第 1 章　C 语言介绍

本章学习要点：

(1) C 程序文件的组成。

(2) 编写及运行 C 程序。

(3) C 程序的书写规范和编程风格。

1.1　C 语言概述

1.1.1　C 语言的起源和 C 语言标准

C 语言是 20 世纪 70 年代初在贝尔实验室为了编写操作系统及其他系统软件而产生的一种具有自身基础理论体系的编程语言。

由于 C 语言具有的一个非常重要的优点是可移植性，因而得到了广泛流行和持续发展。C 语言的快速发展使不同的版本相继出现，这些版本互不兼容，因此 C 语言的标准亟待建立。1983 年，美国国家标准协会(American National Standards Institute，ANSI)制定 C 语言标准。1989 年 12 月，ANSI 批准了第一个 C 语言标准。1990 年，该标准被国际标准化组织(International Standard Organization，ISO)批准，这一版本的 C 语言标准被称为 C89 或 C90 标准。1999 年，C 语言标准委员会将 C++ /Java 语言的一些特性加到 C 语言中，以此提高 C 语言的性能，产生了 C 语言的 1999 标准，也称 C99 标准。

1.1.2　C 语言的优缺点

C 语言具有以下主要特点。

- C 语言是一种小型语言。C 语言提供了一套有限的特性集合，为了使特性较少，C 语言不直接提供输入和输出语句、有关文件操作的语句和动态内存管理的语句等操作，而是由编译系统所提供的库函数来实现的。
- C 语言是一种包容性语言。C 语言语法限制不太严格，不像其他语言一样强制进行详细的错误检查，因此程序设计自由度较大，但对程序员的要求也更高。
- C 语言是一种底层语言。C 语言提供机器级概念的访问，能实现汇编语言的大部分功能，可以直接对硬件进行操作，可用来编写系统软件。因此，C 语言既是通用的程序设计语言，又是成功的系统描述语言。

1. C 语言的优点

- 高效。C 语言的产生是为了编写系统程序，其优点是能够生产高质量的目标代码，使其在有限的内存空间中快速运行，因而 C 语言程序的执行效率高。
- 可移植。C 语言编译器规模小、容易编写，因而容易移植到新系统中，几乎在所有计算机系统中都可以使用 C 语言。
- 功能强大。C 语言拥有丰富的运算符和数据类型，从而使 C 语言具有丰富的运算类型、多样化的表达式类型，能够实现各种复杂的数据结构运算。
- 灵活。C 语言的特点使其编程轻松、自由，使用上的限制非常少。例如，可以使用 C 语言编写从嵌入式系统到各种应用程序；C 语言允许字符与整数或浮点数相加。
- 标准库。C 语言的一个突出优点就是它具有标准库，该标准库包含了数百个可以用于输入/输出、字符串处理、存储分配以及其他实用操作的函数。
- 结构化。C 语言具有结构化的控制语句，例如，if-else 和 switch 选择语句，while、do-while 及 for 循环语句。C 语言用函数作为程序的模块单位，得以实现程序的模块化和结构化。

2. C 语言的缺点

和它的许多优点一样，C 语言的缺点源于它与机器的紧密结合性。

- C 程序更容易隐藏错误。C 语言的特点使用 C 语言编程的出错概率较高，包含很多觉察不到的隐患。例如，一个额外的分号可能会导致无限循环，遗漏一个地址号(&)可能会引发程序崩溃。
- C 程序可能会难以理解。C 程序简要、灵活的特点会使程序员编写的程序让其他程序员读不懂。C 语言有许多其他通用语言没有的特性，且这些特性可以以多种方式结合使用，其中的一些结合方式则会使其他程序员不理解。
- C 程序可能会难以修改。现代编程语言往往提供了“类”“包”之类的结构，使大的程序可以分解成很多模块进行管理。但如果用 C 语言编写大规模的程序则将会使程序难以阅读、修改和管理。

1.1.3 如何使用 C 语言

C 语言既有优点又有缺点，那么，在使用时需要有效利用 C 语言的优点并尽量避免它的缺点。使用 C 语言时可注意以下几点。

- 规避 C 语言的缺点。现在没有一个编译器可以检查出编程语言的所有缺陷。程序员需要通过阅读参考书籍和文献，并不断积累经验等，才能掌握 C 语言的特点，避免其缺陷。
- 使程序更可靠。C 程序员可以通过使用软件、调试工具等方式对程序进行广泛的错误检查、分析和纠错。
- 利用现有的代码库。函数的集合经常被打包成库，使用这些库既可极大地减少错误，又可节省大量的编程时间。这些库有公用、开源、以商品销售等形式。

- 具有编程规范。程序员应当从一开始就遵守并坚持使用符合 C 语言程序准则的、规范化的编程风格，编码规范可以使程序统一，且易于阅读、修改和维护。
- 避免极度复杂和无节制简洁。极其复杂的编程和没有节制的简略都会导致编写的程序难以理解，因而我们提倡简洁但易于理解的编程风格。
- 符合标准。为了提高程序的可移植性，尽量避免使用不符合标准的库函数、特性等。

1.2　C 程序文件

一个 C 程序的结构如图 1-1 所示，它可以由一个或多个源程序文件（简称源文件）组成，通常还有一些头文件（header file）。源文件由预处理指令、数据、函数的声明以及函数组成；函数由函数首部和函数体构成；函数体由数据声明和执行语句构成。

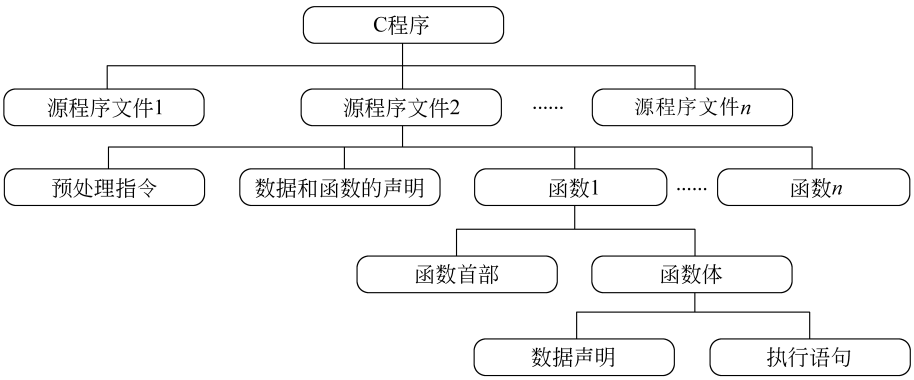

图 1-1　C 程序的结构

1.2.1　源文件

源文件包含预处理指令（如 #include、#define 等）、变量声明、函数等，源文件的扩展名为 c.。一个函数包括函数首部和函数体，函数体包含声明部分和执行部分，声明部分是对有关数据的声明，执行部分是由语句组成的。一个 C 语句经过编译后产生若干条机器指令，可以向计算机系统发出操作指令。多个源文件中必须有一个源文件包含一个名为 main 的主函数，主函数是程序的起始点。

一个 C 程序可以由多个源文件组成，把程序分成多个源文件的优点如下。

- 把相关的函数和变量分组放在同一个文件中，可以使程序的结构清晰。
- 可以分别对每一个源文件进行编译。如果程序规模很大而且需要频繁改变，分成多个源文件的方法可以极大地节约程序运行时间。
- 把函数分组放在不同的源文件中利于再次使用。

1.2.2 头文件

头文件包含了源文件之间共享的信息，如符号常量定义、类型定义、函数原型、变量声明等，头文件的扩展名为.h。这些信息通过预处理指令 # include 实现了共享。预处理指令 # include 指示预处理器打开指定的头文件，并且把此文件的内容插入当前源文件中，这样几个源文件就可以访问相同的信息。也就是说，共享的信息放在一个头文件中，然后在每个源文件中用“# include <共享头文件>”指令就可以把该头文件的内容插入源文件恰当的位置中。源文件指程序员编写的全部文件，包括.c 文件和.h 文件。本书中的“源文件”通常仅指.c 文件。

1.2.3 把程序划分成多个文件

现在应用我们已经知道的关于头文件和源文件的知识来开发一种把一个程序划分成多个文件的简单方法。这里将集中讨论函数，但是同样的规则也适用于外部变量。假设已经设计好程序，换句话说，已经决定程序需要什么函数以及如何把函数分为逻辑相关的组。下面是处理的方法。

- 把每个函数集合放入一个不同的源文件中（比如用名字 foo.c 来表示一个这样的文件）。
- 创建和源文件同名的头文件，只是扩展名为.h（在此例中，头文件是 foo.h）。
- 在头文件中放置源文件中定义的函数原型。
- 每个需要调用定义在源文件中的函数的源文件都应包含头文件。
- 源文件也应包含头文件，这是为了编译器可以检查头文件中的函数原型是否与源文件中的函数定义相一致。
- main 函数将出现在另一个源文件中。main 函数所在的文件中也可以有其他函数，前提是程序中的其他文件不会调用这些函数。

1.3 运行 C 程序

1.3.1 C 程序的运行步骤

运行一个 C 语言程序包含以下几个步骤。

（1）编辑。在计算机通过编辑器创建一个后缀为 .c 的 C 语言源程序。

（2）预处理。源程序被送交给预处理器（preprocessor），预处理器执行以 # 开头的指令。预处理器是一个小软件，它可以在编译前处理 C 程序，可以给程序添加内容及进行修改。预处理器的行为由 # 字符开头的预处理指令控制，如 # define、# include。

（3）编译。经过预处理后的程序进入编译器（compiler），编译器把程序翻译成机器能够识别的目标代码，生成后缀为.obj 的目标文件。然而，这时的目标文件还不可以运行。

(4) 连接。连接器(linker)把由编译器产生的目标代码和其他所需要的附加代码整合在一起,这样才最终产生了后缀为.exe 的可执行文件。这些附加代码包括程序中用到的库函数(如 printf 函数)。

(5) 运行。C 程序经过编辑、预处理、编译、连接几个自动完成的步骤后即可运行,并输出运行结果。

根据编译器和操作系统的不同,编译和连接所需的命令是多种多样的,生成的文件后缀也不同。

1.3.2　集成开发环境

C 程序可以通过在操作系统提供的窗口中输入命令的方式来调用命令行编译器,也可以使用集成开发环境(integrated development environment, IDE)进行编译。集成开发环境是一个应用程序,用户可以在其中编辑、编译、连接、执行、调试程序,组成集成开发环境的各个部分可以协调工作。集成开发环境有很多种,这里不作一一介绍。下面给出在 Visual C++ 6.0 集成环境下的上机操作方法。

(1) 安装 Visual C++ 6.0。在 Visual Studio 光盘运行安装文件 setup.exe,按提示一步步进行操作即可安装成功。

(2) 启动 Visual C++ 6.0。从桌面上选择“开始”→“程序”→Microsoft Visual Studio→Visual C++ 6.0 命令,显示程序主界面。

(3) 建立项目(可选)。在主界面选择 File→New→Projects→Win32 Console Application 命令,在 Project name 中输入对该项目的命名,右侧中间的单选按钮默认选定 Add to current workspace,单击 OK 按钮。在弹出的对话框中选择右侧的 An empty project 单选按钮,单击 Finish 按钮。

(4) 编辑源程序。在主界面中选择 File→New 命令,此时出现一个 New 对话框,选择该对话框左上角的 Files 选项卡,选择其中的 C++ Source File 选项(Visual C++ 6.0 既可处理 C++ 源程序,也可处理 C 源程序),然后在对话框右半部分的 Location 对话框中输入存放该文件的路径和文件夹,在右上方的 File 对话框中输入该源程序的命名后,单击 OK 按钮,这样即将进行编辑的源程序的文件名和路径就创建好了。随后,在编辑窗口打开源程序文件,开始编辑源程序代码即可。编辑完毕,单击 Save 按钮,便在默认路径和文件夹中建立了一个 C 语言源程序文件。

(5) 编译和连接程序。在主界面菜单栏中选择 Build→Compile 命令,系统则开始对当前程序进行编译,若在编译过程发现错误,则将在下方窗格中列出所有错误和警告。修改错误并重新编译,直至修改完所有错误为止。选择 Build→Build 命令,则会在文件夹中生成该源程序的可执行文件,完成编译和连接。

(6) 运行程序。选择 Build→Execute 命令,即开始运行该源程序的可执行文件,执行完毕,输出结果的窗格将显示运行结果。

1.4 编写程序

1.4.1 程序设计的任务

程序设计是针对要解决的问题进行从确定任务直至得到结果、写出文档的整个过程，一般需经历以下几个工作阶段。

(1) 分析问题。对于要解决的问题进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题的方法，这也被称为建立模型。

(2) 设计算法。设计出解决问题的方法，并将其具体步骤清晰地描述出来。例如，要解一个方程式，需要了解应选择什么方法，以及使用该方法求解方程式的每一个步骤是什么。

(3) 编写程序。根据所设计的算法，用高级编程语言编写源程序。

(4) 运行程序。编译系统对源程序进行编辑、编译、连接，然后运行程序，得到程序的运行结果。

(5) 分析结果。对程序运行结果进行分析，判断结果是否正确、合理。如果不正确或不合理，则对程序进行调试和测试。调试(debug)是发现程序错误和排除程序故障的过程。测试(test)是使用多组数据运行程序，尽最大可能发现程序漏洞，并修改程序以便弥补漏洞，使程序能适用于各种情况。

(6) 编写程序文档。程序文档即为程序说明书或用户文档，是软件的重要组成部分，其内容包括程序名称、程序功能、运行环境、程序的装入和启动、需要输入的数据、使用注意事项等。现在商品软件的程序说明书有的以 readme 文件形式提供，有的在程序的帮助下提供，以便于用户阅读和使用程序。

1.4.2 计算机算法

计算机为了解决某个问题而采用的方法和具体步骤称为计算机算法。计算机算法可分为两大类：数值运算算法和非数值运算算法。数值运算算法是求数值解，如求方程的解、求定积分等。非数值运算算法是除数值运算法以外的解决问题的方法和步骤，如解决图书检索、人事管理、车辆调度等问题的算法。

计算机算法的表示方法有自然语言、流程图、伪代码。

(1) 自然语言就是人们日常使用的语言，可以是汉语、英语或其他语言。用自然语言表示计算机算法就通俗易懂，但文字冗长，容易出现歧义。

(2) 流程图是用一些图形来表示各种不同的操作。流程图法又可分为传统流程图和 N-S 流程图法(即结构化流程图)。传统流程图用流程线指出各框的执行顺序，对流程线的使用没有严格限制，因而会导致传统流程图变得没有规律，难以阅读和修改，使算法的可靠性和可维护性难以保证。N-S 流程图将全部算法写在一个由一些基本的矩形框所组成的一个大的矩形框中，这种流程图适用于结构化程序设计。用流程图表示的算法直观形象、易于理解，但是占用篇幅较多，尤其是算法比较复杂时，用流程图描述算法会既费时又困难。

(3) 伪代码是介于自然语言和编程语言之间的文字和符号。用伪代码表示的算法格式紧凑、修改方便、易于阅读,便于向编写程序过渡。

1.4.3　结构化算法或程序

1966 年,Bohra 和 Jacopini 提出了以下 3 种程序的基本结构,可用这 3 种基本结构作为表示一个良好算法或程序的基本单元。

1. 顺序结构

如图 1-2 所示,虚线框内是一个顺序结构。其中 A 和 B 两个框是顺序执行的。即在执行完 A 框所指定的操作后,必然接着执行 B 框所指定的操作。顺序结构是最简单的一种基本结构。

2. 选择结构

选择结构又称选取结构或分支结构,如图 1-3 所示。虚线框内是一个选择结构。此结构中必包含一个判断框。根据给定的条件 c 是否成立来选择执行 A 框或 B 框。例如,条件 c 可以是“x＞＝0”或“x＞y,a＋b＜c＋d”等。

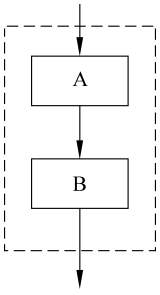

图 1-2　顺序结构

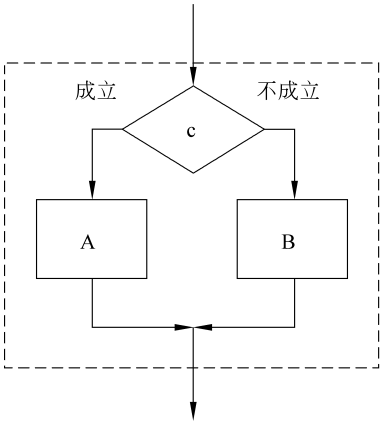

图 1-3　选择结构

3. 循环结构

循环结构又称为重复结构,即反复执行某一部分的操作。有两类循环结构。

(1) 当型(while 型)循环结构。while 型循环结构如图 1-4(a)所示。它的作用是当给定的条件 c1 成立时,执行 A 框操作;执行完 A 后,再判断条件 c1 是否成立,如果仍然成立,再执行 A 框;如此反复执行 A 框,直到某一次 c1 条件不成立为止,此时不执行 A 框,而脱离该循环结构。

(2) 直到型(until 型)循环结构。until 型循环结构如图 1-4(b)所示。它的作用是先执行 A 框;然后判断给定的 c2 条件是否成立,如果 p2 条件不成立,则再执行 A;接着对 c2 条

件作判断，如果 c2 条件仍然不成立，又执行 A；如此反复执行 A，直到给定的 c2 条件成立为止，此时不再执行 A，而脱离该循环结构。

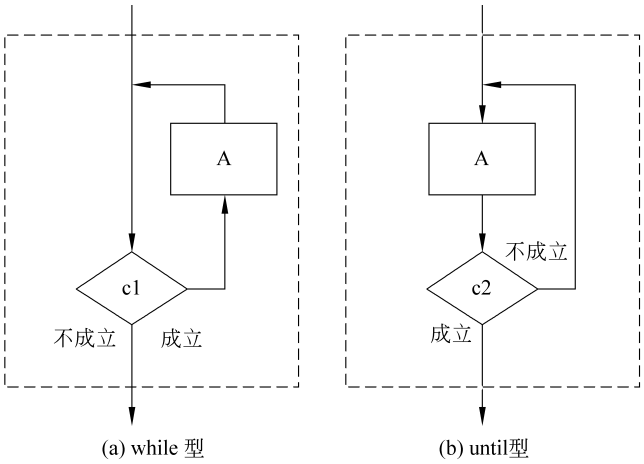

图 1-4　循环结构

以上 3 种基本结构，有以下共同特点。

(1) 只有一个入口。

(2) 只有一个出口。

(3) 结构内的每一部分都有机会被执行到。

(4) 结构内不存在“死循环”(无终止的循环)。

由以上 3 种基本结构构成的算法或程序是结构化的算法或程序，可以解决任何复杂的问题，它不存在无规律的转向，只在本基本结构内才允许存在分支和向前或向后的跳转。

基本结构并不只限于上面 3 种，具有上述 4 个特点的都可以作为基本结构，并可由这些基本结构组成结构化算法或程序。

结构化程序设计是把一个复杂问题的求解过程分阶段进行，每个阶段处理的问题都控制在人们容易理解和处理的范围内。结构化程序设计的基本思想是：强调程序设计风格和算法/程序结构的规范化，提倡清晰的结构。采取以下方法能够得到结构化的程序：自顶向下；逐步细化；模块化设计；结构化编码。结构化程序便于编写、阅读、修改、维护，减少了程序的出错率，提高了程序的可靠性，保证了程序的质量。

例 1-1　写出求解 10!的传统流程图、N-S 图、伪代码三种算法以及 C 程序。

(1) 传统流程图算法如图 1-5 所示。

(2) N-S 图算法如图 1-6 所示。

(3) 伪代码算法。

```
begin
  1 => f
  2 => i
  while i <= 10;
  {
    f * i => f
```

```
    i + 1 => i
  }
  print f
end
```

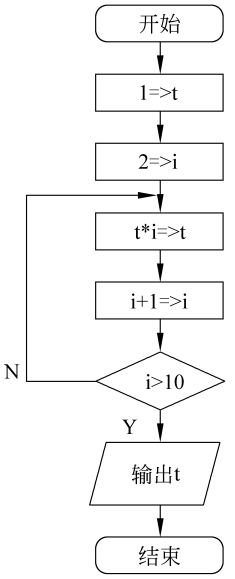

图 1-5　例 1-1 的传统流程图算法

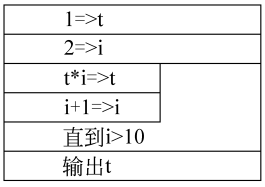

图 1-6　例 1-1 的 N-S 图算法

(4) 编写 C 程序(e1-1.c)。

```
#include <stdio.h>
int main ( )
{
  int i, f;
  f=1;
  i=2;
  while (i<=10)
  {
    f = f * i;
    i = i + 1;
  }
  printf ( "%d\n", f);
  return 0;
}
```

运行结果如下：

```
3628800
```

1.5 C 程序的书写规范和编程风格

1.5.1 书写规范

C 语言允许在记号之间插入任意数量的空格符、换行符，添加空格和空行可以使程序更便于阅读和理解。具体特点说明如下。

- 语句可以分开放在任意多行内。例如，下面的语句如果放在一行太长，可以分成两行。

```
printf ("Dirnensional weight (pounds): %d\n",
(volume+ INCHES_ PER_POUND - 1) I INCHES_PER_POUND);
```

- 记号间的空格使我们更容易区分记号。因此，在每个运算符的前后、每个逗号后边、在圆括号和其他标点符号的两边都可以放个空格，例如：

```
volume = height * length * width;
```

- 缩进有助于轻松识别程序嵌套。例如，为了清晰地表示声明和语句都嵌套某个函数中，应该对它们进行缩进。
- 空行可以把程序划分成逻辑单元，从而使读者更容易辨别程序的结构。

1.5.2 编程风格

良好的编程风格能够提高程序的阅读、调试、测试与维护，促进项目团队的合作与交流，因而我们提倡学习编程的开始就养成良好的编程习惯和编程风格。

(1) 按照 C 程序的基本结构合理安排各部分的位置。C 程序各部分的一般顺序依次为文件信息部分、连接部分、全局声明部分、main 函数部分、自定义函数部分。

(2) 标识符的命名便于理解和记忆。用来命名变量、符号常量、函数、数组、类型等的字符称为标识符。C 语言规定标识符只能由字母、数字、下画线 3 种字符组成，且第 1 个字符不能为数字。

(3) 使用缩进、括号等使程序更便于阅读和维护。

(4) 恰当地加入注释，不仅能增强程序的可读性，而且有助于理解程序的逻辑。

实　　验

1. 实验目的

(1) 学习安装和打开 C 程序集成环境软件及其基本操作。