The C Programming Language

！！仅供学习使用，严禁商业用途转载；内容如果有错，欢迎提出，在此感谢。

第1章 导言

在本书的开篇，我们首先概要地介绍C语言，主要是通过实际的程序引入C语言的基本元素，至于其中的具体细节、规则以及一些例外情况，在此暂时不多做讨论。因此，本章不准备完整、详细地讨论C语言中的一些技术（当然，这里所举的所有例子都是正确的）。我们是希望读者能尽快地编写出有用的程序，为此，本章将重点介绍一些基本概念，比如变量与常量、算术运算、控制流、函数、基本输入/输出等。而对于编写较大型程序所涉及到的一些重要特性，比如指针、结构、C语言中十分丰富的运算符集合、部分控制流语句以及标准库等，本章将暂不做讨论。

这种讲解方式也有缺点。应当提请注意的是，在本章的内容中无法找到任何特定语言特性的完整说明，并且，由于比较简略，可能会使读者产生一些误解；再者，由于所举的例子并没有用到C语言的所有强大功能，因此，这些例子也许并不简洁、精炼。虽然我们已经尽力将这些问题的影响降到最低，但问题肯定还是存在。另一个不足之处在于，本章所讲的某些内容在后续相关章节还必须再次讲述。我们希望这种重复给读者带来的帮助效果远远超过它的负面影响。

无论是利还是弊，一个经验丰富的程序员应该可以从本章介绍的内容中推知他们自己进行程序设计所需要的一些基本元素。初学者应编写一些类似的小程序作为本章内容的补充练习。无论是经验丰富的程序员还是初学者，都可以把本章作为后续各章详细讲解的内容的框架。

1.1. 入门

学习一门新程序设计语言的惟一途径就是使用它编写程序。对于所有语言的初学者来说，编写的第一个程序几乎都是相同的，即：

请打印出下列内容

```
hello, world
```

尽管这个练习很简单，但对于初学语言的人来说，它仍然可能成为一大障碍，因为要实现这个目的，我们首先必须编写程序文本，然后成功地运行编译，并加载、运行，最后输出到某个地方。掌握了这些操作细节以后，其它事情就比较容易了。

在C语言中，我们可以用下列程序打印出“hello, world”：

```
#include <stdio.h>

main()
{
   printf("hello, world\n");
}
```

如何运行这个程序取决于所使用的系统。这里举一个特殊的例子。在UNIX操作系统中，首先必须在某个文件中建立这个源程序，并以“.c”作为文件的扩展名，例如hello.c，然 后再通过下列命令进行编译：

```
cc hello.c
```

如果源程序没有什么错误（例如漏掉字符或拼错字符），编译过程将顺利进行，并生成一个可执行文件a.out。然后，我们输入：

```
a.out
```

即可运行a.out，打印出下列信息：

```
hello,world
```

在其它操作系统中，编译、加载、运行等规则会有所不同。

```
#include <stdio.h>
main()
{
   printf("hello, world\n");
}
```

包含标准库的信息
定义名为main的函数，它不接受参数值
main函数的语句都被括在花括号中
main函数调用库函数printf以显示字符序列；
\n代表换行符

第一个 C 语言程序

下面对程序本身做些说明。一个C语言程序，无论其大小如何，都是由函数和变量组成的。函数中包含一些语句，以指定所要执行的计算操作；变量则用于存储计算过程中使用的值。C语言中的函数类似于Fortran语言中的子程序和函数，与Pascal语言中的过程和函数也很类似。在本例中，函数的名字为main。通常情况下，函数的命名没有限制，但main是一个特殊的函数名——每个程序都从main函数的起点开始执行，这意味着每个程序都必须在某个位置包含一个main函数。

main函数通常会调用其它函数来帮助完成某些工作，被调用的函数可以是程序设计人员自己编写的，也可以来自于函数库。上述程序段中的第一行语句

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

用于告诉编译器在本程序中包含标准输入 / 输出库的信息。许多C语言源程序的开始处都包含这一行语句。我们将在第7章和附录B中对标准库进行详细介绍。

函数之间进行数据交换的一种方法是调用函数向被调用函数提供一个值（称为参数）列表。函数名后面的一对圆括号将参数列表括起来。在本例中，main函数不需要任何参数，因此用空参数表()表示。

函数中的语句用一对花括号{}括起来。本例中的main函数仅包含下面一条语句：

```
    printf("hello, world\n");
```

调用函数时，只需要使用函数名加上用圆括号括起来的参数表即可。上面这条语句将"hello, world\n"。作为参数调用printf函数。printf是一个用于打印输出的库函数，在此处，它打印双引号中间的字符串。

用双引号括起来的字符序列称为字符串或字符串常量，如"hello, world\n"就是一个字符串。目前我们仅使用字符串作为printf及其它函数的参数。

在C语言中，字符序列\n表示换行符，在打印中遇到它时，输出打印将换行，从下一行的左端行首开始。如果去掉字符串中的\n（这是个值得一做的练习），即使输出打印完成后也不会换行。在printf函数的参数中，只能用\n表示换行符。如果用程序的换行代替\n，例 如：

```
    printf("hello, world
    ");
```

C编译器将会产生一条错误信息。

printf函数永远不会自动换行，这样我们可以多次调用该函数以分阶段得到一个长的输出行。上面给出的第一个程序也可以改写成下列形式：

```
    #include <stdio.h>

    main()
    {
       printf("hello, ");
       printf("world");
       printf("\n");
    }
```

这段程序与前面的程序的输出相同。

请注意，\n只代表一个字符。类似于\n的转义字符序列为表示无法输入的字符或不可见字符提供了一种通用的可扩充的机制。除此之外，C语言提供的转义字符序列还包括：\t表示制表符；\b表示回退符；"表示双引号；\表示反斜杠符本身。2.3节将给出转义字符序列的完整列表。

练习1-1 在你自己的系统中运行“hello, world”程序。再有意去掉程序中的部分内容，看看会得到什么出错信息。

练习1-2 做个实验，当printf函数的参数字符串中包含\c（其中c是上面的转义字符序列中未曾列出的某一个字符）时，观察一下会出现什么情况。

1.2. 变量与算术表达式

我们来看下一个程序，使用公式°C=(5/9)(°F-32)打印下列华氏温度与摄氏温度对照表：

```
    1               -17
    20              -6
    40              4
    60              15
    80              26
```

```
100             37
120             48
140             60
160             71
180             82
200             93
220             104
240             115
260             126
280             137
300             148
```

此程序中仍然只包括一个名为main的函数定义。它比前面打印“hello, world”的程序长一些，但并不复杂。这个程序中引入了一些新的概念，包括注释、声明、变量、算术表达 式、循环以及格式化输出。该程序如下所示：

```c
#include <stdio.h>

/* 当fahr=0, 20, ..., 300时，分别
  打印华氏温度与摄氏温度对照表 */
main()
{
   int fahr, celsius;
   int lower, upper, step;

   lower = 0;      /* 温度表的下限 */
   upper = 300;    /* 温度表的上限 */
   step = 20;      /* 步长 */

   fahr = lower;
   while (fahr <= upper) {
       celsius = 5 * (fahr-32) / 9;
       printf("%d\t%d\n", fahr, celsius);
       fahr = fahr + step;
  }
}
```

其中的两行：

```c
/* 当fahr=0, 20, ..., 300时，分别
  打印华氏温度与摄氏温度对照表 */
```

称为注释，此处，它简单地解释，该程序是做什么用的。包含在/与/之间的字符序列将被编译器忽略。注释可以自由地运用在程序中，使得程序更易于理解。程序中允许出现空格、制表符或换行符之处，都可以使用注释。

在C语言中，所有变量都必须先声明后使用。声明通常放在函数起始处，在任何可执行语句之前。声明用于说明变量的属性，它由一个类型名和一个变量表组成，例如：

```
int fahr, celsius;
int lower, upper, step;
```

其中，类型int表示其后所列变量为整数，与之相对应的，float表示所列变量为浮点数（即，可以带有小数部分的数）。int与float类型的取值范围取决于具体的机器。对于int类型，通常为16位，其取值范围在-32768～32767之间，也有用32位表示的int类型。float类型通常是32位，它至少有6位有效数字，取值范围一般在10-38～1038之间。

除int与float类型之外，C语言还提供了其它一些基本数据类型，例如：

```
char            字符——一个字节
short           短整型
long            长整型
double                  双精度浮点型
```

这些数据类型对象的大小也取决于具体的机器。另外，还存在这些基本数据类型的数组、结构、联合，指向这些类型的指针以及返回这些类型值的函教。我们将在后续相应的章节中分别介绍。

在上面的温度转换程序中，最开始执行的计算是下列4个赋值语句：

```
lower = 0;
upper = 300;
step = 20;
fahr = lower;
```

它们为变量设置初值。各条语句均以分号结束。

温度转换表中的各行计算方式相同，因此可以用循环语句重复输出各行。这是while循环语句的用途：

```
while (fahr <= upper) {
   ...
}
```

while循环语句的执行方式是这样的：首先测试圆括号中的条件；如果条件为真(fahr<=upper)，则执行循环体（括在花括号中的3条语句）；然后再重新测试圆括号中的条件，如果为真，则再次执行循环体；当圆括号中的条件测试结果为假(fahr>upper)时，循环结束，并继续执行跟在while循环语句之后的下一条语句。在本程序中，循环语句后没有其它语句，因此整个程序的执行终止。

while语句的循环体可以是用花括号括起来的一条或多条语句（如上面的温度转换程序），也可以是不用花括号包括的单条语句，例如：

```
while (i < j)
   i = 2 * i;
```

在这两种情况下，我们总是把由while控制的语句缩进一个制表位，这样就可以很容易地看出循环语句中包含哪些语句。这种缩进方式突出了程序的逻辑结构。尽管C编译器并不关心程序的外观形式，但正确的缩进以及保留适当空格的程序设计风格对程序的易读性非常重要。我们建议每行只书写一条语句，并在运算符两边各加上一个空格字符，这样可以使得运算的结合关系更清楚明了。相比而言，花括号的位置就不那么重要了。我们从比较流行的一些风格中选择了一种，读者可以选择适合自己的一种风格，并养成一直使用这种风格的好习惯。

在该程序中，绝大部分工作都是在循环体中完成的。循环体中的赋值语句

```
    celsius =  5 * (fahr - 32) / 9;
```

用于计算与指定华氏温度相对应的摄氏温度值，并将结果赋值给变量celsius。在该语句中，之所以把表达式写成先乘5然后再除以9而不是直接写成5 / 9，其原因是在C语言及许多其它语言中，整数除法操作将执行舍位，结果中的任何小数部分都会被舍弃。由于5和9都是整数，5 / 9相除后经截取所得的结果为0，因此这样求得的所有摄氏温度都将为0。

从该例子中也可以看出printf函数的一些功能。printf是一个通用输出格式化函数，第7章将对此做详细介绍。该函数的第一个参数是待打印的字符串，其中的每个百分号（%）表示其它的参数（第二个、第三个、.....参数）之一进行替换的位置，并指定打印格式。例如，%d指定一个整型参数，因此语句

```
    printf(" %d\t%d\n", fahr, celsius);
```

用于打印两个整数fahr与celsius的值，并在两者之间留一个制表符的空间（\t）。

printf函数的第一个参数中的各个%分别对应于第二个、第三个、.....参数，它们在数目和类型上都必须匹配，否则将出现错误的结果。

顺便指出，printf函数并不是C语言本身的一部分，C语言本身并没有定义输入 / 输出 功能。printf仅仅是标准库函数中一个有用的函数而已，这些标准序函数在C语言程序中通常都可以使用。但是，ANSI标准定义了printf函数的行为，因此，对每个符合该标准的编译器和库来说，该函数的属性都是相同的。

为了将重点放到讲述C语言本身上，我们在第7章之前的各章中将不再对输入 / 输出做更多的介绍，并且，特别将格式化输入推后到第7章讲解。如果读者想了解数据输入，可以先阅读7.4节中对scanf函数的讨论部分，scanf函数类似于printf函数，但它用于读输入数据而不是写输出数据。

上述的温度转换程序存在两个问题。比较简单的问题是，由于输出的数不是右对齐的，所以输出的结果不是很美观。这个问题比较容易解决：如果在printf语句的第一个参数的%d中指明打印宽度，则打印的数字会在打印区域内右对齐。例如，可以用语句

```
    printf(" %3d %6d\n", fahr, celsius);
```

打印fahr与celsius的值，这样，fahr的值占3个数字宽，celsius的值占6个数字宽，输出的结果如下所示：

```
  0             -17
 20              -6
 40               4
 60              15
 80              26
100      37
...
```

另一个较为严重的问题是，由于我们使用的是整型算术运算，因此经计算得到的摄氏温度值不太精确，例如，与0°F对应的精确的摄氏温度应该为-17.8°C，而不是-17°C。为了得到更精确的结果，应该用浮点算术运算代替上面的整型算术运算。这就需要对程序做适当修改。下面是该程序的又一种版本

```
#include <stdio.h>

/* print Fahrenheit-Celsius table
   for fahr = 0, 20, ..., 300; floating-point version */
main()
{
  float fahr, celsius;
  float lower, upper, step;

  lower = 0;      /* lower limit of temperatuire scale */
  upper = 300;    /* upper limit */
  step = 20;      /* step size */

  fahr = lower;
  while (fahr <= upper) {
      celsius = (5.0/9.0) * (fahr-32.0);
      printf("%3.0f %6.1f\n", fahr, celsius);
      fahr = fahr + step;
  }
}
```

这个程序与前一个程序基本相同，不同的是，它把fahr与celsius声明为float类型，转换公式的表述方式也更自然一些。在前一个程序中，之所以不能使用5 / 9的形式，是因为按整型除法的计算规则，它们相除并舍位后得到的结果为0。但是，常数中的小数点表 明该常数是一个浮点数，因此，5.0 / 9.0是两个浮点数相除，结果将不被舍位。

如果某个算术运算符的所有操作数均为整型，则执行整型运算。但是，如果某个算术运算符有一个浮点型操作数和一个整型操作数，则在开始运算之前整型操作数将会被转换为浮点型。例如，在表达式fahr – 32中，32在运算过程中将被自动转换为浮点数再参与运算。不过，即使浮点常量取的是整型值，在书写时最好还是为它加上一个显式的小数点，这样可以强调其浮点性质，便于阅读。

第2章将详细介绍把整型数转换为浮点型数的规则。在这里需要注意，赋值语句

```
fahr = lower;
```

与条件测试语句

```
while (fahr <= upper)
```

也都是按照这种方式执行的，即在运算之前先把int类型的操作数转换为float类型的操作数。

printf中的转换说明%3.0f表明待打印的浮点数（即fahr）至少占3个字符宽，且不带小数点和小数部分；

%6.1f表明另一个待打印的数（celsius）至少占6个字符宽，且小数点后面有1位数字。其输出如下所示：

```
  0      -17.8
 20       -6.7
 40        4.4
...
```

格式说明可以省略宽度与精度，例如，%6f表示待打印的浮点数至少有6个字符宽；%.2f指定待打印的浮点数的小数点后有两位小数，但宽度没有限制；%f则仅仅要求按照浮点数打印该数。

```
%d                  按照十进制整型数打印
%6d                 按照十进制整型数打印，至少6个字符宽
%f                  按照浮点数打印%6f 按照浮点数打印，至少6个字符宽
%.2f                按照浮点数打印，小数点后有两位小数
%6.2f               按照浮点数打印，至少6个字符宽，小数点后有两位小数
```

此外，printf函数还支持下列格式说明：%o表示八进制数；%x表示十六进制数；%c表示字符；%s表示字符串；%%表示百分号（%）本身。

练习1-3 修改温度转换程序，使之能在转换表的顶部打印一个标题。

练习1-4 编写一个程序打印摄氏温度转换为相应华氏温度的转换表。