第 1 章　　绪 言

随着计算机硬件技术和软件水平的不断提高，作为人机交流主要工具的计算机程序设计语言也经历了从简单到复杂、从低级到高级的发展过程。在诸多的计算机高级语言中，C 语言是目前国内外最为流行的计算机高级程序设计语言之一，它设计精巧、功能齐全、使用灵活，既适合于编写应用软件，又特别适合于编写系统软件。

C 语言最初是为描述和实现 UNIX 操作系统而设计和实现的。在此以前像 UNIX 操作系统那样的系统软件，一般都是利用汇编语言那种低级语言来编写的，自 C 语言开发成功以来，使得利用高级语言来编写系统软件成为可能。UNIX 操作系统源代码的 90%以上是由 C 语言编写的，UNIX 操作系统的一些主要特点，如便于理解、易于修改及具有良好的可移植性等，在一定程度上都受益于 C 语言，所以，UNIX 操作系统的成功与 C 语言是密不可分的。最初的 C 语言是依赖于 UNIX 操作系统环境的，随着 C 语言的不断发展以及应用的普及，目前，C 语言已经能够在多种操作系统如 UNIX、DOS 等环境下运行，而且，实用的 C 语言编译系统种类繁多，适用于 IBM PC 机运行的就有 Microsoft C（MSC）、Turbo C（TC）、Borland C（BC）和 Lattice C（LC）等。

1.1　C 语言简介

1.1.1　C 语言发展简史

在 20 世纪 60 年代，随着计算机科学体系的形成与完善，高级程序设计语言的研究得到了长足的发展，但是，在当时出现的高级语言中，缺乏用于编写像操作系统编译程序等系统软件的工具，系统程序的设计主要还是依赖于汇编语言。为了改变这种状况，1967 年 Martin Richards 设计并实现了 BCPL（Basic Combined Programming Language）语言，后来，这一语言被移植到了多种计算机上，并得到了广泛的应用。此后不久，Ken Thompson 在 BCPL 语言的基础上设计并实现了 B 语言，并用 B 语言在 PDP－7 机上实现了第一个 UNIX 操作系统。接着，在 1972 年至 1973 年间，D. M. Ritchie 在 B 语言的基础上，又重新设计了一种语言，并在 PDP－11 机上实现，同时用这种语言重写了 UNIX 操作系统。由于这一语言是在 BCPL 语言和 B 语言的基础上开发出来的，因此被称为 C 语言。由于 BCPL 语言和 B 语言是无类型的语言，而 C 语言却能支持多种数据类型，因此 C 语言与 BCPL 语言和 B 语言是不同的，它更能反映当代计算机的体系结构，因而得到了广泛的应用。

1.1.2　C 语言的特点

C 语言能够成为目前应用最为广泛的高级程序设计语言之一，完全是由其语言特点决定的。C 语言的特点可大致归纳如下：

（1）C 语言短小精悍，基本组成部分紧凑、简洁。

C 语言一共只有 32 个标准的关键字、45 个标准的运算符以及 9 种控制语句，不但语言的

组成精练、简洁,而且使用方便、灵活。

(2) C语言运算符丰富，表达能力强。

C 语言能够处理多种运算符，其运算符包含的内容广泛，所生成的表达式简练、灵活，有利于提高编译效率和目标代码的质量。

(3) C 语言数据结构丰富，结构化好。

C 语言提供了编写结构化程序所需要的各种数据结构和控制结构，这些丰富的数据结构和控制结构以及以函数调用为主的程序设计风格，保证了利用 C 语言所编写的程序能够具有良好的结构化,同时,在 C 语言程序设计中，允许将一个复杂的程序分割为多个模块，并可由多人同时编写，当分别调试完成后，再通过连接程序连接到一起，形成一个完整的程序。

(4) C 语言提供了某些接近汇编语言的功能，有利于编写系统软件。

C 语言具有‘ 高级语言 ’和‘ 低级语言 ’的双重特点,它提供的一些运算和操作,能够实现汇编语言的一些功能，如 C 语言可以直接访问物理地址，并能进行二进制位运算等，这为编写系统软件提供了方便条件。

(5) 由 C 语言程序生成的目标代码质量高。

C语言程序所生成的目标代码的效率仅比用汇编语言描述同一个问题低 20%左右,因此,用 C 语言编写的程序执行效率高。

(6) C语言程序可移植性好。

在 C语言所提供的语句中，没有直接依赖于硬件的语句，与硬件有关的操作，如数据的输入、输出等都是通过调用系统提供的库函数来实现的，而这些库函数本身并不是 C 语言的组成部分。因此,用 C语言编写的程序能够很容易地从一种计算机环境移植到另一种计算机环境中。

当然,C 语言本身也有其弱点：一是运算符的优先级和结合性比较复杂，不容易记忆；二是由于 C语言的语法限制不太严格，这在增强了程序设计的灵活性的同时，在一定程度上也降低了某些安全性，这对程序设计人员提出了更高的要求。

总之,由于 C 语言的上述特点，使得 C 语言越来越受到程序设计人员的重视，并且已经在广泛的领域里得到了应用。

1.2 C语言程序的开发步骤

开发一个 C 语言程序的基本步骤如图 1.1 所示。

1. 编辑

选择适当的编辑程序，将 C 语言源程序通过键盘输入到计算机中，并以文件的形式存入到磁盘中。在 DOS 系统下，经过编辑后得到的源程序文件都是以 .C 为其文件扩展名。

2. 编译

通过编辑程序将源程序输入到计算机后，需要经过 C 语言编译成目标程序。在对源程序的编译过程中，可能会发现程序中的一些语法错误，这时就需要重新利用编辑程序来修改源程序，然后再重新编译。在 DOS 系统下，经过编译后得到的目标文件都是以 .OBJ 为其文件扩展名。

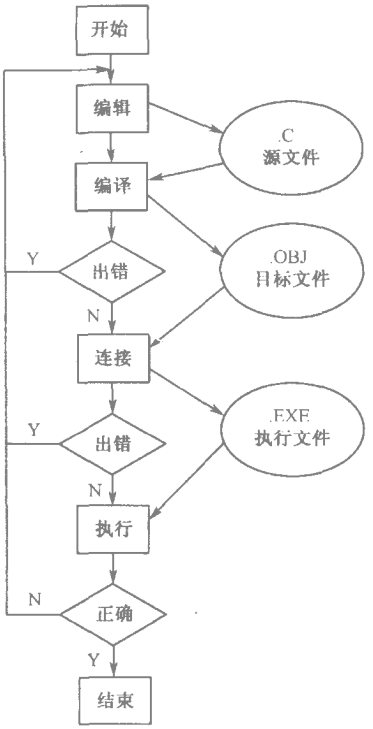

图 1.1 C 语言程序开发步骤

3. 连 接

经过编译后生成的目标文件是不能直接执行的，它需要经过连接之后才能生成可执行的代码。在 DOS 系统下，连接后所得到的可执行文件都是以 .EXE 为其文件扩展名。

4. 执行

经过编译、连接之后，源程序文件就可以生成可执行的文件，这时就可以执行了。在 DOS 系统下,只要键入可执行的文件名,并按‘ 回车 ’键后,就可执行文件了。

现在有许多厂家都推出一种集成环境来处理 C 语言程序,如 Turbo C ，在这种集成环境下，对程序的编辑、编译和连接等操作，都可以在一个窗口下进行，使用起来非常方便。

在本书附录 C 中简单介绍了 Turbo C 集成开发环境的使用方法，感兴趣的读者可以参考使用。

1.3 C语言的程序结构

C 语言的程序结构比较简单，很容易掌握，它主要是通过函数之间的调用来实现指定的功能。在这一节中，将通过编写几个简单的 C 语言程序，来阐述 C 语言的程序结构，同时也对 C 语言的基本语法成分进行相应的说明，以便读者对 C 语言程序有一个概括的了解，为以后的学习打下基础。

【例 1.1】 编写一个 C 语言程序，用于显示字符串“ Hello,World!”。

```
# include "stdio.h"
```

```
main( )
{
        printf("Hello,World! \n");
}
```

上述程序是一个简单而完整的 C 语言程序，经过编辑、编译和连接后，其执行结果是在屏幕的当前光标位置处显示如下字符串：

```
Hello,World!
```

下面对上述程序进行一下说明：

（1）一个 C 语言程序可以由多个函数组成，但任何一个完整的 C 语言程序，都必须包含一个且只能包含一个名为 main 的函数，程序总是从 main 函数开始执行的。

（2）由左右花括号括起来的部分是函数体，函数体中的语句将实现程序的预定功能。在本例中，main 函数的函数体中只有 printf 一个语句，它的功能是进行格式化输出（显示），即将字符串“Hello,World! \n”显示在终端屏幕上。其中，字符串中的字符‘\’和‘n’合起来，表示一个“换行”字符，在‘换行’字符后面输出的任何字符，将被显示在屏幕的下一行上。

（3）C 语言中的每个基本语句，都是以‘;’结束的，分号是 C 语言语句的终结符。

（4）C 语言程序的书写格式比较自由，没有固定的格式要求，在一行内，既可以写一个语句，也可以写多个语句。为了提高程序的可读性，往往根据语句的从属关系，以缩进书写的形式来体现出语句的层次性。

（5）#include 命令是编译预处理命令，其作用是将由双引号或尖括号括起来的文件中的内容，读入到该语句的位置处。在使用 C 语言输入、输出库函数时，一般需要使用 #include 命令将"stdio.h"文件包含到源文件中。有关 # include 命令的作用及其使用方法，将在第 7 章中作详细介绍。

【例 1.2】 从键盘输入 2 个整数，并将这 2 个整数之和显示出来。

```
#include "stdio.h"
int ADDxy(a,b)          /*计算 2 个整数之和*/
    int a,b;
{
        int c;
        c=a+b;
        return(c);
}
main ( )
{
        int x,y,z;
        scanf("%d%d",&x,&y);      /*读入 2 个整数,存入变量 x 和 y 中*/
        z=ADDxy(x,y);
        printf("The sum of %d and %d is %d",x,y,z);
}
```

上述程序经过编辑、编译和连接后，执行情况说明如下：

当此程序从 main 函数开始执行，执行到 scanf 语句时，将等待用户从键盘输入 2 个整型数据后再继续执行，假如用户输入 10 和 20（此时，x 的值为 10，y 的值为 20），则屏幕将显示如下信息：

```
The sum of 10 and 20 is 30
```

下面，对上述程序进行一下说明：

（1）程序中由／＊和＊／括起来的内容是程序的注释部分，它是为了增加程序的可读性而设置的。注释部分对程序的编译过程和执行结果没有任何影响。

（2）C 语言中的所有变量都必须定义为某种数据类型，同时必须遵循“先定义、后使用”的原则，如语句：

```
int x,y,z;
```

定义了 x、y、z 是 3 个整型变量，以后就可以使用这 3 个变量来存放整型数据。

（3）一个 C 语言程序可以由多个函数组成，通过函数之间的调用来实现相应的功能。程序中所使用的函数，既可以是系统提供的库函数，也可以是用户根据需要自己定义的函数。如上述 main 函数中调用的 scanf 函数和 printf 函数，就是系统提供的库函数，这些函数不需要用户自己定义，在需要时，只要按照指定的格式进行调用即可。C 语言编译系统提供的库函数非常丰富，特别是与硬件打交道的部分，很多工作只要调用库函数就可以实现，而函数 ADDxy 是自定义的函数，它是用户为了实现加法功能而自己编写的函数。每一个函数都用于完成一特定的功能，ADDxy 函数的功能是将其参数 a、b 之和通过 return 语句返回给 main 函数中的变量 z。正确地使用函数，将有助于编写易于理解的、结构化好的程序。有关函数的详细说明请参阅第 6 章。

（4）程序中调用的 scanf 函数的作用是进行格式化输入，其中由圆括号括起来的部分是函数的参数部分，不同的函数需要不同的参数，scanf 函数中的参数主要包括两部分内容：一是“格式控制”部分，它用于对输入数据的格式进行说明；二是“地址表”部分（本书中出现的表的概念，如地址表、输出表等，是指用逗号分隔的有限个元素序列），它使用的是存放输入数据的变量的地址。

程序中调用的 printf 函数的作用是进行格式化输出，其参数也包括两部分内容：一是“格式控制”部分，用于对输出数据的格式进行说明；二是“输出表”部分，它使用的是存放输出数据的变量本身。有关数据的输入、输出以及函数的调用形式，将在以后作详细的介绍。

习题

1.1 试简述 C 语言的主要特点及其用途。

1.2 请参照本章例题，编写一个 C 语言程序，用于显示以下信息：

```
Happy New Year!
```

1.3 请参照本章例题，分析下面的 C 语言程序，并给出运行结果。

```
#include "stdio.h"
main( )
{
        int a,b,c;
        a=100;
        b=20;
        c=a-b;
        printf("sum=%d",c);
}
```

1.4 请参照本章例题，编写一个 C 语言程序，用于显示以下信息：

```
# # # # # # # # # # # # # # # # # # # # # # # # #
Wonderful!
# # # # # # # # # # # # # # # # # # # # # # # # #
```

1.5 **请参照本章例题，分析下面的 C 语言程序，并给出模拟运行结果。**

```
#include "stdio.h"
main( )
{
        int x,y,z;
        scanf("%d%d",&x,&y);
        z=x/y;
        printf("Result=%d",z);
```

第 2 章　数据、运算符和表达式

常量、变量及函数等都是程序的基本操作对象，统称为数据。根据数据的取值范围以及能在其上所进行的运算，可把数据分为各种类型，不同类型的数据一般在内存中占用不同的存储空间，同时，数据的类型不同，能够参加的运算也不同。C 语言中的数据类型非常丰富，大体上可划分为基本的数据类型和导出的数据类型两种。基本数据类型主要包括整型，字符型和单、双精度浮点型等；导出数据类型是在基本数据类型的基础上产生的，其中包括数组、结构等。

本章主要讨论 C 语言中的一些基本概念，如变量、标识符等，同时详细说明 C 语言中的几种基本数据类型、算术运算符、关系运算符、逻辑运算符以及利用这些运算符来构成相应表达式的一些规则。

2.1　基本概念

2.1.1　标识符

在计算机语言中，标识符的概念经常被用到。所谓标识符，是指用来标识程序中用到的变量名、函数名、类型名、数组名、文件名及符号常量名等的有效字符序列。

在 C 语言中，标识符的命名规则是：由字母（大、小写皆可）、数字及下划线组成，且第一个字符必须是字母或下划线。

由上述标识符的命名规则可知，下面的标识符名是合法的：

year，Day，ATOK，x1._CWS._change_to

而下面的标识符名是不合法的：

#123..COM.，$100，1996Y，1-2-3

在 C 语言中，大写字母和小写字母是有区别的，即作为不同的字母来看待。如标识符 RAN、Ran 和 ran 分别表示三个不同的标识符，这一点同其他高级语言是有区别的，应引起注意。

2.1.2　常量

常量又称常数，是指在程序运行过程中其值不能被改变的量，例如，100，3.14 等。常量也分为不同的类型，这是由常量本身隐含决定的（将在下面详细介绍）。为了增加程序的可读性，可以用一个名字（字符序列）来代表一个常量，此时的常量被称为符号常量。有关符号常量的使用，将在第 7 章中详细介绍。

2.1.3　变量

变量是指在程序运行过程中其值可以被改变的量。变量被区分为不同的类型，不同类型的变量在内存中占用不同的存储单元，以便用来存放相应变量的值。

组成变量名（标识符）的有效字符数随 C 语言的编译系统而定。有的编译系统允许使用长

达 31 个字符的变量名，而有的编译系统只取变量名的前 8 个字符作为有效字符，后面的字符无效,不被识别,这样,只要变量名的前 8 个字符相同，就被认为是同一个变量。因此，在进行程序设计之前，应首先了解所使用的编译系统中对变量名长度的规定，以免造成变量使用上的混乱。

2.1.4 关键字

关键字，又被称为保留字或保留关键字，也是 C 语言中的一种标识符，它用来命名 C 语言程序中的语句、数据类型和变量属性等。每个关键字都有固定的含义，不能另作其他用途，C 语言中的所有关键字都是用小写字母来表示的。

2.2 基本数据类型

在 C 语言中，最基本的数据类型只有四种，它们分别由如下标识符进行定义：

int	整型
char	字符型
float	单精度浮点型
double	双精度浮点型

C 语言规定，对程序中用到的所有变量，都必须先定义后使用，每个变量只能与一种数据类型相联系。在定义变量时，不能把 C 语言中具有固定含义的关键字（如 int、char 等)作为变量名，同时，同一个函数内所定义的变量不能同名。

2.2.1 整型变量及其常量

1. 整型变量

整型变量可用来存放整型数据（即不带小数点的数）。其定义方式如下：

```
int i1,i2;
```

其中,i1 和 i2 即被定义为整型变量。

2. 整型常量

在 C 语言中，整型常量可以用三种数制来表示：

(1) 十进制整型常量：如 250,－12 等，其每个数字位可以是 0～9。

(2) 十六进制整型常量：如果整型常量以 0x 或 0X 开头，那么这就是用十六进制形式表示的整型常量。例如，十进制数的 128，用十六进制表示为 0x80 或 0X80，其每个数字位可以是 0～9、A～F。

(3) 八进制整型常量：如果整型常量的最高位为 0，那么它就是以八进制形式表示的整型常量。例如，十进制数的 128,用八进制表示为 0200。需要注意的是，八进制数中的每个数字位必须是 0～7。

2.2.2 浮点型变量及其常量

1. 浮点型变量

在 C 语言中，把带有小数点的数称为浮点数，也可以称为实型数。

浮点型变量又被称为实型变量，按其能够表示的数的精度，又被分为单精度浮点型变量和双精度浮点型变量。

单精度浮点型变量的定义方式如下：

```
float  f1,f2 ;
```

其中，f1和 f2 即被定义为单精度浮点型变量。

双精度浮点型变量的定义方式如下：

```
double  d1,d2 ;
```

其中，d1和 d2即被定义为双精度浮点型变量。

单精度浮点型变量和双精度浮点型变量之间的差异，仅仅体现在所能表示的数据的精度上，如果单精度浮点型所提供的精度不能满足要求时，则可以考虑使用双精度浮点型。在不同的系统中，float 型数据和 double 型数据所能提供的精度是有差异的，一般来讲，在同一个系统中，double 型变量值的最大有效位数，通常是 float 型的两倍。

2. 浮点型常量

浮点型常量一般不分 float 型和 double 型，任何一个浮点型常量，既可以赋给 float 型变量，也可以赋给 double 型变量，但由于 float 型变量和 double 型变量所能表示的数的精度不同，所以，在赋值时，将根据变量的类型来截取相应的有效位数。

浮点型常量有如下两种表示形式。

（1）十进制数形式：它是由数字和小数点组成的，例如， 3.141 59，－7.2，9.8等都是用十进制数的形式表示的浮点数。

（2）指数法形式：指数法又称为科学计数法，它是为方便计算机对浮点数的处理而提出的。例如，十进制数的 180 000.0，用指数法可表示为 1.8e5 ，其中 1.8被称为尾数，5被称为指数，字母 e也可以用 E 来表示。又如，0.001 23可表示为 1.23E－3 。需要注意的是，用指数形式来表示浮点数时，字母 e 或 E 之前（即尾数部分）必须有数字，且 e后面的指数部分必须是整数，例如，e－3，9.8e2.1，e5 等都是不合法的指数表示形式。

2.2.3 字符型变量及其常量

1. 字符型变量

字符型变量用于存放一个单个字符。它的定义方式如下：

```
char  c1,c2 ;
```

其中，c1和 c2即被定义为字符型变量。

2. 字符型常量

字符型常量是由一对单引号括起来的一个字符，例如，'A'，'＊'和'8'等都是合法的字符型常量。

在 C 语言中，还允许使用一些特殊形式的字符型常量，这些字符型常量都是以反斜线字符'\'开头的字符序列（又称为转义字符）。下面介绍几个常用的以'\'字符开头的特殊字符：

'\n' 换行字符

'\r' 回车字符

'\b' 退格字符

'\t' 制表字符，又被称为横向跳格字符

'\'' 单引号字符

'\"' 双引号字符

由上面给出的特殊字符可知，如果要显示单引号或双引号字符，则在字符串中需要用上述特殊字符形式给出。

除了上述具有特殊意义的字符外，C 语言还允许在字符'\'后面紧跟 1～3 位八进制数或在'\x'后面紧跟 1～2 位十六进制数来表示相应系统中所使用的字符的编码值。使用这种表示方法，可以表示字符集中的任一字符，包括某些难以输入和显示的“控制字符”，ASCII 码表中编码值小于 0x20 的字符就属于这一类字符。如响铃字符(bell)，在 ASCII 码表中的编码值为 7，在程序处理过程中，为了发出响铃声音，可通过显示'\7'('\07'或'\007'）码来获得响铃效果。

需要注意的是，上面介绍的由'\'开头的特殊字符，仅代表一个单个字符，而不代表多个字符，它仅代表相应系统中的一个编码值。

3. 字符串常量

前已述及，字符常量是由单引号括起来的单个字符。C 语言除了允许使用字符常量外，还允许使用字符串常量。字符串常量是由一对双引号括起来的字符序列，如"string"就是一个字符串常量，从表面上看，"string"是由 6 个字符组成的，但实际上它是由 7 个字符组成的，这是因为在 C 语言中，系统自动地在每个由双引号括起来的字符串的最后补上'\0'字符，即 ASCII 码值为 0 的字符。因此，C 语言中的每个字符串都是以'\0'为结束标志的。在第 5 章中将详细介绍有关对字符串的操作。

需要注意的是，不能将字符串常量赋给一个字符型变量，如果要保存字符串常量的话，需要使用以后介绍的字符数组来存放。

2.2.4 长整型、短整型和无符号整型

上面已经介绍了四种基本数据类型，它们是 int、char、float 和 double。C 语言在 int 型的基础上又发展了三种基本数据类型，它们是长整型、短整型和无符号整型，分别用 long int、short int 和 unsigned int 来表示。

1. 长整型

长整型变量的定义方式如下：

```
long int lv;
```

其中，lv 是长整型变量，关键字 int 可以省略。

使用长整型变量的目的是为了存放比较大的整数。长整型变量能够表示的数值范围与计算机系统有关，一般是 int 型变量有效位数的两倍，int 型变量如果占 2 个字节，则 long int 型变量可能占 4 个字节，这样，长整型变量所能表示的数值范围，将远远超过一般整型变量所能表示

的范围。

在整型常量的末尾加上字母 L 或 l，就构成了长整型常量。例如，可将 987654 写成 987654L 或 987654l.

2. 短整型

短整型变量的定义方式如下：

```
short int sv ;
```

其中，sv 是短整型变量，关键字 int 可以省略。

短整型变量用于存放比较小的整型数，使用短整型的目的是为了节省计算机的内存空间，但一个短整型变量所占的内存字节数与相应的系统有关，在有的计算机系统中，短整型变量所占用的存储空间少于整型变量所占用的存储空间，但在有的系统中， short int 型变量和 int 型变量占用相同的内存空间。

短整型常量与一般的整型常量没有明显的区别，其区别仅仅在于所能表示的数值大小而已。在 IBM PC 机上，短整型变量和整型变量都占用 2 个字节的内存空间，因此，所能表示的数值大小也完全一样，即 －32 768～32 767 之间。如要将 123 赋给短整型变量 sv，则需要使用如下语句：

```
sv＝123;
```

3. 无符号整型

无符号整型变量的定义方式如下：

```
unsigned int uv;
```

其中，uv 是无符号整型变量，关键字 int 可以省略。

无符号整型变量只用于存放非负整型数，由于这种变量与 int 型变量占用相同的内存空间，因此，它能够存放的正整数范围将大于 int 型变量所能存放的正整数范围。

无符号整型常量总是大于等于 0 的正整数。在 IBM PC 机上，无符号整型常量的数值范围一般为 0～65 535 之间。如要将 100 赋给无符号整型变量 uv，则需要使用如下语句：

```
uv＝100;
```

2.2.5 类型定义 typedef

C 语言提供了许多标准类型名，如上面介绍的 int、char 和 float 等，用户可以直接使用这些类型名来定义所需要的变量。同时， C 语言还允许使用 typedef 语句来定义新的类型名，以取代已有的类型名。例如：

```
typedef  int  counter;
```

其作用是定义 counter 来等价于基本数据类型名 int ，以后，就可以利用 counter 来定义 int 型变量了。例如：

```
counter  i,n;
```

其等价于：

```
int  i,n;
```

使用类型定义的优点是能够增加程序的可读性。由上述语句可以看出，当用 counter 来定义 i、n 变量时，就可以判断出 i、n 变量的作用可能是当计数器使用，但如果用 int 来定义的话，

就难以看出这种用途。

使用 typedef 语句需要注意如下几个问题：

（1）typedef 语句不能创造新的类型，只能为已有的类型增加一个类型名，这也就是说原有的类型名仍然可以使用。

（2）typedef 语句只能用来定义类型名，而不能用来定义变量。

2.3 算术运算符、赋值运算符及其表达式

C 语言的运算符很多，本节主要介绍算术运算符、赋值运算符及其相应的表达式。

2.3.1 算术运算符和算术表达式

1. 加、减、乘、除及取模运算符

C 语言中的加、减、乘、除及取模运算符分别由＋、－、＊、/及％来表示。其中减法运算符也可以作为负值运算符使用，例如，－100(表示负 100) 取模运算符又称为求余运算符，要求％的两侧必须为整型数，它的作用是取两个整型数相除的余数，如 21％7的运算结果是 0，21％5 的运算结果为 1。

这里需要指出的是，一个表达式中如果有多个运算符，则计算是有先后次序的，这种计算的先后次序称为相应运算符的优先级。例如，在上述介绍的运算符中，＊、/、％运算符的优先级高于十、一运算符的优先级。除了优先级以外，C 语言中还规定了相应运算符的结合性。所谓结合性，是指当一个运算对象（操作数）两侧的运算符的优先级相同时，进行运算（处理）的结合方向。在 C 语言中，算术运算符的结合方向为自左至右，即运算对象先与左面的运算符结合，而有的运算符的结合方向是自右至左，如赋值运算符。应该看到，C 语言对运算符的优先级及其结合性的规定是比较复杂的，初学者往往会感到不适应，为了帮助初学者了解每个运算符的优先级和结合性，在本书附录 B 中列出了每个运算符的优先级及其结合性。

由算术运算符和圆括号将运算对象连接起来的有意义的式子称为算术表达式。在表达式中，使用左、右圆括号可以改变运算的处理顺序，由于圆括号的优先级最高，因此，圆括号里的运算总是最先进行的。例如：

```
x=a*(b+c);
```

上式中，先计算括号里 $b+c$ 的值，然后乘以 a，并把结果赋给变量 x。

2. 增 1、减 1 运算符

增 1、减 1 运算符是 C 语言特有的运算效率较高的运算符。

增 1 运算符 ＋＋和减 1 运算符－－是两个单目（只有一个运算对象）运算符，它们的运算对象只能是整型或字符型变量，而不是浮点型变量。增 1 运算符是将运算对象的值增 1，减 1 运算符是将运算对象的值减 1。它们既可以作前缀运算符（位于运算对象的前面），如，＋＋n，－－m；也可以作后缀运算符（位于运算对象的后面），如，n＋＋，m－－。尽管＋＋n 和 n＋＋都使 n 的值自增 1，但其结果是有区别的：＋＋n 表示在用该表达式的值之前使 n 的值增 1；n＋＋表示在用该表达式的值之后使 n 的值增 1。同样，－－n 表示在用该表达式的值之前使 n 的值减 1；n－－表示在用该表达式的值之后使 n 的值减 1。因此，在使用增 1 和减 1 运算符时，必须要注意

变量的值和表达式的值在程序上下文中的效果。

如语句：

```
x=n--;
```

相当于下面两个语句的运算结果：

```
x=n;
n=n-1;
```

而语句：

```
x=--n;
```

相当于下面两个语句的运算结果：

```
n=n-1;
x=n;
```

显然,对于变量 n 来讲，上述两个语句的结果是一样的,都使 n 的值减 1,而对于 x 的值来讲就不同了，这一点必须要引起注意。

另外,使用增 1、减 1 运算符进行运算时，还应当注意如下两个问题：一是增 1、减 1 运算符只适用于变量，而不能用于常量或表达式，如 --5 和 (i+j)++ 等都是非法的；二是在只需对变量本身进行增 1 或减 1 操作而不关心整个表达式的值的情况下，前缀运算和后缀运算的效果完全相同。

合理地使用增 1、减 1 运算符,对于编写高质量的 C 语言程序是非常有用的，它的简洁表示形式，对于以后要介绍的结构控制语句以及指针运算等都将带来很大的方便。

2.3.2 赋值运算符和赋值表达式

C 语言的赋值运算符是“=”，它的作用是将赋值运算符右边的表达式（包括算术表达式、关系表达式和逻辑表达式）的值赋给其左边的变量。如：x=a+b 的作用是将 a+b 的结果赋给变量 x。

为了简化程序并提高编译效率，C 语言允许在赋值运算符“=”之前加上其他运算符,以构成复合的赋值运算符。例如：

```
x=x+5
```

可以写成

```
x+=5
```

又如 x=x*(y+1)

可以写成

```
x*=y+1
```

凡是二元（有两个运算对象）运算符，一般都可以与赋值运算符一起组成复合的赋值运算符。例如：

```
+=, -=, *=, /=, %=
```

等都是合法的复合赋值运算符。

2.4 关系运算符、逻辑运算符及其表达式

2.4.1 关系运算符和关系表达式

关系运算是逻辑运算的一种简单形式，主要用于比较操作。

1. 关系运算符

C 语言中的关系运算符共有六种，它们是：

< 小 于
<= 小于等于
> 大于
>= 大于等于
== 等于
!= 不等于

由附录 B 可知，关系运算符的优先级低于算术运算符的优先级，且上述后两种关系运算符（==和！=）的优先级低于前四种关系运算符的优先级。

这里需要注意的是，不要把关系运算符的“==”和赋值运算符的“=”搞混淆了。“==”是关系运算符，仅用于比较操作，而没有赋值运算，而“=”是赋值运算符，它主要进行赋值操作。

2. 关系表达式

由关系运算符将两个表达式连接起来的有意义的式子称为关系表达式。例如：

```
x>y, a+b<-9.8
```

等都是合法的关系表达式。

关系表达式的值是一个逻辑值，即“真”或“假”。在 C 语言中，用 1 来表示“真”，用 0 来表示“假”。例如，当 x>y 为“真”时，其表达式的值为 1；当 x>y 为“假”时，其表达式的值为 0。

可以将关系表达式的运算结果（0 或 1）赋给一个整型变量或字符型变量。例如，当 $a=4$，$b=1$ 时，下面的赋值语句是将 1 赋给变量 c：

```
c=a>b;
```

2.4.2 逻辑运算符和逻辑表达式

为了表示比较复杂的条件，需要将若干个关系表达式组合起来判断，C 语言提供的逻辑运算就是用于实现这一目的的。

1. 逻辑运算符

C 语言中提供了如下三种逻辑运算符：

&& 逻辑与
|| 逻辑或
！ 逻 辑 非

上述逻辑运算符的运算规则如下。

&& 当两个操作数都为“真”时，运算结果为“真”，其他情况运算结果都为“假”。

|| 只要有一个操作数为“真”，运算结果就为“真”，只有当两个操作数都为“假”时，运算结果才为“假”。

！这是一个单目运算符（只有一个操作数），当操作数为“真”时，运算结果为“假”；当操作数为“假”时，运算结果为“真”。

表 2.1 给出了上述三种逻辑运算的真值表。其中 a、b 是两个操作数。

表2.1 三种逻辑运算真值表

a	b	!a	!b	a‖b	a&&b
真	真	假	假	真	真
真	假	假	真	真	假
假	真	真	假	真	假
假	假	真	真	假	假

由附录 B 可知，!优先级高于 &&，&& 的优先级高于||，的优先级高于算术运算符和关系运算符的优先级，而 && 和 ||的优先级又都低于算术运算符和关系运算符的优先级。

2. 逻辑表达式

用逻辑运算符将关系表达式或逻辑量连接起来的有意义的式子称为逻辑表达式。逻辑表达式的值是一个逻辑值，即“真”或“假”。C 语言编译系统在给出逻辑运算结果时，以数字 1 表示“真”，以数字 0 表示“假”；但在判断一个量是否为“真”时，以非 0 表示“真”，以 0 表示“假”。

可以将逻辑表达式的运算结果（0 或 1）赋给整型变量或字符型变量。例如，当 $a=1, b=2, c=3, d=3$ 时，下面的赋值语句是将 1 赋给变量 e：

```
e=(a<=b)&&(c<=d);
```

由于浮点数在计算机中不能非常准确地表示，所以，判断两个浮点数是否相等时，通常不使用关系运算符“等于”（==），而是利用区间判断方法来实现。例如，为了判断 f 是否等于 9.8，可利用如下逻辑表达式：

```
f>9.7 && f<9.9
```

当此逻辑表达式为“真”时，就可以认为 f 等于 9.8。

关系运算和逻辑运算主要用于以后将要介绍的流程控制语句中。

2.5 变量的初始化

在对变量进行定义的同时为其赋初值称为变量的初始化。对变量所赋的初值可以是常量，也可以是常量表达式（即每个运算分量都是常量的表达式），此时，该常量表达式的值即为变量的初值。例如：

```
int age=20;
```

它表示定义整型变量 age，同时将初值 20 赋给它。同样，下面的初始化方式也是合法的：

```
char letter='A';
int convert='b'-'B';
float average=3.2*9.8;
```

下面的语句将为 x、y、z 三个整型变量都赋以初值 10：

```
int x=10,y=10,z=10;
```

也可以对被同时定义的多个变量的一部分赋初值。例如：

```
float px,py,pi=3.1416;
```

它表示 px、py、pi 都是浮点型变量，但只对变量 pi 赋以初值 3.141 6，而其余变量（px 和 py）没有显式地赋以初值。

2.6 不同类型数据之间的转换

在内存中，字符是以系统中所使用的字符的编码值形式存储的，比如，在 IBM PC 机上是以 ASCII 码形式存储的，它的存储形式与整型数的存储形式相似。C 语言允许字符型数据和整型数据之间通用：一个字符型数据，既可以用字符形式输出，也可以用整数形式输出（有关输入／输出部分将在以后介绍）。同时，字符型数据可以赋给整型变量，整型数据也可以赋给字符型变量，只是当整型数据的大小超过字符型变量的表示范围时，需要截取相应的有效位数。

除了上述字符型数据和整型数据之间可以通用之外，不同类型的数据在进行混合运算时，往往需要进行类型转换。这种类型转换有两种方式：一种是自动类型转换，另一种是强制类型转换。

2.6.1 自动类型转换

C 语言允许整型、单精度浮点型和双精度浮点型数据之间进行混合运算，由于字符型数据可以和整型数据通用，所以，下列表达式是合法的：

```
'B'+3.14*560-92
```

显然，在进行混合运算时，不同类型的数据要首先转换成同一类型，然后才能进行运算，而这种转换最终都归结为整型数据和浮点型数据之间的转换问题。下面，给出整型数据和浮点型数据之间的自动转换规则：

（1）当单、双精度浮点型数据赋给整型变量时，浮点数的小数部分将被舍弃。同时，由于浮点型变量所能表示的数的范围远远超过整型变量的表示范围，所以，在赋值时，还需进行相应有效位数的截取操作。

（2）当整型数据赋给浮点型变量时，在数值上不发生任何变化，但整型数据将被转换成浮点型数据的形式后，存储到浮点型变量中。

（3）如果某个算术运算符的两个运算对象都是整数，那么，运算将按照整型数的运算规则来进行，这就意味着对除法运算来讲，其结果的小数部分将被舍弃。需要注意是，在这种情况下，即使运算结果赋给了浮点型变量也是一样的，结果的小数部分也将被舍弃。例如：

```
float f;
int i=4;

f=10/i;
```

此时，f 的值为 2.0，而不是 2.5。

（4）只要某个算术运算符的两个运算对象中有一个是浮点型数据，则运算将按照浮点数的运算规则来进行，即结果的小数部分也被保留下来。例如：

```
float f
int i=4;

f=10.0/i
```

此时，f 的值为 2.5，而不是 2.0。

2.6.2 强制类型转换

当自动类型转换达不到目的时，可以利用强制类型转换。例如，当除法运算符‘／’的两个运

算对象都是整型数据时，其运算将按照整型运算规则来进行，即舍弃结果的小数部分。如果希望按照浮点型运算规则来运算的话，就必须首先把其中某个运算对象的数据类型强制转换为浮点型，然后再进行运算。

强制类型转换的一般形式如下所示：

(类型名)(表达式)

例如，(int)(x+y)是将 x+y 的结果强制转换成 int 型。又如，(float)x/y 是将 x 强制转换成 float 型后，再进行运算。

需要注意的是，在进行强制类型转换时，需要将类型名用圆括号括起来，同时，经强制类型转换后，得到的是一个所需类型的中间变量，原来变量的类型及其数值大小并没有发生任何变化。

2.7 sizeof 运算符

在程序设计过程中，有时需要了解一个变量或某种类型的量在内存中所占的字节数，sizeof 运算符就是用于这一目的的。

sizeof 运算符有两种用法。

(1) sizeof 表达式

其运算结果是得到表达式计算结果在内存中所占用的字节数。

例如，当 x 是整型变量时，sizeof x 的值是 2，它表示在内存中占 2 个字节。

(2) sizeof (类型名)

其运算结果是得到某种类型的量在内存中所占用的字节数。

例如，sizeof(float)的值是 4，它表示 float 型变量在内存中占 4 个字节。

sizeof 运算符可以出现在表达式中，例如：

```
x=sizeof(float)-2;
printf("%d",sizeof(double));
```

2.8 应用举例

【例 2.1】编一程序，定义 2 个整型变量并为其初始化，同时将其相加之和显示出来。

```
#include "stdio.h"
main()
{    int  x=10,y=100,z;
     z=x+y;
     printf("z=%d",z);
}
```

【例 2.2】从键盘输入 1 个整数，并将其值加 1 后显示出来。

```
#include "stdio.h"
main()
{
     int x;
     scanf("%d",&x);
```