《电脑报》计算机普及教程 ②

PC

QUICK BASIC 程序设计及应用

李刚荣 杨胜 编著

成都科技大学出版社

《电脑报》计算机普及教程

之 二

QUICK BASIC 程序设计及应用

李刚荣 杨胜 编著

成都科技大学出版社

(川)新登字 015 号

内 容 简 介

本书以 Quick BASIC 4.0 为背景，分十四章详细地介绍了 Quick BASIC 语言开发平台和程序设计。前者包括编辑器、调试器、库管理工具、命令行的编译连接；后者包括语言基础、结构设计、模块化程序设计、设备和文件 I/O、事件和错误捕获等内容。同时也介绍了程序开发步骤、模块化设计的分而治之策略、流程图分析等设计方法和经验，以及 Quick BASIC 与其它 BASIC、C 语言、汇编语言的联系，是一本难得的学习 Quick BASIC 的好书。

本书紧紧把握程序设计的目的是对数据空间进行操作，而程序只是一组操作符，处处把落脚点放在数据状态的转移和格式的变换两个方面；同时也统一了各种外设和文件 I/O，对于学习其它高级语言也很有帮助。

QUICK BASIC 程序设计及应用
李刚荣 杨胜 编著

责任编辑：哈 森 黎和生
技术编辑：周 勃
封面设计：易高原

* * * * * * * * * * * * * *

成都科技大学出版社出版
全国新华书店经销
《电脑报》社照排部 排版
达川新华印刷厂 印刷

开本：787×1092 毫米 1/16 印张：16.75 字数：435 千字
1994 年 12 月第 1 版 1994 年 12 月第 1 次印刷

ISBN7-5616-2978-8/TP·112

定 价：12.00 元

前言

BASIC 语言以它的通俗易懂、易学，适用范围广等特点受到广大微机用户的欢迎。从程序开发和高级语言模块化发展的需要，BASIC 向新的高度生长过程中孕育出 QuickBASIC 这样一种全新语言产品。

更确切地说，QuickBASIC 是一种语言开发的平台，它拥有一般的集成开发环境那种对用户界面友好的特点，同时在基于已有的 BASIC 语言向数据结构和模块化设计方面有所提高；为了满足各种用户需求，还提供了命令行方式来开发程序。

本书围绕着 QuickBASIC 的工作环境和语言环境两个方面，分别介绍了 SMART 编辑器、调试器、库管理工具以及语言的结构，模块设计等内容，尤其是把设备和文件 I/O 统一起来，从共性之中找区别。

本书具有以下一些特点：

1. 把结构化、模块化程序设计的思想融汇在全书中，使读者养成良好的习惯。在结构化设计中，以流程图这种直观形式来表述，能使读者养成有步骤地解决问题；在模块化程序设计中，从分而治之的策略角度来介绍模块，对于解块实际问题中分解难度大有裨益。

2. 重视基本概念和能力的培养。本书以“程序可看成在数据空间上操作的一组记号”为基本指导思想，仔细介绍了数据的表达、存贮；同时，在介绍语句或函数的过程中，尽量结合操作对数据的改变来阐述语句和函数的功能，以期能达到举一反三的效果，激发读者的学习兴趣。

3. 全书循序渐近，由浅入深，逐步深化。从语言角度来讲，先引入数据的表达和分类，再遵从数据的获取、数据的处理、数据的输出三步曲讲述 QuickBASIC 的基本语言要素，从结构化、模块化角度介绍语言特色，从功能的多样化来分析接口设备，最后统一于几个有特色的语句。从工作环境来说，由初识到逐步涉及调试到命令行，也是遵从了认识规律的。

4. 本书还从 QuickBASIC 程序的建立、编辑、调试、编译、连接和执行，以及如何调用 C 语言和汇编语言子程序等方面作了较为详细的介绍，这对于帮助读者解决在使用 QuickBASIC 中的一些问题是很有好处的。

5. 书中的例子有很强的示范作用，至少对学习 QuickBASIC 语言和编辑风格是一潜在的帮助。

本书由邓仁明、任娜审校。因水平和时间所限，本书难免存在缺点和错误，欢迎读者批评指正。

编者

1994 年 7 月

目　　录

第一章　Quick BASIC 概述 …… (1)
1.1　Quick BASIC 的特点 …… (1)
1.2　Quick BASIC 软、硬件环境 …… (1)
1.3　Quick BASIC 的建立和安装 …… (2)
1.4　Quick BASIC 的工作屏幕 …… (3)
1.5　设置 DOS 环境变量 …… (4)
1.6　Quick BASIC 开发程序的过程 …… (5)
本章小结 …… (7)

第二章　Quick BASIC 语言基础 …… (8)
2.1　数据和数据类型 …… (8)
2.2　提供数据的语句 …… (13)
2.3　数据的操作 …… (19)
2.4　数据的显示和打印语句 …… (34)
2.5　Quick BASIC 语言元素 …… (48)
本章小结 …… (51)

第三章　Quick BASIC 基本程序结构 …… (52)
3.1　程序流程图 …… (52)
3.2　三类基本的程序结构 …… (53)
3.3　选择程序设计 …… (55)
3.4　循环程序设计 …… (70)
本章小结 …… (81)

第四章　Quick BASIC 模块化程序设计 …… (82)
4.1　模块化程序设计 …… (82)
4.2　模块中的变量 …… (83)
4.3　过程：子程序和函数 …… (93)
4.4　递归过程 …… (102)
4.5　字符串处理专题举例 …… (103)
本章小结 …… (113)

第五章　文件和设备 I/O …… (115)
5.1　文件 …… (115)
5.2　光笔、鼠标、扩音器、打印机、键盘 …… (131)
5.3　显示器 …… (133)
5.4　串行通信口 …… (151)
5.5　I/O 语句 …… (152)
本章小结 …… (152)

第六章　事件捕获和错误捕获 …… (154)
6.1　事件捕获 …… (154)
6.2　错误捕捉 …… (158)
本章小结 …… (159)

第七章　Quick BASIC 程序的建立和执行 …… (160)
7.1　编写一个程序 …… (160)
7.2　在 Quick BASIC 中建立可执行文件(构造 EXE 文件) …… (163)
7.3　在程序中使用 SUB 和 FUNCTION 过程 …… (166)
7.4　将模块合并到程序中 …… (167)
本章小结 …… (168)

第八章　Quick BASIC 编辑器 …… (169)
8.1　文本输入 …… (169)
8.2　Smart 编辑器的特征 …… (169)
8.3　何时启动 Smart 编辑器 …… (169)
8.4　自动语法检查 …… (169)
8.5　关闭语法检查 …… (170)
8.6　插入与覆盖输入 …… (170)
8.7　选择文本 …… (170)
8.8　删除和插入文本 …… (170)
8.9　恢复当前行 …… (171)
8.10　文本的移动与复制 …… (171)
8.11　查找和替换 …… (172)
8.12　在程序中使用位置标记符 …… (174)
8.13　输入特殊字符 …… (174)
8.14　缩进 …… (174)
8.15　行的连接 …… (175)
8.16　从其它文件中拷贝文件 …… (175)
8.17　编辑命令总结 …… (176)

第九章　程序调试 …… (177)
9.1　Quick BASIC 的调试手段 …… (177)
9.2　排错 …… (177)
9.3　Quick BASIC 的调试功能 …… (178)
9.4　高级调试 …… (183)
9.4.1　Calls 菜单 …… (183)
9.4.2　与代码图调试程序的兼容性 …… (184)
本章小结 …… (184)

第十章　Quick 程序库 …… (185)
10.1　程序库的类型 …… (185)
10.2　Quick 程序库的优点 …… (185)
10.3　建立一个 Quick 程序库 …… (186)

10.4 使用 Quick 程序库 …… (188)
10.4.1 装入一个 Quick 程序库 …… (188)
10.4.2 Quick 程序库中的浮点运算 …… (188)
10.4.3 查看一个 Quick 程序库的内容 …… (188)
10.5 系统提供的程序库(QB、QLB) …… (189)
10.6 .QLB 文件扩展名 …… (189)
10.7 在命令行上建立一个程序库 …… (189)
10.8 在 Quick 程序库中使用其它语言编写的例程 …… (189)
10.9 使用 Quick 程序库的内存考虑 …… (190)
10.10 创建压缩的可执行文件 …… (190)
本章小结 …… (190)

第十一章 命令行编译、连接和库管理 …… (191)
11.1 命令行方式简介 …… (191)
11.2 命令行编译 …… (191)
11.3 命令行连接 …… (193)
11.4 管理独立库文件 …… (195)
11.5 命令行方式举例 …… (197)
本章小结 …… (198)

第十二章 从 BASIC 程序到 Quick BASIC 的转换 …… (200)
12.1 源文件格式 …… (200)
12.2 在 Quick BASIC 中禁止使用的语句和函数 …… (200)
12.3 需要修改的语句 …… (200)
本章小结 …… (201)

第十三章 Quick BASIC 4.0与低版本的区别 …… (202)
13.1 新增功能 …… (202)
13.2 与低版本在程序设计环境上的不同 …… (207)
13.3 编译和调试中的区别 …… (209)
13.4 关于 BASIC 语言方面的改变 …… (211)
13.5 文件的兼容性 …… (214)
本章小结 …… (214)

第十四章 调用 C 和汇编语言例行程序 …… (215)
14.1 混合语言程序的编制 …… (215)
14.2 混合语言程序设计要素 …… (215)
14.3 BASIC 语言对 C 语言的调用 …… (219)
14.4 C 语言对 BASIC 的调用 …… (225)
14.5 参考传递或真值传递数据 …… (228)
14.6 混合语言程序设计中的数字和字符串数据 …… (229)
14.7 特殊的数据类型 …… (232)
14.8 指针、地址变量和公用块 …… (234)
14.9 B_OnExit 例程 …… (236)

14.10 汇编语言与 BASIC 语言的接口 …… (237)
本章小结 …… (244)

附录一 Quick BASIC 4.0语言函数、语句功能一览表 …… (245)

附录二 Quick BASIC 开发环境命令一览表 …… (253)

符号约定

下面的句法(以 LOCK 语句为例)说明了本书中使用的符号约定:

LOCK[#]filenumber[,{record|[start]TO end}]

其中,[ ]表示里面内容可以省略;当然,必须是"["与"]"配对里面,因为它可以嵌套

{}表示里面内容在"|"两边必取其一

|用来"{}"里起分隔并列参数的作用

因此,上述语句可展开为多种形式:

```
LOCK [#]filenumber
LOCK [#]filenumber,record
LOCK [#]filenumber,TO end
LOCK [#]filenumber,start TO end
LOCK filenumber
LOCK filenumber,record
LOCK filenumber,TO end
LOCK filenumber,start TO end
```

第一章　Quick BASIC 概述

Quick BASIC 简称 QB，是美国 McroSoft 公司推出的一个语言开发环境，它支持鼠标器操作，既可以在集成环境下完成 BASIC 语言程序的开发，也可以采用命令行方式。Quick BASIC 将一整套编程环境和最有效的 BASIC 语言结合起来，为用户提供了编制迅速、紧凑的程序所需的一切手段。

1.1　Quick BASIC 的特点

QB 事实上是指一套软件系统，支持具有一些新功能的 BASIC 语言，提供了命令行和 IDE (Integrated development environment)开发环境，用户界面比较友好。所以 QB 的特点也有两个方面，即编程环境和语言环境。

1. 在 QB 环境中，可任意将编辑、运行、调试混合使用，因为所有编程工具时刻都处于开启和运行状态。下面就是为什么用 QB 能节省时间并能写出较好程序的原因：

①程序在写出的同时可以立即得到执行

②立即求助和句法检查

③立即调试

④多模块

⑤多编辑窗口

⑥完全的图形支持

2. QB 还扩充了 BASIC 语言，使其包括如下很强的功能：

①QB 支持二进制文件，所以程序可以以任何格式来生成和处理文件。

②用户定义的数据类型简化了随机存取文件的输入输出(I/O)，并有助于用数值变量和数值串建立更为复杂的数据结构。

③QB 的子程序和函数可以是递归的(它们可以自我调用)，允许写出更紧凑更有效的代码。

④长(32 位)整数和 8087/80287 协处理(功能)有助于作更快、更精确的数学运算。

⑤高级的控制流程语句，如 SELECT CASE 和 IF…THEN…ELSE 以及 DO…LOOP 等，将改进程序的流程。

⑥数组下标甚至可以为负整数

1.2　Quick BASIC 的软硬件环境

MicroSoft Quick BASIC 有下列硬件需求：

1. 一台 IBM－PC 或能运行 MS－DOS 2.1 以上版本的兼容机

2. 至少一个软盘驱动器

3. 320K 以上内存

同时，MS－QB 也至少需表 1.1 所列软件(DOS 除外)，根据 QB 版本不同，或许稍有不同。

表 1.1　使用 QB 时所需的文件

文件名称	简　介	功能
QB. EXE	集成开发环境	在内存中建立、调试并运行 BASIC 程序；建立独立可执行行程序；建立 Quick 程序库。

BC. EXE	命令行编译程序	在 QB 中或在 DOS 下建立可执行程序需要的目标程序；建立 Quick 程序库。
QB. QLB	Quick 程序库，含支持 DOS 中断的例程	
QB. LIB	独立程序库，含有支持 DOS 中断的例程	
QB. BI	与 QB. QLB 和 QB. LIB 一起使用的包含文件	
BRUN40. EXE	QB 运行时模块	运行由 BRUN40. LIB 生成的 BASIC 可执行程序
BQLB40. LIB	Quick 程序库，运行时支持程序	建立 Quick 程序库。
BRUN40. LIB	QB 运行时模块库	在 QB 中或 DOS 下建立可执行程序。
BCOM40. LIB	QB 交互式运行库	在 QB 中或 DOS 下建立可执行程序；用该库生成的可执行程序在运行时不再需要 BRUN40. EXE。
LINK. EXE	MS 覆盖连接程序	在 QB 中或 DOS 下建立可执行程序；建立 Quick 程序库。
LIB. EXE	MS 程序库处理程序	建立独立程序库
NOCOM. OBJ	用来与不需要通讯支持的程序相连接的目标文件	减小不需要通讯支持的可执行程序的尺寸
MOUSE. COM	与 QB 一起使用的鼠标器驱动程序	使用鼠标器
QB. HLP	QB 的在线求助系统内容	使用在线求助系统

值得一提的是，系统配置文件 CONFIG. SYS 一般要作一些变动，以为 QB 建立一个合适的工作环境，下面举一例说明。

配置文件内容	说　明
DEVICE=MOUSE. COM	支持鼠标器
PATH=C:\BIN;C:\QB	缺省路径
SET LIB=C:\LIB	程序库路径
SET NO87=␣↙	有协处理器(␣表示空格)

1.3 Quick BASIC 的建立和安装

QB 的安装是相当简单的，因为源盘上的文件既没有压缩，也没有加密，所以安装只是简单地把相应文件拷贝到硬盘等目标驱动器。同时，QB 源盘也配备了 SETUP. BAT 安装批处理文件，只需运行即可。

若要在软盘上工作，则建议有三张工作盘，每张盘的内容如下：

第一张	第二张	第三张
QB. EXE	BC. EXE	BCOM40. LIB
用户源程序	LINK. EXE	
QB. QLB(选择)	BRUN40. EXE	
	BRUN40. LIB	
	LIB. EXE	

1.4 Quick BASIC 的工作屏幕

QB 的工作环境表现为一个多窗口的显示屏幕和下拉式菜单提供的命令系统。在这个环境中可以编辑源程序；可以把源程序编译成目标程序；可以建立运行库；可以运行目标程序；还可以对正在运行的程序进行跟踪。具体地讲，用 File 命令操作文件，用 Edit 命令进入观察窗口输入源程序，用 Run 命令来完成运行、调试、建库等工作。程序调试好后，既可以将源程序(.BAS)存盘，也可以将目标程序(.EXE)存入磁盘。

QB 的工作屏幕由四个部分组成：顶部为八项主菜单区，中部为编辑区，下部为信息提示区，底部为系统提示及热键区(见图 1.1)。

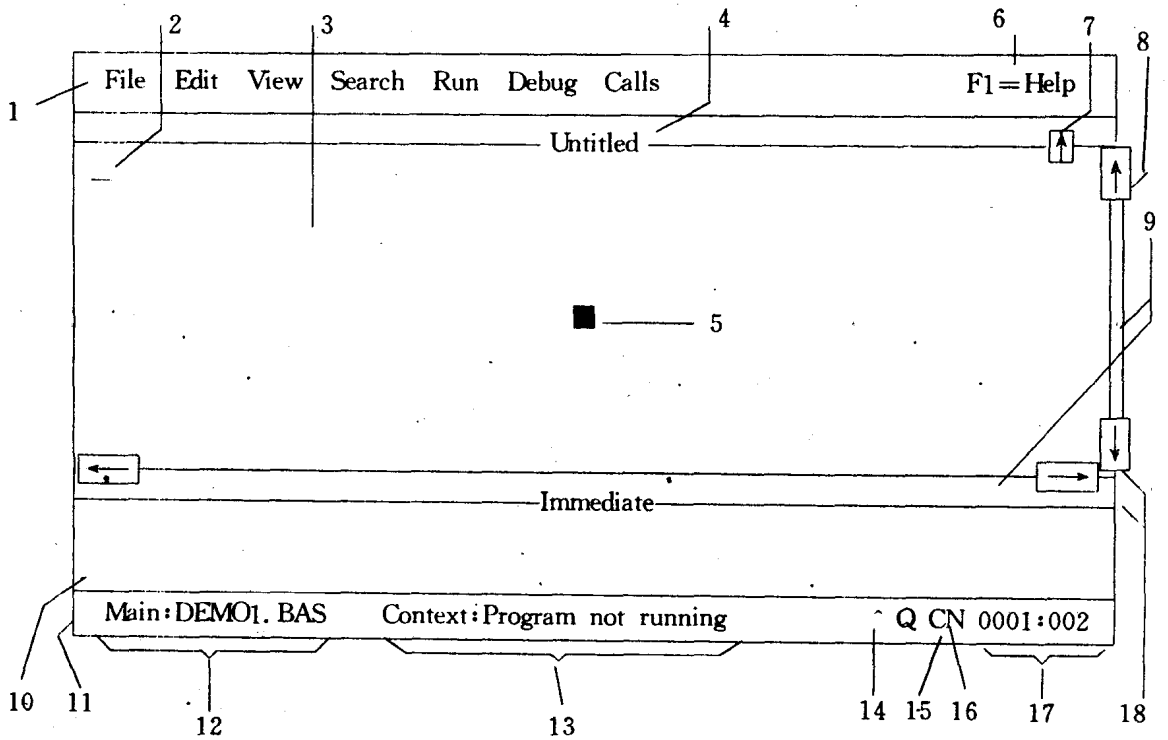

图1.1 QB工作屏幕示意图

下面将按照图 1.1 的编号逐项解释 QB 屏幕的各个部分：

1. 菜单栏。其中含有各种菜单的名称，高亮显示的字母表示缩写键。
2. 光标。是一条闪烁下划线，表示用户键入的文本将要出现的位置。
3. 观察窗。位于文本中央一个窗口(可以分割成两个窗口)。
4. 标题栏。用以显示出现在它下面的窗口中的文件名。
5. 鼠标指针。显示鼠标器在屏幕上的当前位置(仅对鼠标器而言)。
6. F1=Help。表示按下此键后，进入联机帮助状态。若要清除帮助屏，可按 ESC 键。
7. 扩展框。用以扩展当前窗口来填充屏幕(仅在使用鼠标器时使用)。
8. 滚动框。用以显示光标在文件或过程中的相对位置。
9. 滚动带。在当前活动窗口中滚动文本(仅对鼠标器而言)。
10. 立即窗口。在此窗口内能立即执行 BASIC 语句。
11. 状态框。包含文本位置、键盘状态和程序信息。
12. 主模块(Main)。指明程序主模块名。

13. 上下文(Context)。下一条将要执行语句的模块名(或过程名)。

14. ^Q。当键入 Wordstar 字体命令序列时,显示^Q。当输入一个位置标志时,此处将出现^K;当输入一个文字控制符时,此处将出现^P。

15. C。当按下 CapsLock 键时,显示 C。

16. N。当按下 NmLock 键时,显示 N。

17. 行和列计数器。指出文本在活动窗口里的当前光标位置。

18. 滚动箭头。每次滚动文本的一个字符或一行。

1.5 设置 DOS 环境变量

如果用户不愿把所有 Quick BASIC 文件保存在同一个目录中,可把它们放在不同的目录下,然后用 DOS 里的 PATH 和 SET 命令来定义环境变量,用以告诉 Quick BASIC 到何处寻找所要找的文件。

设置环境变量对运行 Quick BASIC 是有帮助的。若不设置环境变量,则 Quick BASIC 仅在当前目录下查找所需文件,如果找不到相应的文件,将显示出信息"Cannot find file,Input path:"

一旦设置了环境变量,则它就一直保持刚设置的功能直到重新给它设置一个不同的值或重新启动或关机为止。

1.5.1 和 1.5.2 两节简要介绍了两个最常用的命令——PATH 和 LIB,并介绍了如何为它们赋值。如果有 8087 或 80287 协处理器芯片,则为使程序忽略该片,可阅读 1.5.3 节关于 NO87 环境变量的设置。

1.5.1 PATH 环境变量

PATH 环境变量告诉编译程序到何处去寻找可执行文件。QB.EXE 和 BC.EXE 都将寻找 PATH 环境变量。

在 DOS 下用 PATH 命令来确定 PATH 变量。键入该命令,一个空格或一个等号,以及一个或多个路径说明符,其间用分号";"隔开。例如:

```
PATH=C:\BIN; C:\QB
```

该设置告诉编译程序先到 C 盘上的\BIN 目录中查找,然后再到 C 盘上的\QB 目录中查找那些可执行文件。

用该例设置的 PATH,可把 QB.EXE 放在\BIN 目录中,然后就可在任何目录下启动 Quick BASIC。

注意,如果用户有 DOS3.0 或更高版本,就可与 SET 命令一起来设定 PATH 变量,如下所示:

```
SET PATH=C:\BIN;C:QB
```

1.5.2 LIB 环境变量

LIB 环境变量告诉 QB 将 QB 命令行中说明的 Quick 程序库定位于何处,并告诉 LINK.EXE 到何处去寻找所需要的程序库。可用 SET 命令定义 LIB 环境变量。例如,下列命令行告诉连接程序到 C 盘的\LIB 目录中寻找所需要的程序库:

```
SET LIB=C:\LIB
```

1.5.3 NO87 环境变量

如果有 8087 或 80287 协处理器,那么用户程序在运行时将使用 8087 或 80287。但是,用户也可以通过设置 NO87 环境变量使协处理器失效,并强迫程序仿真协处理器的功能。下面是设置 NO87 环境变量的两种方法:

```
SET NO87=Use of coprocessor suppressed
```

SET NO87=(一至多个空格)

当用程序代替现有的 8087/80287 执行时，第一种设置就使得屏幕上显示出“Use of co-processor suppressed”(取消协处理器的使用)信息。第二种设置不显示任何信息。

若要关闭对协处理器的强行仿真，可设置 NO87 等于零或空值，例如：

SET NO87=

注意，该例中等号(=)后面没有空格。

1.6 Quick BASIC 开发程序的过程

本节的开发过程是指读者已经在纸上写出一个新的源程序，并且顺利开发的过程，其大致有以下步骤：

1. 进入 Quick BASIC 集成开发环境。

在 DOS 下键入：QB DEMO1. BAS↙(假如源程序名为 DEMO1. BAS)。则进入下列信息和屏幕布局：

```
File  Edit  View  Seareh  Run  Debug  Calls                    F1--help
──────────────────── DEMO1. BAS ────────────────────

──────────────────── Immediate ────────────────────
Main:DEMO1. BAS   Context   Program   not   running          0001:001
```

2. 在编辑窗键入源程序，并保存源文件到盘上。
3. 编译、运行。
4. 退出集成环境，回到 DOS 状态。

一次顺利的开发 Quick BASIC 程序大致可用表 1.2 表示出来。

表 1.2 Quick BASIC 程序开发过程

步　骤	Quick BASIC 使用方法
进入 Quick BASIC 集成开发环境	在 DOS 下键入：QB 文件名↙
输入源文件	从键盘上输入源文件字符
保存源文件	先键入 ALT+F，再选择 SAVE 即可
编译运行	先键入 ALT+R 进入运行环境： →选 START 或 SHIFT+F5 运行 BASIC 程序 →选 F5 重复执行 BASIC 程序 →选 MAKE. EXE 建立执行文件
退出 QB 环境	按 ALT+F+X 三键，即退出 QB，返回 DOS。

前面曾有一个限制，那就是“顺利”的开发过程。事实上，从纸上的源程序到保存在盘上的可执行目标代码文件的开发工作，是很难一帆风顺，一次性成功的。下面就具体来分析一下。

首先，不成功不顺利体现在什么地方呢？大致有几个方面：①程序执行后，没达到预期的效果。比如希望通过程序使一台正在运转的电机停下来，可执行之后却没有停；更简单的如希

望在屏幕上画一个圆，可屏幕上却只出现一个矩形或者什么也没有。②源程序有语法错误，也就是源程序中的语句不能翻译(编译)成有效的机器语言表示的代码。在 Quick BASIC 开发环境中，一般在输入每行之后，编辑器就立即进行了语法检查。若有错，则立即给以亮度提示；若没有错，则翻译成执行代码，并把一些 Quick BASIC 语言保留的关键字变成大写以及进行一定的文本格式调整。③多模块之间连接不成功。④没有语法错误，但调试运行中总是有溢出现象或者死机。

其次，有错又怎么办？可不可以直接去修改机器代码或相应的汇编语言程序，因为它们是最直接地控制着计算机运行的代码。事实上，这几乎不可能，因为：①机器代码难以记忆；②完成一个极其简单的任务，都需很长的机器或汇编语言代码，阅读或调试它，几乎有在森林中迷失方向的感觉，不知道何去何从。常用的方法是修改源程序代码，然后再编译、调试，直到满意为止。

归纳起来，一般的编译型语言的程序开发过程如图 1.2 所示，并且现在的发展趋势是把开发过程中所需的工具，如编辑器、调试器、库管理、连接器、帮助等组装成一个下拉式菜单的软件，即集成开发环境，形成对用户更加友好的界面。Quick BASIC 就是如此。

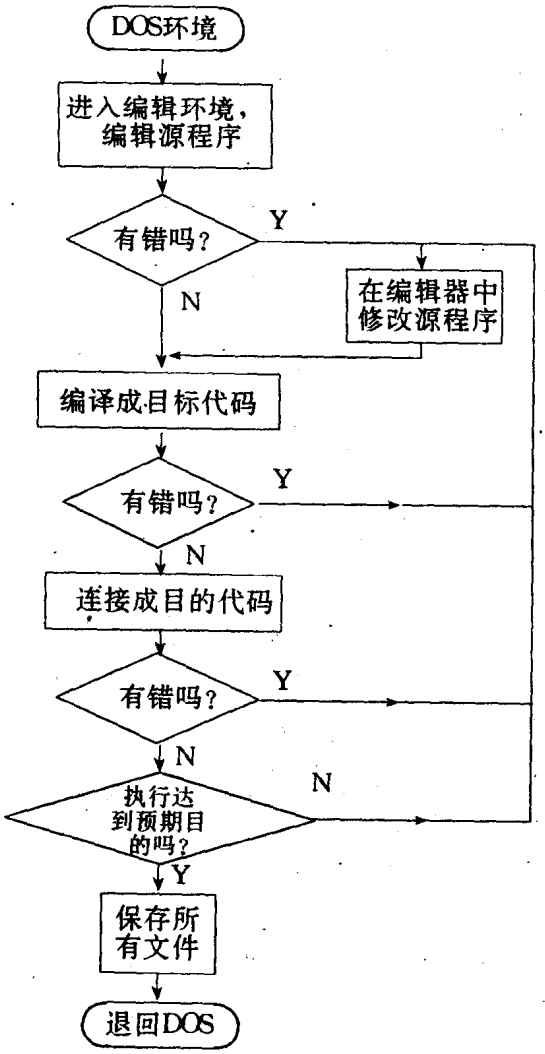

图1. 2 编译型语言的程序开发过程

本章小结

本章主要介绍了 Quick BASIC 的特点、软硬件环境、建立和安装、工作环境等内容，使读者对 Quick BASIC 有一个基本的认识和了解。

Quick BASIC 的工作环境表现为一个多窗口的显示屏幕和由菜单提供的命令系统。在这个环境中可以编辑源程序，可以把源程序编译成目标程序，可以调试运行目标程序等。

本章学习的操作较多，首先要熟练掌握使用 Quick BASIC 的基本过程：

1. 启动 DOS
2. 调用 Quick BASIC
3. 编辑源程序
4. 编译调试源程序
5. 运行程序
6. 退回 DOS

其它操作也是每次上机时经常使用的，尽快记熟它们可以大大提高调试程序的效率。

第二章　Quick BASIC 语言基础

从本章开始，将用四章来系统地学习 Quick BASIC 语言，即各种语句、函数以及编程方法。

2.1　数据和数据类型

计算机能直接执行的机器语言指令一般是由两部分组成，即操作符和操作数。前者告知计算机做什么事，执行什么样的操作，如加法操作；后面是操作中所涉及到的数据。显然，操作数是以二进制存贮的信息。

然而，高级语言如 BASIC 是面向问题的，是应当尽量接近人们的表达习惯。因此学习高级语言必须掌握该语言中数据信息的表示以及存贮方法等。

Quick BASIC 的数据类型见表 2.1 所示，有关各种类型的说明见表 2.2

表2.1 数据类型分类表

- 基本数据类型
  - 数值型数据
    - 整型
    - 长整型
    - 单精度浮点数
    - 双精度浮点数
  - 字符串型数据
    - 可变长度字符串
    - 定长字符串

表 2.2 数据类型说明

类型		字长	范围	举例
数值型	整型	2 字节	−32768～+32767　(−2^{15}～2^{15}−1)	68 +407 −1
	长整型	4 字节	−2147483648～+2147483647　(−2^{31}～2^{31}−1)	95000000
	单精度浮点数	4 字节	精确到 7 个十进制位 正值：1.40129E−45～3.402823E+38 负值：−3.402823E+38～−1.40129E−45	9.0846
	双精度浮点数	8 字节	精确到 15 或 16 十进制位 负数：−1.797693134862316D+308～−4.94065D−324 正数：4.94065D−324～1.797693134862316D+308	4.35D−10
字符串型	变长字符串	<=32767 字节，可变	对于 ASCII 码字符是 0～127； 对于非 ASCII 码字符是 128～255；	"ABCD"
	定长字符串	<=32767 字节，固定	同上	

2.1.1 常量

常量是指预先确定的值，在程序执行过程中它不改变。常量一般有两类：文字常量和符号常量。

1. 文字常量

Quick BASIC 又有两种文字常量：字符串和数值。

例如：LET X=3

PRINT 5,"ABCD"

这里的 3,5 是数值常量，"ABCD"是字符串常量。下面讲一讲常量的表示法。

(1)数值常量的表示法

①十进制整数(长整数)表示法如一般的数学记数法，如 3578

②十六进制整数表示法在数值前面加词头 &H，如 &H32F，长整数还要在数值后加词尾 &，如 &H1AAAAA&

③八进制整数一般在数值前加词头 &o 或 &，如 &o347，&1234，长整数也在数值后加词尾 &，如 &1234567&

④单精度浮点常值一般用 E 表示的指数形式，或者是尾随感叹号!，或者包含小数点且小数少于 15 位，或者没有小数点而又不能表示为长整数，如 9.8406，2235，98E-7，22!

⑤双精度浮点常值一般用 D 表示的指数形式，或者有尾随符 #，有小数点多于 15 位等形式表示，如 3456928.5，-1.09432 D-6，3489.0#。

(2)字符串常量的表示法：用双引号包含的 ASCII 码字符组合来表示，如"ABCD"，"$25,000,000"

2. 符号常量

Quick BASIC 提供的符号常量能用来代替数值或字符串，相当于为文字常量取了一个别名。下面的程序段说明了两个符号常量：

CONST MAXCHR=255,STRFORCON="DO IT AGAIN"

其中，CONST 英文含义为"常量，恒定不变的"。：

常量的类型由类型说明字符决定或由表达式的类型决定。

符号常量的使用有几个优点：

①常量在整个程序里定义一次就可以了

②常量不会因为疏忽而被改变

③在独立程序里，使用常量比使用变量能产生更有效的代码

④常量使程序容易阅读、维护、修改

2.1.2 变量

变量是命名的内存单元，它的值在程序运行时可以改变，见图 2.1。内存中的每个存储单元都有一个固定的地址编码，就如我们的每个房间都有房间号一样，数据就存放在某些单元中。每个单元的地址是不变的，而内存单元中放的数据则是可变的，因此称不变的地址单元为变量名，可变的单元内的值为变量或变量的值。

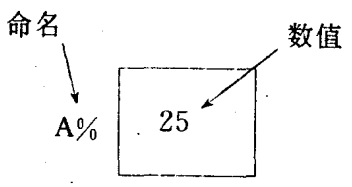

图 2.1 变量示意图

程序员编程首先得为数据选取合适的变量单元，并且取一个合适的变量名，这就需要遵从一定的规则：

①变量名需以字母打头，可含字符集有 A～Z，a～z，0～9，#，!，%，$ 等，一般字符长度不超过 40 个，如 NUMBER3%，STRING7$

②Quick BASIC 中的关键字不能作变量名，即命令、函数、语句等。