第1章　概　　述

C 语言是国内外广泛应用的一种计算机语言，它之所以得到迅速推广，与它的特点是分不开的。本章学习的目的是让初学者在深入学习 C 语言之前，首先对 C 语言有个初步的了解。需要说明的是，这里面有很多的名词概念可能看不懂，这是很自然的，因为既然是概述，就不可避免地涉及后面的内容，没关系，学到后面自然明白，这个章节里主要掌握怎样把 C 语言代码翻译成可执行的程序，学习本章的“第一个 C 语言程序”就够了，其他可以略过。

1.1　程序设计语言

1.1.1　程序设计语言的概念

什么是程序设计语言？

首先来了解一下程序和程序设计的概念。“程序”这个词大家都不陌生，做任何事情都有个“程序”，因此“程序”一词在词典中的解释为：事情办理的先后次序。那么对于计算机而言，“程序”是什么呢？人们要让计算机解决一个问题时，必须把解决这个问题的步骤通过一条条指令的形式告诉计算机，先干什么后干什么，计算机才知道如何去完成任务。一般我们把人们事先准备好的，用来指挥计算机工作的，描述工作步骤的指令序列称为程序。程序的生成过程称为程序设计。

从程序的概念可知：程序是由指令序列组成的，人们通过指令告诉计算机如何工作，指令是人和计算机之间沟通的工具，指令是一些预定的文字描述。构造指令序列来控制计算机工作的程序称为编程。把这些人机交互的指令总和称为计算机语言。

1.1.2　程序设计语言的发展

程序设计语言是人与计算机交互的媒介。人要掌握程序设计语言才能编写程序，计算机也要能“读懂”程序设计语言，才能明白如何按步骤工作。程序设计语言经历了从机器语言到高级语言的发展过程。

1．机器语言

计算机硬件唯一能识别的语言是机器语言（是 0 和 1 组成的二进制代码），如图 1-1-1 所示，计算机控制器直接把这些 0、1 组合翻译成控制硬件的开关信号（图 1-1-2），一个计算机能识别的机器指令总和称为这种计算机的指令系统。在最初阶段，人们直接使用机器语言编写程序，机器语言有其固有的弱点，由于是二进制代码，人们编写和阅读程序十分困难，又非常容易出错，且不同的计算机其指令系统也不同，无法不经修改就在另一种计算机上执行，也就是可移植性较差。用机器语言编写的程序虽然其执行效率较高，但由于与人们日常使用的自然语言差别太大，花费在程序设计和调试运行上的时间太多，整体效率是极低的。

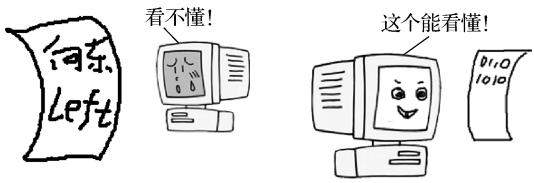


图 1-1-1 计算机硬件只能识别机器语言

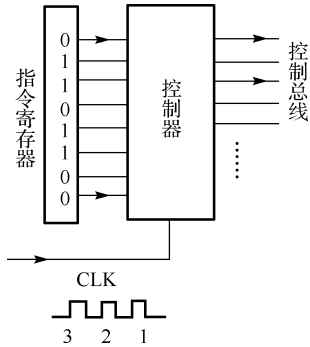


图 1-1-2 控制器把机器指令翻译成控制信号

2．汇编语言

为了解决二进制代码编程带来的困难，人们采用助记码和符号地址来代替机器语言中的二进制指令代码和物理地址，这种采用助记码和符号地址的语言称为汇编语言。例如：

```
mov ax,0231H;
add ax,1;
```

用汇编语言编写出程序，然后通过人们预先设计好程序的翻译程序——“汇编程序”，一对一地翻译成机器语言，再让计算机执行。用汇编语言编写的程序执行效率和机器语言程序一样高，且阅读性提高了，但由于汇编指令和机器指令之间是一对一的，编程者仍然需要较多的计算机硬件知识，也同样存在着编程效率低，可移植性差的缺点。

3．高级语言

随着计算机各种技术的发展，程序设计语言也在不断发展，为了脱离机器硬件不同带来的桎梏，为了方便非计算机专业人员编程，人们开始用一种比较接近于自然语言（主要是英语）和数学语言的语言来编写程序，这样的语言被称为高级语言。例如，用 input 要求计算机执行输入，用 output 要求计算机输出，高级语言是一种面向过程的计算机语言。编写程序时，使用语句描述解决实际应用问题的过程，一条语句往往相当于许多条机器指令，描述方式也与我们日常处理问题的逻辑思维习惯相近，因此高级语言程序更易于理解。另外，高级语言脱离了具体的计算机指令系统，编程时不需要考虑机器的内部结构和存储单元的分配，只要安装了相应的高级语言的编译程序（或解释程序），其源程序就可以不经修改，或只需要很少量的修改就可以在其他类型的计算机上运行，因此，高级语言具有良好的移植性。对于非计算机专业的人员来说，高级语言比较容易掌握，高级语言克服了面向机器结构的缺点，使得程序易读、易维护、易交流。高级语言发展很快，已达数百种，下面列举几种曾经使用和正在使用的高级语言。

（1）FORTRAN 语言（Formula Translation），其名含义是公式翻译语言，诞生于 20 世纪 50 年代中期，常用于科学和工程计算中，现在使用者较少。

（2）BASIC 语言（Beginner’s All-purpose Symbolie Instruction Code），是初学者通用符号指令代码一词的简写，诞生于 20 世纪 60 年代中后期，语言简单易学，是一种会话性语言，适合初学者学习。

（3）Lisp 语言， Lisp 语言最早是在 20 世纪 50 年代末由麻省理工学院（MIT），有人工智能之父之称的 John McCarthy 开发的，是一种适用于人工智能领域的表示语言。

（4）Pascal 语言，诞生于 20 世纪 70 年代初期，是一门结构化程序设计语言，适合于教学、科学计算、数据处理和系统软件开发。后期出现了 Object Pascal，即 Delphi。

（5）C 语言。诞生于 20 世纪 70 年代初，80 年代开始风靡全世界，适用于开发系统软件，数值计算、数据处理等软件。

（6）FoxPro，原名 FoxBase，最初是由美国 Fox Software 公司于 1988 年推出的数据库产品，是一种立足于数据库处理的编程语言。

（7）Java 语言，诞生于 20 世纪 90 年代，是一种新型的跨平台分布式程序设计语言，具有简单、安全、稳定、可移植性强的特性，有可能成为未来网络环境上的“通用语言”。

在诸多高级语言中，大部分语言可用来做日常事务的处理，例如，BASIC、C、FoxPro 被称为通用语言；有些应用领域则有专用性，例如，Lisp 适用于人工智能领域，被称为专用语言。

通常，我们把用高级语言或汇编语言编写的程序称为源程序，把编译后的结果称为目标代码。

1.1.3　程序设计的一般步骤

使用计算机解决一个实际应用问题通常需要经过以下处理步骤：首先要明确需要解决的问题是什么，即提出问题；其次要分析问题中涉及了哪些数据，如何在计算机中进行表示，即描述数据结构；同时还要将复杂的问题分解为计算机可以完成的若干操作步骤，即确定算法；然后用选定的某种计算机语言描述数据结构，并根据算法编写程序；编好的源程序输入计算机后，往往需要反复调试，修正其中的语法错误和逻辑错误，直至得到正确的运算结果。上述整个过程就是程序设计的一般步骤。

1.2　C 语言的发展历史和特点

1.2.1　C 语言的发展历史

在 C 语言诞生之前，像操作系统这样的系统软件只能用汇编语言编写。但汇编语言对计算机硬件的依赖太大，使程序的可读性特别是可移植性很差，所以设计一种既具有高级语言优点又具有低级语言特性的语言，对编写操作系统及其他系统软件具有重要意义，C 语言就是在这种情况下诞生的。

1960 年出现的 ALGOL60 很受程序设计人员的欢迎。用 ALGOL60 描述算法很方便，但离计算机硬件较远，不宜用来编写系统程序。1963 年英国剑桥大学在 ALGOL 语言的基础上推出了 CPL（Combined Programming Language）语言，增加了对硬件的直接处理能力，但由于规模较大，实现比较困难。

1967 年剑桥大学的 Martin Richards 对 CPL 进行了简化，推出了 BCPL（Basic Combined Programming Language）语言。

1970 年美国贝尔实验室的 Ken Thompson 对 BCPL 语言做了进一步简化，突出了硬件处理能力，取名为 B 语言（BCPL 的第一个字母），并用 B 语言编写了 UNIX 操作系统程序。但 B 语言过于简单，功能有限。

1972 年，贝尔实验室的 Dennis M Ritchie 对 B 语言进行了补充和完善，保留了 B 语言的硬件处理能力，扩充了数据类型，强调了通用性，这就形成了 C 语言（BCPL 的第二个字母）。

1973 年，Ken Thompson 和 Dennis M Ritchie 两人一起使用 C 语言改写 UNIX 操作系统，并在 PDP-11 计算机上加以实现，改写后的 UNIX 操作系统有 90%以上的是用 C 语言写成的。C 语言伴随着 UNIX 操作系统成为一种最受欢迎的计算机程序设计语言，C 语言的诞生可用图 1-2-1 表示。

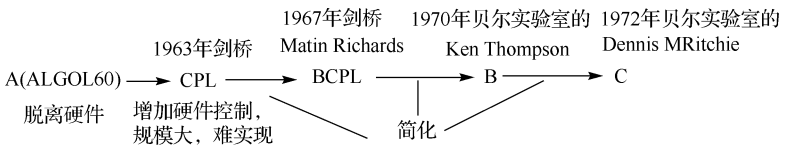


图 1-2-1　C 语言的由来

1977 年，可移植 C 语言编译程序的出现使得 C 语言移植到其他机器时所需做的工作得到很大简化。随着 UNIX 系统在不同硬件平台上的广泛应用，C 语言被移植到大量各种类型的计算机上，并最终脱离 UNIX 和 PDP 平台。

1.2.2　C 语言的特点

从上面的 C 语言的产生过程可以看到，C 语言最初是为了更好地实现 UNIX 操作系统而产生发展来的，因而它不仅具有实现操作系统所必须的语言的特点，同时也兼具高级语言的特点，主要表现在以下几个方面。

（1）生成的可执行代码简短而高效。用 C 语言编写的程序，其效率大约是实现相同功能的 C++代码的 8～10 倍，仅比汇编语言效率低 10%～20%，但是 C 语言的可理解性和易维护性却大大高于汇编语言，这一特点使它特别适合应用于系统资源有限的应用场合，如嵌入式系统。

（2）允许直接访问物理地址，可以直接对硬件进行操作。C 语言可以用来实现汇编语言的大部分功能，如位（bit）操作、字节操作、地址操作、直接对硬件端口进行读写等。使它可以用来编写与硬件较接近的系统软件。

（3）C 语言数据结构和运算符丰富，具有先进程序设计语言普遍配置的数据结构。C 语言具有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型和枚举类型，能用来实现多种复杂的数据结构（如链表、树、栈等）。特别是与地址密切相关的指针类型，使用起来更加灵活多样。C 语言中使用指针的概念实现对内存单元的访问控制。指针本身是一个可以被赋值和进行运算的变量，它指向一个固定长度的内存单元（可以把指针想像成整个内存单元的门牌号码），对其所指向的内存单元及其邻近单元的访问，可以通过对指针简单的加减运算而得到。C 语言包含的运算符很广泛，共有 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理，从而使 C 语言的运算类型极其丰富，表达式类型多样化，灵活使用各种运算符可以实现其他高级语言中难以实现的运算。

（4）可移植性好。与汇编语言相比，C 语言不依赖于任何的机器硬件和操作系统，几乎可以在大多数通用的计算机软硬件平台上不加修改或稍做修改地运行。

（5）结构化语言的显著特点是代码模块化，即程序的各个部分除了必要的信息交流外，

彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。C 语言引入函数的概念来实现程序的模块化，并具有循环、条件分支语句结构化语句，从而可使整个程序划分成若干功能相对独立的模块，便于设计和移植使用。

由于 C 语言同时兼具高级语言和低级语言的特点，因而 C 语言也被称为中级语言，这使它具有其他高级语言无法相比的速度优势和灵活性，而它简单的语法规则和易读性又弥补了汇编语言的繁琐、难以调试和难以理解的缺点，这是 C 语言被广泛推崇的原因之一。

1.3　第一个 C 语言程序

在此节当中，我们开始第一个 C 语言程序的编写，并学习怎样让它能够在计算机上执行，我们使用的环境是 VC++ 6.0，没有安装的需要下载安装，这个软件可以运行在 Windows 2000、Windows XP、Windows 7 上，遗憾的是截至本书交稿时，还没有发现能在 Windows 8 上运行的版本，在本书配套的实验手册上则介绍了如何使用 VS 2013 练习 C 语言程序。

1.3.1　必要的计算机基础——初识内存

计算机内存是由一个个小的电器元件组成的，每个小单元能保存一个低电平或高电平，分别用来表示 0 或者 1，这被称为一个二进制位，英文为 bit，而从早期开始，一直是 8 个小单元用同一根地址信号线控制，这 8 个小单元被称为一个字节 Byte（如图 1-3-1 所示）。我们给每个字节指定一个不同的编号，当 CPU 发送此编号时，地址译码器将使唯一的一根地址控制信号线为高电平，打开这个字节的控制开关，使这个字节可以被读写，这个编号我们就把它称为此字节的地址。

在我们编程过程中，地址一般采用 8 位十六进制数表示，因为我们采用的 VC++ 6.0 诞生于 386、486 时期，那个时期计算机的地址寄存器是 32 位二进制，对应 8 位十六进制数，但后面程序实例中，为了减少同学进制转换带来的分神，把精力集中在理解原理上，有时也把地址直接用十进制数输出。

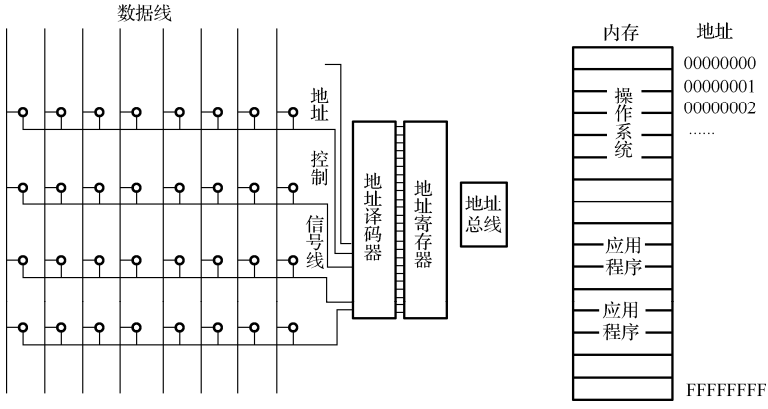
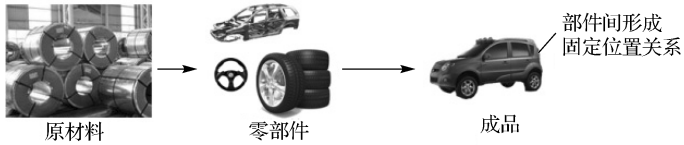


图 1-3-1　内存与地址

1.3.2　一个 C 语言程序从编写到执行的过程

如图 1-3-2 所示，一个 C 语言程序从编写到能够执行，需要经过如下环节。

第一步，需要一个编辑器，把我们要计算机执行的指令，形成一个文件，保存起来，这就是源程序，例如我们编写的程序名叫 p1.cpp。



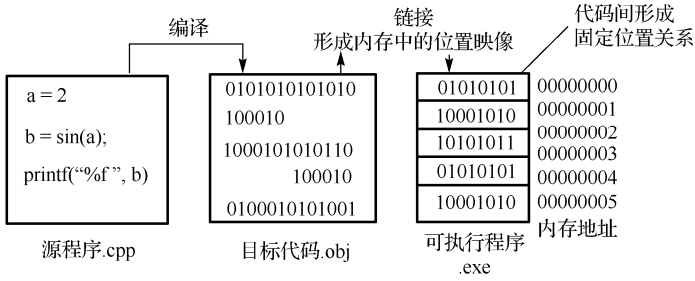


图 1-3-2　C 语言程序编译、链接过程

第二步，使用一个编译器，把源程序的每条语句翻译成计算机能理解的机器指令，这个过程称为编译，编译好的结果称为目标代码（上面程序编译完后会形成文件 p1.obj），这就好比把钢材等原材料加工成了汽车元件，但是没有装配起来之前，汽车不可能跑起来。

第三步，把翻译好的目标代码和用到的系统函数模块组装成一个程序的内存映像，这个过程称为链接（程序菜单是 build，也就是组装、建造），因为计算机访问任何一个内存单元，无论是执行指令还是读/写内存，都需要这个单元的实际内存地址，例如，C 语言语句跳转指令 goto M1；翻译成对应的低级语言指令就变成了 jmp 2135H 形式。2135H 是指一个具体的单元地址是 2135H，里面存储着要跳转到的语句，所以我们的目标程序，每条指令，每个存储单元必须对应到一个具体的内存地址中，这个程序才真正可以运行，就好比把汽车零件组装起来，各零件安装在合适的位置上，彼此能够提供衔接和服务，这个汽车才真正能跑起来。链接的结果在 Windows 操作系统中会形成一个可执行的.exe 文件，例如，上面程序会得到一个 p1.exe 程序文件。

一般的 C 语言编译器都包含了从编辑、编译到链接全部的功能，下面就使用 VC++ 6.0 来编写第一个 C 语言程序。

1.3.3　使用 VC++ 6.0 环境调试程序

1．调出 VC++ 6.0

首先查看自己机器上是否安装了 Microsoft Visual Studio 6.0，如果没有安装，则需要先安装软件，在此不介绍安装步骤。如果已经安装了 Microsoft Visual Studio 6.0，则在[开始]程序菜单内会有如图 1-3-3 所示的菜单，单击[Microsoft Visual C++ 6.0]执行程序。

VC++ 6.0 界面如图 1-3-4 所示。

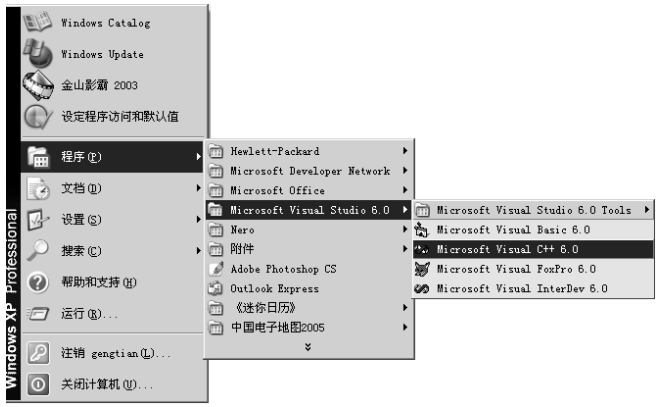


图 1-3-3　VC++ 6.0 菜单

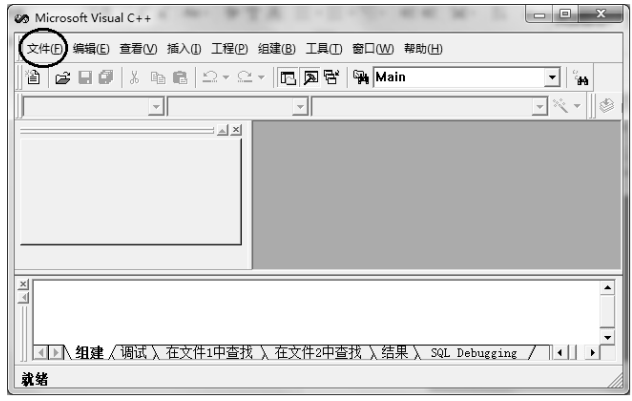


图 1-3-4　VC++ 6.0 界面

2．新建 C 语言源程序

（1）执行[文件]→[新建]菜单命令。

（2）选择“文件”选项卡下的 C++ Source File 种类（如图 1-3-5 所示）。

（3）选择存放文件的位置，如图 1-3-5 所示。

图 1-3-5　创建 C++ Source File 文件

（4）确定源文件的名称，文件的扩展名可以是.cpp 或.c，如果是.cpp，则表示为 C++源文件，文件中除 C 语言的语句外，还允许出现 C++的语句及运算符；如果扩展名为.c，则为纯 C 语言源文件，文件中若出现 C++的语句或运算符，在编译时将会报错。省略扩展名时，默认扩展名为.cpp，在我们的练习中可以选择任意一种。

（5）单击[确定]按钮后开始编辑。

3．编辑源程序

在程序编辑区输入源程序，如图 1-3-6 所示。

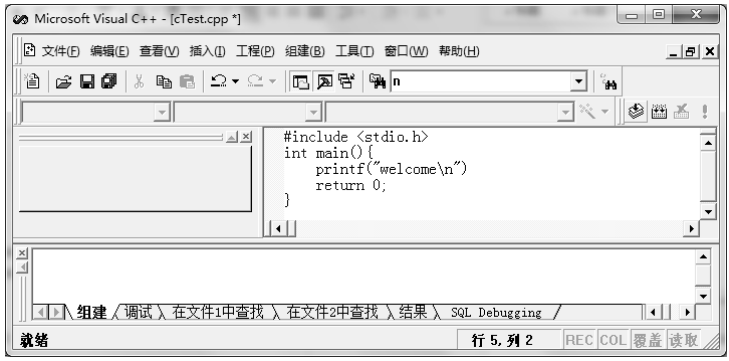


图 1-3-6　输入源程序

4．编译、链接与执行

选择[组建]→[编译]菜单命令或单击“编译”按钮（如图 1-3-7 所示）进行编译，在后面出现的两个提示对话框（如图 1-3-8）中单击“是”按钮。

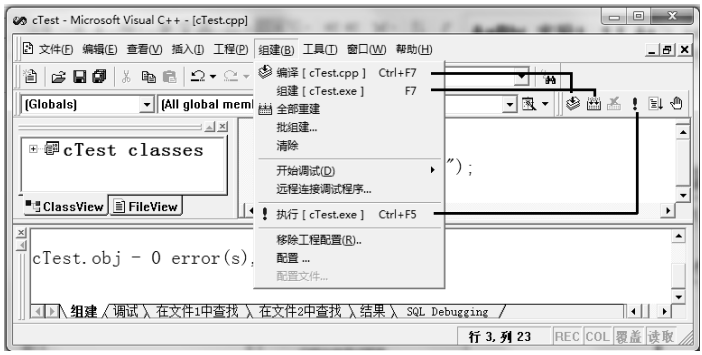


图 1-3-7　编译

图 1-3-8　提示对话框

编译后查看下面输出窗口有无错误指示，如有，根据窗口内的提示进行修改（如图 1-3-9 所示），双击错误提示，可以使系统指出错误发生所在行，例如，双击错误提示“i:\t\cTest.cpp(4) : error C2143: syntax error : missing ';' before 'return'”（return 语句前缺少分号），会在出错行出现一个箭头提示，不断改正错误，然后重新编译，直到没有错误后，会产生 cTest.obj 文件。该文件存在源文件所在文件夹下的 Debug 子文件夹内（如图 1-3-10 所示）。

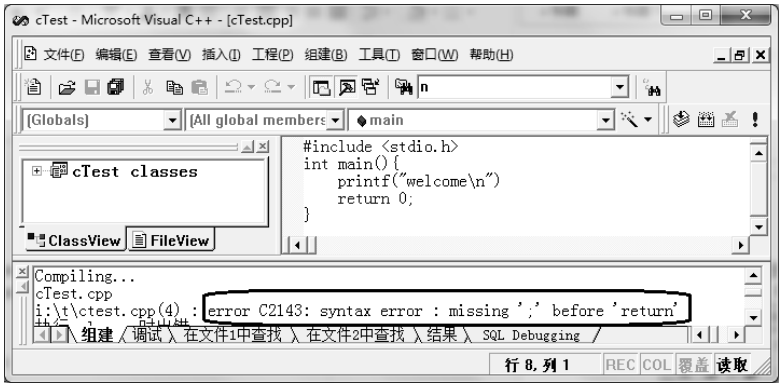


图 1-3-9　错误提示

编译成功后，执行[组建]→[组建]菜单命令或单击“链接”按钮进行链接。若发现错误，根据错误提示进行修改（在此也可能出现一些找不到函数等错误），当链接没有错误时，会在源文件所在目录下的 Debug 子目录内产生.exe 文件，如图 1-3-10 所示。

执行[组建]→[执行]菜单命令或单击“!”执行按钮开始执行，每次执行的结果会显示在一个控制台窗口内（如图 1-3-11 所示）。

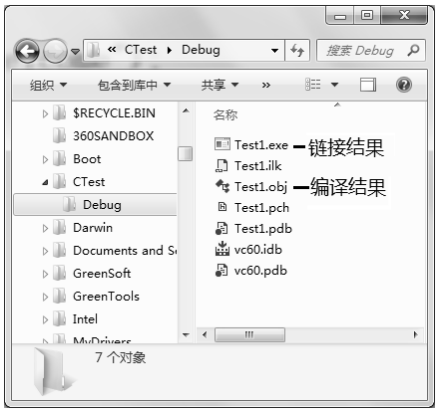


图 1-3-10　编译链接的结果

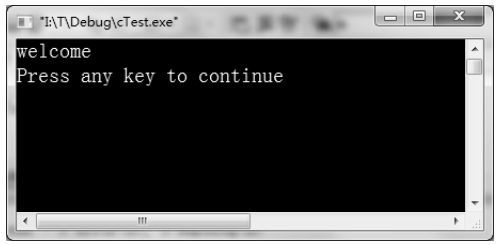


图 1-3-11　结果输出

5．注意事项

显示执行结果的窗口要随手关闭，下次执行还会出现，如果不关闭，再次链接时可能会出现错误。

调试完一个程序后要执行[文件]→[关闭工作空间]菜单命令关闭所有文件，然后再建立新程序，否则在一个程序内会出现两个 main 函数，在链接时会出现错误，如图 1-3-12 所示。

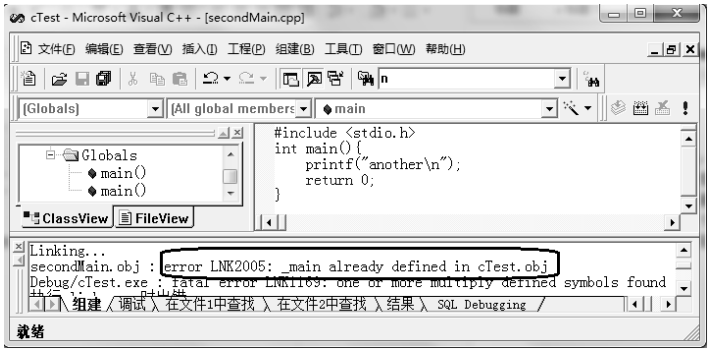


图 1-3-12　多个 main 函数引发的错误

1.3.4　编译时常见错误

写完程序，先要进行编译，大量的语法错误会在此时出现，常见的语法错误有以下几种。

1．Cannot open include file:

嵌入文件打不开，#include 后的文件名拼写错误

2．syntax error : missing ';'

缺少分号或有多余的“}”。

3．newline in constant

常量跨行，一个常量写到多行内，实际上可能是漏写一个双引号“"”。

4．unexpected end of file found

未发现文件结尾，缺少“}”或“{}”不配套。

5．unknown character

错用了中文标点，特别从别处复制的源文件，可能有不可见的中文字符，建议在英文状态下输入代码。

6．XXX:undeclared identifier

不认识的关键字，一般是变量名未声明，或使用了函数却没嵌入头文件。

1.3.5　链接时常见错误

一个程序经过编译没有错误，仅仅表示没有语法错误，但是链接时还可能会出现错误，链接时的错误常见的有以下三种。

1．unresolved external symbol _main

无法找到 main 函数，一般是 main 函数名写错了，大多数写成了 mian。

2．_main already defined in XXX

main 函数已经存在，一个程序中出现了多个 main 函数。

通常出现这种情况的原因是，写完程序后，没有关闭工作空间，又建立了新文件，导致一个程序，两个文件，都有 main 函数，解决办法是，打开工作空间的文件页面，删除一个不需要的文件，如图 1-3-13 所示，单击选中要删除的文件，按 Delete 键删除。

3．cannot open Debug/xxx.exe for writing

无法写入 Debug/xxx.exe 文件，原因是上次执行的 console 窗口未关闭，程序处于执行状态。

解决办法是直接关闭，如果关闭比较困难，可以选择把 VC++关闭后重新打开或者可以按 Ctrl+Alt+Del 组合键，找到运行的进程，强行关闭，如图 1-3-14 所示。

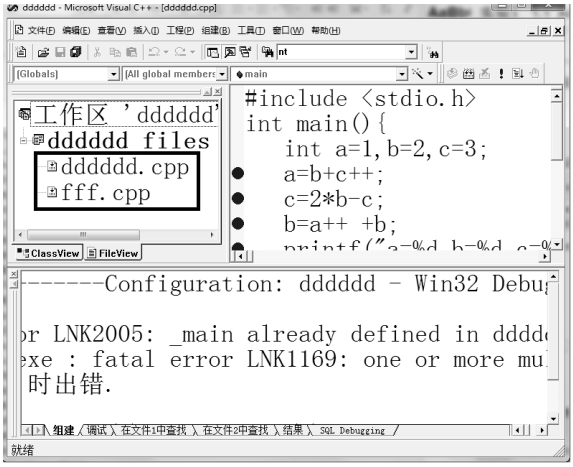


图 1-3-13　删除一个不需要的文件

图 1-3-14　直接关闭程序进程

1.4　C 语言程序结构与注意事项

1．C 语言的字符集

世界上不同的国家用的文字符号是不一样的，同属于拉丁语系的文字符号也不同，例如，法语、德语中的个别字母在英语中是不存在的，因此 C 语言有必要对其中可以使用的文字符号做出规定，C 语言中可以使用的字符包括以下几种。

① 10 个阿拉伯数字：0、1、2、3、4、5、6、7、8、9。

② 52 个大、小写英文字母：a、b、c、...、z、A、B、C、...、Z。

③ 空白符：空格符、制表符、换行符和换页符统称为空白符。它们主要用于分隔，一般无其他特殊意义。C 编译程序对空白符常常忽略不管。但空白符在程序的适当地方使用可增加程序的清晰性和可读性。

④ 29 个特殊字符，即：+、-、*、/、<、=、>、?、{、}、!、 " 、#、%、&、(、)、,、'、.、;、:、[、]、\、^、_、|、～。

⑤ 转义字符：由转义符“\”和特定的一个或几个字符构成。转义符后面的字符不再是原来的含义，而是具有另外的特定意义。如“\n”表示回车换行，而不是字符 n 的原义了，后面有专门学习的章节。

2．初识函数结构

C 语言程序基本组成单位是函数，前面我们输入的代码中，int main( ){…}，就是一个函数，函数分为函数头和函数体，如图 1-4-1 所示，将来我们还可以写更多函数，而 main 函数是必不可少的，而且只能有一个，链接程序在安排执行次序时，先执行的是 main 函数，其他函数要通过 main 函数调用才能得到执行，在此我们只需要仿写，详细的函数内容要看本书的函数章节。

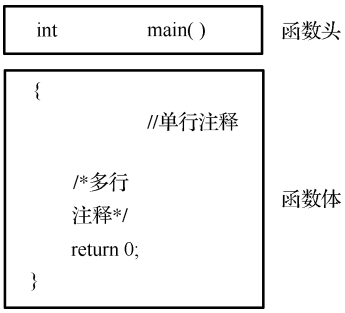


图 1-4-1 函数组成与注释

3．注释语句

我们有时要在代码文件中做一些注解，这些内容只供读者阅读，不需要编译程序翻译成目标代码，这就是注释，C 语言中提供了用“/* */”引起的多行注释方法，C++中又提供了“//”单行注释方法。我们练习编写用的程序可以没有注释，而实际应用的程序必须有足够多的注释，帮助修改程序的人读懂程序，易懂的程序才是好程序。

4．语句结束符，及其输入代码注意事项

C 语言的每条语句都用分号“;”作为语句结束标记，因为有了语句结束符号，所以 C 语言中，既可以在一行上写多条语句，也可以把一条语句写在多行上，但是如果把一个常量或在一个单词的中间换行，则会出现错误，这时必须使用续行符号“\”，例如：

```
//程序 1-4-1
#include <stdio.h>
int main(){
    printf("ABC"); printf("DEF");   //一行多条语句，正确
    printf("ABCDEF"
           );                       //一条语句写在多行上，正确
    printf("ABCDEF
           ");                      //断开常量，错误
    printf("ABCDEF\
           ");                      //断开常量但使用了续行符，正确
    return 0;
}
```

在编写程序中，**程序中使用的具有语法含义的字符，都是英文半角字符**，例如，我们语句结束符号“;”如果写成全角的“；”，将会出现语法错误，在此顺便声明一下，因为中文标点占空间大，看上去更加醒目，教材中可能会有为突出某个标点符号而故意使用中文标点的现象，但实际上，作为程序中控制用的标点、运算符只能是英文半角字符，全角字符或汉字只能出现在字符串的内容里，或出现在注释当中。

5．标识符

标识符是系统或程序员定义的用于命名变量或函数的名字。如同人要有名字一样，在 C 语言中对于要使用的对象，都要为其命名，以便在程序中引用。标识符并不是一个随意的字符序列，标识符必须遵守一定的构成规则，详见第 2 章。

6．关键字

关键字又称为保留字，它是具有特定含义，专门用做语言的特定成份的一类标识符。标准 C 语言的关键字共有 32 个，如下所示。

auto、break、case、char、const、continue、default、do、double、else、enum、extern、float、for、goto、if、int、long、register、return、short、signed、sizeof、static、struct、switch、typedef、union、unsigned、void、volatile、while。

关键字全部是小写，这些单词不必强记，随着学习自然会非常熟悉。

另外，下面几个字虽说不是关键字，但读者也把它当做关键字而不要在程序中使用，以免造成混淆。这些字是：define、undef、include、ifdef、ifndef、endif、line。这些字主要用在 C 语言的预处理中。

7．运算符

运算符用来对运算对象进行规定的运算，并得到一个结果值。运算符的功能是由系统预定义的。运算符通常由一个或几个字符组成，如“+”表示加法运算，“=”表示赋值运算，**值得注意的是**，C 语言中有些运算符是由多个字母组成的，但其仍然是一个运算符，如“==”表示“相等”的判断，sizeof( )表示求变量字节数等。

8．分隔符

分隔符用来分隔各个词法记号或程序正文，用于表示一个实体的结束和另一个实体的开始。常用的分隔符有

{　}、(　)、,、;、:、空白。

这些分隔符不表示任何操作，仅用于构造程序。

9．C 语言区分大小写

C 语言区分大、小写，拼写相同而大、小写不同的两个单词，会被认为是两个不同的个体，这与 VB 等程序不一样，VB 中是不区分大小写的，C 语言中的关键字都是小写的，除规定以外，程序员间也遵守一些约定俗成的习惯，例如，我们后面要学习的宏定义，以及 VC++ 扩充定义的类型习惯上是大写的，但这仅仅是习惯，并没有强制要求。

1.5　本章小结

程序是用来指挥计算机工作，描述计算机工作步骤的指令序列。程序的生成过程叫做程序设计。用来描述指令、编写程序的语言叫做程序设计语言。程序的生成过程实际就是设计指令序列并用语言进行描述的过程。程序设计语言从机器语言发展到汇编语言，最后发展到接近人类自然语言和数学语言的高级语言。

C 语言最初是为了更好地设计并实现 UNIX 操作系统而诞生的。C 语言既具有高级语言简洁高效的特点，又具有低级语言灵活的特点，对编写操作系统及其他系统软件具有独到的优势。同时它是一种结构化的语言，数据类型和运算符丰富，运用灵活，可以实现其他语言所不能实现的复杂运算；可移植性好，可以在不同的平台和操作系统上使用。

C 语言源程序通过编译和链接过程才能转变为可执行程序。编译过程主要包含预处理、编译和汇编输出目标代码文件；目标代码文件再与标准库文件等其他链接文件链接起来，输出可执行文件。

习题 1

1-1 什么是计算机的指令系统?

1-2 计算机硬件能识别哪种语言？

1-3 C 语言源程序变成可执行程序需要经过哪些环节？

1-4 写出下列程序的输出结果。

```
int main(){
  printf("\n");
  printf("Good morning ,everyone!\n");
  printf("Good morning , Mr.Zhong!\n");
}
```

1-5 参照本章例题，编写程序，在屏幕上输出：

```
**************************
      Welcome To C World!
**************************
```

1-6 试把习题 1-4 程序中的\n 去掉，然后重新编译，执行会得到什么结果？据此知道\n 所起的作用是什么？