

神奇电脑通丛书

秦戈

Visual BASIC 6.0

电脑编程 轻松走



电子科技大学出版社

UESTC PUBLISHING HOUSE

JS 11 60

内 容 提 要

Visual Basic 6.0 是微软公司最新推出的综合可视化开发工具“可视化工作室 (Visual Studio)”中的一个组件,它使得开发各种 Windows 平台下的应用程序变得简便易行。本书从 Visual Basic 6.0 的安装和用 Visual Basic 编程的基础知识讲起,详细介绍了 Visual Basic 6.0 所提供的各种开发工具,并针对初学者的特点,对 Windows 编程的基本方法和一些重要概念进行了详细阐述。本书概念清晰,语言通俗易懂,尤其适合于没有 Windows 编程经验的人学习掌握。在介绍程序设计的同时,本书着重突出了设计方法的合理性和在程序设计中的重要性,并对一些常用编程技巧进行了介绍。

使读者能够快速掌握 Windows 编程的基本方法并能设计出具有一定实用价值的 Windows 应用程序是编写本书的宗旨。本书可作为中专和各类计算机培训班的教材,也可供计算机爱好者自学使用。

声 明

本书无四川省版权防盗标识,不得销售;版权所有,违者必究,举报有奖。举报电话: (028)6636481 6241146 3201496

Visual BASIC 6.0 —— 电脑编程轻松走

秦 戈

出 版: 电子科技大学出版社 (成都建设北路二段四号, 邮编 610054)

责任编辑: 文 利

发 行: 新华书店经销

印 刷: 四川建筑印刷厂

开 本: 787×1092 1/16 印张 13.875 字数 329 千字

版 次: 1999 年 9 月第一版

印 次: 1999 年 9 月第一次

书 号: ISBN 7—81065—238—9/TP·133

印 数: 1—4000 册

定 价: 16.80 元

前言

计算机科学的崛起，使世界发生了深刻的变化。计算机技术与计算机应用已经深入到社会生活中的每一个角落，了解并掌握计算机的有关知识已成为当今社会每个成员所必备的一项基本技能。本书正是为了适应

这个趋势，为加快计算机知识普及的步伐而编写的。

随着计算机技术的发展，各种各样的应用软件层出不穷，功能也越来越强大。人们几乎可以不需懂得有关计算机的任何知识就可以使用这些软件。这样虽然方便，但同时也带来了一些弊病：那就是使计算机愈来愈蒙上了一层厚重的面纱，人们变得更难了解计算机内部究竟是怎样工作的了。我们建议读者在掌握计算机应用的同时，应该学习一些有关计算机的基本知识。而程序设计是计算机应用人员的基本训练和基本功，只有打好了这个基础，才能明白计算机是怎样工作的，各种应用软件是怎样设计出来的，也就能更好地使它们为人们服务。

同计算机的发展一样，程序设计语言也呈现出快速发展的趋势。尤其是随着 Windows 操作系统的广泛使用，以及 Microsoft 公司关于 Windows 应用程序的一系列标准的公布，各种专门用于编写 Windows 应用程序的程序设计语言也应运而生了。Visual Basic 正是众多编写 Windows 应用程序的设计语言之一，它是由 Microsoft 的 QBASIC 演变而来的，但它与 QBASIC 又有很大的差别。

Visual Basic 以轻松的结构化的 QBASIC 语言为基础，但又融合了可视化编程的一些新特性。它依赖于当今图形用户界面领域中十分重要的可视化元素，具有完整的基于组件设计的可视化集成开发环境，使程序员可以很快地生成 Windows 应用程序的原型和图形用户界面。

当然，Visual Basic 最大的特点也许是它将编写功能强大的 Windows 应用程序这类繁琐而复杂的工作变得简单而又十分有趣。可以这样说，Visual Basic 是目前 Windows 平台下各种编程语言中最容易掌握的一种。

全书共分十章。第一章介绍了 Windows 应用程序设计的基本概念和方法；第二章介绍了 Visual Basic 6.0 的安装及启动；第三、四章介绍了 Visual Basic 6.0 编程的一些基础知识以及简单用户界面的创建；而第五、第六章则针对 Visual Basic 6.0 中代码的编写，详细讲述了数组、过程以及程序流程控制等几个方面的内容；第七章介绍了 Visual Basic 6.0 中的大多数常用控件；第八章介绍了程序查错与调试的一些基本方法和常用技巧；第九章针对开发 Windows 应用程序的特点，专门讲述了各种程序菜单的设计方法；第十章则介绍了通过 Visual Basic 创建数据库访问程序的方法。

参加本书编写工作的人员还有冉小兵、魏宏、刘勇。在编写过程中难免会有错误和遗漏之处，敬请读者谅解。希望广大读者在通过本书的学习后，不仅

能掌握利用 Visual Basic 6.0 开发 Windows 应用程序的方法，而且能对整个 Windows 编程有个深入的了解，并能举一反三，使得今后在学习其它 Windows 编程语言时能有触类旁通的感觉。

作者

1999 年 8 月

目 录

第一章 WINDOWS 程序设计入门	1
1.1 关于计算机语言	2
1.1.1 计算机与程序	2
1.1.2 程序设计语言	3
1.1.3 程序的编译与解释	3
1.2 程序设计中的算法	4
1.2.1 算法的概念	4
1.2.2 算法的基本结构及表示方法	5
1.3 程序的界面	7
1.4 程序设计方法	9
1.5 WINDOWS 面向对象的事件驱动程序设计	10
第二章 VISUAL BASIC 的安装和启动	13
2.1 什么是 Visual Basic	14
2.1.1 Visual Basic 的由来	14
2.1.2 Visual Basic 的特点	14
2.1.3 Visual Basic 版本简介	15
2.2 VISUAL BASIC 的安装	15
2.2.1 Visual Basic 安装的系统需求	15
2.2.2 Visual Basic 的安装过程	15
2.2.3 卸载 Visual Basic	21
2.3 VISUAL BASIC 6.0 的启动及界面简介	22
2.3.1 Visual Basic 6.0 的启动	22
2.3.2 为 Visual Basic 6.0 建立桌面快捷方式	23
2.3.3 Visual Basic 6.0 的界面简介	25
2.3.4 设置 Visual Basic 的集成开发环境	29
2.4 可视化编程的几个基本概念	30
2.4.1 对象、属性、方法和事件	30
2.4.2 窗体、控件和过程	31
2.5 建立第一个 VISUAL BASIC 程序	31
2.5.1 用 Visual Basic 编程的一般步骤	31
2.5.2 建立第一个 Visual Basic 程序	31
第三章 简单用户界面的创建	38
3.1 用户界面简介	39
3.1.1 用户界面的类型	39

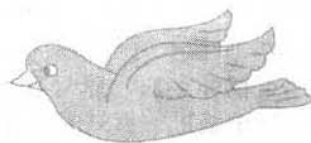
3.1.2 构成用户界面的基本元素	40
3.2 为工程添加窗体	40
3.2.1 为工程添加窗体	40
3.2.2 窗体的属性	41
3.3 设置窗体的属性	42
3.3.1 窗体的“名称”属性	42
3.3.2 窗体的 Caption 属性	42
3.3.3 窗体的大小和位置	42
3.3.4 窗体的颜色和边框样式	43
3.3.5 窗体的控制菜单和控制按钮	45
3.3.6 窗体的 WindowState 属性	46
3.4 窗体的方法和事件	46
3.4.1 窗体的方法	46
3.4.2 窗体的方法和属性的关系	49
3.4.3 窗体的事件	49
3.4.4 显示窗体	50
3.4.5 设置启动窗体	51
3.4.6 结束应用程序	52
3.5 控件概述	52
3.5.1 什么是控件	53
3.5.2 为窗体添加控件	54
3.5.3 使用控件的属性值	54
3.5.4 窗体和控件的命名约定	57
3.6 学会使用命令按钮控件	58
3.6.1 向窗体添加命令按钮	58
3.6.2 设置命令按钮的属性	59
3.6.3 命令按钮的主要事件	60
3.7 学会使用 LABEL 控件	60
3.7.1 Label 控件的用途	60
3.7.2 设置 Label 控件的属性	61
3.7.3 用标签创建访问键	62
3.8 学会使用 TEXTBOX 控件	62
3.8.1 Textbox 控件的用途	62
3.8.2 多行 TextBox 和 WordWrap	63
3.8.3 使用 TextBox 中的文本	64
3.8.4 创建只读文本框	65
3.8.5 打印字符串中的引号	65
3.8.6 取消文本框中的击键值	65

3.8.7 创建密码文本框	66
3.9 学会使用对话框	66
3.9.1 对话框的种类	66
3.9.2 使用 MsgBox 函数创建预定义对话框	67
3.9.3 用输入框来提示输入	70
3.9.4 自定义对话框	71
3.10 焦点概述	75
3.10.1 什么是焦点	75
3.10.2 Tab 键	76
第四章 VISUAL Basic 编程基础	78
4.1 VB 应用程序的结构	79
4.2 编写代码之前的准备	80
4.3 VISUAL BASIC 的代码编写机制	82
4.3.1 代码模块	82
4.3.2 使用“代码编辑器”	83
4.3.3 自动完成编码	84
4.3.4 编码基础	85
4.3.5 Visual Basic 的命名约定	86
4.4 变量、常量及数据类型	87
4.4.1 变量	88
4.4.2 理解变量的范围	91
4.4.3 在编程中使用变量	92
4.4.4 静态变量	95
4.4.5 常数	96
4.4.6 数据类型	98
第五章 数组与过程	105
5.1 数组和数组元素	106
5.2 数组的建立和引用	107
5.2.1 固定数组	107
5.2.2 动态数组	110
5.3 利用过程提高编程效率	111
5.3.1 子过程	112
5.3.2 函数过程	114
5.3.3 使用过程	115
5.3.4 向过程传递参数	117
第六章 程序流程的控制	122

6.1 关系表达式和逻辑表达式	123
6.2 选择结构的基本语句—— IF 语句	125
6.2.1 If...Then 语句	125
6.2.2 If...Then...Else 语句	126
6.2.3 处理 False 条件	127
6.3 多分支选择结构语句—— SELECT CASE 语句	128
6.4 条件循环语句—— DO...LOOP 语句	129
6.5 固定次数循环语句—— FOR...NEXT 语句	131
6.6 针对集合元素的循环—— FOR EACH...NEXT 语句	132
6.7 使用控制结构	132
6.7.1 控制结构的嵌套	132
6.7.2 控制结构的退出	133
6.7.3 退出子过程或函数过程	134
第七章 学会使用更多的控件	136
7.1 为用户提供选择的四种控件	137
7.2 使用 CHECKBOX 复选框	137
7.2.1 用 CheckBox 进行选择	137
7.2.2 CheckBox 的事件过程	138
7.2.3 CheckBox 控件示例	139
7.3 用 OPTIONBUTTON 控件进行选项分组	140
7.3.1 创建选项按钮组	141
7.3.2 在程序中使用选项按钮	142
7.3.3 OptionButton 控件示例	143
7.4 使用 LISTBOX (列表框) 和 COMBOBOX (组合框)	145
7.4.1 在 ListBox 中进行选择	146
7.4.2 更加灵活的 ComboBox	151
7.5 学会使用 SCROLLBAR 控件	153
7.5.1 ScrollBar (滚动条) 的应用	153
7.5.2 用 ScrollBar 创建可滚动的图形窗口	154
7.6 掌握一些定时器 (TIMER 控件) 的知识	158
7.6.1 定制 Timer 控件	158
7.6.2 Timer 控件编程示例	159
第八章 程序查错与调试	162
8.1 程序中的几种错误	163
8.2 避免编译错误	164
8.3 如何处理运行错误和逻辑错误	165
8.4 调试工具如何工作	167

8.5 应用程序的三种状态	168
8.6 如何监测变量的值	169
8.6.1 使用“监视”窗口	170
8.6.2 使用“本地”窗口	172
8.6.3 使用“快速监视”	172
8.7 使用“立即”窗口	174
8.7.1 利用“立即”窗口测试表达式的值	174
8.7.2 通过“立即”窗口给变量或属性赋值	175
8.8 调用堆栈	175
8.9 程序中的断点设置	176
8.9.1 在程序中设置断点	176
8.9.2 使用 STOP 语句	177
8.10 跟踪代码的执行	178
第九章 为应用程序建立菜单	180
9.1 用菜单项控制程序	181
9.2 规划菜单组	182
9.3 建立主菜单	183
9.3.1 创建菜单控件	183
9.3.2 分隔菜单项	185
9.3.3 设置访问键和快捷键	186
9.4 菜单的命名规则	188
9.4.1 对菜单标题的赋值	188
9.4.2 确定菜单的 Name 属性	188
9.5 创建子菜单	188
9.6 为菜单控件编写代码	189
9.7 设计弹出式菜单	190
第十章 访问数据库	193
10.1 理解数据库	194
10.2 关于数据控件	195
10.2.1 什么是数据控件	195
10.2.2 向窗体中添加数据控件	196
10.2.3 与数据库的连接	196
10.3 数据绑定控件	198
10.3.1 绑定控件的作用	199
10.3.2 绑定控件的类型	199
10.3.3 在程序中使用绑定控件	200
10.4 创建简单的数据库应用程序	202

10.5 添加和删除记录	203
10.5.1 为数据库添加记录	203
10.5.2 删除数据库中的记录	204
附录 A ANSI 字符集 (0 ~ 127)	205
附录 B 数据类型	207
数据类型概述	207
Boolean 数据类型	208
Byte 数据类型	208
Currency 数据类型	208
Date 数据类型	208
Decimal 数据类型	208
Double 数据类型	209
Integer 数据类型	209
Long 数据类型	209
Object 数据类型	209
Single 数据类型	209
String 数据类型	209
用户定义数据类型	210
Variant 数据类型	210
附录 C 运算符	212



第一章

Windows 程序设计入门

本章要点

- 计算机与程序
- 编程语言与算法
- 结构化程序设计中常用的逻辑结构
- 程序的界面设计
- 程序设计的一般步骤
- Windows 程序设计的新特性

如果您是一个电脑爱好者，那您就一定知道，Microsoft 公司于 1998 年 6 月推出了其最新的操作系统——Windows 98，而且它还以最快的速度陆续推出了 Windows 98 的中文版、日文版和俄文版等不同语言的版本。作为一种全新的计算机操作系统，Windows 自从 90 年代初一诞生，就迅速地风靡了全球。与以往所使用的操作系统——DOS 不同，Windows 用若干组布置美观、意思明确的图形窗口取代了 DOS 中许多复杂难懂的命令，可以通过鼠标点取，非常直观地完成许多功能，这使得计算机使用起来更容易，也使计算机的使用者也更容易明白计算机正在干些什么。正是由于这些优点，Windows 现在几乎已经成了 PC 机上唯一可选的操作系统了。而许多软件供应商们也正在把他们以前基于 DOS 平台编写的软件改写为支持 Windows 操作系统的新版本，以适应当今电脑技术的发展。可以说，支持 Windows 操作系统的编程已经成为当今 PC 机编程的主流。

虽然 Windows 平台对用户十分友好，但对程序开发者来说却较难编写，直到引入了像 Visual Basic 这样专业的可视化环境开发工具后，才使创建 Windows 应用程序较为容易。

在深入了解 Visual Basic 之前，让先来了解一下有关计算机和编程的一些历史，以及一些可适用于任何编程语言的有关编程的基本概念。理解这些有关编程基础的概念，有助于今后编制出更好、更有效的程序。

1.1 关于计算机语言

1.1.1 计算机与程序

自 20 世纪 40 年代世界上第一台计算机问世至今，短短 50 年的时间，电子计算机已经经历了一个翻天覆地的过程。现代科学技术的发展更是加速了电脑的发展步伐，同时也极大地降低了电脑的生产成本，以至于现在微型电子计算机，也就是俗称的个人电脑，已经像日常的家用电器一样深入到了社会生活中的每一个角落。

电子计算机发展得如此迅猛，那么它究竟能够为人类干些什么，又是怎样去干的呢？简单地讲，人类大多数的不具创造性的工作都可以由电子计算机来完成。完成这些工作，计算机依靠的是人们事先编制好并输入机内的程序。

人们做任何事情都有一定的步骤。利用计算机完成一项工作也有一定的步骤：事先对问题进行分析→确定解决问题的方法与步骤（即算法）→根据算法编制出计算机程序→让计算机执行这个程序，得出最后的结果。计算机操作的每一步都是在相应的命令（这里称为指令）操纵下进行的；不同的指令序列，可以操纵机器完成不同的工作。把这些指令的有序集合就称为程序，由计算机程序员编写的程序也称为软件。

当我们用电子计算器进行数学计算时，程序就存在于的大脑之中，大脑发出一条指令，计算器就执行一步相应的操作。而要使电子计算机实现自动操作，就必须把程序从人脑中分离出来，让计算机记住，并按程序的规定顺序地执行一系列指令。这就是通常所说的程序控制，现在所有的计算机都是在程序控制下进行工作的。

正如小说是由一系列适当安排的句子而形成完整的故事一样，计算机程序也是

由一系列能使计算机完成特定功能的语句按照一定的逻辑关系排列组成的。使用计算机做任何事，如文字编辑排版、机械设计、统计报表、甚至玩游戏等，如果不输入适当的程序，就不能完成。程序就像是计算机的大脑和灵魂一样，如果没有它，计算机就是摆在桌子上的一堆废铜烂铁。正是有了程序，才使计算机成为人们生活中不可缺少的一部分。

1.1.2 程序设计语言

既然计算机是在人们预先编制的程序下进行有目的的工作，那么，人们是用什么形式把程序“告诉”计算机的呢？

通常把表示和交流信息的符号系统及规则称为“语言”，并把给计算机编制程序的符号系统及规则称为计算机的编程语言，简称程序语言或计算机语言。程序语言随着计算机的发展，由低级到高级不断完善、逐步简化和方便。目前计算机的语言大致可分为机器语言、汇编语言和高级语言三大类。

如果可以用日常所使用的语言来编写程序，那么程序员们就将省力得多。遗憾的是，无论哪种计算机都只能执行一种语言，那就是通常说的机器语言。机器语言是由一系列由“0”和“1”组成的二进制代码所构成的，这些代码代表了机器所能执行的程序指令。但是，对程序员来说，机器代码却不能被用来进行快速有效的编程，且难于学习和掌握。而汇编语言可以让程序员直接对计算机的各类端口、寄存器等硬件进行操作，其程序代码多使用命令关键字加 16 进制代码构成。虽然汇编语言的直观性要好于机器语言，但它们都是低级语言，是一种面向机器的语言。用这些语言编制的程序具有占用内存少，执行速度快等优点。然而他们的直观性和可读性较差，而且受具体机器指令系统的束缚，移植性差，编程效率很低。

随着计算机技术的高速发展，人们设想能够寻求一种可摆脱机器指令系统的束缚、通用性强、又比较接近人的语言习惯的计算机编程语言。高级语言的开发成功使人们的这种设想变成了现实。高级语言是一种面向过程或是面向对象的语言，使用这种语言编程，基本上摆脱了机器指令系统的约束，程序员无须再花很大的气力来了解机器内部的逻辑结构，可以集中精力去推敲解题的思路。但高级语言与低级语言相比，占用的内存较大，执行速度较慢，某些情况下还不能代替低级语言。

☞注意： 编程语言的发展与计算机硬件的发展不同，它不是高一代取代低一代，而是多代共存的。

高级语言从 50 年代中后期发展到今天，全世界大约已经有了 1000 多种，并且新的语言还在不断出现。但这些语言中大多数都是一些专用语言，真正在国际上得到广泛应用的只有十几种，如 BASIC、FORTRAN、COBOL、PASCAL、C、ALGOL 68、LISP 及 PROLOG 等等。

1.1.3 程序的编译与解释

用高级语言编写的程序语句称为程序的源代码，这些源代码只有被翻译成机器语言

后才能被计算机执行。根据翻译的方式，可分为编译型和解释型两类。无论是编译方式还是解释方式，都有一个编译程序或解释程序被事先放在机器中。编译方式是，源程序输入计算机后，编译程序将它们全部翻译成机器代码后形成一个磁盘文件存储在计算机中。程序运行时，只需运行这个磁盘文件即可，而不必再进行翻译。而解释方式是，源程序输入后，解释程序对源程序进行逐句翻译，翻译一句交机器执行一句，因而解释方式比编译方式占用内存少，但执行速度较慢。

1.2 程序设计中的算法

1.2.1 算法的概念

人们做任何一件事情都有一定的步骤。同样，计算机也是按程序所规定的内容和步骤进行工作的。因而，当在计算机上为解决一个问题而进行程序设计时，至少应该具备以下两个条件：

首先，必须掌握解决问题的方法和步骤。也就是说，在拿到一个需要求解的问题后，经思考先确定其解决方法，然后将其分解成一系列可实施的操作步骤；

其次，应至少掌握一门高级语言，以使能将所确定的解决问题的方法和步骤通过程序设计来告诉计算机。

为解决一个问题而采取的方法和步骤，就是通常所说的“算法”。广义地说，处理任何问题都有一个“算法”问题。解决一个问题的过程就是实现一个算法的过程。例如，一个菜谱就是一个“算法”，厨师炒菜就是实现这个“算法”。所以，当利用计算机解决一个问题时，实际上就包括了算法的设计和算法的实现两部分工作。可以说，程序设计的灵魂就是算法的设计，而语言只是表达算法的一种形式。有了正确的算法，可以利用任何一种语言来编写程序，使计算机工作，得出正确的结果。

对同一个问题，往往有不同的解题方法。例如要计算：

$1+2+3+\cdots+100$ ，即 $\sum_{n=1}^{100} n$ 的值，就有几种不同的方法。可以先进行 $1+2$ ，再加 3 ，

再加 4 ，一直加到 100 ，得到结果 5050 。也可以采取另外的办法，

$\sum_{n=1}^{100} n = 100 + (99+1) + (98+2) + \cdots + (51+49) + 50 = 100 + 49 \times 100 + 50 = 5050$ 。当然，还可以有其

它的许多方法。人们希望采用好的算法，即方法简单、运算步骤少，又能迅速得出正确结果的算法。因此，为了有效地解决问题，不仅需要保证算法正确，还要考虑算法的质量。用计算机解决问题，则要选择适合计算机的算法。掌握了算法的设计，无论用哪一种语言编写程序都不会太困难。

计算机算法可分为两大类：数值运算算法和非数值运算算法。数值运算算法的目的是求数值解。例如求方程的根，求函数的定积分等，都属数值运算范围。而非数值运算

包括的范围则很广，最常见的是用于管理领域，如人事档案管理、图书情报资料检索、航空气调度等。很多算法现在已经研究得比较成熟，特别是数值运算方面的算法。而非数值运算的算法由于种类比较多，难以规范化，因此往往需要使用者参考已有的类似算法重新设计解决特定问题的专门算法。

1.2.2 算法的基本结构及表示方法

1966 年，Bohra 和 Jacopini 提出了算法的三种基本结构：

- (1) 顺序结构
- (2) 选择结构（或称分支结构）
- (3) 循环结构（或称重复结构）

这三种基本结构是一个良好算法的基本单元。已经证明，由以上三种基本结构顺序组成的算法，可以解决任何复杂的问题。由基本结构构成的算法属于“结构化”的算法，它不存在无规律的转移，只有在本结构内才允许存在分支或向前向后的跳转。这种由基本结构顺序组成的算法便于阅读和修改，从而使算法的可靠性和可维护性得到了保证。

设计一种算法，必然要用一种方式将其表示出来，N-S 结构流程图则是表示算法的一种较为理想的形式。它是在 1973 年由美国学者 I.Nassi 和 B.Shneiderman 提出的。在这种流程图中，全部算法写在一个大矩形框中，在该框内还可以包含一些从属于它的小矩形框。换句话说，它是由一些基本的矩形框组成一个大的矩形框，而且只有三种基本的元素框，分别表示算法的三种基本结构。

在 N-S 结构流程图中用以下一些流程符号来表示算法的三种基本结构。

1. 顺序结构

图 1-1 是用 N-S 流程图表示的顺序结构，其中 A 和 B 两个框内的操作是顺序执行的。

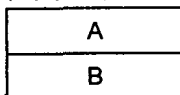


图 1-1 顺序结构

2. 选择结构

如图 1-2 所示，当 P 条件成立时执行 A 操作，P 不成立时则执行 B 操作。注意，图 1-2 所示的是一个整体，代表一个基本结构。

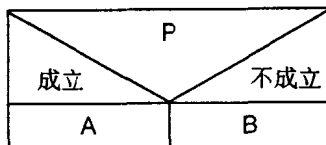


图 1-2 选择结构

3. 循环结构

循环结构的作用是反复执行某一部分操作。它有两类循环结构：当型循环和直到型循环。

(1) 当型循环 (WHILE 型循环)。当型循环的特点是：当指定的条件满足时，就执行循环体，否则就不执行。当型循环又分两种形式：

①. “前测试”型 先测试条件，后执行循环体，如图 1-3 (a) 所示。在执行图 1-3 (a) 所示的循环结构时，先检查 P1 条件是否成立，如果成立就执行循环体 A。然后再检查 P1 是否成立，若成立再执行 A，如此反复到某一次 P1 条件不成立时，就不再执行 A，循环结束。

②. “后测试”型 先执行循环体，然后再测试条件是否成立，如图 1-3 (b) 所示。从图 1-3 (b) 中可以看出：先执行循环体 A，然后才测试条件 P1 是否成立，如果 P1 不成立，则不再返回执行 A。由此可知，后测试当型循环至少执行循环体一次 (前测试的当型循环，若开始时条件 P1 不成立，则一次也不执行循环体)。这是前测试型和后测试型的区别。

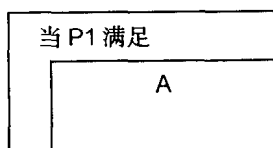
当型循环可以概括为：“当 P1 条件满足时，反复执行循环体”。

(2) 直到型循环 (UNTIL 型循环)。直到型循环的特点是：执行循环体直到指定的条件满足，此时就不再执行循环。

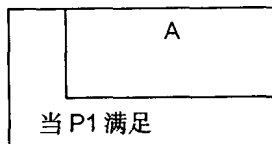
直到型循环同样分为“前测试”型和“后测试”型。

①. “前测试”型 先测试条件，后执行循环体，如图 1-3 (c) 所示。从图 1-3 (c) 中可以看到：执行此循环结构时，先测试条件 P2 是否成立，如成立，就不执行循环体 A，如不成立就执行循环体 A。

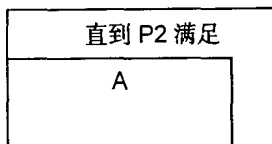
②. “后测试”型 先执行循环体，然后再测试条件是否成立，如图 1-3 (d) 所示。从图可知其执行过程为：先执行一次循环体 A，然后再检查条件 P2 是否满足，如果不满足就重复执行循环体 A，然后再对 P2 条件作测试，如果 P2 仍不满足，又执行循环体 A……直到 P2 条件满足，就结束循环过程。



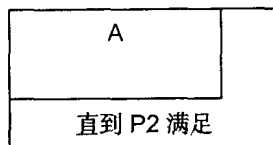
(a) 前测试当型循环



(b) 后测试当型循环



(c) 前测试直到型循环



(d) 后测试直到型循环

图 1-3

直到型循环可以概括为：“反复执行循环体，直到 P2 条件满足。”

图 1-1~图 1-3 中的“A”或“B”可以是一种简单的操作，也可以是三种基本结构之一。

☺提示：基本结构之间是可以互相嵌套的，在一个基本结构中又可以包含一个或多个基本结构，例如在循环结构中 can 包含一个选择结构。

N-S 流程图尤其适合于表示一个结构化的算法，用于结构化的程序设计。

【例 1-1】求 $1 \times 2 \times 3 \times 4 \times \cdots \times n$ ，即 $n!$ ，直到 $n! \geq 1000$ 为止。用 N-S 流程图表示该题的算法。

分析该题，设一个变量 t 用来存放求出来的阶乘值，在累乘的过程中 t 的值是变化的。另设一个变量 i ，用来存放累乘数 1, 2, 3, ... 算法可用图 1-4 表示。

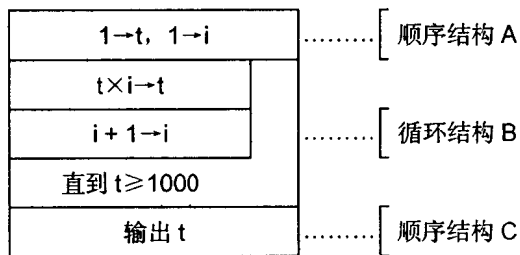


图 1-4 算法图示

可以看出，这是一个后测试直到型循环：先执行循环体，然后检查“ $t \geq 1000$ ”条件是否满足，满足就循环。它可以概括为：执行阶乘操作，直到条件“ $t \geq 1000$ ”成立为止。这个算法由三个基本结构顺序构成，分别是顺序结构 A → 循环结构 B → 顺序结构 C 三个部分。

☞注意：在 N-S 流程图中，基本元素框在流程图中的上下顺序就是执行时的顺序，也就是：图中位置在上面的先执行，位置在下面的后执行。写算法和看算法只需按从上到下的顺序进行就可以了，十分方便。

作为一个刚尝试程序设计的初学者来说，学会用流程图来表示程序的算法设计是非常重要的。流程图是整个程序设计的蓝图，它详细描述了程序的主要步骤、判断条件和决定程序流程的分支。绘制流程图实际上可以帮助考虑程序的设计策略。

1.3 程序的界面

程序设计中另一个需要着重考虑的方面就是程序的用户界面。程序的用户界面是指程序在屏幕上的外观以及用于与用户进行数据通讯的访问窗口的总称。对于一个应用程