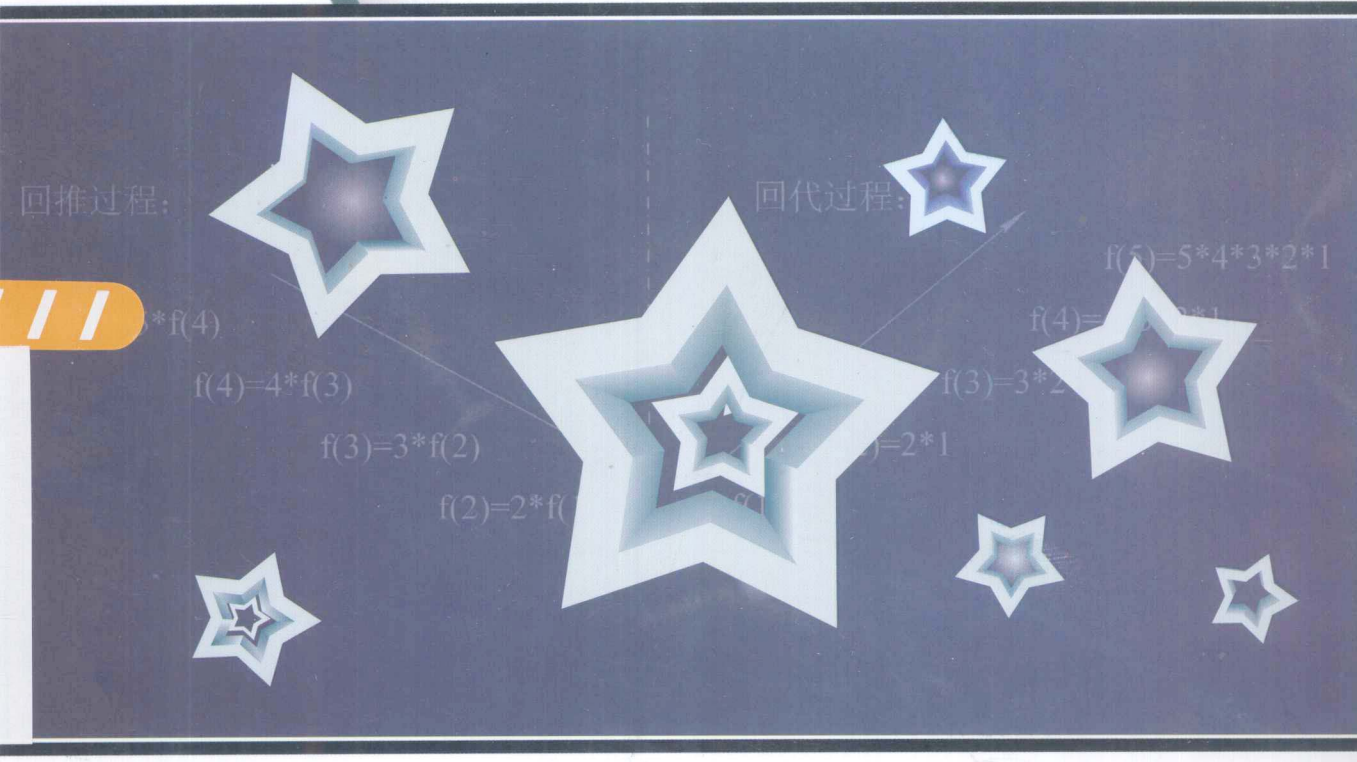
高等学校应用型“十二五”规划教材 • 计算机类
校级重点课改项目成果之一

项目导向——C语言嵌入式应用编程

主　编　何光普
副主编　张自友



西安电子科技大学出版社
http://www.xduph.com

高等学校应用型“十二五”规划教材·计算机类

项目导向——

C语言嵌入式应用编程

主　编　何光普
副主编　张自友
参　编　杨济豪　祝加雄　常峰

西安电子科技大学出版社

内 容 简 介

本书基于乐山师范学院校级重点课题“电子信息工程 3CE 应用型创新人才培养模式的探索与实践”的研究成果，并在“3CE”(即“工程意识、工程能力和工程外化认证”)为核心的应用型创新人才培养模式的理念指导下，密切结合 C 语言在嵌入式方面的应用特点，删繁就简，以应用为中心，依托项目和任务构建嵌入式 C 语言的知识体系。

本书采用项目导向和任务驱动模式构建嵌入式 C 语言的知识体系。根据 C 语言特点，在编写过程中，按照项目之间的逻辑关系进行组织，同时注意循序渐进，结合实际，采用启发式和任务驱动的方法，切实加强学生对基础知识的掌握，提高学生解决实际问题的能力。整个课程教学以四个项目为依托，细分为若干个子任务，从而囊括了课程要求的全部知识点。

本书既可以作为一般本科院校或高职高专电子信息类专业的 C 语言入门级教学用书，还可以作为计算机爱好者的自学参考书和计算机培训班的教材。

图书在版编目(CIP)数据

项目导向：C 语言嵌入式应用编程/何光普主编.
—西安：西安电子科技大学出版社，2013.4
高等学校应用型“十二五”规划教材
ISBN 978-7-5606-3041-0

Ⅰ. ① 项… Ⅱ. ① 何… Ⅲ. ① C 语言—程序设计—高等学校—教材
Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字(2013)第 061784 号

策　　划　李惠萍　张　绚
责任编辑　李惠萍　郭雨薇
出版发行　西安电子科技大学出版社（西安市太白南路 2 号）
电　　话　(029)88242885　88201467　邮　　编　710071
网　　址　www.xduph.com　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西天意印务有限责任公司
版　　次　2013 年 4 月第 1 版　2013 年 4 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 14.5
字　　数　344 千字
印　　数　1～3000 册
定　　价　25.00 元
ISBN 978 – 7 – 5606 – 3041 – 0 / TP
XDUP 3333001-1

前　言

本书采用工学结合和任务驱动的模式编写。在实施过程中，以一个有趣的项目“双色球摇奖机”作为引导，再通过循序渐进的 3 个项目，即“学生成绩管理系统”、“十字路口交通灯系统”、“简易数字钟”逐步展开。通过对项目的分析，又将其分成若干个具体的任务，每个任务都包含着 C 语言的若干个知识点和技能点，如数据类型、输入输出函数、顺序结构、选择语句、循环语句、数组、函数、指针和结构体等。本书强调“任务”的目标性和教学情境的创建，使学生带着真实的任务在探索中学习。

本书以注重培养学生的实践能力为前提，理论知识传授遵循“实用为主、够用为度”的准则，基本知识广而不深、点到为止，基本技能贯穿教学的始终，具体采用“技能需求、问题引导、任务驱动”的方式。

本着为培养应用型人才提供一套精品教材的目的，编写本书时注重体现以下特点：

(1) 以“3CE”人才培养理念为引导。以项目导向和任务驱动模式构建课程知识体系。

(2) 以建构主义学习理论为指导。无论是项目还是具体任务，都从创建情境、案例讲解、拓展应用和扩展阅读几个层次逐步展开。

(3) 注重理论联系实际，强调理论为实践服务，知识内容以够用为度，以应用为目的，力求使学生“明基本概念，懂基本原理，强实际应用”。

(4) 本书选用了大量实例，包含调试运行截图和运行结果分析，内容精练，语言简洁，图表并用，通俗易懂，便于自学。

本书由乐山师范学院何光普教授主编，张自友老师为副主编，电子科技大学邓兴成老师主审。何光普教授负责项目拟定，张自友老师负责全书统稿及编

写项目一，杨济豪老师编写项目二，祝加雄老师编写项目三，常峰老师编写项目四。

在本书的编写过程中得到了邓兴成老师的热情关心与指导，在此表示衷心感谢。

由于作者水平有限，加之时间仓促，本书不当之处在所难免，敬请读者批评指正。

编　者

2013 年 2 月

目　录

项目一

双色球摇奖机

☆ 知识技能

(1) 了解程序和程序设计语言的概念；

(2) 理解 C 程序的基本结构、源代码书写规范及风格；

(3) 掌握用程序实现简单输入输出功能的方法；

(4) 掌握常见的数据类型、运算符及优先级，理解不同数据类型间的转换和表达式的含义；

(5) 掌握表达式语句、复合语句、简单输入输出函数；

(6) 掌握选择结构和循环结构的实现方法，理解结构嵌套，掌握 continue、break 语句的使用。

☆ 项目要求

本项目拟设计一个福彩双色球摇奖机。根据福利彩票双色球摇奖规则，系统能从 01～33 中随机产生 6 个不同的数(红球)，同时从 01～16 中产生一个数(蓝球)，最终用 6 个红球和 1 个蓝球上的数字(共 7 位)组合出一个中奖号码。要求能连续摇出多期号码，并把期号和摇奖结果显示出来。

☆ 项目内容

根据项目要求，本项目可分解为以下几个任务：

任务 1-1　摇奖结果的显示——实现程序的输出功能。

任务 1-2　摇奖号码的控制——利用选择结构和 C 语言丰富的运算符实现对摇奖号码的控制。

任务 1-3　摇奖号码的连续控制——引入循环结构，实现按相同规则产生多个号码的功能。

任务 1-4　摇奖模式的选择——引入输入函数，实现人机交互功能，使得用户可以干预程序的执行，控制程序的走向。

任务 1-5　双色球摇奖机的设计——综合任务 1-1 到任务 1-4 的知识，完成项目设计。

任务 1-1　摇奖结果的显示

一、任务导读

任何程序都必须有输出显示，通过屏幕能看到程序运行的结果。本项目的第一个任务是讨论程序的输出问题。

本任务涉及 C 程序设计的入门知识。其中包含 C 程序的基本结构，程序输出功能的实现，printf()函数的基本用法和变量的定义及输出。

二、案例讲解

【例 1-1】 显示信息“I love C!!!”。

参考源代码如下：

```
1.  #include "stdio.h"
2.  main()
3.  {
4.      printf("I love C!!!");
5.      getche();
6.  }
```

程序运行结果如图 1-1 所示。

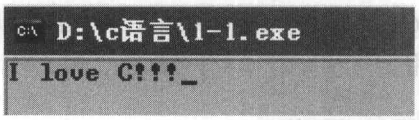


图 1-1　例 1-1 程序运行结果

说明：

(1) 这是最简单的C程序，它显示了C程序的一些必要特征：必须包含 1 个唯一的 main() 函数；函数由若干条语句构成；每条语句以分号“；”结束；函数体由一对花括号“{}”括起来。

(2) 第 1 行为文件预处理命令，stdio 是指 standard buffered input&output，意思是带缓冲的标准输入输出。stdio.h 是一个包括了输入输出函数的文件，用到标准输入输出函数时，就要用预处理命令 #include 将该文件包含进来。本例用到了 stdio.h 中声明的输出函数 printf()，括号里是其参数，该函数的基本功能是原样输出双引号里的普通字符。其详细用法会在本书后面的部分讲到。

(3) 第 5 行调用 stdio.h 文件中声明的 getche()函数，其功能类似于按任意键继续，主要作用是让程序停下来，持续显示结果，直到用户按任意键退出，程序自动关闭返回操作系统。

【例 1-2】 输出一个变量的值。

参考源代码如下：

```
1.  #include "stdio.h"
2.  main()
3.  {
```

```
4.      int x;
5.      x=100;
6.      printf("x=%d",x);
7.      getche();
8.  }
```

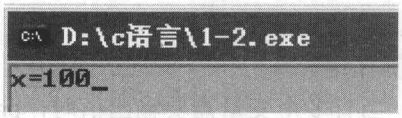


图 1-2　例 1-2 程序运行结果

程序运行结果如图 1-2 所示。

说明：

(1) 相对于例 1-1，本例引入了变量的概念并扩充了 printf() 函数的用法。

(2) 变量在程序设计语言中是一个被广泛使用的术语。变量是一段有名字的连续存储空间。在源代码中通过定义变量来申请并命名这样的存储空间，通过变量的名字来使用这段存储空间。简单来讲，变量有以下属性：变量名、变量类型、变量的值、变量代表的存储空间等。本例中变量名为 x，类型是整型 int，在程序运行期间被赋值为 100，变量占用了内存中 2 个字节的存储空间。

(3) 第 4 行为变量的定义，第 5 行是给变量赋值。C 语言规定可以在定义的同时给变量赋初值，所以第 4、5 行可以合并写成如下形式：

```
int x=100;
```

(4) 第 6 行 printf()函数中“x=”会被原样输出，而“%d”会被变量 x 的值替换后输出。“%”后面的字母表示输出的表达形式，“%d”表示十进制整数，“%f”表示浮点数(即小数)。

【例 1-3】 多个摇奖号码的输出显示。

参考源代码如下：

```
1.  #include "stdio.h"
2.  main()
3.  {
4.      int a=12,b=8,c=20;
5.      printf("the lucky number is:\n");
6.      printf("%d %d %d",a,b,c);
7.      getche();
8.  }
```

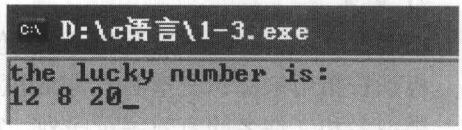


图 1-3　例 1-3 程序运行结果

程序运行结果如图 1-3 所示。

说明：

(1) 本例在第 4 行同时定义了多个变量，彼此用“ , ”隔开，这在 C 语言程序中是常采用的一种做法。当然，也可以用如下方法表示：

```
int a=12;
int b=8;
int c=20;
```

或者写成：

```
int a,b,c;
a=12;b=8;c=20;
```

相比较而言，用本例中给出的方法结构更紧凑，所以一般采用本例的紧凑形式书写代码。

(2) 第 5 行的信息属于友好提示信息，不影响结果的输出，但加上会使程序输出结果更清晰。同时 printf() 函数参数中出现了一个“\n”，其含义是换行输出，从结果中我们很容易看到其作用。

(3) 当有多个变量需要输出值时，printf() 函数的用法如第 6 行所示。为了实现多个值之间的相互间隔输出，在 printf 的参数中多个“%d”之间使用了空格；后面的参数列表中如果有多个变量，之间必须用“ , ”相互隔开。

(4) 第 6 行如果写成 printf("a=%d ,b=%d ,c=%d",a,b,c); 的形式，请读者自行分析程序输出结果。

(5) printf() 函数的具体用法涉及内容较多，请参见本任务中的扩展阅读。

三、拓展应用

设计一个程序，定义一个 int 型变量保存自己的年龄，再定义一个 float 型变量保存自己的体重，并在第 1 行利用 printf 函数输出自己的姓名的拼音，第 2 行输出自己的年龄，第 3 行输出自己的体重(单位为 kg，并保留两位小数)。

四、扩展阅读

1. 常量和变量

1) 常量

常量是指在程序运行的过程中其值不能被改变的量，如 2、4、-1.6 等。常量一般分为普通常量和符号常量。用一个标识符代表一个常量，这样的标识符称为符号常量，如用 PI 代表 3.141 592 6。

注意：符号常量的值在其作用域内不能改变，也不能再被赋值。如在程序中，对 PI 重新赋值“PI=2;”，这样是不允许的。

2) 变量

C 语言规定：在程序运行过程中其值可以被改变的量称为变量。

标识符指用来标识变量名(一般长度不能超过 8 个字符)、符号常量名、函数名、数组名、类型名和文件名的有效字符序列。它只能由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线。如 sum、average、day、month、student、_above、k_1_2_3 和 basic 等都是合法的标识符，也是合法的变量名；而 M.D.John、$123、#33、3D64、a>、-ab 等是不合法的标识符和变量名。

在 C 语言中，要求对所有用到的变量作强制定义，也就是“先定义，后使用”。

2. 整型数据

1) 整型常量

C 语言整型常量可用以下三种形式表示：

(1) 十进制整数：以数字直接开头的常量是十进制数。

(2) 八进制整数：以 0 开头的常量是八进制数。

(3) 十六进制整数：以 0x 开头的常量是十六进制数。

2) 整型变量

(1) 整型变量的分类。

整型变量有基本型、短整型、长整型和无符号型四种，其定义的关键字如下：

① 基本型：以 int 表示，范围为 -32 768～32 767。

② 短整型：以 short int 或 short 表示，范围与基本型相同。

③ 长整型：以 long int 或 long 表示，若一个整型常量后面加上一个字母 l 或 L，则认为是 long int 型常量。长整型变量的范围为 -2 147 483 648～2 147 483 647。

④ 无符号型：在实际应用中变量的值常常是正的，如年龄、工资、成绩等，因此可以将变量定义为“无符号”类型。

无符号型又分为无符号整型、无符号短整型和无符号长整型三种：

无符号整型以 unsigned int 或 unsigned 表示，范围为 0～65 535；

无符号短整型以 unsigned short 表示，范围为 0～65 535；

无符号长整型以 unsigned long 表示，范围为 0～4 294 967 295。

(2) 整型变量的定义。

变量的定义格式如下：

```
数据类型　变量表列;
```

其中若定义多个同类型的变量，则用逗号将多个变量分开。

例如，

```
int a,b;                (指定变量 a、b 为整型)
unsigned short c,d;     (指定变量 c、d 为无符号短整型)
long e,f;               (指定变量 e、f 为长整型)
```

(3) 整型数据的输入。

整型变量键盘输入是通过 scanf 函数实现的。scanf 函数是数据输入函数，格式如下：

```
scanf(格式控制，地址表列);
```

例如，

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

“格式控制”是用双引号括起来的字符串，由“%”和格式字符组成，作用是将输入数据转换为指定的格式。

对于不同的数据用不同的格式字符。d 格式符用来输入十进制整数。因为本任务中变量 a、b 是整型变量，所以输入时，使用 d 格式符。

&a,&b 中的 & 是地址运算符，&a 是指 a 在内存中的地址。上面 scanf 函数的作用是将 a、b 的值放到 a、b 在内存的地址单元中。

(4) 整型数据的输出。

整型数据的输出用 printf 函数来实现。printf 函数的格式如下：

```
printf(格式控制，输出表列);
```

例如，

```
printf("a+b=%d", c);
```

printf 函数的“格式控制”和输入函数 scanf 的“格式控制”基本一致；“输出表列”是需要输出的数据或表达式。在输出整型数据时，格式字符如下：

① %d：按整型数据的实际长度输出。

② %md：m 为输出字段的宽度。如果输出数据的位数小于 m，则左端补以空格；若大于 m，则按实际位数输出。例如：

```
printf("%4d,%4d",a,b);
```

若 a=123，b=12345，则输出结果为：V123,12345(注：V 表示空格字符，下同)。

③ %ld：输出长整型数据。例如：

```
long  a=135790;
printf("%8ld",a);
```

输出结果为：VV135790。一个 int 型数据可以用 %d 或%ld 格式输出。

④ %u：输出 unsigned 型数据，即无符号类型，如 unsigned u;，那么 u 在输出时应该用 u 格式控制符，即输出时应使用语句：

```
printf("%u",u);
```

例子：从键盘输入任意一个整数，输出这个数的平方的值。

```
main( )
{
  int a;
  long s;
  scanf("%d",&a);
  s=a*a;
  printf("s=%ld\n",s);
}
```

3. 实型数据

1) 实型常量

实数在 C 语言中又称浮点数。实数有两种表示形式：

(1) 十进制数形式。由数字和小数点组成(注意：必须有小数点)。

(2) 指数形式。注意字母 e(或 E)之前必须有数字，且 e(或 E)后面的数必须为整数，如 e3、2.1e3.5、.e3、e 都不是合法的指数形式。

2) 实型变量

已知两个数是实数，那么两数之和与积也必定为实数，所以需要设四个实型变量，分别为 a、b、sum、mul。定义语句为：

```
float a,b,sum,mul;
```

C 语言的实型变量分为：

(1) 单精度型(float 型)：一个 float 型数据在内存中占 4 个字节(32 位)。在 Turbo C 中，单精度实数的范围为 -10^{38}～10^{38}。单精度实数提供 7 位有效位，绝对值小于 10^{-38} 的数被处理成零值。

(2) 双精度型(double 型)：一个 double 型数据在内存中占 8 个字节。双精度实数的数值范围为 -10^{308}～10^{308}。双精度实数提供 15～16 位有效位，具体精确多少位与机器有关，绝对值小于 10^{-308} 的数被处理成零值。

3) 实型数据的输入和输出

实型数据的输入也是用 scanf 函数实现的，格式符使用的是 f 字符，以小数的形式输入数据，也可以使用 e 字符，以指数的形式输入数据，如使用 scanf("%f%f",&a,&b);。

实型数据的输出也是用 printf 函数实现的，格式符使用 f 字符，以小数的形式输出数据。输出时应注意：

%f：不指定字段宽度，整数部分会全部输出，并输出 6 位小数。

%m.n：指定输出数据共占 m 列，其中有 n 位小数。如果数值长度小于 m，则左端补空格。

%-m.n：指定输出数据共占 m 列，其中有 n 位小数。如果数值长度小于 m，则右端补空格。若是双精度型变量输出，应使用 %lf 格式控制，例如：double f;，输出时应使用语句：

```
printf("%lf" ,f);
```

4．字符型数据

1) 字符常量

C 语言的字符常量是用一对单引号括起来的单个字符，如 'a'、'b'、'x'、'D'、'?'、'$' 等都是字符常量。除了这样的字符常量外，C 语言还允许使用一种特殊形式的字符常量，即以一个“\”开头的字符序列。例如，前面已经用到，在 printf 函数中的“\n”，它表示一个“换行符”。

例子：字符常量的输出。

```
main( )
{
    printf("ab c\n\tde");
}
```

程序运行结果：

```
ab c
de
```

2) 字符变量

设两个字符型变量 c1 和 c2。定义形式如下：

```
char   c1,c2;
```

它表示 c1 和 c2 为字符型变量，各自可以存放一个字符。可以用下面语句对 c1、c2 赋值：

```
c1='a';   c2='b';
```

因此在内存中一个字符变量只占一个字节。

3) 字符数据的存储形式

字符在内存中存储的不是字符本身，而是它的 ASCII 码，如字符 'a' 的 ASCII 码为 97、'b' 的 ASCII 码为 98，因此字符的存储形式与整数的存储形式是类似的。在 C 语言中，字符型数据和整型数据是通用的。

例子：字符变量的不同输出形式。

```
main( )
{
  char a1,a2;
  a1=97; a2=98;
  printf("%c %c\n",a1,a2);
  printf("%d %d\n",a1,a2);
}
```

程序运行后输出如下：

```
a b
97 98
```

字符型数据和整型数据是通用的，但应该注意，字符数据只占一个字节，它只能存放0～255 范围内的整数。

例子：大小写字母的转换。

```
main( )
{
  char a1,a2;
  a1='a'; a2='b';
  a1=a1 – 32; a2=a2 – 32;
  printf("%c %c\n",a1,a2);
}
```

程序运行结果为：

```
A  B
```

该程序的作用是将两个小写字母 a 和 b 转换成大写字母 A 和 B。'a' 的 ASCII 码为 97，而 'A' 为 65；'b' 为 98，'B' 为 66。从 ASCII 码表(附表 1)中可以看到，每一个小写字母比其相应的大写字母的 ASCII 码大 32。

4) 字符型数据的输入和输出

例子：用 getchar 和 putchar 函数对字符变量进行输入和输出。

```
#include "stdio.h"
main( )
{
  char c;
  c=getchar( );
  putchar(c);
}
```

程序运行结果：

```
a↙        (输入'a' 后，按回车键)
a         (输出变量 c 的值 'a')
```

注意：getchar()只能接收一个字符，putchar()也只能向终端输出一个字符。在使用 getchar 函数和 putchar 函数时，程序的首部需使用预编译命令 #include "stdio.h"。

例子：用 scanf 和 printf 函数对字符变量进行输入和输出。

```
main( )
{
  char c1,c2;
  scanf("%c%c",&c1,&c2);
  printf("%c%c",c1,c2);
}
```

程序运行结果：

```
ab↙
ab
```

在使用 scanf 函数和 printf 函数输入输出字符型数据时，使用 %c 格式控制，用来输入输出单个字符。注意，在用 %c 格式输入字符时，空格将以有效字符输入，如 scanf("%c%c%c",&c1,&c2,&c3);，若输入 aVbVc↙，则将字符 'a' 送给 c1，字符空格 'V' 送给 c2，因为空格也是一个有效字符，字符 'b' 送给 c3。%c 只读入一个字符，如果用空格做间隔就会出现这样的问题。

5) 字符串常量

字符串常量是用双引号括住的字符序列，如 "How do you do"、"CHINA"、"a" 等都是字符串常量。字符串常量可以输出一个字符串，如 printf("How do you do. ");。注意：不要将字符常量与字符串常量混淆。'a' 是字符常量，"a" 是字符串常量，二者不同。

C 语言规定：在每个字符串的结尾加一个“字符串结束标志”，以便系统据此判断字符串是否结束。字符串结束标志为“\0”。“\0”是 ASCII 码为 0 的字符，从 ASCII 码表中可以看到，ASCII 码为 0 的字符是“空操作字符”，即不引起任何操作。

5. 变量的初始化

变量的初始化就是在定义变量的同时给变量赋予初值。对变量进行初始化可以采用先说明变量的类型，然后再赋值的方法，也可以采用在对变量类型说明的同时给变量赋初值的方法。

(1) 先定义后赋值：

```
int a,b,c; a=2; b=5; c=10;
```

(2) 定义和赋值同时进行：

```
int a=5; short b=10; char c='a'; float d=7.8;
```

(3) 对几个变量同时赋一个初值：

```
int a1=10, a2=10, a3=10;
```

也可以写成：

```
int   a1, a2, a3; a1=a2=a3=10;
```

但不可以写成：

```
int a1=a2=a3=10;
```

初始化不是在编译阶段完成的，而是在程序运行时执行本函数时完成的，相当于一个赋值语句。例如，int a=10; 相当于 int a; a=10;，又如 int a,b,c=20; 相当于 int a,b,c; c=20;。

任务 1-2　摇奖号码的控制

一、任务导读

程序的执行原则是一条条顺序往下执行。但有时要根据不同的情况做出不同的处理动作，这就需要引入一种重要的结构——选择结构。另外，我们也可以通过 C 语言丰富的运算符对结果做出各种控制。

本任务涉及 if 语句、switch 语句的基本用法以及常见运算符的使用。

二、案例讲解

【例 1-4】 输出两个变量中值较大的那个数。

参考源代码如下：

```
1.  #include "stdio.h"
2.  main()
3.  {
4.      int a=-10,b=6;
5.      int max;
6.      if(a>b) max=a;
7.      else    max=b;
8.      printf("max= %d",max);
9.      getche();
10. }
```

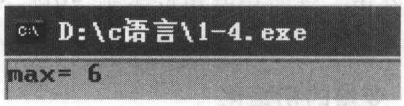


图 1-4　例 1-4 程序运行结果

程序运行结果如图 1-4 所示。

说明：

(1) 本例主要引入了 if 语句来实现分支流程控制，即实现选择结构。

(2) 第 4 行在定义变量的同时给变量赋予初值，int 类型的变量在内存中占 2 个字节，取值范围是 –32 768～+32 767，所以可以把负数赋给它。变量大小的比较规则和数学上的规则意义相同，即所有正数都比负数大。

(3) 第 6 行的 if 语句中，“>”符号是一个关系运算符，“a>b”构成一个关系表达式，当 a 的值大于 b 的值时其返回 1(真或成立)，否则返回 0(假或不成立)。

(4) 本例还定义了一个临时变量 max，用于存放变量 a、b 中值较大的那个。随着程序的执行，max 最终的值是 a、b 中值较大的那个。

(5) 第 6、7 行表明了 if 语句的用法，其格式为：

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

程序执行过程为首先判断条件表达式的值，若为非零，则执行语句 1；否则执行语句 2。总之语句 1 和语句 2 中必有一条且只有一条会被执行。C 语言的 if 语句有三种形式：单分支选择 if 语句、双分支选择 if 语句和多分支选择 if 语句。除了 if 语句，switch 语句也可用于分支流程控制，具体用法会在本书后面的部分讲到。例 1-4 使用了双分支选择 if 语句。

单分支选择 if 语句结构如下：

```
if(条件表达式)          语句 1;
```

多分支选择 if 语句结构如下：

```
if(条件表达式 1)        语句 1;
else if(条件表达式 2)   语句 2;
else if(条件表达式 3)   语句 3;
…
else                    语句 N;
```

多分支选择 if 语句结构会对条件逐个测试，直到满足。而各条语句之间是互斥关系，即语句 1～N 中有且仅有一条语句会被执行。

【例 1-5】 对 3 个分数判断等级。分数大于等于 90 分为 A，分数位于 90 分到 70 分之间为 B，低于 70 分为 C。

参考源代码如下：

```
#include "stdio.h"
main()
{
    unsigned int score1=96,score2=78,score3=62;
    char level;
    switch(score1/10)
    {
        case 10:;
        case 9:level='A';break;
        case 8:
        case 7:level='B';break;
        default:level='C';
    }
    printf("\n%d- Level: %c",score1,level);

    switch(score2/10)
    {
        case 10:;
        case 9:level='A';break;
        case 8:
        case 7:level='B';break;
```