- 解析经典C语言编程实例
- 汇集C语言特色编程技巧
- 提供全部源代码和可执行文件

C语言

实战105例

王为青　张圣亮　编著

内附光盘

人民邮电出版社
POSTS & TELECOM PRESS

C语言实战105例

王为青　张圣亮　编著

人民邮电出版社
北　京

图书在版编目（CIP）数据

C 语言实战 105 例 / 王为青，张圣亮编著. —北京：人民邮电出版社，2007.3

ISBN 978-7-115-15567-2

Ⅰ. C… Ⅱ. ①王… ②张… Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2006）第 147814 号

内 容 提 要

本书共汇集 105 个实例，内容循序渐进，通过实例讲述 C 语言编程。全书分为 8 篇，包括基础篇、数值计算与数据结构篇、文本屏幕与文件操作篇、病毒与安全篇、图形篇、系统篇、游戏篇、综合篇，基本涵盖了目前 C 语言编程的各个方面。

本书全部以实例为线索展开讲解，注重对实例的分析、对方法的详细讲解以及对知识点的归纳总结。书中通过实例来讲解知识点，同时又通过相应的知识点来分析实例，二者相辅相成。

通过阅读本书，初学者不会再为编写程序时无从下手而苦恼，具有一定 C 语言基础的读者也不会再原地踏步，停滞不前。因此，本书不仅可以帮助初学者快速入门，也可帮助中级读者在 C 语言程序设计的殿堂中迈进。

C 语言实战 105 例

◆ 编　　著　王为青　张圣亮
责任编辑　李　岚

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：18.75
字数：470 千字　　2007 年 3 月第 1 版
印数：1 – 5 000 册　　2007 年 3 月北京第 1 次印刷

ISBN 978-7-115-15567-2/TP

定价：36.00 元（附光盘）

读者服务热线：(010)67132692　印装质量热线：(010)67129223

光盘使用说明

本书附带了一张光盘，光盘的内容和使用方法如下。

程序运行环境

硬件环境：计算机 CPU 的主频在 500MHz 以上，内存在 128MB 以上。

软件环境：操作系统是 Windows 98/Me/2000/NT/XP，调试环境是 Turbo C++。另外，“病毒与安全篇”的部分实例需要在 Linux 操作系统中使用 GCC 来调试、运行。

光盘主要内容

1．\code 目录下，包括全书 105 个实例的所有源代码、可执行程序、相应的数据文件、使用说明文件。

2．\tool 目录下，包括一个 Turbo C++编译器。

光盘的使用方法

首先确定 Turbo C++的安装路径（如安装路径是“C:\TC”）。

然后拷贝相应的源代码到此目录（如将\code\001、目录下的 1.c 拷贝到目录 C:\TC 下），并且去掉其只读属性。这样就可以使用 Turbo C++编译器来编译相应的源代码。

Turbo C++的使用方法

首先在 MS-DOS 方式下运行 TC.exe，然后按下面步骤来编辑、编译和运行程序。

1．打开源文件

按“F3”或者执行“File→Open”打开源文件 1.c。如图 1 所示。

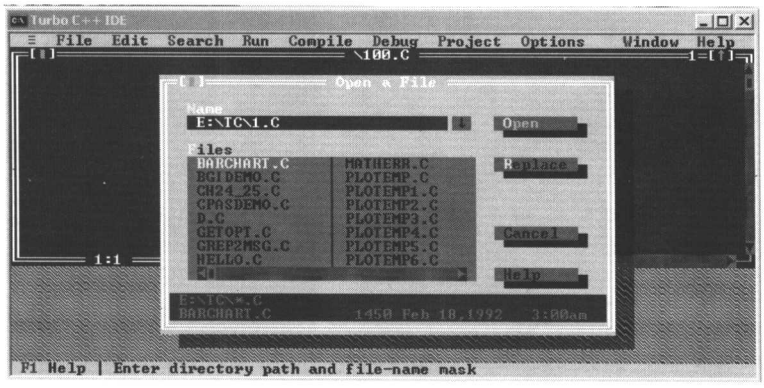

图 1　打开源文件界面

2．编译程序

按“Alt+F9”或者执行“Compile→Compile”来编译源文件。如图 2 所示。

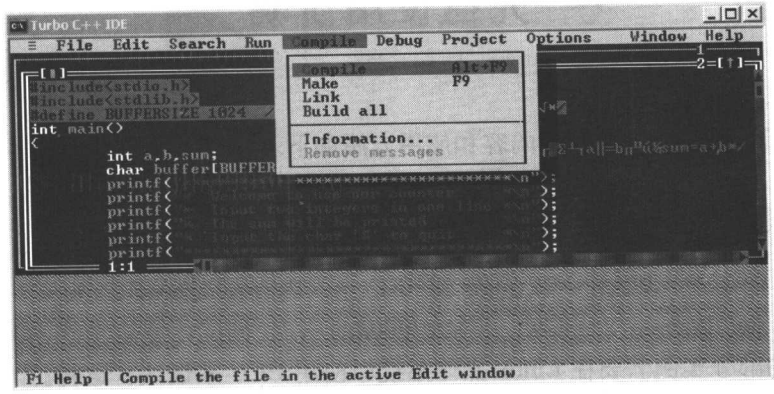

图 2　编译程序界面

3．执行程序

按“Ctrl+F9”或者执行“Run→Run”来执行程序。如图 3 所示。

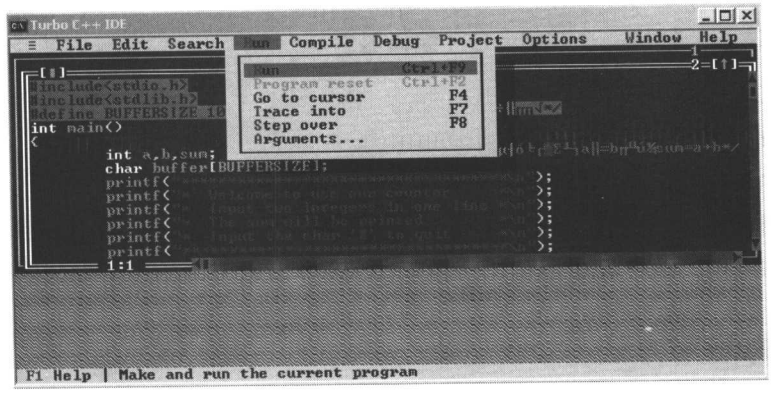

图 3　程序的执行界面

前　言

为什么要学习 C 语言

C 语言是目前被广泛使用的一种基本编程语言，在计算机软件设计中得到了大量应用。它之所以如此流行，最主要的原因就在于它的高效。优秀 C 程序的效率几乎和汇编程序一样高，但是它比汇编语言更易于程序员理解掌握。C 语言还提供了汇编语言的接口，这使得 C 语言成为实现操作系统和嵌入性控制器软件的良好选择。C 语言流行的另外一个原因是具有良好的可移植性。目前，C 编译器在许多机器上都已经实现，ANSI 标准提高了 C 程序在不同机器之间的可移植性。

几乎所有的大专院校都开设了 C 语言程序设计课程，并且有相当一部分高校将其作为计算机专业的基础课程在一年级开设。C 语言不仅是所有软件课程的基础，而且是从事计算机专业的科技人员的一门专业基础课。

本书的优势何在

目前市面上有多种讲解 C 语言程序设计的书籍，也包括一些实例书。本书与它们比起来，无论在内容上，还是在结构安排上都有很鲜明的特点。

本书在内容上，并不是枯燥地讲解知识点，而是以 105 个实例为线展开讲解，注重对实例的分析，对方法的详细讲解，以及对知识点的归纳总结。本书通过实例来讲解知识点，又通过相应的知识点来分析实例。通过阅读本书，初学者不会再为编写程序时无从下手而苦恼，具有一定 C 语言基础的读者也不会再原地踏步，停滞不前。因此，这本书不仅可以帮助初学者快速入门，也会帮助有一定基础的读者在 C 语言程序设计的殿堂继续迈进。

另外，本书在结构安排上，充分考虑了层次性，内容循序渐进。本书将 105 个实例共分为八篇。

基础篇介绍 C 语言编程的基础知识，包括 C 语言的输入输出、数据类型、数组、指针、函数、结构体等相关内容。这部分内容适合读者学习和巩固 C 语言的基础知识，并且指导读者如何灵活运用这些基础知识进行程序设计。

数值计算与数据结构篇包括 0-1 背包问题、中奖彩球问题、储油问题、阶梯计数问题，等多个经典问题。另外，此部分还介绍了常用的数据结构算法，包括排序算法、栈与队列的应用、串操作的实现、图的相关算法等。通过这部分的学习，读者可以逐步建立起算法的思想。

文本屏幕与文件操作篇介绍文件的基本操作和一些实用的文件处理方法。包括文件的加密和解密，两个文件的连接和合并、文件的分割、两个文件内容的同时显示等。通过此部分的学习，读者将会逐步掌握一些实用的文件处理技巧。

病毒与安全篇主要介绍常见病毒的分析与监测，常用的数字加密算法等内容。另外，还实

现了 traceroute、ping 等常用的网络命令。此部分旨在让读者认识病毒，掌握相关原理。

图形篇介绍如何使用 Turbo C 提供的图形函数绘制基本的图形，包括绘制直线、圆、矩形等；如何使用这些基本的图形完成复杂图形的绘制，包括柱状图的应用、三维视图的绘制、绘制按钮、制作音乐动画等。通过本部分的学习，读者将逐步掌握如何使用 C 语言绘制图形。

系统篇主要包括读取系统中的配置信息、鼠标中断处理、获取网卡信息、硬件测试、管道通信等内容。

游戏篇介绍 DOS 环境下的 C 语言游戏编程。包括俄罗斯方块、24 点牌、弹力球、贪吃蛇、潜艇大战、机器人大战、十全十美等经典游戏。

综合实例篇包括通信录、竞技比赛打分系统、实现个人理财等实用程序。本部分将重点向读者介绍如何设计综合的 C 程序，提高读者编写大型程序的能力。

本书的绝大多数实例均在 Turbo C++环境中调试通过，“病毒与安全篇”的部分实例是在 Linux 环境下使用 gcc 编译通过的。

编 者

2006 年 12 月

目　录

第 1 部分　基　础　篇

实例 1　一个价值“三天”的 BUG …… 2
实例 2　灵活使用递增（递减）操作符 …… 5
实例 3　算术运算符计算器 …… 7
实例 4　逻辑运算符计算器 …… 9
实例 5　IP 地址解析 …… 11
实例 6　用 if...else 语句解决奖金发放问题 …… 13
实例 7　用 for 循环模拟自由落体 …… 16
实例 8　用 while 语句求 n！ …… 19
实例 9　模拟银行常用打印程序 …… 22
实例 10　使用一维数组统计选票 …… 26
实例 11　使用二维数组统计学生成绩 …… 29
实例 12　简单的计算器 …… 32
实例 13　时钟程序 …… 35
实例 14　华氏温度和摄氏温度的相互转换 …… 38
实例 15　SimpleDebug 函数应用 …… 40

第 2 部分　数值计算与数据结构篇

实例 16　常用的几种排序方法 …… 46
实例 17　广度优先搜索及深度优先搜索 …… 53
实例 18　实现基本的串操作 …… 59
实例 19　计算各点到源点的最短距离 …… 62
实例 20　储油问题 …… 65
实例 21　中奖彩球问题 …… 67
实例 22　0-1 背包问题 …… 69
实例 23　阶梯计数问题 …… 72
实例 24　二叉树算法集 …… 74
实例 25　模拟 LRU 页面置换算法 …… 79
实例 26　大整数阶乘新思路 …… 82
实例 27　银行事件驱动模拟程序 …… 84

实例 28　模拟迷宫探路 …… 87
实例 29　实现高随机度随机序列 …… 89
实例 30　停车场管理系统 …… 91

第 3 部分　文本屏幕与文件操作篇

实例 31　菜单实现 …… 96
实例 32　窗口制作 …… 97
实例 33　模拟屏幕保护程序 …… 100
实例 34　文件读写基本操作 …… 102
实例 35　格式化读写文件 …… 105
实例 36　成块读写操作 …… 107
实例 37　随机读写文件 …… 108
实例 38　文件的加密和解密 …… 111
实例 39　实现两个文件的连接 …… 113
实例 40　实现两个文件信息的合并 …… 116
实例 41　文件信息统计 …… 118
实例 42　文件分割实例 …… 121
实例 43　同时显示两个文件的内容 …… 123
实例 44　模拟 Linux 环境下的 vi 编辑器 …… 124
实例 45　文件操作综合应用——银行账户管理 …… 128

第 4 部分　病毒与安全篇

实例 46　实用内存清理程序 …… 134
实例 47　如何检测 Sniffer …… 136
实例 48　加密 DOS 批处理程序 …… 139
实例 49　使用栈实现密码设置 …… 141
实例 50　远程缓冲区溢出漏洞利用程序 …… 144
实例 51　简易漏洞扫描器 …… 146
实例 52　文件病毒检测程序 …… 149
实例 53　监测内存泄露与溢出 …… 150
实例 54　实现 traceroute 命令 …… 152
实例 55　实现 ping 程序功能 …… 154
实例 56　获取 Linux 本机 IP 地址 …… 157
实例 57　实现扩展内存的访问 …… 161
实例 58　随机加密程序 …… 164
实例 59　MD5 加密程序 …… 165

实例 60　RSA 加密实例 …… 168

第 5 部分　图　形　篇

实例 61　制作表格 …… 172
实例 62　用画线函数作出的图案 …… 174
实例 63　多样的椭圆 …… 176
实例 64　多变的立方体 …… 177
实例 65　简易时钟 …… 178
实例 66　跳动的小球 …… 181
实例 67　用柱状图表示学生成绩各分数段比率 …… 183
实例 68　EGA/VGA 屏幕存储 …… 187
实例 69　按钮制作 …… 190
实例 70　三维视图制作 …… 193
实例 71　红旗图案制作 …… 194
实例 72　火焰动画制作 …… 196
实例 73　模拟水纹扩散 …… 199
实例 74　彩色的 Photo Frame …… 201
实例 75　火箭发射演示 …… 203

第 6 部分　系　统　篇

实例 76　恢复内存文本 …… 208
实例 77　挽救磁盘数据 …… 210
实例 78　建立和隐藏多个 PRI DOS 分区 …… 213
实例 79　简单的 DOS 下的中断服务程序 …… 216
实例 80　文件名分析程序 …… 219
实例 81　鼠标中断处理 …… 222
实例 82　实现磁盘数据的整体加密 …… 224
实例 83　揭开 CMOS 密码 …… 227
实例 84　获取网卡信息 …… 229
实例 85　创建自己的设备 …… 231
实例 86　设置应用程序启动密码 …… 233
实例 87　获取系统配置信息 …… 236
实例 88　硬件检测 …… 239
实例 89　管道通信 …… 241
实例 90　程序自杀技术实现 …… 244

第7部分 游 戏 篇

实例 91 连续击键游戏 …… 248
实例 92 掷骰子游戏 …… 250
实例 93 弹力球 …… 252
实例 94 俄罗斯方块 …… 253
实例 95 24 点扑克牌游戏 …… 257
实例 96 贪吃蛇 …… 260
实例 97 潜水艇大战 …… 262
实例 98 机器人大战 …… 265
实例 99 图形模式下的搬运工 …… 266
实例 100 十全十美游戏 …… 269

第8部分 综 合 篇

实例 101 强大的通信录 …… 274
实例 102 模拟 Windows 下 UltraEdit 程序 …… 277
实例 103 轻松实现个人理财 …… 279
实例 104 竞技比赛打分系统 …… 281
实例 105 火车订票系统 …… 286

基础篇

实例 1　一个价值“三天”的 BUG
实例 2　灵活使用自增（自减）操作符
实例 3　算术运算符计算器
实例 4　逻辑运算符计算器
实例 5　IP 地址解析
实例 6　用 if...else 语句解决奖金发放问题
实例 7　用 for 循环模拟自由落体
实例 8　用 while 语句求 n!
实例 9　模拟银行常用打印程序
实例 10　使用一维数组统计选票
实例 11　使用二维数组统计学生成绩
实例 12　简单的计算器
实例 13　时钟程序
实例 14　华氏温度和摄氏温度的相互转换
实例 15　SimpleDebug 函数应用

实例 1　一个价值“三天”的 BUG

实例说明

本实例将向读者展示如何使用 sscanf 函数处理行定向的输入。为了帮助说明本实例，程序中实现了输出两个从键盘输入的整数的和的功能。笔者希望通过本例让读者更好地理解 scanf 函数家族的使用，并提醒读者这些函数在使用过程中一些易犯的错误。笔者曾花费“三天”的时间来发现这些错误，这也是本例的名字由来。程序运行结果如图 1.1 和图 1.2 所示。

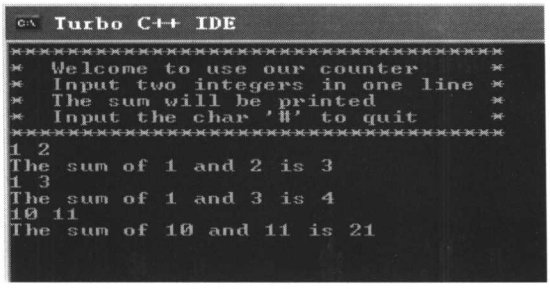

图 1.1　使用 sscanf 处理行定向的输入的运行结果

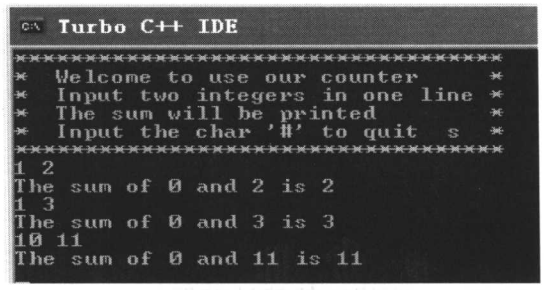

图 1.2　第 1 次修改后的运行结果

实例解析

ANSI I/O 提供了 3 种格式化的行 I/O——scanf 函数家族。scanf 函数家族的原型如下所示。每个函数原型中的省略号表示一个可变长度的指针列表。从输入转换而来的值逐个存储到这些指针参数所指向的内存位置。

```
int fscanf(FILE * stream,char const *format,...)
int scanf(char const *format,...)
int sscanf(char const *string,char const *format,...)
```

这些函数的功能都是从输入源读取字符，然后根据 format 字符串指定的格式代码对读入的字符进行相应转换。fscanf 的输入源是作为参数给出的流（FILE * stream），scanf 的输入源是标准输入，而 sscanf 则从第 1 个参数（char const *string）给出的字符串中读入字符。

当格式化字符串到达末尾或者读取的输入不再匹配格式字符串所指定的类型时，输入就停止。函数的返回值就是被转换的输入值的数目。scanf 函数家族中的 format 字符串参数包含的字符有空白字符、格式代码和其他字符。其中，空白字符可以与输入中的任意个空白字符相匹配，在处理过程中会被忽略；格式代码就是要指定函数如何解释接下来的输入字符；除了空白字符和格式代码之外的其他字符在格式字符串中可以出现，也可以不出现，如果它们出现在格式字符串中，下一个输入的字符必须与之匹配，如果匹配，该输入字符将被丢弃，如果不匹配，函数就不再读取而直接返回。下面详细介绍一下格式代码。

格式代码就是一个字符，用于指定输入的字符如何被解释。它的开始标志是一个百分号（%），百分号后面可以跟如下 4 种字符。

（1）星号（*）：并不存储转换后的值，而是将其丢弃，可以用来跳过不需要的输入字符。

（2）宽度（一个非负整数）：用来限制被读取转换的字符个数，在没有指定宽度的情况下，函数会连续读入字符直到遇到空白字符为止。

（3）限定符：用于修改有些格式代码的含义，比如在格式代码“%d”中使用限定符“h”，即“%hd”，那么“d”表示的不再是默认整型，而是 short int。

（4）格式代码：也就是说格式代码后面还可以有格式代码。

表 1.1 中列出了 scanf 函数家族中常用的格式码。

表 1.1　　scanf 中的基本格式码

格 式 码	含　义
d	将输入解释为十进制整型数据，并按照有符号数存储
u	将输入解释为十进制整型数据，并按照无符号数存储
x	将输入解释为十六进制整型数据，并按照无符号数存储
o	将输入解释为八进制整型数据，并按照无符号数存储
c	期待输入一个字符
s	期待输入一个字符串
f	期待输入一个浮点数
e	期待输入一个浮点数

在使用 scanf 函数家族的时候，读者需要特别注意两点。

（1）指针参数的类型必须是对应格式代码的正确类型。

由于 C 语言采用传址参数传递机制，把内存位置作为参数传递给函数的惟一方法就是传递一个指向该位置的指针。在使用 scanf 函数家族的时候，一个非常容易出现的错误就是忘记加上“&”。省略地址符号“&”将导致将变量的值作为参数传递给函数，而 scanf 函数家族却把它解释为指针。这会导致一个不可预料的内存位置的数据被改写。

（2）要正确使用限定符。

限定符的目的是为了指定参数的长度。如果整形参数比缺省的整形参数更长或者更短时，在格式代码中省略限定符就是一个常见的错误。对于浮点类型也是如此，如果省略了限定符，可能导致一个较长的变量只有部分被初始化，或者一个较短变量的邻近变量也被修改（这取决于它们的相对长度）。笔者在第一次犯这个错误的时候，花费了三天的时间才找出这个错误。这就是本实例为什么称为“一个价值‘三天’的 BUG”。这个错误应该引起读者足够的重视。下面将通过修改程序 1.1 后的程序 1.2 来向读者演示这个错误。这个问题的存在也取决于使用的 C 库的版本，如果您使用的是 TC 2.0 及以下版本，或者您的运行环境是 Linux，这个问题就会存在。但是，如果正确地使用了限定符，那么这个问题就迎刃而解了。

程序代码

【程序 1.1】　使用 sscanf 处理行定向的输入

```
#include<stdio.h>
#include<stdlib.h>
#define BUFFERSIZE 1024   /*允许处理的最长行有 1024 个字符*/
int main()
```

```
{
    unsigned int a,b,sum;              /*将输入的两个数分别存储在变量 a 和 b 中，sum=a+b*/
    char buffer[BUFFERSIZE];
    printf("*************************************\n");
    printf("*   Welcome to use our counter      *\n");
    printf("*   Input two integers in one line  *\n");
    printf("*   The sum will be printed         *\n");
    printf("*   Input the char '#' to quit      *\n");
    printf("*************************************\n");
    /*从标准输入（stdin）读取输入的数据，存储在 buffer 中。
      如果读取的第一个字符是'#'则退出程序*/
    while((fgets(buffer,BUFFERSIZE,stdin)!=NULL)&&(buffer[0]!='#'))
    {
        if(sscanf(buffer,"%d %d",&a,&b)!=2)          /*处理存储在 buffer 中的一行数据*/
        {
            printf("The input is skipped:%s",buffer);  /*如果输入的数字不是两个则报错*/
            continue;                                  /*继续读取下一组数据*/
        }
        sum=a+b;                                       /*计算 a 与 b 的和*/
        printf("The sum of %d and %d is %d\n",a,b,sum);/*输出计算结果*/
    }
    return 0;
}
```

【程序 1.2】　一个价值“三天”的 BUG

```
int main()
{
    unsigned short a,b,sum;            /*将输入的两个数分别存储在变量 a 和 b 中，sum=a+b*/
    /*此程序段与程序 1.1 中相同*/
    return 0;
}
```

归纳注释

本实例中使用到了 sscanf 来处理缓冲区 buffer 中的数据，此外还使用了 fgets 函数，它的函数原型如下：

```
char *fgets(char *buffer,int BUFFER_SIZE,FILE *stream)
```

fgets 的功能是从指定的 stream 中读取字符并把它们复制到 buffer 中。本实例中 stream 被指定为标准输入流（stdin）。当它读取一个换行符并存储到缓冲区之后就不再读取。

读者可以比较程序 1.1 和修改后的程序发现，笔者只是将 a，b 的声明变成了 unsigned short 类型。这时错误的运行结果（如图 1.2 所示）就产生了。因为 sscanf 在处理整型数据时，默认的

整型是 int 类型的，而笔者声明的类型 unsigned short 比缺省的整型值短，这就导致较短变量的邻近变量也被修改，根本原因就在于忽略了限定符的使用。因此修正程序（只列出程序的修改部分）：

```
if(sscanf(buffer,"%hd %hd",&a,&b)!=2)  /*处理存储在 buffer 中的一行数据，使用了限定符*/
```

使用了限定符之后的结果如图 1.1 所示。表 1.2 列出了限定符对常用的格式码的含义的修改。

表 1.2　　使用限定符后格式代码的含义

格 式 代 码	限 定 符 h	限 定 符 l	限 定 符 L
d	short	long	
u o x	unsigned short	unsigned long	
e f		double	long double

实例 2　灵活使用递增（递减）操作符

实例说明

本实例演示了一个整型变量的自增和自减运算结果，以此来说明自增和自减操作符的使用方法，包括它们的基本意义、优先级和结合性。同时也说明了在使用自增自减操作符时易犯的错误。程序的运行结果如图 2.1 所示。

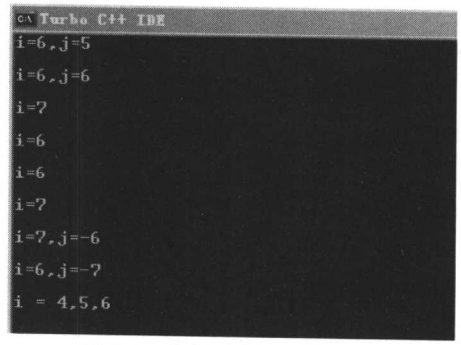

图 2.1　实例 2 程序运行结果

实例解析

自增 1 运算符记为“++”，其功能是使变量的值自增 1。自减 1 运算符记为“--”，其功能是使变量值自减 1。自增和自减运算符的两种用法如下。

（1）前置运算：运算符放在变量之前，比如，++i、--i，其中 i 是一个变量。这种方式的运算规则是，先使变量的值增（或减）1，然后再以变化后的值参与其他运算。

（2）后置运算：运算符放在变量之后，比如，i++、i--，其中 i 是一个变量。这种方式的运算规则是变量先参与其他运算，然后再使变量的值增（或减）1。

自增、自减运算，常用于循环语句中，使循环控制变量加（或减）1，以及指针变量中，使指针指向下（或上）一个地址。但是增 1、减 1 运算都要求运算对象是变量，不能用于常量和表达式。例如，1++、--(x+y)等都是非法的。

程序代码

【程序 2】　自增（自减）操作符的使用

```
#include <stdio.h>
int main()
{
    int i=5,j;
    /*定义了两个整型变量 i 和 j，并对变量 i 进行初始化*/
    clrscr();
    /*清除屏幕*/
    j=i++;
    /*将 i 的值赋予 j 之后，i 自增 1*/
    printf("i=%d,j=%d\n",i,j);
    i=++j;
    /*先使 j 自增 1，然后将 j 的值赋与 i*/
    printf("\ni=%d,j=%d\n",i,j);
    printf("\ni=%d\n",++i);
    /* i 自增 1 后，输出 i*/
    printf("\ni=%d\n",--i);
    /* i 自减 1 后，输出 i*/
    printf("\ni=%d\n",i++);
    /*输出 i 后，  i 自增 1*/
    printf("\ni=%d\n",i--);
    /*输出 i 后，i 自减 1*/
    j=-i++;
    /*取 i 的值加上负号后赋予 j，然后 i 自增*/
    printf("i=%d,j=%d\n",i,j);
    j=-i--;
    /*取 i 的值加上负号后赋予 j，然后 i 自减*/
    printf("\ni=%d,j=%d\n",i,j);
    printf("\ni = %d, %d, %d\n",i,i--,i--);
    getchar();
    return 0;
}
```

归纳注释

自增 1，自减 1 运算符优先级比*、%和/都要高。在本实例中，首先看下面两条语句：

```
j=-i++;
printf("i=%d,j=%d\n",i,j);
```