第 1 章　概　　述

计算机与信息技术的快速发展与普及应用，使得更多的问题可用计算机得到有效的解决。越来越多的人希望了解与程序设计相关的知识和技术，以便利用程序更高效地处理自己工作中遇到的问题。本章简要介绍与程序设计和 C 语言有关的一些基础知识，说明 C 语言程序的基本结构框架，并完成设计简单 C 语言程序的环境准备。

1.1　程序设计基础

1.1.1　问题的求解过程

简单地说，用计算机程序处理一个问题，就是把解题的技术和方法描述成计算机可以执行的一系列操作，并实施这些操作步骤，通常需要包含如下过程：

① 分析问题，建立数学模型；

② 设计解决问题的方法，即设计算法；

③ 确定程序的流程；

④ 编制和调试程序；

⑤ 运行程序，得到结果。

上述过程的表述并不十分严格，也略显抽象，以下将通过一个例子来说明。

我国古代数学家张丘建在《算经》一书中提出了一个“百鸡问题”，描述为：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。以百钱买百鸡，问鸡翁、母、雏各几何？以下是为了在计算机上求解此问题所要经历的主要过程：

（1）建立数学模型

若用 x、y 和 z 分别表示鸡翁、母、雏的数量，可得到如下的数学关系：

$$\begin{cases} x+y+z=100 \\ 5x+3y+z/3=100 \end{cases}$$

这是一个不定方程。因为 x、y 和 z 都是整数，并有一定的取值范围，这就决定了用计算机来描述这些数据时所采用的数据类型，也确定了存储这些值所占用的存储空间大小。这两个等式建立了数据之间的关系，而这些相互依存的数据及其关系构成了数据结构。

（2）设计求解算法

此问题较简单，可以采用“穷举法”来解决。由于 x、y 和 z 都是 0～100 之间的整数，可以将(0,0,0)至(100,100,100)之间的任意一组整数值代入这两个等式进行测试。如果某一组值使二者都成立，就得到了原问题的一个解。很明显，这样的算法不适合手工计算，它说明了计算机求解算法与手算解法之间是有一定差异的，尽管存在简单的方法会更好。

（3）描述算法的流程

长期以来，人们使用了各种技术和工具来描述一个算法（程序）的轮廓，主要是说明实现该算法所要经历的操作步骤，包括伪语言、程序流程图、PAD 图，以及建模工具 UML（Unified Modeling Language，统一建模语言）图等。为了简单起见，本书中使用程序流程图作为描述工具。

（4）编制程序

这是学习计算机语言的主要目的。所谓计算机语言就是为了描述计算机操作步骤而制定的一整套标记符号、表达格式和语法规则。“程序”（Program）就是指为实现特定目标或解决特定问题而用计算机语言编写的命令序列集合，或者说是为实现预期目的而执行的一系列语句和指令，其核心是对数据的描述和处理。

计算机语言分为低级语言和高级语言两类。最低级的语言是机器语言，它能被计算机直接识别，而其他语言则不能。因此，几乎每一种语言都配有一个相应的“翻译”程序，称之为“编译程序”，用于将自己的代码翻译为机器指令。

目前仍在广泛使用的一种低级语言是汇编语言，它是通过直接将机器指令符号化后形成的语言，功能强，且代码效率高，但更接近于计算机硬件，编程困难，软件开发效率低，不易被非专业人员所接受。相比之下，高级语言是为了更好、更直观地描述算法，方便程序设计而发展起来的通用型语言，如 BASIC、FORTRAN、COBOL、PASCAL、C、C++、JAVA、C#和 PYTHON 等。高级语言的语法比较接近数学表达式描述，容易理解和编程。大部分高级语言支持结构化、模块化的程序设计方法，新兴高级语言还提供了对面向对象设计技术的支持，被广泛地用于科学计算和事务处理等诸多方面。

计算机程序可以大体上归结为系统程序和应用程序两大类。其中，“系统程序”是指为维持计算机和网络系统正常运行而设计的程序，典型的系统程序是操作系统中所包含的程序。相对地，“应用程序”是指为某些特殊目的和解决某些特定问题而编制的程序，如数据库管理、绘图、文字处理以及图像处理等。各种程序及其相关的技术文档的总称即为“软件”。

在解决一个问题时，原则上可以使用任何一种语言来编制程序，但每种语言都有自己的特性和适合的领域，所编制程序的效率也有较大差异。长期以来，人们期望能有一种既具有高级语言特点，又兼有低级语言的高效和强硬件操作能力的语言，这导致了 C 语言的诞生。

（5）运行程序

程序编制后，需要进行多方测试以修改其中存在的错误。测试和证实程序的正确性也是一个专门的技术领域。通常，需要使用多组数据集来运行程序进行观察，直到没有错误为止。确信程序无误后，再将其真正投入运行，得到所需结果。

1.1.2 算法及其描述

程序设计主要包括数据结构设计和算法设计。“数据结构”用于指定数据及其相互关系，“算法”则是对特定问题求解步骤的一种逻辑描述，二者密不可分。合适的数据结构可以高效地支持插入、查找和排序等基本操作，有利于简化算法的设计，而每种事务的顺利处理则要依赖有效的算法。

著名计算机科学家沃思（Niklaus Wirth，图灵奖获得者）曾经提出了一个公式，即“数据结构+算法=程序”。实际上，一个程序除了以上两个关键要素外，还应当采用适当的程序设计方法进行设计，并且用一种计算机语言来表示。因此，算法、数据结构、程序设计方法和语言工具是一个程序员必须具备的四个方面知识。

一些常用的基本数据结构和算法可以在“数据结构”、“算法设计与分析”等知识领域中找到，只有了解这些基础知识，才能借助它们构建更复杂有效的解题方法。在实际设计一个算法时需要注意到如下问题：

（1）有穷性

这是指一个算法必须在执行有限个步骤之后结束。

（2）确定性

任何算法的每一步必须被确切定义，不能出现模糊性、多义性或不确定性，对于相同的输入必须得到相同的结果。

（3）可行性

算法的每一步都是能够实现的，或者说是可操作的。

（4）有输出

在算法开始时，可以没有输入量或者有多个输入量，但算法执行完毕后，必须有一个或多个输出量。

设计一个好的算法通常要考虑到正确性、易读性、健壮性、高效率和低存储量等诸多方面的因素。由于本书以介绍语言的语法和程序构建方法为主要目的，较少涉及复杂的算法，一般也只是给出算法的简单流程或语言描述。

采用流程图描述算法时要使用一些标准化的符号。图 1.1 列出了国标 GB1526-89 所推荐的一套流程图标准化符号，这些符号与国际标准化组织 ISO 提出的流程图符号一致。

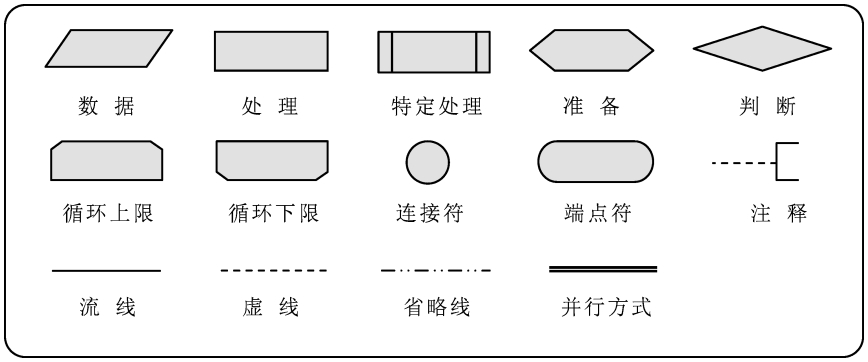

图 1.1　流程图符号

图 1.2 利用流程图描述了 1.1.1 节中的“百鸡问题”的求解算法。注意到流线的箭头画法为：若方向向右或向下，可以标出箭头也可以不标，否则必须标出箭头。开始和结束符号不是圆角矩形。此外，还可以采用循环符号来描述循环结构。

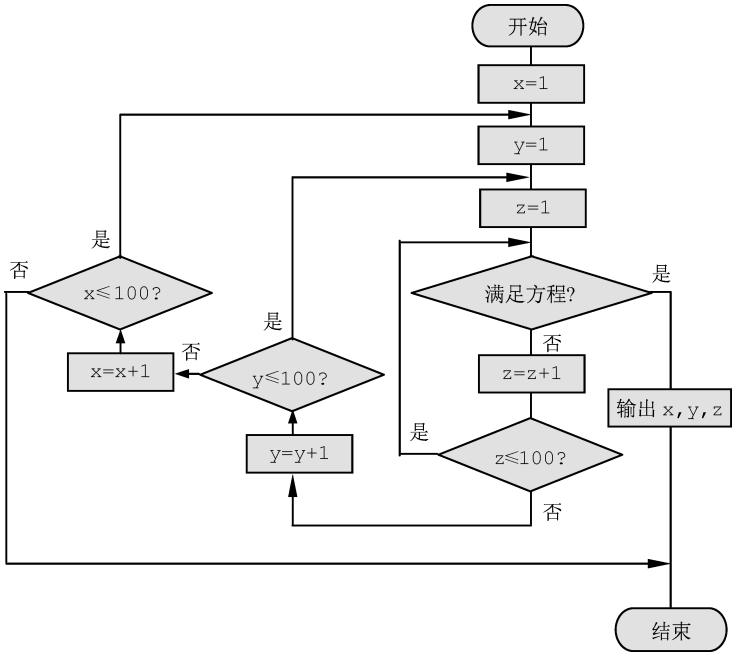

图 1.2　百鸡问题的流程图

1.1.3　模块化与结构化

在软件设计的发展过程中，模块化和结构化是已被证明的有效设计技术和方法。

1．模块化

模块化的概念在计算机软件设计领域中已经沿用了 40 多年，是处理复杂问题所应采取的关键技术，也是使得软件能够被有效管理和维护所应具备的关键特性。

在一个复杂的软件中，任务被划分成若干个可单独命名和编址的部分，这些部分被称为“模块”，它们之间通过有目的的相互连接组成满足应用需要的软件系统。从整体上看，可以将程序中完成各种处理的程序段（模块）看成是构成整个程序的“部件”。

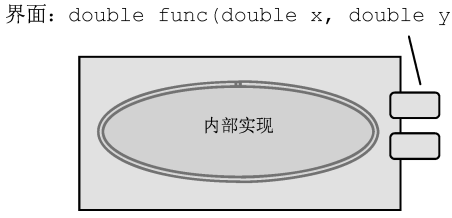

图 1.3　公开的“界面”和隐藏的“内部实现”

模块的基本特征是抽象和实现信息隐藏，分为模块界面（接口）和模块体两部分，前者是对外的“窗口”，后者是模块的具体实现细节，对外是不可见的，如图 1.3 所示。对模块体的修改可以不影响与之相连接的其他模块。

2．结构化

在一个模块的内部要实现其应该完成任务的细节，即规定机器在实施时应遵循的处理流程。在软件设计发展的初期，程序中可以随意根据处理的需要确定流程的走向，从一个局部结构跳转到另一个局部结构，这种跳转称为“无条件转向”或“转移”。早期的高级语言如 BASIC 等就依赖这样的指令（语句）。

1963 年，一些计算机专家提出，程序中大量、无限制地使用无条件转移语句会使程序结构杂乱无章，各部分之间会因为转移语句的牵连而使可读性变差且不易修改，程序的复杂性与所用的无条件转向语句的数量成正比。

1966 年，C. Bohra 提出了“任何程序均可由顺序、选择和循环这三种结构组合而成”的观点并得到证明。由此，人们逐步对结构化程序设计有了统一的认识。作为程序的基本组成元素，顺序结构、选择结构和循环结构具有共同的特点，即只有一个入口和一个出口，这就避免了因随意转移流程的走向而破坏这些基本元素。对于复杂的问题，坚持采用模块化、自顶向下逐步求精的设计原则，可以使程序结构清楚易读，容易维护。图 1.4 说明了三种基本结构的流程。

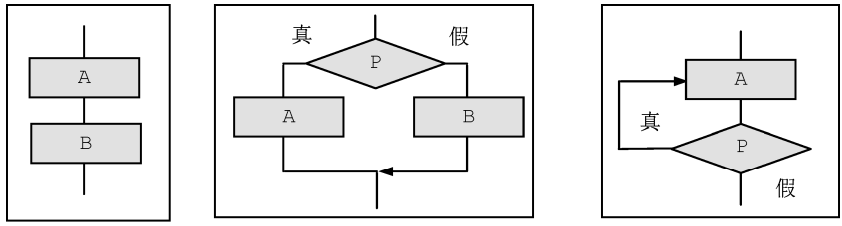

图 1.4　三种基本结构

目前，软件设计领域所使用的高级语言都支持模块化和结构化的程序设计方法。

1.2　C 语言及其特点

C 语言是国际上极为流行且被广泛应用的一种高级语言，既可用于编制系统软件，也可用来编写应用软件。

1.2.1　C 语言的产生和发展

C 语言起源于 ALGOL 60 语言。ALGOL 60 语言曾在 20 世纪 60 年代流行一时，但因距离系统硬件较远，难以编写系统软件而逐渐淡出程序设计领域。后来，该语言逐渐发展成二个分支，分别是 PASCAL 语言和 C 语言，且后者的发展速度相对较快。此分支的最初成果是英国剑桥大学在 1963 年研制出来的 CPL 语言，随之，由该校的马丁·理查德（Martin Ritchards）改进为 BCPL 语言并于 1967 年发表。BCPL 语言是一种无类型的系统程序语言，它的基本数据类型是机器字，使用较多的指针和地址运算，这与其他常见的各种高级语言有着明显的差异。1970 年，美国贝尔（Bell）实验室的肯·汤姆逊（Ken Thompson，UNIX 操作系统的主要研制者）以 BCPL 语言为基础，又发表了一种新的 B 语言，并用汇编语言和 B 语言写成了 UNIX 的初版。B 语言接近机器硬件，但较为简单，而且无常用的数据类型。为了克服 B 语言的局限性，C 语言应运而生。

C 语言是由贝尔实验室的丹尼斯·里奇（Dennis Ritchie）设计的，设计的初衷是为了更好地描述和实现 UNIX 操作系统。1973 年，肯·汤姆逊和丹尼斯·里奇合作，将 UNIX 的 90%以上的代码用 C 语言改写，形成了 UNIX 第 5 版，但此时的 C 语言并没有引起人们太多的注意。随着 C 语言的发展，1977 年出现了不依赖于具体机器的 C 语言编译文本，而 C 语言的真正定义是在 1978 年，在由布莱恩·科尼汉（Brian Kernighan）和丹尼斯·里奇（Dennis Ritchie）所著的 *The C Programming Language* 中被阐述清楚。这是一本影响深远的名著，实际上成了后来的 C 语言标准。

1983 年，美国国家标准化协会（ANSI）整理和扩充了当时的各种 C 语言版本，并制定了一个 C 语言标准，即 83 ANSI C。随后，在 1987 年对其做了重新修订，制定了 87 ANSI 标准 C。1989 年，ANSI 又公布了一个新的 C 语言标准 ANSI X3.159-1989，国际标准化组织 ISO 在 1990 年接受了该标准，使其正式成为国际标准 ISO/IEC 9899：1990，简称为 C90。随后，ISO 在 1995 年和 1999 年分别又对该标准进行过两次修订，为 C 语言增加了面向对象的特征，并命名为 ISO/IEC 9899：1999，简称为 C99。不过，由于通常的 C 语言编译器仅支持 C90，故本书的语法介绍也主要以 C90 标准为基础。

近年来的 C 语言发展十分迅速，以面向对象技术和 C 语言语法相结合的 C++语言在软件设计领域占有举足轻重的地位，其应用的广泛性也已逐渐超过了传统的 C 语言，但它更适合开发大型应用系统。在一些涉及硬件开发的系统和嵌入式应用中，C 语言仍具有重要地位并被广泛应用，传统的以汇编为开发语言的应用也基本由 C 语言取代。同时，作为了解程序设计技术的一种基础语言，C 语言也具有独特的优势，因为大量的新兴语言如 Java、C#等都由 C 语言发展而来，与 C 语言有着大量一致的语法现象和规则。因此，学好 C 语言和程序设计方法也将会为以后的进一步学习奠定良好的基础。

1.2.2　C 语言的主要特点

与其他高级语言相比，C 语言有诸多独到之处，主要可以归纳成如下一些特点。

1．语言简洁、书写自由

C 语言是一种很小的语言，具有精心选择的控制结构和数据类型，摒弃了一切不必要的成分，从而使其规模缩减到最小，代码简洁而高效。C 程序的书写也几乎不受什么限制，可以在一行上书写多

个语句，也可以把一个语句写在多个行上。当然，这种灵活性可能使程序缺少一种可以遵循的标准，因此，学习时应尽量注意程序书写的“规范性”，这一点可以根据本书所提供的示例程序去体会。

2．运算符丰富

运算符的多少体现了语言对数据的加工能力强弱。C 语言提供了极为丰富的运算符，共 34 种（参见附录 B），这使 C 语言可以直接实现其他语言中很多难以实现的运算，同时也提高了语句的能力。

3．数据类型丰富

C 语言提供的多种数据类型使程序员可以灵活、方便地构造各种复杂的数据结构和设计适应各种问题的算法。表面上，C 语言支持的数据类型与 PASCAL 语言接近，但 C 语言中的指针类型比 PASCAL 更灵活，也具有更强的操作能力。

4．结构化良好

C 语言提供了典型的结构化控制语句，是理想的结构化语言，而且 C 语言用函数来组装程序，进而支持程序的模块化。

5．语法限制不严格

这是 C 语言存在争议的一个特点，这是因为，语法限制不严给程序设计带了灵活性，但也对程序的安全性有“不良”影响。例如，最为典型的数组超界问题就是语法限制不严的产物，不论程序使用了数组的多少元素，是否与定义吻合，C 语言的编译器都不会给出错误通知，因为 C 编译器不做数组边界的检查。如果在设计时不能确保引用的正确性，超界的数组元素有可能导致灾难性的后果，使程序彻底崩溃。

6．硬件操作能力强

C 语言具有低级语言所具有的特殊能力，体现在对内存地址的直接操作和位运算。通常，这是低级语言所具有的两种能力，以便更直接地操作机器硬件。作为高级语言的 C 也具有这两种能力，或者说具有低级语言的功能特征，因此有人称其为“中级语言”。

此外，从生成代码的效率来看，C 程序明显优于其他高级语言。借助 C 语言的低级语言能力，一些程序员正在用它代替汇编语言来书写系统程序，这不仅使程序的研制周期得以缩短，也使程序具有较高的可移植性。

当然，上述特点只是使用者对 C 语言的一般性评价，更深刻的体会需要在进一步学习和比较之后才能得到。

1.3 C 语言程序的基本结构

这里给出几个简单的 C 语言程序，目的是建立对 C 语言程序的感性认识，并能理解书中使用的程序结构。阅读时应注重代码后的正文解释，不必仔细研究程序代码中有关输入、输出函数的格式等细节，后续章节中会对这些内容做进一步的讨论。

例 1.1 在屏幕上输出“Hello world!”的 C 语言程序。

最简单的 C 程序仅由下面的代码实现：

```
void  main( )
{
}
```

这是一个完整的 C 程序，仅由一个模块组成，但它不做任何工作。将其编译成可执行文件并运行时，只相当于做了一次空操作。

这个基本的程序框架也可以写成如下形式：

```
int  main( )
{
   return  0;
}
```

尽管此程序还不能输出题目要求的文字，但可以说明 C 语言程序的一些基本特征。

1．由函数组成程序与 main 函数

C 程序是由函数组装而成的，一个完整的 C 程序中必须且只能有一个名字为 main 的函数，称作“主函数”。若程序简单，可以只由这一个函数构成完整的程序，此例即如此。不管组成一个 C 程序的函数有多少，总是从 main 函数的第一个语句开始执行。

2．函数的基本结构

通常，每个函数名之前有一个数据类型，用来表示函数的值是什么类型。main 函数的数据类型可以是 void 或 int。一般应用中的 main 函数仅执行一些处理指令，不带回任何计算结果，可将其类型指定为 void。如果采用 int 作为函数的数据类型，可以将“return 0;”置于最后一行，其作用是向操作系统表明程序正常执行结束。为了避免引入过多的成分，本书主要使用第一种形式，即 void 类型的 main 函数。

代码中跟在数据类型之后的是函数名，除了 main 函数的名字为固定写法以外，其他函数都由设计者自己命名。每个函数名之后都要有一对圆括号，这是函数的标志。函数名和圆括号之后的部分称为“函数体”，置于括号{　}内，这是书写函数代码的地方，用于说明函数所要做的全部工作。

与一些简单的语言如 BASIC 等不同，无论构成程序的代码有多少，不能仅由语句构成 C 程序，必须将代码置于函数框架内。或者说，C 语言程序的基本组成单位是函数而非语句。

由于 C 语言对书写的要求并不严格，因此上述程序中的各部分也可以写在一行内，但这并不是一种可取的书写方式，如：

```
void  main( ) { }                    /* 不可取的书写方式 */
```

下述程序在基本框架中增加了一个语句，使其能够按要求输出文字“Hello world!”。

```
#include <stdio.h>
void  main( )
{
  printf("Hello world!");          /* display a string. */
}
```

以下是程序中所添加内容的含义。

1．输出语句

```
printf("Hello world!");
```

这是一次“库函数”调用。C 语言提供了大量预先设计的函数，以完成一些常见的事务性处理工作，如输入和输出等。只要了解它们的功能和使用方式（格式），就可以直接使用而不用自己去编写，这些函数称为“库函数”。此处的 printf 就是一个用于输出数据的库函数，这样的语句也称为“输出语句”。

2．文件包含指令

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

添加此行代码是因为使用了库函数 printf。通常，C 语言要求对使用者声明函数的格式。这里的格式就是接口或界面。对于自己定义的函数，可以在第 5 章中了解说明方法，而对于每一个库函数，其格式都记录在系统提供的一个文件里。例如，函数 printf 的格式记录在文件 stdio.h 中，或者说 printf 函数定义于文件 stdio.h。C 语言允许使用命令#include <stdio.h>来对库函数 printf 进行声明，以便让编译器从文件 stdio.h 中查找并了解函数 printf 的格式。因此，了解每个库函数所属的文件（称为“头文件”）是必要的任务，并注意在程序开头加上如下代码，称为“头文件包含”：

```
#include  <头文件名>
```

如果几个库函数定义于同一个头文件，只要包含该文件一次就可以了。

提示

stdio.h 中的 std、i 和 o 分别来自单词 **st**a**nd**ard、**i**nput 和 **o**utput，即标准输入输出；字符 h 来自单词 **h**ead，故称为头文件或头部文件。

3．注释

```
/*…*/
```

在这一对符号中间的内容称为“注释”。注释是对程序中的代码、变量等所做的说明，在程序编译时，编译器会舍弃注释而不做任何处理，执行时自然也不会引起任何机器操作。对程序的注释有助于提高程序的可读性，是帮助阅读者理解源程序代码的非常必要的部分。

C 语言程序的注释非常灵活，可以在任何允许插入空格的地方插入注释，但正确的做法通常是用单独的行添加必要的注释，不严格要求时也可以加在行末。限于篇幅，本书的注释以中文形式在行末给出，用于解释所在行代码的功能、含义和应注意的事项。注意，应尽量以某种方式对齐注释，以便于阅读。

注释不可嵌套。

工程

实际项目中通常这样对程序进行注释:

（1）在程序（文件）头，添加有关程序名、功能、作者、目的、用途、修改历史等的注释;

（2）在函数前，添加有关函数的功能、参数、返回值和副作用等的注释;

（3）在变量定义前（或后），说明变量的含义;

（4）在函数体开头，解释算法思路。

工程项目中一般对注释的方法有严格的要求，甚至对注释的数量也非常重视，因为一旦缺少注释，代码将很难理解和维护。另外，应该在编写程序的同时为代码添加适当的注释，而不是在项目开发结束后一次性地添加注释。总之，注释是软件的重要组成部分，而正确添加注释也是软件项目开发的重要工作之一。

通常，一个完整的程序总会包含数据输入、处理以及数据输出等几个部分。

例 1.2　从键盘输入两个整数，输出它们的和。

```
#include <stdio.h>
void  main( )
```

```
{
   int  x, y;                    /* 变量定义 */
   int  z;                       /* 变量定义 */
   scanf("%d,%d", &x, &y);       /* 输入两个整数，数据间以逗号分隔 */
   z = x+y;                      /* 求 x 和 y 的和，并存入 z */
   printf("sum = %d", z);        /* 输出结果，即 z 的值 */
}
```

此程序是一个简单的整数加法器，运行时可以接收用户输入的两个整数，计算并输出它们的和。此例反映了 C 程序中一个函数体内的主要结构，即：

```
{
   定义和声明部分
   可执行的操作部分
}
```

通常，所有变量定义应写在函数体的开头，后面的部分才是对它们的处理。

工程

将不需要的代码暂时注释掉而不是删除，或许，一会儿又要使用它们。直到确信不使用时再彻底删除。

例 1.3　计算两个整数的最大值。

```
#include <stdio.h>
int  max(int  a, int  b);        /* 对函数 max 的格式说明 */
void  main( )
{
   int  a = 10, b = 20;
   int  m;
   m = max(a, b);                /* 求 a 和 b 的最大值并存入 m */
   printf("%d", m);
}
int  max(int  a, int  b)
{
   int  t;
   t = (a>b ? a : b);
   return  t;
}
```

这是一个稍微复杂的例子，它由两个自定义函数 main 和 max 构成。其中，函数 max 可以求出任意两个整数的最大值，而 main 函数则利用 max 得到 10 和 20 的最大值。main 函数中有两个语句使用了其他函数：

```
m = max(a, b);
printf("%d", m);
```

一个函数使用另一个函数被称为“函数调用”。虽然这两次调用的函数分别是自定义函数 max 和库函数 printf，但处理方法是相同的。程序中的第二行是对函数 max 的格式声明，与第一行对 printf 进行格式声明代码的作用一致。

观察以上示例程序可以发现，除了函数的“头部”以外，组成函数的每一行代码之后都有一个分号，每个由分号结束的代码行就是一个“语句”，该分号是属于语句的必要成分，标志着语句结束。此外，在给出的示例中，我们尽可能使程序的格式美观、易读，这是初学者必须注意养成的良好习惯，尽管 C 语言本身无此要求。

工程

程序的良好格式，如一致的缩进编排风格，是使你的程序具有可读性、使你能与他人合作的基础，有时它比解决问题本身还重要。

1.4 高级语言程序的处理过程

如前所述，用高级语言编制的程序并不是计算机能够直接识别并执行的指令集合，而是一个文本文件，称为“源程序”文件。为了能够运行它，必须将其转换为机器指令，实现这一转换的程序就是“编译程序”或“解释程序”。

编译和解释是指程序的两种执行方式，或者说是对源文件的处理方式。从理论上说，每种语言都可以采取这两种形式来处理，而实际上，一般高级语言只使用了其中的一种方式。早期的 BASIC 语言采用了解释方式，而 C 语言使用了编译方式。为此，两种系统分别提供了相应的解释器和编译器。

一个解释器逐行处理源程序。这就是说，解释器每次将一行源代码翻译成机器指令并执行，直到源程序全部处理完毕。此方式使程序的执行速度较慢，且离不开源程序。

一个编译器一次性读入整个源文件，并将其全部转换成机器指令组成的“可执行文件”，从而彻底脱离源程序并有较快的程序执行速度。具体实现时，源程序到可执行程序的转换通常由两个程序完成，其一是编译程序，用于直接处理源程序，生成一个二进制代码文件，称为“目标文件”。另一个是链接程序（或称为连接程序），它处理由编译程序所生成的目标文件，进而得到一个可执行文件。

在网络应用非常普及的今天，程序的可移植性变得非常重要。因此，一些语言如 JAVA、C#等采用了稍微不同的方式，它们将源程序编译并链接为一种与具体机器无关的中间代码文件而非可执行文件。当中间代码执行时，还需要借助一种特殊的软件将其翻译成机器指令才能执行，这种软件称为“虚拟机”。

在目前的 C 语言编程工具中，既存在着逐个步骤单独处理的系统，也有很多“集成化”的开发环境。例如，以下是分步骤编写并运行一个程序的主要过程。

1．编写源程序

可以使用任何一种字处理器（文本编辑程序）如 Notepad、Notepad++等输入程序代码，并保存到磁盘上，这就是源程序文件，不妨假定文件名为 max.c。

提示

不同语言的源程序文件都以该语言的缩写为扩展名，C 语言的源程序文件的扩展名为 C。一个源程序文件的文件名要与实际功能相吻合，如 painter.c、graphics.c 等。

2．编译

系统可能单独提供编译程序和链接程序，如 tcc.exe（Turbo C）和 bcc.exe（C++ Builder），也可以提供一个程序，利用参数来表示编译和链接，如 gcc.exe（GNU 的 C 编译器）。以 tcc.exe 为例，简单的编译方式为：

```
prompt>tcc  max.c↙
```

如果程序正确则会生成一个新文件 max.obj，此即目标文件。当编译器通知出错时，应该修改源程序，然后再重新编译。这里用 prompt 表示系统提示符。

3．链接

如果链接程序为 tlink.exe，简单的链接方式为：

```
prompt>tlink  max.obj  <其他>↙
```

这里的“其他”是指一些必须链接的函数库。在没有发生错误时会生成一个新文件 max.exe，这就是操作系统管理下的可执行文件。如果有错，仍须修改源程序并重新进行编译和链接。

4．运行

执行 max.exe 命令：

```
prompt>max↙
```

在出现文字光标提示时输入必要的数据：

```
10,20↙
```

程序执行并显示运算结果为 20，结束。

完整的程序处理过程参见图 1.5。不过，简述这些步骤的目的只是为了理解高级语言程序的一般处理过程，而实际上却很少这样做，原因是目前 Windows 系统中的 C 语言环境都是集成化的，所有步骤既可以分步也可以一步实现。

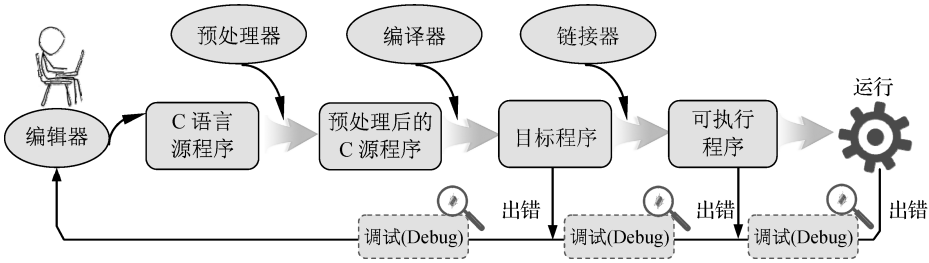

图 1.5　C 程序的编辑、编译与运行

1.5　利用 Visual C++ 6.0 环境编写和运行 C 语言程序

在 Windows 环境中，存在着多种编写 C 语言程序的集成化环境，称其为集成化环境是因为在该系统运行后，能够直接支持程序的编辑、编译、连接和运行，且提供了一定程度的程序调试功能。通常，这些 C 程序开发环境也都支持 C++程序的设计，甚至对 C++的支持更为全面。这里列出了一些主要的 C 语言开发环境。

（1）Turbo C 2.0 和 Borlan C++ 3.1

这是两款早期产品，是十分经典、高效的 C 语言开发环境，且集成了功能很强的程序调试功能。不过，它们工作在 DOS 环境下，界面美观程度和可调整性差，且 Turbo C 不支持鼠标操作，与当前流行的风格和用户的追求有一定程度的脱节，因而影响了它们的使用。

（2）DEV C++、Code::Blocks 与 C-Free

这是一些轻量级的 C/C++集成化开发环境，一般是开源或免费的。这些环境本身较小，通常不支

持可视化的窗口程序开发，集成的额外功能少，非常适合初学者使用。尤其是 DEV C++，甚至支持单文件程序而不必构成一个项目。相对于那些大型的开发环境来说，这些环境的程序调试功能稍微弱一些。

（3）Visual 6.0 和 C++Builder 6.0

二者分别是 Microsoft 公司和 Borland 公司的早期产品，也是两款比较大型和完善的 C/C++开发环境，支持建立各种应用，如控制台程序、窗口应用程序、DLL 等。每种系统都集成了丰富的程序调试功能。

（4）Visual Studio

微软公司的重量级产品，有 2005、2008、2010、2012 和 2015 等多种版本。这些系统非常庞大，同时支持多种语言，包括 C 和 C++，更注重对网络应用开发的支持。

此外，还可以采用 Eclipse 等环境进行 C 程序开发。对于初学者，我们建议使用 DEV C++、Visual 6.0 或 C++ Builder 6.0。这些产品很少需要用户做额外的配置，可以减少用户的负担，尽快将注意力集中到语言的语法和程序设计技术上。同时，它们也是一些考试、竞赛时的指定开发环境。当然，尽管这些编程环境都基本遵循 ANSI 或 ISO 标准，但也存在一些细微差异，包括调试工具、库函数和变量的命名等。

本书中的所有程序均以控制台方式工作，称为控制台程序或 DOS 程序，这是指程序的运行仅限于“命令提示符”窗口的文本方式下，不涉及与标准 Windows 窗口程序和图形相关的操作。

这里简要介绍利用 Visual C++ 6.0（以下简称 VC 6）开发 C 语言程序的过程，但在附录中也介绍了 DEV C++与 C++ Builder 6.0 的使用方法。

1.5.1 VC6 环境的安装与运行

VC6 是 Microsoft 公司 1998 年推出的基于 MFC 类库的 C++编程环境，集成了较完善的程序设计与调试功能，具有良好的操作界面，对网络、数据库等方面的编程都提供了较好的支持，有较大的用户群。

标准 VC 6 安装系统安装简单。在运行安装程序 Setup.exe 后，除了在定制安装（custom）时需要指定一个安装目录外，基本上每次单击“下一步”按钮即可完成安装。安装后，可将程序中的 VC 6 拖曳到桌面形成快捷方式，以便快速启动。

双击桌面上的 VC 6 图标，运行该集成化环境，系统进入如图 1.6 所示的 VC 6 主窗口。

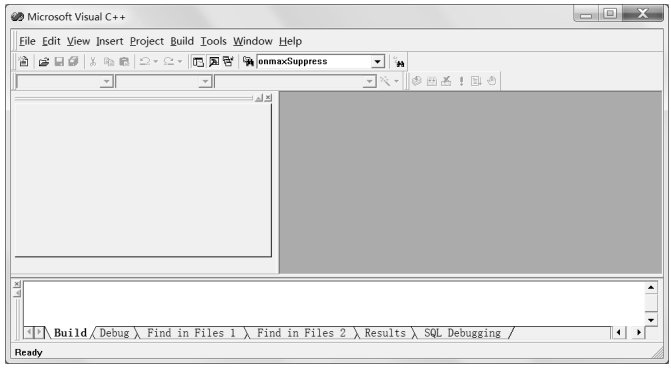

图 1.6 VC 6 的主界面

这是一个标准的 Windows 窗口菜单程序。窗口顶部是 VC 6 的主菜单栏，共包含 9 个菜单项，分别是：File（文件）、Edit（编辑）、View（查看）、Insert（插入）、Project（项目）、Build（构建）、Tools

（工具）、Window（窗口）和 Help（帮助）。每个菜单项都对应着一个子菜单，集成若干与菜单名相关的功能。在调试程序时，此菜单栏中还会增加一个 Debug（调试）菜单项。

主菜单栏的下面是快捷工具栏，排列着一些常用功能的快捷按钮。

窗口的左侧窗格是“项目工作间”（Workspace），用于显示项目中包含的文件、资源等信息。右侧窗格为视图区，显示被编辑的源程序文件。窗口下部窗格为信息提示区，用于显示编译、链接的结果，包括错误提示信息。

1.5.2　编制一个（控制台）源程序

1．创建项目

本质上，VC 6 是以多文档形式组织程序的，无论程序大小，都以完整项目的方式组织在一起。与一个项目有关的所有文档被存储在一个文件夹内。在处理一个项目时，VC 6 会打开一个工作间 Workspace。这样，每个工作间与项目形成对应关系，各工作间之间互不干扰。

为了简单，VC 6 也支持用户仅指定一个源程序，由系统生成缺省的项目。不过，这种以单独源文件组织程序的方式存在诸多限制，如系统不会为其创建文件夹、编译和运行要分步进行等。因此，我们建议以完整项目形式组织自己的程序。

选择菜单项 File→New，系统会弹出如图 1.7 所示的新项目创建窗口。在 Projects 页中单击选择一种项目后，系统会逐步引导和要求用户选择或输入一些必要的信息，进而生成与所选项目相关的缺省文档，这就是所谓的“向导（Wizard）”。此处选择 Win32 控制台类型的项目。

单击“Win32 Console Application”，在右侧的 Location（位置）域中输入或单击 ... 图标选择一个文件夹，如“D:\VC6\MyProjects”。在 Project name（项目名称）域中输入一个项目名，如“HelloWorld”，单击“OK”按钮。此时，系统会弹出一个如图 1.8 所示的窗口，其中只有几个简单选项。在窗口中可以选择生成空的项目（An empty project），单击“Finish（完成）”按钮，系统会弹出一个窗口，显示根据前面用户的选择所生成的项目的简短提示，询问是否接受这些设置。单击“OK”按钮表示接受，系统自动生成一个称为 HelloWorld 的简单项目，并在左侧窗格中增加了一个 ClassView 页和 FileView 页。

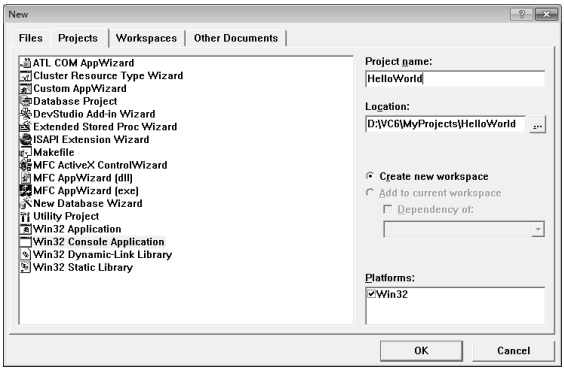

图 1.7　创建新项目对话框

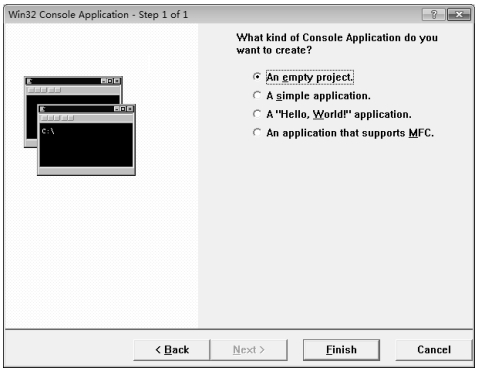

图 1.8　生成一个空的项目

重新选择菜单项 File→New，在图 1.7 的窗口中选择 File 页，单击 C++ Source File 选项，并在右侧的 File 域中输入源程序文件名如 main.c，单击“OK”按钮，系统生成一个空的源程序，返回工作间，并在窗口中间窗格打开编辑窗口，可以输入自己的程序，参见图 1.9。

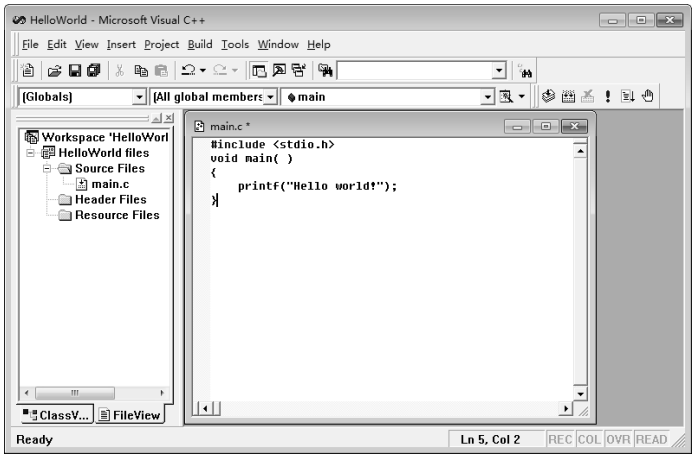

图 1.9 HelloWorld 项目的工作间

在上述步骤中，指定文件的扩展名为“.c”是必要的，它保证我们创建一个 C 而非 C++程序。

在幕后，VC 6 生成了组成一个项目的最基本文档，并保存在 D:\VC6\MyProjects\HellowWorld 文件夹下，包括若干与项目有关的文件，如工作间描述文件（.dsw）、项目文件（.dsp）、选择信息文件（.ncb）和源程序文件 main.c，还包括一个文件夹 Debug，用于保存编译和连接所生成的 obj 和 exe 文件。

2．工作间与重新加载项目

在任何时候，都可以利用 File 菜单下的 Open Workspace 选项和 HelloWorld.dsw 文件重新打开工作间，加载已建立的项目。

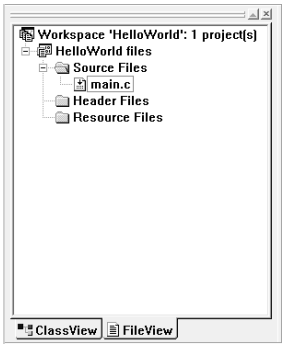

图 1.10 FileView 页

工作间左窗格含有两个页，均以树状的层次结构显示信息，可以单击“+”图标展开。其一是 ClassView，用于显示当前项目中所有类的信息以及外部函数和变量的信息，设计 C 语言程序时此处只有一个 main 函数（位于全局项 Globals 内）。另一个是 FileView 页，用于显示项目所包含的所有源文件信息。展开 HelloWorld files 树的分支后，包括三个文件夹，分别是 Source Files、Header Files 和 Resoure Files，各自包含了项目中的源文件、头文件和资源文件，参见图 1.10。这里的资源是指项目中使用的位图、加速键、字符串等信息，是 Windows 程序特有的组成部分，编写 C 控制台程序时不涉及这一部分。

窗口右侧显示的是由系统自动生成的源程序文件。在简单情况下，可以通过修改和添加程序代码完成设计。如果需要向项目中添加新文件，可以选择菜单命令 Project→Add to Project（增加到工程）→New，再选择所添加的文件类型，输入文件名，就生成了一个空文件，还可以使用菜单命令 Project→Add to Project→Files 向项目中添加已存在的文件。

1.5.3 编译、链接与运行程序

程序的编译、链接等命令集中在“Build（组建）”菜单中，如图 1.11 所示。

菜单中的第一项用来编译当前文件，第二项用于生成.exe 文件，包括编译和链接两个步骤。若程序中含有错误，系统会在下部的窗格中输出错误信息，可根据提示进行修改。在没有错误时，显示“HelloWorld.exe - 0 error(s), 0 warning(s)”形式的通知信息。通常，可以选择菜单命令“Execute HelloWorld.exe”直接运行程序，它包括了对程序的编译、链接和运行的全过程。

如果程序在输入或编辑后没有重新编译而直接选择运行，系统会弹出对话窗口询问是否构建（Build），单击“是”按钮即可。

一个程序运行后会出现控制台窗口，用户需要在此输入数据（如必要）。程序执行结束后，显示“Press any key to continue”的提示。按任意键后返回到 VC 6 主窗口。

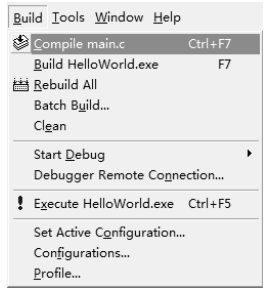

图 1.11　Build 菜单

1.5.4　程序调试技术

除了最简单的错误可以直接观察发现外，更复杂的错误需要借助调试工具来找出。程序调试、纠错（Debug）是学习编程时必须掌握的重要技术，而任何一个优秀的程序设计环境也都会提供本节所介绍的调试工具与手段。

初始时，VC 6 的调试（Debug）功能集成在“Start Debug（开始调试）”菜单的子菜单中，参见图 1.12。利用子菜单中的前 3 项中的功能都可启动调试器，使程序进入调试状态。此后，这些功能和一些新增的功能被自动集中到一个新菜单“Debug（调试）”中，代替原来的 Build 菜单，如图 1.13 所示，其中的主要项目含义如表 1.1 所示。

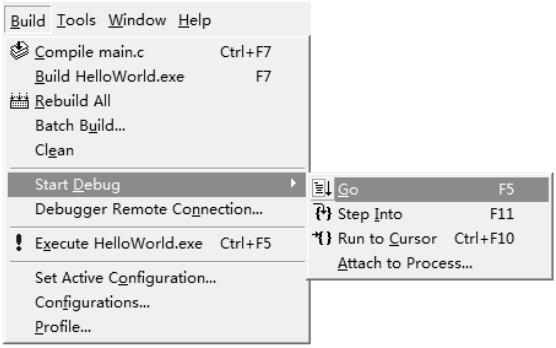

图 1.12　Build 菜单中的调试子菜单

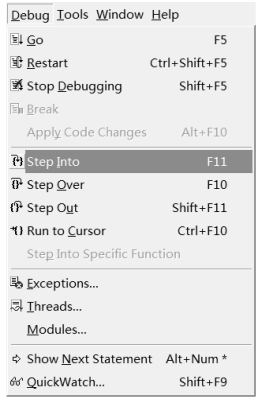

图 1.13　Debug 菜单

表 1.1　主要的调试功能

菜单项	快捷键	功能
Go	F5	调试/继续运行
Step Over	F10	单步，不跟踪到函数内部
Step Into	F11	单步，跟踪到函数内部
Run to Cursor	Ctrl+F10	运行到光标所在行
Insert|Remove Breakpoint		设置/清除断点
	Ctrl+Shift+F9	清除所有断点
Stop Debugging	Shift+F5	结束调试

这里首先说明如下几个概念。

（1）调试状态

也可以称为跟踪状态，是指程序仅执行了部分代码，驻留在内存并且仍可继续执行的运行状态。此时，可以查询甚至修改变量的值。

（2）一行或一个语句

也可称为一步，这是指能够引发运算的一个可执行语句，不包括那些说明性质的语句。例如，下述语句不是可执行的：

```
int  x;                                    //说明性质的语句
extern  float  f(float  x);                //说明性质的语句
```

当程序单步执行时，说明语句、括号、空行及注释等被跳过。

（3）变量及表达式的值

这是指程序执行到光标所在行之前的变量与表达式的当前值，与在此行直接使用输出语句输出的结果相同。

以下结合 Debug 菜单项说明其作用及调试方法。

1．部分执行程序并启动调试器

程序并非要一次性地执行完所有代码，调试者可以指定其执行到某处，即一个指定的行，也可以逐行执行程序代码。为了调试方便，应注意记住这些功能所对应的快捷键。

有两种常用方法使程序执行到指定位置并启动调试器。

（1）运行到光标处 Run to Cursor

此功能将使程序执行到光标所在行。首先，将文字光标移动到某行（该行应该是可执行语句）后选择此功能，则程序执行到该行暂停（但不包括当前行），进入“调试状态”，并在当前行的左侧显示一个黄色的箭头，表示暂停的位置。

（2）执行 Go

将文字光标移到某行，选择“Insert Breakpoint”功能，设置一个“断点”，系统在编辑窗口左侧显示一个红色圆点标记。用此方法可设置多个断点。执行 Go 命令，程序执行到第一个断点处暂停，进入“调试状态”，并在当前行的左侧显示黄色的箭头标志。

2．VC 6 的调试状态

在程序进入调试状态后，VC 6 窗口会产生相应的变化，参见图 1.14。图中新增加了一个由快捷键组成的 Debug 窗口，其功能与 Debug 菜单一致，包括：

① Restart：终止当前的调试过程，重新开始执行程序。

② Stop debugging：停止调试。

③ BreakExecution：终止程序运行，进入调试状态，一般用于终止一个无限循环。

④ Apply Code Changes：当源程序在调试过程中发生改变时，重新进行编译。

⑤ Show Next Statement：显示下一个将被执行的语句。

⑥ Step Into：单步执行，跟踪到函数体内部。

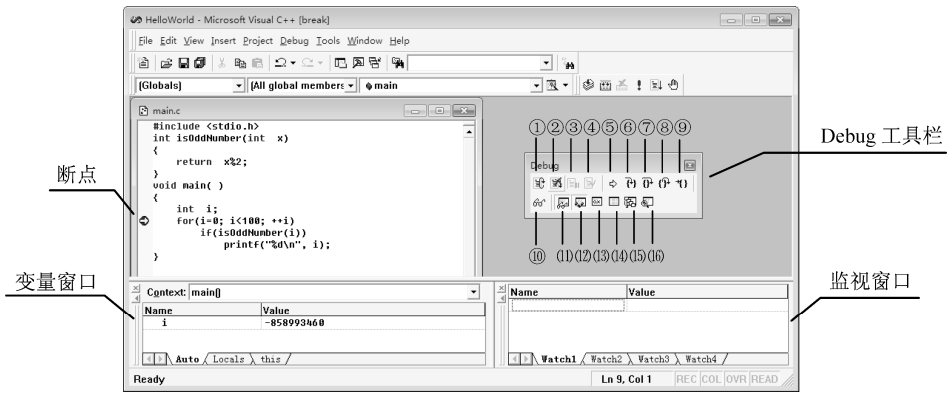

图 1.14　VC 6 的调试状态

⑦ Step Over：单步执行，不跟踪到函数体内部。

⑧ Step Out：从函数体内部跳出。

⑨ Run to Cursor：运行到光标所在位置。

⑩ QuickWatch：快速查看变量或表达式的值。

⑪ Watch：激活观察窗口，观察指定变量或表达式的值。

⑫ Variables：激活变量窗口，观察靠近断点处的变量值。

⑬ Register：激活寄存器窗口，观察当前运行点的寄存器内容。

⑭ Memory：激活内存窗口，观察指定内存地址的内容。

⑮ Call Stack：激活调用栈窗口，观察调用栈中还未返回的被调用函数列表。

⑯ Disassmbly：激活汇编代码窗口，显示被编译代码的汇编语言形式。

在默认情况下，启动调试器时会自动打开 Variables 窗口和 Watch 窗口。

3．一般调试步骤

通常，在程序运行不能得到正确结果时，需要仔细观察代码中的主要变量，调试的基本做法是检查它们的值，逐渐缩小范围直至发现错误。因此，可以按如下过程进行调试。

① 将光标移到源程序的某一需要暂停的行，按 F9 键（Insert Breakpoint）在当前光标处增加一个断点。

② 按 F5 键（Go）开始调试程序，使其运行到断点位置暂停，进入调试状态。也可以不设置断点直接在某行处按 Ctrl+F10 键（Run to Cursor），使程序运行到当前行暂停。

③ 查看。在变量窗口观察到当前环境下涉及的各个变量的值，或者在监视窗口中输入变量和表达式以查看其当前值。

④ 按 F10 键（Step Over）和 F11 键（Step Into）单步执行，或者，按 F5 键执行到下一个断点，或者将光标移到一个新行按 Ctrl+F10 键使程序执行到光标处。反复使用这些功能，观察变量的值，找出程序中的错误。

⑤ 按 Shift+F5 键（Stop Debugging）停止调试。

1.5.5　简单的程序调试与纠错

掌握和熟练运用程序调试技术是一个循序渐进的过程，这里仅通过一些简单示例说明调试与纠错步骤，其中可能涉及目前尚未学到的知识，读者可以先行浏览，在后续学习时重新回顾这一部分内容并尽可能熟练运用这些方法。

1．修正编译和链接时的错误

C 语言编译程序时指出的错误信息有两类，即 Warning（警告错误）和 Error（致命错误）。含有致命错误时不能通过编译，而警告错误一般是类型匹配问题或已定义的变量没有真正使用之类的错误，可以形成目标文件。作为示例，考察如下程序的编译和连接。

```
#include <stdio.h>
void  main( )
{  int  x = 10, y;
   x = x+1;
   x = x-y;
   printf("%d", x);
   print("%d", y);
}
```

（1）编译错误

编译上述程序时，系统提示程序中含有错误，同时，在屏幕下部的信息窗格内显示错误信息，参见图 1.15。

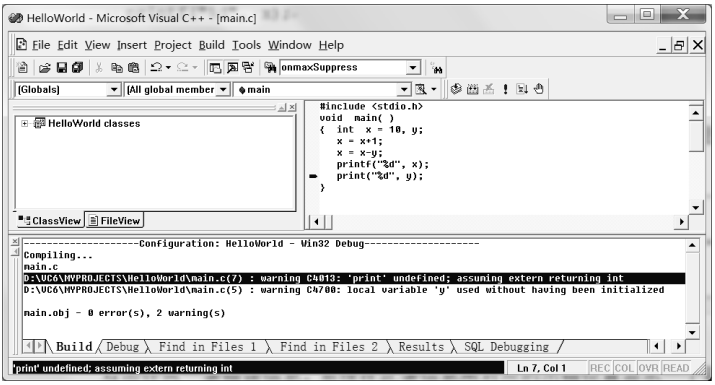

图 1.15　错误信息提示

双击任何一条错误信息，系统在此条信息所对应的出错位置显示一个箭头标志，并自动将文字光标移动到检测出错误的行，以方便修改。

此例中的第一个警告信息是指函数 print 的格式未知，假定其为 int 类型；第二个警告信息说明使用了未初始化的变量 y。

修正上述错误，形成如下代码并重新编译，生成 OBJ 文件。

```
#include <stdio.h>
void  main( )
{  int  x = 10, y = 5;
   x = x+1;
   x = x-y;
   printf("%d", x);
   print("%d", y);
}
```

（2）链接错误

构建 EXE 文件时，系统在信息窗格中显示如下错误信息：

main.obj : error LNK2001: unresolved external symbol _print

Debug/HelloWorld.exe : fatal error LNK1120: 1 unresolved externals

这是指标号 print 没有定义（实际是缺少字符 f）。修改此错误，重新进行编译和链接。

应该说，要尽量了解错误的类型和错误信息的含义，才能快速准确地排除编译、连接中出现的错误。

提示

为弄懂错误的性质并予以修正，任何一个编程者都需要积极了解错误提示的含义，并努力积累正确处理相应错误的经验。

初学者最常出现的错误之一是遗漏了作为语句结束符的分号。如果系统提示错误为“missing ';' before 'do'”，其真实情况是在错误标志的前一行的末尾缺少了分号。