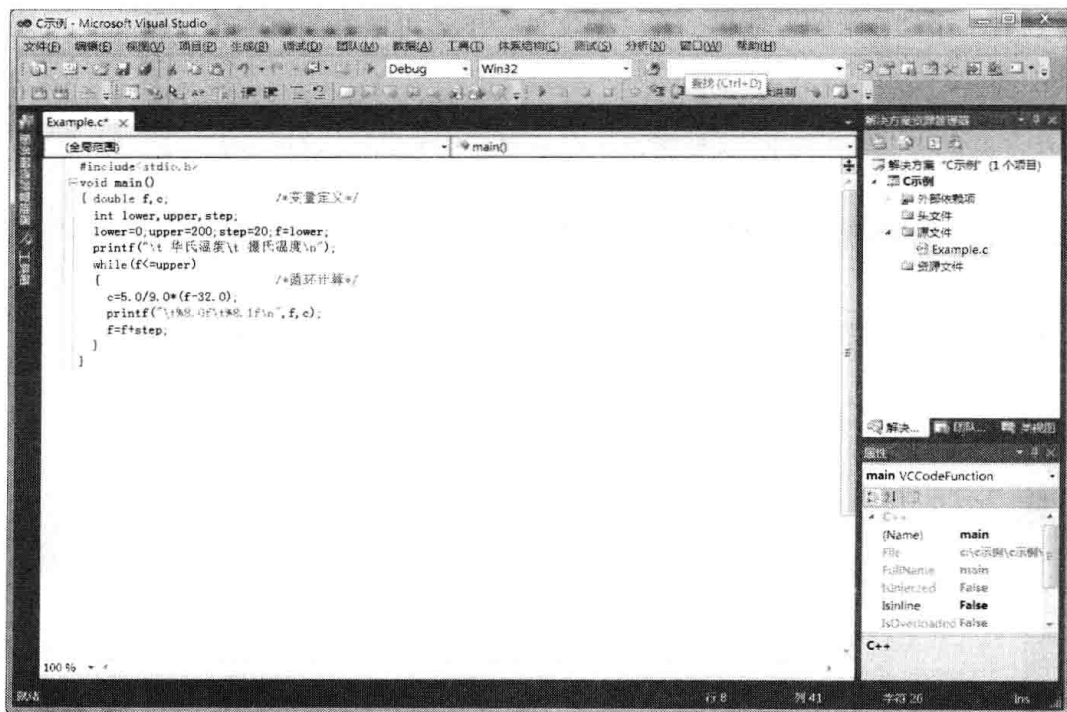


图 4-2-6 在“VS2010 主窗口”中编写 C 源程序

注意：在图 4-2-4 的右窗格“解决方案资源管理器”中，右击“源文件”，在弹出的快捷菜单中选择“添加”→“现有项”命令，可以直接添加已编写好的 C 源文件。

三、打开一个项目及项目中的 C 源文件

1. 在图 4-1-1 所示的 Microsoft Visual Studio 2010 起始页窗口中, 选择“文件”→“打开”→“项目/解决方案”命令, 在屏幕上弹出一个“打开项目”对话框, 如图 4-3-1 所示。

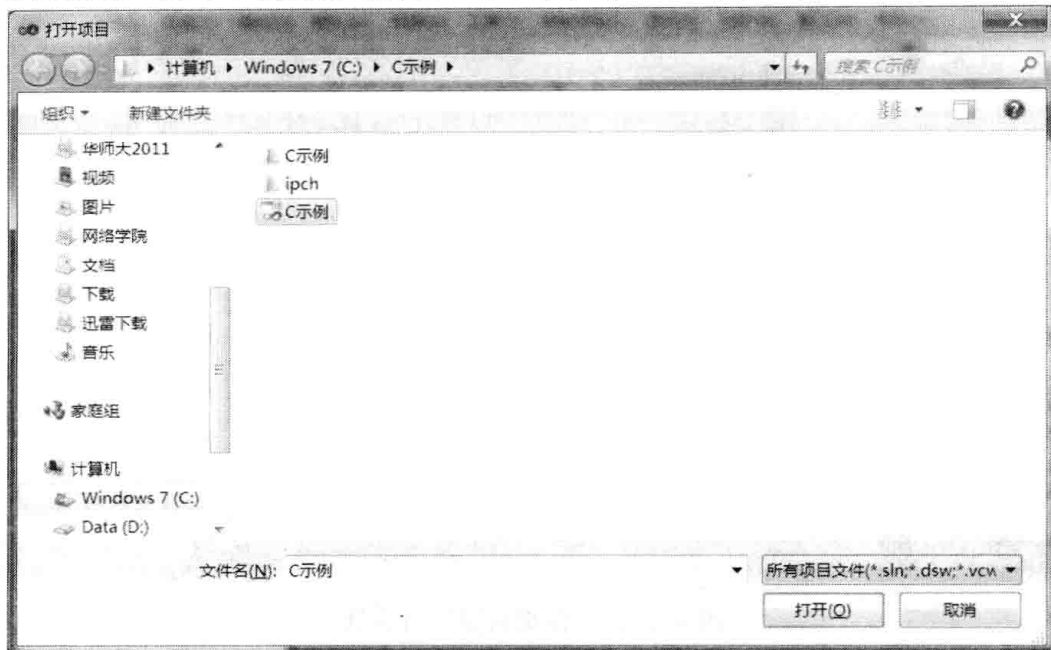


图 4-3-1 “打开项目”对话框

2. 在“打开项目”对话框中, 选择要打开的项目的位置和名称, 如位置在“C 盘根目录下的 C 示例文件夹中”, 项目名称为“C 示例”, 单击“打开”按钮, 返回到“VS2010 主窗口”。在右窗格可以看到有一个“C 示例”项目, 在“源文件”下还可以看到有一个 Example.c 的 C 源文件, 如图 4-3-2 所示。

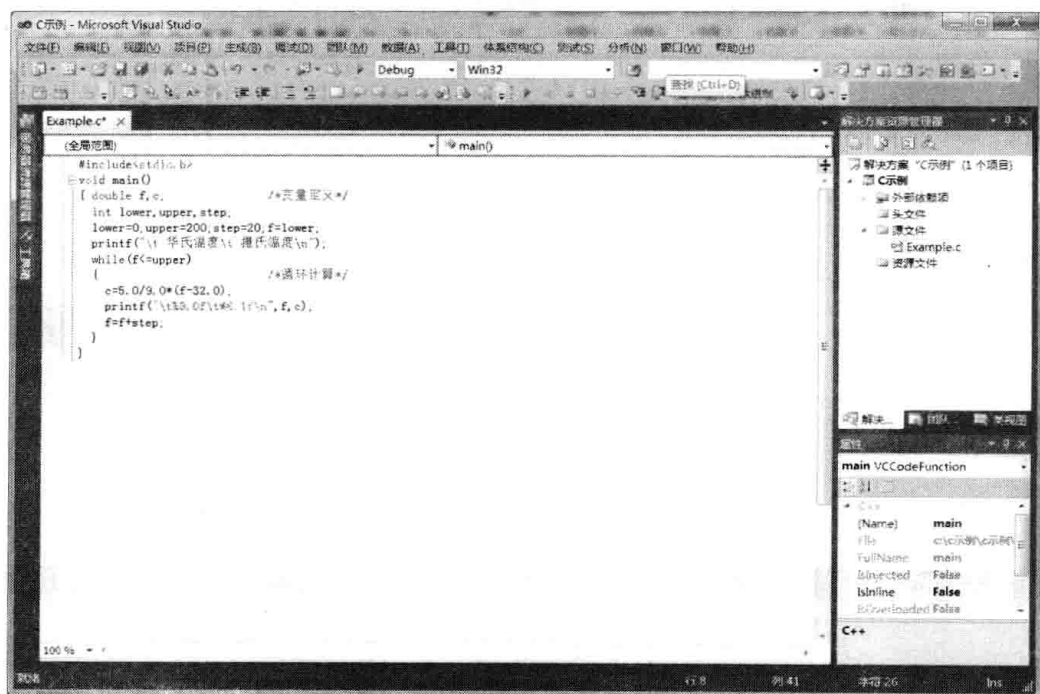


图 4-3-2 在“VS2010 主窗口”中打开的项目及 C 源文件

四、源程序仅编译不运行

当编辑和保存了源程序后，可以对源程序进行编译和运行了。编译的首要目标是检查源程序是否存在语法错误。其次，对于没有语法错误的源程序，产生与源程序相对应的目标文件。

源程序编译的方法如下：

1. 在菜单栏上，选择“生成”→“编译”命令（快捷键【Ctrl+F7】），这时在 VS2010 主窗口下方的“输出”区域显示编译后的信息。若编译成功，在“输出”区域显示“生成成功”信息，如图 4-4-1 所示。



图 4-4-1 编译后显示“生成成功”的 VS2010 主窗口

2. 若编译有错，在“输出”区域显示“生成失败”信息，并告知编译失败的原因，如图 4-4-2 所示。双击提示的错误信息，在“源程序编辑”区域的错误行前会出现一个绿色标记（下例提示在 while 的前面缺少分号），改正后再重新编译，直到“生成成功”为止。

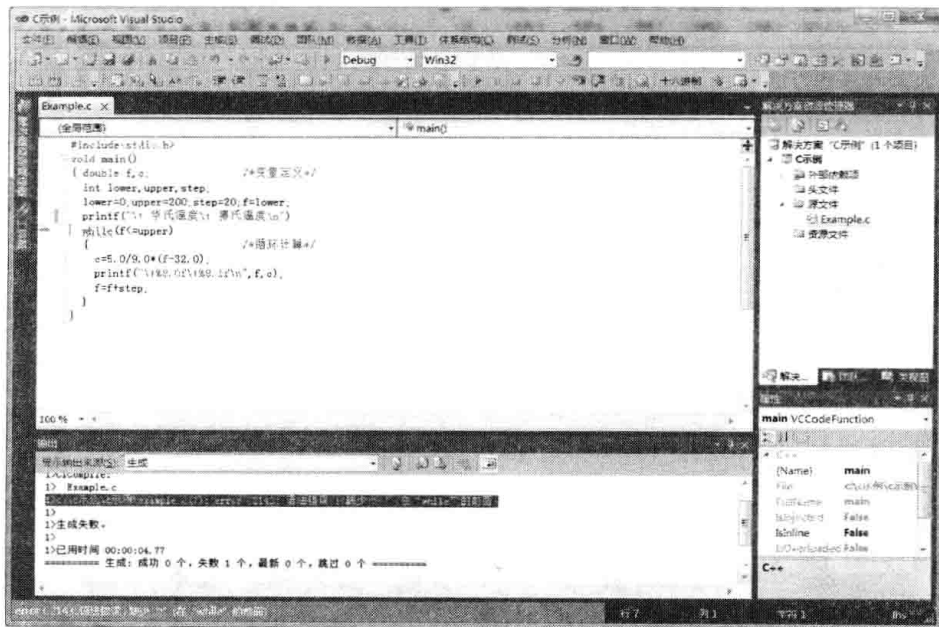


图 4-4-2 编译后显示“生成失败”的 VS2010 主窗口

五、源程序编译与运行

为了快速运行源程序,可以直接选择编译正确后自动运行方法。

源程序编译与运行的方法如下:

1. 在菜单栏上,选择“调试”→“开始执行(不调试)”命令(快捷键【Ctrl+F5】),由于编译成功,自动运行该程序后就显示其结果,如图 4-5-1 所示。

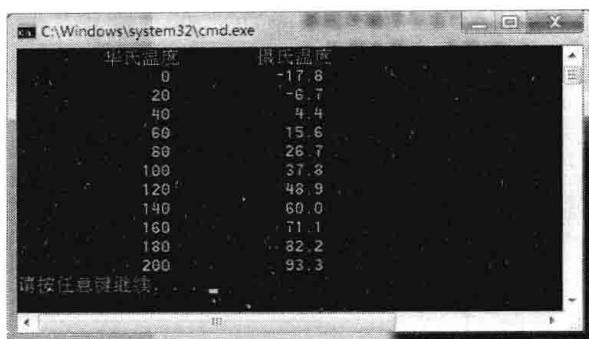


图 4-5-1 显示程序运行后的结果窗口

2. 若编译有错,在“输出”区域显示“生成失败”信息,并告知编译失败的原因,如图 4-4-2 所示,不会继续运行该程序。

六、程序调试

编译系统不能发现程序设计过程中引入的逻辑错误和计算公式等错误。如果程序运行结果有错误,只能通过程序调试去发现错误和改正错误。

为了便于程序调试,VS2010 集成环境中有一个程序调试工具,程序调试方法简述如下:

1. 逐语句调试方法

在菜单栏上,选择“调试”→“逐语句”命令(快捷键【F11】)。“逐语句”调试是指每执行一条语句就会在 VS2010 主窗口底部的“自动窗口”中显示不同变量值的变化过程,如图 4-6-1 所示。

如果不想继续做“逐语句”调试,选择“调试”→“停止调试”命令(快捷键【Shift+F5】)。

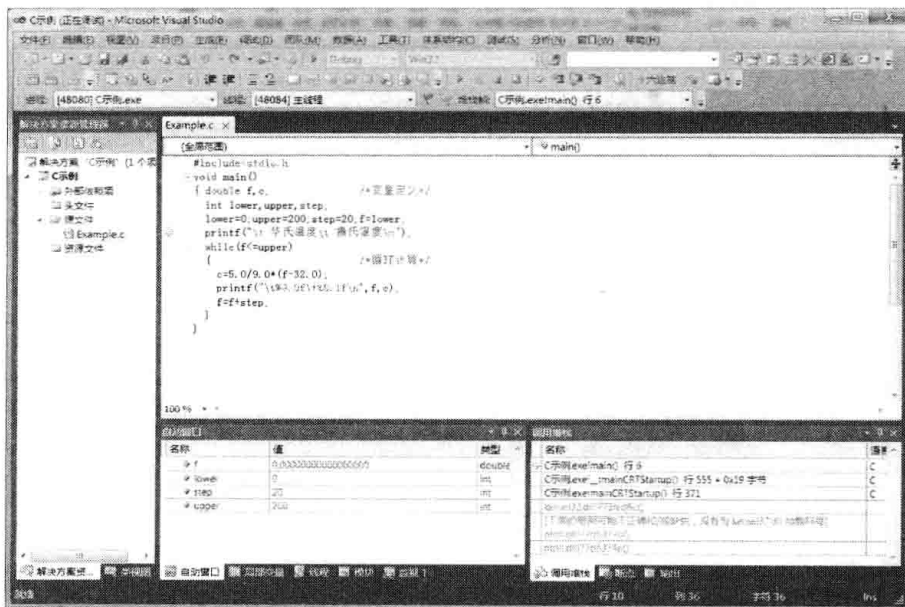


图 4-6-1 “逐语句”调试程序显示变量值的窗口

Images have been losslessly embedded. Information about the original file can be found in PDF attachments. Some stats (more in the PDF attachments):

```
{
  "filename": "MTM2MTgxNTluemlw",
  "filename_decoded": "13618152.zip",
  "filesize": 28236786,
  "md5": "7d2fb55a74ff51fd6460ef0b398e4b30",
  "header_md5": "329c83cf1e2d978e67f6ad985160121a",
  "sha1": "6f5e418b1131f8025fdbaa21cefdd92996ade99c",
  "sha256": "d7b5a6f2cb5dd0e1df748984c7dad722b9f79cddef06c9bab69a69c7a65fcfbf",
  "crc32": 2457113899,
  "zip_password": "",
  "uncompressed_size": 35499906,
  "pdg_dir_name": "C\u2559\u2229\u2564\u2558\u2502\u2560\u2568\u2265\u2554\u03a6\u255d\u255e\u2554\u2567\u2557\u00b7\u2553\u2555\u2561\u255d\u2559\u03b4\u2567\u2591\u2560\u0393\u255c\u0393\u2524\u2261\u2561\u250c3\u2591\u00b5_13618152",
  "pdg_main_pages_found": 165,
  "pdg_main_pages_max": 165,
  "total_pages": 173,
  "total_pixels": 1064614320,
  "pdf_generation_missing_pages": false
}
```