話說調試程序

葛芝宾 编著

清华大学出版社

話說調程序

葛芝宾 编著

清华大学出版社
北　京

内 容 简 介

全书介绍对程序错误进行分析的思路、排查的方法，结合编译原理透彻地解释出错现象；介绍链接错误及其产生原因，以及运行时错误及其产生原因，并采用相应的程序演示运行时错误的调试方法。本书还介绍调试程序逻辑错误常用的策略和技术。最后介绍程序调试测试与用例设计、科学设计测试用例等知识，其中涉及相关的软件测试技术。本书以较流行的 Turbo C2.0 集成环境作为编程环境，但是所述程序调试方法并不局限于 C 语言或 Turbo C2.0 环境。本书各章都引用 C++程序示例，可帮助初学者拓展知识。

本书凝聚了作者多年编程教学和软件开发的经验，可作为高校计算机程序设计专业的学习资料，也可作为计算机编程从业人员的参考资料。

图书在版编目（CIP）数据

话说程序调试 / 葛芝宾编著. —北京：清华大学出版社，2011.6
ISBN 978-7-302-24937-5

Ⅰ. ①话… Ⅱ. ①葛… Ⅲ. ①调试程序 Ⅳ. ①TP311.52

中国版本图书馆 CIP 数据核字（2011）第 040535 号

责任编辑：夏兆彦
责任校对：徐俊伟
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　　总　　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×230　**印　张**：11.25　**字　数**：281 千字
版　　次：2011 年 6 月第 1 版　　**印　　次**：2011 年 6 月第1 次印刷
印　　数：1～4000
定　　价：29.00 元

产品编号：040993 01

前言

FOREWORD

随着计算机技术和应用的不断普及，越来越多的人开始接触计算机程序设计，而我国数千所大学计算机专业的学生更是需要学习编程。编程包括两项任务，即用某种语言编写程序和将所编写的程序调试正确。然而目前的形势是，作为计算机程序设计重要组成的程序调试并没有得到应有的重视，缺少相应的教学和训练环节；尽管当前有关计算机方面的书籍非常之多，但是关于程序调试方面的书籍几乎没有。回忆当年学习编程之初，面对自己编写的程序中那些错误，不知道从何下手，也不知道如何找出并改正，犹如在“黑暗”中摸索，熬过了多少个痛苦、趟过了多少道沟壑，通过大量实践的积累和体会，才走出了“雾里看花”。同样地，在我从事大学计算机专业教学的几十年中，也深切地感受到学生们为学习程序调试所经历的苦恼和迷茫。

编程是一项创造性的劳动，每当编写成功一个新的应用程序，那感觉就像一个新生命呱呱坠地，喜悦之情不亚于面对婴儿的母亲；但是，如果编写的程序无法调试正确，那么，这个婴儿就还不能出生。在学习编程初期，常常被程序调试这道门槛所困。在我国越来越多的大学将培养目标转向培养应用型技术人才的今天，作为计算机程序设计重要组成的“程序调试”技能的教学或培养应该得到相应的重视。基于此，我尝试把自己多年来从事计算机教学和软件开发的心得体会进行梳理、归纳、总结提高，写一本“话说程序调试”的书，帮助那些需要学习计算机程序设计的人们，使其能有一个起步的基础，掌握一些程序调试的策略、方法和技术，拉他们跨过这道门槛。然后，通过他们的积累、提高，逐步地成长为软件开发人员。虽然程序调试更依赖于程序员的功底、悟性和技巧，也就是人们常说的“艺术性”，但如果在学习编程之初能有老师讲讲程序调试的基本知识、基本思路、方法和技术，或者能有这方面的指导性资料，或许就会少一些起初的困惑、迷茫和无奈，就像赛跑一样，也就能有个好的起跑。“艺术”与其他技术也有相通的地方，即都需要基础知识、实践和掌握规律，程序员的功底也是通过学习、实践和提高得来的。

浅显的道理是，程序的错误具有不可预见性和不确定性，这使得本书不可能穷尽可能遇到的各种程序错误，因此，本书侧重于通过典型案例的分析来说明一些分析问题的思路，总结一些查找错误的规律，展示一些方法和技术，这就是本书编写的宗旨：“授人以渔”。

就我的学识和经验来说，还不足以写好这本书，但我希望本书能对学习计算机程序设计的学生和爱好者有所帮助。尽管我尽我所能地来写此书，但是，由于个人水平的局限和各种原因，可能书中存在各种各样的缺陷和不足。如果能起到抛砖引玉的作用，使更多的专家关注这一方面而写出更好的书籍是我所期盼的。

本书给出了一些示例程序，有些可能较长，但对于查找程序中的错误，可能更有“实战”氛围；而且有些示例程序也很有趣，对初学编程者更具有参考价值。因此，为了程序的完整性，我尽可能对程序不作有损的删减。最后，对于读者的意见和批评，我必定虚心接受并对不足之处加以改正。

感谢我的同事王创伟和唐仕喜，本书的例 2.13 和例 4.17 调试实例是他们分别提供的。我还要感谢我的学生们，本书的许多示例是他们的实验程序。

作者

2010.12

于盐城师范学院

目 录 CONTENTS

第 1 章 概述 …… 1
1.1 关于程序调试 …… 1
1.2 程序中会有哪些错误 …… 1
1.3 程序调试环境 …… 2
1.4 一个程序示例 …… 2
1.5 本书的主要内容 …… 8

第 2 章 编译错误的分析与排查 …… 10
2.1 语法错误及其排查 …… 11
2.1.1 相关语法分析知识 …… 11
2.1.2 位置准确，原因不准确 …… 12
2.1.3 位置准确，原因准确 …… 14
2.1.4 位置不准确，原因准确 …… 20
2.1.5 原因不准确，位置不准确 …… 22
2.1.6 语法错误调试综合示例 …… 25
2.1.7 关于外部引用的语法错误简述 …… 29
2.2 语义错误及其排查 …… 29

第 3 章 链接和运行时错误的调试 …… 44
3.1 链接错误及排查 …… 44
3.2 运行时错误及排查 …… 45

第 4 章 逻辑错误的调试策略和技术 …… 58
4.1 调试的准备 …… 58
4.2 流程观察分析法 …… 62
4.3 分离法 …… 73
4.4 屏蔽法 …… 79
4.5 数据透视法 …… 89
4.6 增式加入法 …… 94
4.7 试探法 …… 107
4.8 回溯法 …… 114
4.9 猜错法 …… 123
4.10 事件驱动环境下的程序调试简述 …… 128

第 5 章 调试测试与测试用例设计 …… 136
5.1 调试测试与测试用例 …… 136
5.2 测试用例设计简介 …… 137
5.2.1 复合谓词覆盖测试 …… 137
5.2.2 路径覆盖测试 …… 139
5.2.3 边界值分析 …… 140
5.2.4 等价类划分 …… 141
5.3 测试用例设计示例 …… 143

附录 1 Turbo C2.0 编译出错信息 …… 156

附录 2 示例索引 …… 170

参考文献 …… 172

第1章 概述

1.1 关于程序调试

什么是程序调试？检测、定位、排除程序中错误的过程，称为程序调试。

从心理上说，几乎每个人都会认为自己辛辛苦苦所编写的，甚至经过若干次修改的程序是不存在错误的，然而事实却不是如此。事实是，我们所编写的每个程序都可能存在错误。这些错误及错误的多少通常与对所用程序设计语言的熟悉程度，对算法的理解和实现细节的处理，以及对具体开发环境（语言）默认的、特殊的处理规则是否了解有着密切关系，或者是由于粗心所导致的。因此，我们所编写的程序在调试之前都不能保证不存在错误，而应该认为错误是不可避免的存在。

程序调试的目的就是找出程序中的错误并加以改正。而程序调试的具体内容就是通过对源程序的编译、链接、试运行（包括有目的的测试），将各种信息进行分析，发现和查找程序中的错误，从而改正错误，使得程序能正确地完成所设计的运算（功能）。

1.2 程序中会有哪些错误

程序中总是不可避免地存在错误，从程序所处的阶段看，这些错误一般有如下情形。

（1）编译错误。编译错误包括语法（含词法）错误和语义错误。它们是经常出现甚至是不可避免的。由于种种原因，我们可能写出不符合程序语言规则的语句，或遗忘一些必须具有的标识符（如变量等）声明、函数（过程）的向前引用声明等。语义错误包括类型检查发现的错误，如表达式中各运算分量的类型、形式参数与实际参数的类型是否相容等；以及一般的语义错误，如下标变量、域名的合法性检查，函数调用的合法性，形式参数与实际参数的个数是否一致，每个使用性标识符是否都有定义等。编译程序在对源程序进行编译处理的过程中，会发现这类错误并给出相应的错误报告信息。

（2）链接错误。现今的许多开发集成环境将源程序的编译和链接合并为一个快捷方式，从而使得语法错误和链接错误同时显示，常常使人误以为链接错误也属于语法错误，但链接错误有其不同的产生原因，如果将源程序的编译和链接分开进行，则可以将语法错误和链接错误区分开来，以采取相应的措施加以排除。链接错误多因标识符（如外部）的引用错误、标准库函数（类库中的类）名的拼写错误等产生，并由链接程序给出错误

报告信息。以上这两类错误也可以归并为编译链接阶段错误。

（3）运行时刻错误。这样的错误对应的语句在语法上没有任何问题，编译程序能够顺利地将其生成目标模块（.obj）文件，在链接时也没有链接错误而产生可执行（.exe）文件，但在执行文件运行时会出现问题。例如，除数为 0，空指针的使用，数组下标越界，资源重复释放，等等将导致异常，使得程序停止运行。

（4）逻辑错误。这是最麻烦的一类错误，其产生的主要原因大多是由于程序编写者对算法的理解或在实现细节上出了偏差，从而在将某个成熟的算法转换为程序设计语言的程序时造成了不等价，使得所编写的程序运行后不能得到正确（或预期）的结果。

对于一个成熟的程序员来说，其程序调试的主要精力多花费在后两类错误上。

1.3 程序调试环境

不同程序设计语言的开发环境不尽相同，目前大多采用集成（IDE）开发环境。随着计算机科学技术的发展，有些开发环境采用了许多先进技术来帮助程序员在输入程序时提高速度和检查错误。例如：VB 代码编辑器提供了“属性/方法列表”、“常数列表”、“快速信息”、“参数信息”和“插入关键字”等智能化的功能和语法检查功能，当发现输入语句的语法错误时，就将该语句标记为设定的颜色（如红色）。Delphi 则采用 Code Insight（代码洞察）技术，其 Code Template Expert、Code Completion Expert 和 Code Parameter Expert 不仅能大大提高键入代码的速度，而且能有效避免出现某些输入错误。同时，大多数开发环境还提供功能强大的程序调试工具，如设置断点、单步执行、跟踪等工具，这些调试工具主要帮助程序员查找并发现程序中的逻辑错误和运行时错误。即使早期的 Turbo C2.0 集成环境也向用户提供了诸如 Run、Debug、Break、Watch 菜单中的调试命令。

1.4 一个程序示例

例 1.1 需求描述：校园导游程序。用无向网表示你所在学校的校园景点平面图，图中顶点表示主要景点，存放景点的编号、名称、简介等信息；图中的边表示景点间的通路，存放路径长度等信息。

要求实现以下功能。

（1）能查询各景点的相关信息。

（2）能查询图中任意两个景点间的最短路径。

程序：

```
/*ch1_ex1.c */
#include<stdio.h>
#define MAX_POINT 4
#define INFINITY 32767
#define Error 1
#define OK 0
typedef struct
{
 int bianhao;  /*景点编号*/
 char pointname[20];
 char information[100];
}point;  /*顶点结构体*/
typedef struct
{
 int length;
}bian; /*邻接矩阵元素类型*/
typedef struct
{
 point p[MAX_POINT];
 bian juzhen[MAX_POINT][MAX_POINT];
 int vexnum,arcnum;
}graph;  /*图的类型*/
int LocateVex_M(graph *G,char v[])
{  /*定位顶点函数*/
int j=Error,k;
  for(k=0;k<G->vexnum;k++)
  if(strcmp(G->p[k],v)==0)
    {
      j=k;
      break;
    }
  return (j);
}
int createUDN(graph *G)
{  /*创建无向网函数*/
 int i,j,k,weight;char v1[20],v2[20];
 printf("please input vexnum:");
 scanf("%d",&G->vexnum);
 printf("please input arcnum:");
 scanf("%d",&G->arcnum);
 for(i=0;i<G->vexnum;i++)
   for(j=0;j<G->vexnum;j++)
```

```
        G->juzhen[i][j]. length =INFINITY ;
    for(i=0;i<G->vexnum;i++)
     {
      printf("\nplease input pointname:");
      scanf("%s",&G->p[i].pointname);
      printf("\nplease input pointbianhao:");
      scanf("%d",&G->p[i].bianhao);
      printf("\nplease input point's information:");
      scanf("%s",&G->p[i].information);
    }
    for(k=0;k<G->arcnum;k++)
    {printf("please input two point and weight(v1,v2,3):");
     scanf("%s,%s,%d",&v1,&v2,&weight);
     i=LocateVex_M(G,v1);
     j=LocateVex_M(G,v2);
     G->juzhen[i][j] =weight;/*填入邻接矩阵元素*/
    }
    printf("success!!");
    return(OK);
   }
   int menu()
   {  /*操作菜单显示*/
    int choice;
    char s;
    puts("   Operating Choose Menu   ");
    puts("1.create a graph");
    puts("2.search every point's information");
    puts("3.search two point's shortest distance ");
    puts("4.search two point's every paths");
    puts("5.EXit.    ");
    printf("Please choose: ");
    do{
       s=getchar();
       choice=(int)s-48;
      }while(choice<0||choice>5);
       return choice;
   }
   void shortdistance()
   {}/*以下3个函数暂未编写-笔者注*/
   void seachinform()
   {}
   void seachpaths()
```

```
{}
void main()
{
graph *G;
 int flag=1;
 int ch;
 int i;
 while(flag)
 {
   ch=menu();
   switch(ch)
   {
     case 1:
      {
       createUDN(G);
       break;
      }
    case 2:
      {seachinform();
       break;
      }
   case 3:
      {
       shortdistance();
       break;
      }
  case 4:
      {
      seachpaths();
       break;
      }
 case 5:
      {
       flag=0;break;
      }
default:printf("Error!Input again!\n");
   }
 }
}
```

运行、调试情况：

这是学生实验课编写的程序，用于求解《数据结构》（见参考文献[4]）教材中“校

园导游程序”实习题，学生使用 TC（Turbo C2.0）集成环境下的 Ctrl+F9 键运行该程序，出现了多种不同的错误。图 1.1 和图 1.2 示出两次运行的不正常现象（这些不正常现象不一定重现）。

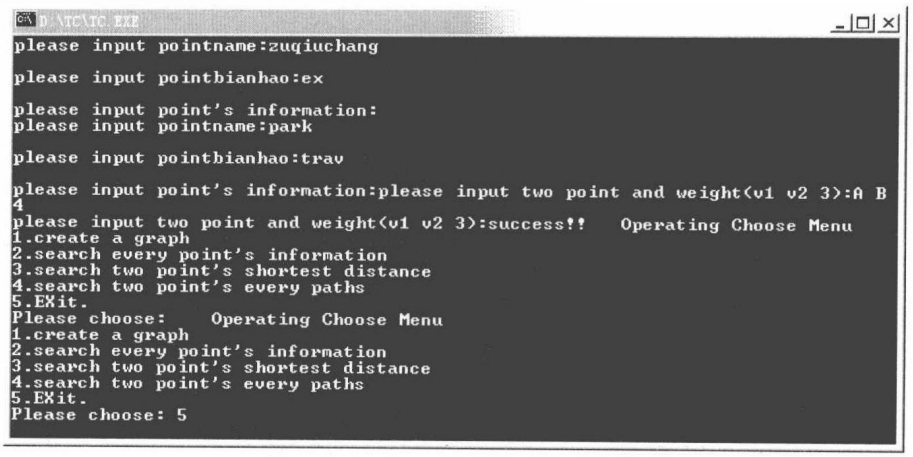

图 1.1　程序运行的第一种情形

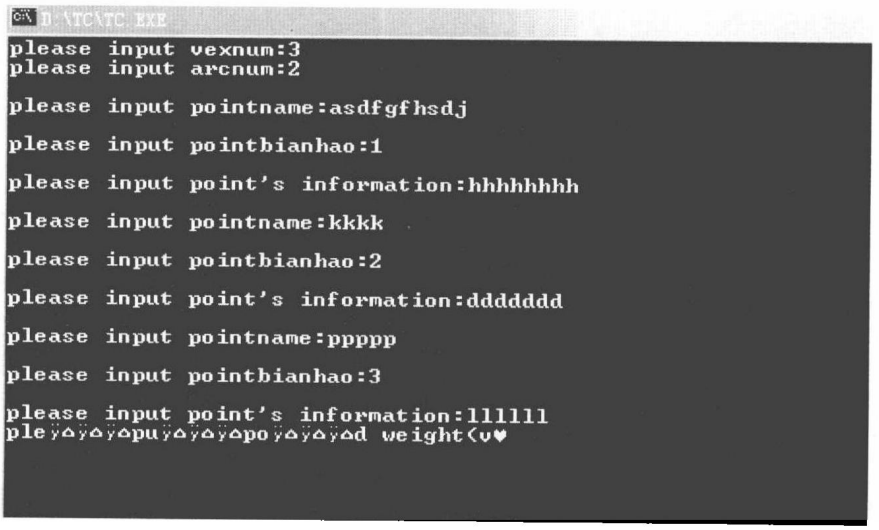

图 1.2　程序运行的第二种情形

图 1.1 的第 6 行表明，不能再继续正常输入景点信息及其后的一些信息，而图 1.2 的最后一行表明，应该输出的提示信息“please input two point and weight(v1，v2，3):”变成了一些乱码。这些都表明，程序存在着错误（尽管有人在程序出现类似的不正常现

象时宁愿怀疑机器出了毛病)。于是，将程序的编译和运行分开进行，果然发现，在程序编译后显示了如下的错误（Warning）警告信息：

```
possible use of 'G' before definition in function main
```

根据这个警告信息查看源程序，发现程序将 G 定义为 graph 类型的指针变量，却没有为其分配存储空间。尽管在许多情况下，在 C 程序中使用未分配空间的指针、程序也能运行，但这是不正确的。应当明白，在本例中，尽管根据定义为变量 G 分配了存储单元，但该指针变量所指向的对象（结构体）却没有存储指针所指数据的内存单元，由于指针 G 未指向确定的结构体单元，所以程序运行时，指针变量 G 的当前值是不可知的(指向未知的单元)，从而导致程序运行的不正常。改正的方法是在 main 函数的 while 循环前增加一条语句：

```
G=( graph *)malloc(sizeof(graph));
```

重新编译后错误消失。但是，再细看程序，发现还存在许多问题。首先，如下结构体的定义是不必要的（原则是：没有必要定义只包含一个简单成员的结构体)：

```
typedef struct
{
 int length;
}bian;
```

其次，下列函数中存在错误：

```
int LocateVex_M(graph *G,char v[])
{int j=Error,k;
 for(k=0;k<G->vexnum;k++)
  if(strcmp(G->p[k],v)==0)
    {
      j=k;
      break;
    }
  return (j);
}
```

例如，其语句 if(strcmp(G->p[k],v)==0)中 strcmp()函数的实参是两个完全不同的类型，实际上应该是 G->p[k].pointname 与字符串变量 v 进行比较，等等（该程序将在第 4 章中进一步调试)。

这个示例程序除去学生已改正的编译错误，几乎是一个“原生态”程序，之所以开篇举出这个例子，是因为觉得许多学生会犯类似的错误。通过这个示例，想提醒初学编

程者以下几点。

（1）养成好的编程习惯，在编写程序前应该准备好源程序，而不是坐到计算机前才边想边输入，否则会使得程序有“东拼西凑”、前后矛盾的乱象。至少需要有明确的算法和数据结构。

（2）定义了指针类型的变量后，一定不要忘记将其指向确定的数据单元，即或者用malloc 函数为其申请空间，或者将某个同类型变量的地址赋给指针变量，否则会留下严重的隐患。

（3）不要为了省事而对一个尚未经过调试的程序直接采用“运行”快捷命令；不要将编译和运行两个步骤合并起来执行，否则就会像本例一样，将会因看不到某些重要的编译警告信息而不能发现并纠正错误，使编译错误后推为程序的运行错误；应该将程序先编译、链接、最后再运行，这样可以将程序的不同错误分别在对应的不同阶段显示，便于将它们区别分析和调试处理。

1.5 本书的主要内容

本书是根据作者多年的教学和软件开发的经验体会，经过加工整理、归纳总结而写成的。为了解释程序调试中的某些现象，探讨某些错误产生原因及其查找规律，笔者还结合叙述了编译原理和实现技术中的相关知识，期望读者能对所述问题有个清晰的认识，以便能举一反三。同样，在调试链接错误、运行时错误以及逻辑错误的示例中，也结合相关的专业知识，对错误进行分析、对调试示例进行清晰的讲解。同时，在不偏离主题的前提下，对一些问题还进行一些拓展探讨。但是，由于各个程序员的各个应用程序所产生的错误是不同的，作者不可能穷尽所有具体程序的具体错误，所以本书仅对一些常见的错误以及相应的调试策略、方法和技巧作出归纳、总结。对于各种程序开发环境的调试工具的使用，本书也不做介绍，因为可以从相关的资料中获取。

作为规范的程序设计语言，Pascal 曾是数十年间世界各大学的程序设计教学语言。但近年来，很多学校均以 C 语言作为程序设计的教学语言。尽管目前流行的软件开发设计语言多为面向对象程序设计语言（OOL），如 VC++（C++）、C#、Java、Delphi、VB 等，但 C 语言不仅是计算机专业学生所熟悉的程序设计语言，而且还是 VC++、C#和 Java 的基础，它们在语法成分上一致，而 C 语言更为计算机专业学生所熟悉。虽然面向过程和面向对象程序设计的主要差别是程序设计范型不同，但在具体代码实现和调试时，过程式的 C 程序设计仍然是基础。因此，本书主要以面向过程的 C 语言程序作为调试的示例程序，并且以较流行的 Turbo C2.0 集成环境作为编程环境，但这并不意味着本书所述程序调试方法局限于 C 语言或 Turbo C2.0 环境。本书第 2、3、4 章也各引用一个 C++示

例程序，可作为初学者的拓展或参考。

本书第 2 章对常见的编译错误进行排查，按编译出错信息所告知的出错原因、错误定位的不同情形，介绍对错误进行分析的思路、排查的方法，还结合编译原理，透彻地解释有关出错现象；第 3 章叙述链接错误、链接错误的产生，运行时错误、运行时错误的产生，并采用相应的程序调试方法对运行时错误的调试进行演示；第 4 章介绍调试程序逻辑错误常用的策略和技术，结合示例、分析思路清楚、目的明确、说理透彻；第 5 章简单介绍程序调试测试及测试用例设计，科学设计测试用例对有效发现程序的逻辑错误是重要的，其中涉及一些相关的软件测试技术。

本书在内容组织和示例的安排上也颇费心思，一方面缺乏可供参照的书籍；另一方面示例程序的选择和安排也较困难。因为讲解程序调试与讲解 C 语言程序设计毕竟有所不同，如果示例程序只有简单的几行，会因为太容易找错误而缺少实战感，也不能较好地说明一些思路和技术。因此，第 4 章以后的多数示例程序相对较长，算法也较复杂，对于初学 C 语言的读者可能有些难度，请予以理解！此外，本书给出的示例程序尽量注意了结构化程序规范，而不像一些教材上的例题程序只有一个 main 函数。作者希望这些示例程序能给读者一个示范，从而自觉地遵循结构化程序设计规范，设计出风格良好的程序，而结构良好的程序会给程序调试带来许多的好处。

书后的一些附录供读者在排查编译错误时查阅。

第 2 章　编译错误的分析与排查

编译程序在对源程序进行编译的时候，通过对源程序的语法检查和语义检查，总能主动报告所发现的语法错误、语义错误；而在利用语言集成环境对目标模块进行链接的时候，也能报告有关的链接错误。由于集成环境提供了快捷生成执行模块并运行的命令，初学编程者大多不愿意将编译、链接和运行分步进行。但是，链接工作毕竟不是编译程序所为，因此，笔者宁愿将编译错误和链接错误区分开来。因此，本章介绍的编译错误排查，就包括程序的语法错误和语义错误，统称为编译错误。

编译错误是由所用语言的编译器对用户源程序进行扫描检查、发现并主动报告的错误。图 2.1 所示是 Turbo C2.0 集成环境编译源程序后的画面。有两类不同性质的编译错误：一般性错误（Errors）和警告性错误（Warnings）。一般性错误将导致程序无法成功编译（即不能生成目标文件），而警告性错误则不影响目标文件的生成。由于编译程序时能发现和报告相应的错误，所以程序员可以借助这些信息查找和改正源程序中的错误。

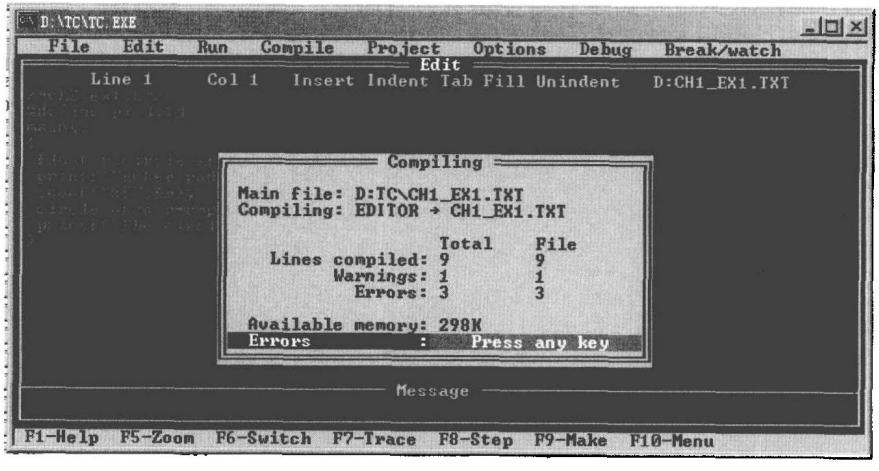

图 2.1　Turbo C2.0 编译界面

对于编译错误，各个程序员针对不同应用采用不同设计语言所编写的源程序可能会出现各种各样的错误。谭浩强教授在《C 程序设计》（文献[1]）第 359 页至 371 页列举了 29 种初学者易犯的语法（编译）错误，对避免初学者犯一些 C 程序的语法错误提供了很好的指导。然而，我们无法穷尽所有程序员遇到的错误。因此，笔者从寻找编译错

误的排查规律的角度，示例解释编译错误的查找和改正方法。这里不妨假定程序员（不论是“新手”还是“高手”）对所采用的程序设计语言的语法是基本了解和熟悉的，也就是说，不会发生太“离谱”的错误。

2.1 语法错误及其排查

出现语法错误的原因大多是对语言语法掌握的熟练程度不够，或者是程序员输入源程序时的疏漏。一些教材上说：“这类语法错误比较容易发现和纠正”，这能算得上是真理，因为语法错误可以依赖编译程序报告，如果源程序中存在语法错误，编译程序是决不放过的；其次，与程序的逻辑错误相比，语法错误的纠正也较容易，因为编译程序报告的出错信息中均指示了出错的原因和位置（错误所在程序行）。语法错误的排除关键是能否准确定位到错误，然后再根据出错信息和语言的语法规则加以纠正即可。

程序中的语法错误在编译程序、并同时进行词法分析（该步骤一般不明显）、语法分析阶段均可被发现，同时给出相应的错误报告信息（一般称为出错信息）。如前所述，编译程序报告的语法错误信息有两部分内容：错误位置和错误原因。看起来，根据编译程序报告的错误信息似乎就能很容易地找到错误，并根据错误原因容易地改正错误。然而，事实并非如此，初学编程的人往往不能很容易地找出错误和改正错误。其主要原因是：有些出错信息对错误原因的分析与实际致错的原因并不相符；或者出错信息所报告的错误位置离真正出错的程序行有一定距离。这是初学者排除程序语法错误的困难所在。为什么编译程序报告的错误信息会出现这种情况？这得了解编译程序对源程序的语法分析的相关知识。尽管这些专业课的理论知识读起来有点苦涩，但是可以帮助我们理解上述的一些现象。当然，笔者也尽可能将这些理论知识说得通俗些，且请读者耐心地看下去。

2.1.1 相关语法分析知识

编译程序所处理的原始数据是源程序，程序设计语言所编写的源程序可以看做一个（文本）字符串，编译程序对源程序的编译通常经历词法分析、语法分析、语义分析、中间代码生成和中间代码优化、目标代码生成这些步骤。词法分析的任务是从源程序中识别（分离）出一个个单词（又称为符号）、并将单词转换为内部表示（TOKEN 码）。所谓单词，是独立的最小语义单位，如源程序中的保留字、标识符（如变量名、函数名等）、常量、运算符等。在识别单词时，分界符（界限符）是帮助分离单词的重要符号。语法分析阶段则是根据语言的语法规则对单词的内部表示序列进行语法检查和分析。检查各个语句是否符合语言的语法规则是语法检查的主要任务之一，语句是语法分析的基本单