第 1 章 C 语言概述

C 语言是一种通用的程序设计语言，它具有丰富的运算符和表达式，以及先进的控制结构和数据结构。C 语言具有表达能力强、编译目标文件质量高、语言简单灵活、容易移植及容易实现等优点。

1.1 程序与程序设计语言

1.1.1 程序

随着计算机走入寻常百姓家，“程序”已经不再是计算机科学使用的专用词汇了。在日常生活中，我们其实在不断地编写程序并执行，只不过人们并没有明确地意识到而已。举个例子，我们现在要用全自动洗衣机洗衣服，应该怎么做呢？尽管简单，我们还是按照一般人的习惯来描述一下吧。

第一步，就是要把脏衣服扔进洗衣机；

第二步，打开上水的水龙头并安装好电源插头；

第三步，放入洗衣粉；

第四步，按下洗衣机的开始按钮；

第五步，等待洗衣机洗完衣服（当然，不妨去干点什么别的事情）。在洗衣机提示洗完的蜂鸣声响了以后，就可以从洗衣机中拿出干净衣服去晾晒了。

上面所描述的五个步骤，就是人们洗衣服的“程序”。也许不同的人使用的步骤并不完全一样，例如将第一步和第二步互换一下，也同样能将衣服洗干净，所以干一件事的“程序”可以不惟一，这也是计算机程序的一个特点。

对于计算机来说，程序就是由计算机指令构成的序列。计算机按照程序中的指令逐条执行，就可以完成相应的操作。更准确一点，计算机执行由指令构成的程序，对提供的数据进行操作。计算机程序的操作对象是“数据”。这里的数据不是简单的阿拉伯数字，而是包括了各种现代计算机能够处理的字符、数字、声音、图像等。

实际上计算机自己不会做任何工作，它所做的工作都是由人们事先编好的程序来控制的。程序需要人来编写，使用的工具就是程序设计语言。

1.1.2 程序设计语言

目前，通用的计算机还不能识别自然语言，而只能识别特定的计算机语言。

计算机语言一般分为低级语言和高级语言。

低级语言直接依赖计算机硬件，不同的机型所使用的低级语言是完全不一样的。高级语言则不再依赖计算机硬件，用高级语言编写的程序可以方便地、几乎不加修改地用在不同类型的计算机上。

需要强调的是，无论采用何种语言来编写程序，程序在计算机上的执行都是由 CPU 所提供的机器指令来完成的。机器指令是用二进制表示的指令集。每种类型的 CPU 都有与之对应的指令集。

1. 低级语言

低级语言包括机器语言和汇编语言。

直接使用二进制表示的指令来编程的语言就是机器语言。使用机器语言编写程序时，必须准确无误地牢记每一条指令的二进制编码，才能编写程序。如果程序员面对的是“101110001110100000000011”这样的编码序列，能不头痛吗？而且，有时还要求把这些二进制编码再转换成八进制或十六进制数才能输入计算机，这不但加大了程序员的工作量，而且还增加了程序出错的机会，将大量的二进制编码序列准确地转换成八进制或十六进制数，可不是一件容易的事。

机器语言的优点是执行速度快，并且可以直接对硬件进行操作，例如主板上的 BIOS 及一些设备的驱动程序等。

机器语言的缺点也是显而易见的。首先是可读性差，就是编写程序语句“101110001110100000000011”的人也未必马上就能看懂该句表示的是什么命令；其次，是可维护性差，别的程序员编写的程序（甚至是程序员自己编写的）很难看懂，如何谈维护呢？

再者，就是可移植性差，因为不同的机型有自己的一套机器指令，与其他机型的机器指令不兼容。另外，用机器语言编写程序的生产效率低下，并且不能保证程序有好的质量。

为了能够更方便地编写程序，人们用一些符号和简单的语法来表示机器指令，这就是汇编语言。例如，“101110001110100000000011”用汇编语言表示就是“mov ax,1000”，该指令的功能是“将 1000 送入寄存器 AX 中”，是不是清楚多了？但是 CPU 并不能识别汇编语言，因此，需要一个“翻译”程序将汇编语言翻译成机器语言，我们把这种将汇编语言翻译成机器语言的程序叫做“汇编器”。汇编语言与机器语言的指令是一一对应的，所以，除了提高了一些可读性，汇编语言从根本上并没有改变机器语言的特点。可以说，汇编语言是面向机器语言的。当然，汇编语言也仍然具备机器语言的优点。许多大型系统（例如操作系统）的核心部分都是用汇编语言写的，因为这部分工作需要很高的效率，直接和硬件打交道。

那么，有没有办法真正提高程序的可读性、可维护性和可移植性呢？回答是肯定的，就是使用高级语言。

2. 高级语言

高级语言是一种比较接近自然语言和数学语言的程序设计语言。高级语言的出现大大提高了程序员的工作效率，降低了程序设计的难度，并改善了程序的质量。用高级语言编写的程序看起来更像是英语，很容易读懂，不但使程序具备良好的可读性和可维护性，而且使更多的人掌握了程序设计方法，从而使计算机技术得到迅速的应用和普及。

例如，

语句段

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

表示的是“如果 a 大于 b，则 c=a，否则 c=b”。是不是很容易理解？当然，要注意，这里的“ = ”与数学语言等号是有根本的区别的，我们将在介绍 C 语言的运算符时，详细地加以讨论。

另外，用高级语言编写的程序还具有很高的可移植性。从高级语言到机器语言要经过编译程序进行“翻译”，而高级语言几乎为每一种机器都创建了各自的编译程序，从而可以将用高级语言编写的程序几乎不加修改地运行在不同的计算机平台上。

编译程序分为两种，一种是解释系统，另一种是编译系统。解释系统是对高级语言编写的程序翻译一句执行一句；而编译系统是将高级语言编写的程序文件全部翻译成机器语言，生成可执行文件以后再执行。高级语言几乎在每一种机器上都有自己的编译程序。 C 语言的编译程序属于编译系统。

1.2 C 语言发展概述和主要特点

1.2.1 C 语言的发展历史

C 语言与 UNIX 操作系统有着密切的关系，它的发明者是 Dennis Ritchie，Dennis Ritchie 开发 C 语言的主要目的是为了更好地描述 UNIX 操作系统。

1969 年，美国贝尔实验室的 Ken Thompson 在一台报废的 DEC PDP-7 上做了一些程序来辅助软件开发。

1969 年~1972 年，Ken Thompson 与 Dennis Ritchie 合作，用了不到两年的时间就把这些程序发展为一个操作系统——UNIX。

在此期间，早期的 UNIX 是用汇编语言写的，我们已经在前面谈到了汇编语言的缺点，Thompson 为了摆脱汇编语言的困扰，在 1970 年决定开发一种高级语言以便更有效地描述 UNIX，他以 BCPL 语言为基础开发了一种新的语言——B 语言。但 B 缺乏丰富的数据类型，又以字长编址，有一定的缺陷，因而未能流行起来。为了改进“ B ”语言，从 1971 年开始，D.Ritchie 用了一年左右的时间，在 B 语言的基础上加入了丰富的数据类型和强有力的数据结构，从而形成了 C。之所以命名为 C 是因为：“BCPL ”语言是 B 的先驱，“BCPL ’这串字符中 C 字符在 B 字符的后面；同时，按 26 个字母的顺序（ABCD……），B 字符后面的也是 C。

1978 年，Brian W.Kernighan 、Ken Thompson 与 Dennis Ritchie 三人合作，写了一本著名的书《The C Programming Language》，该书介绍的 C 语言被称为标准 C。

而后，C 语言由于其本身的优点，先后被移植到各种计算机平台上，得到了广泛的使用。同时，也出现了很多的编译系统版本。

1983 年，美国国家标准化协会（ANSI）建立了一个委员会，着手制订 ANSI 的标准 C。

1988 年，ANSI 公布了标准 ANSI C。这个标准的大部分特性已经由现代的编译系统所支持。

1989 年，国际标准化组织（ISO）也采用了 ANSI C 标准，称 ANSI/ISO standard C。

1994 年，ISO 修订了标准，称 ISO C。

本书将以 ANSI C 为基础讲解 C 语言。

1.2.2 C 语言的主要特点

也许，我们在此讨论 C 语言的主要特点，并不能给读者带来切身的感受，但是，希望读者能从下面的叙述中对 C语言的特点有一个基本的了解，并在随后的学习中不断加深对它们的理解。

首先，我们来看 C语言具备的优点。

如果用最简单的语言来概括，C 语言的优点是：简洁、灵活、表达能力强、产生的目标代码质量高、可移植性好。具体讲，有以下几点：

（1）C 语言同时具备了高级语言和低级语言的特征。高级语言应该具备的优点 C 语言都有，例如可读性好、容易记忆、可移植性强等，同时 C语言还提供了某些接近于汇编程序的功能，如地址处理、二进制位运算以及指定用寄存器存放变量等。因此，有人认为 C 语言是中级语言。C 语言适合编写系统程序和各种软件工具。

（2）C 语言是结构化程序设计语言，具有结构化程序设计所要求的控制语句，如条件语句 if，循环语句 for、while、do while 等。

（3）C 语言支持模块化程序设计。 C 语言的程序是由函数构成的，每个函数可以单独编写和调试。因此，遇到大型程序，程序员们可以分别编写不同的模块，这使得管理和调试工作变得简单和方便，并且可以实现软件重用，即重复使用那些经常需要使用的程序模块。

（4）C语言具有丰富的数据类型。C 语言支持各种高级语言普遍使用的基本数据类型，并允许用基本数据类型构造复杂数据类型。

（5）C 语言的运算符种类多、功能强大。

（6）C 语言的基本组成部分紧凑、简洁，关键字少。

（7）C 语言有大量的标准化的库函数。这些库函数不但包括了各种数学计算的函数，还有用于输入输出的库函数以及系统函数，给程序员编写程序带来了极大的方便。

（8）生成代码质量高。C 语言与汇编语言生成的代码相比，前者只比后者低 10%~20%。

（9）特别良好的可移植性，应用性广。可在许多软件平台和硬件平台上应用。

任何事物都不是十全十美的，C 语言也有一定的缺陷，了解 C 语言的缺点，有助于我们在编写程序的时候扬长避短。具体讲，有以下两点：

（1）C语言比较灵活，在语法上不如一些著名的高级语言（例如 Pascal，Ada ）严格，错误检查系统不够坚固。例如，有些语句用在 Pascal 程序中，会被 Pascal 编译程序指出有语法错误，但类似的语句用在 C程序中，会轻而易举地通过 C 编译系统，因此给程序的调试带来困难，尤其是对初学者。

（2）如果不加以特别的注意，C 程序的安全性将会降低。例如对指针的使用没有适当的限制，指针设置错误，可能引起内存中的信息被破坏，如果经常出现这种错误，极有可能导致系统的崩溃。

1.3 C语言的基本结构

本节，我们将通过几个简单的程序例子，使读者对C程序的组成有个感性的认识。

1.3.1 第一个程序

【例 1.1】在屏幕上显示 welcome字样。

```
/*-------A program to print welcome--------*/
#include "stdio.h"
void main()
{
   printf("welcome");
}
```

这是一个最简单的C程序。尽管简单，但是已经充分说明了C程序的基本组成。该程序包括了三部分：注释、预处理命令及函数定义。下面是对例 1.1 的分析与说明。

（1）程序开始用 /* 和 */ 括起来的是注释行。注释行用于说明程序的功能和目的，编译系统会跳过注释行，不对其进行翻译。如果想做一个好的程序员，必须习惯为程序写出详细的注释。按照惯例，一般要在程序的最开始说明整个程序的目的和功能，并在必要时，为每一组代码写出注释，以增加可读性。

使用 /* 和 *／括起来的语句并不一定在一行，可以是多行。

例如，可将例 1.1 中的注释语句写成：

```
/*-----------------------------------------
A program to print welcome----------------------*/
```

（2）以＃开始的语句是预处理命令。这些命令是在编译系统翻译代码之前需要由预处理程序处理的语句。本例中的#include "stdio.h" 语句是请求预处理程序将文件 stdio.h包含到程序中来，作为程序的一部分。文件 stdio.h 中是一些重要的定义，没有它，printf("welcome"); 语句不能通过编译系统的“翻译”。

（3）每个 C 程序都必须包含一个主函数 main()，也只能包含一个主函数。用{}括起来的部分是一个程序模块，在C语言中也称为分程序，每个函数中都至少有一个分程序。C程序的执行是从主函数中的第一句开始，到主函数中的最后一句结束。

（4）分 号“ ；”是C语言的执行语句和说明语句的结束符。

（5）C 语句在书写上采用自由格式。书写 C语句时不含行号，不硬性规定从某列开始书写,但是好的程序员应该学会使用缩进格式,例如 printf("welcome");语句在 main 函数内部，书写时不能与 main对齐，而是向右移动了几个格。

（6）C 语言的关键字和特定字使用小写字母。 main 是关键字，include 是特定字，都必须用小写。

（7）printf 是 C 语言提供的标准输入输出库函数，它的功能是将用两个双引号括起来的内容 welcome输出到标准输入输出设备显示器上。

因此例 1.1 的运行结果是：

```
welcome
```

1.3.2 第二个程序

【例 1.2】计算 a+b，并在屏幕上显示结果。

```
/*------------sum of a add b -----------------*/
#include "stdio.h"
void main()
{
    int a,b,sum;
    a=1;
    b=2;
    sum=a+b;
    printf(" a add b is %d \n",sum);
}
```

运行结果：

```
a add b is 3
```

分析与说明：

（1）变量的数据类型定义。变量是由程序命名的一块计算机内存区域，用来存储一个可以变化的数值。每个变量保存的是一个特定的数据类型的数值，例如整型、字符型。 int a,b,sum; 定义了三个存储空间，分别命名为 a 、b 和 sum，这三个存储空间的数据类型为整型(int)，int 是类型说明符。在 C 语言中规定，任何变量都要经过数据类型的定义，以便在程序运行时分配相应的存储空间。

（2）直接常量（又称无名常量或文字常量）。常量是在程序执行过程中不会变化的数值，直接常量就是在代码中直接书写的数值，没有名字。例如 a=1;语句中的 1 和 b=2;中的 2。

（3）赋值运算符 =。注意，这里的 = 与数学上的等号在概念上完全不同。赋值运算符最简单的用法是：赋值运算符的左边是一个变量，右边是一个常量。其功能是将右边常量的值送到左边的变量中，使变量中的内容与常量相等。例如 a=1;就表示使 a 中的内容变为 1。

（4）运算符 + C 语言的算术运算符与数学符号很相像，sum=a+b;表示将 a 的内容与 b 的内容相加以后，赋值到 sum 变量中。

1.3.3 printf 使用初步

printf 是一个标准输出函数。它执行格式化输出，其格式是：

```
printf("格式信息", 数据参数 1, 数据参数 2,…);
```

其中，数据参数可有可无。

用两个双引号括起来的格式信息用于控制数据参数的输出格式。

（1）格式信息中字符除了以“\”和“%”开头的字符，其他字符原封不动按照原样输出到屏幕上。

（2）格式信息中的%和其后面的字符 d 分别是转换说明符和转换字符（合起来称为转换说明），它指定了显示参数时的格式。在%和转换字符之间还可以加一些特殊字符，用来控制输出的域宽等。printf("%d",i);表示将参数 i 按整型十进制输出。C 语言规定，转换说明符的个数应与数据参数的个数相等。例如：

```
printf("%d  %d  %d\n",x,y,z);
```

（3）格式信息中的\n是字符转义序列。\n表示换行。

1.3.4 第三个程序

【例 1.3】通过调用自定义函数计算 a+b，并在屏幕上显示结果。

```
 *------------ sum of a add b (  using Function)---------------*/
#include "stdio.h"
void show(int x,int y);                    /*（自定义)函数说明*/
void main()                                /*  main 函数定义 */
{   int a,b;
    a=1; b=2;
    show(a,b);
}
void show(x,y)                             /*（自定义）函数定义*/
int x,y;
{   int sum;
    sum=x+y;
    printf("a add b is %d \n",sum);
}
```

运行结果：

```
a add b is 3
```

分析与说明：

C 语言中除主函数以外，程序员还可以自己定义其他函数，这些函数可以像前两例中的 printf 一样被调用。printf是系统提供的库函数，使用时不必定义。

例 1.3 是由例 1.2 演变来的。本例共有两个函数，主函数 main 及 show 函数。main 函数中有一句是函数调用语句：show(a,b); 语句将参数 a、b 的值传送给 show 函数，程序转到 show 函数执行，show 函数计算 sum 的值并显示出来，show 执行完后，返回到 main 函数调用语句的下一句继续执行。函数中的 return 语句是隐含的。

注意，程序中函数的排列顺序并不决定函数的执行顺序，执行顺序是通过函数调用来决定的。

自定义函数的函数定义、函数调用和函数说明将在专门的章节讨论，读者不必为看不懂例 1.3 中的细节而烦恼。

1.4 C 程序的调试

1.4.1 调试步骤

C 语言的编译程序属于编译系统。要完成一个 C 程序的调试，必须经过编辑源程序、编译源程序、连接目标程序和运行可执行程序四个步骤。简单一点，可将四个阶段称为编辑、编译、连接、运行。

C 的源程序就是符合 C 语言语法的程序文本文件，文本文件又称为源程序文件，扩展名为.c。许多文本编辑器都可以用来编辑源程序，例如 Windows 写字板、Word 以及 DOS 的 Edit 等，要注意的是 C 源程序的存储格式必须是文本文件，在保存的时候要选择文本文件格式。

编辑完成以后是编译，对编辑好的文本文件进行成功编译后将生成目标程序，目标程序文件的主文件名与源程序的主文件名相同，扩展名是.obj。编译程序的任务是对源程序进行语法和语义分析，若源程序的语法和语义都是正确的，才能生成目标程序，否则，应该回到编辑阶段修改源程序。

编译成功以后，目标文件依然不能运行，需要将目标程序和库函数连接为一个整体，从而生成可执行文件。可执行文件的主文件名与源程序的主文件名相同，扩展名是.exe 。

最后一步就是运行可执行文件了，可执行程序要装入内存执行。如果在运行过程中发现可执行程序不能达到预期的目标，我们必须重复“编辑、编译、连接、运行”这四个步骤。

调试过程如图 1-1 所示。

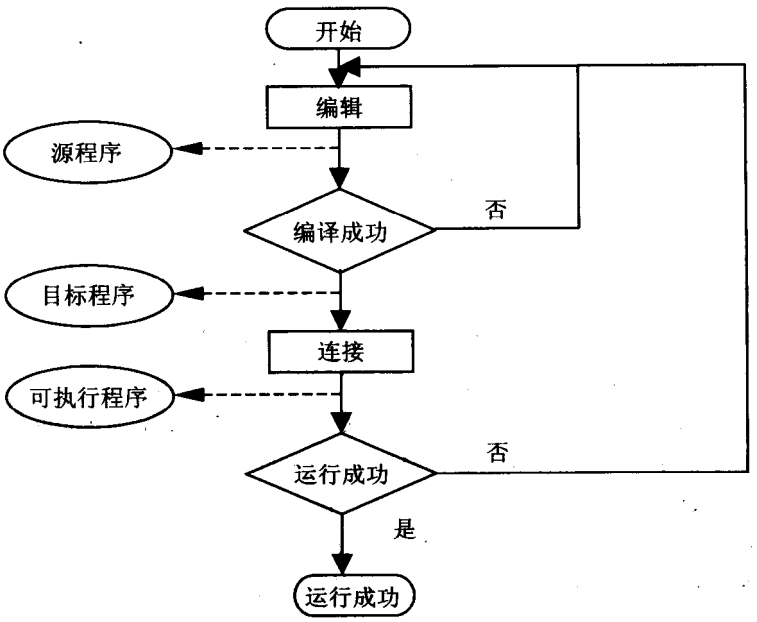


图 1-1 C 程序调试过程示意图

1.4.2 Turbo C 集成开发环境

下面将简单地介绍使用 Turbo C++ 3.0 编译程序调试 C 程序的方法。尽管 Turbo C++ 3.0 是 C++的版本，但是 C++是在 C 的基础上扩展的，所以 C 程序也能够在该环境下正确调试。

Turbo C++ 3.0 是全屏幕编辑环境，编辑、编译、连接、运行都可以在它控制下完成。

在下面的介绍中将例 1.1 生成源程序 first.c。

调试程序的具体步骤是：

（1）Turbo C++ 3.0 是 DOS 环境下的 C++调试环境，因此首先要进入 Windows 的 DOS 命令行环境，然后利用 DOS 命令进入 Turbo C++ 的 bin 目录，在\tc\bin\目录下执行 tc 命令，就可以启动运行 Turbo C++ 3.0 了。进入 Turbo C++ 3.0 集成开发环境后，看到的 Turbo C++ 3.0 界面如图 1-2 所示。

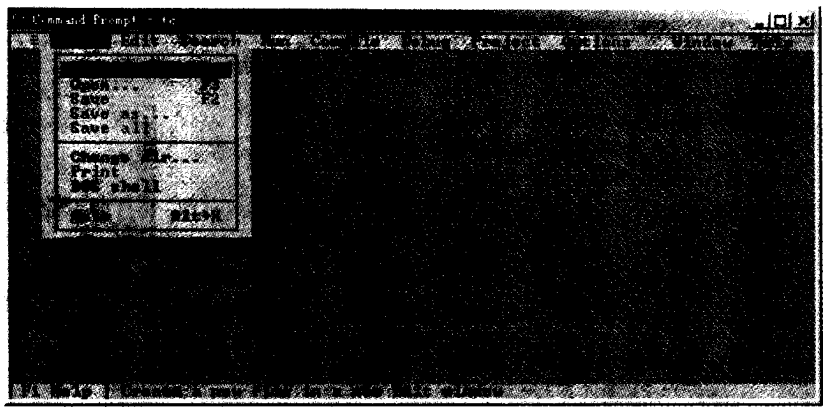


图 1-2　Turbo C++ 3.0 的集成开发环境

（2）编辑 C 的源程序文件。在【File】菜单下选择【New】，系统的光标处于等待输入 C 源程序状态，如图 1-3 所示。现在，就可以在编辑区输入程序了。输入源程序之后（结果如图 1-4 所示），必须将源程序文件保存。保存文件的方法是在【File】菜单下选择【 Save】选项，并在随后的保存文件对话框中，输入文件名 first.c，选择【 OK】命令按钮，如图 1-5 所示。

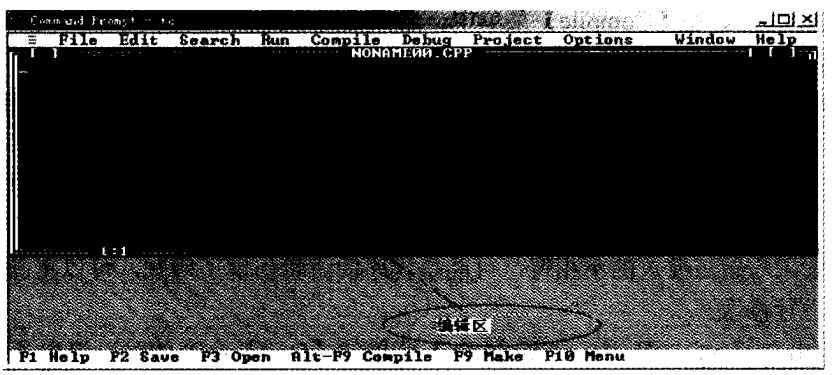


图 1-3　Turbo C++ 3.0　的编辑区

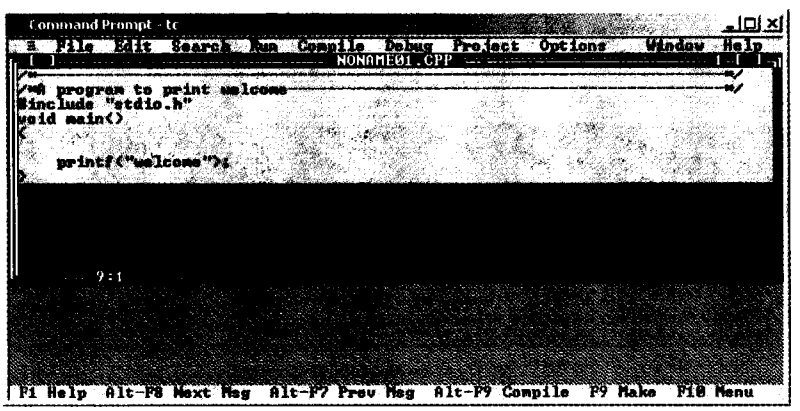


图 1-4　源程序的输入

图 1-5　另存文件对话框

（3）编译源文件。在【Compile】菜单中，选择【Compile】，或者按下快捷键【Alt+F9】，程序将被编译，如图 1-6 所示。如果程序出现语法错误，系统将具体显示错误的原因，此时，按任何一个键都可以退出显示错误的对话框，按【Enter】键回到编辑状态程序出错的地方，方便我们对程序的修改。

（4）连接程序。在【Compile】菜单中，选择【Link】，程序将被连接为可执行程序，如果我们调用的函数未被定义过，系统会提示错误。

（5）执行程序。在【Run】菜单中，选择【Run】（如图 1-7 所示），或按快捷键【Ctrl+F9】。

程序运行以后必须按下快捷键【Alt + F5】，才会看到程序的运行情况。例 1.1 程序的执行结果如图 1-8 所示。

若运行效果与题目要求的不相符，可能是程序的逻辑出现了问题，可以重新回到编辑状态对源程序进行修改。

（6）退出 Turbo C++环境。选择【File】菜单的【Quit】命令就可以退出了。

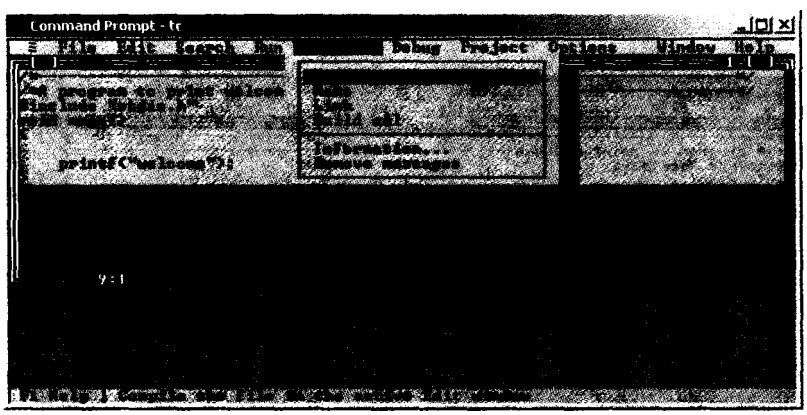


图 1-6　“编译”命令

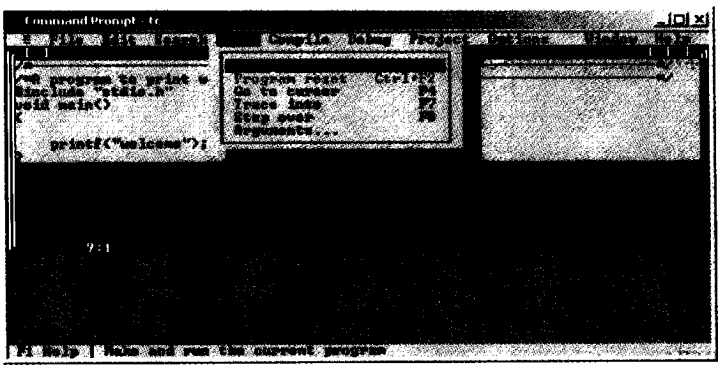


图 1-7 “运行”命令

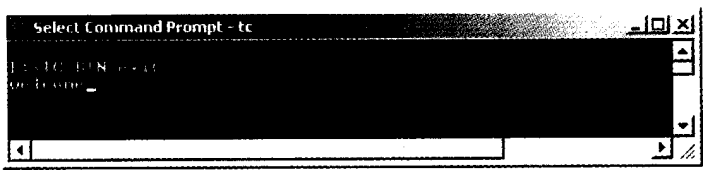


图 1-8 “first ”程序的运行

限于篇幅，对其他命令的使用方法不再赘述，读者可以参阅其他资料，尤其是 Turbo C++ 3.0 的用户说明书。在实践中积累经验也是熟悉掌握开发环境的有效方法。

本章小结

（1）计算机执行由指令构成的程序，对提供的数据进行操作。计算机程序的操作对象是“数据”。

（2）计算机语言一般分为低级语言和高级语言。

（3）从高级语言到机器语言要经过编译程序进行“翻译”，而高级语言几乎为每一种机器都创建了各自的编译程序。编译程序分为两种，一种是解释系统，另一种是编译系统。

（4）C 语言同时具备了高级语言和低级语言的特征。

（5）C 程序的构成：由函数构成，至少包含一个 main 函数。

（6）C 语言的编译程序属于编译系统。要完成一个 C 程序的调试，必须经过编辑源程序、编译源程序、连接目标程序和运行可执行程序四个步骤。

习 题

【1-1】观察生活，写一个日常生活中的程序。

【1-2】有三个同样大小的瓶子，一瓶是醋，一瓶是黄酒，剩下一瓶是空的，请用语言描述如何将装醋的瓶子装酒，而将装酒的瓶子装醋。

【1-3】低级语言和高级语言的特点是什么？

【1-4】简述 C 语言的特点。

【1-5】根据本章例 1.1 和例 1.2，自己编写一个程序，尽量使用实例中介绍过的规则。

程序功能自定。

【1-6】编写程序输出下列图案：

```
    *
   ***
  *****
 *******
```

【1-7】编写程序输出下列字符：

```
--------------------
your name:
--------------------
your password:
--------------------
```

【1-8】调试 C 程序，需要经过几个步骤？每个步骤生成的文件的扩展名是什么（如果有的话）？

第 2 章 C 语言的基本知识

本章介绍的知识是编写 C 程序的基础，如果不掌握它们，就象盖楼之前没打地基，楼是会塌的。程序设计的核心是数据运算，常量和变量是用来存储数据的，本章就来讨论如何定义变量和常量，以及它们的输入输出方法。在定义变量和常量时，要考虑它们的数据类型，本章讨论的数据类型是 C 语言的基本数据类型。

2.1 字符集和标识符

2.1.1 字符集

字符集是高级语言的编译系统所能识别的字母、数字和特殊符号。每种高级语言都有自己特定的字符集合。

C 语言的字符集合包括：

（1）大、小写英文字母： A ，B，… ，Z，a，b，…z

（2）数字：0，1，2，…，9

（3）运算符：+ - * / % > < = & | ? ! ^ ~

（4）括号：() {} []

（5）标点符号： ' " : ;

（6）特殊符号：\ $ #

（7）空白符：空格符、换行符、制表符

字符集中的字符按照 C 语言语法组合起来，就能通过编译系统的语法和词法分析。不在字符集中的字符可以在两个双引号（我们称之为字符串）之间出现。例如 "@"。

2.1.2 标识符

标识符是用来标识在 C 程序中的变量、常量（指符号常量）、数据类型和函数的，是一个字符序列。在 C 语言中，任何一个标识符必须符合下列语法规则：

（1）字母或下划线中的任一字符打头。

（2）在第一个字符后，可以是任意的数字、字母、下划线组成的序列。长度不超过 8 个。

C 语言的标识符分为三类：
- 关键字（保留字）
- 特定字
- 用户定义字

1. 关键字

关键字又称保留字，一般为小写字母。关键字是 C编译程序预先登录的标识符，它们代

表固定的意义，用户不能随便使用。若随便使用，可能出现意想不到的错误，编译能通过，但运行结果不对，且不容易检查错误之所在。

2. 特定字

特定字是具有特殊含义的标识符。它们虽然不是关键字，但是在习惯上把它们看成关键字。所以一般用户定义的标识符也不要使用它们。特定字包括：

define　undef　include　ifdef　ifndef　endif　line

这些特定字是 C程序的预处理命令。

3. 用户定义字

用户定义字是用户按照语法规则定义的标识符。用户定义字可以用来标识用户自己使用的变量、符号常量、数据类型以及函数等。通俗一点说，用户定义字就是程序员在程序设计时为变量、常量以及函数起的名字。

使用时需要注意：

（1）不能使用关键字和特定字。

（2）用户定义字为了标识不同的对象，标识符的前 8 个（甚至 7 个）字符要有区别。因为 C语言通常只识别 8个字符。如 example1 与 example2 在 VAX_11 上是相同的，编译程序根本不识别。

（3）标识符最好根据它所代表的含义取其英文或汉语拼音缩写，便于阅读和检查。

（4）根据经验，避免使用容易混淆的字符。如 1与 l，0 与 o, z 与 2 等。

（5）大、小写代表不同的意义。如 STUDENT 与 student代表不同的标识符。

（6）C语言有许多库函数。用户定义字尽量不要与其中某个函数同名。

2.2　变量与常量

数据是计算机程序处理的对象。计算机程序在运行过程中，会将程序所需要的数据从内存中读入 CPU 寄存器，在 CPU 中计算出结果；然后，再将结果写入内存，或者写入到某个输出设备上（例如显示器）。因此，程序设计的一个主要任务是控制数据所在的位置。例如，运算结果存放在何处。

数据有常量和变量之分。

变量——在程序执行过程中值是可变的。

常量——在程序执行过程中值是不变的。

让我们回忆一下程序例 1.2，在该程序当中，a、b和 sum 是变量，1 和 2 是常量。

不论是常量或是变量，在程序运行期间都要有内存空间来存放它们。存储的格式和长度都与它们的数据类型有关。

2.2.1　变量

变量是由程序命名的一块计算机内存区域，是用来存储一个可以变化的数值。

在使用一个变量之前，程序员必须为每个变量起个名字，同时还要声明它的数据类型，以便编译系统根据不同的数据类型为其静态地分配内存空间。我们称之为定义变量。

在 C 语言中，所有的变量都必须先定义后使用。这样做的原因是：

（1）编译系统会根据定义为变量分配内存空间，分配空间的大小与数据类型有关，例如，字符型的变量占一个字节，整型的变量占 2 个字节等。

（2）未经过定义的标识符，系统将不允许其作为变量名使用，这样会给程序员调试程序带来方便。

（3）编译系统可以根据变量的类型检查对该变量的运算是否合法。

定义变量的格式为：

类型说明符　　变量名表；

使用逗号分隔变量名表中的多个变量，并使用分号结束语句。

类型说明符指定了变量的数据类型，包括 int、float、double、char 等。变量名的命名规则要符合用户定义字的命名规则，一般使用小写字母。

例如：

```
int date,step;
char a,c;
```

如果不使用变量名表，也可分别定义每个变量。

```
int date; int step;
char a;
char c;
```

需要引起注意的是，在函数内部定义的普通变量，在没有用赋值号对其赋值之前，其初始值为不定值。例如，修改例 1.2 的程序为：

```
/*------------sum of a add b ------------------*/
#include "stdio.h"
void main()
{
   int a,b,sum;
   sum=a+b;
   printf("a add b is %d \n",sum);
}
```

这个程序运行以后的结果是不固定的，即所谓的“不定值”。这是因为 int a,b,sum;一句只负责分配空间，并不会将该空间的内容赋一个特殊的值。所以，内存中原来是什么就是什么，是前一次被赋值时遗留的内容。

2.2.2 常量

大多数程序都要用到常量。与变量一样，常量也是存储在内存中的，但是，常量的数值在程序执行过程中不会发生改变。我们在例 1.2 中见到的常量是直接常量，在很多书中又把它叫做无名常量或文字常量。其特征是直接书写数值，不必为该数值命名。例如 a=1; 中的 1 在 C 语言中，对于不同数据类型的常量，其表示方法是不一样的。例如，2是整型常量，'a' 是字符型常量。我们将在介绍 C 语言的基本数据类型时详细地介绍每一种数据类型常量的书写格式。

尽管直接书写常量比较简单，但是有时候也会给程序员带来麻烦：可读性差，容易出错。

例如，如果一个常量的数字很长，又需要多次使用，在编辑程序时就很可能出错。要解决这个问题，可以使用预处理命令#define 为常量起一个名字。

例如，如果在程序的开始有这样一句预处理命令： #define PI 3.1415926 ，那么，C预处理程序会将程序中所有的 PI用 3.1415926来代替。因此，编写程序时，用到圆周率的地方可以直接书写成 PI，而不必每次都写成 3.1415926。

我们可以把这种常量叫做符号常量，但是注意这种符号常量与其他高级语言（例如 Pascal和 C++）用 const方法定义的符号常量还是有区别的。用 const 定义的符号常量是在定义一个存储块的同时赋一个初值给它，并且规定这个值不能在程序运行的过程中被修改。与预处理程序做替换完全不同。不论是直接常量或符号常量，常量存放的空间不需要程序员定义。

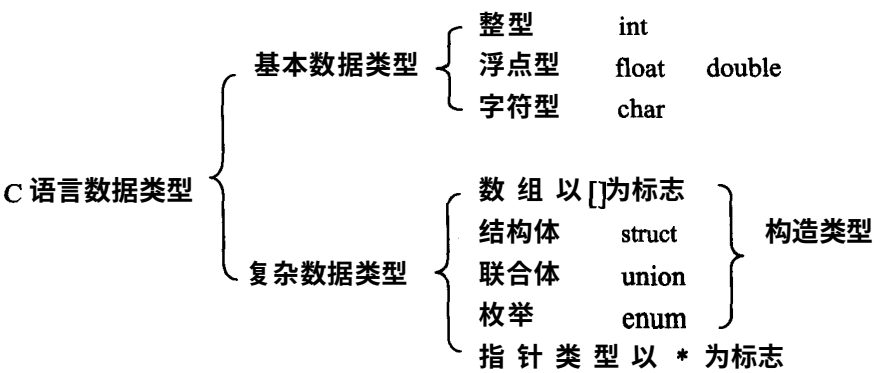
用预处理命令#define 为常量命名时，通常用大写字母以区分变量名。

2.3 C语言的数据类型

数据类型在高级语言中是一个很重要的概念。不同的数据类型在内存中的存储方式是不相同的，不同数据类型的数据在内存中所占的字节数也大多不一样。高级语言能表示的数据类型越多，程序编写起来就越简单。

2.3.1 C语言有哪些数据类型

C语言具有丰富的数据类型：



其中，基本数据类型比较简单，定义时可直接使用。

构造类型是由基本数据类型或其他构造类型聚集而成。

指针在 C 语言中使用极为普遍，指针提供了动态处理变量的能力，是 C语言的精髓。

2.3.2 基本数据类型

C语言的基本数据类型是构造其他类型的基础，本章我们只介绍基本数据类型的用法。C语言的基本数据类型包括：

整型
- 短整型 short int
- 长整型 long int
- 普通整型 int

浮点型
- 普通浮点型 float
- 双精度浮点型 double

字符型 char

不同数据类型的数据由于其在内存的存储方式不同，存储所占的二进制位（bit）大多不相同。即使是相同类型的数据在不同种类的计算机所占位（bit）数也不完全相同，如表 2-1 所示。

表 2-1 数据在不同计算机所占的二进制位数

类型 \ 机型 长度	DEC PDP-11	Honeywell 6000	IBM 370	IBM-PC
char	8 （bit）	9 （bit）	8 （bit）	8 （bit）
int	16	36	32	16
short int	16	36	16	16
long int	32	36	32	32
float	32	36	32	32
double	64	72	64	64

一个数据所占二进制位数的多少，决定了它表示的数据的最小值和最大值。就像两位的十进制正整数，最大值是 99，最小值是 00。

2.4 整型数据

2.4.1 整型变量

在 C 语言中，按照整型变量所占的二进制位数来分类，整型变量分为 short（短整型）、int（普通整型）和 long（长整型）三种。

而根据整型变量是否带符号位来分类，可以分为不带符号的整型变量和带符号的整型变量。无符号用关键字 unsigned 表示，用 unsigned 与短整型、普通整型和长整型三种类型相匹配，又可以构成无符号短整型、无符号普通整型和无符号长整型三种类型。

说明整型变量的语法是：

限定词 int 变量名表；

限定词包括 long 、short 和 unsigned。在有限定词的情况下，int 可以省略。

例如：

```
short (int) data1;     /* 定义了一个短整型变量 data1*/
long (int) data2;      /*定义了一个长整型变量 data 2 */
unsigned data3;        /*  定义了一无符号普通整型变量 data 3 */
```

限定词可以是一个，也可以是两个。例如， unsignde short (int) data4;定义了一无符号短