中学研究性课程和竞赛辅导教材

计算机程序设计

吴伟国　王建德　编著

内 容 提 要

本书是一本中学研究性课程和竞赛的辅导教材，其作者长期从事数学和计算机竞赛辅导，该书是他们根据多年指导学生参赛的实践经验编写而成的。书中通过“Pascal 语言”、“数据结构”和“算法分析”三个篇章将程序设计的知识融成一体，详细介绍了 Pascal 语言的语法规则和 Turbo Pascal 集成环境的操作技能，为数据结构和算法的实现提供语言支持；由浅入深地介绍了线性表、非线性表的知识和应用，为各种抽象运算定义了数据类型；详尽地介绍了一些常用的算法和解题策略，剖析了编程的基本思想和技术。这些内容均涵盖了中学生程序设计竞赛的大纲要求。

序

随着计算机技术的飞速发展，计算机应用日益普及，尤其是近年来 Internet 应用的普及，以计算机技术为核心的信息技术正在改变人们的思考方式和获取知识的途径。世界各国紧紧抓住这一机遇，重新调整人才的培养模式，使学生从传统的知识习得型向能力发展型过渡，提高捕捉、组织和处理信息的能力，掌握用整体、系统的观念处理复杂问题的方法。应该说，信息时代将基础教育推上了信息化的平台，并为其发展开辟了前所未有的理论空间和实践空间。

软件是计算机的“灵魂”，而程序设计是软件开发的基础。

何谓程序设计？程序设计包括两方面的内容，一是定义数据结构，为程序中用到的每一个变量确定其类型和结构；另一为算法设计，确定解决问题的方法或过程。概括地说，程序设计＝数据结构＋算法。

程序设计一般分两步：第一步是宏观设计，定义数据模型级上的运算步骤。在这一步中不需要明确变量的数据结构，运算带有抽象性质，不含具体细节。第二步是微观设计，为每一个变量定义数据结构，为每一个抽象运算编写程序。微观设计是宏观设计的具体实现，它依赖于宏观设计中定义的那些抽象运算。同样只有通过微观设计选择合适的语句形式和数据类型，才能最终实现宏观设计中定义的抽象运算并确定其效率。而在这两步中都同时包含着定义数据结构和设计算法两个方面的内容。

程序设计与应用软件的使用不同，它要求编程者以某种程序设计语言为媒介，通过构造算法去解决由现实生活中抽象出来的各种问题，而这些问题往往不是一般应用软件所能解决的。如果说计算机应用是“人脑延伸”的话，程序设计就是实现这一延伸的一种基本手段。

要设计出好的程序，编程者除了掌握一门计算机编程语言之外，还应具备以下一些知识和技能：

1. 良好的数学基础和算法知识，能够对问题域客观存在的事物及其所要解决的问题产生正确的认识和理解，包括弄清事物属性、行为及其彼此之间的关系；

2. 娴熟的编程技术，能够把对问题及其解法的认识用编程语言正确地表述出来，最终产生一个能够在计算机上执行的程序；

3. 相应的实践能力和创造能力，能够独立思考、提出质疑，拓延思路、洞悉规律，创造性地运用知识于不同问题情景。

所以可以说，程序设计在某种程度上反映了学生的抽象思维和逻辑等方面的综合能力。每年的国际国内奥林匹克信息学竞赛都将其作为考核内容，竞赛获奖的高中毕业生成了各大学优先录取的对象。

许多重点中学开办了奥林匹克信息学竞赛辅导班，一些中学还计划开办信息学方面的研究性课程。在这里，我向中学计算机教师和有兴趣参与计算机竞赛的中学生推荐《计算机程序设计》一书，这本书可以作为中学研究性课程或竞赛培训的教材。

这本书将数据结构、算法和 Pascal 程序设计语言融合成一本教材，体现了在程序设计过程中数据结构、算法和程序设计语言之间密不可分的关系。在“Pascal 语言”一篇中，作者详细介绍了 Pascal 语言的语法规则和 Turbo Pascal 集成环境的操作方法，为数据结构和算法的实现提供语言支持；在“数据结构”一篇中，作者由浅入深地介绍了线性表、非线性表的知识和应用；在“算法分析”一篇中，作者详尽地介绍了常用的一些算法和解题策略，介绍了编程的思想和技术。这些内容基本涵盖了程序设计和信息学竞赛的基本知识。

本书在系统性、入门性和实用性上颇有特色，理论描述正确，始终围绕编程实践，针对中学生的知识基础和学习特点，积极创设相关知识的问题情景，讲过程、讲思路、讲方法。书中的例子作者都经过了仔细的推敲，具有典型性。

本书的两位作者长期从事中学奥林匹克数学竞赛和信息学竞赛的辅导工作，编写出版了多本关于奥林匹克信息竞赛试题解析的著作，培养的学生在国际奥林匹克信息学竞赛中获十几块奖牌，可谓硕果累累。这本书是在他们多年教案的基础上梳理修改而成的。我曾经问他们：“为什么会想到编写程序设计的启蒙教材？”他们回答：“为了使更多的中学生理解和接受程序设计。”一句话真切地道出了这两位作者爱生敬业的精神。

软件人才的紧缺是一个全球性的问题。要使我国信息产业能持续高速发展，需要大批的软件开发人员。2001 年开始，教育部将在全国建立 30 所软件应用技术学院，每年培养 1.5万名软件方面的硕士生。随着中学“程序设计”研究性课程的开设和竞赛培训活动的普及，程序设计能够从大学的课堂走到中学的课堂，会有越来越多的学生在中学阶段就参与程序设计的学习和实践，为他们今后从事进一步的学习奠定一个良好的基础，同时推动计算机教育的普及程度和水平。

中国计算机学会常务理事　上海市计算机学会理事长
教育部高等学校计算机科学与技术教学指导委员会副主任
复旦大学首席教授　博士生导师
施伯乐

2002.1.1

前　言

我们两位作者长期从事数学和计算机竞赛的辅导工作，曾编写过十几本国际和全国奥林匹克信息学竞赛的辅导书籍。但近年来一直在反思一个问题：国际和全国奥林匹克信息学竞赛属于全国中学生最高学术层次的竞赛，每年每个省市仅派出三四人参加全国赛，全国仅选拔四人参加国际赛。有关国际和全国奥林匹克信息学竞赛方面的书籍尽管受到了参与培训的学生欢迎，甚至成为一些大学程序设计课程的参考读物，但是“阳春白雪，和者为寡”，读者自然寥寥。而全国奥林匹克信息学竞赛分区联赛是经国家教育部批准、每一位中学生可自愿报名的学科竞赛，同国际和全国奥林匹克信息学竞赛一样，获奖的高中生亦可获得直升大学或高考加分的资格。这项优惠政策每年吸引了几十万中学生“跃跃欲试”。问题是，目前程序设计尚未正式列入中学课程，是一块尚未开垦的“处女地”。在信息化、数字化高速发展的时代，程序设计的重要性日益凸显。大学理科类专业的课程大都涉及程序设计知识（例如 C++、数据结构、算法分析等）；在中学数学教育中，离散数的组合分析开始涉入教材，2001 年上海数学高考试卷甚至出现了一道计算机流程图的试题。面对这样一个时代背景，面对这么大一个“嗷嗷待哺”的学习群体，如果能够为全国奥林匹克信息学竞赛分区联赛的辅导培训和重点中学的研究性课程写一本程序设计方面的教材，其影响和意义恐怕要比写国际和全国奥林匹克信息学竞赛的辅导书籍更广泛、更深远。正是在这种动机的驱使下，我们着手编写这本教材。

问题是，这本书究竟应该写成什么模样。按照我们的意愿，“系统性、入门性和实用性”应该是本书体现的三大特点。

系统性：现在书店里也有一些全国奥林匹克信息学竞赛分区联赛的辅导书籍，但每个知识点自成一本，需要读五六本书才能达到分区联赛的大纲要求。而本书通过“Pascal 语言”、“数据结构”和“算法分析”三个篇章将分区联赛的知识结构融成一体。在“Pascal 语言”一篇中，我们详细介绍了 Pascal 语言的语法规则和 Turbo Pascal 集成环境的操作技能，为数据结构和算法的实现提供了语言支持；在“数据结构”一篇中，由浅入深地介绍了线性表、非线性表的知识和应用，为各种抽象运算定义了数据类型；在“算法分析”一篇中，详尽地介绍了一些常用的算法和解题策略，剖析了编程的基本思想和技术。这些内容均涵盖了分区联赛的大纲要求。

入门性：始终围绕编程实践，积极创设相关知识的问题情景，讲过程、讲思路、讲方法。理论描述务必准确，力避“误导”；文字尽可能通俗流畅，尽量把学生从过分专业化的语法规则、编程技术拉回到客观世界，拉回到人们的日常思维方式中来，实现返璞归真。对于所有的程序示例，不仅其正确性经过了上机验证，而且讲究时空效率，所有命令标识符简化为小写（Turbo Pascal 集成环境界面通过不同颜色标注命令标识符），程序短小精悍，结构错落有致，尽可能体现“严谨、简洁、统一、和谐、清晰和可靠”的风格。

实用性：全书列举的示例都是经过精选的，具有典型意义。分区联赛初赛涉及的知识放

到第一篇和第二篇讲解 在第三篇中,按照算法归类解析了 1996—2001 年分区联赛复赛的全部试题。对于后两篇中的试题解析,我们并未为读者提供直接上机运行的源代码,而是采用比较贴近自然语言的类 Pascal 来描述算法的基本思想和步骤的。每一章节后附有相关例题,这些例题始终与学生‘最近发展区’和分区联赛的难度要求相适应。这些做法既为学生的学习实施了科学实用的知识导向,又为学生上机实践、自主探究留下了空间。

我们提出的课程目标是

- 提高程序设计能力。100 句以内的源程序能自如地编写,300 句以上较复杂的程序能独立地设计和测试。
- 娴熟基本的数据结构和算法,了解编程原理和技术所涉及的课程。
- 初步能以正确的方法和风格处理程序问题。

课时安排一般为 120 课时:

- Pascal 语言　　40 课时
- 数据结构　　30 课时
- 算法分析　　50 课时

在编写本教材过程中,曾经在上海市计算机竞赛学校,山东省、浙江省和宁波、绍兴、温州、上海杨浦区等地区的竞赛辅导班,上海复旦附中理科班,控江中学研究性课程试讲过,以上目标基本达到。譬如,在 2001 年上海市获分区联赛一等奖的三十几位学生中,三分之二以上的学生系统学习过教材,其中有四名满分的学生(上海满分的学生共有五名)。应当承认,学生为此付出的代价是比较大的,课内课外的学时比超过 1∶2。一些学生之所以能成为编程好手,是因为他们在正规的、正确的思想指导下勤读程序范例,常上机实践,多做难题和大题。当他们感到有用和有兴趣时,他们愿意付出时间和精力。

目前为止,已有不少学校对我们的想法积极呼应。复旦大学计算机系的领导对中学阶段开设程序设计课程表示支持;上海华师大二附中、复旦附中、控江中学不仅准备在新学年将《程序设计》课程列入理科班、研究性课程、课外活动的教学,而且还计划在全校计算机课程中挤出 15～20 课时学习第一、二章,让全体学生了解程序设计的 ABC。

最后要说的是,在本书的编写过程中,全国著名的计算机专家施伯乐教授给予了热忱支持,并拨冗作序。复旦大学出版社孙未未博士认真审改了全书,并提出了不少真知灼见。在此谨表感谢!

作　者

2002 年 1 月

目　　录

第一篇　Pascal 语 言

第二篇　数 据 结 构

第三篇 算 法 设 计

第一篇　Pascal 语言

第一章　绪　　论

§1.1　Turbo Pascal 概述

Pascal 是 1971 年瑞士苏黎士工学院 N. Wirth 教授专为结构化程序设计而设计的一种高级语言，历时 30 载，经久不衰，已成为世界上广泛流行的程序设计语言之一。Pascal 之所以受到欢迎，是因为它有以下一些特点。

1. Pascal 是结构化的语言

Pascal 语言全面清晰地体现结构化思想。不仅提供了直接实现“顺序结构”、“选择结构”和“重复结构”的语句，而且还可以定义子程序（“函数”和“过程”）。在编程时可以完全不使用转向语句，使程序的执行顺序与行文顺序保持一致，便于理解。

2. Pascal 的数据类型丰富

Pascal 语言不仅提供了整数类型（简称整型）、实数类型（简称实型）、字符类型和布尔类型等标准数据类型，而且还允许用户按照语法规则自定义枚举类型、子界类型、数组类型、集合类型（简称集合）、记录类型（简称记录）、指针类型（简称指针）和文件类型（简称文件），便于数值计算和非数值信息处理。

3. Pascal 语言可以实现模块的独立化要求

Pascal 语言允许在函数和过程内部定义局部变量，允许在主程序和子程序之间传递参数，使得每一个子程序模块都能反映一个相对独立的性质，模块之间的联系简单清晰，便于修改。

由于 Pascal 语言精确地表达了程序设计的基本概念，结构性好，表达能力强，因此成为很多学校程序设计课程的教学语言，以及计算机会考和奥林匹克信息学竞赛的规定语言。本书将详尽地讲解 Pascal 的语法规则和 Turbo Pascal 的集成环境，所有示例都采用 Pascal 语言或类 Pascal 语言编写。

Turbo Pascal 是编译型的程序设计语言，编译程序将源文件（源程序文件）转换成目标文件（二进制代码文件），其过程如图 1.1.1 所示。

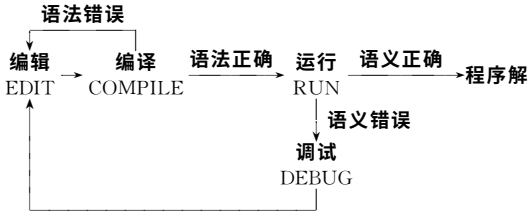

图 1.1.1

所有编辑、编译、运行和调试过程集中在 Turbo Pascal 集成环境(IDE)中进行。IDE 界面如图 1.1.2 所示。

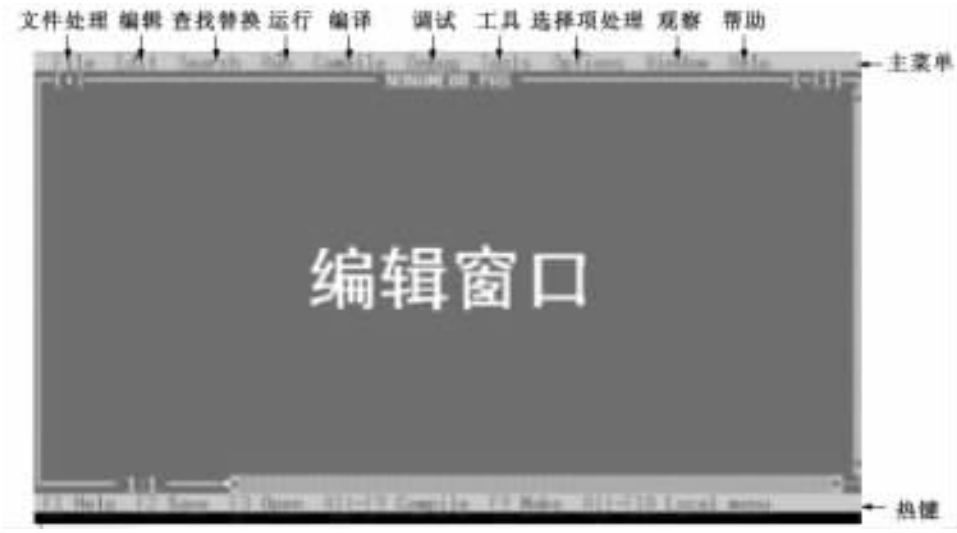

图 1.1.2

屏幕中央为编辑窗口,用户在该窗口输入和修改源程序。屏幕的底行为"热键—命令"表,在集成环境的任何地方,按动热键即可执行对应的命令。屏幕顶行为主菜单。主菜单有十项,它们是:

File	文件处理子菜单
Edit	编辑子菜单
Search	查找、替换子菜单
Run	运行子菜单
Compile	编译子菜单
Debug	调试处理子菜单
Tools	工具子菜单
Option	选择项处理子菜单
Window	观察子菜单
Help	帮助子菜单

在集成环境的任何地方,只要按热键 F10 就可以激活主菜单。在主菜单上可以用←,→选择相应的子菜单。选中的菜单会以彩色或高亮度标出,这时按 Enter 键后便可进入。更简便的方法是同时按 Alt 键和所选子菜单的首字母,进入相应的子菜单。在子菜单状态下按 Esc 键,可使子菜单消失,返回主菜单。

子菜单是一个下拉式菜单,内含多个可供选择的命令。例如 File 子菜单共有十项:

Open	F3	打开一个文件
New		建立一个新文件
Save	F2	存盘
Save as		另存为

Save all		全部存盘
Print		打印
Get into		改变目录
DOS shell		进入 DOS 环境
Exit	Alt+X	退出 Turbo Pascal 集成环境

我们可按↑,↓选择相应的命令,亦可通过键入命令的首字母进行选择。F3, F2 和 Alt+X 为热键,含热键的命令可在集成环境的任何地方使用。例如,无论何时按 F2 键,都可以将当前编辑窗口中的文件存盘。不含热键的命令只能在 File 子菜单中使用。

在主菜单中,最有吸引力的恐怕莫过于帮助子菜单(Help)了。该子菜单列出了所有帮助信息的详细目录,是一个完善的帮助系统。这个帮助系统不仅为初学者自学提供了丰富的资料,而且是成熟的编程者备查的"字典"。为了方便大家随时查询,系统还专门设置了热键 F1,在集成环境的任何地方,只要按该键就可进入帮助子菜单。

Help 子菜单共有五项,由上而下依次是:

Contents		帮助内容
Index	Shift+F1	帮助索引
Topic search	Ctrl+F1	主题查找
Previous topic	Alt+F1	上一次的帮助主题
Help on line		在线帮助

在编辑程序文本时,如果想了解某个 Pascal 结构(函数、变量、类型等)的帮助信息,可把光标移至该处,按 Ctrl+F1 键(同时按下 Ctrl 和 F1 键),系统将立刻回答操作者的查询。

我们在这里强调帮助子菜单,是因为它的完整性和实用性是任何一本 Pascal 的教科书或工具书所不能及的。它涵盖了 Pascal 的语法细节,并提供了大量实用的库程序说明,用户上机编程时可以"信手拈来"。经常学习帮助子菜单中的内容,不仅可以提高对系统的熟练程度,而且可以提高英文的阅读水平,更重要的是可以获得比知识本身更有价值的东西——自主探索、学会学习。

§1.2 实例的演示

在这一章节中,我们想通过两个实例的演示,使读者初步了解 Pascal 程序的基本结构,掌握 Turbo Pascal 系统编辑、存储、编译、调试和运行用户程序的一般过程。

【例 1.2.1】五环图

调用 Turbo Pascal 的作图程序单元,画出如图 1.2.1 所示的五环图。

输入

r(圆的半径);$x0$, $y1$(左上角圆的圆心坐标);d(x 坐标的位移);$y2$(下方两个圆的圆心的行坐标)。

输出

五环图形状,直至按回车键为止。

注:Turbo Pascal 系统中有一个作图程序单元 graph,里面包含了许多库程序。例如"Circle(x,y,r)"在屏幕上画一个以(x,y)为圆心、以 r 为半径的圆。编程者可以利用这些

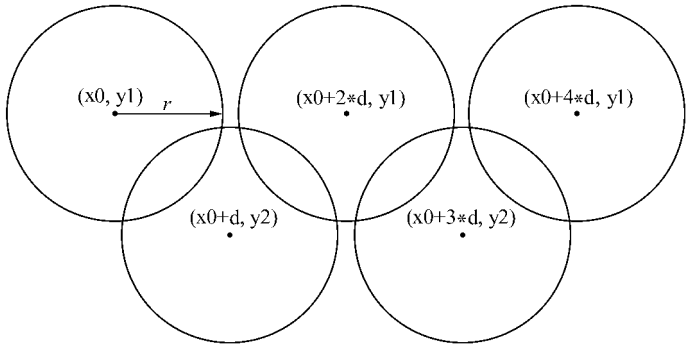

图 1.2.1

库程序作图。程序开始时，设置调用作图程序单元的命令“uses graph”，并在调用库程序前通过“DetectGraph(gd,gm)”设置驱动程序和图模式编号，“InitGraph(gd, gm, 'BGI')”初始化图形系统。在调用了单元 graph 中所需要的库程序后，必须通过“CloseGraph”命令关闭图形系统。

我们在 Turbo Pascal 所在的子目录下键入“BP”，进入 Turbo Pascal 集成环境，然后按 Alt+E 键，选择编辑子菜单。此时，光标位于编辑窗口的左上角，等待键入程序：

```
program circles;
uses graph;                          {调用 Turbo Pascal 系统内名为 graph 的程序单元}
var
gd,gm :integer;                      {gd——图设备的驱动程序编号;gm——图模式编号}
r, x0, d, y1, y2 :integer;           {r——半径;(x0, y1)——左上角圆的圆心坐标;
                                     d——x 坐标的位移;y2——下方两个圆的圆心的 y 坐标}
begin
DetectGraph(gd,gm);                  {设置驱动程序和图模式编号}
InitGraph(gd,gm,'BGI');              {初始化图形系统}
Readln(r, x0, y1, d, y2);            {读入五环图的信息}
Circle(x0,y1,r);Circle(x0+2*d, y1, r);Circle(x0+4*d,y1,r);
                                     {画上一行的三个圆}
Circle(x0+d, y2, r);Circle(x0+3*d, y2, r);   {画下一行的两个圆}
CloseGraph;                          {关闭图形系统}
end. {main}
```

1. 程序的一般结构

由上例可以看出程序的一般结构：

program 程序名；

说明部分

begin

语句部分

end.

(1) 除需要通过“ uses 单元名 ”调用子程序库的情况外,任何程序都以“ program 程序名 ”开始。例如上述程序的首部为“ program circles ”,circles 为编程者自定义的程序名。“ program 程序名 ”只是从形式上给程序定义一个名称,与程序的执行无甚影响,因此可以省略。

(2) 源程序中用到的常量、变量、转移位置、函数和过程、用户自定义的数据类型,必须在说明部分中定义,程序不允许使用未经定义的标识符。每一种说明都以特定的保留字开始(如表 1.2.1 所示)。

表 1.2.1

说明内容	变量说明	常量说明	类型说明	行号说明	函数说明	过程说明
保 留 字	var	const	type	label	function	proceduer

例 1.2.1 中关于五环图的上述程序仅有变量说明(对程序所使用的变量及其类型加以说明),说明程序使用变量 gd,gm 和 r,x0,d,y1,y2,它们都为整数型。

(3) 语句部分以“ begin ”开始,以“ end. ”结束。语句间用“ ;”隔开。

(4) 程序的书写方式灵活,即允许一行写一条语句;也允许一行写几条语句;甚至允许空行。允许在程序的任何位置插入注释(注释用 { } 括起来),以便阅读。注释对程序的执行不起任何作用。

2. 编辑技巧

我们可以通过下述方法提高编辑效率。

(1) 使用多重并行窗口。Turbo Pascal 允许在屏幕上设置多个编辑窗口,同时显示多个源程序。打开的窗口数最多可达 100。但在同一时刻,只能对一个窗口中的文件进行编辑,该窗口称为“ 当前窗口 ”。

① 打开窗口,输入文件:

(i) 打开一个空的文本窗口(File|New):当前窗口中的文件取名为 nonami. pas,i 为打开窗口的顺序($0 \leqslant i \leqslant 99$)。

(ii) 在窗口中打开文件(File|Open—F3):选择了子命令 File | Open—F3 后,屏幕中央显示如下信息(图 1.2.2)。

② 控制窗口管理:

(i) 选择窗口:

Windows|File——同时显示所有窗口,可通过 Alt+窗口编号选择窗口;

Windows|List · · ——显示打开窗口和关闭窗口的文件列表;

Windows|Next(F6)——选择后一个窗口为当前窗口;

Windows|Preuious——选择前一个窗口为当前窗口。

(ii) 设置窗口显示方式:

Windows|Size|Move(Ctrl+F5)——通过 Shift+方向键(← ↑ → ↓)扩大或缩小当前窗口;

Zoom(F5)——满屏显示当前窗口与恢复原窗口大小位置的切换;

Windows|Cascade——串行显示所有窗口;

Windows|Close——关闭当前窗口。

窗口。将光标移至目标位置,选择 Edit|Paster(Shift+Ins)。此时,记录板中的文本块被粘贴。

3. 文件存盘

编辑后的文件在运行前必须存盘,以免出现意外情况(例如,死机或陷入死循环)而丢失信息。存盘的方式有:

(1) 若以原名存盘,则选择 File|Save(F2)。选择该命令后,原文件被覆盖。若当前窗口用 New 创建,则需输入文件名;

(2) 若待存储的文件需要以新文件名命名,则选择 File|Save as。选择该命令后,原文件变成副本保留在内存;

(3) 若需要将所有被打开窗口中的文件存盘,则选择 File|Save all;

文件存盘后,可以退出 Turbo Pascal 集成环境,在 DOS 或 Windows 下检查文件是否已存入指定位置。退出的方式有两种:若需要释放 Turbo Pascal 系统所占的内存,则选择 File|Exit(Alt+X);若需要保留 Turbo Pascal 系统所占的内存,则选择 File|DOS shell。在 DOS 状态下键入"Exit"命令,便可返回 Turbo Pascal 集成环境。

4. 程序的编译和运行

文件存盘后,便可以放心编译或运行程序了,不必担心出现"前功尽弃"的事。选择 Compile|Compile 命令,编译当前编辑的源文件。若源文件有语法错误,则返回编辑窗口,光标指向错误行,顶行显示错误类型;若源文件语法正确,则产生可执行代码文件(即类型为.exe的文件),该文件可以脱离 Turbo Pascal 集成环境,直接在计算机中运行;如果不需要产生可执行代码文件的话,则可避开编译过程,直接在集成环境中运行源程序。

我们采取后一种运行方式,按 Ctrl+F9 键运行五环图程序。当输入五环图的信息(左上角圆的圆心坐标(x0, y1),x 坐标的位移 d,下方两个圆的圆心行坐标 y2)后会发现程序返回了 Turbo Pascal 集成环境。是不是程序没有被执行呢?当然不是。原因在于计算机的速度太快,显示在 DOS 屏幕上的运行结果瞬间被 Turbo Pascal 集成环境覆盖了。那么,怎样才能观察到运行结果呢?有两种途径:

(1) 若希望在编辑窗口下方开辟 DOS 窗口,则选择 Windows|output;

(2) 若希望整个屏幕作为 DOS 窗口,则选择 Windows|uses screen (Alt+F5)。在 DOS 状态下按 Esc 或 Alt+F5 键,可返回 Turbo Pascal 的集成环境。

除了使用 Turbo Pascal 的集成环境下的命令外,我们还可以通过程序本身来解决这个问题:在关闭图形系统前(即"CloseGraph;"前)增加一条语句:

"readln;"

即当五环图显示在 DOS 屏幕时,程序并未运行结束,在等待回车。直至回车后才返回 Turbo Pascal 的集成环境。我们今后编写程序时,不妨可以借鉴这个经验。

在运行五环图程序时,五环图的大小、位置和美观程度取决于输入数据(r, x0, y1, d, y2)。建议输入

"66 180 200 72 272"

看一看屏幕上的五环图是不是达到了最佳效果。

5. 程序的调试

程序设计工作是一项容易出差错的工作。一般来说,程序的错误有两种:

(1) 语法错。这类错误可由编译程序查出。避免这类错误发生的办法只有一个——认

真学习和切实掌握语言的功能和规范。

(2) 语义错,即源程序经编译通过之后,在运行中出现的各种错误。通常有以下几种:

① 书写错误的计算公式;

② 忽视很少出现的极端情况,如溢出、越界、死锁、死循环等;

③ 程序结构或逻辑出错。

因此,要根据运行结果查验程序是否按预定的要求工作。如果程序被证明语义有错,就转入调试,确定程序中可疑错误的确切性质,改正错误。调试程序的方法一般有两种:其一是静态查错,即不用执行程序,仅根据程序和框图对求解过程的描述来发现错误;其二是动态查错,即在程序执行的过程中寻找错误的确切位置。

Turbo Pascal 的集成环境中含大量的动态调试功能,这些功能主要集中在主菜单 Run, Debug, Option 中,这些主菜单如图 1.2.4 所示。

图 1.2.4

下面举一个实例,介绍如何利用集成环境中的调试功能进行程序测试和排错的一些经验。

【例 1.2.2】计算数列

计算数列

$$s = 1 + 3 + 5 + 7 \cdots$$

的前 $n(1 \leqslant n \leqslant 500)$ 项的和。

输入

n。

输出

数和 s。

题解

由数列的形式可以看出,第 1 项数为 1,相邻两个数等差为 2。显然,第 n 项数为 2 * n −1。

输入下述程序(注意:程序左端的数码是为解释调试过程设置的,不用输入):

```
1. var
2. i,n :integer;                {——当前项;n——项数。i 和 n 的数据类型为整数}
3. s :longint;                                  {和,数据类型为长整型}
4. begin
```

```
5. s:=0;                                    {初始时和为 0}
6. i:=1;                                    {第一项为 1}
7. readln(n);                               {读项数}
8. while i<2*n-1 do                         {若当前项小于等于上限,则循环}
9. begin
10. s:=s+i;                                 {当前项计入和}
11. i:=i+2;                                 {计算下一项}
12. end;{while}
13. writeln('total=',s);                    {输出和}
14. readln;                                 {空等回车}
15. end. {main}
```

注意 在 Pascal 程序中,“:=”表示赋值,“=”表示大小相等的关系,与人们的习惯写法不同。

由于程序无语法错误,编译一次通过。输入 5(回车),屏幕上显示“total=16”。显然,数列的和应为 25。这个答案表明程序有语义错误。为此,我们利用集成环境中的调试功能进行动态查错。选择 Debug | Watch 命令,在屏幕下方开辟一个观察窗口,该窗口能在程序运行过程中显示被观察变量的当前值。通过 Debug | Add watch(Ctrl+F7)设置被观察变量 i, s,然后按 Ctrl+F2 键(Run|Program reset),从头开始执行。在各种执行方式中,选择一次执行一行的方式调试程序。连续按 F7 键(Run|Trace into),白色亮条随当前被执行的行上下移动,各个观察变量的当前值显示在观察窗口。当白色亮条移至第 7 行,系统便进入了 DOS 屏幕,等待用户输入数据。在输入数据“5”后系统又返回集成环境,白色亮条在 8～12 行间上下移动,表明这几行程序是一个循环结构,其中 9～12 行是循环体,可以看成逻辑上独立的一条语句。循环的条件是 i<2*n-1,每执行一次循环,当前项 i 记入数列和 s。但问题是,循环了四次后程序便退出循环(由第 8 行跳至第 13 行),输出前四项的和 16,这说明最后一项并未记入数和。我们将循环条件改为 i≤2*n-1(即第 8 行改为“while i<=2*n-1 do”),程序便可以给出正确的解了。

通过上述调试过程,我们对 while 语句和 begin...end 语句的涵义已经有了一种直觉。由此可见,集成环境中的调试功能不仅可以用来进行动态查错,而且还可以作为学习语言的工具。借助于对程序的跟踪调试,可以使我们了解语法的形成过程,有利于建立一种生动活泼的编程思想。

图 1.2.5 介绍了使用 Turbo Pascal 集成环境调试功能的一般过程。读者在今后学习语言或调试查错时可以借鉴。

(1) 调试初始化。

① 设置检查状态:

(i) 检查数值越界的情况。查出越界后,将产生出错信息。置 Option|Compile 菜单中的 Range checking 项为 On。

(ii) 过程或函数执行前检查堆栈可用内存情况。若堆栈没有足够空间,会产生运行错误。置 Option|Compile 菜单中的 Stack checking 项为 On。

最简便和周全的办法是按 Ctrl+O+O 键,编辑窗口的第 1 行显示所有编译开关状态