

# 前言

## 内容简介

本书是普通高等教育“十一五”国家级规划教材《C 语言程序设计》的上机学习指导书。其主要针对主教材课后习题和本书实验项目进行了详细解答，并且介绍了 Turbo 2.0 /3.0 及 Visual C++ 6.0 编译系统的使用，对 C 语言程序的编辑、编译、连接、运行和调试等方法进行了详细介绍和指导。全书共分四个部分，主要内容包括：C 语言上机指导、C 语言实验项目、C 语言课程设计、教材习题和实验项目题解。

本书着眼于教材中的重点和难点知识分析以及习题解析，为学习者指出了在 C 语言上机学习过程中需要注意的问题。希望学习者经过本书的指导学习以后，在解决更多的疑难问题上得到指点和帮助。

## 本书特色

1. 本习题指导针对主教材中的习题进行了全面的解答。在提供正确的源程序的基础上，增加了算法分析、结果分析和易错点提示等内容，对于学习者正确地、系统地理解和学习 C 语言很有帮助。

2. 在实验项目中，增加了一项综合性强、应用范围广的课程设计。这项设计不仅可以加深学习者对教材内容的领悟，同时也提高了学习者对 C 语言知识的综合运用能力。除此之外，还能激发学习者的学习热情和兴趣。

3. 本习题指导参考了很多资深程序员的编程经验。在讲解知识的同时，还提供了很多优秀的编程风格和习惯，更能培养学习者实际的软件开发能力。

4. 本习题指导的附录收集了很多编程的常见错误以及编译器的出错信息，使得本书在指导习题的同时，成为了一本 C 语言学习和开发的工具书。学习者通过本习题指导，可以在自身的软件开发能力方面得到全面的提高。

## 鸣谢

本书由电子科技大学黄迪明、许家珩、胡德昆，西南科技大学孙立欧编写。在本书的编写过程中，还得到了杜海涛、张大愚、邹波、张紫微、刘鹏等人的热情帮助，在此表示诚挚的谢意！

由于我们的水平和时间有限，书中难免存在缺点和错误，敬请读者和同行专家指正。

编者

2008 年 5 月于电子科技大学

# 目 录

第 1 章 C语言程序的上机步骤.....	1
第 2 章 Turbo C 2.0/3.0 使用指南.....	4
2.1 Turbo C 简介 .....	4
2.2 Turbo C 2.0 文件简介 .....	4
2.3 Turbo C 2.0 的启动 .....	5
2.4 Turbo C 2.0 集成开发环境的使用 .....	5
2.5 Turbo C 2.0 的配置文件 .....	12
2.6 Turbo C 调试系统 .....	14
2.7 常见的编译错误和程序调试.....	18
2.7.1 常见的错误类型 .....	18
2.7.2 程序调试.....	23
第 3 章 Visual C++环境下运行C程序 .....	26
3.1 启动VC++ .....	26
3.2 新建/打开C 程序文件.....	27
3.3 程序的编辑和保存.....	27
3.4 执行程序 .....	28
3.5 关闭程序工作区.....	30
3.6 命令行参数处理.....	30
3.7 程序调试 .....	31
第 4 章 实验概述 .....	34
4.1 本课程实验的任务.....	34
4.2 本课程实验简介.....	34
4.3 本课程适用专业.....	34
4.4 本课程实验涉及核心知识点.....	34
4.5 本课程实验重点与难点.....	34

<b>第 5 章 实验项目和课程设计 .....</b>	<b>35</b>
5.1 实验项目一 .....	35
5.2 实验项目二 .....	35
5.3 实验项目三 .....	36
5.4 实验项目四 .....	36
5.5 实验项目五 .....	37
5.6 实验项目六 .....	37
5.7 课程设计 .....	37
5.7.1 课程设计一 .....	37
5.7.2 课程设计二 .....	38
5.7.3 课程设计三 .....	38
5.7.4 课程设计四 .....	38
<b>第 6 章 主教材习题参考答案 .....</b>	<b>39</b>
6.1 习题 1 .....	39
6.2 习题 2 .....	42
6.3 习题 3 .....	51
6.4 习题 4 .....	66
6.5 习题 5 .....	83
6.6 习题 6 .....	91
6.7 习题 7 .....	109
6.8 习题 8 .....	117
6.9 习题 9 .....	121
6.10 课程设计解答 .....	132
6.10.1 设计一 .....	132
6.10.2 设计二 .....	133
6.10.3 设计三 .....	135
6.10.4 设计四 .....	137
<b>附    录 .....</b>	<b>139</b>
附录一 C语言常见错误总结 .....	139
一、编程的常见错误 .....	139
二、连接时的常见错误 .....	140
三、运行时的常见错误 .....	141
附录二 C语言编译常见错误信息及处理方法 .....	141

## 第1章 C语言程序的上机步骤

按照C语言语法规则编写的C程序称为C源程序。在主教材中，我们已经了解到，源程序由字母、数字及其他符号（标识符）等构成，在计算机内部用相应的ASCII码表示，并保存为扩展名为“.c”源程序文件（在本书中又被称为源代码）。源程序是无法直接被计算机运行的，因为计算机的CPU只能执行二进制的机器指令。这就需要把源程序的ASCII码先“翻译”成机器指令，然后计算机的CPU才能运行翻译好的程序，也就是ASCII码。源程序“翻译”过程由两个步骤实现：编译与连接。首先对源程序进行编译处理，即把每一条语句用若干条机器指令来实现，以生成由机器指令组成的目标程序。但目标程序还不能马上交给计算机直接运行，因为在源程序中，常用函数（如scanf、printf）运算并不是用户自己编写的，而是直接调用系统函数库中的库函数。因此，必须把“库函数”的处理过程连接到经编译生成的目标程序中，生成可执行程序，并经机器指令的地址重定位，才可由计算机运行，最终得到结果。

C语言程序的调试、运行步骤如图1.1所示。

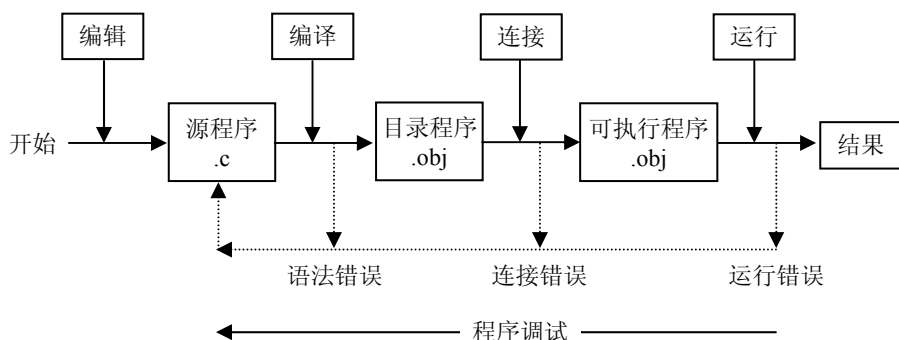


图 1.1 C 程序的运行步骤

在图 1.1 中，实线表示程序从开始编辑到得到结果的过程，虚线表示当某一步骤出现错误时的修改路线。程序在运行时，无论是出现编译错误、连接错误，还是运行结果不正确（源程序中有语法错误或逻辑错误），都需要修改源程序，并对它重新编译、连接和运行，直至将程序调试正确为止。

在积累了一定的程序设计经验后，我们会了解到：除了较简单的情况，一般的程序很难一次就能做到完全正确。在上机过程中，根据出错现象找出错误并改正称为程序调试。我们要在学习程序设计的实践过程中，逐步培养调试程序的能力。它不可能靠几句话讲清楚，要靠自己在上机中不断摸索总结，可以说是一种经验积累。

程序中的错误大致可分为三类：

1. 程序编译时检查出来的语法错误。
2. 连接时出现的错误。
3. 程序执行过程中的错误。

编译错误通常是编程者违反了 C 语言的语法规则,如保留字输入错误、花括号不匹配和语句少分号等。连接错误一般由未定义或未指明要连接的函数,或者函数调用不匹配等因素引起,对系统函数的调用必须要通过“include”说明。

对于编译连接错误,C 语言编译系统会提供出错信息,包括出错位置(行号)、出错提示信息等。编程者可以根据这些信息,找出相应错误所在并修改。有时系统提示了一大串错误信息,并不表示真的有这么多个错误,往往是因为程序中的一两个错误带来的,所以当纠正了几个错误后,应该重新编译连接一次,然后根据最新的出错信息继续纠正,这是程序调试的一个好方法。在本书中将会介绍更多的程序调试方法供学习者参考,但如果要较好地掌握,需要多多进行上机实践,积累调试的经验。

有些程序通过了编译连接,并能够在计算机上运行,但得到的结果和预期的结果不一样,这类错误被称为逻辑错误。这类在程序执行过程中的错误往往难以改正。错误的原因一部分是程序书写错误带来的,例如应该使用变量 x 的地方写成了变量 y,虽然没有语法错误,但意思完全错了;另一部分可能是程序的算法不正确,解题思路不对,得到的结果和预期的结果不一样,例如预期求两个整数的和,在程序中却写为两个整数的差,得到的结果肯定会和预期的不一样;还有一些程序计算结果有时正确,有时不正确,例如求一个输入整数除以 2 的商,如果将这个商定义为 int 型变量,那么在该整数为偶数时正确,奇数时就会错误,这些现象往往是编程时对各种情况考虑不周所致。

解决运行错误的首要步骤就是错误定位,即找到出错的位置和错误的原因,才能予以纠正。通常我们需要先设法确定错误的大致位置,然后通过 C 语言提供的调试工具找出真正的错误。但需要大家注意的是,在本书中,大部分的程序在调试时,调试工具都能直接找到程序的错误,但也有部分比较复杂的程序,当程序执行出错时,调试工具发现的错误未必就一定是程序中的真正错误,这些例子我们会在第三部分中为大家介绍。在将来的实际软件开发中,这样的情况将会出现得更多,所以有经验的程序员往往都认为,寻找程序的错误不能只依靠计算机,也需要我们自己掌握好的方法,在实际的程序调试中积累丰富的经验。以下就将介绍几种比较好的方法:

为了确定错误的大致位置,可以先把程序分成几大块,并在每一块的结束位置,手工计算一个或几个阶段性结果,然后用调试方式运行程序,到每一程序块结束时,检查程序运行的实际结果与手工计算是否一致,通过这些阶段性结果来确定各程序块是否正确。对于出错的程序块,可逐条仔细检查各语句,找出错误所在。如果出错块程序较长,难以一下子找出错误,可以进一步把该块细分成更小的块,按照上述步骤进一步检查。在确定了大致出错位置后,如果无法直接看出错误,可以通过单步运行相关位置的几条语句,逐条检查,这样肯定能找出错误的语句。

当程序出现计算结果有时正确有时不正确的情况时,其原因一般是算法对各种数据处理情况考虑不全面。解决办法最好多选几组典型的输入数据进行测试,除了普通的数据外,还应包含一些边界数据和错误的输入数据。比如确定正常的输入数据范围后,分别以最小值、最

大值、比最小值小的值和比最大值大的值，多方面运行检查自己的程序。能够处理边界数据和不正确的数据，也是程序健壮性的体现。

本书第二、三章分别以 Turbo C 2.0 和 Visual C++ 6.0 为上机平台，对 C 程序编译、连接和调试进行简单介绍。建议一开始学习上机时，把注意力放在程序的编译、连接和运行，以能运行为目标，而把调试部分放到学习了教材第四章内容后再看。只有具有了一定的程序语句量，调试才有作用。

## 第2章 Turbo C 2.0/3.0 使用指南

### 2.1 Turbo C 简介

Turbo C是美国Borland公司的产品，Borland公司是一家专门从事软件开发、研制的大公司。该公司相继推出了Turbo系列软件，如Turbo BASIC，Turbo Pascal，Turbo Prolog，这些软件很受用户欢迎。该公司在1987年首次推出Turbo C 1.0 产品，其中使用了全然一新的集成开发环境，即使用了一系列下拉式菜单，将文本编辑、程序编译、连接以及程序运行一体化，大大方便了程序的开发。1988年，Borland公司又推出Turbo C 1.5版本，增加了图形库和文本窗口函数库等，而Turbo C 2.0则是该公司1989年出版的。Turbo C 2.0在原来集成开发环境的基础上增加了查错功能，并可以在Tiny模式下直接生成“.COM ”（数据、代码和堆栈处在同一64KB内存中）文件，还可对数学协处理器（支持8087/80287/80387等）进行仿真。Borland 公司后来又推出了面向对象的程序软件包Turbo C++，它继承发展了Turbo C 2.0的集成开发环境，并包含了面向对象的基本思想和设计方法。1991年为了适用Microsoft 公司的Windows 3.0 版本，Borland 公司又将Turbo C++ 进行了更新，即Turbo C 的新一代产品Borland C++。

Turbo C 2.0可运行于IBM-PC系列微机，包括XT、AT及IBM 兼容机。此时要求DOS 2.0或更高版本支持，并至少需要448KB的RAM，并且它可在任何彩、单色80列监视器上运行。它支持数学协处理器芯片，也可进行浮点仿真，这将加快程序的执行。

### 2.2 Turbo C 2.0 文件简介

下面对Turbo C 2.0的主要文件进行简单介绍：

INSTALL.EXE 安装程序文件

TC.EXE 集成编译

TCINST.EXE 集成开发环境的配置设置程序

TCHELP.TCH 帮助文件

THELP.COM 读取 TCHHELP.TCH 的驻留程序

README 关于 Turbo C 的信息文件

TCCONFIG.EXE 配置文件转换程序

MAKE.EXE 项目管理工具

TCC.EXE 命令行编译

TLINK.EXE Turbo C 系列连接器

TLIB.EXE Turbo C 系列库管理工具

C0?.OBJ 不同模式启动代码

C?.LIB 不同模式运行库

GRAPHICS.LIB 图形库

EMU.LIB 8087 仿真库

FP87.LIB 8087 库

\*.H Turbo C 头文件

\*.BGI 不同显示器图形驱动程序

\*.C Turbo C 例行程序（源文件）

其中，上面的?分别为：T Tiny（微型模式），S Small（小模式），C Compact（紧凑模式），M Medium（中型模式），L Large（大模式），H Huge（巨大模式）。

## 2.3 Turbo C 2.0 的启动

安装完毕的Turbo C将在C盘根目录下建立一个TC子目录，TC下还建立了两个子目录LIB和INCLUDE，LIB子目录中存放库文件，INCLUDE子目录中存放所有头文件。运行Turbo C 2.0时，只要在TC子目录下键入TC并回车即可进入Turbo C 2.0 集成开发环境，如图2.1所示。

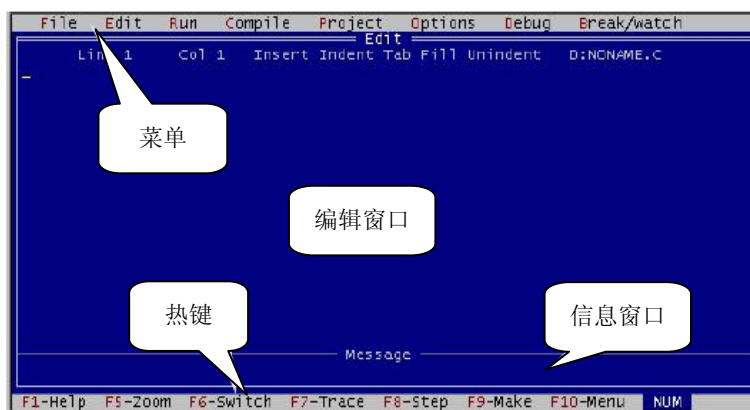


图2.1 Turbo C 2.0集成开发环境

## 2.4 Turbo C 2.0 集成开发环境的使用

进入Turbo C 2.0集成开发环境后，屏幕显示画面如图2.1所示。

其中顶上一行为Turbo C 2.0 主菜单，中间窗口为编辑区，接下来是信息窗口，最底下一行为参考行。这四个窗口构成了Turbo C 2.0的主屏幕，以后的程序编辑、编译、调试以及运行都将在这个主屏幕中进行。下面详细介绍主菜单的内容。

### 1. 主菜单



主菜单 在Turbo C 2.0主屏幕顶上一行，显示下列内容：

**File Edit Run Compile Project Options Debug Break/watch**

除Edit外，其他各项均有子菜单，只要用【Alt】键加上某项中第一个字母（即大写字母），就可进入该项的子菜单中。

#### （1）File（文件）菜单

按【Alt】+【F】键可进入File菜单，该菜单包括以下内容：

- Load（加载）

装入一个文件，可用类似DOS的通配符（如“\*.C”）来进行列表选择，也可装入其他扩展名的文件，只要给出文件名（或只给路径）即可。该项的热键为【F3】，即只要在主菜单中按【F3】键即可进入该项，而不需要先进入File菜单再选此项。

- Pick（选择）

将最近装入编辑窗口的8个文件列成一个表让用户选择，选择后将该程序装入编辑区，并将光标置在上次修改过的地方。其热键为【Alt】+【F3】。

- New（新文件）

说明文件是新的，缺省文件名为NONAME.C，存盘时可改名。

- Save（存盘）

将编辑区中的文件存盘，若文件名是NONAME.C时，将询问是否更改文件名。其热键为【F2】。

- Write to（存盘）

可由用户给出文件名将编辑区中的文件存盘，若该文件已存在，则询问要不要覆盖。

- Directory（目录）

显示目录及目录中的文件，并可由用户选择。

- Change dir（改变目录）

显示当前目录，用户可以改变显示的目录。

- Os shell（暂时退出）

暂时退出Turbo C 2.0到DOS提示符下，此时可以运行DOS命令，若想回到Turbo C 2.0中，只要在DOS状态下键入EXIT即可。

- Quit（退出）

退出Turbo C 2.0，返回到DOS操作系统中，其热键为【Alt】+【X】。

说明：以上各项可用光标键移动色棒进行选择，回车则执行。也可用每一项的第一个大写字母直接选择。若要退到主菜单或从它的下一级菜单列表框退回均可用【Esc】键，Turbo C 2.0所有菜单均采用这种方法进行操作，以下不再说明。

#### （2）Edit（编辑）菜单

按【Alt】+【E】键可进入编辑菜单，若再回车，则光标出现在编辑窗口，此时用户可以进行文本编辑。可用【F1】键获得有关编辑方法的帮助信息。

与编辑有关的功能键如下：

【F1】键：获得Turbo C 2.0编辑命令的帮助信息

【F5】键：扩大编辑窗口到整个屏幕

【F6】键：在编辑窗口与信息窗口之间进行切换

【F10】键：从编辑窗口转到主菜单

编辑命令简介：

【PageUp】键：向前翻页

【PageDn】键：向后翻页

【Home】键：将光标移到所在行的开始

【End】键：将光标移到所在行的结尾

【Ctrl】+【Y】键：删除光标所在的一行

【Ctrl】+【T】键：删除光标所在处的一个词

【Ctrl】+【K】 【B】键：设置块开始

【Ctrl】+【K】 【K】键：设置块结尾

【Ctrl】+【K】 【V】键：块移动

【Ctrl】+【K】 【C】键：块拷贝

【Ctrl】+【K】 【Y】键：块删除

【Ctrl】+【K】 【R】键：读文件

【Ctrl】+【K】 【W】键：存文件

【Ctrl】+【K】 【P】键：块文件打印

【Ctrl】+【F1】键：如果光标所在处为Turbo C 2.0库函数，则获得有关该函数的帮助信息

【Ctrl】+【Q】键：查找Turbo C 2.0双界符的后匹配符

【Ctrl】+【Q】 【】键：查找Turbo C 2.0双界符的前匹配符

说明：

a. Turbo C 2.0的双界符包括以下几种符号：

花括号 { 和 }

尖括号 < 和 >

圆括号 (和 )

方括号 [ 和 ]

注释号 /\* 和 \*/

双引号 "

单引号 '

b. Turbo C 2.0在编辑文件时还有一种功能，就是能够自动缩进，即光标定位和上一个非空字符对齐。在编辑窗口中，【Ctrl】+【O】 【L】键为自动缩进开关的控制键。

(3) Run (运行) 菜单

按【Alt】+【R】键可进入Run菜单，该菜单有以下各项：

● Run (运行程序)

运行由Project/Project name项指定的文件名或当前编辑区的文件。如果上次编译后的源代码未做过修改，则直接运行到下一个断点（没有断点则运行到结束），否则先进行编译、

连接后才运行。其热键为【Ctrl】+【F9】。

- Program reset (程序重启)

中止当前的调试, 释放分给程序的空间。其热键为【Ctrl】+【F2】。

- Go to cursor (运行到光标处)

调试程序时使用, 选择该项可使程序运行到光标所在行。光标所在行必须为一条可执行语句, 否则提示错误。其热键为【F4】。

- Trace into (跟踪进入)

在执行一条调用其他用户定义的子函数时, 若用Trace into项, 则执行长条将跟踪到该子函数内部去执行。其热键为【F7】。

- Step over (单步执行)

执行当前函数的下一条语句, 即使用户函数调用, 执行长条也不会跟踪进函数内部。其热键为【F8】。

- User screen (用户屏幕)

显示程序运行时在屏幕上显示的结果, 其热键为【Alt】+【F5】。

#### (4) Compile (编译) 菜单

按【Alt】+【C】可进入Compile菜单, 该菜单有以下几个内容:

- Compile to OBJ (编译生成目标码)

将一个C源文件编译生成.OBJ目标文件, 同时显示生成的文件名。其热键为【Alt】+【F9】。

- Make EXE file (生成执行文件)

此命令生成一个.EXE的文件, 并显示生成的.EXE文件名。其中.EXE文件名是下面几项之一。

- a. 由 Project/Project name 说明的项目文件名。
- b. 若没有项目文件名, 则是由 Primary C file 说明的源文件。
- c. 若以上两项都没有文件名, 则为当前窗口的文件名。

- Link EXE file (连接生成执行文件)

把当前.OBJ文件及库文件连接在一起生成.EXE文件。

- Build all (建立所有文件)

重新编译项目里的所有文件, 并进行装配生成.EXE文件。该命令不进行过时检查。

(上面的几条命令要进行过时检查, 即如果目前项目里源文件的日期和时间与目标文件相同或更早, 则拒绝对源文件进行编译)。

- Primary C file (主C文件)

当在该项中指定了主文件后, 在以后的编译中, 如没有项目文件名则编译此项中规定的主C文件, 如果编译中有错误, 则将此文件调入编辑窗口, 不管目前窗口中是不是主C文件。

- Get info

获得有关当前路径、源文件名、源文件字节大小、编译中的错误数目和可用空间等信息。

#### (5) Project (项目) 菜单

按【Alt】+【P】键可进入Project菜单, 该菜单包括以下内容:

- Project name (项目名)

项目名具有.PRJ的扩展名,其中包括将要编译、连接的文件名。例如有一个程序由file1.c、file2.c和file3.c组成,要将这3个文件编译装配成一个file.exe的执行文件,可以先建立一个file.prj的项目文件,其内容如下:

```
file1.c
file2.c
file3.c
```

此时将file.prj放入Project name项中,以后进行编译时将自动对项目文件中规定的三个源文件分别进行编译,然后连接成file.exe文件。如果其中有些文件已经编译成.OBJ文件,而又没有修改过,可直接写上.OBJ扩展名。此时将不再编译而只进行连接。

例如:

```
file1.obj
file2.c
file3.c
```

将不对file1.c进行编译,而直接连接。

说明: 当项目文件中的每个文件无扩展名时,均按源文件对待,另外,其中的文件也可以是库文件,但必须写上扩展名.LIB。

- Break make on (中止编译)

由用户选择是否有Warning(警告)、Errors(错误)和Fatal Errors(致命错误)时或Link(连接)之前退出Make编译。

- Auto dependencies (自动依赖)

当开关置为on,编译时将检查源文件与对应的.OBJ文件日期和时间,否则不进行检查。

- Clear project (清除项目文件)

清除Project/Project name中的项目文件名。

- Remove messages (删除信息)

把错误信息从信息窗口中清除掉。

(6) Options (选择菜单)

按【Alt】+【O】键可进入Options菜单,该菜单对初学者来说要谨慎使用。

- Compiler (编译器)

本项选择又有许多子菜单,可以让用户选择硬件配置、存储模型、调试技术、代码优化、对话信息控制和宏定义。这些子菜单如下:

- Model: 共有Tiny、small、medium、compact、large和huge 六种不同模式可由用户选择。
- Define: 打开一个宏定义框,用户可输入宏定义。多重定义可用分号,赋值可用等号。
- Code generation: 它又有许多任选项,这些任选项告诉编译器产生什么样的目标代码。

- ◆ Calling convention: 可选择C或Pascal方式传递参数。
- ◆ Instruction set: 可选择8088/8086或80186/80286指令系列。
- ◆ Floating point: 可选择仿真浮点、数学协处理器浮点或无浮点运算。
- ◆ Default char type: 规定char的类型。
- ◆ Alignment: 规定地址对准原则。
- ◆ Merge duplicate strings: 进行优化用, 将重复的字符串合并在一起。
- ◆ Standard stack frame: 产生一个标准的栈结构。
- ◆ Test stack overflow: 产生一段程序运行时检测堆栈溢出的代码。
- ◆ Line number: 在.OBJ文件中放进行号以供调试时用。
- ◆ OBJ debug information: 在.OBJ文件中产生调试信息。

#### ■ Optimization

- ◆ Optimize for: 选择是对程序小型化还是对程序速度进行优化处理。
- ◆ Use register variable: 用来选择是否允许使用寄存器变量。
- ◆ Register optimization: 尽可能使用寄存器变量以减少过多的取数操作。
- ◆ Jump optimization: 通过去除多余的跳转和调整循环与开关语句的办法以压缩代码。

#### ■ Source

- ◆ Identifier length: 说明标识符有效字符的个数, 默认为32个。
- ◆ Nested comments: 是否允许嵌套注释。
- ◆ ANSI keywords only: 是只允许ANSI关键字还是也允许Turbo C 2.0关键字。

#### ■ Error

- ◆ Error stop after: 多少个错误时停止编译, 默认为25个。
- ◆ Warning stop after: 多少个警告错误时停止编译, 默认为100个。
- ◆ Display warning: 显示警告信息。
- ◆ Portability warning: 移植性警告错误。
- ◆ ANSI Violations: 侵犯了ANSI关键字的警告错误。
- ◆ Common error: 常见的警告错误。
- ◆ Less common error: 少见的警告错误。

#### ■ Names

用于改变段(segment)、组(group)和类(class)的名字, 默认值为CODE、DATA和BSS。

#### ● Linker (连接器)

本菜单设置有关连接的选择项, 它有以下内容:

- Map file menu : 选择是否产生.MAP文件。
- Initialize segments: 是否在连接时初始化没有初始化的段。
- Default libraries: 是否在连接其他编译程序产生的目标文件时去寻找缺省库。
- Graphics library: 是否连接graphics库中的函数。
- Warn duplicate symbols: 当有重复符号时产生警告信息。

- Stack warning: 是否让连接程序产生No stack的警告信息。
- Case-sensitive link: 是否区分大、小写字。

- Environment (环境)

本菜单规定是否对某些文件自动存盘及制表键和屏幕大小的设置。

- Message tracking: 跟踪邮件。
- Current file: 跟踪在编辑窗口中的文件错误。
- All files: 跟踪所有文件错误。
- Off: 不跟踪。
- Keep message: 编译前是否清除Message窗口中的信息。
- Config auto save: 选on时, 在Run、Shell或退出集成开发环境之前, 如果Turbo C 2.0的配置被改过, 则所做的改动将存入配置文件中; 选off时不存。
- Edit auto save: 是否在Run或Shell之前, 自动存储编辑的源文件。
- Backup file: 是否在源文件存盘时产生后备文件 (.BAK文件)。
- Tab size: 设置制表键大小, 默认为8。
- Zoomed windows: 将现行活动窗口放大到整个屏幕, 其热键为【F5】。
- Screen size: 设置屏幕文本大小。

- Directories (路径)

规定编译、连接所需文件的路径, 有下列各项:

- Include directories: 包含文件的路径, 多个子目录用“; ”分开。
- Library directories: 库文件路径, 多个子目录用“; ”分开。
- Output directory: 输出文件 (.OBJ、.EXE、.MAP文件) 的目录。
- Turbo C directory: Turbo C 所在的目录。
- Pick file name: 定义加载的pick文件名, 如不定义则从current pick file中取。
- Arguments (命令行参数): 允许用户使用命令行参数。
- Save options (存储配置): 保存所有选择的编译、连接、调试和项目到配置文件中, 缺省的配置文件为TCCONFIG.TC。
- Retrieve options: 装入一个配置文件到TC中, TC将使用该文件的选择项。

(7) Debug (调试) 菜单

按【Alt】+【D】可选择Debug菜单, 该菜单主要用于查错, 它包括以下内容:

- Evaluate (估值)
  - Expression: 要计算结果的表达式。
  - Result: 显示表达式的计算结果。
  - New value: 赋给新值。
- Call stack: 在Turbo C debugger 时用于检查堆栈情况。
- Find function: 在运行Turbo C debugger时用于显示规定的函数。
- Refresh display: 如果编辑窗口偶然被用户窗口重写了可用此恢复编辑窗口的内容。

### (8) Break/watch (断点及监视表达式)

按【Alt】+【B】键可进入Break/watch菜单，该菜单有以下内容：

- Add watch: 向监视窗口插入一监视表达式。
- Delete watch: 从监视窗口中删除当前的监视表达式。
- Edit watch: 在监视窗口中编辑一个监视表达式。
- Remove all watches: 从监视窗口中删除所有的监视表达式。
- Toggle breakpoint: 对光标所在的行设置或清除断点。
- Clear all breakpoints: 清除所有断点。
- View next breakpoint: 将光标移动到下一个断点处。

上面我们只对TC 中主要的功能进行了介绍，对于其他的操作读者可以自己试验，或参考有关Turbo C 手册。

### (9) 常用快捷键小结

- 【F2】键：保存程序
- 【F3】键：调入程序文件
- 【F4】键：程序运行到光标处暂停
- 【Alt】+【F5】键：查看运行结果
- 【F7】键：单步执行（可进入函数）
- 【F8】键：单步执行（不能进入函数）
- 【Ctrl】+【F2】键：结束程序调试运行
- 【Alt】+【F3】键：调入最近曾经用过的文件
- 【F5】键：放大/缩小窗口
- 【F6】键：窗口切换
- 【Ctrl】+【F7】键：增加查看变量
- 【Ctrl】+【F8】键：把光标所在行设为断点或取消断点
- 【F9】键：编译、连接程序
- 【Alt】+【F9】键：编译程序
- 【Ctrl】+【F9】键：编译、连接、执行程序
- 【F10】键：菜单选择
- 【F1】键：帮助
- 【Alt】+【x】键：退出系统

## 2.5 Turbo C 2.0 的配置文件

所谓配置文件是包含Turbo C 2.0有关信息的文件，其中存有编译、连接的选择和路径等信息。可以用下述方法建立Turbo C 2.0的配置：

1. 建立用户自命名的配置文件。

可以从Options菜单中选择Options/Save options命令，将当前集成开发环境的所有配置存入一个由用户命名的配置文件中。下次启动TC时只要在DOS下键入：

tc/c<用户命名的配置文件名>

就会按这个配置文件中的内容作为Turbo C 2.0的选择。

2. 若设置Options/Environment/Config auto save 为on，则退出集成开发环境时，当前的设置会自动存放到Turbo C 2.0配置文件TCCONFIG.TC中。Turbo C 在启动时会自动寻找这个配置文件。

3. 用TCINST设置Turbo C的有关配置，并将结果存入TC.EXE中。Turbo C 在启动时，若没有找到配置文件，则取TC.EXE中的缺省值。

注：常见问题

1. Turbo C 2.0和Turbo C 3.0编译过程中遇到如下问题：

- a. 找不到 stdio.h、conio.h 等 include 文件；
- b. 出现 cos.obj 无法连接之类的错误。

这些问题是由于没有设置好路径引起的，目前下载的TC2和TC3按安装分类大概有两种版本：一是通过install安装，这类应该已经设置好了路径；二是直接解压后建立TC.EXE的快捷方式，在Windows下双击即可运行（DOS下直接运行TC.EXE），目前国内大多用户采用这种方式，因此使用前请注意路径设置，设置方法如图2.2所示。



图2.2 路径设置

如图2.2所示的设置方法为：

OPTION → DIRECTORIES:

INCLUDE: [TC2/3所在目录]/include

LIB: [TC2/3所在目录]/lib

output: 输出目录请自己设置一个工作目录，以免混在一起

最后还提醒一点：FILES中的Change dir（改变当前目录）中应设置为当前程序所在目录。

2. 命令行参数的输入。

#### ● 菜单方式

在编程中，如果使用命令行参数，可以执行option菜单的Arguments菜单项，输入命令行参数（不包括可执行文件名，如果参数多于一个，则参数间用空格隔开），回车结束输入。



按Esc键隐去菜单，然后按Ctrl+F9键运行程序，参数便能被主函数接收。

- 命令行方式

对源程序进行编译连接生成对应的“.exe”文件后，按【Alt】+【F】进入文件菜单，选择DOS SHELL进入DOS命令提示符状态后，在命令行输入“文件名.exe 参数1 参数2...”执行命令行参数的程序。

## 2.6 Turbo C 调试系统

在开发一个C语言程序的过程中，当完成了程序的编制、编译和连接并在此过程中改正了各种错误以后，程序的运行并不一定正确。一个程序在设计过程中会出现很多错误，包括逻辑错误、变量名错误和运算符错误等。例如，我们希望编制一个程序，它读入了10个整数，求他们中的最大值，并把他们中的每个数的个位数显示出来。这个程序在编制时出现了某些错误，如下所示：

```
1. main()
2. {
3.     int a[10],b[10],max_value,i;
4.     for(i=0;i<10;i++)
5.         scanf("%d",&a[i]);
6.     max_value=32767;
7.     for(i=0;i<10;i++)
8.     {
9.         if(a[i]>max_value)
10.            max_value=a[i];
11.         b[i]=a[i]/10;
12.     }
13.     printf("The max value is%d\n", max_value);
14.     printf("The first bit of each number is:\n");
15.     for(i=0;i<10;i++);
16.         printf("%4d",b[i]);
17. }
```

程序中除了第7行和第15行的语法错误外，还有两个错误：

1. 变量max\_value的初值应为-32 767，这是在微型机上整型变量的最小值，但语句中却赋成了最大值，由于输入的数据中没有大于32 767的数，故每次程序的输出结果都是“max\_value=32 767”。

2. 取一个整数a[i]的个位应该用运算a[i]%10，而程序中却写成了a[i] / 10，因此程序的输出结果是各整数整除10的商，而不是它们的个位。

这些问题在程序的编译和连接中是发现不了的，因为它们并不造成语法或语义错误，但

这些问题会造成程序的执行错误。当程序本身很大时，要发现这些错误并不容易。为此，Turbo C提供了一个调试程序，专门用来对程序进行动态调试，以发现程序中的各种逻辑错误及其他错误。

下面简单地介绍Turbo C的调试程序的使用方法。

首先进入Turbo C的集成开发环境，与程序调试有关的屏幕菜单有：

Debug菜单

Break / watch菜单

Run菜单

先分别介绍这几个菜单。

### 1. Break / watch 菜单

在主菜单下按【Alt】+【B】键，就进入Break / watch菜单，如图2.3所示。

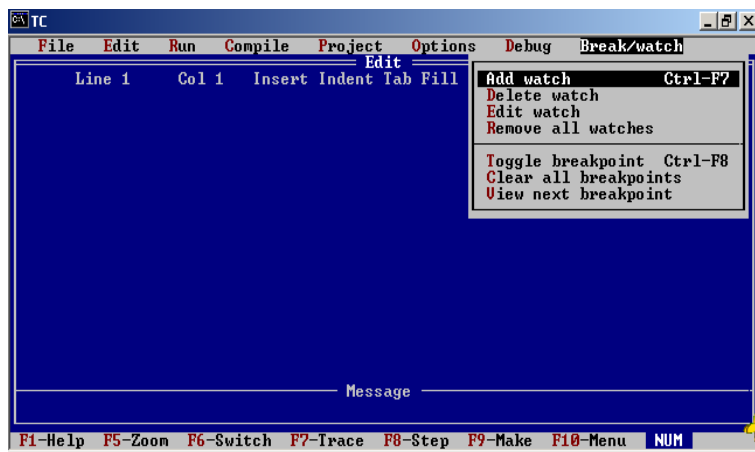


图2.3 Break/watch菜单

这里Watch指的是屏幕的监视窗口，它位于屏幕的编辑窗口的下部。在调试程序时，可以从这个监视窗口中看到你所指定的某些变量或数组元素在程序的执行过程中的变化。选择项Add watch允许键入一个变量名或一个表达式，调试程序把这个变量或表达式送入监视窗口，这样，当程序运行时，就可以看到这个变量或表达式的值的变化情况。

- 选择项Delete watch允许把一个变量或表达式从监视窗口中删除。
- 选择项Remove all watches允许把监视窗口中的所有变量或表达式删除。
- 选择项Edit watch允许修改监视窗口中的表达式，假如上次监视的是变量b[2]，现在希望监视变量b[3]，则可用Edit watch输入b[3]并替换掉原监视窗里的b[2]。
- Breakpoint指的是断点，即程序执行的暂停点。当在程序中设置了断点后，程序在每次执行过程中，一旦碰到了断点，就停止运行。这时，就可以通过监视窗口和其他方式检查程序的运行情况，以便于发现和纠正错误。
- 选择项Toggle breakpoint设置或去除一个断点。为了设置一个断点，首先把光标定在编辑窗口中某个语句所在行，然后选择Toggle breakpoint项，调试程序就在这个语句处设置了一个断点。调试完毕后把光标移到编辑窗口中断点所在行，再一次选

择Toggle breakpoint，就可以去除这个断点。

- 选择项Clear all breakpoints从程序中删除所有断点。
- 选择项View next breakpoint将光标移到程序中的下下一个断点的位置，而不是让程序运行到下一个断点处。

## 2. Debug 菜单

在主菜单下按【Alt】+【D】键，就进入Debug菜单，如图2.4所示。

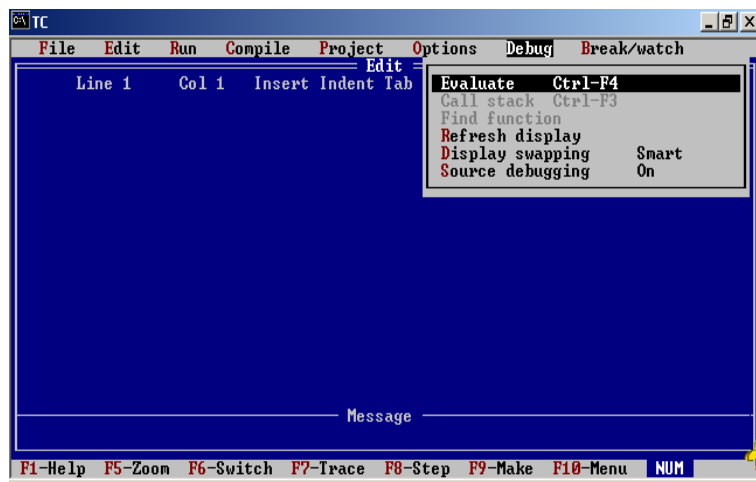


图2.4 Debug 菜单

在调试过程中使用得较多的是Evaluate选择项和Source debugging选择项。

- Source debugging选择项用来接通集成调试程序，当要使用调试程序进行调试以前，一定要把这个选择项设为on状态。
- Evaluate选择项计算变量或表达式的值，并允许用户对其修改。

该命令打开一个新的窗口，该窗口分为三个段：计算段、结果段和新值段。可以在计算段里键入一个变量或一个表达式，调试程序将在结果段中显示这个变量或表达式的当前值。如果需要修改变量或表达式的值，则在新值段中键入新的数值，这个新值将替换掉程序中相应变量的当前值。

该命令还可以用重复表达式来显示一系列数据元素的值。例如对于整型数组a，可用“a[0], 5”显示a[0], a[1], ..., a[4]五个整数的值。

在结果段显示的数据和在新值段修改的数据都可以有各种不同的数据格式，如十进制、十六进制、字符和字符串等。数据格式的说明是在计算段中表达式的后面跟上一个格式控制符。常用的格式控制符有：

- c: 显示字符
- s: 显示字符串
- d: 显示十进制数
- x: 显示十六进制数
- f: 显示浮点数

例如：“a[0], x”表示按十六进制显示变量a[0]的值。

### 3. Run 菜单

在主菜单下按【Alt】+【R】键，就进入Run菜单，如图2.5所示。

其中，选择项Run运行程序，若程序中没有设置断点，则程序一直运行到底；若设有断点，则运行到断点处。

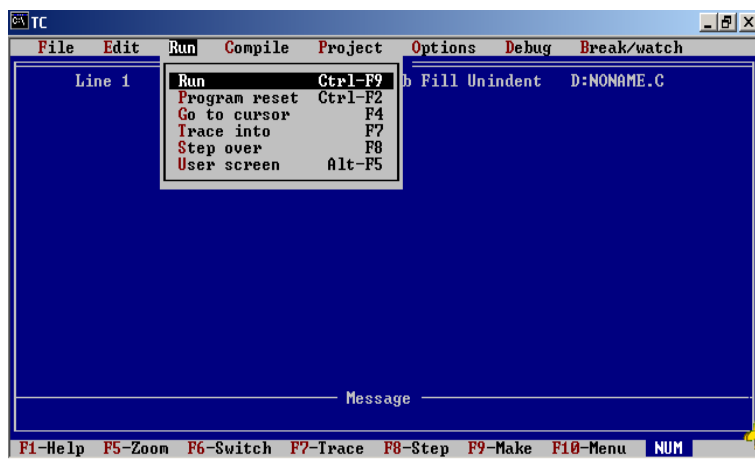


图2.5 运行菜单

- 选择项Program reset终止当前的程序调试。
- 选择项Go to cursor使程序从当前的高亮度长条运行到编辑窗口的光标所在行。
- 选择项Trace into和Step over都用于执行一条语句，但前者可以跟踪进入函数定义内部，而后者只是简单地单步执行一条语句。

下面我们举例说明调试程序的使用方法。如果已经更正了第7行和第15行的语法错误，对于2.6节给出的错误程序：

```

1. main()
2. {
3.     int a[10],b[10],max_value,i;
4.     for(i=0;i<10;i++)
5.         scanf("%d",&a[i]);
6.     max_value=32767;
7.     for(i=0;i<10;i++);
8.     {
9.         if(a[i]>max_value)
10.            max_value=a[i];
11.        b[i]=a[i]/10;
12.    }
13.    printf("The max value is %d\n",max_value);
14.    printf("The first bit of each number is:\n");

```

```
15. for(i=0;i<10;i++);
16.     printf("%4d",b[i]);
17. }
```

当我们运行一次这个程序后发现运行结果是错误的。此时，我们用【Alt】+【B】进入监视和断点菜单，选择Toggle breakpoint，在第7行处设置一个断点，并用Add watch功能把max\_value和b[i]送到监视窗口里。

下一步在运行菜单（【Alt】+【R】）中选择Run命令，令程序开始执行，此时从键盘上输入10个整数：

```
12 34 56 23 78 45 34 2 3 90
```

程序运行到第7个语句时停止，然后在运行菜单(【Alt】+【R】)中选择Step over或直接用F8（单步运行）键，一次一个语句的运行第7行~11行构成的循环体。此时可以在监视窗口中看到变量max\_value和b[i]的值，并立即可以发现变量max\_value的值始终保持不变，而b[i]的值是a[i]整除10以后的商。

发现错误以后，可以直接在编辑窗里进行修改。此时把程序的第6行改为：

```
max_value=-32767;
```

把第10行改为：

```
b[i]=a[i]%10;
```

再按上述方法调试一次程序，证明程序的运行结果是正确的。此时，可以在断点和监视菜单里再一次使用Toggle breakpoint或使用Clear all breakpoints清除断点，并用Remove all watch清除监视窗里的内容。程序的调试到此为止。

## 2.7 常见的编译错误和程序调试

### 2.7.1 常见的错误类型

#### 1. 遗漏花括号

花括号通常用来构成两个或多个语句构成的复合语句，当要对多个语句一起进行处理时，如果忘记了使用花括号构成复合语句，则只有第一个语句能按原意图执行。

例如：

```
while(i<MAX)
    if(score[i]>59)
        printf("NO%2d:%3d",i+1,score[i]);
    i++;
```

该程序段的目的是打印出全部及格学生的代号和分数。但由于遗漏了花括号，只能处理第一个学生的成绩，并且在while语句中不能执行i++语句，循环控制变量不能改变，而进入死循环。上面程序段的正确编程应写为：

```
while(i<MAX)
```

```
{
    if(score[i]>59)
        printf("NO%2d:%3d",i+1,score[i]);
    i++;
}
```

通常在其他复杂的控制语句中，如for、if和do等，编程者可能忘记加花括号或使之成对。当遗漏一个花括号时，可能出现更复杂的情况。通常这与后续的语句有关系。例如：

```
main()
{
    int i,sum=0;
    for(i=1;i<=10;++i)
    {
        sum=sum+i;
        printf("%d%d\n",i,sum);
    }
}
```

这里，for循环语句行的末端缺少一个右花括号。编译程序在处理时，把用来结束main()的花括号，当成用来结束for循环了，认为缺少main()的右花括号。因此有的编译程序可能产生“不期望的文件结束”的错误信息，因此当编好一个程序，特别是用了许多花括号时，应仔细检查一下，看左、右花括号是否配对。

## 2. 遗漏分号或错放分号

例如：start=top

这个语句由于未用分号结束，编译程序可以查出这一种错误，但有时，某些编译程序可能产生一种“误解”的出错信息。因为当编译程序进行扫视时，它必须进行语法检查，确定语句是否用的是C语言结构。如果遗漏分号，编译程序可能把下一行看成是该行的一部分，根据这两行的内容，编译程序就可能产生误解的出错信息。

如果编译时指出某一错误行，但并未检查出问题。通常应该去检查它的前一行，看是否遗漏了分号。遗漏分号可能引起许多出错信息。然而，一旦这个错误被改正，一切就立即正常。

另一个常见的错误是错放了分号。例如：

```
for(n=1;n<=10;n++);
    printf("%d",n*n);
```

for语句的原意是输出自然数1~10的平方值。但是，由于在圆闭括号后错用了一个分号，实际上printf()语句不受for控制。为了更清楚地说明这个问题，上述语句可写成如下形式：

```
for(n=1;n<=10;n++)
;
    printf("%d",n*n);
```

for循环所控制的语句是空语句NULL（只有一个分号），这是什么都不做的循环。由于

空语句在语法上是有效的,编译程序不能检查出错误,但程序运行时却不能得到正确的结果。这类错误也常常出现在其他控制语句中。

### 3. 将等号“=”误用为赋值号“=

这种错误通常出现在if、while和do等控制语句中,当需要进行关系“等于”测试时,常常误用了赋值号。由于编译程序无法查出这种错误,程序运行时可能得到错误结果。例如:

```
main()
{
    int x=0;
    while(x<=100)
    {
        ++x;
        if(x=50)
            printf("x=%d",x);
    }
}
```

该程序原意是当x的值自增到50时,打印出x的值。但由于将等号“=”误用为赋值号“=”,这样,由于x赋了一个非零值,if语句的判定条件总是为真,每次循环都执行printf语句。只有将“=”改为“==”,程序才能正确运行。

### 4. 注释引起的错误

编译程序对注释/\*和\*/之间的内容不编译。在程序中适当地加以注释,不仅使程序易于阅读,而且对编译后程序的大小或运行速度不会产生影响。但使用不小心时,就可能带来麻烦。例如,如果在注释时用了个/\*,但忘记了另一个\*/,编译程序编译时会把/\*后面的内容全部当成注释,而不进行编译直到遇到一个\*/为止。如果遇不到,有的编译程序会显示出错误信息“unexpected end\_of file”,以表示文件在不期望的情况下结束。这一信息为查找这种错误提供了方便。和使用花括号一样,编程者也必须注意注释符号的配对使用。

### 5. 顺序性错误

在C语言中使用加1和减1运算时,必须注意变量和运算符的位置。例如:

```
x=20;    x=20;
y=x++;   y=++x;
```

y的值在上面两种情况下是不相同的。在左边,x的值赋给y后再加1,y为20;在右边,x的值先加1后再赋给y,y为21。

又例如:

```
while(c=getchar()!='*')
    putchar(c);
```

该语句段原意是将从键盘输入的字符,只要不等于自定义的一个结束标志符号'\*,就显示在屏幕上。但忘记了使用圆括号将c=getchar()括起。由于“不等”检测运算符的优先级高于赋值运算符。因此c的值由getchar()函数返回的值是否等于'\*'而定,若不相等,c为1;

若相等，c为0。运行结果达不到预期目的。正确的写法应为：

```
while((c=getchar())!='*')
    putchar(c);
```

## 6. 使用错误的数组下标值

数组下标的有效范围是从0至数组包含的元素数目减1。在使用数组时，应注意正确使用数组上下限下标值。例如有一个包含100个整型数据的数组a[100]，若要求各元素之和，使用了下面的for循环语句。

```
for(i=1;i<=100;++i)
    sum+=a[i];
```

由于循环变量i的初值“1”和终值“100”与数组a的下标初值“0”和终值“99”不同，因此，由此获得的是不正确的运行结果。

在使用字符数组时，编译器自动给字符串末尾加上一个字符串结束标志NULL（空字符），需要增加一个存储单元。例如：字符串“score”需要6个存储单元，而不是5个。

## 7. 调用参数错误

在函数调用时，必须保证传递给函数的实参数与形式参数类型相同。

如果函数的形式参数是浮点型，则必须传递浮点型实参，否则程序不能正确运行。

例如：下面是一个求两个实数之商的程序，但编写有错。

```
main()
{
    int x,y;
    scanf("%d%d",&x,&y);
    printf("%d",div(x,y));
}

float div(a,b)
float a,b;
{
    return(a/b);
}
```

这里，不仅用整型实参数x和y调用div()函数是错误的，而且要div()函数返回整型值也是错误的。

在scanf()函数调用中，必须使用参数的地址，而不是参数的值，因此在非指针变量前必须加单目运算符“&”。例如，下面是一个错误使用scanf()函数的程序。

```
main()
{
    int no;
    char name[10];
```



```
float score;
scanf("%d%s%f",no,name,score);
printf("%d%s%f",no,name,score);
}
```

该程序scanf语句中，no和score两个变量前遗漏了“&”符号，但name表示字符串的首地址，在name前面不用加“&”符号，因此正确的scanf语句应为：

```
scanf("%d%s%f",&no,name,&score);
```

这里出现的错误，是使用scanf语句最容易出现的错误，应引起注意。

### 8. 函数说明的疏漏

只要函数返回值不是整型，就必须在定义函数时说明其返回值类型的情况下，必须在调用该函数的函数内说明被调用函数原型。

要使前面列举的求两个实数的商的程序能正确运行，除了必须修改实参数的类型外，还必须在调用div()函数前，增加说明div()函数返回float数据类型的说明语句。正确的程序应该是：

```
main()
{
    float x,y;
    float div(float,float);
    scanf("%f%f",&x,&y);
    printf("%f",div(x,y));
}

float div(a,b)
float a,b;
{
    return(a/b);
}
```

### 9. 函数重名错误

如果自定义函数与标准库函数同名，那么标准库函数将被自定义函数重新定义，多数编译器会用自定义函数代替标准库中的函数，从而造成一些难以发现的错误。

为了避免产生这类错误。通常不要把自己写的函数与标准库中的函数取相同的名字。

### 10. 使用指针错误

在使用指针前，必须记住对指针进行初始化。如果不给指针赋一个地址，那么就无从知道它指向何处。例如，下面使用指针的方法是错误的：

```
char *c;
*c='a';
```

## 2.7.2 程序调试

只要程序中有错误，就必须对程序进行“调试”，以分离错误并校正错误。调试过程通常包括三个步骤：（1）发现错误；（2）分离错误；（3）校正错误。

### 1. 错误的分类

#### （1）语法错误

最容易发现的错误是语法错误。如前所介绍的那些错误，在多数情况下，编译程序能发现错误并分离出来，并指出出错的源程序行号。校正这类错误的方法是编辑出错行，改错后重新编译。

#### （2）程序出错

这是比较严重的一类错误。由于是“逻辑上的”错误，编译程序无法发现，程序虽能执行，但不是正确的运行结果。在这种情况下，编程者必须根据已知的错误，从程序中查找并分离出错误来源。通常分离错误是调试中的一个关键步骤。

### 2. 查错和排错

正如前面指出的编译可以找出语法和语义错误，但它不能查出程序是否编得恰当以及算法是否正确的错误。

当程序可以执行，但得不到期望的结果时，为了查出错误，程序员就必须从头到尾仔细地对整个程序进行检查。如果算法是正确的，那么查找错误的过程可以先使用一组检查数据，把已知的数据送入程序，并且把程序“划小”进行处理，直到分离出错误来源为止。程序的划小可以通过在程序中放入若干打印语句printf()，通过检查一些指定变量的值或中间结果，就可能把出错的原因分离出来。

为了避免潜在错误，对程序应使用极限值进行测试。如果有错，根据产生错误的数据源，应查找程序中与数据有关的部分，确定出错原因。

### 3. 调试的一般原则

编程和调试方法各式各样，因人而异。但事实上，人们在实践中已找到了一些具有较多优点的方法。在调试方面，逐步检验编程模块被认为是费用最低、时间最省的方法。这种方法的步骤是，在程序开发过程中，先建立一个基础的工作程序块，当新的程序块增添到这个单元，就对它进行检验和调试。使用这种方法，编程者很容易发现错误，因为错误很有可能出现在新增添的程序部分。虽然，采用这种方法，程序的编制表面上看起来似乎慢了些，但逐步检验能保证程序在研制过程中每个程序模块的质量。只要程序到了能运行的程度就应试运行一下，并对已完成的部分全面试验一遍。以后，在程序中每增加一个模块，就将新加入的模块同已能运行的部分一起调试一次。这样做可以保证所有的故障都集中在很小的范围内。

### 4. 调试过程

TC 提供了必要的调试手段和工具，下面按照使用过程予以介绍。

#### （1）让程序执行到中途暂停以便观察阶段性结果：

方法一：使程序执行到光标所在的那一行暂停。

①把光标移动到需暂停的行上；

②按【F4】键或执行菜单Run 中的“Go to Cursor”操作。当程序执行到该行将会暂停。如果把光标移动到后面的某个位置，再按【F4】键，程序将从当前的暂停点继续执行到新的光标位置，第二次暂停。

方法二：把光标所在的那一行设置成断点，然后按【Ctrl】+【F9】键执行，当程序执行到该行将会暂停。设置断点的步骤为：

①把光标移动到需暂停的行上；

②按【Ctrl】+【F8】键或执行菜单“Break/watch ”中的“Toggle breakpoint ”操作。

不管是通过光标位置还是断点设置，其所在的程序行必须是程序执行的必经之路，亦即不应该是分支结构中的语句，因为该语句在程序执行中受到条件判断的限制，有可能因条件的不满足而不被执行。这时程序将一直执行到结束位置或下一个断点位置。

## (2) 设置需观察的结果变量

按照上面的操作，可使程序执行到指定位置时暂停，其目的是为了查看有关的中间结果。按【Ctrl】+【F7】键或菜单“Break/watch ”中的“Add watch ”操作，屏幕上将会弹出小窗口供输入查看变量，如图2.6所示，我们输入了变量i 进行查看。



图2.6 输入查看变量

对于图2.6中的例子，我们先把光标移动到第五行，然后按【F4】键执行，程序到第五行暂停，如图2.7 所示，查看（Watch）窗口中就会显示查看变量i 的当前值。绿色光条表示当前将被执行的程序位置（或暂停位置）。

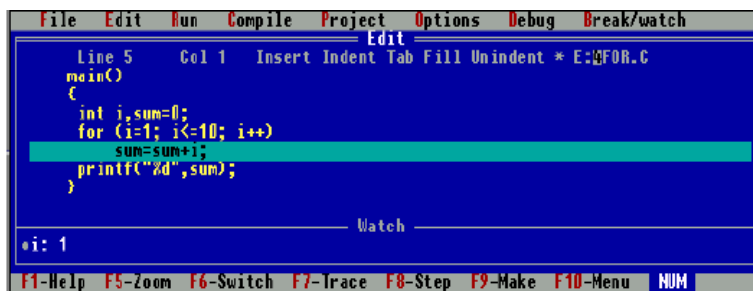


图2.7 查看中间结果

多次使用【Ctrl】+【F8】键可增加多个新的查看变量。如果想改变查看变量的名字或删除查看变量，可以按【F6】键，使查看窗口成为操作窗口，然后按回车键【Enter】可以改变查看变量，按删除键【Delete】可以删除查看变量。这些菜单功能分别在“Break/watch”中。

### (3) 单步执行

当程序执行到某个位置时发现结果已经不正确了,说明在此之前肯定有错误存在。如果能确定一小段程序可能有错,先按上面步骤暂停在该小段程序的头一行,再输入若干个查看变量,然后单步执行,即一次执行一行语句,逐行检查下来,看看到底是哪一行造成结果出现错误,从而能确定错误的语句并予以纠正。

单步执行按【F8】键或执行菜单Run 中的“Step over”操作。如果遇到自定义函数调用,想进入函数进行单步执行,可按【F7】键或执行菜单Run 中的“Trace into”操作。对不是函数调用的语句来说,【F7】键与【F8】键作用相同。但一般对系统函数不要使用【F7】键。

### (4) 断点的使用

使用断点也可以使程序暂停。但一旦设置了断点,不管你是否还需要调试程序,每次执行程序都会在断点上暂停。因此调试结束后应取消所定义的断点。方法是先把光标定位在断点所在行,再按【Ctrl】+【F8】键或执行菜单“Break/watch”中的“Toggle breakpoint”操作,该操作是一个开关,第一次按是设置,第二次按是取消设置。被设置成断点的行将呈红色背景。如果有多个断点想一下全部取消,可执行菜单“Break/watch”中的“Clear all breakpoints”操作。

断点通常用于调试较长的程序,可以避免使用【F4】键(运行程序到光标处暂停)功能,因为【F4】功能,经常需要把光标定位到不同的地方,而对于长度为上百行的程序,要寻找某位置并不太方便。

如果一个程序设置了多个断点,按一次【Ctrl】+【F9】键会暂停在第一个断点,再按一次【Ctrl】+【F9】键会继续执行到第二个断点暂停,依次执行下去。

### (5) 结束调试

TC 中通过“结束程序运行”(Program reset)来结束程序调试。按【Ctrl】+【F2】键或执行菜单Run中的“Program reset”操作实现。

### (6) 循环的调试实例

下面用单步执行功能来看一下for 语句的执行流程,如图2.8所示。程序中把for(;;)分成三行,以便观察执行流程。先把光标移动到第四行,然后按【F4】键或执行菜单Run 中的“Go to Cursor”操作,按【Ctrl】+【F7】键输入查看变量i,由于此时i 未赋过值,所以显示的是一个随机数。再连续按【F8】键单步执行,可以观察绿色光条的位置变动和查看变量i 的变化,绿色光条的位置变动就是程序执行的过程,可以充分体会到for 语句的执行流程。

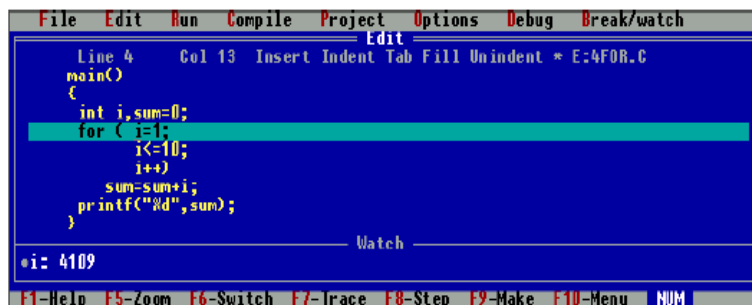


图2.8 循环调试实例

通过这个例子,读者可以举一反三调试自己的程序了。

## 第3章 Visual C++环境下运行C程序

Visual C++ 6.0 (以下简称VC++) 是微软公司开发的面向Windows编程的重要开发工具, 属于Visual Studio集成开发套件中一个重要的部分。相对于传统的开发环境, Visual C++能大大缩短开发周期, 降低软件的开发费用, 所以在实际的程序设计开发中, 使用非常广泛。加上Visual C++比TC的可视化更加优越, 操作更加方便, 调试方法更加多样和灵活, 使得Visual C++同样成为我们学习C/C++语言的一种重要的程序设计工具。虽然Visual C++主要是开发语言为C++, 但该工具同样可以用C语言进行程序设计, 下面我们就对Visual C++进行详细的使用介绍。

### 3.1 启动 VC++

VC++是一个庞大的语言集成工具, 经安装后将占用几百兆磁盘空间。另外, 使用者编写的程序经过保存后还会占用一部分磁盘空间。

首先, 打开Visual C++ 6.0。从“开始”→“程序”→“Microsoft Visual Studio 6.0”→“Microsoft Visual C++ 6.0”, 可启动VC++, 或者直接双击桌面上的图标“Microsoft Visual C++ 6.0”, 也可以启动VC++, 屏幕上将显示, 如图3.1 所示的窗口。

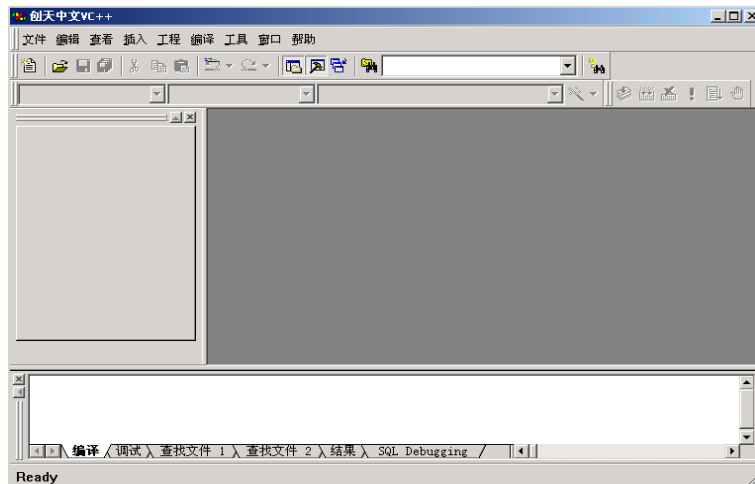


图3.1 VC++界面窗口

整个窗口基本可以分为“菜单栏”、“工具栏”、“编辑区”、“工作区”和“输出窗口”, 如图3.2所示。各个部分的基本功能如下:

菜单栏: 包含了Visual C++ 6.0所有的功能, 并将功能分类在不同的菜单。用户可以根

据需要，在菜单中直接使用Visual C++ 6.0提供的功能。

工具栏：提供了Visual C++ 6.0最常用的一些功能，如“保存”、“打开”和“复制”等。用户可以直接在界面上使用。

编辑区：进行程序编辑的主要区域，在新建一个C source file后，用户可以在该区域进行程序的编辑和修改等操作。

工作区：主要进行软件项目工程的管理。

输出窗口：主要用于显示程序调试或运行的结果、具体的错误信息、警告信息和详细的调试信息。

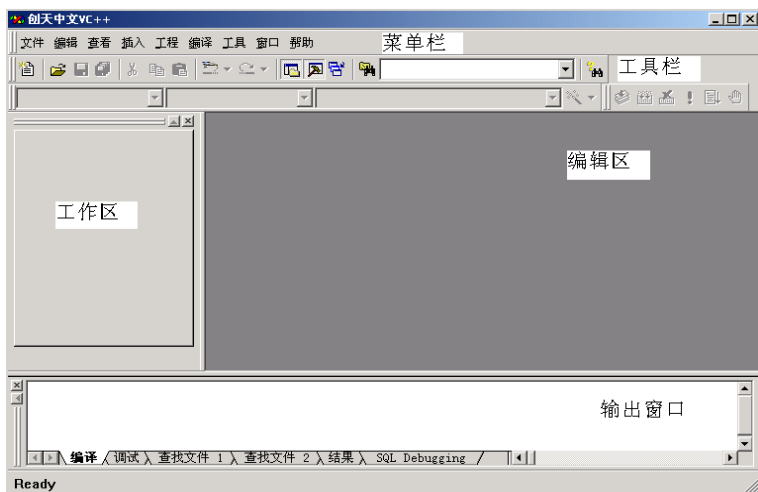


图3.2 VC++窗口的功能介绍

## 3.2 新建/打开C程序文件

在菜单栏中，选择“文件”菜单的“新建”菜单项，单击如图3.3所示的“文件”标签，选中“C++ Source File”，按“确定”。然后就可以在主界面的编辑区中开始编辑程序。另外，在图3.3中，右边的“文件”栏可以输入文件名，“C目录”栏可以设定文件保存的路径和地址。

如果程序已经输入过，可选择“文件”菜单的“打开”菜单项，并在查找范围中找到正确的文件夹，选择调入需要编辑的C源代码程序文件。打开后，在主界面的编辑区内，将显示这个程序，用户可以进行直接的编辑。

## 3.3 程序的编辑和保存

在打开的VC++界面上，可直接在编辑区输入程序，由于完全是Windows界面，输入及修改可借助鼠标和菜单进行，十分方便。初学者可以参考本书教材部分第一章的部分简单程序进行编辑，借此熟悉整个的编辑环境。当输入结束后，应该保存文件，选择“文件”菜单的“保存”菜单项进行保存。注意在程序保存时，对于本书中的C语言程序应指定扩展名“.C”，

否则系统将按默认的C++扩展名“.CPP”保存。如果不保存，在程序的执行阶段，系统同样会要求用户保存程序，方法同上。如图3.4 所示。

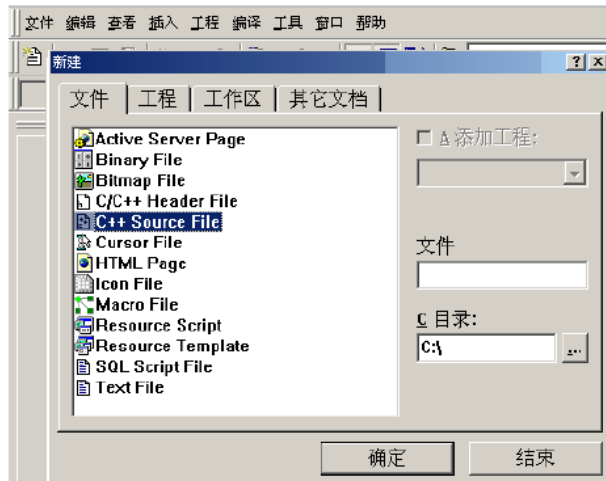


图3.3 新建文件

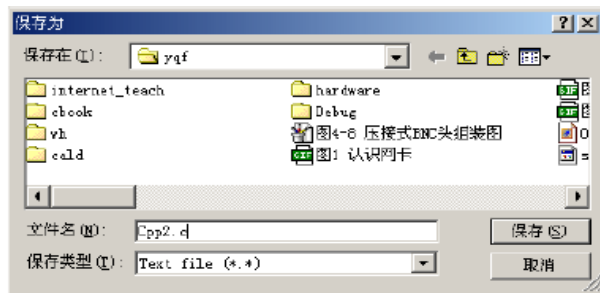


图3.4 指定保存文件名和扩展名

### 3.4 执行程序

首先我们编辑一个程序，在屏幕上显示“hello”，代码如图3.6所示。程序编辑完成后需要生成可执行文件，可使用VC++“编译”菜单中的“构件”菜单项，如图3.5 所示，也可使用快捷键【F7】。在编译连接过程中VC++将保存该新输入的程序，并生成一个同名的工作区。保存文件时须填入文件名，如“1-1.C”。假如不指定扩展名“.C”，VC++会把扩展名定义为“.CPP”，即C++程序。



图3.5 编译连接菜单

如果程序没有错误，将显示如图3.6所示信息窗口中的内容：

0 error(s) 0 warning(s)

表示没有任何错误和警告。需要初学者注意的是，程序有时会出现几个警告性信息（warnings），警告不影响程序执行，只有致命性错误（errors）才影响。

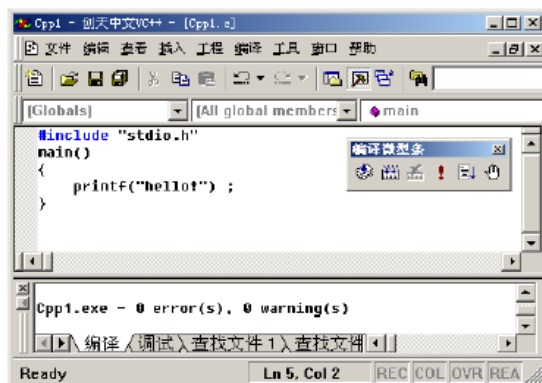


图3.6 编译连接正确

假如有致命性错误（errors），如图3.7所示，双击输出窗口某行出错信息，程序编辑区中会用蓝色箭头指示对应出错位置。根据信息窗口的提示分别予以纠正。在程序修改后，再进行编译，如果还有错，反复调试编译直到正确，最后用“编译”菜单中的“执行”菜单项（或快捷键【Ctrl】+【F5】）执行程序。

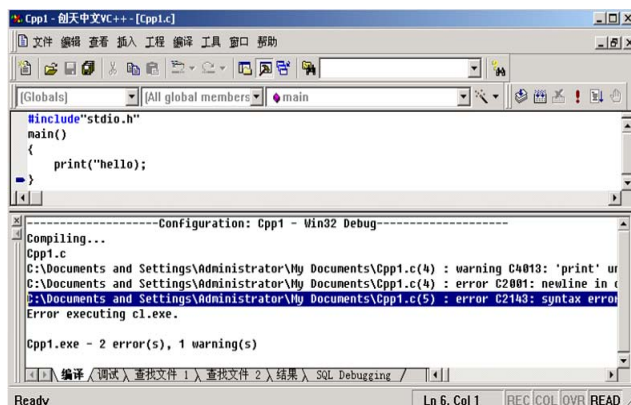


图 3.7 编译连接出错

当运行C程序后，VC++将自动弹出数据输入输出窗口，如图3.8所示。按任意键将关闭该窗口（Press any key to continue）。

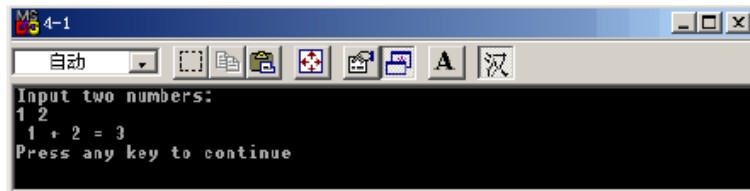


图 3.8 数据输入输出窗口

对于编译连接执行操作，VC++还提供了一组快捷工具按钮（编译微型条）。该工具组一



一般在 VC++界面的右上方，但可以拖动，放在用户指定的位置，如图 3.9 所示。



图 3.9 编译连接执行快捷工具按钮组

### 3.5 关闭程序工作区

在 VC++环境中，当一个程序编译连接后，系统会自动产生相应的工作区，以完成程序的运行和调试。若想执行第二个程序，必须关闭前一个程序的工作区，然后通过新的编译连接，产生第二个程序的工作区。否则的话运行的将一直是前一个程序。有的初学者在编辑了多个程序后，发现运行的结果还是第一个程序，就是这个原因造成的。“文件”菜单提供关闭程序工作区功能，如图 3.10 所示，执行“关闭工作区”菜单功能，然后在如图 3.11 所示对话框中选择“否”，此时会关闭左边窗口中的工作区，但还保留着编辑区中的源程序。如果选择“是”，将同时关闭源程序窗口。



图 3.10 关闭程序工作区

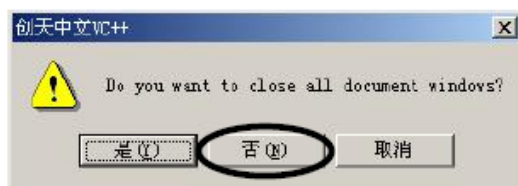


图 3.11 关闭工作区对话框

### 3.6 命令行参数处理

VC++是一个基于窗口操作的 C++系统，没有提供命令行参数功能。我们需要在 Windows 的“MS-DOS 方式”窗口里以命令方式实现。具体步骤参考如下：

- (1) 正确编译连接，生成可执行程序；

- (2) 通过“我的电脑”或“资源管理器”找到所运行的C源程序（设为a.c）；
- (3) 进入debug文件夹（它包含a.c程序的可执行文件a.exe）；
- (4) 执行“开始”菜单的“运行”，填入command，然后“确定”；
- (5) 在打开的“MS-DOS方式”窗口中输入：a 参数1 参数2…，带参数运行程序。

### 3.7 程序调试

VC++是一个完全基于Windows的系统，所以它的调试过程通过鼠标比较容易进行。

(1) 程序执行到中途暂停以便观察阶段性结果：

方法一：使程序执行到需要的那一行暂停。

①在需暂停的行上单击鼠标，定位光标；

②如图3.12所示，分别点击菜单“编译”→“开始调试”→“Run to Cursor”，或按【Ctrl】+【F10】键，程序将执行到光标所在行会暂停。如果把光标移动到后面的某个位置，再按【Ctrl】+【F10】键，程序将从当前的暂停点继续执行到新的光标位置，第二次暂停。

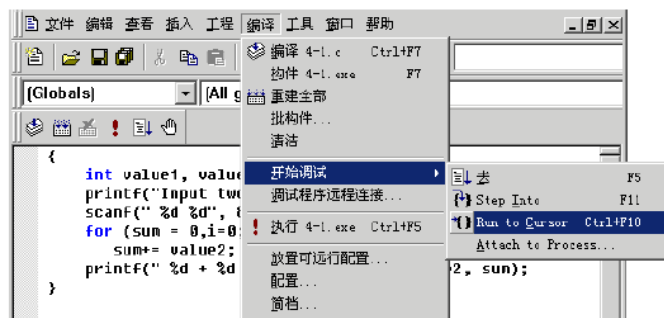


图 3.12 执行到光标所在行暂停

方法二：在需暂停的行上设置断点；

①在需设置断点的行上单击鼠标，定位光标；

②按“编译微型条”中最右面的按钮，如图3.13所示，或按【F9】键。



图 3.13 设置断点

被设置了断点的行前面会有一个红色圆点标志。需要注意的是，不管是通过光标位置还是断点设置，其所在的程序行必须是程序执行的必经之路，即不应该是分支结构中的语句，因为该语句在程序执行中受到条件判断的限制，有可能因条件的不满足而不被执行。这时程序将一直执行到结束或下一个断点为止。

(2) 设置需观察的结果变量

按照上面的操作，使程序执行到指定位置时暂停，目的是为了查看有关的中间结果。在图3.14中，左下角窗口中系统自动显示了有关变量的值，其中value1和value2的值分别是

3、4，而变量 i、sum 的值是不正确的，因为他们还未被赋值。图中左侧的箭头表示当前程序暂停的位置。如果还想增加观察变量，可在图中右下角的“Name”框中填入相应变量名。

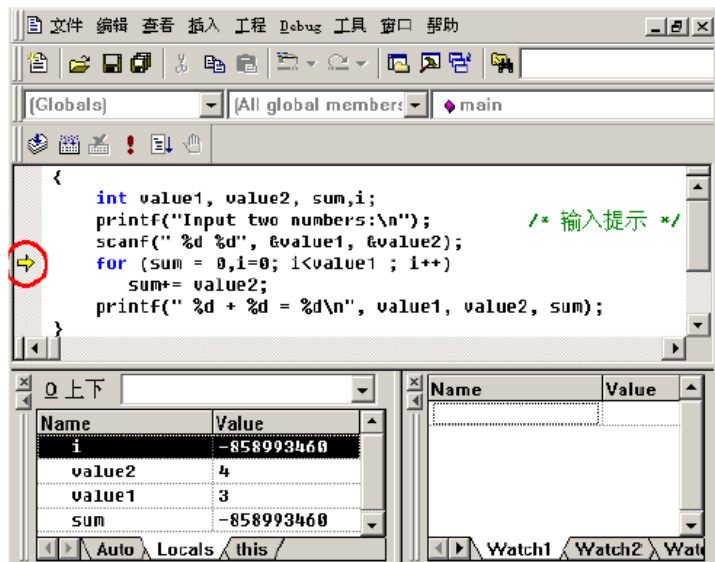


图 3.14 观察结果变量

### (3) 单步执行

当程序执行到某个位置时发现结果已经不正确了，说明在此之前肯定有错误存在。如果能确定一小段程序可能有错，先按上面步骤暂停在该小段程序的头一行，再输入若干个查看变量，然后单步执行，即一次执行一行语句，逐行检查下来，看看到底是哪一行造成结果出现错误，从而能确定错误的语句并予以纠正。单步执行按“调试条”中“Step Over”按钮或【F10】键，如图 3.15 所示。如果遇到自定义函数调用，想进入函数进行单步执行，可按“Step Into”按钮或【F11】键。当想结束函数的单步执行，可按“Step Out”按钮或【Shift】+【F11】键。对不是函数调用的语句来说，【F11】键与【F10】键作用相同。但一般对系统函数不要使用【F11】键。

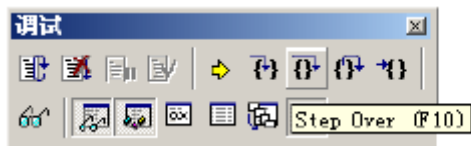


图 3.15 单步调试

### (4) 断点的使用

使用断点也可以使程序暂停。但一旦设置了断点，不管你是否还需要调试程序，每次执行程序都会在断点上暂停。因此调试结束后应取消所定义的断点。方法是先把光标定位在断点所在行，再按“编译微型条”中最右面的按钮或【F9】键，该操作是一个开关，按一次是设置，按两次是取消设置。如果有多个断点想全部取消，可执行“编辑”菜单中的“断点”菜单项，屏幕上会显示“Breakpoints”窗口，如图 3.16 所示，窗口下方列出了所有断点，按“Remove All”按钮，将取消所有断点。断点通常用于调试较长的程序，可以避免使用“Run to Cursor”（运行程序到光标处暂停）或【Ctrl】+【F10】功能，因为该功能经常要

把光标定位到不同的地方，而对于长度为上百行的程序，要寻找某位置并不太方便。如果一个程序设置了多个断点，按一次执行键【Ctrl】+【F5】会暂停在第一个断点，再按一次【Ctrl】+【F5】键会继续执行到第二个断点暂停，依次执行下去。

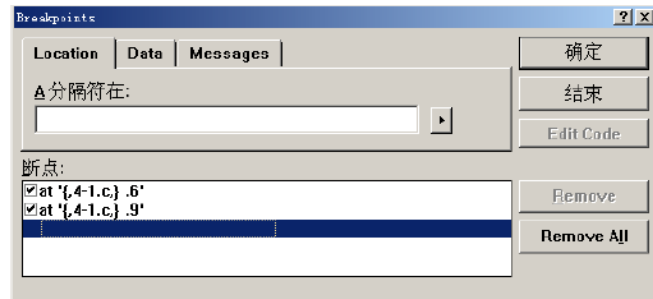


图 3.16 取消所有断点

#### (5) 停止调试

使用“Debug”菜单的“Stop Debugging”菜单项，或【Shift】+【F5】键可以结束调试，从而回到正常的运行状态。

上面我们只对 Visual C++中主要的功能进行了介绍，对于其他的操作读者可以自己试验，或参考有关 Visual C++手册。

## 第 4 章 实验概述

本课程是工科类专业基础课程之一,通过本课程的实验应使学生了解计算机语言的功能和适用领域,具备程序设计的知识和能力。学生应在每次实验课之前完成题目的编程工作,熟练地掌握 C 语言。

### 4.1 本课程实验的任务

- ◆ 熟悉程序开发的全过程;
- ◆ 熟悉流程图的画法及常用算法的思想和实现;
- ◆ 熟练掌握 TC 和 VC 集成编译环境;
- ◆ 熟练掌握使用 C 语言进行程序设计的方法和技术;
- ◆ 熟练掌握程序设计风格和调试方法,培养学生养成良好的编程习惯和较强的程序调试能力,为今后实际从事软件开发工作打下良好的基础。

### 4.2 本课程实验简介

本课程实验与主教材配套,是主教材的补充和提高。共设计了六次实验和四次课程设计,由浅入深,从简单实例入手,让学生了解程序开发的全过程,熟悉程序设计风格和调试方法,指导学生养成良好的编程习惯。

### 4.3 本课程适用专业

工科类专业。

### 4.4 本课程实验涉及核心知识点

结构化程序的三种基本结构, C 语言的函数、指针、数组、结构和文件的处理。

### 4.5 本课程实验重点与难点

循环嵌套、指针、函数的参数传递、变量的存储类型、结构与文件。

## 第5章 实验项目和课程设计

### 5.1 实验项目一

1. 实验项目名称：简单程序设计

2. 实验项目的目的和任务：

掌握在 VC++/TC 环境下编辑、编译、连接和运行 C 程序的方法和过程。

通过运行简单的 C 程序，掌握 C 语言源程序的特点：C 程序的一般构造形式；C 语言中各种常量的表示形式及变量的定义；C 语言中各种运算符的作用、优先级和结合性；能熟练运用各种表达式；标准输入输出函数的一般使用方法。

3. 实验内容：

熟悉 VC++/TC 2.0 开发环境（参考本书第 1、2 章内容）；

教材第 1 章习题 1.9；

教材第 1 章习题 1.10；

教材第 2 章习题 2.6；

教材第 2 章习题 2.9；

教材第 2 章习题 2.22；

教材第 2 章习题 2.23。

4. 学时数：4 学时。

### 5.2 实验项目二

1. 实验项目名称：控制语句的应用。

2. 实验项目的目的和任务：

掌握顺序、分支和循环结构各种语句的一般功能和执行过程；

学会应用各种语句进行编程；

掌握流程图的一般表示方法；

掌握结构化程序设计的基本方法。

3. 实验内容：

教材第 3 章习题 3.4；

教材第 3 章习题 3.7；

教材第 3 章习题 3.14；

教材第 3 章习题 3.16；

教材第 3 章习题 3.17。

4. 学时数：4 学时。

### 5.3 实验项目三

1. 实验项目名称：数组和结构的应用。

2. 实验项目的目的和任务：

掌握数组的定义及其应用；

掌握结构类型的定义及结构变量的定义、引用和初始化；

掌握结构数组的应用；

了解数组和结构的综合应用。

3. 实验内容：

教材第 4 章习题 4.3；

教材第 4 章习题 4.5；

教材第 4 章习题 4.7；

教材第 4 章习题 4.8；

教材第 4 章习题 4.10（选做）。

4. 学时数：4 学时。

### 5.4 实验项目四

1. 实验项目名称：指针和函数的应用。

2. 实验项目的目的和任务：

掌握指针变量的定义和使用方式；

掌握指针与数组的密切关系；

掌握使用指针数组的程序设计方法；

掌握 C 语言中定义函数的方法；

掌握函数传值和传址调用的方法。

3. 上机实验内容：

教材第 5 章习题 5.6；

教材第 5 章习题 5.9；

教材第 5 章习题 5.11；

教材第 6 章习题 6.2；

教材第 6 章习题 6.5；

教材第 6 章习题 6.10。

4. 学时数：4 学时。

## 5.5 实验项目五

1. 实验项目名称：联合、枚举和函数。
2. 实验项目的目的和任务：  
掌握函数与结构的应用；  
学会使用函数解决较难的问题；  
掌握联合、位域和枚举的定义和使用。
3. 上机实验内容：  
教材第6章习题 6.15；  
教材第6章习题 6.17；  
教材第7章习题 7.2；  
教材第8章习题 8.6。
4. 学时数：4 学时。

## 5.6 实验项目六

1. 实验项目名称：输入、输出和文件操作。
2. 实验项目的目的和任务：  
学会使用实现文件打开/关闭、读/写等基本操作的函数；  
掌握各种输入输出操作。
3. 上机实验内容：  
教材第9章习题 9.2；  
教材第9章习题 9.4；  
教材第9章习题 9.9；  
教材第9章习题 9.11。
4. 学时数：4 学时。

## 5.7 课程设计

课程设计的目的和任务：本课程设计是综合性强的编程，编程中会用到较多的语法知识和较复杂的算法，目的是进一步激发学生的程序开发兴趣，提高学生的程序设计能力。

### 5.7.1 课程设计一

用结构类型编写一个程序，完成生命游戏的功能：（选做）

20世纪70年代，一种被称作“生命游戏”的小游戏曾风靡一时。这种游戏相当简单，假设有一个像棋盘一样的方格网，每个方格中放置一个生命细胞，生命细胞只有两种状态：“生”或“死”。游戏规则如下：



- 如果一个细胞周围有3个细胞为生（一个细胞周围共有8个细胞），则该细胞为生，即该细胞若原先为死，则转为生，若原先为生，则保持不变；
- 如果一个细胞周围有2个细胞为生，则该细胞的生死状态保持不变；
- 在其他情况下，该细胞为死，即该细胞若原先为生，则转为死，若原先为死，则保持不变。

依此规则进行迭代变化，使细胞生生死死，会得到一些有趣的结果。该游戏之所以被称为“生命游戏”，是因为其简单的游戏规则，反映了自然界中的生存规律：如果一个生命，其周围同类生命太少的话，会因为得不到帮助而死亡；如果太多，则会因为得不到足够的资源而死亡。

写一个程序，模拟生命游戏过程。

### 5.7.2 课程设计二

最佳运动员评选问题

从100名优秀运动员中评选出10名最佳运动员。具体规则如下：

- (1) 运动员按1, 2, 3, ..., 100顺序编号；
  - (2) 由键盘输入所收到的选票，每张选票至多可写10个不同的编号；
  - (3) 对应名次的运动员编号可以有空缺，但必须用0表示；
  - (4) 若编号超出规定的范围，或编号重复出现，作废票处理；
  - (5) 按选票中所列最佳运动员顺序给他们计分，计分标准如下：从第一名至第十名所得分数依次为：15, 12, 9, 7, 6, 5, 4, 3, 2, 1；
  - (6) 按各运动员所得分数高低进行排队，列出最佳运动员前10名排名表，格式为：

名次	运动员编号	合计得分	合计得票数
如果得分相同，则得票多的排名在前；如果得分和得票都相同，则编号小的在前。			
- （提示：评选过程分为3个步骤：接收选票、选票检查是否为废票，以及有效票统计。）

### 5.7.3 课程设计三

制作一个同学通讯录，用于管理同学们的通信方式和资料，可以实现查找、添加和删除等功能。基本要求如下：

- (1) 用结构实现通讯信息；
- (2) 通讯资料包括姓名、学号、专业、手机号和家庭地址等；
- (3) 添加一个记录时，一次必须输入所有的信息；
- (4) 能够通过姓名、学号和手机号来进行查询；
- (5) 当出现错误或者过期的记录时，可以进行删除。

（提示：各种不同的功能由具体的函数实现。）

### 5.7.4 课程设计四

试编制一个五子棋游戏，要求每一颗棋子放在棋盘上时有相应的伴音，每一局棋结束时播放一段音乐。

## 第6章 主教材习题参考答案

### 6.1 习题 1

#### 1.1 C 语言具有哪些主要特点？

答：(1) C 语言是一种兼有高级语言和汇编语言优点的语言；

(2) C 语言是一种结构化程序设计语言；

(3) 语言数据类型丰富；

(4) C 语言具有种类丰富的运算符；

(5) C 语言具有预处理功能。

#### 1.2 C 语言的主要用途是什么？

答：作为一种系统程序设计语言，C 语言已广泛用于为各种不同的计算机系统编写有关的系统软件，如操作系统、编译系统、汇编器及编辑器等；作为一种应用程序设计语言，C 已广泛用于编写各种应用领域的应用软件，如数据库管理软件、CAD / CAM 软件、文字处理软件、图形软件、办公自动化软件、科学计算及工程应用软件等。

#### 1.3 简述 C 程序的结构特点。

答：C 程序为函数模块结构，每个 C 程序都是由一个或多个函数组成，其中至少有一个 main() 函数。程序从 main() 函数开始执行，程序在执行中可以调用由编译系统提供的各种标准库函数和由用户自定义的函数。

#### 1.4 简述 C 程序开发的一般步骤。

答：C 程序的运行一般要经过四个步骤。即源程序的编辑、源程序的编译、目标程序的链接和可执行程序的运行。

#### 1.5 以下几个语句的显示结果是什么？

```
printf("first second third");  
printf("\nfirst second third\n");  
printf("first\n second\n third\n");  
printf("first\n\n second\n\n third\n");
```

显示结果为：

```
first second third  
first second third  
first
```

second

third

first

second

third

注意: \n 为换行符。

1.6 以下程序完成的功能是什么?

```
#include <stdio.h>
main()
{
    int i,j,k;                /*定义变量*/

    printf("Please input the two integers:i and j\n");
    scanf("%d,%d",&i,&j);    /*调用 scanf 函数输入变量*/
    k=i*i+j*j;                /*关键算法*/
    printf("The result of i*i+j*j is %d\n",k); /*调用 printf 函数输出结果*/
}
```

答: 计算两个数的平方和。

1.7 以下哪些标识符是合法的?

答: 合法的标识符有: int\_do, dO, DWMax\_Value, TWO\_AND\_THREE。

原因: int 为 C 语言关键字; 32data 是由数字开头; one \$ 含有 '\$', 不合法。

1.8 试编写一个 C 语言程序, 它输入一个浮点数, 计算它的倒数并将结果输出。

参考程序:

```
#include <stdio.h>
main()
{
    float x,y;

    printf ("\nPlease input a float number:");
    scanf ("%f",&x);
    y=1/x;                /*求倒数*/
    printf ("\nThe reciprocal of  %f  is  %f\n",x,y);
}
```

1.9 试编写一个 C 语言程序, 它输入三个整数, 计算它们的和并将结果输出。

参考程序:

```
#include <stdio.h>
main()
{
    int x,y,z,sum;

    printf("请输入 3 个整数 (x, y, z): \n");
    scanf ("%d, %d, %d", &x, &y, &z) ;
    sum=x+y+z;                      /*三个整数求和*/
    printf("三个整数 %d+%d+%d 的和是:%d\n",x,y,z,sum);
}
```

1.10 试编制输出如下信息的 C 语言程序。

\*\*\*\*\*

**Turbo C**

\*\*\*\*\*

参考程序:

```
#include <stdio.h>
main()
{
    printf("*****\n");
    printf("      Turbo C      \n");
    printf("*****\n");
}
```

本章习题常见错误为:

1. 习题 1.8, **x, y 是 float 型**, 所以在 scanf ()和 printf ()函数中, **格式字符为%f**。很多初学者写为%d 或%x, 详细请参见主教材表 2.7.1。
2. 习题 1.9, **scanf ()函数中, 3 个整数格式符%d 用逗号分开**, 所以在输入 3 个整数时也应该用逗号分开, 而不是空格等其他符号, 否则会发生错误。详细请参见主教材 2.7.2 小节内容。
3. 习题 1.9, **scanf ()函数中, 参数前须加上取地址符&**, 这是 scanf ()和 printf ()函数的区别。如果没有&, 在编译的时候不会出现 error, 但在运行时会出现内存读取的错误, 初学者往往不易察觉。
4. 习题 1.9, 作者在答案中使用了中文输入, 这在 VC++工具中是允许的。需要注意的是, 除了输出信息外, 程序中其余的标识符、关键字等都不能使用中文输入法输入, 必须在英文输入法下输入, 比如同样是分号“;”, 使用中文输入法输入的“;”将会出错。
5. 习题 1.8 和 1.9 中, **printf ()函数格式控制中的格式说明符与输出参数的个数和类型**

必须一一对应，否则将会出现错误。

6. `scanf()`函数不要使用'`\n`'和'`\t`'等转义字符，如果需要，可以在前后的 `printf()`中使用。

## 养成良好的编程习惯

C 语言是一种应用性很强的编程语言，不仅适合进行编程语言的学习，而且也同样适合进行软件工程的开发。作为 C 语言的学习者，不仅需要掌握 C 语言的语法和算法知识，还需要从学习的一开始就养成良好的编程习惯，用良好的编程风格进行程序设计。良好的编程习惯和风格，可以使人更方便和迅速地理解程序的结构，从而最大限度地提高程序设计的效率。细节决定成败，看似微不足道的小习惯，往往决定了一个程序项目的成败。另外，学习者在实际的学习和开发中，需要不断积累和体会，才能真正养成自己熟悉的良好编程习惯，终生受用。

在后面各章节中，我们将结合教材内容陆续介绍一些常用的编程习惯。这些习惯和风格是由很多经验丰富的程序员总结而成，实用性很强，也容易掌握，希望学习者能理解并应用于编程学习中。

## 编程好习惯一：规范存储

在每次上机学习时，为本次编程建立单独的文件夹目录，并对文件夹进行规范命名，同时将本次上机编程内容存储到该文件夹中。如果编程是在实验室机房中进行，在下机时，应该将程序文件夹拷贝到 U 盘、移动硬盘等设备中带走。规范存储有利于程序的修改和查找，也避免了不同版本的程序冲突和误删除。对于较为大型的程序，需要长时间开发，规范存储也能保证编程日志和各种版本的完整。

## 6.2 习题 2

### 2.1 C 语言中的数据类型主要有哪些？

答：C 语言中的数据类型主要分为：基本类型、构造类型、指针类型和空类型。其中基本类型又分为：整型、字符型、单精度浮点型和双精度浮点型。构造类型又分为：数组、结构、联合和枚举。详细请参见主教材 2.1 节。

### 2.2 C 语言为什么要规定对所有用到的变量“先定义后使用”？这样做有什么好处？

答：原因是：1. 编译系统会根据定义为变量分配内存空间，分配空间的大小与数据类型有关；2. 系统可以根据变量的类型检查对该变量的运算是否合法，这样就给程序员调试程序带来方便。详细参见主教材 2.3 节。

### 2.3 下面哪些是合法的常量？

123, 3.1415926, 0892, 'M', '\n', 0xabc, 0.618E-6, "Morning!", 3.8e-3.14

答：合法的有：123, 0892 和 0xabc 是整型常量；3.1415926 和 0.618E—6 是浮点型常量，但 0.618E—6 不规范；'M'和'\n'是字符常量；"Morning!"是字符串常量。3.8e—3.14 不合法。详细参见主教材 2.2 节。

2.4 字符常量和字符串常量有什么区别？下述字符串常量的长度各是多少？在内存中存储时各自占用的内存单元数又是多少？

(1) "Hello! "

(2) "ABC\n\\TH\064?"

答：字符常量和字符串常量的表示方法不同：字符型常量是用单引号括起来的一个字符；字符串常量是用一对双引号括起来的零个或多个字符组成的序列。最主要的区别是存储的不同：字符型数据在存储时，并不是将该字符本身放到内存单元中，而是将该字符的相应 ASCII 码值存放到该存储单元中；而字符串常量是将字符存储，并自动在其末尾加上'\0'作为字符串结束标志，所以字符串常量在内存中所占用的字节数为字符串长度+1。

"Hello!"长度是 6 个字节，占用内存单元数是 7。

"ABC\n\\TH\064?"长度 9，占用内存单元数 10。需要注意的是，在该字符串中有转义字符，\n 为换行，\\为反斜杠字符，\064 为八进制转义字符，对应 ASCII 码为 52 的字符'4'。详细参见主教材 2.2。

2.5 在 ASCII 字符集中，字母 C 的 ASCII 码值是 67，以下程序的输出结果是什么？

```
#include "stdio.h"
void main()
{
    char a='C';
    int b=10;
    a=a+b;
    printf ("%c,%d\n",a,a) ;
}
```

答：结果：M，77。在执行 a=a+b 后，字符变量 a 的 ASCII 码变为了 77。

2.6 写出以下程序运行的结果。

```
#include "stdio.h"
void main()
{
    int x;
    x= -30 + 4*7-24;
    printf("x=%d\n", x);
    x= -30*5%-8;
    printf("x=%d\n", x);
}
```

答：x=-26

x=-6。第二个运算是取余操作。

2.7 下列对变量进行定义的语句哪些正确？哪些不正确？为什么？请将不正确的改正过来。

- (1) char c1,int a2;      (2) INT a,b;FLOATx,y;      (3) a,b:char;      (4) char if;  
(5) int a,b      (6) Int a:b:c;      (7) int a,x;float x,y;

答：(1) char c1,int a2;错误，因为 c1 和 a2 是不同类型的变量，两次定义应该用分号分开。

(2) INT a,b;FLOATx,y;错误，因为：①C 语言需要区分大小写，所以关键字 FLOAT 和 INT 必须小写为 int 和 float；②关键字和参数间应该空格，float 和 x 应该加空格。

(3) a,b:char;错误，应该写为 “char a,b;”。

(4) char if;错误，if 是 C 语言的关键字，重新命名变量。

(5) int a,b 错误，变量定义语句应该以分号表示结束，在语句后加 “;”。

(6) int a:b:c;错误，应该使用逗号分隔变量名表中的多个变量。

(7) int a,x; float x,y;错误，变量 x 重复定义。

2.8 写出下面程序的输出结果。

```
#include "stdio.h"
void main()
{
    printf("%d\n",NULL);
    printf("%d,%c\n",50,50);
    printf("%d,%c,%o\n",68+10,68+10,68+10);
}
```

答：结果是：0

50, 2

78, N, 116

结果分析：第 1 次输出的是 NULL 对应的 ASCII 码；第 2 次输出的是十进制整数 50 和 ASCII 码 50 对应的控制字符 2；第 3 次输出的是十进制整数 78、ASCII 码 78 对应的控制字符 N 和十进制整数 78 转换为八进制的整数 116。

2.9 分析下面程序的运行结果，并上机予以验证。

```
#include "stdio.h"
void main()
{
    char i='a';
    char j='b';
    char k='c';
    char m='\101';
    char n='\116';
```

```
printf("a%cb%c\tc%c\tabc\n",i,j,k);
printf("\tb%c%c%c"m,n);
}
```

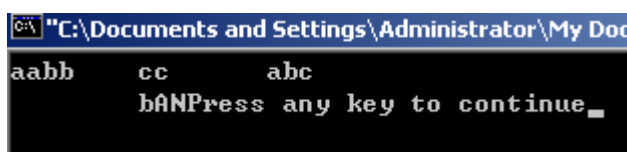
程序中 char i='a';char j='b';char k='c';char m='\101';char n='\116';

这几个语句能否改成如下语句:

```
int i='a';int j='b'; int k=99;int m='\101';int n='\116';
```

为什么?

答: 该程序的结果如下:



结果分析: 第1次按字符输出, 读者应该不难理解。第2次输出需要注意的是'\101'和'\116'是八进制转义字符, 输出的是 ASCII 码为 65 和 78 对应的控制字符'A'和'N'。

当把变量类型全部替换为 int 后, 结果依然正确。因为在这样的情况下, 赋给变量的值为这些字符对应的 ASCII 码。

#### 2.10 求下列表达式的值。

(1) 假设 x=5.6, a=8, y=12.3

```
x+a%5*(int)(x+y)%3/5
```

(2) 设 a=21, b=30, x=4.2, y=8.4

```
(float)(a+b)/6+(int)x%(int)y
```

答: (1) 5.600000。 (2) 12.500000

结果分析: 本题运算过程不难, 需要读者注意的是变量类型的强制转换。详细参加主教材 2.5 节。

#### 2.11 若 x=13, y=20, z=4, 下列各表达式的结果是什么?

(1) (z>=y>=x) ? 1: 0

(2) z>=y&& y>=x

(3) !(x<y)&&!x||z

(4) x<y?x++:++y

(5) z+=x>y?x++:++y

答: (1) 0; (2) 0; (3) 1; (4) 13; (5) 25。

本题需要注意的是: (4) 中, 注意自加操作的前缀和后缀; (5) 中 "?" 优先级大于赋值, 所以先进行条件运算再赋值。

#### 2.12 用 C 语言描述下列命题。

(1) i 小于 j 或小于 k。



- (2) i 和 j 都小于 k。
- (3) i 和 j 中有一个小于 k。
- (4) i 是非正整数。
- (5) i 是奇数。
- (6) i 不能被 j 整除。

答：(1)  $i < j \parallel i < k$ ; (2)  $i < k \&\& i < j$ ; (3)  $i < k \parallel j < k$ ; (4)  $\text{int } i; i \leq 0$  或者  $\text{mod}(i, 1) == 0 \&\& i \leq 0$ ;  
 (5)  $\text{int } i; i \% 2 = 1$  或者  $\text{mod}(i, 2) = 1$ ; (6)  $i \% j != 0$  或者  $\text{mod}(i, j) != 0$ 。

### 2.13 用条件表达式描述：

- (1) 取三个数中的最大值。
- (2) 任意两个整数存放在变量 m 和 n 中，让小数存放在 m 中，大数存放在 n 中，并输出大数。

答：(1) 最常用的方法是定义一个中间变量 t，执行下列语句：

```
t=a>b?a:b;
t=c>t?c:t;
```

最后 t 的值就是题目所求。在学习了条件语句后，读者可以尝试用 if-else 语句完成以上功能。

或者：(max=(a>b?a:b))>c?max:c

```
(2)  int x,y,m,n; //定义变量
      scanf("%d,%d",&x,&y);//输入两个值
      x>y?(n=x,m=y):(n=y,m=x);//按题目要求进行条件运算
      printf("%d",n);//输出最大值
```

### 2.14 分析下面程序的运行结果，并上机予以验证。

```
#include "stdio.h"
void main()
{
    int x,y,z;
    x=13;
    y=25;
    z=0;
    x=x&& y||z;
    printf ("%d\n",x) ;
    printf ("%d\n",x||!y&& z) ;
    x=013;
    y=025;
    z=01;
    printf("(1)%d\n", x|y&z);
```

```

printf("(2)%d\n", x|y&~z);
printf("(3)%d\n", x^y&~z);
x=1;
y=-1;
x=x<<3;
printf("%d\n", x);
y<=3;
printf("%d\n", y);
}

```

答：结果为：1

```

1
(1) 11
(2) 31
(3) 31
8
-8

```

结果分析：本题需要主要四种位操作和移位运算，详细内容参见主教材 2.4 节。

2.15 用下面的 scanf 函数输入数据，使 i=40, j=78, k=56.89, m=2.3, c1='R', c2='T'。

请问在键盘上如何输入？

```

#include "stdio.h"
void main ()
{
    int i,j;
    float k,m;
    char c1,c2;
    scanf("i=%d**j=%d",&i,&j);
    scanf("%f## %f",&k,&m);
    scanf("%c,%c",&c1,&c2);
}

```

答：输入：i=40\*\*j=78 回车

56.89##2.3 回车

R,T

结果分析：详细请参见主教材 2.7 节。

2.16 用下面的 scanf 函数输入数据，使 a=50, b=18, x=2.89, y=-21.3, z=89.2, c1='A', c2='B'。请问在键盘上如何输入？

```
scanf("%5d%6d%c%c%f%f%f",&a,&b,&c1,&c2,&x,&y,&z);
```

答：50    18    AB2.89    -21.3    58.54    89.2

2.17 找出下面程序中的错误，并予以分析。

```
/*This is a program with some errors.*/
#include "stdio.h"
void main ()
{
    int x,y;
    printf('Input;x=?\n');
    scanf("%d",x);
    printf("square(x)= %d",x*x);
    printf("Y=%d\n",y);
}
```

答：(1) 第 1 个 printf() 中应该使用双引号；  
 (2) scanf() 中的变量 x 没有取地址符号；  
 (3) 变量 y 没有初始化。

2.18 以下语句的输出应是什么？

```
int sum=10,cap=10;
cap=sum++,cap++,++cap;
printf ("%d\n",cap) ;
```

可选答案是：

(1) 10      (2) 11      (3) 12      (4) 13

答：(3) 12。

2.19 已知在 ASCII 代码集中，字母 A 的序号是 65，以下程序输出结果是什么？从以下可选答案中选择你认为正确的一个：

```
#include "stdio.h"
void main ()
{
    char c1='B',c2='Y';
    printf ("%d,%d\n",++c1,—c2) ;
}
```

(1) 66, 89  
 (2) 67, 88  
 (3) 67, 89  
 (4) B, Y  
 (5) C, X  
 (6) 输出格式不合法，输出错误信息

答：(2) 67, 88。注意是 ASCII 码的运算。

2.20 以下程序的输出结果是什么？从可选答案中选择一个。

```
#include "stdio.h"
void main ()
{
    int a=0100,b=100;
    printf("%d,%d\n",--a,b++);
}
```

可选答案：

(1) 99, 101      (2) 63, 100      (3) 63, 101      (4) 077, 100

答：(2) 63, 100。分析：八进制数 0100 化为十进制是 64，并执行前缀自减操作；100 执行后缀操作。

2.21 分析下面程序的运行结果。

```
#include "stdio.h"
void main()
{
    int x,y,z;
    x=y=z=1;
    x+=y+=z;
    printf("(1)%d\n",x<y?y:x);
    printf("(2)%d\n",x<y?x++:y++);
    printf("x=%d,y=%d\n",x,y);
    printf("(3)%d\n",z+=x<y?x++:y++);
    printf("x=%d,y=%d,z=%d\n",x,y,z);
    x=5;
    y=z=6;
    printf("(4)%d\n",(z>=y>=x)?1:0);
    printf("(5)%d\n",z>=y&& y>=x);
}
```

答：(1) 3

(2) 2

x=3,y=3

(3) 4

x=3,y=4,z=4

(4) 0

(5) 1

结果分析：本题一共出现了 7 次输出 printf()，其中有 5 次是运算后输出，我们也就只

对这5次进行讨论。第1次运算前，“ $x=3, y=2, z=1$ ;”所以第一次输出是 $x$ 的值为3，第2次运算，输出的是 $y$ 的值为2，并且 $y$ 进行后缀自加操作，这里注意：由于条件运算不执行 $x++$ 操作，所以 $x$ 的值不变，初学者往往在这里犯错。第3次运算，先条件运算后赋值，输出的值为 $z+=3$ ，结果是4。第4次运算也需要注意， $z>=y>=x$ 并不是 $6>=6>=5$ ，而是先运算得到 $x>=y$ ，结果是1，再求 $1>=x$ ，所以结果为0。

**2.22** 编写一个程序，求出给定半径 $r$ 的圆的面积和周长，并且输出计算结果。 $r$ 的值由用户输入，用浮点型数据处理。

参考程序：

```
#include "stdio.h"
main()
{
    float r, area, per; //定义 area 为面积，per 为周长

    printf("请输入圆的半径:\n");
    scanf("%f", &r);
    area = 3.14 * r * r; //求面积
    per = 2 * 3.14 * r; //求周长
    printf("该圆的面积为%8.2f, 周长为%8.2f\n", area, per);
}
```

作者在结果输出的时候对字段宽度进行了控制，读者应该掌握占用的方法，使得结果更加规范化。

**2.23** 已知华氏温度与摄氏温度之间的转换公式是：

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

编写一个程序，将用户输入的摄氏温度转换成华氏温度，并予以输出。

参考程序：

```
#include "stdio.h"
main()
{
    float c, f;

    printf("请输入一个摄氏温度\n");
    scanf("%f", &c);
    c = (float)5/9 * (f - 32); //使用了强制类型转换
    printf("华氏温度为%f", f);
}
```

**2.24** 编写程序输入年利率 $I$ （例如2.52%），存款总额 $S$ （例如100 000元），计算一年后的本息合计并输出。

参考程序：

```
#include "stdio.h"
main()
{
    float I,S,sum;    //定义 sum 为总和

    scanf("%f,%f",&I,&S);
    sum=(1+I)*S;
    printf("sum is %8.2f",sum);
}
```

本章习题常见错误:

1. C 语言需要区分大小写, 所以读者一定要注意程序中的大小写输入, 避免错误。
2. 定义的变量没有初始化, 造成结果错误。
3. 多次条件运算一定要注意变量值的变化, 分支语句的运算是由条件决定的, 避免多余的计算。
4. 自加操作和自减操作一定要区分前缀和后缀。

小技巧: 很多 C 语言的初学者在编程序的时候, 都容易忘记 ( ) 和 {} 的匹配, 往往缺少后半。在简单程序中, 还比较容易找到错误, 但一旦程序复杂, 出现类似错误就很难找到和纠正了。有经验的程序员往往在输入 ( ) 和 {} 等符号时, 都同时一起输入, 再退格输入中间的程序, 这样能很好地保证这些符号的匹配, 减少错误。

## 编程好习惯二: 见名知意

本章主要介绍了变量的类型和基本运算方法。见名知意, 指我们对程序中的各种标识信息 (包括常量、变量、宏和自定义数据类型等) 进行命名时采用系统、规范和易理解的命名方式。系统而规范的命名不仅可以增加系统的可读性, 还能避免重名, 提高编程的效率。本书使用命名方式为: 变量使用英文单词命名; 宏定义使用大写的字母; 自定义数据类型采用小写等。C 语言的命名规范也不唯一, 读者可以根据实际需要选择不同的命名方式。

### 6.3 习题 3

#### 3.1 编写一个程序从终端上输入两个整数, 检查第一个数是否能被第二个数整除。

参考程序:

```
#include <stdio.h>
main()
{
    int i,j;
```

```
printf("请输入两个整数: ");
scanf("%d,%d",&i,&j);
if(i%j)    //条件判断
    printf("第一个数不能被第二个数整除\n");
else
    printf("第一个数能被第二个数整除\n");
}
```

结果分析：本题较简单，对于关键的条件判断语句 `if(i%j)`，直接使用了运算结果进行比较。初学者如果不容易理解，也可以使用一个中间变量进行运算 `temp=i%j` 再对 `temp` 进行判断。

### 3.2 以下程序的功能是什么？

```
main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    a=a<b?a:b;
    printf("%d\n",a);
}
```

答：输入两个整数，求数值较小者并输出。

### 3.3 编写一个程序输入某人的身高和体重，按下式确定此人的体重是否为标准、过胖或过瘦：

- (1) 标准体重 = (身高 - 110) 公斤；
- (2) 超过标准体重 5 公斤为过胖；
- (3) 低于标准体重 5 公斤为过瘦。

参考程序：

```
#include<stdio.h>
main()
{
    float high,weight,nomalw;

    printf("请输入某人的身高(cm)， 体重(kg)\n");
    scanf("%f,%f",&high,&weight);
    nomalw=high-110;
    if(weight>=nomalw+5)
        printf("过胖，请运动一下！\n");
```

```

else if(weight<=nomalw-5)
    printf("过瘦, 请多吃点吧");
else
    printf("标准, 请保持! ");
}

```

本题目较简单, 注意三种条件的判断方法。

### 3.4 编写程序计算:

$$\text{result} = \begin{cases} 1+2+\cdots+i & i \leq 5 \\ 100 - i - (i-1) - \cdots - 1 & 5 < i \leq 10 \\ i*i & i > 10 \end{cases}$$

参考程序:

```

#include<stdio.h>
main()
{
    int i,j,result=0;

    printf("please input a integer:\n");
    scanf("%d",&i);
    if(i<=5)
        for(j=1;j<=i;j++)
            result+=j;
    else if(i>10)
        result=i*i;
    else
    {
        result=100;
        for(j=i;j>=1;j--)
            result-=j;
    }
    printf("result is %d\n",result);
}

```

或者

```

#include<stdio.h>
main()
{
    int i,j,result=0;

```



```
printf("please input a integer:\n");
scanf("%d",&i);
if(i<=5)
    for(j=1;j<=i;j++)
        result+=j;
else if(5<i&& i<=10)
{
    result=100;
    for(j=i;j>=1;j--)
        result-=j;
}

else
{
    result=i*i;
}
printf("result is %d\n",result);
}
```

结果分析：第一种算法较容易理解；第二种算法中，需要大家注意的是当（ $5 < i \leq 10$ ）时，在条件判断中不能写为（ $5 < i \leq 10$ ），这样是错误的，而应该写成（ $5 < i \&\& i \leq 10$ ），这是个较常见的错误。

**3.5 编程序，完成以下功能：输入 5 个整数，求其中数值最大者。**

参考程序：

```
#include<stdio.h>
main()
{
    int i,j,max=-32768;

    for(j=1;j<=5;j++)
    {
        printf("请输入第%d 个整数\n",j);
        scanf("%d",&i);
        if(max<i)
            max=i;
    }
}
```

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

结果分析：本题有两点需要读者注意：（1）if(max<i)时，执行 max=i 运算，这样就保证了每次循环的时候，变量 max 都存储的是前面已经循环的最大值。（2）为什么变量 max 的初始值是-32 768 呢，请大家思考 int 变量的取值范围是什么？是-32768 到 32 768。所以将 max 初始为-32 768，保证了输入任何一个整数都能满足题目的算法，比如，将 max 初始为 0，那么当输入的 5 个整数都小于 0 时，算法就失效了。请大家考虑如果本题是求最小值，又应该怎么定义和初始变量呢？

### 3.6 编写一个程序计算 $x^y$ ，其中 x 是浮点数，y 是正整数。

参考程序：

```
#include<stdio.h>  
main ()  
{  
    float x,answer=1;  
    int i,y;  
  
    printf("Please input a float and a integer:");  
    scanf("%f,%d",&x,&y);  
    for(i=1;i<=y;i++)  
        answer=answer*x;  
    printf("%6.2f^%d=%f\n",x,y,answer);  
}
```

3.7 编写一个程序，使其能读入并计算一个只包含加减运算的表达式，每一个输入的数据都是浮点数，除第一个数以外，其余每个数前面都有一个运算符，例如：

23+43-233+234;表达式以分号“;”结束。

参考程序：

```
#include<stdio.h>  
main()  
{  
    float temp,sum;  
    char ch,operat;  
  
    printf("please input a digital expression:");  
    scanf("%f",&sum);//输入整数  
    ch=getchar();//输入运算符
```

```

while(ch!=';')//循环结束条件
{
    scanf("%f",&temp);
    switch(ch)
    {
        case '+': sum=sum+temp;//加运算
            break;
        case '-': sum=sum-temp;//减运算
            break;
        default: printf("error\n");
    }
    ch=getchar();
}
printf("%f",sum);
}

```

输入 23+43-233+234，得到的结果是：67.00000。

结果分析：在本题中，使用到了 `getchar()`、`while()` 和 `switch` 三个新函数。`Getchar()` 是字符输入函数，保存字符“+”、“-”。当输入一个整数和一个运算符号后，进行一次循环运算，直到“;”结束。

### 3.8 编写一个程序计算

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!}$$

参考程序：

```

#include<stdio.h>
main ()
{
    float x,ex=1,temp,mul;
    int n,i,j;
    double y;

    printf("请输入数 X:");
    scanf("%f",&x);
    printf("请输入 N 值(建议 N<35):");
    scanf("%d",&n);
    for (i=1;i<=n;i++)
    {
        temp=1;

```

```
        mul=1;
        for (j=1;j<=i;j++)
        {
            temp=temp*x;//计算分子
            mul=mul*j;//计算分母
        }
        ex=ex+temp/mul;//各项求和
    }
    printf ("e^%6.3f=%f\n",x,ex) ;
}
```

结果分析：本题中，将复杂的运算拆分为 3 步进行，分别求出了每一项的分子和分母后，再进行求和运算。将复杂运算分为很多简单的小部分，是一种良好的程序设计方法。

### 3.9 编写一个程序求满足以下公式的变量 k 的最大值。

$$2^k \leq m$$

其中，m 是程序输入的一个正整数。

参考程序：

```
#include<stdio.h>
main ()
{
    int i,k=0,m,temp=1;

    printf("Please input a integer(m):");
    scanf("%d",&m);
    do //循环运算，每循环一次，k++
    {
        k++;
        temp=temp*2;
    } while (!(temp>=m));
    printf("m=%d\tk=%d\t2^%d=%d\n",m,k,k,m);
}
```

结果分析：本题作者使用了 do-while 语句，详细内容请参见主教材 3.6 节。在循环运算中，temp 每乘以 2，k 自加，直到循环结束，这样 k 的值就等于乘法运算的次数。本题还可以使用 while、if 等条件判断语句完成，请读者思考。

### 3.10 以下程序输入一个整数，然后依次显示该整数的每一位。

```
main()
{
```

```
int number,digit;
scanf("%d",&number);
do
{
    digit=number%10;
    printf("%d",digit);
    number/=10;
}while(number!=0);
printf("\n");
}
```

该程序对输入的正整数可以正常工作，但如果输入负数，则会得到错误的结果。请改进该程序，使它在输入负数时也能工作。例如，如果输入的数是-4567，则输出 7654-。

解题思路：针对题目的要求，我们首先来分析正数和负数有什么不同：在本题中，主要的区别就是整数没有符号，而负数有符号“-”。找到区别后，本题不难解决。在已知算法的基础上增加正负判断，如果是负数，在输完数值的每一位后再输出“-”。

参考程序：

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

```
main ()
```

```
{
```

```
    int number,digit,sign;
```

```
    printf("Please input a integer:");
```

```
    scanf("%d", &number);
```

```
    if(number<0) sign=-1;
```

```
    else sign=1;
```

```
    do
```

```
    {
```

```
        digit=sign*(number%10);    //保证了负数的每一位输出为正数。
```

```
        Printf("%d", digit);
```

```
        number/=10;
```

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

```
    if(sign==-1)
```

```
        printf("-");
```

```
    printf("\n");
```

```
}
```

3.11 编写一个程序，计算一个整数的各位数字之和。例如输入的数是 2568，该程序计算并显示 2+5+6+8 的值。

解题思路：由上题 3.10 我们已经了解到了怎么样求出一个整数的各位，在本题中该方法同样适用。不同的是，本题的循环需要知道次数，所以先求出整数的位数，再求出各位，最后再将各位的数值相加。

参考程序：

```
#include<stdio.h>

main ()
{
    int temp,number;
    int i,j,k;
    static int count,sum;    //定义为静态变量

    printf("Please input a short type integer:");
    scanf("%d",&number);
    temp=number;
    do
    {
        temp/=10;
        count++;
    } while (temp!=0);    //求出整数的位数并赋值给 count

    temp=number;
    for(i=count;i>1;i--)
    {
        k=1;
        for(j=1;j<i;j++)
            k=k*10;    //参见结果分析
        temp=number/k;    //temp 保存的是每一位的数值
        number=number-k*temp;    //取完一位后，将未处理的部分再赋值给 number
        printf("%d%c",temp,'+');
        sum=sum+temp; //求和操作
    }
    sum=sum+number;
    printf("%d=%d\n",number,sum);
}
```

输入 2568, 输出为  $2+5+6+8=21$ 。

结果分析: 由于在每次循环后需要保存 count 和 sum 的值, 所以定义为静态变量 static, 这个概念将在第 7 章中学习。程序中的  $k=k*10$  是保证 number 能与十、百、千、万……各位相除。

3.12 编写一个程序, 当输入一个整数时, 用英语单词输出该数的每一位数字。例如: 输入 3567, 输出:

first-digit	second-digit	third-digit	four-digit
seven	six	five	three

解题思路: 经过 3.10 和 3.11 两题练习后, 我们已经掌握了求整数每一位的方法。

在本题中, 不同就是用英文字符串输出表示数字。因为阿拉伯数字一共有 10 个, 所以适合使用 switch-case 语句。

参考程序:

```
#include<stdio.h>
main ()
{
    int i,number,digit;
    static int count;

    printf("Please input a integer:");
    scanf("%d",&number);
    printf("first-digit  second-digit");
    printf("third-digit  fourth-digit  fifth-digit\n");
    do
    {
        count++;
        digit=(number%10);
        switch(digit)
        {
            case 0: printf(" zero ");
                    break;
            case 1: printf(" one ");
                    break;
            case 2: printf(" two ");
                    break;
            case 3: printf(" three ");
                    break;
```

```
        case 4: printf (" four " ) ;
                break;
        case 5: printf (" five " ) ;
                break;
        case 6: printf (" six " ) ;
                break;
        case 7: printf (" seven " ) ;
                break;
        case 8: printf (" eight " ) ;
                break;
        case 9: printf (" nine " ) ;
    }
    number/=10;
} while (number!=0);
for(i=count; i<5; i++)
printf (" zero " ) ;
printf ("\n") ;
}
```

### 3.13 编写一个程序找出 1~100 中的所有素数。

解题思路：根据素数的定义，针对每一个整数，需要对小于它的所有数相除，才能确定是否为素数，所以需要两次循环才能得到结果。

参考程序：

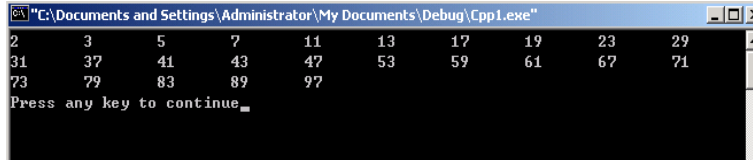
```
#include<stdio.h>
main()
{
    int i,j,flag;//flag 为标志位，起到标记素数的作用

    for(i=2;i<101;i++)//循环每个数
    {
        flag=1;
        for(j=2;j<=i/2&&flag==1;j++)//循环被除的每个数
            if(i%j==0)//判断素数
                flag=0;
        if(flag)
            printf("%d\t",i);//按格式输出
    }
}
```



```
printf("\n");
}
```

运算结果为:



3.14 编写一个程序，找出被 2、3 和 5 整除时余数均为 1 的最小的 10 个自然数。

参考程序:

```
#include<stdio.h>
main ()
{
    int i=1;
    static count;

    printf("\n 能被 2、3 和 5 整除时余数为 1 的最小 10 个自然数是:\n");
    while (count<=10)
    {
        if((i%2==1)&&(i%3==1)&&(i%5==1))//判断条件
        {
            count++;
            printf("%d ",i);
        }
        i++;
    }
}
```

结果分析: 本题算法比较简单, 读者应该不难理解。

3.15 编写程序, 输入一组数, 以 0 作为输入结束标志, 然后显示与第一个数符号相同的所有数。

解题思路: 本题难度不大。首先需要对第一个数的符号进行判断, 其次对后面的每个数进行判断, 满足条件就输出。

参考程序:

```
#include<stdio.h>
main ()
{
    float i,number;
```

```
int sign; //定义一个标志位

printf("\nPlease input a group of numbers(end sign:0):");
scanf("%f",&number);
printf("The first number is:%f\n",number);
if (number<0)
sign=-1;
else
sign=1;
while (number!=0)
{
    scanf("%f",&number);
    if ((sign==1)&&(number<0))    //判断负数
        printf("%9.3f ",number);
    if ((sign==1)&&(number>0))    //判断正数
        printf("%9.3f ",number);
}
printf("\n");
}
```

### 3.16 编写一个程序，求两个正整数 x 和 y 的最小公倍数。

解题思路：最小公倍数的求法最常用的就是欧几里德算法，又被称为辗转相除法，用两个数的积除以最大公约数即是最小公倍数。但需要注意的是：输入两个的数并没有大小之分，所以需要先判断，确保大小关系，再计算。

参考程序：

```
#include<stdio.h>
main ()
{
    int x,y;
    int temp,num1,num2;

    printf("Please input two integers:");
    scanf("%d,%d",&x,&y);
    if (x>y) //判断两个数的大小并将大者赋给 x，小的赋给 y
    {
        temp=x;
        x=y;
    }
```

```
        y=temp;
    }
    num1=x;
    num2=y;
    while(num2!=0)    //辗转相除，求出最大公约数
    {
        temp=num1%num2;
        num1=num2;
        num2=temp;
    }
    printf("两个数(%d,%d)的最小公倍数=%d\n",x,y,x*y/num1); //求出结果
}
```

3.17 编写一个程序，输入两个整数  $i$  和  $j$ ；如果  $j$  的值大于 0，则把  $i$  循环左移  $j$  位；如果  $j$  的值小于 0，则把  $i$  循环右移  $j$  位，最后输出  $i$  的值。

参考程序：

```
#include<stdio.h>
main ( )
{
    int i,j,temp;

    printf("Please input two integers:");
    scanf("%d,%d",&i,&j);
    printf("Being shifted number is i=0x%x\n",i);
    if (j>0)
    {
        printf("Circle left shift bits:%d\n",j);
        temp=i>>(16-j);
        i<<=j;
        i|=temp;    //进行位或赋值
    }
    else
    {
        j=-j; //需要先对 j 取绝对值
        printf("Circle right shift bits:%d\n",j);
        temp=i<<(16-j);
        i>>=j;
```

```
        i|=temp;
    }
    printf("After shift: i=%d\n",i);
}
```

### 3.18 编写一个程序，输入两个整数 i 和 j，显示 i 的第 j 个二进制位。

解题思路：本题常规的解法有两种，一是求出 i 的二进制再移位，二是将一个第 j 位是 1 其余位是 0 的二进制数与 i 取与操作。作者将第二种方法列出，第一种方法请读者设计。

参考程序：

```
#include<stdio.h>
main ()
{
    int i,j;
    int flag=1;    //定义一个中间量

    printf("Please input two integers:");
    scanf("%d,%d",&i,&j);
    flag=flag<<j; //flag 变成了一个第 j 位是 1 其余位是 0 的二进制数
    if(flag&i)
        printf("The %dth bit of %d is:1\n",j,i);
    else
        printf("The %dth bit of %d is:0\n",j,i);
}
```

本章常见错误是：

1. for 和 if 等语句后面没有 “;”，否则将造成错误。
2. 在 if、else 和 for 等语句后，运算如果是复合语句，必须用 {} 括起来，否则将造成错误。
3. while 和 do-while 语句在用法上有区别，do-while 语句需要用 “;” 结束，使用时需要注意。

## 编程好习惯三：有条不紊

本章主要学习了各种控制语句的使用。在较为复杂的条件判断语句中，if-else、case 语句都会多层嵌套使用，如果不规范编写，将极大影响程序的可读性。可读性是判断程序优秀与否的一个重要标准，如果程序格式混乱，条理不清，将很难阅读，修改起来也就更加困难。格式缩进和对称是这方面的基本要求。格式缩进指在某些语句的开头使用空格，不同功能的语句进行不同长度的缩进，如 case 语句；对称指一组相匹配的语句进行相同的缩进，比如

if-else 语句。在较为复杂的程序中，因为嵌套层数太多，格式缩进和对称能更好地判断语句间的匹配关系和查找错误。现在的编译器基本上都能实现简单的自动的格式缩进和对称，但在大型和复杂的程序中，还是需要程序员自己对语句的规范进行控制。

## 6.4 习题 4

4.1 将一个数组中的值按逆序重新存放。例如，原来顺序为 8，5，4，6，1，要求改为 1，6，4，5，8。

解题思路：按照主教材的算法提示，我们只要枚举数组的下标，从 0 到数组长度 N 的一半，每次枚举时，如果枚举到 i，则将数组的第 i 个元素与第 N-i 个元素进行交换即可。

参考程序：

```
#include<stdio.h>
#define N 10 //使用宏定义数组的大小
main()
{
    int a[N]; //定义整型数组
    int i; //定义循环变量
    int temp; //定义一个临时变量

    for(i=0; i<N; i++) //输入每个数组元素
    {
        printf("a[%d]", i);
        scanf("%d", &a[i]);
    }
    for(i=0; i<N/2; i++) //枚举并交换
    {
        temp=a[i];
        a[i]=a[N-i-1];
        a[N-i-1]=temp;
    }
    for(i=0; i<N; i++) //输出结果
        printf("%d\t", a[i]);
    printf("\n");
}
```

程序分析：

(1) 本题的意思是将数组元素“逆序存放”，而不是“逆序输出”。如果是“逆序输出”，程序将简单很多，只需要从数组的最后一个元素向前枚举，从高到低循环数组的下标即可。

(2) 在参考程序中，使用了宏定义#define N 10。这个概念将在第 7 章的预处理功能中

学习到。这样的好处是：程序中的 N 都将被 10 替换，这样就给程序的修改、扩充、阅读和移植带来了便捷。但需要读者注意的是，在一些编译器（如 VC 等）在数组定义时往往不能引用宏，所以在数组定义的时候需要写出宏定义具体的数。

(3) 在枚举循环语句 `for(i=0;i<N/2;i++)` 中，i 只循环到了 N/2，而不是 N-1。因为在交换语句中，作者是对 `a[i]` 和 `a[N-i-1]` 进行交换，如果循环到 N-1，交换操作将进行两次：① `a[i] ↔ a[N-i-1]`，② `a[i] ↔ a[N-i-1]`，最终的结果将不变。很多同学在设计算法的时候都容易在这里出错。

(4) 两个数的交换操作在 C 语言中经常出现，最常用的方法就是定义一个中间变量，再进行交换。例如交换两个变量 a、b，需要定义一个中间变量 temp，然后执行以下语句：

```
temp=a; //将 a 的值存放到 temp 中
```

```
a=b;    //将 b 的值赋给 a
```

```
b=temp; //将 temp 的值，也就是 a 的初值赋给 b
```

这三条语句的顺序请理解并牢记，赋值错误或者顺序错误都将引起错误。

4.2 N 盏灯排成一排，从 1 到 N 依次编号。有 N 个人也从 1 到 N 依次编号。第一个人（1 号）将灯全部打开，第二个人（2 号）将凡是 2 和 2 的倍数的灯关闭。第三个人（3 号）将凡是 3 和 3 的倍数的灯做相反处理（即将打开的灯关闭，关闭的灯打开），以后的人都和 3 号一样，将凡是与自己编号相同的灯和是自己编号倍数的灯做相反处理，请问当第 N 个人操作之后，哪几盏灯是点亮的？试编程求解这个问题，N 由键盘输入。

参考程序：

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

```
#define MaxN 1000//定义一个最大的处理数，暂时定为 1000
```

```
main()
```

```
{
```

```
    int Lamp[MaxN+1]={0};//初始化每盏灯的状态，0 为关闭，1 为打开
```

```
    int n;//定义灯数量
```

```
    int i,j;//定义循环变量
```

```
    printf("请输入灯数量:");
```

```
    scanf("%d",&n);//输入灯数量
```

```
    for(i=1;i<=n;i++)
```

```
        for(j=i;j<=n;j=j+i)
```

```
            Lamp[j]=1-Lamp[j];//参考程序分析
```

```
    for(i=1;i<=n;i++)
```

```
        if(Lamp[i]==1)//判断灯是否打开
```

```
            printf("第%d 盏灯是亮的\n",i);
```

```
}
```

运行结果为：请输入灯数量:10

第 1 盏灯是亮的

第 4 盏灯是亮的

第 9 盏灯是亮的

程序分析：此题需要解决三个问题，一是如何存储多盏灯的状态，这里是用一个数组来存储，且用数组元素的值为 1、0 分别表示灯的明灭；二是需要对哪些灯进行处理，这里使用了双重循环，外循环枚举人，内循环枚举灯；三是如何完成对灯的状态的处理，这里用的是  $Lamp[j]=1-Lamp[j]$  来实现  $Lamp[j]$  的值由 0 变 1，由 1 变 0，当然也可以用  $Lamp[j]=!Lamp[j]$  来表示。

#### 4.3 用简单选择法对 10 个整数排序。

解题思路：首先我们将主教材的算法提示按步分解。

(1) 使用数组存储这些整数，数组下标为  $i$  对应第  $i$  个元素，如下标 1 代表第 1 个元素，数组下标为 0 的元素不存储排序数；

(2) 从 10 个整数中找到最小的一个，将其与第一个元素交换；

(3) 从第二个到第十个元素中找到最小的一个，将其与第二个元素交换；

(4) 以此类推，每次从所有未排序元素中找到最小的元素，将其与未排序元素中的第一个交换；

(5) 共需要进行 9 次排序。

具体算法是：

(1) 用数组  $a$  存储待排序元素， $n$  存储待排序的元素的个数。

(2) 让  $i$  表示排序的遍数，其值为从 1 到  $n-1$ 。

(3) 在每次排序中，从  $a[i]$  到  $a[n]$  中找到最小元素，将其与  $a[i]$  交换。

(4) 找  $a[i]$  到  $a[n]$  中的最小值，使用以下的方法：令  $minValue$  表示最小值，初值为  $a[i]$ ，令  $minNum$  表示最小值的序号，初值为  $i$ ，让  $j$  从  $i$  到  $n$  循环；如果  $a[j]<minValue$ ，则更换  $minValue$  和  $minNum$ 。

参考程序：

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

```
const int n=10;//定义一个外部变量 n 表示待排序数的个数
```

```
main()
```

```
{
```

```
    int a[n+1]={0};//用数组存储待排序的数
```

```
    int i;//循环变量，表示排序遍数
```

```
    int j;//循环变量，表示要比较的元素下标
```

```
    int minValue=0;//存储最小值
```

```
    int minNum=0;//存储最小值的下标
```

```
    for(i=1;i<=n;i++)//输入 10 个数
```

```
    {
```

```
        printf("a[%d]",i);
```

```
scanf("%d",&a[i]);
}
for(i=1;i<n;i++)//循环, 第 i 遍排序
{
    minValue=a[i];
    minNum=i;
    for(j=i+1;j<=n;j++)//参考算法分析
        if(a[j]<minValue)
        {
            minValue=a[j];
            minNum=j;
        }
    a[0]=a[i];//以 a[0]为临时变量交换 a[i]与 a[minNum]
    a[i]=minValue;
    a[minNum]=a[0];
}
for(i=1;i<=n;i++)//输出结果
    printf("%d\t",a[i]);
printf("\n");
}
```

程序分析: 本题有两点需要注意。(1) 程序中第 23、24 行用于设置最小值及最小元素的下标的初始值, 在每次排序中, 都必须先对这两个变量赋值。再进行本遍扫描中的比较, 这两行语句初学者很容易放错位置, 比如放在第 20 行 for 语句之前, 这样将引起错误。(2) 数组 a 的下标 1 到下标 n 的位置用于存储待排序元素, 而下标 0 的位置并未存储待排序元素, 表面上是多用了—个存储单元, 但这样使用也有其好处: 一是数据元素的下标与其序号有更明显的对应关系, 用下标 0 对应第 1 个元素, 用下标 i 对应第 i+1 个元素不易理解, 且容易错误; 二是在进行交换时, 不需另外定义变量来作为临时变量, 而是直接以 a[0]来作临时变量。这对这段用于进行简单选择排序的程序的扩展性和可重用性是有利的。因此, 在实际使用中, 用数组存储一个序列时, 常采用这种方法, 将下标 0 的元素空置不用, 而用下标 i 对应第 i 个元素。当然, 在定义数组长度时, 其长度要比序列的长度大 1。

**4.4 所谓幻方, 就是一个 n 行 n 列的正方形, 当 n 为奇数时, 称为奇数阶幻方。共有  $n^2$  个格子, 将 1, 2, 3, ...,  $n^2$  这些数字放到这些格子里, 使其每行的和、每列的和及两条对角线的和都是一个相同的数。试编程由键盘输入一个奇数 n, 输出一个 n 阶幻方。**

解题思路: 首先我们将主教材的算法提示按步分解。

(1) 定义一个数组 a[n][n]来存储 n 阶幻方, 这里 n 为奇数, 数组各元素的初值均为 0, 表示该位置上还没有填数;

(2) 用一个整型变量 k 来表示当前要赋的数, 按题意 k 将开始从 1 到  $n^2$  循环, 因为 n 阶幻方将有  $n^2$  个数;



(3) 定义整型变量  $i, j$  表示要赋值的行号和列标,  $i$  初值为 0,  $j$  初值为  $n/2$ , 表示第一个数要放在第一行中间的那个格子里;

(4) 填数字开始,  $k$  开始循环, 完成下面的 (5) 到 (10);

(5) 先将  $k$  赋到相应的位置, 即将  $a[i][j]$  赋为  $k$ , 再将  $i, j$  的值分别用整型变量  $iold$  和  $jold$  暂时保存, 注意是暂时;

(6) 将  $i$  的值减 1,  $j$  的值增 1, 表示后一个数放在这个数的右上格;

(7) 如果  $i < 0$  且  $j < n$ , 即右上格从上面超出, 则  $i = n - 1$ , 表示后一数放到与右上格同列的最后一行;

(8) 如果  $i \geq 0$ , 且  $j \geq n$ , 即右上格从右面超出, 则  $j = 0$ , 表示后一数放到与右上格同行的第一列;

(9) 如果  $i < 0$  且  $j \geq n$ , 即右上格既从右面超出又从上面超出, 则将后一数放在前一数的下面, 即  $i = 1, j = n - 1$ ;

(10) 如果  $a[i][j] > 0$ , 即右上格已被数字填充, 则后一数放在前一数的下面, 即  $i = iold + 1, j = jold$ 。

参考程序:

```
#include<stdio.h>
const int n=5;//定义待幻方的阶次, 按题目要求, 应该为奇数, 暂定为 5
main()
{
    int a[n][n]={0}; //定义数组并初始化, 表示该幻方
    int k=1; //定义赋值的数, 从 1 到 25
    int i=0;
    int j=n/2;
    int iold,jold; //存储旧的行号和列标
    int maxk=n*n; //定义循环的结束值

    for(k=1;k<=maxk;k++) //对每一个要赋值的数循环
    {
        a[i][j]=k;
        iold=i;
        jold=j;
        i=i-1; //向上移一行
        j=j+1; //向右移一列, 以上两语句实现移到右上格
        if(i<0&&j<=n) //从右边超出
            j=0;
        else if(i<0&&j<n) //从上面超出
            i=n-1;
        else if(i<0&&j>=n) //从右和上同时超出
        {
```

```
        i=1;
        j=n-1;
    }
    if(a[i][j]>0)//右上格已经被填充
    {
        i=iold+1;
        j=jold;
    }
}
for(i=0;i<n;i++)//输出
{
    for(j=0;j<n;j++)
        printf("%d\t",a[i][j]);
    printf("\n");
}
```

输出结果即是书上的五阶幻方。

程序分析：本题看起来很难，初学者往往无从下手。但经过算法提示并仔细分析后发现，程序并不难。以下方法可以帮助大家解决较难的问题：

- (1) 仔细分析题目，提出一个合理的算法。
- (2) 将算法分解为独立的步骤，并用自然语言描述。
- (3) 将(2)中的结果用C语言进行设计。
- (4) 编译、执行和结果分析，如果算法不对，再重复执行以上过程。

另外，在输出语句中，也和一般的数组的输出语句不同，主要是{}的位置有所区别。所以，合理地使用{}也能对程序的运行结果有很大影响。

**4.5 输入一个字符串，要求按相反的顺序输出各个字符。例如，输入为 AbcD，则输出为 DcbA。**

算法分析：此题不难，只需要将字符数组反方向输出，但初学者需要注意的是必须先求出字符串的长度，这样才知道反方向输出的开始值。

参考程序：

```
#include<stdio.h>
#include<string.h>
main()
{
    char str[100]="";//定义字符数组，存储字符串
    int i;//定义循环变量
    int len;//字符串长度
```

```
printf("请输入字符串: ");
gets(str);//输入字符串
len=strlen(str);//求出字符串的长度
printf("\n 逆序输出的结果是: ");
for(i=len-1;i>=0;i--)//输出逆序字符
    putchar(str[i]);
printf("\n");
}
```

程序分析: strlen()函数包含在 string.h 中, 需要调用。

#### 4.6 输入一行字符, 统计其中包括多少单词, 单词之间用空格分隔。

参考程序:

```
#include<stdio.h>
#include<string.h>
main()
{
    char str[200];//定义字符数组, 存储字符串
    int i;
    int space=0;//空格标志, 0 表示新空格, 1 表示连续空格
    int num=0;//单词数量

    printf("请输入字符串: ");
    gets(str);
    if(str[0]==' '){//去掉第一行开头的空格
        space=1;
    }
    for(i=0;str[i]!='\0';i++)
    {
        if(str[i]==' '){//处理连续空格的情况, 当前字符为空格
            {
                if(space==0)//新空格
                {
                    space=1;//表示连续空格
                    num=num+1;
                }
            }
        }
        else
            space=0;//新空格
    }
    if(space==0)//如果字符串不以空格结束, 则单词数增 1
```

```
    num=num+1;
    printf("单词总数为: %d\n",num);//输出结果
}
```

程序分析: 本题容易出错的地方主要是对字符串前后空格的判断。

#### 4.7 编写一个学生管理系统 (以下省略)

算法分析: 这是结构数组的一个简单应用, 只需按顺序将学生的信息输入到结构数组中, 并在输入的同时判断并存储各门课程的平均成绩、最高分和最低分, 在输入完成后再按顺序判断并输出得到最高分的学生的信息。需注意的是结构数组的长度要大于学生人数的最大值, 以免出现数组越界错误。

参考程序:

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

struct date // 结构 date 用于表示日期信息
{
    int year; // 年
    int month; // 月
    int day; // 日
};

struct student // 定义结构 student 表示学生信息
{
    char name[20]; // 姓名
    int sex; // 性别
    struct date birthday; // 生日
    double height; // 身高
    double C; // C 语言成绩
    double Calculous; // 微积分成绩
};

const int MaxStudent=50; // 最大学生人数

void main() // 主函数
{ // 主函数开始
    student stus[MaxStudent+1]; // 定义 stus 用于存储所有学生的信息
    int n,i=0; // n 为学生人数, i 为循环变量
    double avgC=0,maxC=0,minC=100; // C 语言平均成绩、最高分、最低分
    double avgCal=0,maxCal=0,minCal=100; // 微积分平均成绩、最高分、最低分
    printf("请输入学生人数 (<=%d) : ",MaxStudent);scanf("%d",&n);
    for(i=0;i<n;i++) // 按顺序输入各同学的信息
    {
        printf("请输入第%d 位同学的信息: \n",stus[i+1]+1);
```

```

printf("姓名:\t");scanf("%s",stus[i].name); // 输入姓名
printf("性别(1 男 0 女):\t");scanf("%d",&stus[i].sex); // 输入性别
printf("生日:\n"); // 输入生日
printf("\t 年: ");scanf("%d",&stus[i].birthday.year);
printf("\t 月: ");scanf("%d",&stus[i].birthday.month);
printf("\t 日: ");scanf("%d",&stus[i].birthday.day);
printf("身高(m):\t");scanf("%lf",&stus[i].height); // 输入身高
printf("C 语言成绩:\t");scanf("%lf",&stus[i].C); // 输入 C 语言成绩
printf("微积分成绩:\t");scanf("%lf",&stus[i].Calculus); // 输入微积分成绩
avgC=avgC+stus[i].C; // 累加 C 的成绩
avgCal=avgCal+stus[i].Calculus; // 累加微积分的成绩
if(maxC<stus[i].C)maxC=stus[i].C; // 统计 C 语言的最高分
if(minC>stus[i].C)minC=stus[i].C; // 统计 C 语言的最低分
if(maxCal<stus[i].Calculus)maxCal=stus[i].Calculus;
if(maxCal>stus[i].Calculus)minCal=stus[i].Calculus;
}
avgC=avgC/n; // 计算 C 语言平均分
avgCal=avgCal/n; // 计算微积分平均分
printf("\n 课程\t%8s\t%8s\t%8s\n","平均分","最高分","最低分");
printf("C\t%8.4f\t%8.4f\t%8.4f\n",avgC,maxC,minC);
printf("最高分学生信息:\n");
printf("%10s\t 性别\t%14s\t%8s\t%8s\n","姓名","生日","C","微积分");
for(i=0;i<n;i++)
    if(fabs(stus[i].C-maxC)<=1e-6)
        printf("%10s\t%2d\t%4d 年%2d 月%2d 日\t%8.4f\t%8.4f\n",
            stus[i].name,stus[i].sex,stus[i].birthday.year,
            stus[i].birthday.month,stus[i].birthday.day,stus[i].C,
            stus[i].Calculus);
printf("\n 微积分\t%8.4f\t%8.4f\t%8.4f\n",avgCal,maxCal,minCal);
printf("最高分学生信息:\n");
printf("%10s\t 性别\t%14s\t%8s\t%8s\n","姓名","生日","C","微积分");
for(i=0;i<n;i++)
    if(fabs(stus[i].Calculus-maxCal)<=1e-6)
        printf("%10s\t%2d\t%4d 年%2d 月%2d 日\t%8.4f\t%8.4f\n",
            stus[i].name,stus[i].sex,stus[i].birthday.year,
            stus[i].birthday.month,stus[i].birthday.day,stus[i].C,
            stus[i].Calculus);
}

```

4.8 在用 C 语言编制图像处理程序时, 常使用结构来表示平面上的点。试编程求解下面的问题:

- (1) 输入两个点的坐标, 求这两点间的距离。
- (2) 输入三个点的坐标, 判断这三个点是否共线。

算法分析: 这道是结构的应用, 需要定义一个结果变量表示平面上的点, 由两个成员变量横坐标  $x$  和纵坐标  $y$  唯一确定这个点。

参考程序:

```
#include<stdio.h>
#include<math.h>
struct POINT//定义结构, 表示一个点
{
    double x;//横坐标
    double y;//纵坐标
};
void main()
{
    struct POINT p1,p2,p3;//定义 3 个点
    double distance=0;

    printf("请输入两个点的坐标: \n");
    printf("第一个点的坐标(x,y): \n");
    scanf("%lf,%lf",&p1.x,&p1.y);
    printf("第二个点的坐标(x,y):\n");
    scanf("%lf,%lf",&p2.x,&p2.y);

    distance=sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));//求两点距离
    printf("这两点的距离为: %f\n",distance);

    printf("请输入第三个点的坐标(x,y): \n");
    scanf("%f,%f",&p3.x,&p3.y);

    if(fabs(p3.x-p1.x)<1e-7&&fabs(p2.x-p1.x)<1e-7)//判断横坐标是否相等, 参考程序分析
        printf("三点共线: \n");
    if(fabs(p3.x-p1.x)<1e-7||fabs(p2.x-p1.x)<1e-7)
        printf("三点不共线: \n");
    else if(fabs((p3.y-p1.y)/(p3.x-p1.x) - (p2.y-p1.y)/(p2.x-p1.x))<1e-7)//直线方程
        printf("三点共线: \n");
    else
```

```
printf("三点不共线: \n");  
}
```

程序分析：本题需要注意以下几点：

- (1) `sqrt()`（开方）和 `fabs()`（求绝对值）均是数学函数，包含在头文件 `math.h` 中；
- (2) 判断三点共线时需要检查这三个点的横坐标是否相等，容易忽略；
- (3) 判断两个浮点数相等时，不能直接用 `==` 比较，而要使用这两个数之差的绝对值是否小于某一个很小的正数来判断，本题中使用的是 `1e-7`，请读者理解。

**4.9 用一个数组存放图书的信息，每本书包含书名、作者、出版年月、出版社、借出数目、库存数目等信息。编写程序存放若干本书的信息，按书名排序后输出。**

算法分析：虽然题目的要求是用数组，但实际上本题最关键的地方不仅是数组，还需要结构来表示每本书的信息，这些信息都是由不同类型的数据组成，定义的时候需要考虑变量类型。

参考程序：

```
#include<stdio.h>  
#include<string.h>  
const int MaxNum=10;//定义一个最大数量，暂定为 10  
struct BookInfo//定义一本书的信息  
{  
    char name[100];  
    char author[20];  
    char publishdate[20];  
    char publisher[100];  
    int brownum;  
    int stornum;  
};  
void main()  
{  
    struct BookInfo books[MaxNum+1];  
    int i;  
    int j;  
    int changed=0;//标志位  
  
    for(i=1;i<=MaxNum;i++)//输入各个书的信息  
    {  
        printf("\n 请输入第%d 本书的信息: \n",i);  
        printf("书名: \n");  
        gets(books[i].name);  
    }
```

```
printf("作者: \n");
gets(books[i].author);
printf("出版日期: \n");
gets(books[i].publishdate);
printf("出版社: \n");
gets(books[i].publisher);
printf("借出数量: \n");
scanf("%d",&books[i].brownum);
printf("库存数量: \n");
scanf("%d",&books[i].stornum);
}

for(i=1;i<MaxNum;i++)
{
    changed=0;//初始本次循环未交换
    for(j=1;j<MaxNum-1;j++)
        if(strcmp(books[j].name,books[j+1].name)>0)//比较, 如果逆序则交换
        {
            books[0]=books[j];
            books[j]=books[j+1];
            books[j+1]=books[0];
            changed=1;//交换标志
        }
    if(changed==0)
        break;
}

for(i=1;i<=MaxNum;i++)//按交换后的结果输出
{
    printf("书名:\t%s\n",books[i].name);
    printf("作者:\t%s\n",books[i].author);
    printf("出版日期:\t%s\n",books[i].publishdate);
    printf("出版社:\t%s\n",books[i].publisher);
    printf("借出数量:\t%d\n",books[i].brownum);
    printf("库存数量:\t%d\n",books[i].stornum);
}
}
```



程序分析：本题有一定的难度，综合数组、结构和前面各章的知识，还应用了很多本章节题目算法和思路，希望大家多思考。

4.10 在某次实弹射击训练中，班长、战士分子弹问题。（以下省略）

参考程序：

```
#include<stdio.h>
#include<string.h>
const int n=10;//战士人数
main()
{
    int a[n]={10,2,8,22,16,4,10,6,14,20};//初始化数组，用于存储战士的子弹
    int i,k=0;//循环变量
    int x;//临时变量，用于暂存 a[n-1]

    printf("%2d",k);//输出初始的战士子弹数
    for(i=0;i<n;i++)
        printf("%4d",a[i]);
    printf("\n");
    do//循环，直到所有战士子弹数相同
    {
        for(i=0;i<n;i++)//检查是否为偶数
            if(a[i]%2==1)
                a[i]++;
        x=a[n-1];
        for(i=n-2;i>=0;i--)
            a[i+1]=(a[i]+a[i+1])/2;//按要求分子弹
        a[0]=(a[0]+x)/2;
        k+=1;
        printf("%2d",k);//输出每次分配后的结果
        for(i=0;i<n;i++)
            printf("%4d",a[i]);
        printf("\n");
    } while(!(a[0]==a[1]&&a[1]==a[2]&&a[2]==a[3]&&a[3]==a[4]&&a[4]==a[5]&&
        a[5]==a[6]&&a[6]==a[7]&&a[7]==a[8]&&a[8]==a[9]));
}
```

运行结果为：

```

C:\Documents and Settings\Administrator\My Documents\Debug\Cp
0 10 2 8 22 16 4 10 6 14 20
1 15 6 5 15 19 10 7 8 10 17
2 17 11 6 11 18 15 9 8 9 14
3 16 15 9 9 15 17 13 9 9 12
4 14 16 13 10 13 17 16 12 10 11
5 13 15 15 12 12 16 17 14 11 11
6 13 15 16 14 12 14 17 16 13 12
7 13 15 16 15 13 13 16 17 15 13
8 14 15 16 16 15 14 15 17 17 15
9 15 15 16 16 16 15 15 17 18 17
10 17 16 16 16 16 16 16 17 18 18
11 18 17 16 16 16 16 16 17 18 18
12 18 18 17 16 16 16 16 17 18 18
13 18 18 18 17 16 16 16 17 18 18
14 18 18 18 18 17 16 16 17 18 18
15 18 18 18 18 18 17 16 17 18 18
16 18 18 18 18 18 18 17 17 18 18
17 18 18 18 18 18 18 18 18 18 18
Press any key to continue

```

程序分析：本题在程序设计上难度不大，难点在于算法的设计，特别是对战士分配子弹数这个过程的实现。

一种较简单的方法是多用一个数组来保存上一次分配后的子弹数，则下一次的分配就只是一个简单的赋值而已。这个方法比较容易实现，请读者思考。但这种算法多用了一倍的空间来进行存储，这样不满足 C 程序高效性和扩展性的特点。

作者使用的这种算法只多用了—个整数存储单元，先将最后一位战士的子弹数保存下来，再按照分配原则应将最后一位战士的子弹重新分配，然后是倒数第二位依次向前至第二位战士，因为对这些战士分配子弹时都只需他自己的子弹数和他前一位战士的子弹数即可。对第一位战士的分配特殊一些，他分配后的子弹数与最后一位战士的子弹数也有关系，由于前面对最后一位战士的子弹数进行了修改，因此，必须先暂存起来，否则对第一位战士的分配就将出现错误。

#### 4.11 保龄球计分问题。（以下省略）

算法分析：本题可以对轮次进行循环，依次求出一局中各轮的分数。在每一轮中如果第一次滚出 10 分，则其得分与下两次滚球的分数相关；如果第一次未滚出 10 分，则两次滚得的分数之和如果不为 10，则本轮得分即为两次滚得的分数之和；否则，本轮得分与下一轮的得分有关。因为—轮的得分与其是 strike、spare 还是 normal 的状态有关，因此需用变量存储。又因为—轮的得分最多只与其后两轮的得分有关，因此只需用两个变量来存储当前轮的前两轮的状态即可。

因此各轮得分的计算方法是，先求出第 1、2 轮的暂时得分和状态，再从第 3 轮开始，依次计算本轮前两轮的得分及本轮的暂时得分和状态，当然如果本轮的状态为 normal，则其得分马上可以得到，前两轮的得分也可以确定。

对第 10 轮，因为可以滚 3 次球，因此其得分为如果第一次得 10 分，或前两次共得 10 分，则最后得分为三次得分之和，否则最后得分为两次得分之和。

参考程序：

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

```
const int normal=1;//定义三种状态
const int spare=2;
const int strike=3;
void main()
{
    int game[10]={0};//数组存储各轮得分
    int state1=0,state2=0;//上两轮的状态
    int i,score,state;//i 为循环变量，score 存储当前得分，state 为本轮状态
    int totalscore=0;//总得分

    for(i=0;i<10;i++)//计算每轮得分
    {
        scanf("%d",&score);//第一次击球分数
        if(score<0&&score>10)//保证输入的正确性
            return;
        game[i]=score;

        if(state2==strike&&state1==strike)
            game[i-2]=game[i-2]+score;//求上两轮的得分
        if(state1>=spare)//根据上一轮的状态累加得到上一轮的得分
            game[i-1]=game[i-1]+score;
        if(score==10)//如果第一轮全中，则无第二次
            state=strike;//保持本轮状态
        else//第二次击球
        {
            scanf("%d",&score);
            if(score<0&&score>10)
                return;
            game[i]=game[i]+score;//累加得到本轮暂时得分
            if(game[i]>10)
                return;
            if(game[i]==10)
                state=spare;
            else
                state=normal;
            if(state1==strike)//根据上一轮的状态累加得到上一轮的得分
                game[i-1]=game[i-1]+score;
        }
        state2=state1;
```

```
        state1=state;
    }
    if(state1==spare)//根据第 10 轮的当前状态求得到最后两轮的得分
    {
        scanf("%d",&score);
        if(score<0&&score>10)
            return;
        game[9]=game[9]+score;
    }
    else if(state1==strike)
    {
        scanf("%d",&score);
        if(score<0&&score>10)
            return;
        game[9]=game[9]+score;
        if(state2==strike)
            game[8]=game[8]+score;
        scanf("%d",&score);
        if(score<0&&score>10)
            return;
        game[9]=game[9]+score;
    }
    for(i=0;i<10;i++)//求总得分
    {
        printf("%d\t",game[i]);
        totalscore+=game[i];
    }
    printf("%d\n",totalscore);
}
```

程序分析：本题难度较大，不仅条件较多，而且第 10 轮的情况还比较特殊。请读者仔细理解并掌握。

#### 4.12 素数排列问题。（以下省略）

参考程序：

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

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

```
const int n=2000;//定义素数的范围
```

```
const int num=1898;//累加和的值
```

```
void main()
```

```
{
    int prime[n+1]={0};//表示 2000 内的素数, 0 表示素数
    int i,j,MaxGene;//i, j 循环变量, MaxGene 为最大可能正因子
    int totalprime=1;//素数数量
    int primediff[n/2]={0};//素数之差
    int primesum=0;//累加和

    MaxGene=(int)sqrt(n);//最大正因子为 n 的平方根
    for(i=2;i<=MaxGene;i++)//枚举所有正因子
        if(prime[i]==0)
            for(j=2*i;j<=n;j=j+i)//去除正因子倍数
                prime[j]=1;
    prime[0]=2;//保存素数的值
    for(i=3;i<=n;i++)//求素数的差, 并保存到 primediff 中
        if(prime[i]==0)
        {
            primediff[totalprime-1]=i-prime[totalprime-1];//求素数的差
            prime[totalprime]=i;//保存素数的值
            totalprime=totalprime+1;//累加素数数量
        }
    for(i=0;i<totalprime;i++)//枚举素数差
    {
        primesum=0;
        for(j=i;j<totalprime&&primesum<num;j++)
            primesum=primesum+primediff[j];
        if(primesum==num)//满足条件
            printf("%d 与%d 之间的素数差之和为%d\n",prime[i],prime[j],num);
    }
}
```

程序结果:

```
C:\Documents and Settings\Administrator\My...
3与1901之间的素数差之和为1898
53与1951之间的素数差之和为1898
89与1987之间的素数差之和为1898
101与1999之间的素数差之和为1898
Press any key to continue.
```

## 编程好习惯四：先动脑再动手

本章习题的难度较大,经过练习后大家可能会发现,语法错误已经不再是编程难点,一个优秀的算法才是设计一个完整的程序最重要的。在前面我们已经了解到,程序错误主要有两种:语法错误和逻辑(算法)错误。前者很容易被发现和解决,但后者却不同,一旦出现,修改的难度较大,特别对于大型的程序,算法错误是致命的。所以在动手编程前,应该先对算法进行思考,用数学或者自然语言设计好算法,验证了正确性后再进行实际的编程。如果出现了较难的算法,还可以参考一些成功的例程,这样将很大程度上减少逻辑和算法错误。

### 6.5 习题 5

#### 5.1 请指出以下程序段中的错误。

程序中的错误有:

- (1) `p=i`: 类型不匹配。
- (2) `q=*p`: `q` 是指针, `*p` 是指针 `p` 指向变量的值。
- (3) `t='b'`: `t` 是指针类型。

解释: 指针变量是一种存放地址的特殊变量,其特殊性表现在类型和值上。指针变量的类型是指针变量所指向的变量的类型,而不是自身的类型。指针变量赋值应该是地址值。详细内容请参考主教材 5.1 节。

正确程序应为:

```
main ()
{
    int i,j,*p,*q;
    char ch1,ch2,*t,*s;

    i=3;
    p=&i;
    j=*p/2+10;
    q=p;
    ch1='a';
    s=&ch1;l
    *s='c';
    *t='b';
    ch2=*t;
}
```

## 5.2 以下程序的输出结果是什么？

```
main()
{
    char *point[]={ "one", "two", "three", "four" };

    while(*point[2]!='\0')
        printf("%c", *point[2]++);
}
```

答：输出结果是 three。

结果分析：指针数组是指数组的每一个元素都是一个指针变量的数组，定义了指针数组 pd，它由 pd[0]~pd[3] 4 个数组元素组成。

## 5.3 以下程序的输出结果是什么？

```
main()
{
    char *point[]={ "one", "two", "three", "four" };

    point[2]=point[0];
    printf("%s", point[2]++);
}
```

答：输出结果是 one。原因参考习题 5.2。

## 5.4 请对以下程序进行修改，用指针完成对数组元素的访问：

```
main()
{
    int data[12]={12,34,56,12,34,56,3,54,6,7,89,12};
    int i,sum;

    sum=0;
    for(i=0;i<12;i++)
        sum+=data[i];
    printf("The sum is %d\n",sum);
}
```

算法分析：本题需要抓住数组和指针的联系，讨论数组时，对数组元素的访问是采用的下标法，即是以数组的下标来确定数组元素的。在引入指针变量后，我们可以利用一个指向数组的指针来完成对数组元素的存取操作及其他运算，这种方法称为指针法。

修改后的程序为：

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

```
main ()
{
    int data[12]={12,34,56,12,34,56,3,54,6,7,89,12};
    int i,sum;
    int *p;

    sum=0;
    p=data;
    for (i=0;i<12;i++)
    {
        sum+=*p;
        p++;
    }
    printf ("The sum is %d\n", sum) ;
}
```

### 5.5 指出并更正以下程序的错误。

```
main ()
{
    char data[]="There are some mistakes in the program";
    char *point;
    char array[30];
    int i,length;

    length=0;
    while(data[length]!='\0')
        length++;
    for(i=0;i<length;i++,point++)
        *point=data[i];
    array=point;
    printf("%s\n",array);
}
```

经过分析，我们可以看出程序的目的是将这个字符串输出，原程序中的主要错误是在指针运算前没有赋初始地址，于是进行以下修改：

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



```

char data[]="there are some mistakes in the program";
char *point;
char array[100];//定义一个数组
int i,length;

length=0;
while(data[length]!='\0')//求字符串的长度
    length++;
point=array;//给指针赋首地址，主要修改部分
for(i=0;i<=length;i++,point++)//输出字符串
    *point=data[i];
printf("%s\n",array);
}

```

5.6 编写一个程序输入两个字符串 string1 和 string2，检查在 string1 中是否包含有 string2。如果有，则输出 string2 在 string1 中的起始位置；如果没有，则显示“NO”；如果 string2 在 string1 中多次出现，则输出在 string1 中出现的次数以及每次出现的起始位置，例如：

```
string1="the day the month the year";
```

```
string2="the"
```

输出结果应为：出现三次，起始位置分别是：0,8,18

又如：

```
string1="aaabacad"
```

```
string2="a"
```

输出结果应为：出现五次，起始位置分别是 0,1,2,4,6

算法分析：本题需要对两个字符串进行循环，但与普通循环不同的是，这两个循环并不是同时进行的。string1 开始循环后，string2 并不一定开始循环，而是需要判断 string2 的首字母是否和 string1 中循环到的字母相同，如果相同，则两个字符串开始同时循环，并记录下开始的位置，然后判断 string2 中后面的字母是否和 string1 中的相同，相同则继续，不同则停止，直到 string2 循环完成。此时用当前的位置减去 string2 的长度就是 string2 的起始位置。在本题中，还需要注意的是，由于 string2 可能多次出现，所以需要有一个数组来保存多次出现的位置。

参考程序：

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

```
main()
```

```
{
```

```
    char string1[100],string2[10];//定义数组存储两个字符串，暂定第一个在 100 内，第二个在 10 内
```

```
    char *p,*q;//定义指针遍历字符串
```

```
int locat[10];//定义数组保存出现的位置
int j,len2,i=0,posit=0;//len2 表示第二个字符串的长度

printf("请输入字符串 1:\n");
gets(string1);
printf("请输入字符串 2:\n");
gets(string2);
q=string2;
for(j=0;*q!='\0';j++,q++)//求第二个字符串的长度
    len2=j+1;
p=string1;//指针赋首地址
q=string2;
do//循环进行判断
{
    if(*p!=*q)//字符不相同的情况
    {
        p++;
        posit++;
    }
    else//第一个字符相同，继续对后面字符进行判断
    {
        while((*q!='\0')&&(*q==*p))
        {
            q++;
            p++;
            posit++;
        }
        if(*q=='\0')//第二个字符串循环结束
        {
            locat[i]=posit-len2;
            i++;
        }
    }
    q=string2;//参考程序分析
}while(*p!='\0');
printf("字符串 1:%s\n 字符串 2:%s\n",string1,string2);
printf("出现%d 次，起始位置分别是:",i);
for(j=0;j<i;j++)//输出位置
    printf("%d",locat[j]);
```

```
    printf("\n");
}
```

程序分析: `q=string2` 这句的作用是什么呢? 是保证 `q` 循环后能再次初始化, 这样才能进行多次判断, 并且防止错误。例如: `q` 和 `p` 的第 1 个、第 2 个字符相同, 但第 3 个不同, 此时 `q` 指针已经指向了第 3 个字符, 如果不再次初始化, 下一步比较就将从 `q` 的第 3 个字符开始, 出现错误。

### 5.7 给定一个整数数组:

```
num[]={23,45,345,23};
```

请定义一个指针变量 `point`, 并令它指向数组的第一个元素, 然后回答以下问题:

- (1) `num[2]` 的值等于什么?
- (2) `*(point+2)` 的值等于什么?
- (3) `*++point` 的值等于什么?

答: (1) 345, 即是数组的第 3 个元素。

(2) 345, 指针指向了数组的第 3 个元素。

(3) 45, 指针指向了数组的第 2 个元素。

### 5.8 修改主教材例 5-22 中的程序, 令其按 ASCII 码的方式显示内存单元的内容。

程序分析: 本题和主教材上例 5-22 的区别主要在输出的类型上, 所以修改较简单。

参考程序:

```
#include <stdio.h>

main()
{
    char *point;
    long int b_addr,e_addr,i,j;

    printf("please enter the beginning and the end addr in hex\n");
    scanf("%lx%lx",&b_addr,&e_addr);
    for(i=b_addr; i<e_addr; i+=16)
    {
        printf("%05lx: ",i);
        point=(char*)i;
        for(j=0;j<16;j++)
        {
            if(j==8)
                printf(" ");
            printf("%3d ",*point);
            point++;
        }
    }
}
```

```
    printf("\n");  
}  
}
```

### 5.9 编写程序输入一个字符串，分别统计输出该字符串中的字母个数和数字个数。

算法分析：判断字母和数字的核心方法是依靠 ASCII 码进行，所以对字符串中的每个字符逐个判断即可得到结果。实现方法依然是依靠指针。

参考程序：

```
#include<stdio.h>  
main()  
{  
    char str[100];//定义一个数组存储字符串  
    char *p;  
    int n_count=0,c_count=0;  
  
    p=str;  
    printf("请输入一个字符串： \n");  
    gets(str);//输入字符串  
    do  
    {  
        if((*p>='0')&&(*p<='9'))//判断数字  
            n_count++;  
        if((*p>='a')&&(*p<='z'))//判断小写字母  
            c_count++;  
        if((*p>='A')&&(*p<='Z'))//判断大写字母  
            c_count++;  
        p++;  
    }while(*p!='\0');  
    printf("字符串%s 中的字母个数是： %d， 数字个数是%d",str,c_count,n_count);  
}
```

程序分析： 本题需要注意的有两点：

1. char 类型的变量，初学者往往理解为是不能直接比较大小的，实际上直接比较大小时，是将类型转换为了 int 后比较的 ASCII 码值。这样的方法相对与手动转换为 ASCII 码比较而言更加高效和方便，希望读者掌握。

2. 字母包含大小写，这点容易遗漏，在程序设计时应该考虑更全面。

另外，这道题还可以扩展为求数字、字母和其他字符的数量，请读者思考。

## 5.10 以下程序的输出结果是什么？

```
main()
{
    char *point[]=
    {
        "111111111",
        "222222222",
        "333333333",
        "444444444",
        "555555555"
    };
    int i,j;

    for(i=1;i<3;i++)
    {
        for(j=1;j<5;j++)
            printf("%c",*(point[j]+i));
        printf("\n");
    }
}
```

答：输出结果为：2 3 4 5

2 3 4 5

结果分析：此题较简单，参考主教材 5.5 节指针数组的内容。

## 5.11 编写一个程序，输入两个字符串，比较它们是否相等。

算法和习题 5.6 类似。

参考程序：

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

```
main()
```

```
{
```

```
    char str1[100],str2[100];
```

```
    char *p,*q;
```

```
    printf("请输入第一个字符串:\n");
```

```
    gets(str1);
```

```
    printf("请输入第二个字符串:\n");
```

```
    gets(str2);
```

```
p=str1;
q=str2;
if(*p!=*q)
    printf("两个字符串不相等。");
else
{
    while((*p==*q)&&(*p!='\0')&&(*q!='\0'))//循环判断是否每个字符相等
    {
        p++;
        q++;
    }
    if((*p=='\0')&&(*q=='\0'))
        printf("两个字符串相等: ");
    else
        printf("两个字符串不相等: ");
}
}
```

## 编程好习惯五：多写注释

一段优秀的程序不仅包括算法和程序语句，还包括正确和全面的注释。注释的作用是告诉阅读者这段程序的作用或者目的，注释是说明你的代码做些什么，而不是怎么做的，而且，要试图将注释放在一个合理的地方。通常在一段程序的关键部分，如变量定义、函数定义、循环和核心算法等部分增加注释，能大大增强程序的可读性，对于程序的修改和扩展都很有帮助。

在本书中，除了一些比较简单的程序和语句，笔者在大多数程序中，都使用了注释。

## 6.6 习题 6

6.1 写一个函数，使其能将一个二维数组（5×3）中的数据进行行列互换（参考函数原型：void tran(int array[5][3])）。

算法分析：本题的关键是用两层循环完成对二维数组的转秩，较简单。

参考程序为：

```
#include<stdio.h>
int b[3][5];
void tran(int array[5][3])//定义转秩函数，无返回值
```

```
{
    int i,j;
    for(i=0;i<5;i++)
        for(j=0;j<3;j++)
            b[j][i]=array[i][j];//交换
}
main()
{
    int i,j;
    int a[5][3]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
    printf("交换前是: \n");
    for(i=0;i<5;i++)
    {
        for(j=0;j<3;j++)
            printf("%4d",a[i][j]);
        printf("\n");
    }

    tran(a);//调用交换函数实现对数组 a 的转秩

    printf("\n 交换后是: \n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<5;j++)
            printf("%4d",b[i][j]);
        printf("\n");
    }
}
```

需要说明的是：在一些编译器中循环语句还可以这样实现“for(int i=0;i<n;i++)”。也就是将循环变量的定义放到循环体中。本题的函数实现的功能并无返回值，所以函数定义为 void，且不需要再使用 return 返回结果。

**6.2 写一个函数，使其能统计主调函数通过实参传送来的字符串，对其中的字母、数字和空格分别计数（要求在主函数中输入字符串及输出统计结果）（参考函数原型：void count(char\* str)）。**

算法分析：在经过了习题 5.9 的练习后，我们已经掌握了判断字母和数字的方法，本题只是将判断的过程放在函数中实现。

参考程序：

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

```
int b[3];//定义外部变量，用于存储各种字符的个数
void count(char *str)//定义统计函数
{
    while((*str)!=0)
    {
        if(('a'<=*str&&*str<='z')||('A'<=*str&&*str<='Z'))//求字符个数
            b[0]++;
        if('0'<=*str&&*str<='9')//求数字个数
            b[1]++;
        if(*str==' ')//空格个数
            b[2]++;
        str++;
    }
}
void main()
{
    char str[100];
    printf("请输入字符串: \n");
    gets(str);
    printf("字符串%s 中",str);
    count(str);//调用统计函数
    printf("\n 字母个数:%d",b[0]);
    printf("\n 数字个数:%d",b[1]);
    printf("\n 其空格个数:%d",b[2]);
}
```

程序分析：本题需要注意的是定义了一个外部数组变量 b[3]用于存储各种字符的个数。外部变量可以被全部函数共同访问，所以就不需要在统计函数和主函数中分开定义各自的统计变量，更加方便和节省空间。请掌握这样的方法。

**6.3 写一个函数，使其能处理字符串中除字母（大小写）、数字外的其他 ASCII 字符，对多于一个连在一起的相同字符，将其缩减至仅保留一个。（参考函数原型：void del(char\* str)）**

算法分析：本题判断方法和习题 6.2 类似。难点在于处理连在一起的相同字符。首先定义一个字符数组 str2[ ]，用于存储判断处理后的字符串。字符串 str 开始循环后，如果是字母或者数字，就赋给 str2；如果是其他字符，先判断是否为连续，再进行赋值。连续的情况需要特殊处理，循环直到是其他字符后再赋值。

参考程序：

```
#include<stdio.h>
char str2[80];//定义一个数组存储字符串
```



```
void del(char *str)//定义函数
{
    int i=0;
    while((*str)!=0)
    {
        if(('a'<=*str&&<='z')||('A'<=*str&&<='Z')||('0'<=*str&&<='9'))//判断 ASCII 码
        {
            str2[i]=*str;
            str++;
            i++;
        }
        else//处理连在一起的相同字符
        {
            while(*str==*(str+1))
                str++;
            str2[i]=*str;
            str++;
            i++;
        }
    }
}

void main()
{
    char str[80];
    printf("请输入字符串:");
    gets(str);
    del(str);
    printf("\n 新字符串: %s\n",str2);
}
```

6.4 设有一个 3 位数，将它的百、十和个位 3 个单一数，各自求立方，然后加起来，正好等于这个 3 位数，如  $153=1+125+27$ 。写一个函数，找出所有满足条件的数。（参考函数原型：int find(int n)）

参考程序：

```
#include<stdio.h>
int find(int n)
{
```

```
int i,j,k;//i 表示百位, j 表示十位, k 表示个位
int flag;//标志位, 是否满足条件

i=n/100;//求百位
j=n/10-i*10;//求十位
k=n%10;//求个位
if(n==i*i*i+j*j*j+k*k*k)//判断
    flag=1;
else
    flag=0;
return flag;
}
```

```
void main()
{
    int f,n;
    printf("符合条件的数有: ");
    for(n=100;n<1000;n++)//循环判断所有的三位数
    {
        f=find(n);
        if(f)
            printf("%4d",n);
    }
    printf("\n");
}
```

运行结果: 153 370 371 407

6.5 写一个函数, 使其能求出一元二次方程的解。(参考函数原型: void s(int a,int b,int c), a、b、c 分别代表一元二次方程的系数)

参考程序:

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

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

```
float x1,x2;
```

```
void s(int a,int b,int c)
```

```
{
    int s;
    s=b*b-4*a*c;
    x1=(sqrt(s)-b)/(2*a);//求两个解
```

```
    x2=(-sqrt(s)-b)/(2*a);
}
main()
{
    int a,b,c;

    printf("请输入一元二次方程的系数: ");
    scanf("%d,%d,%d",&a,&b,&c);
    if(a==0)//去掉 a 为 0 的不合法方程
    {
        printf("a 不能为 0! \n");
    }
    else
    {
        s(a,b,c);
        printf("方程的解为: %f 和%f\n",x1,x2);
    }
}
```

6.6 写一个程序，从键盘输入 5 个正整数，然后求出它们的最小公倍数，并显示输出（通过调用对两个正整数求最小公倍数的函数实现）（参考函数原型：int find(int i,int j)）。

参考程序：

```
#include<stdio.h>
int find(int i,int j)
{
    int temp,a,b;
    if(i<j)//保证用较大的数除以较小的数
    {
        temp=i;
        i=j;
        j=temp;
    }
    a=i;
    b=j;
    while(b!=0)//辗转相除
    {
        temp=a%b;
        a=b;
        b=temp;
    }
}
```

```

    }
    return(i*j/a);
}
main()
{
    int a[5],b,i;
    printf("请输入 5 个整数: ");
    for(i=0;i<5;i++)
        scanf("%d",&a[i]);
    b=find(a[0],a[1]);//调用函数对 5 个整数进行处理
    b=find(b,a[2]);
    b=find(b,a[3]);
    b=find(b,a[4]);
    printf("5 个整数的最小公倍数是%d\n",b);
}

```

6.7 如果一个数正好是它的所有约数（除了它本身以外）的和，此数称完备数，如：6，它的约数有 1、2、3，并且  $1+2+3=6$ 。求出 30 000 以内所有的完备数，并显示输出（求完备数用函数实现）（参考函数原型：void find(int j)，直接在子函数中输出完备数及其所有约数）。

算法分析：对于整数 j，要求出所有的约数，需要对小于 j 的所有数进行循环。在每求出一个约数的同时，j 减去这个约数，再将结果赋给 j，如果是完备数，最后的结果将是 0。需要注意的是，j 的值在不断变化，为了保证 j 一直用初值进行求约数操作，所以首先将 j 的值赋给一个中间变量 s，利用 s 来进行减操作，让两步操作不相互影响。

参考程序：

```

#include<stdio.h>
int k[20];
void find(int j)
{
    int i,n,s;//i 为因子
    n=1;
    s=j;
    for(i=1;i<j;i++)
    {
        if((j%i)==0)
        {
            n++;
            s=s-i;//判断约数的和
            k[n]=i;//将每个因子赋给数组 k
        }
    }
}

```

```

    }
}
if(s==0)
{
    printf("%d 是一个完备数，它的因子是：",j);
    for(i=0;i<n;i++)
        printf("%6d",k[i]);//输出
    printf("%6d\n",k[n]);
}
}
main()
{
    int n;
    printf("符合条件的数有：");
    for(n=1;n<30000;n++)
        find(n);
}

```

运行结果为：

```

C:\Documents and Settings\Administrator\My Documents\Debug\Cpp1.exe
符合条件的数有：6是一个完备数，它的因子是： 1 2 3
28是一个完备数，它的因子是： 1 2 4 7 14
496是一个完备数，它的因子是： 1 2 4 8 16 31 62 124
8128是一个完备数，它的因子是： 1 2 4 8 16 32 64 127
254 508 1016 2032 4064
Press any key to continue

```

6.8 如果有两个数，每一个数它的所有约数（除了它本身以外）的和正好等于对方，则称这两个数为互满数，求出 30 000 以内所有的互满数，并显示输出。求一个数它的所有约数（除了它本身以外）的和用函数实现（参考函数原型：int factor(int j)）。

参考程序：

```

#include "stdafx.h"
#include<stdio.h>
int k[40];
#define N 30000
int factor(int j)
{
    int i,n,s;
    n=-1;
    s=j;
    for (i=1;i<j;i++)
    {

```

```

        if((j%i)==0)
        {
            n++;
            s=s-i;
            k[n]=i; // 将每个因子赋给数组K
        }
    }
    s=1;
    for (i=1;i<=n;i++)s+=k[i]; // 将每个因子累加
    return s;
}

void main()
{
    int s1,s2;
    for (int m=2;m<N;m++)
    {
        s1=factor(m);
        for (int n=m+1;n<N;n++)
        {
            s2=factor(n);
            if(s1==s2)printf("%d和%d是互满数\n",m,n);
        }
    }
}

```

程序分析：本题和习题 6.7 类似，区别在于调用两次函数得到结果后进行比较。

### 6.9 函数实现将输入的一组数据逆序输出（参考函数原型：void isort(int a[]））。

算法分析：长度为  $n$  的数组  $a[n]$  逆序的实现方法就是将  $a[i]$  和  $a[n-i-1]$  交换。

参考程序：

```

#include<stdio.h>
#define N 5
void isort(int a[])
{
    int i,temp;
    for(i=0;i<N/2;i++)//交换
    {
        temp=a[i];
        a[i]=a[N-i-1];
    }
}

```

```
        a[N-i-1]=temp;
    }
}
main()
{
    int a[5],i;
    printf("请输入%d 个数据: ",N);
    for(i=0;i<N;i++)
        scanf("%d",&a[i]);
    isort(a);
    printf("\n 逆序输出的结果是: ");
    for(i=0;i<N;i++)
        printf("%d",a[i]);
    printf("\n");
}
```

6.10 一个数组，内放 10 个整数，要求编写一函数找出最小的数和它的下标（参考函数原型：int min(int a[]),int minlocat(int a[])）。

算法分析：因为要求最小数及其下标，所以需要两个函数来实现。

参考程序：

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

```
int min(int a[])/求最小值
```

```
{
    int i,min=32768;

    for(i=0;i<10;i++)
    {
        if(a[i]<min)
            min=a[i];
    }
    return min;
}
```

```
int minlocat(int a[])/求最小值的位置
```

```
{
    int i,min=32768,locat;

    for(i=0;i<10;i++)
    {
        if(a[i]<min)
```

```
        {
            min=a[i];
            locat=i;
        }
    }
    return locat;
}
main()
{
    int a[10];
    int b,c;
    int i;

    printf("请输入 10 个整数: ");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    b=min(a);
    c=minlocat(a);
    printf("最小数为%d, 位置为%d",b,c+1);
}
```

程序分析：本算法仅是针对最小值不重复出现的情况，请读者思考，重复出现的情况应该怎么解决。

6.11 编写一函数，将两个字符串比较大小（不用标准库函数）（参考函数原型：int cmp(char s1[],char s2[]））。

参考程序为：

```
#include<stdio.h>
#define N 100
int cmp(char s1[],char s2[])
{
    int i=0,v;//v 为二者比较的结果
    while((s1[i]==s2[i])&&(s1[i]!='\0'))
        i++;
    if(s1[i]=='\0'&&s2[i]=='\0')//两字符串相同
        v=0;
    else//两字符串不同
        v=s1[i]-s2[i];
    return v;
}
```



```
main()
{
    char s1[100],s2[100];
    int v;

    printf("输入字符串 1: \n");
    gets(s1);
    printf("输入字符串 2: \n");
    gets(s2);
    v=cmp(s1,s2);
    printf("比较的结果是%d\n",v);
}
```

程序分析：关于字符串比较大小请参考主教材附录。

**6.12 编写一函数，实现两个字符串的复制（不用标准库函数）（参考函数原型：void copy(char s1[],char s2[]）。**

算法分析：本题较简单，将两个字符串保存在两个数组中，根据对字符串的循环逐个完成赋值。

参考程序：

```
#include<stdio.h>
#include<string.h>
void copy(char s1[],char s2[])
{
    int i;
    for(i=0;i<=strlen(s1);i++)
        s2[i]=s1[i];//逐个对字符赋值
}
```

```
main()
{
    char s1[100],s2[100];
    printf("请输入字符串: ");
    gets(s1);
    copy(s1,s2);
    printf("复制后的字符串为:%s\n",s2);
}
```

**6.13 从键盘输入 10 名学生的成绩，显示其中的最低分、最高分及平均成绩。求最低分、最高分及平均成绩利用子函数实现（参考函数原型：void minmax(int s[ ]）。**

算法分析：在习题 6.10 中，针对求多个结果的情况，我们采用的是用多个函数来实现。

在本题中，我们尝试使用一个函数来完成求 3 个值的操作，需要注意的是存储这 3 个值的变量都是外部变量，这样将大大减少程序的复杂度。

参考程序：

```
#include<stdio.h>
#define N 10
int mina,maxa;
float av;
void minmax(int s[])
{
    int i;
    float avs;
    mina=s[0];maxa=s[0];
    avs=s[0];//参考程序分析

    for(i=1;i<N;i++)
    {
        if(mina>s[i])
            mina=s[i];
        if(maxa<s[i])
            maxa=s[i];
        avs+=s[i];
        av=avs/N;
    }
}
main()
{
    int s[N],i;
    printf("请输入%d 个学生的成绩：",N);
    for(i=0;i<N;i++)
        scanf("%d",&s[i]);
    minmax(s);
    printf("最高分为： %d，最低分为： %d，平均分为： %f\n",maxa,mina,av);
}
```

程序分析：作者注释的这句话往往被很多初学者忽略，为什么需要给 avs 赋初值呢？因为在后面有行语句 avs+=s[i]，如果 avs 没有初值，编译器将会随机给 avs 赋初值来完成与 s[i] 的赋值运算，造成结果错误。

6.14 在总数为  $n$  的对象中, 任意取  $p$  个的不同组合, 可用  $C(p,n)$  来表示, 其中  $p \leq n$ 。写一个程序, 在给出  $n$  和  $p$  的情况下, 计算并输出结果 (能输出具体的组合情况则更好) (参考函数原型: `float f(int n)`)。

算法分析: 本题是将 C 语言程序设计和组合数学知识结合的习题, 只需要知道  $C(p,n)$  的计算公式即可设计出程序。

参考程序:

```
#include<stdio.h>

float f(int n)
{
    float s=1;
    int i;
    for(i=2;i<=n;i++)//阶乘
    {
        s*=i;
    }
    return s;
}

main()
{
    float c;
    int p,n;
    printf("请输入 C(p,n)中 p 和 n: ");
    scanf("%d,%d",&p,&n);
    if(p>n)
        printf("请重新输入, 要求 p<=n!\n");
    else
    {
        c=f(n)/f(p)/f(n-p);//求 C 的数学公式用 C 语言描述
        printf("C(p,n)=%.10f\n",c);
    }
}
```

6.15 定义函数判断一个点与坐标原点的距离是否小于 1, 是否在单位圆内。写一个通过蒙特卡罗方法计算圆周率值的程序: 每次计算随机生成两个 0 与 1 之间的实数 (利用标准库随机数生成函数产生这种实数), 看这两个值形成的点是否在单位圆内。生成一系列随机点, 统计单位圆内的点数与总点数, 看它们之比的 4 倍是否趋向  $\pi$  值。生成 100, 200, ..., 1000 个数据点做试验 (参考函数原型: `float pi(int n)`)。

参考程序:

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

```
#include<stdlib.h>
#define N 100
float pi(int n)
{
    float x,y,inno=0;//inno 为单位圆的点数
    long i;

    for(i=1;i<=n;i++)
    {
        x=(float)(rand())/RAND_MAX;//参考程序分析
        y=(float)(rand())/RAND_MAX;
        if(x*x+y*y<1.0)
            inno++;
    }
    return 4*inno/n;//参考程序分析
}
main()
{
    float PI=pi(N);
    printf("圆周率是:%8.6f",PI);
}
```

程序分析：程序中的两行注释的注意点是：（1）rand()函数是返回一个0到RAND\_MAX间的随机整数，具体使用方法参考主教材附录。在本程序中，经过处理，生成的数为0到1间的随机数。在一些编译器中直接可以调用某些函数，生成0到1间的随机数，请读者去学习。（2）为什么要将单位圆的点数与总点数之比乘以4呢，因为x和y都是(0,1)范围，而单位圆的坐标范围为(-1,1)。

**6.16** 请实现一积分函数，使它能求出一个定义好的数学函数在某区间的数值积分值。试采用矩形方法、梯形方法，考察它们在不同分割情况下的表现，以及加细分割对积分值的影响。用不同的数学函数试验程序。如果函数在积分区间出现奇点，程序将出现什么问题？考虑有什么处理办法（参考函数原型：float jf(int n)，n为划分的积分区间个数，用宏定义计算函数值）。

参考程序：

```
#include<stdio.h>
#include<stdlib.h>
#define N 1000
#define FUN x*x+5//宏定义函数
float jf(int n)
{
```

```
float x,f,s=0,i;
for(i=0;i<n;i++)
{
    x=(float)(i)/n;
    f=FUN;
    s+=f/n;//矩形方法
}
return s;
}
main()
{
    float p=jf(N);
    printf("函数积分是: %8.6f",p);
}
```

6.17 定义比较两个身份证（假设身份证均为 18 位，从第 7 到第 14 位代表出生年月日）大小的函数：以出生年月日作为标准；以身份证号作为标准。考虑应该使用结构参数，还是使用结构指针参数（以出生年月日比较两个身份证的参考函数原型：int cmpyyyymmdd(char \*s1,char \*s2)）。

参考程序：

```
#include<stdio.h>
#include<string.h>
int cmpyyyymmdd(char *s1,char *s2)
{
    int flag=1,i;
    char temps1[9],temps2[9];

    for(i=0;i<8;i++)
    {
        temps1[i]=*(s1+6);//取 7 到 14 位的数
        temps2[i]=*(s2+6);
        s1++;
        s2++;
    }
    temps1[9]='\0';
    temps2[9]='\0';
    flag=strcmp(temps1,temps2);//字符串比较
    return flag;
}
```

```
main()
{
    char s1[9],s2[9];
    int flag;

    printf("请输入第一个人的身份证的号码: ");
    gets(s1);
    printf("\n 请输入第二个人的身份证 2 的号码: ");
    gets(s2);
    flag=cmpyyymmdd(s1,s2);
    if(flag)
        printf("第二个人比第一个人先出生! \n");
    else
        printf("第一个人比第二个人先出生! \n");
}
```

6.18 完成正文中基于结构的学生成绩处理程序,提供按照成绩排序输出的功能。其中结构定义如下:

```
struct stu
{
    int num;
    char *name;
    char sex;
    float score;
}
```

参考函数原型: void sort(struct stu \*ps)

算法分析: 此题是综合性较强的一道习题,综合了本章节的很多知识,很具有代表性。

参考程序:

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

```
struct stu
```

```
{
    int num;
    char *name;
    char sex;
    float score;
}
```

```
boy[5]={ {101,"Li ming",'M',45},{102,"Zhang ping",'M',62.5},
```

```
{103,"He fang",'F',92.5},{104,"Cheng ling",'F',87},{105,"Wang ming",'M',58}};
```

```
void sort(struct stu *ps)
{
    int i,j,tempnum;
    float maxscore,minscore,tempsex;
    char tempsex;
    char *tempname;

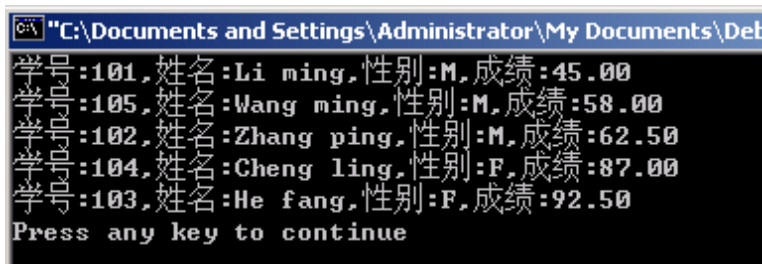
    for(i=0;i<5;i++)
    {
        maxscore=(ps+i) ->score;
        for(j=i+1;j<5;j++)
        {
            minscore=(ps+j) ->score;
            if(maxscore>minscore)
            {
                tempnum=(ps+j) ->num;(ps+j) ->num=(ps+i) ->num;(ps+i) ->num=tempnum;
                tempname=(ps+j) ->name;(ps+j) ->name=(ps+i) ->name;(ps+i) ->name=tempname;
                tempsex=(ps+j) ->sex;(ps+j) ->sex=(ps+i) ->sex;(ps+i) ->sex=tempsex;
                tempsex=(ps+j) ->score;(ps+j) ->score=(ps+i) ->score;(ps+i) ->score=tempsex;
            }
        }
    }
}

main()
{
    struct stu *ps;
    int i;

    ps=boy;
    sort(ps);
    for(i=0;i<5;i++)
    {
        printf("学号: %d, 姓名: %s, 性别: %c, 成绩: %5.2f\n", ps->num, ps->name, ps->sex,
            ps->score);
        ps++;
    }
}
```

```
}
```

运行结果:



```
"C:\Documents and Settings\Administrator\My Documents\Deb
学号:101,姓名:Li ming,性别:M,成绩:45.00
学号:105,姓名:Wang ming,性别:M,成绩:58.00
学号:102,姓名:Zhang ping,性别:M,成绩:62.50
学号:104,姓名:Cheng ling,性别:F,成绩:87.00
学号:103,姓名:He fang,性别:F,成绩:92.50
Press any key to continue
```

## 编程好习惯六：分工明细

一个复杂的程序实现的功能往往很多，如果只用一段程序或者一个 `main()` 函数来实现，难度很大而且容易出错。我们应该对程序功能进行划分，规划得越详细，模块分工越明确，就越容易完整和高效地实现程序所要求的功能。这好比搭积木的游戏，一个再复杂的模型都是由一个个小的积木组成，你也可以把你的积木块组合成各种各样的形状。

通过本章学习后，我们可以将一个程序的功能细化，再将细化后的功能由函数实现，最后在主函数中对各个自定义函数进行调用。这样将大大提高效率，一旦程序出错，只需要在函数中进行错误查找和修改，同时也满足了 C 语言模块化开发的要求。

## 6.7 习题 7

7.1 编写函数，计算 20 个数中的最大值、最小值和平均值，由 `main()` 调用该函数，并输出结果。（要求：使用全局变量 `max` 和 `min` 返回最大值和最小值）

参考程序：

```
#include<stdio.h>
int max=-32768,min=32768;//定义全局变量
void fun(int b[],int n)
{
    int i;
    for(i=0;i<n;i++)
        if(b[i]>max)
            max=b[i];
        else if(b[i]<min)
            min=b[i];
}
main()
```



```
{
    int i,a[20];

    for(i=0;i<20;i++)
    {
        printf("请输入第%d 个数:",i+1);
        scanf("%d",&a[i]);
    }
    fun(a,20);
    printf("最大值:%d\n",max);
    printf("最小值:%d\n",min);
}
```

7.2 编写函数，计算两个整数的最大公约数和最小公倍数。由 main()调用该函数，并输出结果，两个整数由键盘输入。（要求：使用全局变量 gysh 和 gbsh 返回最大公约数和最小公倍数）

参考程序：

```
#include<stdio.h>
int gysh,gbsh;//gysh 为公约数， gbsh 为公倍数
void func(int a,int b)
{
    int c,r;

    c=a*b;
    r=a%b;
    while(r!=0)
    {
        a=b;
        b=r;
        r=a%b;
    }
    gysh=b;
    gbsh=c/b;
}
main()
{
    int m,n;

    printf("请输入两个整数： \n");
```

```
scanf("%d,%d",&m,&n);
func(m,n);
printf("最大公约数:%d\n",gysh);
printf("最小公倍数:%d\n",gbsh);
}
```

### 7.3 编写程序，使用动态内存分配方式，对 10 个数按从小到大排序。

提示：动态内存分配请参考主教材 7.7 节的知识，需要注意的是在调用分配函数后，还需要 free() 释放存储空间。

参考程序：

```
#include<stdio.h>
#include<stdlib.h>
main()
{
    int i,j,k,temp,*p,n=10;
    p=(int*)malloc(n*sizeof(int));

    printf("请输入 10 个数： \n");
    for(i=0;i<n;i++)
        scanf("%d",&p[i]);
    for(i=0;i<n-1;i++)
    {
        k=i;
        for(j=i+1;j<n;j++)
            if(p[k]>p[j])
                k=j;
        temp=p[i];
        p[i]=p[k];
        p[k]=temp;
    }
    printf("排序后为： \n");
    for(i=0;i<n;i++)
        printf("%d",p[i]);
    printf("\n");
    free(p);
}
```

程序分析：本题也可以使用 calloc() 函数实现，方法为：p=(int\*)calloc(n,sizeof(int))。

7.4 分别编写两个源程序文件，在一个文件中输入 10 个整数并输出运算结果，在另一个文件中对这 10 个数排序。（提示：使用 extern 将外部变量的作用域扩展到其他源程序文件）

算法提示：两个源文件也就是将主函数和自定义函数分别在两个文件中实现。详细算法和习题 7.3 相似。

参考程序：

文件一：主函数

```
#include<stdio.h>
int a[10];
void sort();
main()
{
    int i;

    printf("请输入 10 个整数： \n");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    sort();
    printf("排序后为： ");
    for(i=0;i<10;i++)
        printf("%5d",a[i]);
    printf("\n");
}
```

文件二：自定义函数

```
extern a[10];
void sort()
{
    int i,j,k,temp;

    for(i=0;i<9;i++)
    {
        k=i;
        for(j=i+1;j<10;j++)
            if(a[k]>a[j])
                k=j;
        temp=a[i];
        a[i]=a[k];
        a[k]=temp;
    }
}
```

7.5 阅读下列程序，写出运行结果。

```
#include <stdio.h>
int a=10;
void main()
{
    extern b;
    printf("a=%d,b=%d\n",a,b);
    printf("a*b=%d\n",a*b);
}
int b=20;
```

答：结果为： a=10,b=20  
a\*b=200

7.6 阅读下列程序，写出运行结果。

```
#include <stdio.h>
int x=100;
int y=200;
void main()
{
    static int z=300;
    int x=155;
    printf("x=%d\n",x);
    printf("y=%d\n",y);
    printf("z=%d\n",z);
}
```

答：结果为： x=155  
y=200  
z=300

7.7 阅读下列程序，写出运行结果。

```
#include <stdio.h>
void fun(int i,int j)
void main()
{
    void fun(int,int);
    int i,j,k,l,x,y;
    i=2;j=3;k=4;l=5;
    x=6;y=7;
    fun(i,j);
    printf("i=%d,    j=%d\n",i,j);
}
```

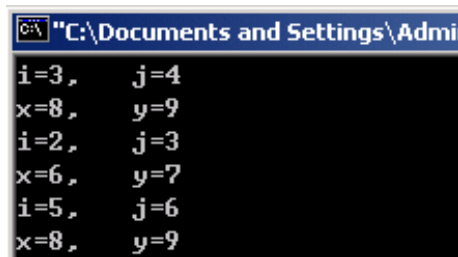
```

    printf("x=%d,    y=%d\n",x,y);
    fun(k,l);
}

void fun(int i,int j)
{
    int x,y;
    x=8;y=9;
    i=i+1;
    j=j+1;
    printf("i=%d,    j=%d\n",i,j);
    printf("x=%d,    y=%d\n",x,y);
}

```

答：结果为：



```

i=3,    j=4
x=8,    y=9
i=2,    j=3
x=6,    y=7
i=5,    j=6
x=8,    y=9

```

结果分析：本题需要注意的是两次调用了函数 fun()，此时将执行操作的 fun()函数中的 x 和 y 的值，在函数调用后，并不改变 i 和 j 的值。

7.8 阅读下列程序，写出运行结果。

```

#include <stdio.h>
void main()
{
    void add();
    float a=1.0,b=2.0,c=3.0;
    printf("a=%f,b=%f,c=%f\n",a,b,c);
    add();
    add();
}

void add()
{
    int a=1;
}

```

```
register int b=1;
static int c=1;
a=a+a;
b=b+b;
c=c+c;
printf("a=%d,b=%d,c=%d\n",a,b,c);
}
```

答：运行结果为：a=1.000000,b=2.000000,c=3.000000

a=2,b=2,c=2

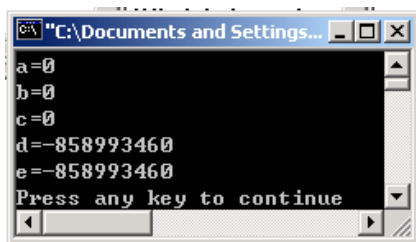
a=2,b=2,c=4

结果分析：本题需要注意的是静态变量的概念和使用方法。

### 7.9 阅读下列程序，写出运行结果。

```
#include <stdio.h>
int a;
static int b;
void main()
{
    static int c;
    register int d;
    int e;
    printf("a=%d\n",a);
    printf("b=%d\n",b);
    printf("c=%d\n",c);
    printf("d=%d\n",d);
    printf("e=%d\n",e);
}
```

答：运行结果为：



结果分析：全局变量和静态变量的初始缺省值为 0，自动变量和寄存器变量初始缺省值不确定。

7.10 下面有两个求一个数的平方值的宏定义，按下面情况调用，求 a 和 b 的值，请问哪一个结果正确？为什么？

```
#define SQUARE1(x) x*x
```

```
#define SQUARE2(x) (x)*(x)
```

```
┆
```

```
a= SQUARE1(5+1);
```

```
b= SQUARE2(5+1);
```

答：a 错误，b 正确。原因是 a 执行的运算是： $5+1*1+5$ 。

7.11 使用带参数宏 MAX(a,b)，编写从 a、b、c 三个数中找出最大者的程序。

参考程序：

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

```
#define MAX(a,b) (a>b)? a:b//二目运算
```

```
main()
```

```
{
```

```
    int a,b,c,max;
```

```
    printf("请输入 3 个整数： ");
```

```
    scanf("%d,%d,%d",&a,&b,&c);
```

```
    max=MAX(a,b);//求 a,b 的最大值
```

```
    max=MAX(max,c);//求 max 和 c 的最大值
```

```
    printf("最大值是:%d\n",max);
```

```
}
```

7.12 定义一个求两个值中小者的宏 MIN，并编写一个测试该宏定义的程序。

参考程序：

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

```
#define MIN(a,b) (a<b)? a:b
```

```
main()
```

```
{
```

```
    float a,b,min;
```

```
    printf("请输入 2 个数： ");
```

```
    scanf("%f,%f",&a,&b);
```

```
    min=MIN(a,b);
```

```
    printf("最小值是:%f\n",min);
```

```
}
```

7.13 使用带参数的宏，编写输入两个整数，求其相除的商和余数的程序。

参考程序：

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

```
#define printf_ms printf("Input two integers:")
```

```
#define input(a,b) scanf("%d,%d",&a,&b)
```

```
#define shang(a,b) ((a)/(b))
```

```
#define yushu(a,b) ((a)%(b))
main()
{
    int x,y;

    printf_1ms;
    input(x,y);
    printf("shang=%d\n",shang(x,y));
    printf("yushu=%d\n",yushu(x,y));
}
```

程序分析：本题将系统函数进行了宏替换操作。

## 编程好习惯七：巧找替身

相信大家都观看过精彩的动作类电影，电影里一个个特技镜头让人着迷。但这些特级镜头都是电影明星亲自出演的吗，大家肯定会回答“NO，是替身演员”。对，正是这些替身完成了明星演员不能完成的工作，同时也让影片更加精彩。在 C 语言当中，宏就扮演了一个“替身”的角色。

合理地使用宏，能提高程序运行的效率。一旦“演员”（主程序）完成起来低效或者困难，就找“替身”（宏）来实现；一旦出错，也只需要“替身”再修改一次，不需要所有“演员”再出马，从而达到事半功倍的效果。

## 6.8 习题 8

8.1 重写例 8-4 程序，使用函数 swap()将输入数据的高位字节和低位字节交换后返回。main()函数调用这个函数，实现程序的功能。

参考程序：

```
#include<stdio.h>
union body
{
    struct BYTE
    {
        unsigned char l,h;
    }byte;
    unsigned int word;
};
union body swap(union body x)
```



```
{
    union body y;

    y.byte.h=x.byte.l;
    y.byte.l=x.byte.h;
    return y;
}
main()
{
    union body a,b;

    printf("请输入一个数: ");
    scanf("%x",&a.word);
    b=swap(a);
    printf("%x->%x\n",a.word,b.word);
}
```

8.2 改写例 8-15 程序, 将设置显示器工作模式改为读取显示器当前工作模式。BIOS 显示器服务程序中, 第 15 号功能为读当前的显示器工作模式。中断服务后, AL 中的返回值是当前显示模式。

参考程序:

```
#include<stdio.h>
#include<dos.h>
#include<process.h>
#include<stdlib.h>

int readmode()
{
    union REGS inregs,outregs;
    int x;

    inregs.h.ah=15;
    int86(0x10,&inregs,&outregs);
    x=outregs.h.al;
}
main()
{
    int mode=0;
```

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

8.3 阅读以下程序，给出正确的运行结果。

```
#include <stdio.h>
main()
{
    union EXAMPLE
    {
        struct
        {
            int x;
            int y;
        } in;
        int a;
        int b;
    } e;
    e.a=1;
    e.b=2;
    e.in.x=e.a*e.b;
    e.in.y=e.a+e.b;
    printf("\n%d,%d",e.in.x,e.in.y);
    printf("\n%d,%d",e.a,e.b);
}
```

程序的运行结果是： 2,3  
                          1,2

8.4 写出下列枚举常量的值。

```
enum coin
{
    penny,
    nickel,
    dime,
    quarter,
    half_dollar=50,
    dollar
};
```

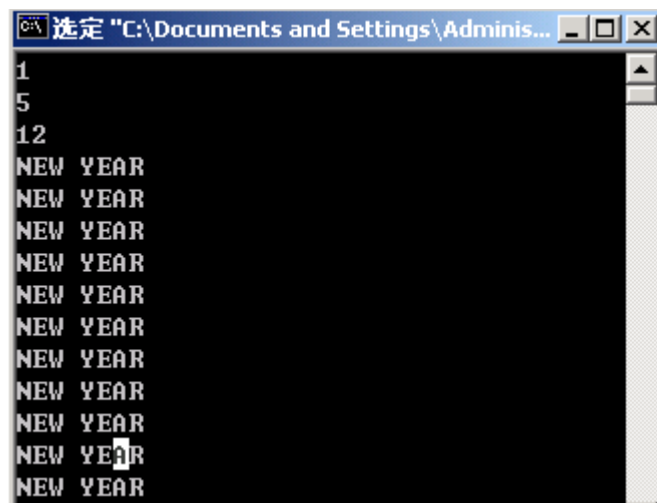
结果为:

```
penny:    0
nickel:   1
dime:     2
quarter:  3
half_dollar: 50
dollar:   51
```

8.5 读程序, 写出下面程序运行结果。

```
main()
{
    int i;
    enum
    onth {JAN=1,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC} month;
    month=JAN;
    printf("%d\n",month);
    month=MAY;
    printf("%d\n",month);
    month=DEC;
    printf("%d\n",month);
    for(i=JAN;i<DEC;i++)
        printf("NEW YEAR\n");
}
```

运行结果为:



```
1
5
12
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
NEW YEAR
```

以上两题请参考主教材关于枚举相关章节的知识。

8.6 编写一个程序，定义 modem 变量为位域结构 struct bits 类型，并初始化 b0 为 0，b1 为 1，b2 为 1，b3 为 0，b4 为 1。要求程序输出 modem 变量中的每个位域成员的值。

算法分析：参考主教材 8.1 节的相关知识可以很快解答这道题。

参考程序：

```
#include<stdio.h>
main()
{
    struct bits
    {
        unsigned b0:1;
        unsigned b1:1;
        unsigned b2:1;
        unsigned b3:1;
        unsigned b4:4;
    }modern={0,1,1,0,1};

    printf("b0:%d,b1:%d,b2:%d,b3:%d,b4:%d\n",modern.b0,modern.b1,modern.b2,modern.b3,modern.b4);
}
```

## 编程好习惯八：不骄不躁

我们在前面所介绍的好习惯主要针对程序本身，而这点“不骄不躁”主要针对程序员心理。程序设计是一个复杂、漫长和充满挑战性的工作，很难一蹴而就。一般情况下，一个程序很难一次就编写得很完美，但也不会永远都无法实现。所以这就需要程序员保持良好的心态，在程序运行通过后仔细找找不足或者是漏洞，使程序更加完美，这叫“不骄”；一旦编程出现问题和困难，也仔细找错误，重新设计，直到调试成功，这叫“不躁”。良好的心态是程序设计的关键因素之一。

## 6.9 习题 9

9.1 编写一个程序，把从键盘输入的信息存入指定的文件中，要求文件名用命令行参数指定。

参考程序：

```
#include<stdio.h>
main(int argc,char *argv[])
{
```

```

FILE *fp;
char ch;

if(argc==1)
    printf("usage:no linenum filename\n");
else if(argc>2)
    printf("Too many arguments to linenum\n");
else if((fp=fopen(argv[1],"w"))==NULL)//打开文件，请注意打开的方式
{
    printf("can't open %s\n",argv[1]);
    return;
}
else
{
    printf("Input information through keyboard.\n");
    while(ch!='\n')
    {
        ch=getchar();//从键盘得到信息
        puts(ch,fp);//存入指定的文件中
    }
}
fclose(fp);//关闭文件
}

```

程序分析：首先对命令行参数的合法性进行判断，然后打开文件，完成指定操作。最后需要关闭文件。

**9.2 编写一个将字符串“Data Structure”，“Operating System”，“Computer Graphics”，“Software Engineering”写入文件中去的程序。**

本题和上题类似，参考程序：

```

#include<stdio.h>
void main()
{
    int i;
    FILE *fp;
    char *buf[4]={"Data Structure","Operation System","Computer Graphics","Software Engineering"};

    if((fp=fopen("file 9-2","w"))==NULL)
    {

```

```
        printf("can't open %s\n", "myfile");
        return;
    }
    else
        for(i=0; i<4; i++)
        {
            fputc(buf[i], fp);
            putc('\n', fp);
        }
    fclose(fp);
}
```

9.3 重写主教材 9-6.cpp 程序，将在显示屏上输出文件内容改为计算文件中的字符个数。

参考程序：

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

main(int argc, char *argv[])
{
    FILE *fp;
    char ch;
    int count=0; //统计字符个数

    if (argc==1)
        printf("usage: linenum filename\n");
    else if (argc>2)
        printf("Too many arguments to linenum\n");
    else if ((fp=fopen(argv[1], "r"))==NULL)
    {
        printf("Can't open %s\n", argv[1]);
        return;
    }
    else
    {
        while((ch=getc(fp))!=EOF)
            count++; //判断后进行统计
    }
    printf("File--%s has %d chars.\n", argv[1], count);
}
```

```
    fclose(fp);  
}
```

9.4 重写主教材 9-6.cpp 程序，把要处理的行文文件的内容全部改为大写后，写入一个新文件中。

参考程序：

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

```
#include<process.h>
```

```
main(int argc,char *argv[])
```

```
{
```

```
    FILE *oldf,*newf;
```

```
    char ch;
```

```
    if (argc!=3)
```

```
        printf("usage:linenum filename\n");
```

```
    else if ((oldf=fopen (argv[1],"r") )==NULL)
```

```
    {
```

```
        printf("Can't open %s\n",argv[1]);
```

```
        return;
```

```
    }
```

```
    else if ((newf=fopen(argv[2],"w"))==NULL)
```

```
    {
```

```
        printf("Can't open %s\n",argv[2]);
```

```
        return;
```

```
    }
```

```
    else
```

```
    {
```

```
        while((ch=getc(oldf))!=EOF)
```

```
        {
```

```
            if (ch>='a'&&ch<='z') ch-=32;//转换为大写
```

```
            putc(ch,newf); //写入新文件中
```

```
        }
```

```
    }
```

```
    fclose(oldf); //关闭旧文件
```

```
    fclose(newf); //关闭新文件
```

```
}
```

9.5 重写主教材 9-6.cpp 程序, 将由命令行参数指定的文件在显示屏上输出, 计算并输出文件包含的行数和字符个数。

参考程序

```
#include<stdio.h>
#include<process.h>
main(int argc,char *argv[])
{
    FILE *fp;
    char ch;
    int count_char=0,count_row=0;

    if (argc==1)
        printf("usage:linenum filename\n");
    else if (argc>2)
        printf("Too many arguments to linenum\n");

    else if ((fp=fopen(argv[1],"r"))==NULL)
    {
        printf("Can't open %s\n",argv[1]);
        return;
    }
    else
    {
        while((ch=getc(fp))!=EOF)
        {
            putc(ch,stdout);
            count_char++;//统计字数
            if (ch=='\n')
                count_row++;//行数
        }
    }
    printf("\nFile--%s: has %d lines,%d chars.\n",
        argv[1],count_row,count_char);
    fclose(fp);
}
```



### 9.6 使用 I / O 重定向，把主教材 9-6.cpp 程序改写成一个拷贝文件的命令。

参考程序：

```
#include<stdio.h>
main()
{
    int c;
    while ((c=getchar())!=EOF)
        putchar(c);
}
```

### 9.7 编写一个统计由命令行参数指定的文件中最长行所具有的字符个数的程序。

参考程序：

```
#include<stdio.h>
#include<process.h>
main(int argc,char *argv[])
{
    FILE *fp;
    char ch;
    int count_char=0,count_row=0,max=0;

    if (argc==1)
        printf("usage:linenum filename\n");
    else if (argc>2)
        printf("Too many arguments to linenum\n");
    else if ((fp=fopen(argv[1], "r"))==NULL)
    {
        printf("Can't open %s\n",argv[1]);
        exit(0);
    }
    else
    {
        while((ch=getc(fp))!=EOF)
        {
            count_char++; //统计每行的长度
            if (ch=='\n')
            {
                if (count_char>max)

```

```
        max=count_char;//找到最长的一行
        count_char=0;
        count_row++;
    }
}
}
printf("File--%s:has %d lines.\n",argv[1],count_row);
printf("The longest line has %d chars.\n",max-1);
fclose(fp);
}
```

9.8 编写一个比较两个文件的程序，要求显示两个文件中不相同的行的行号以及该行中不相同的字符的开始位置。

参考程序：

```
#include<stdio.h>
#include<process.h>
# define MAX 5
main(int argc,char *argv[])
{
    FILE *fp1,*fp2;
    char c1,c2;
    int crow=0,cchar=0;
    static int row[MAX][2];
    int i=0;
    int j,sign;

    if (argc!=3)
        printf("usage:linenum filename\n");
    else if ((fp1=fopen(argv[1],"r"))==NULL)
    {
        printf("Can't open %s\n",argv[1]);
        exit(0);
    }
    else if ((fp2=fopen(argv[2],"r"))==NULL)
    {
        printf("Can't open %s\n",argv[2]);
        exit(0);
    }
```

```
    }
else
    {
        crow++;
        c1=getc(fp1);
        c2=getc(fp2);
        while(!feof(fp1)&&!feof(fp2))
        {
            sign=0;
            cchar++;
            while (c1!='\n'&&c2!='\n')
            {
                if (c1!=c2&&sign==0)
                {
                    i++;
                    row[i][0]=crow;
                    row[i][1]=cchar;
                    sign=1;
                }
                c1=getc(fp1);
                c2=getc(fp2);
                cchar++;
            }
            while (c2=='\n'&&c1!='\n')
                c1=getc(fp1);
            while(c1=='\n'&&c2!='\n')
                c2=getc(fp2);
            crow++;
            cchar=0;
            c1=getc(fp1);
            c2=getc(fp2);
        }
    }
for(j=1;j<=i;j++)
    printf("line %d: start position is %d\n",row[j][0],row[j][1]);
fclose(fp1);
```

```
    fclose(fp2);  
}
```

9.9 编写一个程序将命令行指定的一个文件的内容追加到另一个文件末尾。

参考程序:

```
#include<stdio.h>  
#include<process.h>  
  
main(int argc,char *argv[])  
{  
    FILE *fp1,*fp2;  
    char ch;  
  
    if (argc!=3)  
        printf("usage:linenum filename\n");  
    else if ((fp1=fopen(argv[1],"a"))==NULL)  
    {  
        printf("Can't open %s\n",argv[1]);  
        return;  
    }  
    else if ((fp2=fopen(argv[2],"r"))==NULL)  
    {  
        printf("Can't open %s\n",argv[2]);  
        return;  
    }  
    else  
        while((ch=getc(fp2))!=EOF)  
            putc(ch,fp1);//追加到另一个文件的末尾  
    fclose(fp1);  
    fclose(fp2);  
}
```

9.10 编写一个程序将指定文件的 m 行到 n 行的每一行写到显示屏上, m 和 n 的值从键盘输入。

参考程序:

```
#include<stdio.h>  
#include<process.h>  
main(int argc,char *argv[])
```

```
{
    FILE *fp;
    char ch;
    int count_row=1;
    int m,n;

    if (argc==1)
        printf("usage:linenum filename\n");
    else if (argc>2)
        printf("Too many arguments to linenum\n");
    else if ((fp=fopen(argv[1],"r"))==NULL)
    {
        printf("Can't open %s\n",argv[1]);
        exit(0);
    }
    else
    {
        printf("Please input line:m,n:");
        scanf("%d,%d",&m,&n);
        while((ch=getc(fp))!=EOF)
        {
            if (ch=='\n')
                count_row++;
            if (count_row>=m&&count_row<=n)
                putc(ch,stdout);
        }
    }
    fclose(fp);
}
```

9.11 编写一个能在终端显示一个文件内容的程序，要求一次显示 20 行，在每 20 行的结尾，程序等待从键盘键入一个字符。如果该字符为 q，则程序将停止显示文件内容；如果是其他字符，则显示该文件的下 20 行内容。

参考程序：

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

```
main(int argc,char *argv[])
{
    FILE *fp;
    char c1,c2='c';
    int count_row=0;

    if (argc==1)
        printf("usage:linenum filename\n");
    else if (argc>2)
        printf("Too many arguments to linenum\n");
    else if ((fp=fopen(argv[1],"r"))==NULL)
    {
        printf("Can't open %s\n",argv[1]);
        return;
    }
    else
    {
        while(((c1=getc(fp))!=EOF)&&(c2!='q'&&c2!='Q'))
        {
            putc(c1,stdout);
            if (c1=='\n')
                count_row++;
            if (count_row==20)
            {
                c2=getch();
                count_row=0;
            }
        }
    }
    fclose (fp) ;
}
```

## 编程好习惯九：交流互助

在经过所有的习题训练后，大家应该发现，大型的程序设计是有很大的难度的，一个人很难将其实现，更何况完整了，所以需要程序员多多进行交流讨论，在互联网上查找相关的

知识和问题的解决方案,拓展自己的思路,这样才能很好地完成一段较为复杂的程序。

## 6.10 课程设计解答

### 6.10.1 设计一

算法分析:

可以用一个 $M \times N$ 像素的图像来代表 $M \times N$ 个细胞,其中每一个像素,代表一个细胞,像素为黑色表示细胞为生,像素为白色代表细胞为死。

设定图像中每个像素的初始状态后依据上述的游戏规则演绎生命的变化,由于初始状态和迭代次数不同,将会得到令人叹服的优美图案(具体图型函数请参考相关资料)。

参考程序:

```
#include <graphics.h>
main()
{
    int orgData[100][100],resData[100][100];
        /*分别记录每次迭代的初始和结果状态*/
    int nCount,nRows,nCols,i,j,times;          /*times记录迭代次数*/
    int GraphDriver=DETECT,GraphMode;
    for (i=0;i<100;i++)                        /*初始化数据,令每一个细胞为生*/
        for (j=0;j<100;j++)
            orgData[i][j]=1;
    initgraph(&GraphDriver,&GraphMode,""); /*初始化屏幕显示*/
    setcolor(WHITE);
    rectangle(270,190,370,290);                /*作显示边框*/
    for (times=1;times<200;times++)
    {
        for (nRows=1;nRows<99;nRows++)
        {
            for (nCols=1;nCols<99;nCols++)
            {
                /*计算每一个细胞周围的活的细胞数*/
                nCount=orgData[nRows-1][nCols-1]+orgData[nRows-1][nCols]
                    +orgData[nRows-1][nCols+1]+orgData[nRows][nCols-1]
                    +orgData[nRows][nCols+1]+orgData[nRows+1][nCols-1]
                    +orgData[nRows+1][nCols]+orgData[nRows+1][nCols+1];
                switch(nCount)
```

```
{
    /*周围有3个活细胞，该细胞为生，在屏幕上用黑色像素表示*/
    case 3: putpixel(nCols+210,120+nRows,BLACK);
            resData[nRows][nCols]=1;
            break;
    /*周围有2个活细胞，该细胞不变，在屏幕显示也不变*/
    case 2: resData[nRows][nCols]=orgData[nRows][nCols];
            break;
    /*其他情况下，细胞为死，在屏幕上用白色像素表示*/
    default:resData[nRows][nCols]=0;
            putpixel(nCols+210,120+nRows,WHITE);
}
}
```

```
for (i=1;i<99;i++)
for (j=1;j<99;j++)
orgData[i][j]=resData[i][j];
getch();
}
```

### 6.10.2 设计二

算法分析：运动员得分数组

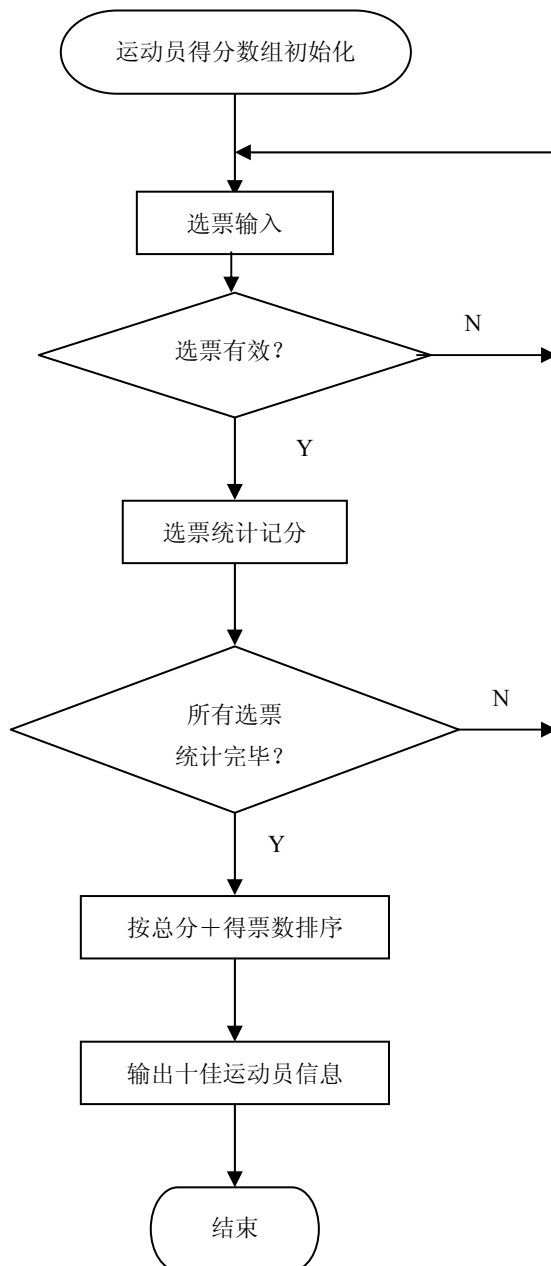
```
struct sporter
{
    int score;
    int piao_num;
}spot[100];
```

选票结构

```
struct piao
{
    int first_num;
    int second_num;
    int third_num;
    int forth_num;
    int fifth_num;
```



```
int sixth_num;  
int seventh_num;  
int eighth_num;  
int ninth_num;  
int tenth_num;  
};  
算法流程:
```



### 6.10.3 设计三

本题的解答方法比较多，在此给出一些比较通用的思想。

(1) 使用结构定义出通讯录的一条记录，包括姓名、学号和手机号等元素的类型和大小。

(2) 定义主菜单进行功能的选择，如 add 添加、del 删除和 save 保存等。

(3) 使用函数实现各个功能。

以下给出部分函数的实现算法，请读者参考：

struct address()//定义结构，表示学生信息

```
{
    char name[30];
    char number[20];
    char major[30];
    char phone[20];
    char home[30];
}
```

void enter()//输入一个学生的通讯信息

```
{
    struct address *info;
    for(;;)
    {
        info=(struct address *)malloc(sizeof(list_enter));
        if(!info)
        {
            printf("\n out of memory");
            return;
        }
        inputs("enter name:",info->name,30);//调用 inputs 函数
        if(!info->name[0])
            break;
        inputs("enter number",info->number,20);
        inputs("enter major",info->major,30);
        inputs("enter phone",info->phone,20);
        inputs("enter home",info->home,30);
    }
}
```

inputs(char \*prompt,char \*s,int count)//定义输入函数

```
{
    char p[255];
    do
    {
        printf(prompt);
        gets(p);
        if(strlen(p)>count)
            printf("\ntoo long");
    }while(strlen(p)>count);
    strcpy(s,p);
}
```

void search()//查找函数，使用姓名查找

```
{
    char name[30];
    struct address *info,*find();
    printf("enter name to find:");
    gets(name);
    if(!(info=find(name)))
        printf("not found");
    else
        display(info);//调用 display 函数显示
}
```

void save()//保存记录

```
{
    struct address *info;

    FILE *fp;
    if((fp=fopen("mlist","wb"))==NULL)
    {
        printf("can not open file");
        exit(1);
    }
    printf("\nsaving file\n");
    info=start;
    while(info)
        fwrite(info,sizeof(struct address),1,fp);
}
```

```

    fclose(fp);
}

```

#### 6.10.4 设计四

算法设计思想:

任何一种棋类游戏的关键是对当前棋局是否有正确的评分标准, 评分越准确则电脑的 AI (人工智能) 越高, 五子棋游戏也是如此。在打分之前, 我们先扫描整个棋盘, 把每个空位从八个方向上的棋型填入数组 `gStyle(2, 15, 15, 8, 2)`, 其中第一个下标为 1 时表示黑棋, 为 2 时表示白棋, 第二和第三个下标表示(x,y), 第四个下标表示 8 个方向, 最后一个下标为 1 时表示棋子数, 为 2 时表示空格数, 如:

`gStyle(1,2,2,1,1)=3` 表示与坐标(2,2)在第 1 个方向上相邻的黑棋棋子数为 3;

`gStyle(1,2,2,1,2)=4` 表示与坐标(2,2)在第 1 个方向上的最近的空格数为 4。

在定义方向时, 也应该注意一定的技巧, 表示两个相反的方向的数应该差 4, 在程序中可以这样定义:

```

Const DIR_UP = 1
Const DIR_UPRIGHT = 2
Const DIR_RIGHT = 3
Const DIR_RIGHTDOWN = 4
Const DIR_DOWN = 5
Const DIR_DOWNLEFT = 6
Const DIR_LEFT = 7
Const DIR_LEFTUP = 8

```

如下图所示:

```

-----
-----
---00---
-OX*XX---
-----
-----

```

图中的\*点坐标为(4,4) (打\*的位置是空位), 则:

`gStyle(2,4,4,1,1)=1` 在(4,4)点相邻的上方白棋数为 1;

`gStyle(2,4,4,1,2)=2` 在(4,4)点的上方距上方白棋最近的空格数为 2;

`gStyle(1,4,4,3,1)=2` 在(4,4)点相邻的右方黑棋数为 2;

`gStyle(1,4,4,3,2)=1` 在(4,4)点的右方距右方黑棋最近的空格数为 3。

一旦把所有空点的棋型值填完, 我们很容易地得出黑棋水平方向上点(4,4)的价值, 由一个冲 1 (把有界的棋称为冲) 和活 2 (两边无界的棋称为活) 组成的。而白棋在垂直方向上点(4,4)的价值是一个活 1, 而在“/”方向也是活 1, 所以, 只要把该点的对于黑棋和白棋的

价值算出来,然后就取棋盘上各个空点的这两个值的和的最大一点作为下棋的点。对各种棋型应该取什么值我们可以先进行如下假设:

$F_n$  表示先手  $n$  个棋子的活棋型,如:  $F_4$  表示先手活四;

$F_n'$  表示先手  $n$  个棋子的冲棋型,如:  $F_4'$  表示先手冲四;

$L_n$  表示后手  $n$  个棋子的活棋型,如:  $L_3$  表示后手活三;

$L_n'$  表示后手  $n$  个棋子的冲棋型,如:  $L_3'$  表示后手冲三。

根据在一行中的棋型分析,得到如下关系:

$$L_1' \leq F_1' < L_2' \leq F_2' \leq L_1 < F_1 < L_2 < F_2 < L_3' \leq F_3' < L_4' < F_4 = F_4$$

这个关系包含了进攻和防守的关系(当然,可以自己定义这些关系)。对这些关系再进一步细化,如在一个可下棋的点,其四个方向上都有活三,也比不上一个冲四,所以我们可以又得到  $4 * F_3 < L_4'$  这一关系,同样,我们还可以得到其他的关系,如:  $4 * F_2 < L_3$ 、 $4 * L_3 < F_3 \dots$ , 这些关系由于每个人的定法可能不一样,计算机的 AI 也就不一样。最后我们把分值最小的  $L_1'$  值定为 1,则我们就得到了下面各种棋型的分值,由 C 语言表示为:

```
F[2][5]={0,2,5,50,16000},{0,10,30,750,16000};
```

```
L[2][5]={0,1,5,50,3750},{0,10,30,150,4000};
```

$F$  数组表示先手,第一个下标为 0 时表示冲型,第二个下标表示棋子数,则  $F_2'$  对应  $F[0][2]$ ;  $L$  数组表示后手,第一个下标为 0 时表示冲型,第二个下标表示棋子数,则  $L_2$  对应  $F[1][2]$ 。棋型的分值关系确定好了以后,把每一个可下点的四个方向的棋型值相加(包括先手和后手的分值),最后选择一个最大值,并把这一点作为计算机要下的点。

算法存在的问题:

1. 最大值也许不止一个点,但在程序中可只选择第一个最大点,当然你可以用几个随机数来决定。
2. 选择哪一个最大值点,也可以对这些最大值点再作进一步的分析。
3. 在这个算法中只考虑了周围有棋子的点,而其他点没有考虑。
4. 可以再更进一步,用这个算法来预测以后的几步棋,再选择预测值最好的一步,这样电脑的 AI 就更高了。
5. 这个算法没有考虑黑棋的禁手(双三、双四和多于五子的连棋)。

# 附录

## 附录一 C 语言常见错误总结

### 一、编程的常见错误

- (1) C 语言对大小写编译有区别，书写标识符时，忽略了大小写字母。
- (2) 使用了未定义的变量。C 语言中对变量应该先定义再使用，所有常量和变量必须加以说明。
- (3) 变量赋给了不匹配的数据类型，如将浮点数 3.5 赋给了整形变量。
- (4) 忽略了变量的类型，进行了不合法的运算，比如使用了 `float` 类型变量进行求余运算。
- (5) 将字符常量与字符串常量混淆。字符常量是由一对单引号括起来的单个字符，字符串常量是一对双引号括起来的字符序列。
- (6) 变量定义时与保留字重复，将出现错误。
- (7) 忽略了“=”与“==”的区别。“=”是赋值运算符，“==”是关系运算符。
- (8) 不正确使用分号和引号，或者遗漏了分号和引号。
- (9) “{”和“}”、“(”和“)”的不匹配，推荐读者在使用时将两个同时输入，再退格进行程序输入，将很大程度上避免类似错误的发生。
- (10) `scanf` 输入时，输入了错误类型的数据项。应该提示用户输入正确的类型。
- (11) `scanf` 输入变量时忘记加地址运算符“&”。这个错误 VC 编译器不能检查，但在运行时会出现内存错误，所以请格外注意。
- (12) 输入数据的方式与要求不符。如要求 `scanf("%d, %d",&a,&b)`，输入 3（空格）4，则会出现错误，应该输入 3，4。
- (13) 输入输出的数据类型与所用格式说明符不一致。
- (14) 输入数据时不能规定精度，如 `scanf("%7.2f",&a)`，则会出错。
- (15) 不能区分自加和自减操作，以及前缀和后缀操作。
- (16) 条件语句中的 `if` 和 `else` 语句不匹配。
- (17) `switch` 语句中漏写 `break` 语句。
- (18) 忽视了 `while` 和 `do-while` 语句在细节上的区别。详细内容请参见主教材。
- (19) C 语言不允许对数组的大小进行动态定义，比如 “`int n;scanf("%d",&n);int a[n];`” 将是错误的。
- (20) 在定义数组时，将定义的“元素个数”误认为是可使用的最大下标值。实际应该是 `n-1`，数组下标由 0 开始。

- (21) 用 `scanf` 函数输入字符数组名, 不必要再加地址符 `&`。
- (22) 同时定义了形参和函数中的局部变量。形参应该在函数体前定义, 而局部变量在函数体内定义。
- (23) 函数参数的求值采取的是自右向左的顺序求参数值。
- (24) 函数的实参和形参类型不一致。
- (25) 形参和实参关系错误。把一个变量传递给函数时, 就构造出一个变量的副本。切记, 为保持函数之间各个变量的独立性, 函数是不接收实际变量的 (即值传送), 但数组和指针是个例外, 当函数接受数组时, 实际上是接受数组的首地址 (即地址传送), 形参的改变返回给实参。
- (26) `float` 类型数据进行关系运算中的等于运算时, 结果不正确, 改为使用 `double` 型即可。
- (27) 不同类型指针的混用, 造成错误。
- (28) 使用指针前一定要初始化, 指向一个确实分配了的空间, 比如数组。
- (29) 错误使用指针。应该注意指针什么时候用 `"*"`, 什么时候使用 `"&"`。请参考主教材指针相关章节。
- (30) 混淆了结构变量和结构类型的区别, 对一个结构类型赋值。
- (31) 遗漏或者没有正确引用头文件。应该使用 `#include` 语句进行引用。
- (32) 系统函数或者关键字写错。
- (33) 使用文件时忘记打开文件、打开方式不正确以及打开后忘记关闭文件。
- (34) 被调用的函数在主函数之后定义。应该将被调用的函数在主函数之前定义, 或者在主函数前增加一个对被调用函数的说明。
- (35) 在执行算术运算时需要注意:
  - ① 根据语法规则书写双精度数字。尽管 C 语言会自动把整型转换成双精度型, 但书写双精度型是个好习惯, 让 C 语言做强行转换效率不高, 有可能导致转换错误。
  - ② 用 0 除是一个灾难性错误, 它会导致程序失败。
  - ③ 确保所有的双精度数 (包括那些程序输入用的双精度数) 是在实数范围之内。
  - ④ 各种类型的数都应该保证在其类型范围内使用, 不能越界。
- (36) 数组使用时需要注意:
  - ① 引用数组元素应该使用方括号而不应该用圆括号。
  - ② 对二维或者多维数组定义和引用方法不对。
  - ③ 误以为数组名代表数组中全部元素。

## 二、连接时的常见错误

- (1) 多个文件连接时, 应该生成或者指定项目文件 (`.PRJ` 文件), 否则编译器无法查找错误。
- (2) 子函数在说明和定义时类型不一致。
- (3) 程序调用的子函数没有定义。

### 三、运行时的常见错误

(1) 警告错误太多。虽然警告信息不影响程序的运行，但在某些编译器中，当警告错误数目大于某一规定值（如缺省为 100）时，应该将编译器中 `options/compiler/errors` 中有关警告错误检查开关设为 `off`。在实际编写程序时，应该尽量修改程序，去除警告，因为警告有时也会导致执行结果与预期结果不同。

(2) 在 DOS 环境下运行程序（如文件操作），反斜杠 “\” 表示一个 DOS 的路径，但 C 语言表示一个换码字符（例如 `\n` 换行，`\t` 制表符），因此一定要在表示路径时应用两个反斜杠。

(3) 在循环语句中，循环变量没有进行修改，或者没有设定循环结束值，都可能引起死循环。

(4) 用动态内存分配函数 `malloc()` 和 `calloc()` 分配的内存区使用完后，应该使用 `free()` 函数释放。如果没有释放，在函数的前几次调用中都会正常，而后面调用有可能会发生死机现象，不能返回操作系统。这是因为没有内存空间可以分配了。这种问题在配置较低的计算机上更容易出现。

(5) 使用了动态分配内存不成功的指针，造成系统破坏。

## 附录二 C 语言编译常见错误信息及处理方法

C 语言常见的编译器的诊断错误分为三种，即致命错误、一般错误和警告。在程序编译的过程中都会找到这些错误，并在结果信息栏输出提示用户修改。

致命错误：一般是内部的编译错误。发生此类错误时，编译会立刻停止。处理办法是找到错误并修改后重新启动编译程序。

一般错误：主要原因是语法错误，磁盘、内存存取错误或命令行错误等。编译程序将根据事先设定的出错个数来决定是否停止编译。编译程序在每个阶段（预处理、语法分析、优化和代码生成）尽可能多地发现源程序中的错误。这类错误一般在经过修改后，重新编译即可解决。

警告：被怀疑的情况，而这些情况本身又有可能合理地成为源程序的一部分。一般不影响程序的运行。过多的警告在某些情况下也会造成编译失败，具体参见附录一的内容。

C 语言编译常见错误信息及处理方法对照表

#### 1. 致命错误信息表及处理方法

出错信息提示	错误原因	处理方法
Bad call of in-line function	内部函数非法调用	在使用一个宏定义的内部函数时，没能正确调用。一个内部函数以两个下划线开始和结束
Irreducible expression tree	不可约表达式树	这种错误指的是文件行中的表达式太复杂，使得代码生成程序无法为它生成代码。这种表达式必须避免使用
Register allocation failure	存储器分配失败	这种错误指的是文件行中的表达式太复杂，代码生成程序无法为它生成代码。此时应简化这种繁杂的表达式或干脆避免使用它



## 2. 一般错误信息表及处理方法

出错信息提示	错误原因	处理方法
#operator not followed by macro argument name	#运算符后没跟宏变元名	“#”号后必须跟一个宏变元名
‘xxxxxx’ not an argument	xxxxxx 不是函数参数	在源程序中将该标识符定义为一个函数参数,但此标识符没有在函数中出现
Ambiguous symbol ‘xxxxxx’	二义性符号 xxxxxx	两个或多个结构的某一域名相同,但具有的偏移、类型不同。在变量或表达式中引用该域而未带结构名时,会产生二义性,此时需修改某个域名或在引用时加上结构名
Argument # missing name	参数#名丢失	参数名已脱离用于定义函数的函数原型。如果函数以原型定义,该函数必须包含所有的参数名
Argument list syntax error	参数表出现语法错误	函数调用的参数间必须以逗号隔开,并以一右括号结束
Array bounds missing	数组的界限符“]”丢失	在数组右边加上“]”
Array size too large	数组元素太大	定义的数组太大,超过了可用内存空间。减少数组元素个数
Assembler statement too long	汇编语句太长	内部汇编语句最长不能超过 480 字节
Bad configuration file	配置文件不正确	配置文件命令选择项必须以短横线开始
Bad file name format in include directive	在 include 语句中,缺少引号(或尖括号)或无文件名	包含文件名必须用引号("filename.h")或尖括号(< filename.h >)括起来
Bad ifdef directive syntax	ifdef 指令语法错误	修改条件编译控制行中的标识符
Bad ifndef directive syntax	ifndef 指令语法错误	修改条件编译控制行中的标识符
Bad undef directive syntax	undef 指令语法错误	修改条件编译控制行中的标识符
Bad file size syntax	位字段长语法错误	一个位字段长必须是 1—16 位的常量表达式
Both return and return of a value used in function	函数作了 return 的表达式	修改 return 语句
Call of non-functin	调用未定义函数	添加函数声明或者修改表达式
Cannot modify a const object	不能修改一个常量对象	常量的操作请参见主教材
Case outside of switch	无 switch, 或者 Case 出现在 switch 外	检查 switch 语句和括号匹配关系
Case statement missing: in function	case 常量表达式后无冒号或者缺少 case 语句	添加冒号或者检查冒号前的符号
Character constant too long	字符常量过大	修改字符常量的赋值, 字符常量只能是一个或两个字符长

(续表)

出错信息提示	错误原因	处理方法
Code has no effect in function	多方面原因, 诸如: 变量无类型/存储类型说明; 指针相加造成; 数组或结构未给尺寸; 使用了非 C 语言	
Compound statement missing } in function	复合语句缺少花括号}	检查花括号的匹配关系
Conflicting type modifiers	类型修饰符冲突	对同一指针, 只能指定一种变址修饰符; 而对于同一函数, 也只能给出一种语言修饰符
Constant expression required	需要常量表达式	修改常量表达式的定义
Could not find file 'xxxxxx.xxx'	找不到 xxxxxx.xx 文件	编译程序找不到命令行上给出的文件, 应该修改文件名
Declaration missing ';' in function	缺少分号	检查程序语句
Declaration needs type or storage class	说明必须给出类型或存储类	修改变量说明, 指出变量类型或者存储类
Declaration syntax error	函数说明出现语法错误	主要由以下的语法错误引起: ①数组缺少左方括号; ②字符型变量赋了两个以上的字符; ③struct/union 说明后少分号; ④函数调用时参数不匹配; ⑤用关键字 typedef 做变量使用; ⑥变量无类型/存储类别说明
Default outside of switch	Default 语句在 switch 语句外出现	检查括号的匹配
Define directive needs an identifier	Define 指令必须有一个标识符	#define 后面的第一个非空格符必须是一个标识符
Division by zero	除数为零	修改表达式的除数
Do statement must have while	do 语句中必须有 While 关键字	添加 while 语句
Do while statement missing; (\)	Do while 语句中缺少分号或者反斜杠	添加分号或者斜杠
Duplicate Case	Case 的常量表达式定义重复	修改 Case 的常量表达式
Enum syntax error	enum 语法错误	修改 enum 说明的标识符表的格式
Enumeration constant syntax error	枚举常量语法错误	枚举的使用方法请参见主教材, 赋给 enum 类型变量的表达式不为常量
Error Directive : xxxx	错误指令: xxxx	修改错误指令
Error Writing output file	写输出文件错误	检查磁盘空间, 保证空间足够能够输出文件
Expression syntax error	表达式语法错误	检查是否出现两个连续的操作符、括号不匹配或缺少括号、前一语句漏掉了分号等

(续表)

出错信息提示	错误原因	处理方法
Extra parameter in call	调用时出现多余参数，实参多于形参	修改函数定义或者形参
File name too long	文件名太长	文件名长度通常不能超过 64 个字符
For statement missing )(;	for 语句缺少括号或者分号	修改 for 语句
Function call missing )	函数调用缺少 “)”	检查函数的括号匹配
Function definition out of place	函数定义位置错误	函数应该定义在主函数前，或者在主函数前添加函数说明
Function doesn't take a variable number of argument	函数不接受可变的参数个数	将函数的参数个数定义为常量
Goto statement missing label	goto 语句缺少标号	添加 goto 语句的标号
If statement missing )	If 语句缺少括号	添加左括号或者右括号
Illegal initialization	非法初始化	修改初始化语句
Illegal octal digit	非法八进制数	检查八进制常数中是否包含了非八进制数字
Illegal pointer subtraction	非法指针相减	非指针变量减去指针变量，参考主教材指针运算
Illegal structure operation	非法结构操作	参考主教材结构运算
Illegal use of floating point	浮点运算非法	参考主教材浮点数运算
Illegal use of pointer	指针使用非法	关于指针的使用请参见主教材
Incompatible storage class	不相容的存储类型	检查是否有不相容的存储类型在进行操作
Incompatible type conversion	不相容的类型转换	参考数据类型的强制转换
Incorrect command line argument:xxxxxx	不正确的命令行参数: xxxxxxx	修改命令行参数或者操作
Incorrect command file argument:xxxxxx	不正确的配置文件参数: xxxxxxx	修改配置文件参数
Incorrect number format	不正确的数据格式	修改数据格式，如浮点数的科学记数法
Initializer syntax error	初始化语法错误	对照主教材各个章节中关于变量初始化的内容进行修改
Invalid indirection	无效的间接运算	间接运算操作符 (*) 要求非 void 指针作为操作分量
Invalid macro argument separator	无效的宏参数分隔符	宏参数间的分隔符必须以逗号相隔
Invalid pointer addition	无效的指针相加	参考主教材指针运算
Invalid use of dot	点使用错	需要注意的是在结构、枚举等数据类型中的使用方法
Invalid use of arrow	箭头使用错	->右边必须跟标识符
Macro argument syntax error	宏参数语法错误	赋值操作符左边必须是一个地址表达式

(续表)

出错信息提示	错误原因	处理方法
Macro expansion too long	宏扩展太长	以上两个错误参考主教材宏使用方法
Mismatch number of parameters in definition	定义中参数个数不匹配	修改参数个数以及类型
Misplaced decimal point	指数部分使用了小数点	修改为整数
Misplaced *****	*****位置错误	修改错误位置
Must be addressable	必须是可编址的	修改变量或者内存地址
Must take address of memory location	必须是内存一地址	检查内存地址值是否正确
No file name ending	无文件终止符	添加文件终止符
No file names given	未给出文件名	以上两个错误请参考文件章节的内容
Non-portable pointer assignment(comparison)	对非指针的错误操作	修改指针赋值操作使其合法化
Non-portable return type Conversion	不可移植的返回类型转换	在返回语句中的表达式类型与函数说明中的类型不同,但如果一函数的返回表达式是一指针,则可以进行转换,此时,返回指针的函数可能送回一个常量零,而零被转换成一个适当的指针值
Not an allowed type	不允许的类型	说明了禁止的类型
Out of memory	内存不够	
Pointer required on left side of	操作符左边须是一指针	修改指针赋值操作
Parament is never used in function	调用函数时, 参数少于该函数的参数	修改实参类型或者数量
Redeclaration of 'xxxxxx'	变量重复定义	只保留一句变量定义
Size of structure or array not known	结构或数组大小不定	修改结构或者数组的大小, 使用常量
Statement missing;	语句缺少 “; ”	添加 “; ”
Structure or union syntax error	结构或联合语法错误	参考主教材相关章节内容
Structure size too large	结构太大	减少结构中元素个数
Too few parameters in call	函数调用参数太少	修改实参的类型或者数量
Too few parameter in call to xxxxxx	调用 “xxxxxx” 时参数太少	修改实参的类型或者数量
Too many cases	case 太多	减少 case 语句
Too many decimal points	十进制小数点太多	修改常量的表达式
Too many default cases	在 case 语句中 default 多于一个	减少 default 个数
Too many exponents	阶码太多	
Too many initializers	变量的初始化过多	
Too many storage classes in declaration	说明中存储类太多	

(续表)

出错信息提示	错误原因	处理方法
Too many types in declaration	说明中类型太多	
Too much auto memory in function	函数中自动存储太多	
Too much global define in file	文件中定义的全局数据太多	
Two consecutive dots	使用了两个句号	
Type mismatch in parameter #	参数“#”类型不匹配	
Type mismatch in parameter # in call to XXXXXXXX	调用 XXXXXXXX 时参数#类型不匹配	修改类型
Type mismatch in parameter XXXXXXXX	参数 XXXXXXXX 类型不匹配	
Type mismatch in parameter XXXXXXXX in call to YYYYYYYY	调用 YYYYYYYY 时参数 XXXXXXXX 数型不匹配	修改实参类型或者数量
Type mismatch in redeclaration of XXX	重定义类型不匹配	
Unable to creat output file XXXXXXXX.XXX	不能创建输出文件 XXXXXXXX.XXX	检查磁盘已满或者写保护
Unable to execute command 'xxxxxxx'	不能执行“xxxxxxx”命令	检查拼写错误和命令行参数错误
Unable to open include file 'xxxxxxx.xxx'	不能打开包含文件“xxxxxxx.xxx”	检查拼写错误, 或者检查系统文件是否包含文件 xxxxxx.xxx
Undefined xxx yyy	类型 xxx 的变量 yyy 未定义	定义错误、使用错误或类型说明错误
Unexpected end of file in comment started on line #	源文件在某个注释中意外结束	检查注释的语法是否有错误
Unexpected end of file in conditional stated on line #	源文件在#行开始的条件语句中意外结束	检查#右边的表达式
Unknown preprocessor directive 'xxx'	不认识的预处理指令: xxx	检查拼写和语法错误
Unterminated character constant	未终结的字符常量	右边添加单引号
Unterminated string or character constant	未终结的字符串或字符常量	右边添加引号
User break	用户中断	
value required	未赋值	
While statement missing )	While 语句漏掉 ()	
Wrong number of arguments in of 'xxxxxxx'	错误的参数个数	修改实参类型或者数量