第1章

C语言程序设计基础

本章通过简单而典型的C语言程序实例，引入C语言的主函数、C程序所使用的基本数据结构和表达式，以及实现输出和输入的方法，从而建立C程序设计的基本概念。

1.1 C程序及其主函数

C语言是20世纪70年代初期美国贝尔(Bell)实验室Dennis M. Ritchie设计的一种程序设计语言，在1975年用C语言编写的UNIX操作系统第六版公诸于世之后，C语言才引起广泛重视，并成了最流行的程序设计语言之一。

ANSI(美国国家标准协会)于1983年成立了一个专门委员会，为C语言制定了ANSI标准。当时比较流行的有Turbo C，它不仅满足ANSI标准，还提供了一个集成开发环境，同时也按传统方式提供了命令行编译程序版本以满足不同用户的需要。本书虽然以Microsoft Visual C++ 6.0为编程环境，但程序则严格按照新标准。附录A给出C语言新版本与老版本的主要差别。

用C语言编写的程序称为C语言源程序，简称C程序。C程序一般是由一个或若干个函数组成，在组成一个程序的若干函数中必须有一个且只能有一个名为main的函数(主函数)，运行C程序时总是从main函数开始执行。在一个函数名字之后一定要有一对圆括号，圆括号中是否有参数由编程者决定，目前只介绍无参数的main函数，多个函数的C程序结构将在第3章介绍。C程序所在的文件以后缀“.c”作为文件扩展名。

1.1.1 简单的C程序

C程序至少要有一个主函数，下面是一个简单的标准C程序实例。

【例1.1】 打印字符串。

```
/* 只有主函数的示例程序
   功能:打印字符串 */
#include <stdio.h>                    /* 包含头文件 */
int main()                            /* 主函数 */
{
```

```
    printf("Hello! How are you?");          /* 打印字符串 */
    return 0;                               /* 主函数 main 的返回值 */
}
```

1. 注释语句

这是一个完整的 C 程序，“/*”与“*/”之间的内容是注释，注释在编译时不产生目标代码。所谓目标代码，就是程序可以执行的代码。

例 1.1 的这种注释方法可以注释一行或者多行，但一定要配对使用“/*”和“*/”，否则就要出错。目前大部分是使用 C++的编译环境（例如 Visual C++ 6.0），这些 C++环境提供一个行注释符“//”，用来告诉编译器，从“//”号开始向右的内容均不产加编译。例如：

```
#include <stdio.h>                  // 包含头文件
```

考虑到目前编写 C 程序时，主要是使用 C++编译器以及初学者比较容易漏掉“*/”号的实际情况，为了减少这个错误，并熟悉 C++的习惯，为以后学习提供方便，本书将在行注释中使用符号“//”作为注释符。

2. 主函数和库函数

在例 1.1 的程序中有一对花括号“{}”，可以把它看作程序体括号，还可以用一对花括号括起任何一组语句构成一个复合句（分程序），要注意在一个函数中至少应有一对花括号。C 程序的一般函数或主函数 main()之后应有一个“{”，在函数的最后应是一个“}”。在一个 C 程序或一个函数中，“{”和“}”必须成对出现。语句

```
printf("Hello! How are you?");
```

是一个函数调用语句，它调用名为 printf 的库函数，圆括号内由双引号括起来的部分是该语句所带的参数，即要打印的内容。printf 是标准输出函数，对应于输出设备终端显示器，上述语句表示要在其上输出字符串"Hello! How are you?"。在右括号“}”之后的分号“;”是语句结束标志。也就是说，C 语言必须用“;”号作为语句的结束标志。

C 语言函数分为两类：系统本身提供的库函数和自定义函数。所谓自定义函数，就是程序设计人员根据编程的需要自行设计一段程序函数完成一个特定的功能。C 语言中的主函数也被称为函数并且将其定义为 int main()。因此必须记住：一个 C 语言程序必须有一个主函数 main()，而且一个可执行的 C 语言程序总是从 main()函数开始执行的。这里使用 int main()，要求 main 返回一个整数值。但程序确实无需返回值，所以用语句“return 0;”返回一个为 0 的值以满足 int main()的要求。

库函数又称标准函数，例如标准输出函数 printf。标准函数定义在相应的头文件（头文件的后缀是.h）中。如果要使用这些标准函数，要先在主函数之前使用 #include 语句将相应的头文件包含，在需要调用它们的地方直接写上函数名，带上参数即可。例如 printf 函数的头文件是 stdio.h，使用如下语句将其包含。

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

然后就可以在程序中使用 printf 库函数实现输出功能。

C语言有非常丰富的库函数，应该尽量利用它们。例如，用来求一个整数的绝对值库函数abs，它是在头文件math.h中定义的，使用时需要包含math.h。

【例1.2】 使用库函数求一个整数绝对值的例子。

```
#include <stdio.h>
#include <math.h>
int main()
{
    int a;                              //声明一个整数变量
    printf("请输入一个整数：");
    scanf("%d",&a);                     //读入输入值
    printf("%d的绝对值的是%d\n",a,abs(a));
    return 0;
}
```

语句“int a;”是声明一个整型变量，在程序执行过程中，变量a的值可以改变，用它来存储从键盘输入的整数值。

假如求－82的绝对值，程序运行结果如下所示。

```
请输入一个整数：-82 <CR>
-82的绝对值的是82
```

如果编译系统支持中文，则可以在程序中使用汉字。在以后的例题中，有时为了方便学习，将采用汉字，读者如在西文操作系统下工作，换成相应的西文语言即可（中文并不影响程序的正确性）。

这里用符号“＜CR＞”代表按下回车（Enter）键。意思是在输入－82之后，要按一下回车键以表示输入完毕，以便通知程序将这个输入值取走并赋给变量a。一般来讲，程序要求输入时，均要以回车符作为输入认可，即在按回车键之前，还可以修改输入，一旦按了回车键，就无法修改了。

注意，以后除非是需要强调的地方使用“＜CR＞”以提示之外，不再给出这个符号。

scanf用来读取用户从键盘输入的值。scanf和printf的使用方法将在1.3节介绍。

3. 文件包含语句

#include语句是文件包含语句，它指的是一个程序把另一个指定文件的内容包含进来。书写时，可以使用引号也可以用尖括号。例如：

```
#include "filename"
```

或者

```
#include <filename>
```

都是在程序中把文件filename的内容（引号或尖括号是一定要的）包含进来。

另外还要注意，文件名是用双引号还是尖括号，其含义并不一样。使用尖括号时，C编译系统将首先在C语言系统指定的目录中寻找包含文件，如果没有找到，就到当前工作目录中去寻找，这是引用系统提供的包含文件所采用的方法。自己定义的包含文件一般都放在自己指定的目录中，所以在引用它们时，就采用双引号以通知C编译器在用户

指定的目录下和当前的目录下寻找包含文件。例如用户自定义的包含文件 myfile.h，引用形式为

```
#include "myfile.h"
```

在程序设计中，文件包含语句是非常有用的。一般 C 系统中带有大量的 .h 文件，用户可根据不同的需要将相应的 .h 文件包含起来。在例 1.2 中，因为要用到 C 语言提供的数学运算库，所以要用

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

语句。而标准输入输出是定义在库函数 stdio.h 中的，所以要用

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

语句。一般的 C 程序都离不开这条语句，初学 C 语言的读者也最容易遗漏这条语句。

本节介绍的虽然是最基本的 C 程序结构，但已经能够演出五彩缤纷的剧目来。

1.1.2 程序语句

一条完整的 C 程序语句必须以分号“;”结束。可把程序语句分成如下几类。

1. 声明及赋初值语句

用来声明变量的类型。int main()将主函数声明为整型，即 main 返回一个整数值。如果将例 1.2 的语句改为 int a=5;，则声明一个整型变量 a 并将 5 赋给变量。

2. 表达式语句

由一个表达式构成一个语句，用以描述算术运算、逻辑运算或产生某种特定动作。最典型的用法是由赋值表达式构成一个赋值语句。例如 a=3 是一个赋值表达式，而 a=3; 就是一个赋值语句。从中可以看到，一个表达式的最后加一个分号就构成了一个程序语句。一个程序语句最后必须出现分号，分号是程序语句中不可缺少的一部分。又例如

```
i = i + 1
```

是表达式，不是语句。而

```
i = i + 1;
```

是语句，作用是使 i 的值加 1。由此可见，任何表达式都可以加上分号而成为语句。

3. 程序控制语句

用来描述语句的执行条件与顺序的语句。如程序中的条件分支 if 语句。C 语言的控制语句有：

if()...else...	条件语句	for()...	循环语句
while()...	循环语句	do...while()	循环语句
continue	结束本次循环语句	break	中止循环或 switch 语句
switch	多分支选择语句	goto	转移语句
return	从函数返回语句		

以上9种语句中的括号()表示其中内容是一个条件,...表示内嵌的语句。例如,if()...else...的具体语句可写成:

```
if(x>y) z = x; else z = y;
```

详细的使用方法参见第2章。

4. 复合语句

C语句又分为简单语句和复合语句两种。在C语言中,将诸如

```
x = 1
printf("%d\n",x)
```

之类的表达式之后加上分号,即变成

```
x = 1;
printf("%d\n",x);
```

简单语句,分号是语句的终结符。

花括号"{"和"}"把一些说明和语句组合在一起,使它们在语法上等价于一个简单语句,被称为复合语句(或分程序)。例如,在程序段

```
if(a>=0)                                //1
{                                       //2
      printf("输入为:%d\n ",a);          //3
      return a;                         //4
}                                       //5
else                                    //6
{                                       //7
      printf("输入为:%d\n ",a);          //8
      return(-a);                       //9
}                                       //10
```

中,当a>=0的条件成立时,执行if后的复合语句,否则执行else之后的复合语句。结束一个复合语句的右花括号之后不能带分号(语句5和10),否则有时可能会导致错误;不能遗漏在复合语句的最后一条语句与右花括号之间的分号(语句4和9)。复合语句可由若干语句组成,这些语句可以是简单语句,也可以是复合语句,从而使C语言的语句形成一种层次结构。原则上可以不断地扩大这种层次。复合语句在程序中是一种十分重要的结构。

5. 函数调用语句

这是由一次函数调用加一个分号而构成的一个语句。例如:

```
printf("How are you?");
```

6. 空语句

下面是一个空语句。

```
;
```

即只有一个分号的语句，它什么也不做。

1.1.3 大小写字母的使用

C语言中严格区分大小写字母，如变量B和b是完全不同的两个变量。C语言惯用小写字母，而且以下划线“_”字符开头的标识符一般由系统内部使用，用户最好不要用它作为标识符的第一个字符。另外，习惯上把自己定义的标识符用大写字母表示。

1.1.4 程序的书写格式

C语言的格式很自由，一行也可以写几条语句。不过，C语言的适当格式对于充分理解这种语言非常重要。一个适当格式的程序和一个不适当格式的程序就像一封打得很漂亮的信和一封写得非常凌乱的信，给人的印象是大不一样的。应该使源代码易于理解，特别是容易被输入这些程序的程序员所理解，这有助于复杂程序的调试及修改以前输入的代码。

应使用缩进式和必要的空行的书写风格，可使源代码具有层次性和逻辑性，以增加程序的可读性和可操作性。

一般来讲，每次缩进5个字符的位置，并按程序特性设置空行。在本书中，为了节省篇幅，有意识地减少空行及缩进字符的数量。读者在编写程序时不要模仿，应注意养成良好的书写风格。

在书写程序语句时，一般应注意如下规则。

(1) 括号紧跟在函数名的后面，但在for和while后面，应用一个空格与左括号隔开以增加可读性。

(2) 数学运算符的左右各留一个空格以与表达式区别。

(3) 在表示参数时，逗号后面留一个空格。

(4) 在if、for、do...while和while语句中，合理使用缩进、一对花括号和空行。

(5) 在if...else之类的语句及其嵌套语句中，注意书写格式要易于排错并提高可读性。

1.1.5 简单C程序的基本结构模式

从例1.2可以总结出只有主函数的简单C程序的基本结构模式如下：

```
文件包含部分          //包含头文件等
int main()            //主程序
{                     //主程序开始
    声明语句部分      //声明程序中所要使用的变量
    执行语句部分      //程序的可执行语句
    return 0;         //返回值
}                     //主程序结束
```

声明部分目前只涉及变量，不管以后章节增加何种声明，这些声明必需都放在可执行语句之前，即所有声明放在一起。应该牢牢掌握并熟练使用这种结构，它已经完全能解决很多程序设计问题。

1.2 基本的输入与输出

输入输出设施不是C语言的一部分，而是以标准函数形式提供。在每个引用库函数的源程序文件的开头处含有如下语句：

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

文件stdio.h定义了I/O库所用的某些宏和变量，使用#include语句把它包含进来，一起编译。虽然有的C编译器使用scanf和printf函数不需要包含它，但建议养成使用这条语句的习惯。下面简要介绍格式化输入和输出函数(scanf和printf)，以方便目前的学习，关于它们的详细使用方法参见1.7、1.8节。

C语言标准库中提供了很有用的格式化输入、输出标准函数，为程序员实现各种格式化输入、输出提供了方便。

格式化输出函数printf的形式如下：

```
printf(控制字符串,参数1,参数2,…);
```

printf的功能是按照控制字符串将参数进行转换，按格式在标准输出设备上输出。在控制字符串中包含两种字符：一种是普通字符，将其原样输出；另一种是格式符，它是以“%”开头后跟格式字符，说明其对应参数的输出格式。下面先介绍最常用的4种格式符。

(1) d：将参数按十进制整数形式输出。

(2) c：将参数看作单个字符输出。

(3) s：将参数所指出的至空字符为止的字符串输出。

(4) f：参数按浮点数形式输出。

例如：

```
x1 = 1234;
x2 = 2345;
printf("\nx1 = %d x2 = %d\n",x1,x2);
```

其中，"\nx1＝%d x2＝%d\n"是控制字符串，x1、x2是参数，控制字符串中的x1＝和x2＝都是普通字符，因此按原样输出；两个%d是格式说明符，依次说明x1、x2的值均按十进制(整型)形式输出。\n是换行符，第一个\n表示先换行再输出x1和x2及其值。第二个\n是在输出x1和x2之后再换到下一行。上述语句执行后的输出结果如下：

```
x1 = 1234 x2 = 2345
```

格式化输入函数scanf提供的格式与printf类似，其格式为：

```
scanf(控制字符串,参数1,参数2,…);
```

scanf函数实现从标准输入设备(终端)上按控制字符所规定的格式输入数值或字符，并将输入内容存入参数所指定的单元中。对于参数的书写，程序设计人员要特别注意，在printf中的参数是给出要输出值的变量名，而scanf中的参数是要给出接收数据的变量地址。例如语句

```
scanf("%d %d",&x,&y);
```

实现从终端输入两个十进制数分别赋给 x 和 y。这里 &x 和 &y 表示 x 及 y 的地址。在输入两个十进制数之间用空格分隔。

【例 1.3】 编写一个无返回值的主程序，输入正方形的边长 A，求它的周长。

```
#include <stdio.h>
void main() //void 不需要返回值，所以不使用 return 语句
{
  int A;
  printf("请输入边长:");
  scanf("%d",&A);
  printf("边长 = %d 周长 = %d\n",A,4 * A);
}
```

输入 45，输出如下所示。

```
请输入边长:45
边长 = 45 周长 = 180
```

1.3 初学者最容易出现的错误

编程中总难免会有错误，关键是要知道如何去检查错误和排除错误。严格来说，错误并无规律，而且是千奇百怪，检错和排除确实没有什么技术，但是又都离不开这一步，程序员心中必须对此有个正确的认识。尽管现在还没有真正接触 C 语言的语句，但本节还是及早提醒读者在书写 C 语句时，千万注意避免最容易犯的语法规则错误。而人们往往也是在最容易的地方出错。下面是几点注意事项。

1. 忘记主程序 main()的返回类型和包含 stdio.h 文件

C 语言提供的标准主函数是 int，需要语句 return 0；与 main 配合。尽管有些语言环境不需要使用语句 #include <stdio.h>，但建议要养成使用这条语句的习惯。这种习惯对以后学习 C++很有好处。注意不能在包含库文件和自定义的函数名称后面使用分号，例如语句 int main();和 #include <stdio.h>; 都是错误的。这是一个常见错误，就是高水平的专业软件工作者，有时也会犯此类错误。

2. 遗漏分号和花括号，或者增加分号和花括号

初学者容易在 printf 语句后遗漏结束符“;”，而在 #include 语句中又误用“;”号作为结束符。

花括号是成对出现的，遗漏花括号会改变程序的运行方式。一般来说，人们更容易忘记右边作为程序结束的花括号。

3. 在 scanf 语句中忘记使用“&”号或多了“\n”号

scanf 语句中的变量前面应加上 & 号，下面的语句应为 &x，格式符中多了\n 号：

```
scanf("%d\n",x);          //错误语句
```

4. 程序中的注释符号使用不当

程序中的注释符号左边是 /*，右边是 */。若右边的符号错成 /* 或遗漏，而后面又有注释，就可能会使许多行的程序变成注释，影响运行结果。下面是一个假设的理想例子。

```
…;          /* 右面的注释符号遗漏
…;
…;          左面的注释符号遗漏 */
```

编译通过，结果造成中间的语句没有参加编译。建议在行中使用//号注释。

1.4 使用C程序解题的完整过程

使用计算机完成解题分为编辑、编译和运行程序 3 个过程。

1.4.1 程序的编辑、编译和运行的基本概念

用一种高级语言写成的程序称为源程序，可以在具有该种语言编译系统的不同计算机上使用。源程序必须翻译成机器语言才能执行。逐条翻译并执行的翻译程序称为解释程序，例如 BASIC 语言解释程序。而将源程序一次翻译成目标程序然后再执行的翻译程序称为编译程序，例如 C 编译程序。

要得到一个用 C 语言设计的，名为 mycfile.exe 的可执行文件，其过程可以分为如下几个步骤：

(1) 使用编辑器编辑一个 C 程序 mycfile.c，又称其为 C 的源程序。

(2) 使用 C 编译器对这个 C 程序进行编译，产生文件 mycfile.obj。

(3) 使用连接程序(又称 Link)，将 mycfile.obj 变成 mycfile.exe 文件。

在 C 语言的最初应用阶段，这些工作需要分步骤进行。现在一般都使用集成环境。所谓集成环境，就是将 C 语言的编辑、编译、连接和运行程序都集成到一个综合环境中。尤其是目前 Windows 应用软件界面的标准化，菜单名称和作用的规范以及 Windows 程序使用的普及，大大降低了集成开发环境使用的困难，使用集成环境的帮助文件更给编程和学习带来方便。

1.4.2 熟悉使用集成环境的重要性

计算机著名科学家沃思提出“数据结构＋算法＝程序”的思想，现在人们不仅认识到“程序设计方法”的重要性，而且对编程环境也给予足够重视。目前的编译环境都朝集成环境和可视化编程方向发展。由于集成环境功能强大，也使其具有一定的复杂性。典型的是 Visual C++ 6.0，提供了 MFC 类库，实现许多 Windows 自动编程功能。因此，如何用好这些语言工具和环境，也是提高编程效率的因素之一。目前认为可以用下面的公式表示程序：

数据结构 + 算法 + 程序设计方法 + 编程工具 = 程序

一个程序设计人员应该掌握以上4个方面的知识，人们一般容易忽视第4个因素——编程工具。有些人能写出正确的程序，却无法使用编程工具产生可执行文件。为了消除这种现象，本书每章均给出实验题以加强训练，使学生既学会“做什么”，也掌握“如何做”的全过程。

1.4.3 解题的简单过程

首先要对计算机解决问题的步骤有个初步认识。下面就结合一个实际问题进一步说明使用C语言解题的步骤。

(1) 设计一个解题的方法。

(2) 使用一种方式把它描述出来。

(3) 把它们转化成程序形式。

(4) 在计算机上编辑成C的源文件。

(5) 编译C程序源文件的过程，同时也是查错的过程。如果不能正确编译，进行查错，直到产生正确的OBJ文件。

(6) 将它们连接成EXE文件，如果连接出错，返回步骤(4)，再次检查和编译源文件。

(7) 运行EXE文件，得出初步结果。

(8) 验证结果是否正确。如果结果不正确，就要返回查找原因，直到运行结果正确为止。

表面上看，运行结果不正确要返回步骤(4)检查程序，其实这是对步骤(3)的复查。如果算法设计不对，要转到步骤(1)，也即重复步骤(1)～(4)。如果题目复杂，则要从头开始。因此，一定要重视前3个步骤。

下面结合一个简单例子，具体说明前面的几个过程。

【例1.4】 编写将输入的两个整数相加，然后输出它们的和的程序。

(1) 设计解题方法。显然，这是一个顺序执行的过程，要求按顺序输入两个整数，再把它们相加，最后输出它们的和。

(2) 使用自然语言方式把它描述出来。例如：

```
程序开始
要求提示输入2个整数
赋值给变量number1,number2
计算 sum = number1 + number2
输出 sum
程序结束
```

如果使用英文及语句说明，如下所示：

```
BEGIN
 Print InputMessage
 Input number1,number2
 sum = number1 + number2
 Print sum
END
```