21世纪计算机系列教材

CAD程序设计方法

CAD Chengxu Sheji Fangfa

郑忠俊 沈予洪 张兴亮 编著

上海交通大学出版社

21世纪计算机系列教材

CAD程序设计方法

郑忠俊　沈予洪　张兴亮　编著

上海交通大学出版社

内容提要

本书详细介绍了在微机上应用 AutoCAD 软件及其开发工具进行 CAD 程序设计的方法及开发技术。内容包括四篇：第一篇介绍了 Visual LISP 编程方法；第二篇介绍了 VBA 编程方法；第三篇介绍了 ObjectARX 编程方法；第四篇介绍了多平台交差编程方法。书中有大量的程序实例，是作者多年从事教学科研的总结，有较大的参考价值，读者完全可以放心地借鉴和使用。

本书主要面向大专院校学生、研究生以及从事 CAD 工作的各专业技术人员。

图书在版编目(CIP)数据

CAD 程序设计方法/郑忠俊，沈予洪，张兴亮编著. —上海：上海交通大学出版社，2004

(21 世纪计算机系列教材)

ISBN 7-313-03838-0

Ⅰ. C… Ⅱ. ①郑… ②沈… ③张… Ⅲ. 计算机辅助设计：程序设计-高等学校-教材 Ⅳ. TP391.72

中国版本图书馆 CIP 数据核字(2004)第 080262 号

CAD 程序设计方法

郑忠俊　沈予洪　张兴亮　编著

上海交通大学出版社出版发行

(上海市番禺路 877 号　邮政编码 200030)

电话：64071208　出版人：张天蔚

立信会计出版社常熟市印刷联营厂印刷　全国新华书店经销

开本：787mm×1092mm 1/16　印张：15.5　字数：382 千字

2004 年 8 月第 1 版　2004 年 8 月第 1 次印刷

印数：1～5 050

ISBN 7-313-03838-0/TP・600　定价：26.00 元

前　言

在微机上进行 CAD 程序设计是广大工程技术人员经常从事的工作，掌握 CAD 程序设计的基本方法，特别是最新的方法，是大家的迫切需要。有关 CAD 程序设计，这方面的专著也不少，但是较全面地介绍各种方法的实用技术，特别是最新的程序设计技术的书籍并不多。

AutoCAD 是目前国内外最受欢迎的微机 CAD 软件之一。该软件自 1982 年问世以来，版本不断升级，功能不断增强。AutoCAD 现已拥有世界上 18 种语言的相应版本，拥有数以千万计的用户群体，占据着 CAD 领域的主导地位，其图形数据文件格式已成为一种事实上的国际工业标准。AutoCAD 在 20 年左右的时间里获得如此巨大的发展，究其原因，除了软件本身具有强大的图形处理功能外，主要是它的开放型结构吸引了众多的 CAD 工作者，它提供的多种开发工具及网络设计功能为各行各业的 CAD 软件开发人员提供了功能极其强大的软件平台以及丰富多彩的开发工具，使用户可以开发出适合于各专业领域的 CAD 应用软件。

迄今为止，AutoCAD 提供的开发工具主要是这三种：VL(Visual LISP)、VBA(Visual Basic for Applications)及 Object ARX(AutoCAD Runtime extension)，本书就围绕这三种工具，研究在微机上进行 CAD 程序设计的基本方法。

本书主要包括四篇：第一篇介绍了 Visual LISP 编程方法；第二篇介绍了 VBA 编程方法；第三篇介绍了 Object ARX 编程方法；第四篇介绍了多平台交差编程方法。由于 CAD 内容十分丰富，不可能用一本书全面介绍它们的所有功能，所以，编写时作者力求用自己多年从事 CAD 教学和开发的成熟程序实例，让读者尽快掌握在微机上进行 CAD 程序设计的基本技术。

本书内容总结了作者多年的教学和科研经验，书中实例的所有程序均经作者多次升级处理并在 AutoCAD 最新版本下调试通过，读者完全可以放心地借鉴和使用。

本书由四川大学郑忠俊、沈予洪及成都电子机械高等专科学校张兴亮编著。

本书在编写过程中得到四川大学 CAD 中心、工程设计中心同行们的大力协助，在此深表谢意！

由于作者水平有限，书中不当之处，恳请读者及同行批评指正。

编　者

目　　录

第一篇　Visual LISP 编程方法

第二篇　VBA 编程方法

第三篇　ObjectARX 编程方法

第四篇 多平台交差编程方法

第一篇 Visual LISP 编程方法

1 Visual LISP 编程环境

1.1 AutoLISP 程序的结构及特点

LISP(List Processing Language)是一种计算机的表处理语言，是迄今为止人工智能领域广泛应用的一种程序设计语言，而 AutoLISP 语言是一种嵌入在 AutoCAD 内部的 LISP 编程语言，它是 LISP 语言和 AutoCAD 有机结合的产物。AutoLISP 针对 AutoCAD 增加了许多新的功能，比如增加了图形数据库处理的有关函数，利用 AutoLISP 可直接调用 AutoCAD 的全部命令。因此，它既具有一般高级语言的基本结构和功能，又具有一般高级语言所没有的强大的图形处理功能，故它是当今计算机辅助设计及自动绘图最广泛采用的语言，也是开发智能 CAD 系统的得力工具。

AutoLISP 程序具有如下结构特点：

(1) AutoLISP 程序是一个由许多子表组成的一个大表。所谓表，就是由圆括号括起来的，用空格分隔的若干元素形成的数据结构形式，如表(A B C D)，表中有四个元素，表的长度为 4。表的结构可层层嵌套，如(A (B C) (D))，该表有三个元素，即一个原子 A，两个子表(B C)和(D)。表是有序的，如(A B C)不同于(B C A)。若表中无元素，称为空表，可写作()。以下是一个求圆周长及面积的 LISP 程序：

```
(defun yuan ()
   (setq r (getreal  "请输入圆半径：  "))
   (setq L (* 2 pi r) S (* pi r r))
 )
```

该程序中我们定义了一个函数 yuan，几个子表的功能是实现让用户输入一个半径值后，自动计算圆的周长及面积。自定义函数的方法，本书后面章节将有详尽的论述。

(2) 表中两个元素之间至少有一个空格，多个空格的作用与一个空格的作用相同。

(3) 一段完整的 LISP 程序，左右括号数必须相等，即括号匹配的原则。

(4) AutoLISP 语言采用前缀表示法，即运算符放在操作数之前。例如：

```
(* 2 3)
(/ A B)
```

(5) 可在程序的任何地方对程序进行注释，以分号标志开始到行末的全部内容作为注释，AutoLISP 对注释内容不解释也不执行。

(6) 除字符串外，AutoLISP 程序中的字母可随意大小写。

(7) 程序中""号表示回车。

(8) AutoLISP 调用 AutoCAD 绘图命令的编程顺序与键入绘图过程对应。例如：

(command "pline" P1 "w" 0.4 "" P2 P3 "")

该程序实现画一条从 P1 点到 P2、P3 点，宽度为 0.4 的多义线。

1.2 Visual LISP 编程环境

Visual LISP(VLISP) 是 Autodesk 公司从 AutoCAD R14 起提供给用户的可视化的、功能强大的开发 LISP 程序的集成开发环境。前期版本的 AutoLISP 一般没有提供任何编辑器或编译器，用户大多选择一种自己熟悉的文本编辑器；而 Visual LISP 所提供的集成开发环境可以自行编辑、编译、修改及调试 AutoLISP 源程序，可以说是完全针对 AutoLISP 程序语法规则设计的专业编辑器。

1.2.1 启动 Visual LISP

Visual LISP(VLISP) 集成开发环境在单独的窗口中运行，必须启动 VLISP，才能在它的集成开发环境中工作。启动 AutoCAD 后，可采用以下任何一种方法进入 VLISP：

- command：VLISP 或者 VLIDE。
- 菜单→Tools→AutoLISP→Visual LISP Editor。

1.2.2 Visual LISP 界面

启动 VLISP 后出现如图 1.1 所示界面。

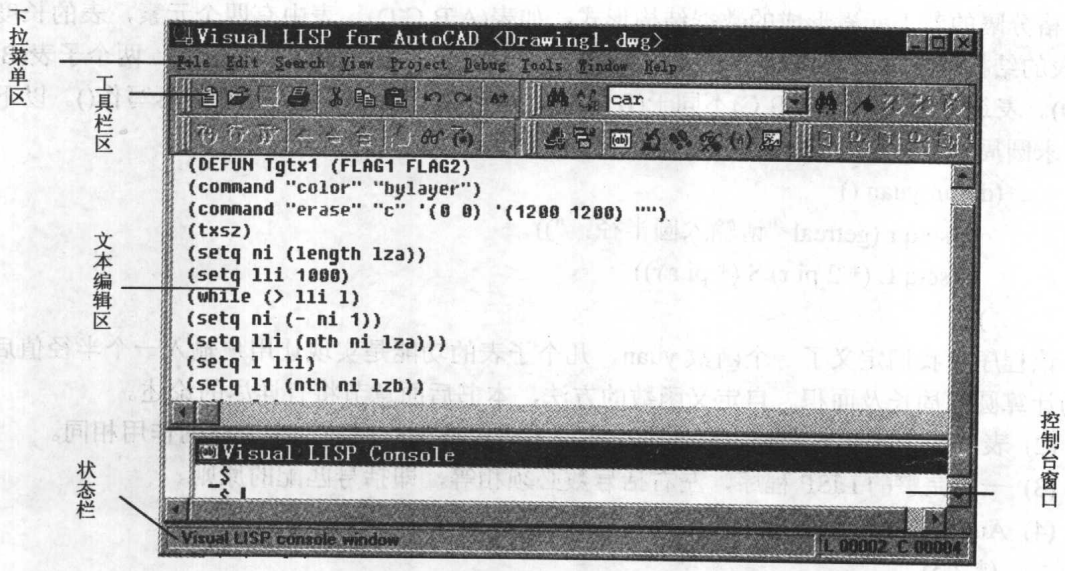


图 1.1 Visual LISP 界面

1.2.2.1 下拉菜单区

下拉菜单区集中了 VLISP 的主要功能，单击下拉菜单项，可以执行相应的命令。VLISP 的下拉菜单内容如下：

(1) File：

New File：编辑一个新文件。

Open File：打开用户指定的文件。

Reopen：VLISP 编译器自动记录曾经打开过的所有文件，用户可以在记录中查询并重新打开这些文件。

Save/Save AS：将当前编辑窗口中的文件存盘或更名存盘。

Save All：保存所有编辑窗口的文件。

Close：关闭当前编辑窗口。

Revert：重新打开当前编辑窗口中的文件，放弃对它的编辑和修改。

Close All：关闭除控制台窗口以外的所有窗口。

Print：打印当前窗口(编辑窗口或其他窗口)的文档。

Print Setup：打印设置。

Make Application：利用编译向导，创建一个 ARX 或者 VLX 的应用程序。

Load File：调用“Load Lisp File”对话框，加载用户指定的应用程序。

Exit：退出 VLISP

(2) Edit：

Undo：放弃上一次操作。

Redo：恢复上一次 Undo 放弃的操作。

Cut：剪切选中的内容到剪贴板。

Copy：复制选中的内容到剪贴板。

Paste：粘贴剪贴板中的内容到当前光标位置。

Delete：删除当前选中的内容。

Delete All：删除当前窗口中的所有内容。

Parentheses Matching：匹配括号。

Extra Commands：实用的编辑工具，如附加注释、块操作、字符大小写等操作命令。

(3) Search：

Find：调用“Find”对话框在用户指定范围(选中部分、当前文件、指定工程文件或者指定路径下指定类型的所有文件)内查找用户指定的字符串。

Replace：调用“Find”对话框在用户指定范围(选中部分或当前文件)内查找并替换指定的字符串。

Find/Replace Next：查找或替换下一个相匹配的字符串。

Bookmarks：在程序行切换或者清除书签。

Go To Line：移动光标到用户指定的程序行。

Go To Last Edited：将光标移动到当前窗口最后的编辑修改位置。

(4) View：

Inspect：打开一个检查(inspector)窗口。

Trace Stack：打开一个堆栈跟踪(Trace Stack)窗口。

Error Trace：为当前调试程序的最后一个堆栈错误打开一个检查(Inspector)窗口。

Symbol Service：为用户指定符号打开一个“Symbol Service”对话框。

Watch Window：打开“Watch”窗口。

Apropos Window：为用户指定符号打开“Apropos”窗口。

Breakpoint Window：调用“Breakpoints”对话框，列出编辑器中所有程序的断点。

Output Window：显示最后的“Output”窗口。

LISP Console：切换当前窗口为控制台(Console)窗口。

Toolbars：调用“Toolbars”对话框。

(5) Project：

New Project：创建一个用户指定名称的工程。

Open Project：打开用户指定的工程。

Close Project：关闭当前工程。

Project Properties：浏览和设置当前工程的属性。

Load Project FAS File：加载当前工程的 FAS 文件。

Load Project Source Files：加载当前工程的 LSP 源文件。

Build Project FAS：编译当前工程为一个 FAS 文件。

Rebuild Project FAS：重新编译当前工程为一个 FAS 文件。

(6) Debug：

Step into：单步执行命令。

Step over：对当前表求值光标移到表尾。

Step out：确定当前程序中断发生的位置，判断该位置所属函数，继续执行程序到该函数结束。

Continue：从中断点恢复程序执行，直到下一个断点或者程序结束。

Reset to Top Level：终止当前所有有效的 Break Loop，并把控制返回到 Top Loop。

Quit Current Level：终止当前 Break Loop，返回到其他 Break Loop 或者 Top Loop。

Add Watch：在 Watch 窗口增加变量。

Watch Last Evaluation：打开 Watch 窗口检查系统变量 LAST-VALUE 的结果。

Toggle Breakpoint：在当前位置设置，或取消断点的切换。

Clear All Breakpoints：清除所有断点。

Last Break Source：显示引起最后一个中断的中断源。

Trace Command：跟踪命令调用。

Stop once：立即停止切换开关。

Break on Error：出错时中断切换开关。

Animate：Animation 类型切换开关。

(7) Tools：

Load Selection：加载当前选中的程序代码段。

Load Text In Editor：加载当前编辑窗口中的程序。

Check Selection：检查当前选中的 AutoLISP 程序代码段的语法。

Check Text In Editor：检查当前编辑窗口中的 AutoLISP 程序代码段的语法。

Format AutoLISP In Selection：格式化当前选中的 AutoLISP 程序代码段。

Format AutoLISP In Editor：格式化当前编辑窗口中的 AutoLISP 程序代码。

Interface Tools：调试编辑窗口的 DCL 文件，预览用户定义的对话框。

Window Attributes：自定义窗口属性，如字体、颜色等。

Enviroment Options：设置 Visual LISP 开发环境选项，如 AutoLISP 程序格式、打印页面等等。

Save Settings：保存用户设置。

(8) Window：

Tile Horizontally：水平并列所有打开的窗口。

Tile Vertically：垂直并列所有打开的窗口。

Cascade：层叠所有打开的窗口。

Zoom：最大化当前窗口。

Organize：整理 Visual LISP 当前显示窗口。

Iconize All：将所有打开的 Visual LISP 的窗口最小化。

Arrange Icons：从 Visual LISP 界面的左下角重新排列所有图标。

Close Windows：关闭用户指定的窗口或者所有窗口。

Activate AutoCAD：切换到 AutoCAD 图形窗口。

(9) Help：为用户提供在线帮助和版本信息等。

1.2.2.2 工具栏区

Visual LISP 的工具栏包括“Standard”、“View”、“Search”、“Debug”、“Tools”等五个工具栏，如图 1.2 所示，这些工具栏也称为浮动工具栏，所谓“浮动”是指用户可以把工具栏拖拉到窗口的任意位置，用户也可以关闭它们。

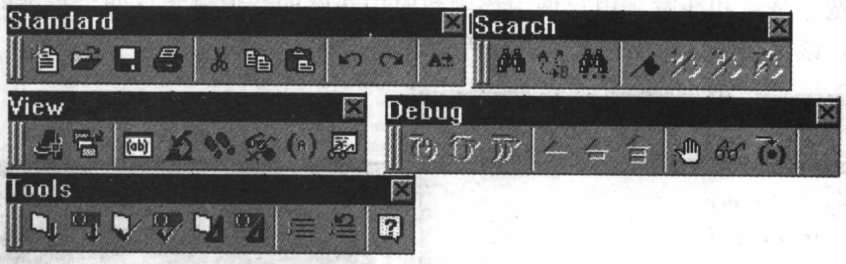


图 1.2 浮动工具栏

将鼠标的光标放在工具栏图标上，系统将在该位置显示简短的命令提示，用户可按需要选择执行相应的功能。

1.2.2.3 控制台窗口

控制台(Console)窗口相当于 AutoCAD 的命令行，用户可以在这里运行程序或者输入 AutoLISP 命令，并得到相应的返回值或者信息。与在 AutoCAD 命令行变量求值不同的是不

必借助于“！”专用求值符，只需在“-$”提示符后输入相应变量回车即可。

1.2.2.4 文本编辑区

用户可在文本编辑窗口中创建、打开、编辑、修改、保存、打印任意数目的 AutoLISP、DCL、C/C++等程序源文件。

1.2.2.5 状态栏

状态栏位于窗口的底部，并随时刷新，显示当前下拉菜单或者浮动工具栏等的简短提示。

1.3 在 Visual LISP 环境下编辑并运行程序

我们通过一个实例说明如何在 Visual LISP 编程环境下编辑并运行用户自己设计的一个程序，实现过程如下：

(1) 进入 AutoCAD。

(2) Command：VLISP 进入 VLISP 编程环境。

(3) 下拉菜单→File→New File 创建一个新文件。

(4) 在文本编辑区键入以下程序：

```
(defun box (x y p1 lw)
   (setq p2 (polar p1 0 x)
      p3 (list (+ (car p1) x) (+ (cadr p1) y))
      p4 (Polar p3 pi x)
   )
   (command "pline" p1 "w" lw "" p2 p3 p4 "c")
)
```

该程序的功能是定义一个画矩形的函数，此函数有四个形参：x、y、p1 及 lw，分别代表矩形的长、宽、基点和线宽，用户调用该函数时用实际的数值赋予这四个形参即可画出任意尺寸的矩形。该函数的编辑结果如图 1.3 所示。

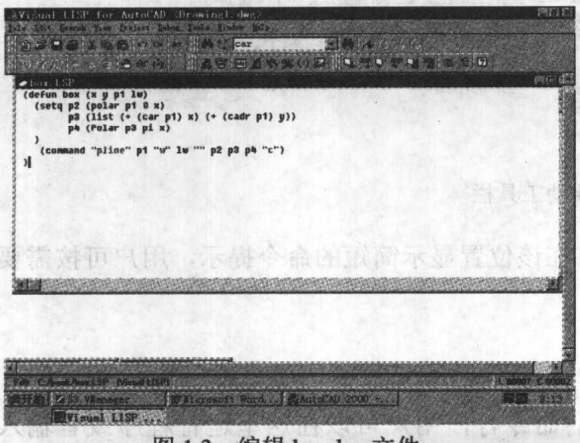


图 1.3 编辑 box.lsp 文件

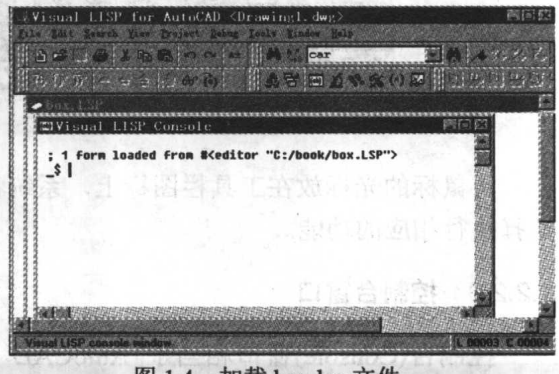


图 1.4 加载 box.lsp 文件

(5) 下拉菜单→File→Save，保存该文件，取名为：box.lsp。

(6) 下拉菜单→Tools→Load Text in Editor，加载当前编辑窗口中的 box.lsp 文件入内存，如果程序没有错误，控制台窗口将显示一条调入程序的结果信息，如图 1.4 所示。如果程序有语法错误，控制台窗口中将提示错误信息，用户须回到编辑窗口进行修改。

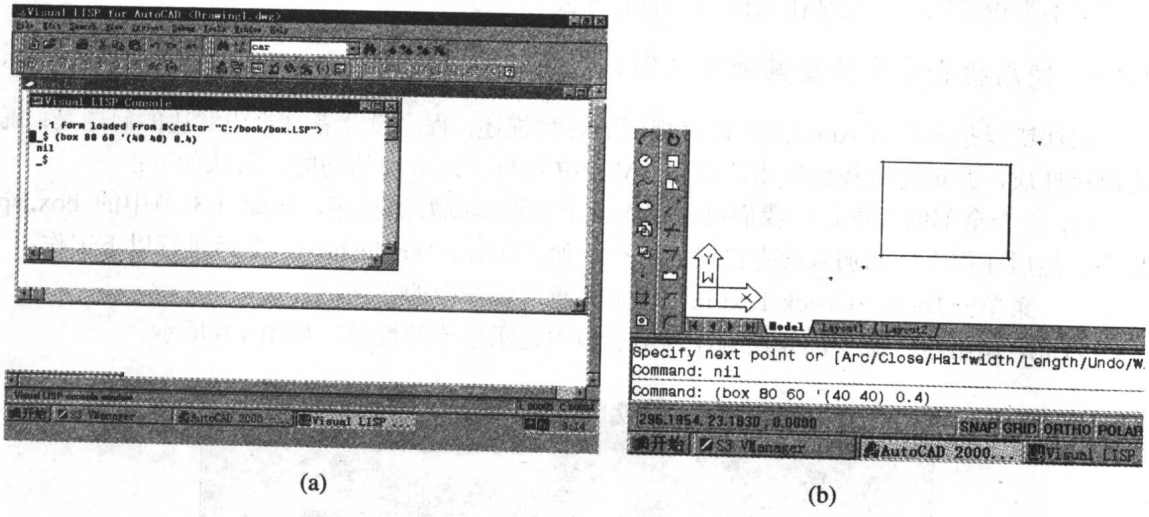
(7) 运行应用程序，可用以下两种方法中的任意一种实现：

① 在 VLISP 控制台窗口键入(box 80 60 '(40 40) 0.4)并回车，如图 1.5(a)所示。

在下拉菜单→Window→Activate AutoCAD，或在工具栏 View→可切换到 AutoCAD 图形窗口观察由程序 box 绘出的图形，如图 1.5(b)中所绘出的矩形所示。

② 在 AutoCAD 图形窗口中：Command：(box 80 60 ' (40 40) 0.4)并回车，结果如图 1.5(b)所示。

以上两种方法均绘出宽为 80，高为 60，基点在(40 40)，线粗为 0.4 的矩形。



(a)　　(b)

图 1.5　运行 box.lsp 程序

1.4　在 VLISP 编辑环境下查找 LISP 程序常见错误的方法

无论是初学者还是有经验的编程人员，在编写 LISP 程序时都可能出现这样或那样的错误，其中最常见的错误是：括号匹配错误、标准函数拼写错误、由于未留必要的空格引起的语法错误等。VLISP 编程环境提供了解决这些常见问题的方法。

1.4.1　检查括号匹配

AutoLISP 最常出现的语法错误是左右括号不配对，特别是对于较大型的程序，要找到配对的括号是很困难的。在 VLISP 环境下加载应用程序，经常提示的错误信息是：malformed list(残缺的表)，或 extra right paren(多余的右括号)，至于在哪里丢了括号或多了括号，VLISP 却无法指出，因此，准确迅速地找到丢失或多余的括号的位置，是程序设计者十分关心的一件事。

利用 VLISP 的括号匹配功能，搜寻以确定丢失(或多余) 的括号的方法是：将光标放在要

检查的起始位置上之后，按下相关的快捷键：

Ctrl+]：向程序前方匹配括号

Ctrl+[：向程序后方匹配括号

比如，当前光标在一个左括号处，Ctrl+]实现将光标移到配对的右括号处；Ctrl+[实现将光标移到上一层次的左括号处。设计者根据括号配对的顺序即可较迅速地找到丢失或多余的括号。

1.4.2 用语法分色检查拼写错误

VLISP 编程环境具有按语法分色的功能，系统内的标准函数(如 Setq、Defun、Getpoint 等) 被显示为蓝色；而 VLISP 不能识别的内容将显示为黑色，如用户定义的变量；字符串用粉红色。如果应当是系统函数而没有呈蓝色显示，就一定是函数名拼写错误。如果字符串中的文字不为粉红色，一定是在当前程序的后方丢了引号。

1.4.3 使用检查命令检查其他语法错误

语法错误是不符合 AutoLISP 语言规则的各种描述，程序设计者最希望解决的问题是：确定出错性质，并迅速地找到出错的位置，VLISP 提供了这方面的功能，做法如下：

(1) 检查全部的程序行。我们可先调入一个已经编制好的程序，比如 1.3 节中的 box.lsp 程序，然后在程序尾部加入含有错误的一行，如：(setvar "cmdecho")，之后进行以下工作：

菜单→Tools→Check Text in Editor(检查编辑器中的文字)

VLISP 调用检查命令，并在编译输出窗口中显示出错的性质，如图 1.6 所示。

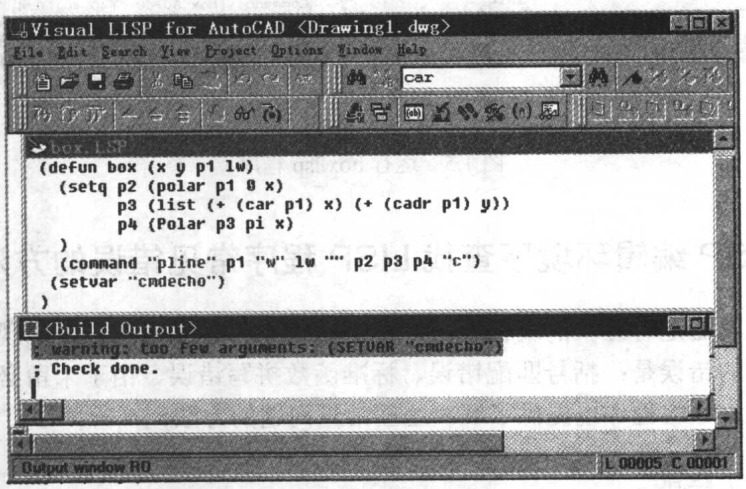


图 1.6 在编译输出窗口中显示出错性质

在编译输出窗口中双击这个错误信息，VLISP 就会激活编辑窗口，并将光标放在出错的程序行的头部，并亮显相关表达式，如图 1.7 所示。

(2) 检查选定的若干行程序片断。先在编辑器中选定需要检查的若干行程序片断，然后执行以下操作：

菜单→Tools→Check Selection(检查编辑器中选定的程序)

之后再重复第一步所做的类似工作，即可亮显出错位置。

VLISP 每次检查只能发现一处错误，应当在纠正错误后，再次进行检查，直到正确为止。

VLISP 还提供许多编辑调试程序的功能，诸如断点设置、单步运行、出错时中断等，读者可参考 VLISP 的相关书籍。

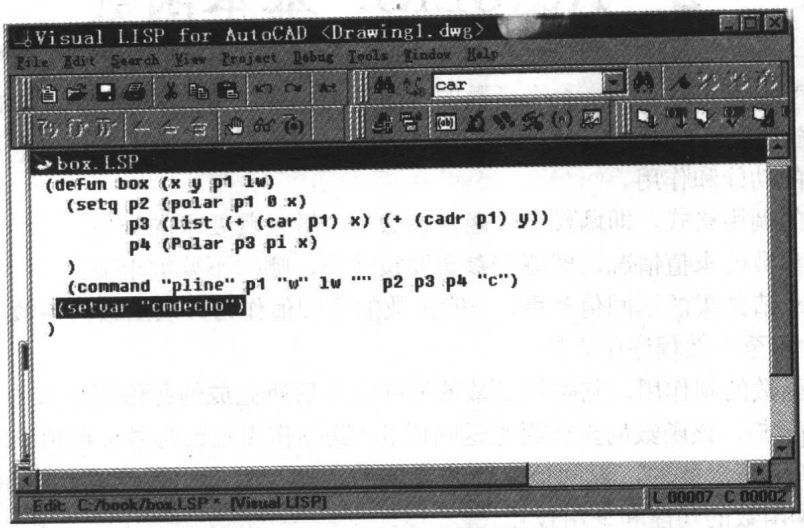


图 1.7　亮显出错位置