第1章

初识C语言

【内容导读】

本章首先从“什么是C语言”和“为什么学习C语言”两个问题入手，既可激发读者的学习兴趣，又可使读者认识到学习C语言的重要性。其次，通过具体的C语言程序实例，说明C程序的基本架构。最后，结合具体的C语言集成开发环境VC++ 6.0，给出了C程序的具体执行过程及基本的C程序调试方法。

【学习目标】

(1) 了解计算机语言及C语言的发展历程；

(2) 了解C语言的特点及其应用；

(3) 掌握C语言程序的基本架构及其运行过程。

1.1 什么是C语言

1.1.1 计算机语言

C语言对于初学者是陌生的，但语言对于大家来说是很熟悉的，人和人之间的沟通离不开语言，通过语言的有效组织可以进行各种各样的事务处理。现在我们的生活离不开计算机，人与计算机之间进行交互的语言就是计算机语言。按照计算机语言的发展历程，可以大致将其分为机器语言、汇编语言和高级语言三个阶段，各阶段语言的构成及特点简介如表1-1所示。

表1-1 计算机语言的发展

计算机语言	构 成	优 点	缺 点
机器语言	0,1	所编写的代码能被计算机直接识别和接受，不需要翻译，执行速度快	难以学习、阅读和调试，使用人群为极少数的计算机专业人员，可移植性非常差
汇编语言	符号	相对于机器语言而言，较易学习、阅读和调试	所编写的代码不能被计算机直接识别，需要经过汇编程序进行汇编即翻译后才能执行，可移植性差

续表

计算机语言	构　成	优　　点	缺　　点
高级语言(分为面向过程和面向对象两种)	接近于自然语言	易于学习,功能强大,可读性强和可移植性好	所编写的代码不能被计算机直接识别,需要经过解释或编译后才能执行

注意：不同型号计算机的机器语言和汇编语言往往是不能通用的,因为它们是一种面向机器的语言,更贴近于计算机硬件结构和特性,故也将二者统称为计算机的低级语言,而在20世纪50年代所出现的计算机高级语言,正是相对于低级语言的一种称呼方法。

由计算机语言的发展过程不难看出,越是高级的语言,其语言结构越贴近于人类的自然语言,但离计算机的硬件结构越远,对于使用人员的计算机专业知识要求也越低,因而伴随着计算机语言的飞速发展,也极大地推动了计算机的普及。

1.1.2　C语言的由来

C语言是一种面向过程的计算机高级语言,其雏形为1967年英国剑桥大学的Martin Richards推出的BCPL(Basic Combined Programming Language)语言。1970年美国AT&T贝尔实验室的Ken Thompson为了在UNIX操作系统上设计FORTRAN语言的编译器,在BCPL语言的基础上设计出B语言。由于其过于简单,功能有限,1972—1973年间,又由美国贝尔实验的D. M. Ritchie在B语言的的基础上设计出NB语言,即C语言。C语言发展简史如图1.1所示。

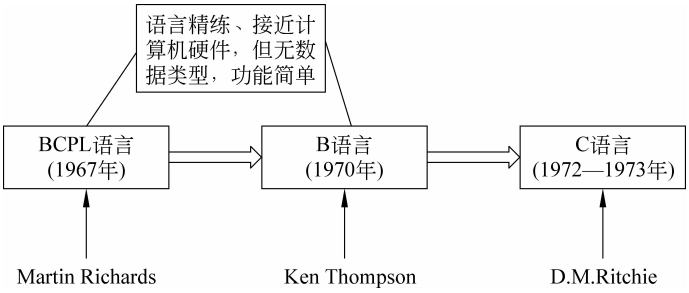

图1.1　C语言发展简史

1.2　为什么学习C语言

自20世纪90年代初,随着C语言在我国计算机行业的推广普及,目前绝大多数高等院校的理工科专业都开设了“C语言程序设计”课程。作为计算机基础平台课程之一,它一般设置在大一阶段,而学生在刚刚接触之初都会很迷茫,不知道为什么要学习C语言,又该如何学习。

C语言从出现、发展到标准的制定,再到目前的备受青睐。C语言的特点和超越其他编程语言的优越性无不展示着它强劲的生命力,在短短的几十年间,其成为备受欢迎的编程语言之一。图1.2是TIOBE分别在2014年和2015年公布的程序设计语言受欢迎程度的排

名情况，从中也可看出，C 语言始终处于前两位，不愧为编程界的常青树。下面就从 C 语言的特点及应用范围两个方面来剖析学习 C 语言的原因。

Aug 2015	Aug 2014	Change	Programming Language	Ratings	Change
1	2	^	Java	19.274%	+4.29%
2	1	v	C	14.732%	-1.67%
3	4	^	C++	7.735%	+3.04%
4	6	^	C#	4.837%	+1.43%
5	7	^	Python	4.066%	+0.95%
6	3	v	Objective-C	3.195%	-6.36%
7	8	^	PHP	2.729%	-0.14%
8	12	^^	Visual Basic .NET	2.708%	+1.40%
9	10	^	JavaScript	2.162%	-0.01%
10	9	v	Perl	2.118%	-0.10%
11	11		Visual Basic	1.781%	-0.23%
12	24	^^	Assembly language	1.760%	+1.11%
13	13		Ruby	1.416%	+0.17%
14	18	^^	Delphi/Object Pascal	1.407%	+0.49%
15	21	^^	MATLAB	1.232%	+0.50%
16	14	v	F#	1.232%	+0.14%
17	23	^^	Swift	1.179%	+0.51%
18	15	v	Pascal	1.138%	+0.09%
19	20	^	PL/SQL	1.137%	+0.35%
20	30	^^	R	1.010%	+0.49%

图 1.2　2014 和 2015 年度统计的编程语言的流行趋势排名图

（资料来源：http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html）

1.2.1　C 语言的特点

C 语言作为一种功能强大、使用灵活的面向过程的程序设计语言，具有诸多特点，可大致归纳为以下四点。

1. 语言简洁紧凑，运算符和数据类型丰富且使用灵活

C 语言中共包括 32 个关键字和 9 种控制语句，其基本结构紧凑，采用函数作为程序设计的基本单位，有利于实现程序的结构化和模块化。C 语言中丰富的数据类型和运算符，既可用来描述各种复杂的数据结构，又能进行各种数据运算的处理。对各运算符的灵活使用，能实现在其他高级语言中难以完成的运算。C 语言的代码书写格式自由，且对语法检查不严格，从而加大了程序设计的自由度。

2. 允许直接访问物理地址

通过 C 语言中的取地址运算符、位运算、指针类型等，可以直接对计算机硬件进行操

作，实现汇编语言的多种功能，而且为了满足计算机硬件编程的需要，C 语言还支持与汇编语言的混合编程，从而使其兼具高级语言和低级语言各自的优势。C 语言的这种双重特性，使它既可以进行底层系统程序的开发，又可以进行上层应用程序的开发。

3. 生成的目标代码质量高，程序执行效率高

一般情况下，针对同一问题用 C 语言编写程序，其生成目标代码的执行效率仅比用汇编语言编写的代码低 10%～20%，但 C 语言编程相对于汇编语言却要容易得多，且易于调试、修改，故许多以前用汇编语言处理的问题，现已改为用 C 语言来处理了，且 C 语言也比较适用于如云计算、并行计算等对执行效率要求比较高的环境。

4. 可移植性好

所谓可移植性是指程序从一个系统环境下不加或稍加改动即可以搬到另一个完全不同的系统环境中运行。可移植性好也是 C 语言得到广泛应用的一个重要原因。

1.2.2 C 语言的应用

通过前面对 C 语言发展及特点的介绍，可以将 C 语言的应用简单概括为以下五个方面：

(1) 用于操作系统、编译程序等系统软件的开发。

C 语言源自于 UNIX 操作系统的设计，它具有很好的可移植性及很强的数据处理能力，至今仍然是编写操作系统、某些系统软件的不二之选。另外，C 语言还是很多程序设计语言的开发语言，如 Java、C＃等。

(2) 用于对程序的性能、执行效率要求比较苛刻的环境，诸如网络服务器端底层、云计算、高效的并行计算等领域。

(3) 对现有 C 语言编写的程序维护、二次开发等。

(4) 管理信息系统、游戏等软件的开发。

(5) 单片机、嵌入式系统开发。

C 语言所具有的低级语言特点，非常适合于底层软件的开发，如利用单片机开发的温湿度控制、灾情报警等系统，还有大家非常熟悉的手机、PAD 等时尚消费类电子产品的应用软件、游戏等，大部分也都是以 C 语言为基础进行嵌入式开发的。

另外，针对正在高校就读的学生而言，C 语言还有一个不容忽视的“二级”意义，就是全国计算机等级二级考试，而且部分高校还会将该水平考试的分数与学生最终的本科学位挂钩，希望无论学习何种专业的学生，都能够在大学阶段掌握一种利用高级计算机语言编写、调试程序的基本技能。

在计算机无所不在的今天，只有更多地了解计算机，才能更好地利用它并让它更好地为我们服务。C 语言最大的好处就是它不仅可以“干大事”，而且也为我们打开了一扇了解计算机的窗口。所以，既然已经开设了这么一门颇具特色又有很好的发展趋势和应用前景的课程，就不要轻言放弃。应以顺利通过计算机二级考试为基础，以训练逻辑思维和程序设计能力为目标，在学习过程中挖掘出自己计算机方面的更大潜能，最终成为互联网＋时代的领军人物。

至于该如何学习，建议遵循“读程序—分析程序—写程序”三部曲。先从他人编写的程序入手，经过深入分析思考，完成自我解题到计算机解题思维方式的转化，最终一定能编写

出满足自己需要的程序。

1.3　认识 C 语言程序

1.3.1　计算机程序

人类进行各项事务的处理，都离不开对自然语言的有效组织。如今通过计算机信息化处理方式来解决日常事务，当然也需要对人与计算机间交流的媒介——计算机语言进行组织，从而使它能够按照用户需求完成一定的工作，这就形成了计算机程序。所谓计算机程序就是一组能够被计算机识别和执行的指令的集合。计算机的一切操作都是由程序控制的，离开程序，计算机将一事无成。因而，我们学习 C 语言的目的，也是围绕一定的功能需求，对其进行合理组织，最终编写出完成指定功能的 C 语言程序。下面通过几个简单的例子来了解 C 语言程序的基本结构。

1.3.2　C 语言程序的基本结构

【例 1-1】 要求在计算机屏幕上显示(输出)以下一行信息：

```
This is my first C program!
```

C 程序代码如下：

```
#include <stdio.h>                                  //编译预处理命令
int main()                                          //main 函数首部
{                                                   //main 函数体的开始标志
    printf("This is my first C program!\n");
    return 0;
}                                                   //main 函数体的结束标志
```

运行结果：

```
This is my first C program!
Press any key to continue
```

【例 1-2】 输入两个整数，求二者之中的较大者。

C 程序代码如下：

```
#include <stdio.h>                                  //编译预处理命令
int main()                                          //main 函数首部
{                                                   //main 函数体的开始标志
    int max(int x,int y);                           //max 函数的声明
    int a,b,c;
    printf("请输入两个整数:");
    scanf("%d,%d",&a,&b);
    c=max(a,b);
    printf("max=%d\n",c);
    return 0;
}                                                   //main 函数体的结束标志
```

```
int max(int x,int y)                                    //max 函数首部
{                                                       //max 函数体的开始标志
    int z;
    if(x>y)
      z=x;
    else
      z=y;
    return z;
}                                                       //max 函数体的结束标志
```

运行结果：

```
请输入两个整数:3,5
max=5
Press any key to continue
```

通过以上两个例子，可以看出 C 语言程序的基本结构如下。

(1) C 程序都是由函数构成的，必须有且只能有一个 main 函数。

由两个实例的对比发现，C 程序可以由一个函数构成(如例 1-1)，也可以由若干个函数构成(如例 1-2)，且其中必须有而且只能有一个 main 函数。可见，main 函数在整个 C 程序的构成中具有非常重要的地位。

(2) 无论 main 函数处于 C 程序的什么位置，C 程序的执行总是从 main 函数开始，并最终在 main 函数结束。

(3) 函数的基本结构。

函数是构成 C 程序的基本单位。函数可以是系统提供的(称为标准库函数或系统函数)，也可以是用户根据需要自己编写的(称为用户自定义函数)。如例 1-2 中的 printf 和 scanf 函数为系统函数，max 函数为用户自定义函数。

只有掌握了函数结构，才能顺利编写出自己想要的 C 程序。函数的基本构成如图 1.3 所示。

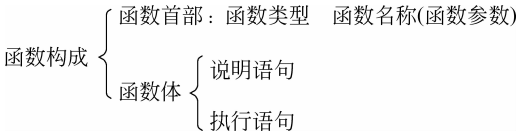

图 1.3　C 函数构成

① 函数首部。

构成函数首部的三部分依次说明该函数的类型、名称及函数参数(包括参数类型和参数名称)，函数参数位于函数名称后面的圆括号中；函数类型和函数名称之间至少要用一个空格隔开；函数参数若多于一个，各个参数都要有与该参数相对应的参数类型，且各个参数之间要用逗号隔开。如例 1-2 中 max 函数的首部：

```
int max(int x,int y)
```

其中，max 为函数名，它的类型由 max 前面的 int 指出，表明 max 函数是一个整型函数。圆

括号中有两个参数 x 和 y，各自用 int 说明参数类型，且参数间用逗号隔开。

根据需要，也允许一些函数不带参数，但参数外侧的圆括号不能省略，即函数名后面是一对空的圆括号。如例 1-1 中 main 函数的首部：

```
int main()
```

C 语言中将不带参数的函数称为无参函数，带有参数的函数称为有参函数。

② 函数体。

位于函数首部下面用一对花括号{}括起来的部分称为函数体，通常由说明语句和执行语句两部分构成。一般来讲，说明语句为该函数定义数据，执行语句则是为了完成函数的功能。如例 1-2 中 max 函数的函数体：

```
{
  int z;                    //说明语句:用于定义变量
  if(x>y)                   //执行语句:用于求得 x 和 y 之中的较大者,并存于变量 z 中
    z=x;
  else
    z=y;
  return z;                 //执行语句:用于返回变量 z 的值
}
```

注意：在 C 语言中允许函数体为空，此时的函数称为空函数，它什么也不做。在设计大型程序时会用到。如：

```
void empty()
{
}
```

(4) 分号是 C 语句必不可少的构成部分。

单独的一个分号，也是合法的 C 语句，称为空语句。

函数体是由一条一条的语句构成的，细心的读者会发现，每条语句后面都会有一个分号。大家可千万不要小看这个分号，如果丢掉它，C 程序的编译系统将会提示错误“syntax error：missing ';'”，程序不能顺利执行。

(5) 编译预处理命令。

当 C 程序中使用到系统函数时，需要通过编译预处理命令将该函数所在的标准库函数(扩展名为 h，也称为头文件)包含到本程序中，成为程序的一部分。

如例 1-1 和例 1-2 中用到了系统函数 scanf 和 printf，它们的相关说明信息包含在头文件 stdio.h 中，所以在程序的开始位置加入了编译预处理命令：

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

或

```
#include "stdio.h"
```

以上两种 #include 命令格式的区别是：

① 前者采用尖括号形式，称为标准方式。此时编译系统到存放 C 编译系统的子目录中

查找所包含的头文件(如 stdio.h)。

② 后者采用双引号形式,称为非标准方式。此时编译系统先到用户的当前目录(一般是用户存放 C 源程序文件的子目录)中查找所包含文件,若找不到,再按标准方式查找。

可见,如果所包含文件为系统库函数,应采用标准方式,以提高效率;如果所包含文件为用户自己编写文件,应采用非标准方式,或在非标准方式的双引号内明确给出所包含文件的路径。如:

```
#include "d:\eg1_1\file1.c"
```

表明所包含文件为 d 盘 eg1_1 文件夹下的文件 file1.c。

(6) C 程序的书写格式。

① C 程序的书写格式非常自由,允许一条语句写在多行,也允许多条语句写在一行。但为了增强程序的可读性,应尽量避免一行内书写多条语句,而且提倡根据程序内容的从属关系,采用缩进的书写格式。如:

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

② C 程序中的字母是区分大小写的。如变量定义语句 int a,A;,此时 a 和 A 是两个不同的整型变量。习惯上变量名、函数名等采用小写字母,常量名采用大写字母。

③ C 程序中所出现的标点符号(双引号中的部分除外),如逗号、分号、括号等均为英文状态下的符号。

(7) C 程序中的注释。

C 程序中的注释有两种格式:

① 单行注释,其格式为: //注释内容　　说明:其注释内容不能超出一行

② 多行注释,其格式为: /*注释内容*/　　说明:其注释内容可以分写在多行

注意:

① 注释内容可以由任何字符构成,注释可以出现在程序的任何位置,它们不参与程序的执行,只是为了帮助人们阅读理解程序。在用户编写程序时,提倡为程序做注释,这也是优秀程序员所必备的良好习惯。

② 多行注释符号/*和*/必须成对出现,且/和*之间不能有空格,单行注释符号的两个/之间也不能有空格。

1.4 运行 C 程序的步骤和方法

1.4.1 C 程序的一般运行步骤

例 1-1 和例 1-2 都是用 C 语言编写的程序。如何将 C 程序输入到计算机并执行呢? 如图 1.4 所示,一般需要经过以下几个步骤。

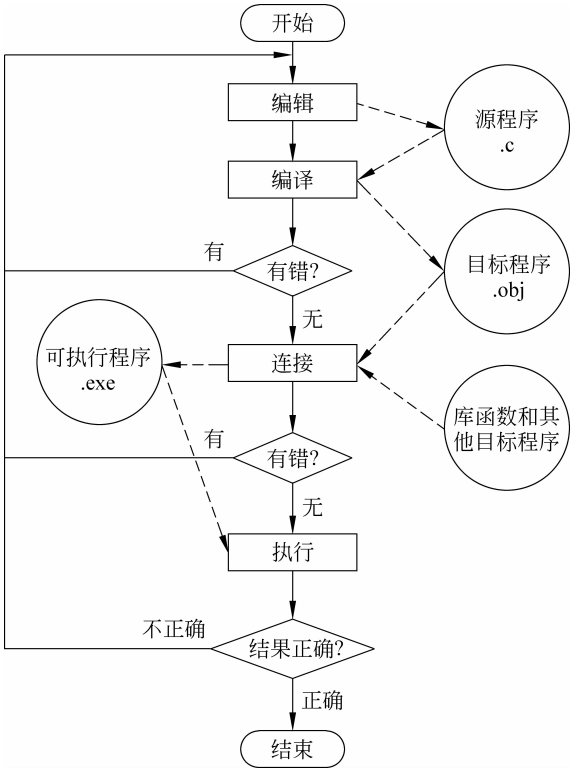

图 1.4　C 程序的上机步骤

1. 编辑源程序

按照 C 语言的语法规则编写程序，并保存成扩展名为 c 的文件，此文件就称为 C 源程序。

2. 编译

C 语言作为计算机高级语言的一种，其源程序是不能被计算机直接识别和运行的，必须用编译程序（也称为编辑器）将其翻译成二进制形式的目标程序，其扩展名为 obj。

3. 连接

将该目标程序与系统的标准函数库（如 stdio.h）及其他的目标程序连接起来，最终形成可执行的二进制程序，其扩展名为 exe。

4. 运行程序

在 C 程序设计过程中，不可避免地会出现一些错误，大体可以分为以下三类：

（1）编译时错误，又称语法错误，指因违背了 C 语言的语法规则而产生的错误，如前面提到的语句缺少分号等。

（2）连接时错误，指 C 程序中使用系统函数时，因函数名错、缺少该函数所属函数库的包含文件或包含文件路径错误等原因而导致的错误。

（3）运行时错误，指 C 程序通过了编译、连接，能够运行，但得到的运行结果与预期的结果不一致，如发生了除零错误、死循环等。

在程序执行过程中，无论出现上述哪类错误，都需要修改 C 源程序，再重新编译、连接、

执行，直到将程序调试正确为止。因此，C 程序从编写到运行成功，大多数情况下不是一次完成的，而是需要将上述步骤反复进行多次，这就是程序的调试过程。

1.4.2 使用集成开发环境 Visual C++ 6.0 实现 C 程序

由 C 程序的执行过程可见，C 源程序的执行必须有 C 语言编译工具的支持，早期常用的有 Turbo C 2.0、WinTC 等。目前广泛使用的是美国微软公司出品的一个可视化集成开发环境 Visual C++ 6.0，它不仅支持 C++ 语言，而且提供了对 C 语言的完全支持，并具有 Windows 可视化界面，操作简单方便。基于这些优点，本书以 Visual C++ 6.0 为 C 程序开发环境，结合例 1-1 来介绍 C 程序的具体执行过程。

1. 启动 Visual C++ 6.0 集成开发环境

单击【开始】|【所有程序】|【Microsoft Visual Studio 6.0】|【Microsoft Visual C++ 6.0】，进入 Visual C++ 6.0 集成环境，其界面如图 1.5 所示。

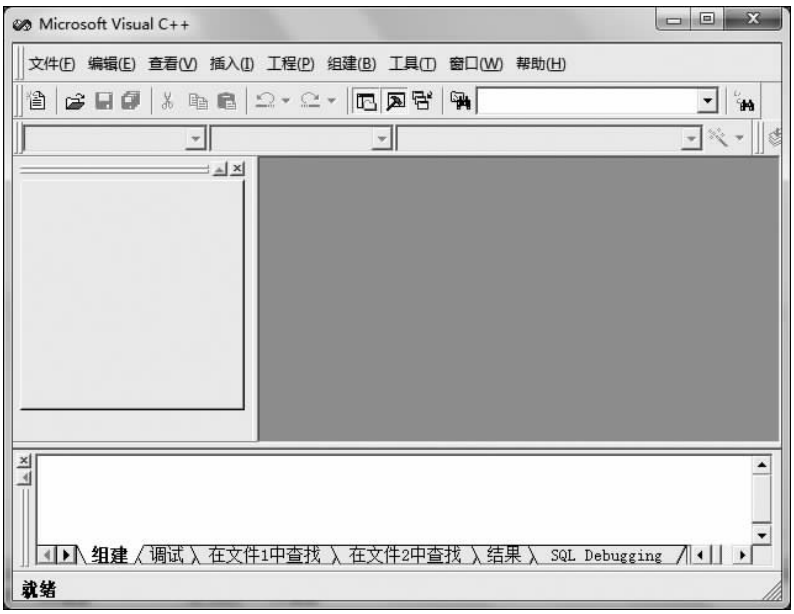

图 1.5 Visual C++ 6.0 集成开发环境界面

2. 使用集成开发环境 Visual C++ 6.0 实现 C 程序

(1) 创建和编辑 C 源程序文件

启动 VC++ 6.0 后，选择菜单【文件】|【新建】，系统将打开名为【新建】的对话框，如图 1.6 所示。单击该对话框的【文件】选项卡，在其列表框中选择【C++ Source File】选项，在右侧【文件名：】处输入此次建立的 C 语言源程序文件的名称，如 eg1_1.c，在右侧【位置：】处输入该文件的存储路径如 D:\EG1_1，也可以点击...按钮，打开如图 1.7 所示的【选择目录】对话框，选择文件的存储路径并单击【确定】按钮。

单击【确定】按钮后，进入 C 源程序的编辑界面，如图 1.8 所示。此时，可在空白的编辑区域，输入用户编写的 C 程序。在编辑窗口名称右侧，可能会出现【*】号，表明输入的程序代码还有未被保存的内容，一定要执行【文件】|【保存】或单击工具栏中的保存按钮，至此

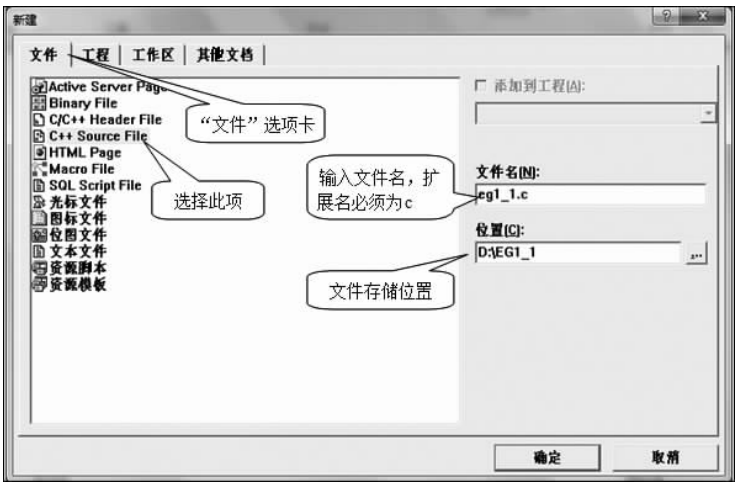

图 1.6　【新建】对话框

图 1.7　【选择目录】对话框

C 源程序编辑完毕。图 1.8 左侧部分为"工作空间"窗口，选择该窗口下方的 FileView 选项卡，可以查看当前工程项目中的相关文件信息。

注意：

① 在创建 C 源程序时，文件名一定要以 c 作为扩展名，否则系统将按默认扩展名 cpp 保存(这是 C++ 源程序文件扩展名)，而且文件名最好不要用中文。

② 建议用户在创建 C 源程序前，先在资源管理器的某个磁盘上新建一个文件夹，以利于 C 源程序相关文件的存放。

(2) 编译源程序

程序编写完毕后，选择菜单【组建】|【编译】或单击工具栏中的编译按钮，会出现如图 1.9 所示的对话框，提示【此编译命令要求一个有效的工程工作空间，是否创建?】。

此时必须选择【是(Y)】，从而为 C 源程序自动创建一个默认的工程工作区，同时在

图 1.8　编辑 C 源程序

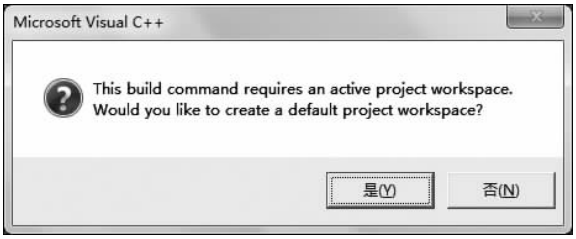

图 1.9　创建的 C 源程序第 1 次编译时出现的对话框

图 1.8 C 源程序编辑区下方的调试信息窗口中会显示如图 1.10 所示的编译报告：0 error(s)，0 warning(s)，表示源程序顺利通过编译，生成了 C 源程序 eg1_1.c 的目标程序 eg1_1.obj，无编译错误。

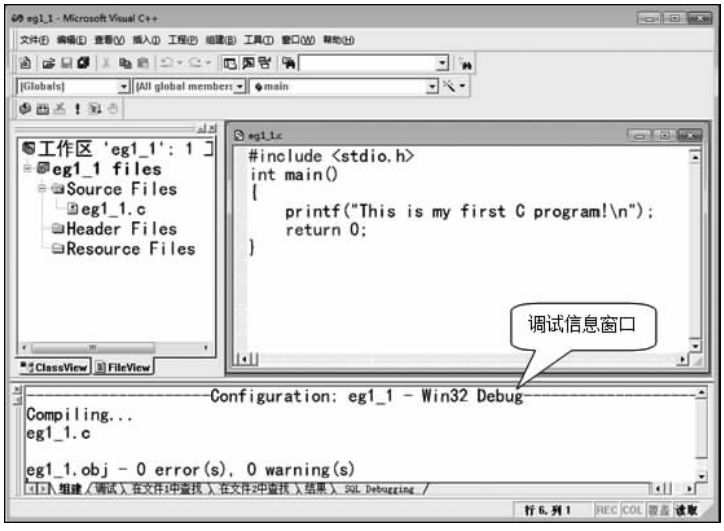

图 1.10　C 源程序编译成功

如果编译出错，出现如图 1.11 所示的编译信息，提示 C 源程序有 1 个 error(事实上并不一定只有一处 error 错误)和 0 个 warning。用鼠标拖动调试信息窗口右侧和下方的滚动条，可以查看程序出错的位置及具体的出错原因。

图 1.11　C 源程序编译出错

调试方法：无论在调试信息窗口中提示了多少条 error 类错误信息，一定要找到第一条 error 类错误，并用鼠标双击它。此时，在程序的编辑窗口中会出现一个粗箭头指向可能出错的程序行。根据出错原因提示信息(missing';'before return)，检查 C 源程序，发现在第 4 行末漏写了分号。改正后再进行编译调试，若还存在 error 类错误，重复上述过程，直至编译信息为："0 error(s)，0 warning(s)"，表示编译成功。

注意：在分析编译报错信息时，会发现编译系统所提示的错误位置与 C 程序中错误发生的实际位置可能是不一致的，但错误的实际位置一般在出错提示的上下行。

(3) 连接，构建可执行文件

程序通过编译后，选择【组建】|【组建】选项，此时调试信息窗口中会显示如图 1.12 所示的连接报告："0 error(s)，0 warning(s)"，表示源程序顺利通过连接，生成了可执行程序 eg1_1.exe，无连接错误。

如果连接出错，调试信息窗口会显示所有错误和发生错误的可能原因，但不能定位代码行。如出现图 1.13 所示的连接信息，提示源程序有 1 个 error(事实上并不一定只有一处 error 错误)和 0 个 warning。用鼠标拖动调试信息窗口右侧和下方的滚动条，可以查看程序出错原因。

调试方法：找到第一条连接错误，根据出错原因提示信息(unresolved external symbol_print)，关键就是 symbol 后面的函数名称，检查 C 源程序，发现在第 4 行书写 printf 函数名时漏写了 f。改正后再进行编译连接调试，若还存在连接错误，重复上述过程，直至连接信息为："0 error(s)，0 warning(s)"，表示连接成功。

(4) 运行可执行文件

成功构建.exe 文件后，选择【组建】|【! 执行】选项或单击工具栏中的 ! 执行按钮运行程序。如果运行期间没有出现错误消息提示，并得到预期的运行结果，则 C 程序的执行过

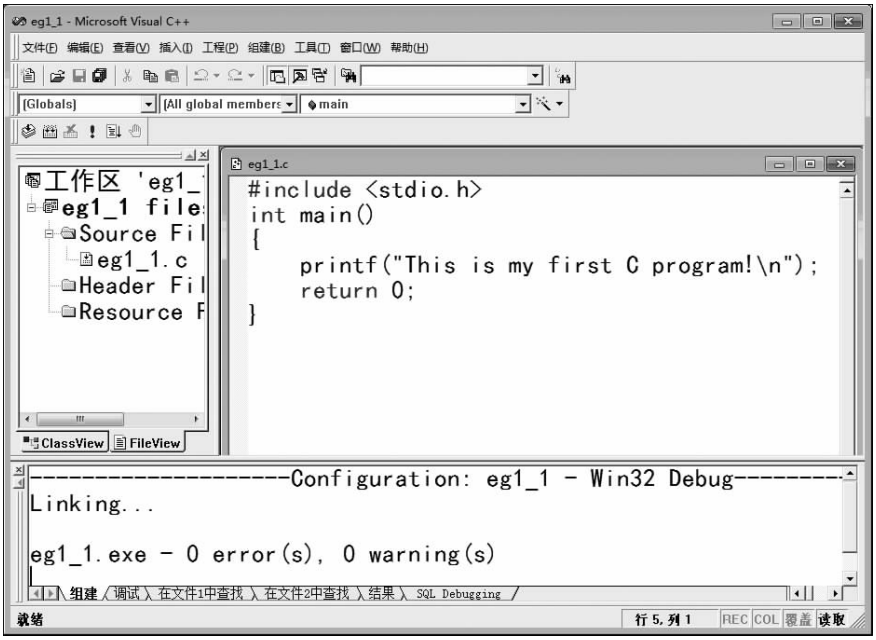

图 1.12 C 源程序连接成功

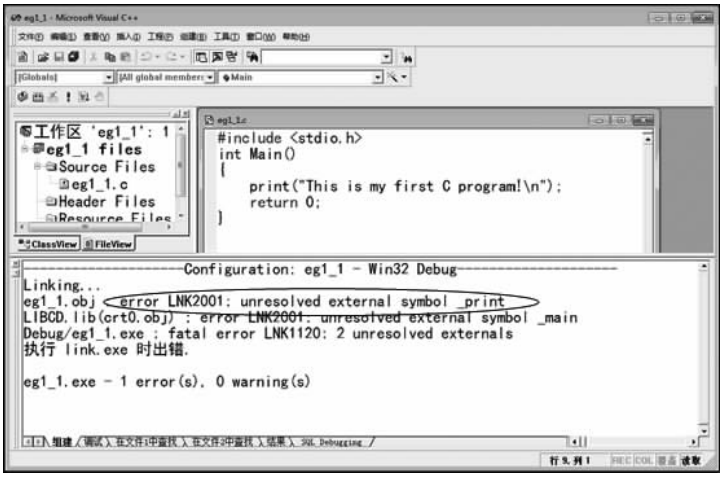

图 1.13 C 源程序连接出错

程圆满完成,按任意键关闭运行结果窗口即可,如图 1.14 所示。但如果出现错误消息提示或结果不正确,则表明存在运行时错误。该类错误也是不能定位代码行的,需修改源程序、再编译、连接和执行,直至得到用户满意的结果。运行时错误的具体调试方法见 1.4.3 节。

图 1.14 成功执行 C 源程序后的运行结果窗口

注意：若想创建第二个C程序，必须通过选择【文件】|【关闭工作空间】命令关闭上一个程序的工作空间，然后再重复上述过程。

1.4.3　C程序的调试方法

程序调试的目的在于发现和改正程序中的错误，使程序能够正常运行。在C程序调试过程中，语法错误和连接错误是C编译系统能自行检查的，此时用户只需根据系统给出的错误提示找到相应位置进行修改即可，但有些错误（如运行时错误），往往是因用户所编写程序存在逻辑问题而导致结果与预期结果不一致。C编译系统对于这类错误无能为力，需要用户使用一定的调试方法，查出错误位置并进行程序的修改，直至得到正确结果。

1. 程序执行到中途以便观察阶段性结果

方法一：使程序执行到光标所在的那一行暂停

（1）在需暂停的行上单击，定位光标；

（2）选择【组建】|【开始调试】|【Run to Cursor】或按快捷键＜Ctrl＋F10＞，程序将执行到光标所在行暂停，进入跟踪状态，如图1.15所示。暂停的语句前面会出现一个黄色的小箭头，表示该语句是下一条将要执行的语句。如果把光标移动到后面的某个位置，再按＜Ctrl＋F10＞，程序将从当前的暂停位置继续执行到新的光标位置，第二次暂停。

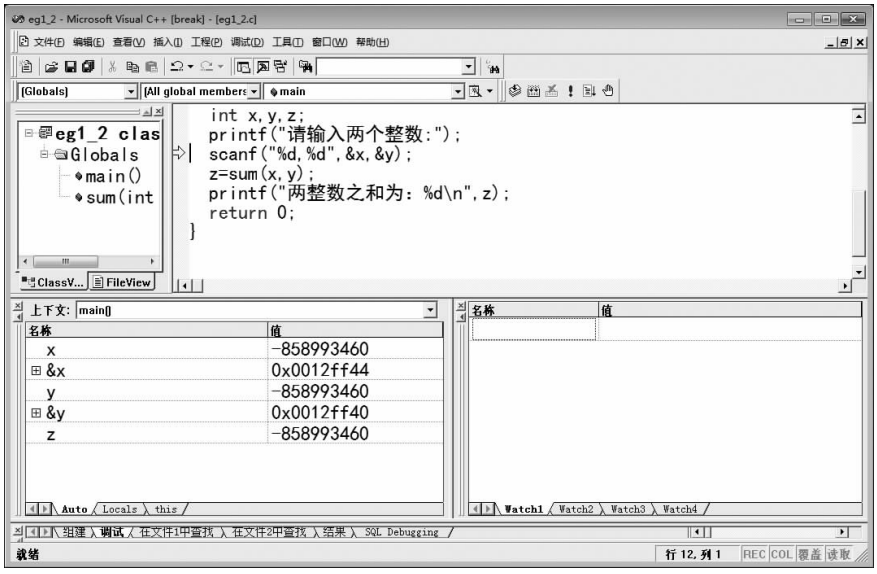

图1.15　C程序执行到光标所在行暂停

注意：程序一旦进入调试状态，原来的【组建】菜单会变成【调试】菜单。

方法二：在需要暂停的行上设置断点

（1）将光标定位在需设置断点的行上，单击工具栏的【编译微型条】中最右侧的按钮，或按快捷键F9设置断点。该行代码前面会出现一个红色的圆点，说明设置断点成功。

（2）选择菜单【组建】|【开始调试】|【GO】或按快捷键F5，程序遇到断点暂停，进入跟踪状态，如图1.16所示。

图 1.16　C 程序执行到断点所在行暂停

说明：

① 如果工具栏中未出现【编译微型条】，则在其他工具栏的任意位置右击，在弹出的菜单中选择【编译微型条】即可。

② 可以根据需要在程序中设置任意多个断点。按一次快捷键 F5 会暂停在第一个断点，再按一次 F5 键会继续执行到第二断点暂停，依次执行下去。

③ 程序中一旦设置断点，每次执行程序都会在断点上暂停。因此调试程序结束后应取消断点。方法是：先把光标定位在断点所在行，再单击【编译微型条】中最右侧的按钮，或按快捷键 F9，则取消断点。若再按 F9 则再次为该行设置断点。故该操作相当于开关，按一次是设置断点，按两次是取消断点。

如果有多个断点想全部取消，可选择菜单【编辑】|【断点】，屏幕上会显示【Breakpoints】窗口，如图 1.17 所示，该窗口下方将列出所有断点，单击【全部移除】按钮，将取消所有断点。

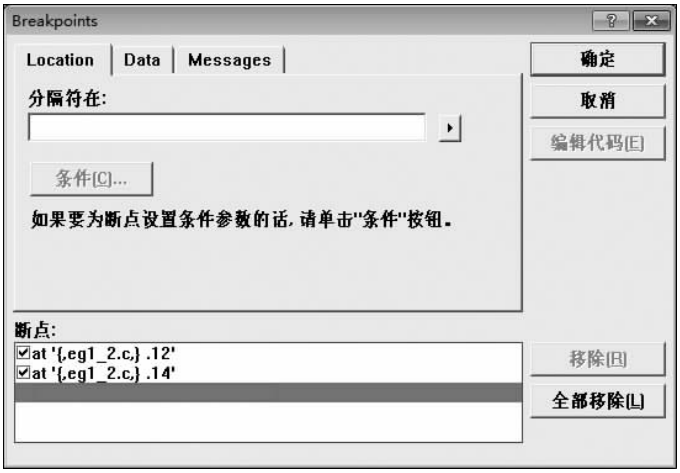

图 1.17　【Breakpoints】窗口

④ 设置断点的方法相比较于【Run to Cursor】(运行程序到光标处暂停)，更适用于调试

较长的程序。

2. 观察变量或函数返回结果的值

按照上面的操作，使程序执行到指定代码行暂停，目的是为了查看有关变量的值，以分析确定C程序结果不正确的原因。VC++ 6.0中有两个窗口用于查看变量的值，一是变量(Variables)窗口，一是监视(Watch)窗口，二者均可在【调试工具栏】中打开/关闭，如图1.16所示。如果没有出现【调试工具栏】，则在其他工具栏的任意位置右击，在弹出的菜单中选择【调试】即可。

(1) 变量窗口

变量窗口有多个选项卡。其中，自动窗口(Auto)用于显示在当前代码行和前面代码行中使用的变量，若有函数的返回值也会进行显示。局部变量窗口(Locals)通常用于显示位于当前上下文即正在执行的函数中的变量。通过局部变量窗口可以在程序执行过程中给变量赋一个新值，以更好地分析程序结果。

(2) 监视窗口

在监视窗口中可以添加要监视其值的变量或表达式。

3. 单步执行，进一步观察分析变量的值和函数的返回值

当程序执行到某个位置时发现结果已经不正确了，说明在此之前肯定有错误存在。如果确定了出错程序段的范围，就可以按照上面的步骤暂停在该程序段的第一行，再输入需要观察的变量名，然后单步执行，即一次只执行一条语句，逐行检查代码再配合观察变量值的变化，以确定错误发生的具体位置。

开始单步执行后，在程序的编辑窗口会有一个黄色箭头马上指向下一条待执行语句。单步执行的方式一般分为两种，一种是可以不进入函数内部的单步执行，另一种是进入函数内部的单步执行。

(1) 不进入函数内部的单步执行

方法：选择菜单【调试】|【Step Over】或单击【调试工具栏】中的或按快捷键F10。

(2) 进入函数内部的单步执行

若黄色箭头所指代码是一个函数调用语句，想进入该函数内部进行单步执行，其方法为：

选择菜单【调试】|【Step Into】或单击【调试工具栏】中的或按快捷键F11。

说明：对于不是函数调用的语句来说，键F10与键F11的作用相同；一般对系统函数不要使用进入函数内部的单步调试键F11。

4. 控制调试的步伐

(1) 如果程序的调试排错过程结束，想彻底终止调试状态，可选择菜单【调试】|【Stop Debugging】或单击【调试工具栏】中的或按快捷键<Shift+F5>停止调试。

(2) 在调试状态下，如果想全速继续运行程序，可以按键F5，程序会一直运行到结束或再次遇到断点暂停。

(3) 单步执行时，若当前黄色箭头所指为某函数内部代码，如果不想在函数内部逐条跟踪了，可选择菜单【调试】|【Step Out】或单击【调试工具栏】中的或按快捷键<Shift+F11>，就可运行出函数，直接返回到函数的调用处。

(4) 单步执行时，若当前黄色箭头所指为某循环体代码，如果不想反复跟踪这个循环了，可将光标定位于该循环之后的语句代码上，选择菜单【调试】|【Run to Cursor】或单击

【调试工具栏】中的{}或按快捷键<Ctrl+F10>，就可快速完成该循环，继续向下运行。

本小节主要是针对C编译系统无法检查的运行时错误，提供了调试程序的步骤与方法。对于初学者来讲，可能暂时还用不到这些相对较为复杂的调试方法，但随着学习的深入，所编写的程序也会越来越复杂，调试程序就显得尤为重要了。所以这部分内容有待读者在学习中反复实践，为了帮助大家更快、更好地查找出程序中存在的错误，附录E中列出了C语言中常见的编译、连接错误信息表，供大家参考查阅。

1.5 重点内容小结

重点内容小结如表1-2所示。

表1-2 重点内容小结

知 识 点	说 明
C程序的构成	函数是C程序构成的基本单位，C程序由一个或若干个函数组成，但必须有且只能有一个main函数。 复杂C程序也可由多个C源文件构成
C程序执行特点	无论main函数的位置如何，C程序总是从main函数开始执行并最终在main函数中结束
用户自定义函数的构成	函数由函数首部和函数体两部分构成。其中， 函数首部由函数类型、函数名称和函数参数三部分构成； 函数体以左花括号“{”开始，以右花括号“}”结束。花括号内由C语句构成，分号是C语句必不可少的构成部分
C程序的书写特点	为清晰起见，习惯上C程序的每行只写一条语句，而且提倡根据程序内容的从属关系，采用缩进的书写格式； C程序中的字母区分大小写，C程序以小写字母为主，习惯上常量名用大写字母表示
C程序的一般执行步骤	编辑(.c)→编译(.obj)→连接(.exe)→执行

简单C程序的基本架构	#include <stdio.h>
	int main()
	{
	return 0;
	}
	在该结构上添加处理对象——数据和功能操作，就可实现简单的C程序

习 题

一、单选题

1. 以下叙述中错误的是(　　)。(二级考试真题)

(A) C语言中的每条可执行语句和非执行语句最终都将被转换成二进制的机器指令

(B) C 程序经过编译、连接步骤之后才能形成一个真正的可执行的二进制机器指令文件

(C) 用 C 语言编写的程序称为源程序，它以 ASCII 代码形式存放在一个文本文件中

(D) C 语言源程序经编译后生成后缀为.obj 的目标程序

2. 以下叙述正确的是(　　)。

(A) C 语言规定必须用 main 作为主函数，程序总是从此开始执行，并在该函数结束

(B) 可以在程序中由用户指定任意一个函数作为主函数，程序将从此开始执行

(C) C 语言程序将从源程序的第一个函数开始执行

(D) main 的各种大小写拼写形式都可以作为主函数名，如：Main、MAIN 等

3. 下列叙述中错误的是(　　)。(二级考试真题)

(A) C 程序可以由多个程序文件组成

(B) 一个 C 语言程序只能实现一种算法

(C) C 程序可以由一个或多个函数组成

(D) 一个 C 函数可以单独作为一个 C 程序文件存在

4. 以下叙述中正确的是(　　)。(二级考试真题)

(A) C 语句必须在一行内写完

(B) C 程序中的每一行只能写一条语句

(C) C 语言程序中的注释必须与语句写在同一行

(D) 简单 C 语句必须以分号结束

5. 以下叙述中错误的是(　　)。

(A) C 程序在运行过程中的所有计算都以二进制方式进行

(B) C 程序在运行过程中的所有计算都以十进制方式进行

(C) 所有 C 程序都需要编译连接无误后才能运行

(D) 计算机能直接执行的程序是二进制可执行程序

二、填空题

1. 一个 C 程序一般由若干个函数构成，其中必须且只能有一个_____函数。

2. 一个函数由两部分构成，分别是_____和_____。

3. 一个函数体以_____开始，以_____结束。

4. C 语言源程序文件的扩展名为_____，经编译后生成的二进制目标文件扩展名为_____，经过连接后生成的二进制可执行文件扩展名为_____。

5. C 语言中的单行注释说明必须以_____开头，多行注释说明必须以_____开头，以_____结束。

三、程序设计题

1. 参考本章例题 1-1，编写 C 程序，输出以下信息：

```
*   *   *   *   *   *   *   *   *   *   *   *
*            Welcome to learn C language            *
*   *   *   *   *   *   *   *   *   *   *   *
```

2. 参考本章例题 1-2，编写 C 程序，从键盘上输入三个整数分别给 a、b、c，输出三者之和。

第2章

C程序设计基础

【内容导读】

本章以“数据结构＋算法＝程序”为主线。首先介绍了数据类型、数据的表现形式、数据间的基本运算。其次，介绍了算法的概念、特性及描述方法。最后，给出程序的三种基本结构及结构化程序设计方法，为简单的C程序设计奠定了基础。

【学习目标】

(1) 掌握基本数据类型的使用；

(2) 掌握变量和符号常量的定义及使用方法；

(3) 掌握算术、赋值、自增自减、逗号等运算符的优先级、结合性和各类表达式的求值规则；

(4) 了解算术表达式中不同类型数据间的转换和运算规则；

(5) 理解算法的概念、特性及描述方法；

(6) 理解结构化程序的三种基本结构及程序设计方法。

2.1 C数据类型概述

数据作为计算机程序处理的对象，是程序设计必不可少的构成部分。而与数据密切相关的就是类型、生存期、作用域等重要属性，下面主要讲解其中之一——数据类型，首先从总体上认识C语言的数据类型及设置类型的目的。

2.1.1 C数据类型

C语言提供了丰富的数据类型，其常见类型如图2.1所示。

其中：

(1) 基本类型是系统预先设定的，用户可以直接使用，分为数值形式的整型和实型及非数值形式的字符型。

(2) 构造类型是在基本类型的基础上，根据用户需要所定义的数据类型，包括数组、结构体等类型。