第 1 章　C语言概述

C语言是世界上最流行的程序设计语言之一。人们之所以选择 C 语言，主要是由于它具有一些吸引人的优点。作为一个初学者，了解它的这些特点很有必要。通过对这些特点的了解，可以对它有一个概括性的认识。随着今后的学习，这种认识将会得到逐步理解和加深。

C语言的特点是由它的历史所决定的。本章就是在回顾 C语言的历史的基础上介绍它的一些特点，并通过几个例子使读者能对 C 语言有一个整体的认识。

1.1　C语言的发展历史和特点

1.1.1　C语言的发展历史

C语言是伴随着 UNIX 操作系统的发展而产生的，了解 C语言的历史应先从 UNIX 操作系统的历史谈起。

UNIX 操作系统最初是由美国贝尔实验室的两个研究人员 K.Thompson 和 D.M.Ritchie 共同开发出来的。从 1969 年开始，这两人在 PDP-7 机器上为自己的开发软件开发环境，他们用了不到两年的时间就完成了一个系统，并为它取名为 UNIX，这就是 UNIX 操作系统的雏形。此时的 UNIX 是用汇编语言写成的，由于汇编语言不可移植，可读性差，而且描述问题的效率又不如高级语言高，因此， K.Thompson 决定开发一种高级语言来更有效地描述 UNIX 系统。1963 年英国剑桥大学曾推出 CPL 语言，由于规模比较大难以实现，该校的 Matin Richards 对 CPL 语言作了简化后推出了 BCPL 语言。K.Thompson 决定在 BCPL 语言的基础上开发自己的高级语言，于是，在 1970 年，K.Thompson 对 BCPL 语言作了进一步的简化工作，设计出简单而且很接近硬件的一种高级程序设计语言，取名为 B 语言（取 BCPL 的第一个字母），并于 1971年在 PDP-11/20 上实现了 B 语言，而且用 B 语言编写了 UNIX 操作系统和绝大多数实用程序。由于 B 语言面向字存取、功能过于简单、数据无类型和描述问题的能力有限，而且 B 编译程序产生的是解释执行的代码，运行速度慢，因此，没有流行起来。在这种情况下， D.M.Ritchie 于 1971 年开始用一年多的时间在 B 语言的基础上设计出一种新语言，这种新语言克服了 B 语言功能过于简单、数据无类型和描述问题的能力有限的缺点，又保留了 B 语言的简洁、接近硬件的优点，同时，扩充了很多适合于系统设计和应用开发的功能，因为这种新语言是在 B 语言的基础上开发出来的，不管是在英文字母序列中也好，还是在 BCPL 这个名字中也好，排在 B后面的均为 C，因此将这种语言命名为 C。于是，C 语言就诞生了。1973 年，K.Thompson 和 D.M.Ritchie 用 C 语言将 UNIX 操作系统 90% 以上的部分重写了一遍，并加入了多道程序设计等新的功能。这就是 UNIX 操作系统的第 5 版。

C 语言最初是为了描述和开发 UNIX 操作系统而设计的，因此，C 语言出现后的初期，它只是在贝尔实验室内部使用，直到 1975 年 UNIX 操作系统第 6 版推出后，C 语言众多的突出优点才得到人们的普遍关注。1977 年出现了不依赖于具体机器的 C 语言编译文本——可移植 C 语言编译程序，使 C 语言的可移植性大大加强了，这就推动了 UNIX 操作系统在各种机器上的实现。因此，如今 UNIX 操作系统在世界范围内的巨大成功与 C 语言是分不开的。另一方面，UNIX 操作系统的成功以及广泛应用，也使人们认识了 C 语言。UNIX 操作系统和 C 语言就是在互相依存、互相促进中共同发展的。1978 年以后，C 语言被移植到各种机器上，并独立于 UNIX 操作系统。如今，C 语言已经风靡全球，成为世界上应用最广泛的程序设计语言之一。

UNIX 操作系统和 C 语言巨大成功使得它们的两位主要开发者 K.Thompson 和 D.M.Ritchie 于 1983 年双双获得了计算机科学和技术领域中的最高奖——图灵奖。

C 语言出现以后，随着微型机的迅速普及，逐渐出现了许多 C 语言系统，它们之间存在许多差异，给程序的移植带来很大的困难。为了改变这种状况，1983 年，美国国家标准化协会（ANSI）根据 C 语言诞生以来各种版本对 C 语言的发展和扩充，制定了新的标准，也就是常说的 ANSI C。1987 年，ANSI 又公布了新标准——87 ANSI C 。目前流行的 C 编译程序都是以它为基础的。尽管这样，各种版本的 C 编译系统还是略有差异，因此，读者在使用具体编译系统时，还应参考相关的手册以了解具体的规定。

1.1.2 C 语言的特点

C 语言之所以能够取得巨大成功，人们之所以选择 C 语言，主要是取决于 C 语言所具有的独特的优点。这些优点主要表现在如下几个方面：

1. 简洁

C 语言在语言的表达方式上尽可能的简洁，将在其他语言中通常要用多个语句才能表达的操作用一个运算就完成了，比如它独有的运算符条件运算符“？：”就是在一个表达式中完成了条件判断和由此决定的表达式的选择（二选一），因此，对于语句

```
x=a>b?a:b;
```

等价于下列语句

```
if(a>b)
   x=a;
else
   x=b;
```

又比如它独有的增量运算符同时具有提供表达式的值和变量本身增值两个作用，因此，对于语句

```
a=i++;
```

等价于下列语句

```
a=i;
```

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

显然，在这两个例子中前者要比后者简洁。再比如它的复合赋值运算符使得表达式更加简洁，对于表达式

```
a+=b
```

等价于

```
a=a+b
```

这些简洁的表达方式不仅使程序的编写更加精练，而且减少了程序员的书写量，大大地提高了编程效率。C 语言的简洁还体现在它摒弃了一切不必要的成分，使语言只保留了32个保留字和 9种控制语句，语言本身并不提供输入输出机制和字符串处理等内容，但这并不是说用 C 语言不能实现输入输出和字符串处理等方面的应用，而是把输入输出和字符串处理等内容放到函数库中以函数的形式提供给程序员，程序员在需要的时候可以调用相应的函数，这样就保持了语言的简洁，使语言简单易学，编出来的程序短小精悍， C 编译系统的实现也变得比较简单。

2. 语言表达能力强

C 语言是一种高级程序设计语言，能完成数值计算、字符、数据等的处理，具有高级语言的通用性。同时，它又能对物理地址进行访问，对数据的位进行处理和运算，具备低级语言的特点。C 语言这种兼具高级语言和低级语言功能的特点使得它能够代替低级语言开发系统软件和应用软件，著名的 UNIX 操作系统的 90%以上的代码就是用 C 语言实现的。

3. 运算符丰富

C 语言除了具有其他高级程序设计语言所具有的运算符外，还具有 C 语言特有的运算符，比如增量运算符、赋值运算符、逗号运算符、条件运算符、移位运算符和强制类型转换运算符等。如此多的运算符使得 C 语言的绝大多数的处理和运算都可以用运算符来表达，提高了 C 语言的表达能力。

4. 数据结构丰富

C 语言具有其他高级语言所具有的各种数据结构，而且 C语言又赋予了这些数据结构更加丰富的特性，比如 C 语言的整型数又分为长整型、短整型、无符号整型以及基本的整型，这样，用 C语言的数据类型能够实现复杂的数据结构，更加准确地描述客观世界。

5. 生成的代码质量高

C 语言之所以能够用来开发系统软件，还有一个很重要的原因，就是用它开发的程序生成的代码的质量高。因为开发系统软件要求用来开发的语言生成的代码质量要高，否则系统的开销很大，使系统很难实用。试验表明，用 C语言开发的程序生成的目标代码的效率只比用汇编语言开发同样程序生成的目标代码的效率低 10%到 20%，而由于用高级语言开发程序描述算法比用汇编语言描述算法要简单、快捷，编写的程序可读性好，修改、调试容易；因此，用 C 语言开发程序总的效率并不比用汇编语言开发程序的效率低多少。所以，C语言就成为人们用来开发系统软件和应用软件的一个比较理想的工具。

6．可移植性好

由于 C 语言程序本身不依赖于机器的硬件系统；因此，用 C 语言能够开发出可移植性很好的程序。所谓可移植性就是说一个程序需要修改多少才可在不同的硬件环境中运行。用 C 语言编制的程序只需少量修改，甚至可以不用修改就可以在其他的硬件环境中运行；因此，C 语言程序的可移植性好。正是因为 C 语言的可移植性好，UNIX 操作系统才可以迅速地在各种机型上得以实现和使用。

7．结构化语言

C 语言是结构化的程序设计语言，其主要结构成分是函数，可通过函数实现程序段的共享。另外，C 语言具有结构化的控制语句，支持多种循环结构，使用复合语句也支持实现程序的结构化。这些特点使得 C 语言层次清晰、结构紧凑，比非结构化的语言更易于使用和维护。

C 语言虽然具有上述这些优点，但也并非尽善尽美，它也存在一些缺点。比如， C 语言使用起来比较灵活，没有太多的限制，这是一个优点，但同时也是一个缺点，因为对程序员没有太多的限制就有可能带来一些副作用。例如，C 语言允许程序员对任意的物理地址进行访问，而且不加检验，这就使程序员有可能访问系统使用的禁止其他人访问的内存单元，造成程序错误甚至系统瘫痪。又比如，C 语言不对数组下标是否越界进行检验，这就有可能出现读取非本数组的数据或修改了其他单元的数据，使程序不能正确执行。上述错误都是 C 编译系统不检验因而也发现不了的错误，带有这些错误的程序虽然能被 C 语言编译系统认可，但却是错误的程序。因此，要使用 C 语言对程序员的要求比较高，要求程序员首先要透彻地理解 C 语言，能够熟练地运用 C 语言，这样才可避免可能出现的错误。尽管 C 语言有这些缺点，但仍不失为一种优秀的程序设计语言。

对于 C 语言的上述这些特点读者可能暂时还不能完全理解，随着以后的学习会逐渐有所体会。

1.2 C 语言程序结构简介

为了使读者能够对 C 语言程序有一个感性的认识，这一节我们先列举并分析几个简单的程序。如果读者不能够完全理解也没有关系，我们将在后面的章节逐步介绍 C 语言的各个语言点。

【例 1.1】分析下面这个最简单的 C 程序。

```
1   #include "stdio.h"
2   main()
3   {
4       printf("Hello,world!\n");
5   }
```

程序运行的输出结果如下：

```
Hello,world!
```

【程序分析】该程序是C语言最经典、最简单的入门程序，它的功能只是输出一个句子，即Hello,world!。程序的第1行是一条预编译命令，其中的stdio.h是一个嵌入文件，又称为头文件，该文件包含了C语言提供的标准输入输出库函数的头部说明。该预编译命令的作用是在程序编译时，要先把文件stdio.h嵌入到源程序的前面，然后将文件stdio.h的内容和源程序一起形成的文件进行编译。我们之所以要嵌入该文件，是因为在我们的这个程序中用到了标准输入输出库函数printf。C语言编译系统提供了许多头文件，每一个头文件都包含了一类标准库函数的头部说明。在C语言中规定，在源程序中不论用到哪一个标准库函数，都要在该源程序的前面嵌入该类库函数的头文件。因此，第1行用了这个预编译命令。

程序的第2行是被定义的函数的头部，其中main是函数名，因为在C语言中规定，当在一个标识符的后面紧跟有一对圆括号时说明这是一个函数。圆括号中是形式参数表，在本函数中的这个圆括号中什么都没有，表示没有形式参数。值得说明的是main是C语言中主函数的专有名，C语言规定，在C语言的源程序中，必须有而且只能有一个main函数（即主函数）。每次运行C语言程序时，都要从主函数（main函数）开始运行。

程序的第3行是一个左大括号“{”，表示函数的开始，程序的第5行是一个右大括号“}”，表示函数的结束。有开始则必有结束，因此，在程序中左大括号和右大括号必须成对出现。在这两个大括号之间是main函数的函数体，该函数需要执行的语句和操作将书写在这两个大括号之间。在本程序中只有一条语句，那就是第4行的语句printf("Hello,world!\n");。这条语句中printf后面紧跟有一对圆括号，说明它是一个函数，实际上printf是一个标准输出函数，本语句中，该函数的作用是将它的实参（字符串Hello,world!\n）输出到屏幕上。可是在屏幕上没有见到“\n”，这是因为“\n”表示一个不可打印的字符，叫换行符，它的作用是使屏幕显示的光标或打印机的打印头移到下一行的开始，在这之后如果再有输出语句，则显示的普通字符将在下一行开始显示。另外，在这条语句的最后是一个分号“；”，这个分号是语句的终止符，表示一条语句的结束，C语言规定，在C语言的每一条语句的结尾都必须有这个分号。

【例1.2】求两个整数之和。

```
1   /*求两个整数之和*/
2   #include "stdio.h"
3   main()
4   {
5       int a,b,sum;  /*定义变量*/
6       a=12;
7       b=32;
8       sum=a+b;
9       printf("a+b=%d;\n"
           sum);
10  }
```

程序运行的输出结果为：

```
a+b=44;
```

首先看上面书写的程序中每一行的首行都有一个数字，这个数字实际上不属于程序本身包含的内容。我们在此之所以要人为地加上这个数字是想用它来作为行号，表示程序语

句所在的行数。这是为了方便我们在分析程序时对程序语句的引用，当我们要分析那条语句时，只要指出其行数，我们就可以一目了然地看到那一条语句了。值得注意的是我们在编写程序时千万不要写行号，在录入书上的例子程序时，也千万不要将行号录入进去。对于本书后面例子中的行号也是如此，以后不再重述。

【程序分析】 程序的第 1 行是一个注释行，其作用是对程序进行注释和说明，用来说明整个程序、某个程序段或程序中的数据或语句的作用和功能以及算法等情况。在编译系统对源程序进行编译时，注释将被忽略，不形成目标码，因此，注释对程序的编译和运行不起任何作用。C 语言中规定，注释可以是以“ /* ”开始并以“ */ ”结尾的任意字符串，在“ /* ”和“ */ ”之间可以加入任何说明性的文字。注释主要是用于提高程序的可读性、帮助阅读程序而设计的，不管是阅读别人的程序还是阅读自己编写的程序，程序中的注释都能起到帮助理解程序的作用。因此，建议读者在自己编写程序时要养成加入足够数量注释的良好习惯，以便在别人阅读你的程序时或者自己修改自己的程序时能够尽快和更好地理解程序。注释可以出现在程序的任何地方，也可以出现在语句的后面，一条注释还可以占用多行。使用注释时应当注意以下几个方面：

（1）“/* ”和“ */” 必须成对出现。

（2）注释不能嵌套。也就是说在注释之间不能再加注释。因此， /*.../*...*/...*/形式的注释是不合法的。

（3）注释不能出现在一个字符常量或一个字符串常量之间。

程序的第 2 行是一条预编译命令，第 3 行是 main 函数说明的函数头部，第 4 行和第 10 行是一对大括号，表示函数 main 的开始和结束，这些都和例 1.1 的程序一样。在这对大括号之间，也就是程序的第 5 行到第 9 行是 main 函数的函数体部分，这部分与例 1.1 的程序有区别。程序的第 5 行是一条变量说明语句和一条注释语句，在变量说明语句中， int 是一个关键字，表示变量的类型，其作用是说明后面的标识符代表整型变量；因此， a、b 和 sum 都代表整型变量。使用变量说明语句时要注意，在变量的类型关键字和变量名之间要有空格，如果要在一条变量说明语句中同时说明多个变量时，在各个变量名之间要加入逗号，而且在整个语句的最后要加上分号作为终止符。

程序的第 6 行和第 7 行是两条赋值语句，分别给变量 a 和 b 赋值 12 和 32。一个变量代表一个存储单元，里面保存着该变量的值，给该变量赋值就等于将该值存储在该变量所代表的存储单元。因此，执行第 6 行和第 7 行两条赋值语句后，变量 a 和 b 所代表的存储单元就分别保存着整数 12 和 32，这两个值也称为这两个变量的值。

程序的第 8 行也是一条赋值语句，是先计算表达式 a+b 的值，然后将计算结果赋值给变量 sum。因为变量 a 和 b 的值分别为 12 和 32，因此 a+b 的值为 44。所以，运行完第 8 行的语句后，变量 sum 的值就为 44 了。

程序的第 9 行是一条标准输出语句，其中的函数 printf 有两个参数，分别为“ a+b=%d;\n”和 sum。第一个参数表示输出格式的控制信息，其中的“ %d ”是输出转换说明，用来指定输出数据的类型和格式，“%d ”表示十进制整数类型。当运行该函数调用时，输出格式控制信息中的普通字符如“ a+b= ”按原样输出。其中类似“ %d ”的输出转换说明的位置用后

面相应参数的值按指定的类型和格式依次代替输出，此时后面参数 sum 的值为 44，因此，第 9 行运行后输出

```
a+b=44;
```

“;”在 44 后面原样输出，“\n”是换行符，在屏幕上没有显示，只是使光标移到下一行。

注意，在程序的一行中也可以编写多条语句，比如本程序的第 6、7、8 行就可以合在一起写在一行中，写成

```
a=12;b=32;sum=a+b;
```

因为每条语句的最后都有一个分号作为终止符。在一行中，当遇到一个分号后，如果再遇到一个其他的普通字符，编译程序就将它看作下一条语句的开始。这样，编译系统就可以自动地识别一行中的多条语句。虽然在程序的一行中，可以编写多条语句，但为了清晰起见，我们还是建议在一行中只编写一条语句。

【例 1.3】 求两个整数之中的最大值。

```
1   #include "stdio.h"
2   int max(a,b)
3   int a,b;
4   {
5     if (a>b)
6       return a;
7    else
8      return b;
9
10  main()
11  {
12    int x,y;
13    printf("please input x and
          y:\n");
14    scanf("%d%d",&x,&y);
15    printf("the max of  %d and
        %d is %d.\n",x,y,
        max(x,y));
16  }
```

程序运行的输出结果为：

```
please input x and y:
```

然后，程序并没有结束，而是等待用户用键盘输入变量 x 和 y 的值。此时，若用键盘输入

```
23，59<回车>
```

程序则继续运行，输出

```
the max of 23 and 59 is 59
```

然后程序结束。

【程序分析】在这个程序中，共定义了两个函数，一个函数是主函数（main 函数），还有一个函数是 max 函数。程序的第 1 行仍然是一条预编译命令。程序的第 2 行至第 9 行定义了函数 max。

程序的第 2 行是函数 max 的头部说明，因为在 max 后紧跟有一对圆括号，说明 max 是一个函数名。在 max 的前面有一个类型关键字 int，该关键字是用来说明该函数的返回值

的类型的。本例中，max 函数的返回值的类型为整型。另外，在 max 后面的括号中有“ a,b”，它称为形参表，说明 max 函数有两个输入参数，用 a 和 b 表示。

程序的第 3 行是一个类型说明语句，用来定义本函数的形参的类型。此语句说明 max 函数的两个形参 a 和 b 为整型。

程序的第 5、6、7、8 行为一个复合语句。第 5 行判断表达式 a>b 的值，如果 a>b 成立，即 a>b 的结果为真（值 1），则执行其后面第 6 行的语句，该语句（第 6 行的语句）用变量 a 的值作为该函数的结果值并结束该函数返回；如果 a>b 不成立，即 a>b 的结果为假(值 0），则运行第 8 行的语句，该语句（第 8 行的语句）用变量 b 的值作为该函数的结果值并结束该函数返回。从以上分析可以看出，若 a 的值大于 b 的值，则函数返回的结果值等于 a 的值，否则，函数返回的结果值等于 b 的值，即返回变量 a 和 b 之中的最大值。

程序的第 10 行至第 16 行是程序的主函数说明部分。第 12 行至第 15 行是主函数的函数体。程序的第 12 行定义了两个整型变量 x 和 y。程序的第 13 行是一个标准输出函数，用来输出字符串 please input x and y:，显示该字符串来提示用户从键盘输入两个数据作为变量 x 和 y 的值。程序的第 14 行是一个标准输入函数 scanf，用来从键盘接收用户输入的数据。该函数在本程序中有 3 个参数，第 1 个参数“ %d%d ”是用于输入控制说明，用来说明从键盘接收数据的类型，后面的两个参数“ &x” 和“ &y” 分别代表变量 x 和 y 的地址。执行该函数的结果是按照输入控制说明指定的类型格式将从键盘输入的数据依次存入后面相应的参数指定的地址所指的存储单元中。在本程序中，执行第 14 行语句后，将把用户从键盘输入的两个数据分别存入变量 x 和 y 的存储单元中；因此，当输入 23 和 59 后，变量 x 和 y 的值就分别变成 23 和 59。

程序的第 15 行是一个标准输出函数，其输出格式的控制信息为

```
the max of  %d and %d is %d.\n
```

在输出时，其中的普通字符原样输出，而其中的 3 个“ %d ”是 3 个输出转换说明，在输出时分别用后面的 3 个参数 x、y 和 max(x,y) 的值代替。max(x,y) 是一个函数调用组成的表达式，当调用 max 函数时，将变量 x 和 y 的值作为实参值传给函数，并运行 max 函数。max 函数开始运行后，先将主程序传过来的实参值分别赋值给形参 a 和 b，然后运行函数 max，max 运行结束后返回一个值作为 max 函数调用的结果值。因此，当用户用键盘输入两个整数 23 和 59 后，运行第 15 行程序，输出结果为

```
the max of 23 and 59 is 59.
```

然后，程序结束。

通过对上面 3 个程序的分析，相信读者一定会对 C 语言程序有一个初步的感性认识了。虽然对其中的某些细节问题可能还不能完全理解，但至少应当了解到 C 语言程序结构以下几个方面的特点：

（1）C 语言程序是由函数构成的。一个 C 语言程序可以由一个或多个函数构成，但其中必须有而且只能有一个主函数，这个主函数名必须为 main。主函数是 C 语言程序的入口点，每次执行 C 语言程序时都要从主函数开始执行。

（2）在一个函数中可以调用另一个函数（非主函数也可以调用函数自身），这个被调

用的函数可以是用户定义的函数，也可以是系统提供的标准库函数（比如 printf 和 scanf）。这样通过把一个问题分解成若干个子问题，将子问题用独立的函数来实现，然后通过函数调用把整个程序组织起来，实现了程序的模块化，从而实现对复杂问题的解决。使用函数时，建议读者尽量使用库函数，这样不仅能够缩短开发时间，也能提高软件的可靠性，从而开发出可靠性高、可读性好以及可移植性好的程序。

（3 函数定义的一般形式如下：

```
函数类型  函数名（形式参数表）
形式参数类型说明
{
   数据说明部分
   执行语句部分
 }
```

（4）可以在程序的任何位置给程序加上注释，注释的形式为

```
/*注释内容*/
```

注释是为了提高程序可读性的一个手段，它对程序的编译和运行没有任何影响。使用注释时要注意不要将注释嵌套。

（5）C 语言程序的书写格式非常自由，一条语句可以在一行内书写，也可以分成多行书写，而且一行可以书写多条语句。尽管这样，我们还是建议在一行只写一条语句，而且采用逐层缩进的形式，这样使得程序的逻辑层次一目了然，便于对程序的阅读、理解和修改。

（6）C 语言中每条语句都以分号结尾，分号是 C 语言语句不可缺少的有机组成部分。

1.3 C 语言程序的开发过程

我们开发 C 语言程序的最终目的是要让所开发的程序在机器上运行起来，在运行的过程中完成我们需要的工作。

上一节我们了解了 C 语言程序的结构，并分析了几个 C 语言程序。如果把上一节中的例子程序输入到计算机中形成文件，该文件能够在计算机上运行吗？答案是否定的。因为计算机不能识别源程序，它只能识别二进制代码组成的机器指令数据，并按指令预定的含义执行一系列动作。这种计算机能够识别的二进制代码就是所说的目标代码。我们要想让计算机执行我们的程序，就需要首先将我们的源程序转换成目标代码。这种转换工作手工很难完成，通常需要一种特定的软件工具，我们称这种软件工具为编译程序，而把这种转换工作称为程序编译。

C 语言程序的编译过程一般经历如下几个步骤：

（1）预处理。在该步骤编译程序主要对源程序中的宏定义、文件包含以及条件编译等进行预先处理。

（2）第一遍扫描。在该步骤中编译程序要对源程序进行词法分析、语法分析以及符号

表维护，并为表达式构造语法树。

（3）第二遍扫描。在该步骤主要为表达式生成代码。

（4）内部汇编优化。把上一步生成的内部汇编语言代码进行汇编和优化，生成机器可识别的目标代码。

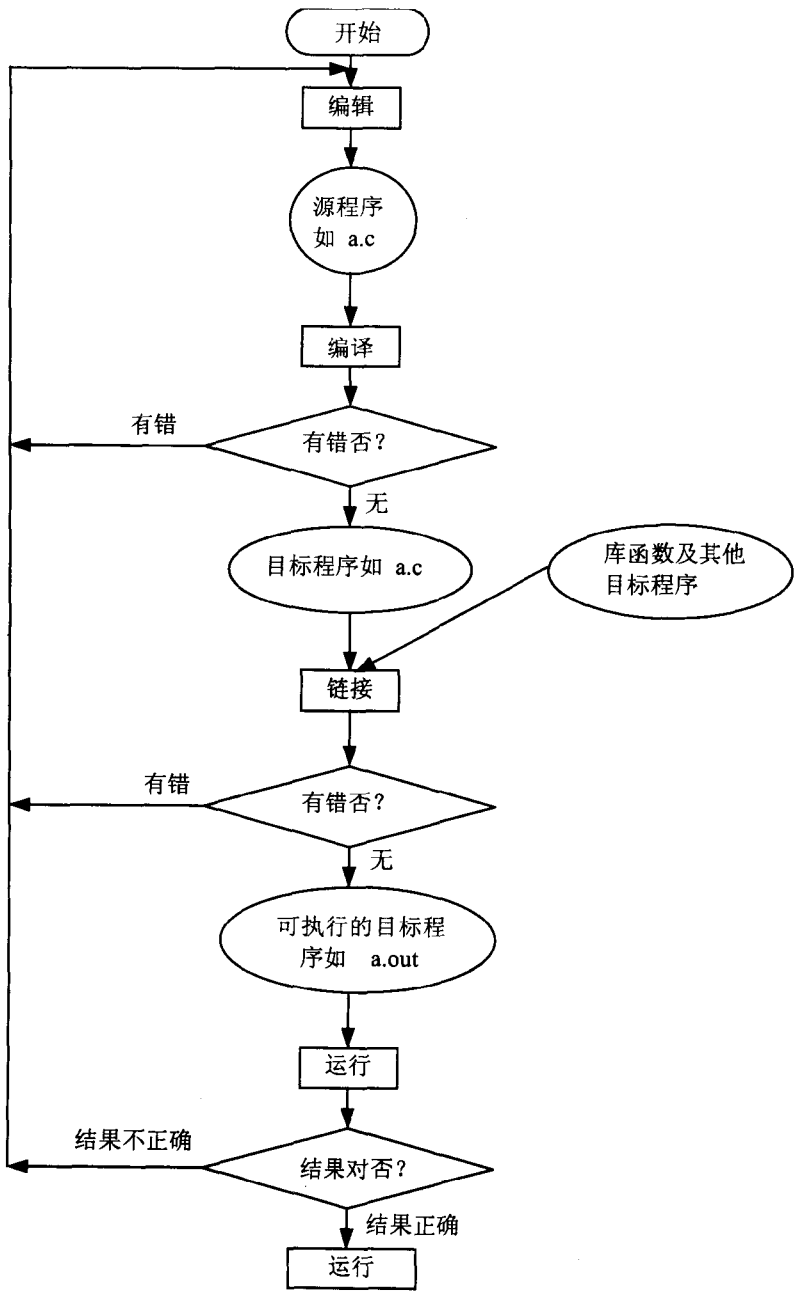

图 1.1 C语言程序的开发过程

在程序编译的过程中，编译程序通常会发现程序中的语法错误。当编译程序发现程序中有语法错误时，就会在编译结束后向用户报告有关程序的错误信息，比如错误可能出现的位置以及错误的类型等。此时，我们要对源程序进行修改，然后再重新编译。如果程序没有语法错误，则编译结束后就会生成一个目标文件。

此时的目标文件还不能在计算机上运行。因为程序中可能会用到库函数或其他函数，需要将它们连成一个统一的整体。这一步工作称为连接。经过连接就把分离的目标文件连成一个完整的可执行程序，对应的文件就是可执行文件。

能够生成可执行文件不等于程序正确。在开发程序的过程中，除了上面我们提到的语法错误之外，还有另一类错误称为逻辑错误。所谓逻辑错误，就是程序的编写符合语言的语法要求，但由于算法对问题的描述与问题不一致，造成程序不能够按照预定的要求执行，也就是程序运行的结果错误。出现这种错误就需要修改源程序的算法，修改程序后再重新进行编译，直到程序运行的结果正确为止。

根据上面所述，可得到 C 语言程序的开发过程如图 1.1 所示。

图 1.1 为 C 语言程序开发的一般过程。其中的方框表示相应的执行动作，圆圈表示上面动作执行后所得的结果。

由于 C语言可以在许多的软件平台上运行，因此，当它的运行环境不同时，上机操作的具体细节也不同。因此，建议读者在上机前要阅读有关的上机手册，了解具体的上机命令及操作过程。

为方便读者上机练习，下面给出在 UNIX 操作系统环境下和在 Turbo C环境下开发 C 程序的简要步骤。

1.3.1 在 UNIX 操作系统环境下开发 C 程序的步骤

1．编辑源文件

登录进入 UNIX操作系统，在命令提示符状态下输入 vi 命令，进入编辑状态。在编辑状态下将 C语言的源程序输入计算机，经修改确认无误后将该源程序存入文件系统，假设该源程序的文件名为 file.c。

2．编译和链接

为了利用 C编译程序对源程序文件进行编译，可在命令行上输入如下命令：

```
cc file.c (回车)
```

编译程序将开始对源程序 file.c 进行编译。如果在编译过程中发现源程序中有语法错误，则编译系统在编译后会输出错误信息，报告用户在源程序中出错的位置和错误原因。此时，用户应返回到第一步，利用编辑程序对源程序文件 file.c进行修改。改正错误并将修改后的源程序文件存盘后再重新利用编译程序对源程序文件 file.c 进行编译。如果编译过程中没有发现源程序文件有语法错误，将生成一个目标程序 file.o。然后，编译程序 cc 将自动对生成的目标程序和库函数或其他的目标程序进行连接，生成一个可执行文件，系统将其自动命名为 a.out。如果不想利用系统提供的可执行文件名 a.out，假设想将生成的可执行文件的文件名命名为 file.out ，则可以在编译时输入如下的命令：

```
cc -o file.out file.c（回车）
```

这样，生成的可执行文件就保存在文件 file.out 中了。

3．运行

在系统的命令提示符下输入生成的可执行文件的文件名就可以运行开发出的程序了。比如，输入如下命令：

```
a.out （回车）
```

或

```
file.out（回车）
```

则生成的可执行程序就开始运行了。这时，可以看一看程序运行的结果是否正确。如果不正确，则要返回到第一步，对源程序文件进行修改和编辑，然后再重复第 2 步和第 3 步，直到程序的运行结果正确为止。

1.3.2 在 Turbo C 环境下开发 C 程序的步骤

Turbo C 环境是一个集成开发环境。所谓集成开发环境，就是将源程序的编辑、编译、链接和运行都集成到一个软件中了，进行其中的任何一项工作都不必脱离 Turbo C 软件环境而运行其他的软件。因此，这种集成开发环境极大地方便了用户的使用，提高了软件的开发效率。

1．启动 Turbo C

首先，进入 Turbo C 程序所在的目录，然后，在 DOS 提示符下输入如下命令：

```
tc（回车）
```

之后，在屏幕上就会出现 Turbo C 的集成开发环境，如图 1.2 所示。

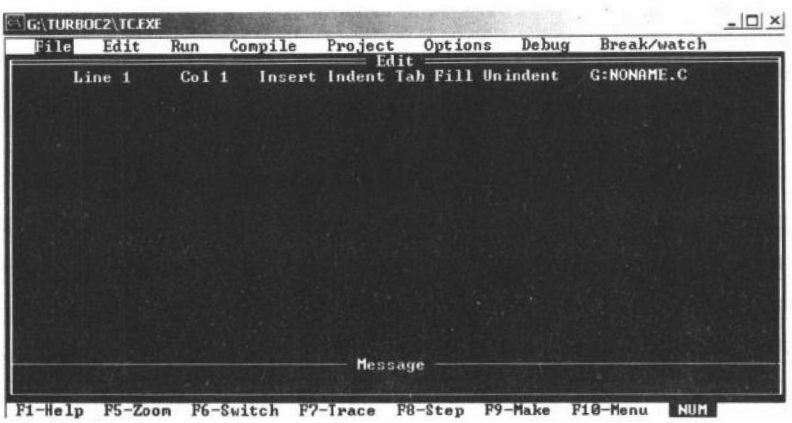

图 1.2 Turbo C 集成开发环境

在集成开发环境的最上面出现的菜单称为主菜单，共包括 8 个选项。这 8 个选项的名

称及其功能如表 1.1 所示。

表 1.1　Turbo C 主菜单选项及功能

菜单名称	主要功能
File	装入或保存文件、管理目录、调入 DOS 和退出 Turbo C
Edit	调用 Turbo C 的编辑程序
Run	编译、链接和运行装入环境下的当前程序
Compile	编译该环境下的当前程序
Project	管理多文件工程
Options	设置编译程序和链接程序的各种选项
Debug	设置各种调试选项
Break/watch	设置断点跟踪，查看内部值

其中每一个选项都有一个子菜单。限于篇幅，我们在此不作详细介绍，下面仅简要介绍有关的一些操作。

2．编辑

利用键盘上的左右箭头键移动主菜单上的光标，将光标移动到 File 菜单上，按回车键，将出现一个子菜单，如图 1.3 所示。

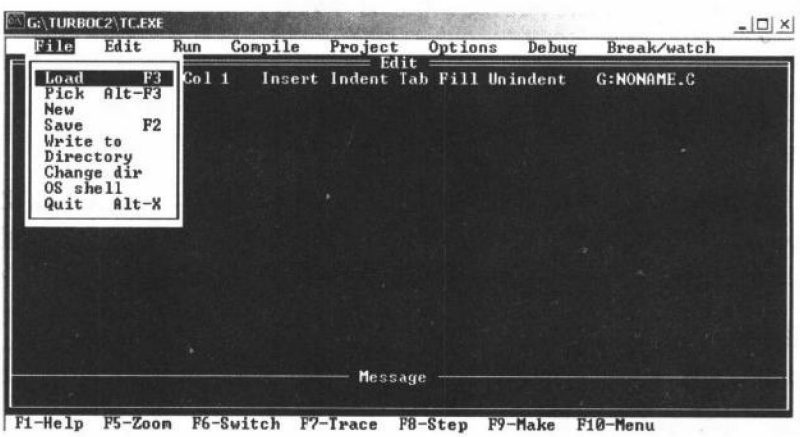

图 1.3　Turbo C 的 File 子菜单

利用上下箭头键将光标移到 Load 菜单项上按回车，就会出现一个提示框。在输入框中输入要调出修改或将要建立的源程序的文件名，然后回车。Turbo C 就会进入对输入的源文件的编辑状态。

在 Turbo C 的编辑窗口对源程序进行修改或编辑。修改或编辑完毕后按“F10”功能键，将光标移到主菜单上。打开 File 下拉式菜单，将光标移到 Save 菜单项上，按回车，则所修改和编辑的源程序就保存到盘上了。

3. 编译

按“ F9 ”功能键,或打开主菜单上的 Compile 下拉式菜单并选择 Make EXE file 菜单项，系统就会对当前正在编辑的源程序文件进行编译和链接，然后生成一个可执行文件。如果在编译和链接的过程中出现错误，则会在屏幕上显示出错信息。此时，按回车键，将在 Message 栏内显示错误信息，并将光标停留在编辑区内的源程序的错误行上。这时，可根据 Message 栏内错误提示信息对源程序进行修改，然后再重新进行编译和链接，直到没有错误为止。

4 运行

按“ F10 ”功能键，将光标移到主菜单上，选择 Run 菜单，按回车键，则开始执行已经编译好的可执行程序。并将运行结果在屏幕上显示出来。如果程序中需要输入数据，则系统会在相应的位置处停下来，等待你输入数据。当你输入完数据，程序将继续运行，并将结果显示出来。

1.4 C语言的字符集及词法约定

1.4.1 C语言的字符集

C 语言和其他的高级语言一样，有着自己的一套字符集和使用规则。这套字符集包括数字、字母、图形符号、转义序列以及三联字符序列等。

1. 数字

常用的十进制的 10个数字包括 0、1、2、3、4、5、6、7、8、9。

2. 字母

包括大写和小写的英文字母各 26 个。它们是 A、B、C、D、... X、Y、Z 和 a、b、c、d、...、x、y、z。应当注意的是，在程序中，大写字母和小写字母是有区别的，分别表示两个不同的字符。在程序中通常用的是小写的字母。

3. 图形字符

包括 29 个可见的字符，它们是：!" # % & '( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~
这些字符主要用来表示各种运算。

4. 转义序列

在源程序中往往会用到一些表示特定功能的符号，而这些特定符号是不可见的（即不可直接打印的）或难以打印的字符。比如，换行符。为了在程序中显式地表示它们，就采用转义序列的方式。比如，换行符可表示成\n，虽然看起来是两个字符（字符\和字符 n），但实际上这两个字符合起来表示一个换行符。这里的字符\是转义符,表示它后面的字符(字母）失去了原来的含义，转换成表示另外的特定意义。表 1.2 列出了 C语言中常用的一些转义序列及其含义。

表 1.2　C语言中常用的转义序列及其含义

字符名	表示形式	含义
换行	\n	把打印（显示）位置移到下一行的起始位置
水平制表	\t	把打印位置移到当前行的下一个制表点（通常是右移 8 个字符的间隔）
垂直制表	\v	把打印位置移到下一行制表点起始位置
退格	\b	把打印位置在当前行上向后退一个字符位置
回车	\r	把打印位置移到当前行的起始位置
换页	\f	把打印位置移到下一个逻辑页开头的起始位置
报警	\a	产生可听或可见的报警，位置不变
问号	\?	打印问号
单引号	\'	打印单引号
双引号	\"	打印双引号
反斜线	\\	打印反斜线

在表 1.2 中，前 7 个字符是非图形字符，主要用于控制打印位置，使输出格式整齐美观。后 4 个字符纯粹是转义序列，以便打印出转义符\之后的字符。

5．三联字符序列

这是标准 C语言中新增加的内容。在源程序中出现如下 3 个字符组成的序列（称作三联序列）都被相应的单个字符所代替，如表 1.3所示。

表 1.3　三联字符序列

三联序列	替代字符
??=	#
??(	[
??/	\
??)	]
??'	^
??<	{
??!	|
??>	}
??-	~

在标准 C语言中只定义了这 9 个三联序列。另外，不作为三联序列开头的问号（？）不被转换。例如，在源程序中的语句

```
printf("yes or no ???/n");
```

将被替换成

```
printf("yes or no ?\n");
```

引入三联序列是对 ASCII 字符集的扩展,因为 C 语言中用作运算符的一些字符在德语、法语、西班牙语等语言中用作字母扩展符。

1.4.2 C语言的词法约定

在 C语言中，基本的词法单位是单词。它主要包括 6 类：标识符、关键字、常量、字符串常量、运算符和标点符。空格符、制表符、换行符和换页符统称为空白符，主要用作分隔单词，一般没有实际意义。下面对关键字、标识符和标点符作一介绍，其他几类单词将在以后的章节中介绍。

1. 关键字

关键字是具有特定含义、专门用作语言的特定成分的一类单词，不能用于其他场合。标准 C语言中共有 32 个关键字，按其用途大致可分为 4 类：数据类型、存储类、流程控制和运算符。表 1.4列出了全部关键字及其用途。

表 1.4 C语言的关键字及其用途

类别	关键字	用途
数据类型	char	字符量
	int	整型量
	long	长整型量
	short	短整型量
	float	单精度浮点量
	double	双精度符点量
	unsigned	无符号量（最高位不作为符号位）
	signed	有符号量（最高位作为符号位）
	struct	结构型量
	union	联合型量
	enum	枚举型量
	void	无值量
	const	常量
	volatile	易变量
存储类	auto	自动量
	extern	外部量
	static	静态量
	register	寄存器量
	typedef	类型命名
流程控制	if	条件语句
	else	条件语句的另一种选择
	for	for循环语句
	while	while和 do-while 循环语句

（续表）

类别	关键字	用途
流程控制	do	do-while循环语句
	break	间断语句
	continue	接续语句
	goto	跳转语句
	return	返回语句
	switch	多路选择语句
	default	多路选择语句中的默认情况
	case	多路选择语句中的情况选择
运算符	sizeof	计算字节数

C语言的关键字都是小写的，所以 int 是关键字，而 INT 不作为关键字。关键字都有专门用途，不能把它们用作普通变量名或函数名。在某些已经实现的C语言的编译程序中还出现了另外一些关键字，如 asm，fortran 等，用于表示特定语言的名称，这些都与具体实现有关。

2. 标识符

标识符是一个有效的字符序列，用来表示变量名、符号常量名、数据类型名和函数名等。在C语言中的标识符必须符合下列语法规则：

（1）标识符的第一个字符必须是字母（大小写均可）或下划线（_）；

（2）在第一个字符之后，可以是任意字母、下划线或数字组成的字符序列，也可以是空串。

根据上述两条规则，下列字符序列是合法的标识符：

```
a1,sum,_id,Total,num_5,f,average
```

而下面的字符序列不是合法的标识符：

5th　　（打头的是数字而不是字母）
!yan　　（打头的字符不是字母或下划线）
ptr*　　（字符*不是字母、下划线和数字）
LINE 1　　（中间有一个空格）

在程序设计中选择标识符时，除了必须遵循上述规定以外，还应注意下面几点：

（1）在选择变量名、函数名或其他的标识符时，应选用有相应含义的英文单词（或者缩写）作为标识符，如 sum，name，count 和 number等。这样，一见到标识符就可想象出它所代表的数据的意义，从而增加程序的可读性。当然，如果所编写的程序较短，或者在一个程序中数据的个数不多，或者某个标识符在程序中使用的次数很少，也可用代数符号（如 a，b，c，x和 y 等）作为变量名。在本书的有些示例中，由于程序较短，算法也比较