微计算机丛书

# IBM PC Pascal 程序设计

## (DOS和UCSD P-系统Pascal)

〔美〕K. W. 鲍耶 S. J. 汤姆布林 著
张书杰 何德祥 译 徐则琨 潘国楠 校



電子工業出版社

# IBM PC Pascal 程序设计

## (DOS和UCSD p-系统Pascal)

〔美〕K·W·鲍耶　S·J·汤姆布林　著
张书杰　何德祥　译　徐则琨　潘国楠　校

電子工業出版社

## 内 容 简 介

本书介绍的是用于IBM PC的Pascal程序设计语言及方法。针对DOS Pascal和UCSD P-系统Pascal。

主要内容有：Pascal的基本特性；选择控制结构；循环结构；数组；过程和函数；用户自定义类型；文件的I/O；声音和图形及程序开发等。

本书适合于Pascal的初学者及广大IBM PC用户、教师及科技工作者阅读。


Pascal for the IBM PC

IBM DOS Pascal

and

UCSD p-System Pascal

Kevin W.Bowyer

Sherryl J.Tomboulian

Robert J.Brady Co.(1983)

**IBM PC Pascal 程序设计**

(DOS和UCSD p-系统Pascal)

〔美〕K·W·鲍耶 S·J·汤姆布林 著

张书杰 何德祥 译 徐则琨 潘国楠 校

责任编辑 王惠民

*

电子工业出版社出版（北京市万寿路）

新华书店北京发行所发行 各地新华书店经售

北京市通县曙光印刷厂印刷

*

开本：787×1092 1/16 印张：15.125 字数：346千字

1987年11月第一版 1987年11月第一次印刷

印数：1～7000册 定价：3.05元

统一书号：15290·574

ISBN7-5053-0104-7/TN64

# 前　言

本书是IBM PC Pascal程序设计的入门书，对象是Pascal的初学者。读者即使从未写过程序也能有效地使用本书。对已有编程经验的读者，可能开头几章会显得进度过慢，但定会有不少收益。这几章通过一些完整的实例程序显示了IBM PC Pascal所独有的特性，包括图、声音、文件处理及其它一些标准Pascal的扩充性能。

除了大量的实例外，还有一些练习和问题，以助自学。练习用以加强基本概念，其答案随附题后。问题因要求稍高，未给出答案。

用于IBM PC的Pascal已有多种型式，用得最普遍的是DOS Pascal，其次是UCSD p-系统Pascal。DOS Pascal在IBM DOS操作系统下运行，后者在p系统下运行，它们都与所谓的“标准”Pascal有所不同。

本书包含了DOS Pascal和p-系统pascal。这样做的原因是多方面的，其一是为了使本书适用于更广泛的读者，其二是任何一种Pascal的特性中，总有许多是非标准的。我们希望通过标准Pascal的两种常用不同扩充方法的讨论，使读者意识到标准Pascal的局限性。在换用其它操作系统或计算机的Pascal时，DOS Pascal和p-系统Pascal正是需要重新学习的东西。

第一章引导读者了解适合于Pascal的PC配置，需认真阅读。第二、三章讨论Pascal的基本特性，包括输入和输出。第四章阐述选择控制结构。第五章是循环结构。选择和循环是改变程序控制流的主要方法。第六章着眼于数组。第七章论述过程和函数。至此，Pascal的所有基本特性讨论完毕。第八章讨论程序开发。第一章到第八章可以作为程序设计引论课程。第九章涉及用户自定义类型，这是Pascal的一个高级题目，应仔细阅读，从而真正体会到Pascal不同于其它语言的风格。第十章论及了适用于文件的I/O。第十一章讨论声音和图形的发生。随着读者编程能力的提高，这两章所介绍的各种非标准特性将会有更多的用处。

# 目　录

# 第一章　关于IBM PC的一些情况

## 第一节　操作系统和Pascal的类型

在写这本书时，IBM公司已经把PC的三种不同的操作系统投入市场：DOS、UCSD p-系统和CP/M-86。而且，其它的卖主也正在销售UCSD p-系统和CP/M的一些竞争性版本，还有一些类似于UNIX操作系统。用于PC的操作系统的种类肯定还会增长，它们之中的大部分很可能有它们自己的Pascal版本。虽然现在有“标准”Pascal，但要建立一种且仅仅执行这种标准Pascal的编译程序或许是不可能的。

在这本书中我们只考虑IBM DOS Pascal编译程序和UCSD p-系统的编译程序。书中的一些章节叙述了两种Pascal的区别，这在目录中已清楚地标明。

DOS和p-系统Pascal之间一个基本的差别是p-系统Pascal不被全部地编译。编译程序把高级语言写的程序翻译为计算机内处理器实际使用的语言。IBM PC以INTEL8088微处理器为基础，DOS Pascal编译是把Pascal语言源程序翻译为可以由8088CPU直接执行的机器代码。p-系统编译程序是把Pascal翻译为称为“P代码”。P代码比Pascal更接近于机器语言，但比任何一种特殊处理器语言更为通用化。在Pascal程序执行时，该程序的P代码形式被翻译为准确的INTEL8088语言（这个过程被认为是解释地执行）。作为一种结果，在DOS Pascal下所写的程序一般要比在P-系统下所写的程序执行得快。但是，使用p-系统的注释代码发生器会挽回速度方面的某些损失。在“p-系统用户指南”中对p-系统编译程序的选择项进行了说明。

DOS和p-系统的软盘文件格式是不兼容的。也就是说，在一种操作系统下建立的文件不能直接被在另一操作系统下运行的程序所存取。

## 第二节　显　示　设　备

读者对PC的显示设备的选择实际上是从连接器的选择开始的。选用单色监控连接器自然会局限于使用单色监控显示设备（之所以称为单色而不叫黑白是因为显示器的磷光质是绿色）。单色显示器的主要优点是高质量的正文显示，而主要的缺点是只能显示正文而不能显示图形。

选用彩色/图形连接器，则可选择的显示设备就多些。你可以使用：

1. 家用黑白电视机。
2. 家用彩色电视机。
3. 标准黑白监控器。
4. 标准彩色监控器，复合型或RGB型。

1 和 2 项选择需要购买RF调制器，连接在彩色/图形连接器和TV之间。第 4 种选择可能需要一条电缆连接监控器。从第 1 到第 4 的任一种选择都可以做图形操作。图形功能取决于连接器，而不取决于监控器，但显示器决定图形的显示质量。彩色图形只能由第 2 项和第 4 项选择来做，因为彩色的形成是显示设备的一种性能。

## 第三节　声音和图形

如果使用p-系统Pascal，那么IBM SPECIAL和TURTLE GRAPHICS库中的一些命令可用于产生声音和图形。如果使用DOS Pascal，系统没有标准命令，使用声音和图形就不那么容易了。

我们将在第十一章介绍一组声音和图形的命令。

## 第四节　文本编辑、编译和执行

文本编辑器是一种程序，它用来从键盘输入正文并把正文以文件的形式存贮在盘上。通过文本编辑器输入的Pascal正文文件由编译程序翻译为可被系统处理器实际使用的格式。

如果你使用UCSD p-系统，那么适用于屏幕的编辑器便是系统的一个标准部分。适于屏幕的“编辑器”与适于行的“编辑器”是有区别的，屏幕编辑器允许你在屏幕的任何地方使正文变化，而行编辑器需你确定一特定的行，而且要给出针对该行的编辑命令。企图用一种更好的编辑器来取代标准的UCSD p-系统屏幕编辑器会有很多麻烦，不值得尝试。

如果你使用DOS1.0或1.1版本，那么随系统所带的标准编辑器是相当简单的行编辑器（EDLIN）。你可能要认真地考虑来购买屏幕编辑器，它在建立程序文本方面将会给予你更强的功能和更好的灵活性，会明显地提高你的工作效率。可以应用的屏幕编辑器有数十种。对于DOS来说，以Unicorn商标经销的MINCE编辑器是相当不错的。你至少应当走访当地的计算机商店，看看在这方面有什么可取的。

附录A和B详细地叙述了编辑、编译和执行程序所必须的一些过程。其中所描述的编辑器是标准的UCSD p-系统屏幕编辑器和DOS EDLIN编辑器，如果你以前没有在PC上准备和执行过程序，那么，现在你可以看一看相应的附录。读这些附录可以得到有关的概念。当你编写头几个程序例子时，可能你还需要参看这些附录。最终还要参阅你的系统所带的资料，以便学习编辑器和编译程序的一些高级性能。

## 第五节　关于本书的应用

本书是围绕着如下几个原则而写的：

- 开头要慢
- 边干边学
- 各章节的自然顺序不是针对有经验之人，而是针对初学者的。

这本书对各个专题的讨论是慢慢展开的，自认在Pascal方面已经是专家的那些人可能会不喜欢本书中的课题顺序。Pascal的某些重要属性，象用户自定义的类型，被安排在本书的

最后，其原因是我们相信初学者并不具备与专家们相同的接受能力。

书中有大量的例子，都是些完整的程序。应运行这些程序！我们认为，要通过运行这些程序来学习，而不是阅读语言的抽象描述（如果你不想亲自打入这些程序，可订购这些程序的软盘①，软盘具备在屏幕上把程序列表的功能）。每个程序例子阐明一个或几个要点，应该逐个做。通读程序文本并理解它与运行程序并体会其操作是二个不同的知识水平。

尽管本书是针对初学者的，但仍然包括了对已熟练使用Pascal者的有用的材料。这些行家们不难根据此书的结构找到阅读的方法，跳过他们不需要的部分。初学者则需要所有部分，本书的结构对他们来说是适用的。

② 本译本没有此软盘——译者

# 第二章　Pascal 初论

## 第一节　程序格式和简单输出

### 一、程序的概念

程序是计算机能够执行的一些命令及其有关信息的组合。可以把程序看作类似“食谱”的东西。你从基本的信息——原料和匙子、搅拌杯及量杯等一些所需的器皿着手，然后按照所列出的步骤得出结果。然而要注意，得到最后产品的方法可能不只一种。制作巧克力薄饼的食谱就有许多种。但是，你应当按正确的食谱来做。如果你要制作安琪蛋糕，就不能使用薄饼的食谱。

### 二、什么是计算机语言？

要与计算机通信，必须使用一种计算机语言——在这里是Pascal。还是用烹调作比喻，食谱使用一种专门的语言，即标准英语和有关烹调的术语缩写的组合。Pascal遵循同样的原理。只要可行就用类似英语的短语，另外还有计算机环境所特有的专门词汇。

要提醒读者的是，学习一种语言要掌握一些词汇和一套结构规则，然后才能实际应用。在学习外语时，从最简单的短语开始，如“Good morning, sir, How are you?”和“I am fine, Thank you。”而不会从阅读古典文学起步。学习计算机语言也是这样，也要从基本的语句和结构开始。正因为如此，本书的许多练习和例子看起来都是人为的。事实上，大多数只是为了说明某一特性的应用。要有耐心，一旦掌握了各个结构块，你便可能有长足的前进，解决你所感兴趣的重要问题。

### 三、Pascal程序结构

一个程序的结构包括：说明程序名的标题，表明程序开始的BEGIN，描述要执行的动作的命令和最后的 END 语句。标题是由词 PROGRAM、程序名和分号组成。BEGIN 和 END标志着命令的开始和完成，它们是必须的，若是没有它们，计算机便不会知道程序在何处开始和结束。回到食谱的比喻上来，若没有BEGIN/END的话，会有可能把蛋糕和蛋奶酥混合起来，最后得到的是极糟的大杂烩。

例如：

```
PROGRAM name_you_give_it;
BEGIN
    { statements }
END.
```

注意，在程序的标题后要使用分号，分号告诉计算机一条语句完毕。计算机术语把分号

说成是两个语句之间的界线。界线只是把东西隔开的一种方法。也并不总是用分号，这在后面会讲到。

### 四、什么是语法？

语法指的是定义语言结构的规则。例如，英语句子头一个词的第一个字母要大写，最后用句点来结束，这便是英语语法的一部分。Pascal的语法要求按照上述的格式来定义程序。

到此，我们只是知道了一个Pascal程序的外表轮廓，为了写程序我们需要某些实际操作的语句。WRITELN语句将在显示器上写一些信息然后跳到下一行。WRITELN语句的语法是在词WRITELN后接一个起始圆括号，一个引号后接着一些所要求的字符（其长度以一行能容纳为限），一个终结引号再跟着一个终结圆括号。

例如：

```
WRITELN ('any text you like');
```

WRITELN语句要位于程序的BEGIN和END之间。把下面的程序(我们的第一个)键入编辑器，若在此时你不知如何进行的话，请参阅用于DOS pascal的附录A或用于p-系统Pascal的附录B。

**程序2.1**

```
program sample2_1;
BEGIN
   WRITELN('This is an example.');
   WRITELN('I am going to be a great programmer.');
END.
```

一旦程序已经键入，你便希望执行它，但现在还不行。首先要通过编译程序将它翻译为不同的格式，编译程序将Pascal翻译为机器能够执行的简单语言（这种编译程序的使用在附录A和B中也作了描述）。Pascal能被人理解，但不能被机器理解。你的Pascal程序必须被翻译为计算机可用的格式。

编译程序将扫描每一行程序并力图将其翻译。遗憾的是编译程序是愚笨的，它只能忠实地照你的话去做，既不能多做也不能少做。正是由于这样，在编译时通常会产生一些语法错误。语法错误意味着编译程序不理解程序中的某个内容。举例来说，如果你在END后面遗漏了句点，那么就产生语法错。当这样的错误发生时，你必须重新返到编辑程序，修正错误再试图编译。

还要提醒两句话。当你刚刚着手编写程序时会出现许多错误，每个人都是如此。你可能有点灰心，但是当你得到更多的实践之后便会大有进步。要提醒的第二点是：当你得到与打字错误有关的信息以及编译程序告知的出错行时，可能并不是真正地说明你所犯的错误。举例来说，如果在语句中丢掉分号，那么在这个语句之后才会发现。当发现错误后，你只是知道在编译程序指示之处以前的某个地方有错误。

一旦程序已被编译便可以执行。计算机执行程序的部分称为处理机。处理机取出每一行程序，依次地执行专门的任务。当程序2.1运行时，下面这几行将显示在屏幕上：

```
This is an example.
I am going to be a great programmer.
```

## 五、注释语句

如上所述，编译程序仔细地检查每一行程序以保证与Pascal语法相一致。但是也有例外，不管什么只要放在花括号{ }内或带有星号的圆括号（**）内，那么编译程序便不予考虑。这是一种在正文中插入注释的方法，你可以用注释语句来给自己注释或向今后要读你的程序的人提供信息。下面的例子产生与程序2.1完全相同的结果。

程序2.2

```
program sample2_2;
  { this is a program that demonstrates the use of comments }
(* author: jane programmer        written: feb 2, 1982 *)
BEGIN
  WRITELN('This is an example.');   {comment after statement}
  WRITELN('I am going to be a great programmer.');
END.
```

我们使用花括号是因为它比带星号的圆括号敲的键要少。在Pascal中这两者都是有效的，但对于在键盘上没有花括号的计算机来说，就只能用带星号的圆括号了。应当养成采用同样的通用方式注释你所有的程序的习惯。使用注释语句是为了使你的程序目的和操作尽可能让读者明白。我们已经介绍了注释程序的作者和时间的惯例，后面我们还要在使用注释语句方面给予更多的介绍。

为了使程序更加易读，还可以在程序之中设置一些空行或空格。注意，我们把BEGIN和END块中的所有语句一起后缩了几格。

### 小　结

1. 程序的语法是：

```
PROGRAM name_you_make_up;
BEGIN
  { statements }
END.
```

2. WRITELN用来向显示器输出信息，在引号里无论什么东西都要印出，进一步的输出语句要在下一行开始。其语法是：

```
WRITLN('Anything you like as long as it fits on one line.');
```

3. 要运行一个程序则需要编译它，编译程序把Pascal语句翻译为计算机可用的格式，你可以执行编译后的程序版本。

4. 语法错意味着一个语句不合乎Pascal语言规则。编译程序可以检测出语法错误，在一行的末尾遗漏分号、丢掉BEGIN或END、在END后丢掉句点、在WRITELN语句中丢掉一个或两个引号都是最常犯的错误。

## 练　习

1. 修改这一节的例子程序，使其第一个输出行是：

```
''This is another example.''
```

2. 修改例子使其输出第3行：

```
"This is fun. RAH! RAH! RAH!"
```

3. 建立一个程序，它印出“忠诚誓约”中的第一行。在程序中放入注释语句描述这个程序要做什么。

## 解　答

1.
```
program solution;
BEGIN
   WRITELN('This is another example.');
   WRITELN('I am going to be a great programmer.');
```

2.
```
program solution;
BEGIN
  WRITELN('This is an example.');
  WRITELN('I am going to be a great programmer.');
  WRITELN('This is fun. RAH! RAH! RAH!');
END.
```

3.
```
program solution;
{   This program prints the first lime of the pledge of
    allegiance. }
BEGIN
   WRITELN('I pledge allegiance to the flag');
   WRITELN('of the United States  of America.');
END.
```

# 第二节　有关输出的进一步叙述

Pascal语言有两个十分相似的输出语句：WRITELN和WRITE语句。我们已经用WRITELN语句输出程序信息。输出的信息都是包含在两个单引号内的一串正文，每个语句只有一串，而且每一个语句的信息只能出现在它自己的行内。这一节我们将学习如何利用一个输出语句输出几项信息，如何既能印出正文又能印出数值结果，以及如何实现几个语句在同一行上输出，而不是每个语句各印一行。

首先，让我们来看程序2.3，这个程序有5个WRITELN语句，每一个输出一个字母串——用计算机术语来说就是正文串。在两个单引号之间字母串可以有任意的字符顺序，但不能跨越程序行。有一个问题读者可能已经意识到了，这就是如何使得单引号在字母串中出现。这样来解决这个问题：把两个单引号一起放在字母串中你要输出一个单引号的地方。程序2,3的第4个WRITELN语句说明了这个问题。

程序2.3

```
program sample2_3;
{ auther: joe programmer      written:  feb 3, 1982
  this is a program to demonstrate the WRITELN statement }
BEGIN
  writeln ('This ');
  writeln ('is ');
  writeln ('how ');
  writeln ('''WRITELN''');{double quote makes one on output }
  writeln ('works ');
END.
```

程序2.3的输出如下：

```
This
is
how
'WRITELN'
works
```

WRITE语句的作用类似于WRITELN语句，但它印出之后不能自动地换行，它从上一语句停止输出的地方开始输出。为了理解这一点，请把程序2.3中凡是出现“writeln”的地方都改写为“write”。将其修改成程序2.4。读者仅用一、二个编辑命令就可以完成这样的改动。如果你不知道简便的改动办法，那么就请多花些时间阅读一下编辑程序的资料。

程序2.4的目的是显示一个单行输出。

```
This is how 'WRITE' works
```

每个WRITE语句开始于上一语句的输出终止处，所有5个WRITE的输出都在一行上。

在往下讲之前，我们对WRITE、WRITELN和一般的语句再多讲几句。程序2.5展示了这些语句。

程序2.4

```
program sample2_4;
{  auther: joe programmer      written:  feb 4, 1982
   this is program to demonstrate the WRITE statement }
BEGIN
  write ('This ');
  write ('is ');
  write ('how ');
  write ('''WRITE ''');  { double quote makes one on output }
  write ('works ');
END.
```

注意，程序2.5中的头一个WRITE语句跨了三行，随后两个WRITELN语句出现在同一行上。Pascal语句可以按照自由格式键入，也就是说它们不一定要出现在一行的特定位置。这种自由度可用来使程序排列得尽可能的可读（不应是程序2.5的排列）。第二个WRITELN语句的每个输出字符都作为输出表列中的独立项。WRITE和WRITELN两个语句都可以包括一些要输出的表列，表列中的元素由逗号隔开。当输出正好是某一个字母串的字符时，这样做会增加许多额外的工作量，但是当你要求数字和正文混合输出时就会带来方便。这在下一节要讲到。判断程序2.5的输出，然后执行程序2.5，以验证你的判断是否正确。

程序2.5

```
program sample2_5;
{   author: joe programmer     written:  feb 5, 1982
    this is a program to demonstrate free-format typing,
    more about WRITE and WRITELN                    }
BEGIN
  write
          ('This ')
                    ;
  writeln ('is ');        writeln ('h','o','w');
  writeln ('''WRITE'' and ''WRITELN'' mixed together ');
  write ('work ');
END.
```

## 第三节 数 值

Pascal语言中关于数字的操作基本上遵循标准的算法。加号（+）表示加法，减号（-）表示减法，星号（*）表示乘法（除法请看后面的讨论）。在同一个表达式中使用几个不同的运算符时乘除法的计算先于加减法，包含在圆括号中的运算最先进行。

| 运 算 | 结 果 |
|---|---|
| 1+1 | 2 |
| 4-7 | -3 |
| 2*3 | 6 |
| 4+2*3 | 10 |
| (4+2)*3 | 18 |

Pascal有两种数值类型：整型和实型。整型数值是一些整数（…-2，-1，0,1,2…），实型数值可以包含小数（1.5，3.1415，-2.7）。几乎所有的计算机语言都存在这两种数值类型，其原因是由于数在计算机内存贮的方式。对于两种数值类型，Pascal有两种不同的除法运算符，除号/用于实型，短语DIV用于整型。由DIV产生的结果是被“截断”的，也就是说结果中不保留任何小数部分。

| 运 算 | 结 果 |
|---|---|
| 9/6 | 1.5 |
| 9DIV6 | 1 |
| 13/4 | 3.25 |
| 13DIV4 | 3 |
| 12.5/2 | 6.25 |

为了扩大整型数的用途，Pascal提供了MOD运算。MOD代表模数（modulus），表示两数相除得其余数。

| 运 算 | 结 果 |
|---|---|
| 9MOD6 | 3 6除9，商1余3 |
| 6MOD2 | 0 2除6，整除 |
| 7MOD2 | 1 2除7，商3余1 |

输出语句中的数值表达式。

WRITELN语句可以用来输出数值表达式的结果，把表达式放在圆括号中即可实现，但

不必放在引号内。语句

```
WRITELN ('7') ;
```

或

```
WRITELN (7) ;
```

将都印出数字7，但二者存在着重要的区别。头一个语句印出的数字作为一个字母串，而第二个语句印出的是一个表达式的结果。为了使区别更清楚，请看两个语句

```
WRITELN ('7*2') ;
```

和

```
WRITELN (7*2) ;
```

第一个语句将印出字符串7 * 2，而第二个语句将印出数字14，代表圆括号中表达式的结果。我们可以把这两种输出结合在一个语句中

```
WRITELN ('7*2=', 7*2) ;
```

执行这个语句将输出

```
7*2=14
```

用这样的方法可以把算式和结果都打印出，供用户查看。试着执行程序2.6，以便熟悉怎样输出数值结果。

**程序2.6**

```
program sample2_6;
{ auther: joe programmer      written:  feb 6, 1982
  this is a program to demonstrate numeric output }
BEGIN
  writeln ('7+2 = ', 7+2 );
  writeln ('5*6 = ', 5*6 );
  writeln ('15 drivided by 5 is ', 15 DIV 5);
END.
```

这种写出正文来标识数值结果的输出技术很有用。但是应该指出，引号内所写的内容并不影响计算结果。可以有这样的语句

```
WRITELN ('1+1=', 1+2) ;
```

它可以照样执行。计算机并不知道它印出的是什么，只能分辨合法或非法的Pascal语法，而不管输出是否合乎逻辑或者是否正确。

DOS和p-系统的MOD运算

现在我们碰到有关计算机运算问题。尽管整数的概念是大家所熟悉的，而且MOD运算又是基本的整数运算，但是在DOS和p-系统中的MOD运算却是不同的。当I的值为负时，DOS和p-系统中表达式I MOD J所产生的结果不一样。p-系统MOD运算的结果总是正的，而在I为负值时DOS的MOD运算的结果也是负的（或者是零）。

| 表达式 | | 结果 | |
|---|---|---|---|
| I MOD J | | DOS | p-系统 |
| + | + | + | + |
| + | − | + | + |
| − | + | − | + |
| − | − | − | + |

举例来说，－3 MOD 2，在p-系统中结果为＋1，而在DOS中结果为－1。这种区别

看起来好象无关紧要，但是当试图在两个操作系统之间进行程序转换时这种区别就不能小看了。

**练　习**

1. 解下列表达式

a. 3+1*2　　f. 20 MOD 5

b. 3+ (1*2)　　g. 20 MOD 6

c. (3+1) *2　　h. (4+5) *3* (5DIV2)

d. 20 DIV 5　　i. 3+2−7*15/10

e. 20 DIV 6

2. 给程序2.6增加一行，输出4乘7的积被5DIV的结果。

3. 写一个程序：取出数2和3，并把它们相加、3减去2、2减去3、3乘以2、用两种除法 做 3 除 以2。（这是全部的六种不同的操作）。

**解　答**

1. a. 5　　d. 4　　g. 2

b. 5　　e. 3　　h. 54

c. 8　　f. 0　　i. −55

```
2. program solution;
   { program to illustrate output of string and expressions }
   BEGIN
     WRITELN( '7+2 =', 7+2 );
     WRITELN( '5*6 =', 5*6 );
     WRITELN( '15 DIV 5 = ', 15 DIV 5 );
     WRITELN( '(4*7) DIV 5 = ', (4*7) DIV 5 );
   END.
3. program solution;
   { program to illustrate different operations }
   BEGIN
     WRITELN(2+3);
     WRITELN(3-2);
     WRITELN(2-3);
     WRITELN(3*2);
     WRITELN(3/2);
     WRITELN(3 DIV 2);
   END.
```

## 第四节　变　　量

使用WRITELN输出表达式的结果得不到十分有趣的程序。为了以有效方式进行表达式的运算，需要有一种方法来保持一条语句的值给后面的语句。这种方法就是使用变量。变量象个容器，它可把一个值存贮在里面，后来需要时又可使用这个值。我们以前处理数的方式，在WRITELN语句中使用数值好似把一匙花生酱倒在柜台上，因为我们无容器存放。我们需要的是一只花生酱罐，对计算机而言，是一个变量，这样我们就可以把某些算法的结果贮存起来，以备以后使用，而不是只把它印出。

为了能够使用一个变量须要对它加以说明，“说明”一词是关于一种语句的术语，这种语句告诉计算机程序将如何操作。在这里，这种说明语句标识一个名称，我们希望用这一名称来保持一个特定的数据类型。说明语句出现在程序标题之后BEGIN之前。VAR这个词表明将对变量进行说明。说明语句由两部分组成：变量名和它的类型。变量名是用户指定的，一