

ARITY PROLOG 5.0V

使用、参考手册

- Arity//Prolog 完全符合 DEC—10/20 标准。
- 提供解释、编译两种运行方式。
- 内存可利用到 16M；外存可利用 2000M 虚拟空间。
- 可以调用汇编、C、Pascal 及 Fortran；反之也行。
- 支持嵌入式 C 语言编译，允许与 C 混合编程。
- 有较高的运行速度。
- 提供301个固有谓词。
- 总观 Arity/Prolog 在诸多方面优于 Turbo/Prolog。

中国科学院软件研究所

时运电脑公司、海声软件开发公司、

北京科海培训中心联合印制

一九八九年八月

译者：张 敏 孙 华 查良钊 彭春龙

审核：白光野 章继红 付正翔 庞 林 朱 莉

本译书已经北京市新闻出版局核准。版权归中国科学院软件所所有，凡私自翻印者均属侵权行为。本所必究。

凡希望获得 Arity/Prolog 汉化软件的单位与个人，或已获得此软件，但尚有疑问的单位与个人请与下述单位联系：

- 单位名称：中国科学院软件研究所
- 通讯地址：北京中关村8718信箱
- 联系人：付正翔，张敏
- 联系电话：2565681或2563344转598

(内部发行)

译者序

Prolog 是逻辑程序设计 (PROgramming in LOGic) 的缩写, 它的核心思想由 R.Kowalski 于七十年代初期首先提出。1972年法国马赛大学Alain Colmerauer研究小组研制成功了第一个 Prolog 解释系统, 1977年英国爱丁堡大学在 DEC System10上实现的Prolog系统功能强而且包含编译程序, 是最早能与LISP竞争的系统。Prolog由于其文法简洁、表达能力强和独特的非过程性, 很快引起越来越多的注意并赢得了世界上广泛的支持。目前Prolog 已经和LISP一样成为最主要的人工智能程序设计语言, 许多自然语言处理系统和专家系统外壳是用Prolog写的。

1983年以来, Prolog开始在我国引起注意并逐渐推广应用, 已经成为人工智能领域尤其是专家系统建造的主要工具。到目前为止, 我国除了VAX-11, Altos, 3B, 386等机的VMS, UNIX (XENIX) 系统下有著名的 CProlog 和 HProlog 以外, 在 IBM-PC MS-DOS(CCDOS) 下可以运行从UNIX移植下来的CProlog, HProlog和Prolog-I, Prolog-II, Prolog/i, Prolog86以及Turbo-Prolog, 它们符合或基本符合DEC-10/20标准, 另外还有Micro-Prolog系列, 其中以Prolog/i, CProlog, Prolog-I, II性能为佳。近两年Turbo-Prolog由于其速度快而倍受喜爱。但是Turbo-Prolog由于过分强调速度而丢弃了许多Prolog的重要特性。又由于它没能突破MS-DOS 640K内存空间的限制, 不能支持较大规模的知识库而仅能应用于小模型试验, 很难构造实用人工智能系统。1988年美国Arity公司推出的Arity/Prolog 5.0版克服了Turbo-Prolog的缺点, 使得Prolog在IBM-PC机上用于开发人工智能程序尤其是建造知识库系统成为现实。它是我国能在IBM-PC机MS-DOS (CCDOS) 下使用的最优秀的Prolog系统, 其性能简介如下:

一 Arity/Prolog完全符合DEC10/20标准

在Arity/Prolog环境下, 用户程序既可以编译成可执行文件也可以解释执行, 因此保留了Prolog的询问等重要功能。它的解释器可以由用户使用Prolog程序、C汇编、Pascal/FORTRAN等进行扩充, 增加固有谓词, 而Turbo-Prolog仅有编译工作方式(包括内存编译)。它虽然声称符合DEC-10/20标准, 实际上差异很大, 很难利用已经大量存在的Prolog程序。Arity/Prolog完全符合DEC-10/20标准, 固有谓词及工作方式与Clocksion所著《Programming in Prolog》和CProlog完全一致。CProlog是英国爱丁堡大学在VAX-11、PDP-11的UNIX, VMS系统上用C语言实现, 可以在很多小型机上运行的著名Prolog系统, 我们已将其移植到386 (UNIX)、IBM-PC (CCDOS) 等计算机上。因此有很多小型机上编制的自然语言处理, 专家系统、知识库系统、CAN、CAD等人工智能程序可以很方便地利用。

一致化(Unification)和项的定义是Prolog的重要特点。Prolog的项可以是原子、数字(整数, 浮点数)、变量、结构(复合项)和表, 一个变量可以与原子、数字、结构和表任

意匹配一致化，表中的元素可以是任意一类项，也就是说一个变量既可以是数据（原子，数字，表）又可以是程序（谓词，子句）。Turbo-Prolog过分强调提高速度，为此它要求在程序开始部分定义变量和谓词，而且对变量传递的类型和格式要求十分严格。它要求每个项和变量必须严格定义为独立的整数、串、符号、整型表、浮点表、串表等，不允许是谓词子句而且变量传递和项之间匹配的类型必须严格一致。这些要求不但使用户编程十分不便，更重要的是Prolog语法简洁描述能力强的优点也受到限制。随之而来的问题是，与此有关的一些固有谓词在Turbo-Prolog中被舍弃了。例如：

1. 在自然语言处理、机器翻译、定理证明中非常有用的定义操作符谓词（op）在Turbo-Prolog中被丢掉了。

2. Turbo-Prolog去掉了调用目标谓词（call（GOAL）），而且增删子句谓词（assert，asserta，assertz，retract等）只能增删预先定义的事实，但是这些谓词对机器学习、自动程序生成等工作十分有用。

3. 有关结构成分的建立和取接谓词（STRUCT = .LIST，functor，arg等）和与子句有关的谓词（clause等）在Turbo-Prolog中都被舍弃，而它们是Prolog具有灵活的控制结构并且能够自动生成程序的重要基础。

因此，熟练的Prolog程序员在使用Turbo-Prolog时常常感到不便，而以上特性在Arity/Prolog的解释器和编译器中均被保留并与CProlog一致。

二 Arity/Prolog支持大容量知识库

Arity/Prolog 备有一个内部数据库系统，其内部数据库管理谓词与CProlog一致（例如：record，recorda，recordz，instance等）。并针对IBM-PC机扩充功能如下：

1. 内部数据库可以使用直致2千兆的虚拟存储器空间，实际范围由你的系统磁盘决定。

2. 内部数据库可以使用Lotus/Intel/Microsoft EMS扩展内存，也就是说，在IBM-PC的MS-DOS（CCDOS）下可以管理十几兆的物理存储器空间。

3. 除可以使用标记外（例如用instance（Ref，Term））内部数据库增加了B+树索引和Hash表索引功能。

内部数据库的设立和扩充为在IBM个人计算机上构造知识库系统提供了良好的基础。

三 Arity/Prolog吸取了过程语言的优点

Prolog是一种非过程性语言。然而，在实际应用中常会遇到一些情况采用过程处理更方便，效率也更高。例如在过程语言中经常用到for next和if then等语句，在Prolog编程时遇到此类情况就显得笨拙了。Arity/Prolog为此扩充了相应的过程性语言功能，可以进行加1、减1计数循环，也可以执行开关语句（case），还有if then、if then else等。

更令人高兴的是，Arity/Prolog备有一个嵌入式C语言，Arity/Prolog允许与C语言混合编程。在Prolog程序中可以使用C语言的宏定义、常量、变量、结构、联合、枚举、指针、地址、数组、函数、表达式、包含文件等等。

Turbo-Prolog可以与汇编、C、Pascal/FORTRAN连接，但是由于没有提供方便的界面，很难实用。Arity/Prolog 为汇编提供了20个宏和库程序，为C、Pascal/FORTRAN 分别提供了17个库函数。它们在Prolog和相应语言之间对变量、短整数、长整数、浮点数、原子、串、函数进行类型判别和交换。

不但Arity/Prolog可以调用汇编、C、Pascal/FORTRAN,而且C、Pascal/FORTRAN程序也可以调用Arity/Prolog。

四 Arity/Prolog保持了高速度

Arity/Prolog 在保留并扩充了 Prolog 原有特性的基础上仍然保证了很高的执行速度。我们在IBM-PC/AT兼容机上对几种有代表性的Prolog系统测试速度如下表：

	汉诺塔（12个）	八皇后
Arity/Prolog（解释）	10.27（秒）	167.58（秒）
Arity/Prolog（编译）	1.37（秒）	37.90（秒）
Turbo-Prolog（编译）	0.38（秒）	10.05（秒）
Turbo-Prolog（内存）	0.38（秒）	10.33（秒）
Prolog/i（解释）	260（秒）	585（秒）
Prolog-I（解释）	136（秒）	330（秒）
Prolog86（解释）	240（秒）	560（秒）
CProlog（解释）	270（秒）	610（秒）
HProlog（解释）	60（秒）	170（秒）
Micro-Prolog（解释）	450（秒）	760（秒）

五 Arity/Prolog的其他特性

Arity/Prolog提供了301个固有谓词、20个汇编语言宏和子程序、17个C语言联接函数、7个Pascal/FORTAN联接函数。而Turbo-Prolog 仅提供了 129 个固有谓词，CProlog 也只有130个固有谓词。固有谓词的增加表明功能的增强。例如：

对数据处理Arity/Prolog不但提供了类似C语言的数学计算谓词和嵌入式C语言而且支持8087，80287等浮点处理器。

对回溯控制则增加了剪切谓词，它是截断谓词的扩充，使对回溯的控制更方便灵活。

增加了12个串处理谓词，提高了对字符串的处理能力。

针对I/O管理，增加了类似C语言流式文件管理能力和低级I/O接口。

提供了良好的交互介面，如光标、时间、日期、窗口、菜单和鼠标等控制管理谓词。并提供了一个全屏幕文本编辑谓词，与Turbo-Prolog不同的是用户生成的文本编辑窗和编辑控制键可以自由定义。

本书的出版由于时间仓促，再加上一些其它原因造成文中难免会有这样或那样的错误。请读者原谅！

Arity/Prolog 解释器和编译器使用手册

第一部分 启动 Arity/Prolog

总 目 录

序言.....	(1)
Arity/Prolog 解释器和编译器使用手册.....	(1)
Arity/Prolog 语言参考手册.....	(2)

第一部分 启动 Arity/Prolog

第 1 章 什么是Prolog.....	(3)
第 2 章 安装系统.....	(4)

2.1 启动 Arity/Prolog.....	(4)
2.2 退出Arity/Prolog.....	(4)
2.3 可选择的Arity/Prolog装配过程.....	(4)
2.3.1 从任一目录区中调用Arity/Prolog.....	(5)
2.3.2 启动解释器时将信息加入数据库.....	(5)
2.3.3 使用Arity/Prolog编辑器之外的编辑器.....	(5)
2.3.4 设置打开文件限制数.....	(6)

第 3 章 Arity/Prolog 菜单.....	(7)
----------------------------	-------

3.1 移动光标并进行选择.....	(8)
3.1.1 在菜单行方式下移动光标.....	(9)
3.1.2 在下拉菜单方式下移动光标并选择.....	(9)
3.1.3 快速键.....	(10)
3.1.4 功能键.....	(10)
3.1.5 会话框内的光标移动和选择.....	(11)
3.2 菜单选择项.....	(12)
3.2.1 File 菜单选择项.....	(13)
3.2.2 Edit 菜单选择项.....	(14)
3.2.3 Buffers菜单选择项.....	(15)
3.2.4 Info 菜单选择项.....	(17)
3.2.5 Debug 菜单选择项.....	(17)
3.2.6 Switch 菜单选择项.....	(18)
3.2.7 Help 菜单选择项.....	(18)

第4章 使用 Arify/Prolog.....	(19)
4.1 Arity/Prolog 编辑器.....	(19)
4.1.1 启动编辑器.....	(19)

4.1.2	使用编辑器	(20)
4.1.3	移动光标	(20)
4.1.4	删除、拷贝和恢复文本	(21)
4.1.5	文本查找和替换	(22)
4.1.6	文件合并	(23)
4.1.7	使用缓冲区	(23)
4.1.8	保存文件	(24)
4.1.9	使用辅助缓冲区	(25)
4.2	输入 Prolog 程序	(25)
4.2.1	保存示例文件	(28)
4.2.2	装入示例文件	(28)
4.2.3	错误处理	(28)
4.3	运行 Prolog 程序	(29)
4.3.1	浏览数据库	(29)
4.3.2	进行询问	(30)
4.3.3	删除已输入的内容	(31)
4.3.4	具有1000以上优先级算符的谓词	(31)
4.3.5	谓词命名的约定	(32)
4.4	加载文件到数据库	(32)
4.4.1	用于装入文件的谓词	(34)
4.4.2	嵌入的 consult 指令	(34)
4.5	转换到 MS-DOS	(35)
4.6	使用调试器	(35)
4.6.1	调试器是如何工作的	(36)
4.6.2	运行调试器	(37)
4.6.3	调试一个程序	(38)
4.6.4	关闭调试器和消除监测点	(40)
4.6.5	从解释器中调用调试器	(41)
4.7	保存数据库的内容	(41)
4.7.1	恢复数据库内容	(42)
4.7.2	消除数据库的变化	(42)
4.7.3	保存和恢复数据库的谓词	(42)
4.8	Help 文件包	(43)
4.8.1	使用 Help 文件	(43)
4.8.2	增加 Help 文件	(44)
4.9	Arity/Prolog 信息	(44)
4.10	退出解释器	(47)
第5章	开发应用软件	(48)
5.1	使用窗口环境	(48)

5.2 程序设计风格.....	(49)
第6章 在解释器中加入可求值谓词.....	(51)
 第二部分 建立编译的应用软件	
第7章 Arity/Prolog 编译器介绍.....	(57)
7.1 apc命令.....	(58)
7.2 连接每个文件建立一个可执行映象.....	(59)
第8章 编译应用软件的要点.....	(61)
8.1 进一步的了解.....	(61)
8.1.1 可执行文件.....	(61)
8.1.2 数据库文件.....	(61)
8.1.3 环境文件.....	(62)
8.1.4 浮点协处理器.....	(62)
8.2 软件开发过程简介.....	(62)
8.3 Prolog 文件的结构.....	(63)
第9章 被编译的程序代码说明.....	(65)
9.1 为程序标识主入口点.....	(65)
9.2 restart 谓词.....	(66)
9.3 公用说明.....	(66)
9.4 外部说明.....	(67)
9.5 显式说明.....	(67)
9.5.1 default (invisible) 说明.....	(69)
9.6 模块说明.....	(69)
9.7 使用后辍interp.....	(69)
9.8 装入程序数据库.....	(70)
9.8.1 在一个被编译程序的数据库中装入子句.....	(70)
9.8.2 启动时装入数据库.....	(72)
9.9 编译时和运行时的操作符定义.....	(73)
第10章 编译与连接.....	(74)
10.1 编译.....	(74)
10.1.1 编译命令行的示例.....	(74)
10.1.2 ctrlvis 文件.....	(75)
10.1.3 使用批处理文件.....	(75)
10.2 连接.....	(76)

10.2.1	使用连接文件.....	(77)
10.2.2	使用 make 文件.....	(77)
10.3	出错处理.....	(78)
第11章 环境文件.....		(79)
11.1	建立环境文件.....	(80)
11.1.1	局部栈大小.....	(80)
11.1.2	全局栈大小.....	(81)
11.1.3	尾随栈大小.....	(81)
11.1.4	设置最小和最大页面分配.....	(81)
11.1.5	溢出文件.....	(82)
11.1.6	IDB 文件.....	(82)
11.1.7	扩展的存储器支持.....	(83)
11.2	缺省环境文件设置.....	(83)
第12章 远 PROLOG.....		(84)
12.1	段说明.....	(84)
12.2	远代码段的说明调整.....	(84)
12.3	clone 实用程序.....	(85)
12.4	连接远代码段.....	(85)
第13章 优化编译代码.....		(87)
13.1	包含 mode 说明.....	(87)
13.2	编译算术运算优化.....	(88)
第14章 说明编译器.....		(89)

第三部分 与其它程序设计语言相接

第15章 书写可调用的汇编语言子程序.....		(93)
15.1	说明汇编语言子程序名字和自变量.....	(93)
15.1.1	命名汇编语言子程序.....	(93)
15.1.2	说明子程序是外部的和可见的.....	(93)
15.2	堆栈和寄存器的使用.....	(94)
15.3	汇编语言宏.....	(95)
15.3.1	evalsucced	(95)
15.3.2	evalfail	(95)
15.3.3	evalargref	(95)
15.4	汇编语言子程序.....	(95)

15.4.1	getshort_a	(96)
15.4.2	getint_a	(96)
15.4.3	putshort_a	(96)
15.4.4	putint_a	(96)
15.4.5	getlong_a.....	(96)
15.4.6	putlong_a.....	(97)
15.4.7	整数存储.....	(97)
15.4.8	gettxt_a	(97)
15.4.9	puttxt_a	(97)
15.4.10	putatm_a	(98)
15.4.11	getflt_a.....	(98)
15.4.12	putflt_a.....	(98)
15.4.13	getfunctor_a.....	(98)
15.4.14	getfuncarg_a.....	(99)
15.4.15	putfunctor_a.....	(99)
15.4.16	findtype_a.....	(99)
15.4.17	eqrefs_a.....	(100)
15.5	汇编语言求值谓词实例.....	(100)

第16章 书写可调用的 c 函数.....(104)

16.1	说明C函数的名字和自变量.....	(104)
16.1.1	说明函数为外部的和可见的.....	(104)
16.2	转换数据类型.....	(105)
16.3	Arity/Prolog C 子程序.....	(106)
16.3.1	getshort_c	(106)
16.3.2	getint_c	(106)
16.3.3	putshort_c	(106)
16.3.4	putint_c	(107)
16.3.5	getlong_c.....	(107)
16.3.6	putlong_c.....	(107)
16.3.7	整数存储.....	(108)
16.3.8	gettxt_c	(108)
16.3.9	puttxt_c	(108)
16.3.10	putatm_c.....	(109)
16.3.11	getflt_c	(109)
16.3.12	putflt_c	(109)
16.3.13	getfunctor_c	(109)
16.3.14	getfuncarg_c	(110)
16.3.15	putfunctor_c	(110)

16.3.16	findtype_c	(110)
16.3.17	eqrefs_c	(111)
16.4	返回成功或失败	(111)
16.5	编译Lattice C函数	(111)
16.6	编译Microsoft C函数	(111)
16.7	连接C 模块	(112)
16.8	程序设计例子	(112)
16.8.1	把C求值谓词加到解释器中	(113)
16.8.2	调用C函数的独立程序	(115)
16.9	从C函数调用 Prolog	(117)
第17章	书写可调用的PASCAL和FORTRAN 过程	(118)
17.1	说明函数名字和自变量	(118)
17.1.1	Pascal子程序的外部性和可见的说明	(118)
17.2	转换数据类型	(119)
17.3	Arity/Prolog Pascal 函数	(119)
17.3.1	getshort_p	(120)
17.3.2	getint_P	(120)
17.3.3	putshort_P	(120)
17.3.4	putint_p	(120)
17.3.5	getlong_p	(120)
17.3.6	putlong_p	(121)
17.3.7	整数存储	(121)
17.3.8	gettext_p	(121)
17.3.9	puttxt_p	(121)
17.3.10	putatm_p	(122)
17.3.11	getflt_p	(122)
17.3.12	putflt_p	(122)
17.3.13	getfunctor_p	(122)
17.3.14	getfuncarg_p	(123)
17.3.15	putfunctor_p	(123)
17.3.16	findtype_p	(123)
17.3.17	eqrefs_p	(123)
17.4	返回成功或失败	(124)
17.5	编译Pascal 函数	(124)
17.5.1	连接Pascal 函数	(124)
17.6	Pascal 程序设计例子	(124)
17.7	Fortran程序设计例子	(127)
17.8	从Pascal 函数中调用 Prolog	(128)

附录A	词汇表	(130)
附录B	Arity/Prolog信息	(132)
B.1	MS—DOS 信息	(132)
B.2	语法错误	(132)
B.3	运行时错误	(133)
B.4	编译器信息	(135)
附录C	用其它语言书写的求值谓词	(137)
附录D	谓词一览	(139)
附录E	Arity/Prolog 编辑器快速参考指南	(151)
附录F	推荐参阅文献	(153)

Arity/Prolog 语言参考手册

符号约定	(157)
第1章 语言结构和控制	(158)
1.1 一致化	(158)
1.2 Prolog 数据类型	(158)
1.2.1 变量	(159)
1.2.2 原子	(159)
1.2.3 整数	(159)
1.2.4 浮点数	(159)
1.2.5 串	(160)
1.2.6 数据库参引数	(160)
1.2.7 结构	(160)
1.2.8 表	(161)
1.3 程序控制	(161)
1.3.1 回溯	(162)
1.3.2 repeat—fail 循环	(163)
1.3.3 递归	(164)
1.3.4 截断	(164)
1.3.5 截断—失败组合	(165)
1.3.6 使用剪切	(166)
1.3.7 使用case 控制算符	(167)
第2章 算术运算及算术表达式	(169)
2.1 算术运算符	(170)
2.2 算术求值谓词	(171)

2.3 计数器谓词	(172)
第3章 处理项	(173)
3.1 分类项	(173)
3.2 使项一致化	(174)
3.3 比较项	(175)
3.4 转换项	(177)
3.4.1 struct = .. List	(177)
3.4.2 functor(? struct, ? Name, ? Arity)	(177)
3.4.3 arg(+N, +Term, -Value)	(177)
3.4.4 argo(+N, +Term, -Value)	(178)
3.4.5 argrep(+Term, +N, +Arg, -Newstruct)	(178)
3.4.6 name(? Atom, ? List)	(178)
3.4.7 length(+List, -N)	(179)
3.5 收集项	(179)
3.5.1 bagof(+Term, +Goal, -Bag)	(179)
3.5.2 setof(+Term, +Goal, -Set)	(180)
3.5.3 findall(+Term, +Goal, -List)	(181)
第4章 管理数据库	(182)
4.1 数据库中的项	(183)
4.1.1 recorda(+Key, +Term, -Ref)	(184)
4.1.2 recordz(+Key, +Term, -Ref)	(185)
4.1.3 record_after(+Ref, +Term, -NewRef)	(185)
4.1.4 recorded(+Key, +Term, -Ref)	(185)
4.1.5 recorded_tro(+Key, +Term, -Ref)	(185)
4.1.6 instance(+Ref, -Term)	(185)
4.1.7 Key(+Key, -Ref)	(185)
4.1.8 keys(+Key)	(185)
4.1.9 nref(+Ref, -Next)	(186)
4.1.10 pref(+Ref, -Prev)	(186)
4.1.11 nth_ref(Key, +N, -Ref)	(186)
4.1.12 replace(Ref, +Term)	(186)
4.1.13 erase(+Ref)	(186)
4.1.14 eraseall(+Key)	(186)
4.1.15 expunge	(186)
4.1.16 hard-erase(+Ref)	(187)
4.1.17 如何使用recorded, recorded—tro, erase, hard-erase	(187)
4.2 数据库中的子句	(189)

4.2.1	clause(+ Head, - Body)	(189)
4.2.2	asserta(+ Clause)	(189)
4.2.3	assertz(+ Clause)	(190)
4.2.4	assert(+ Clause)	(190)
4.2.5	retract(+ Clause)	(190)
4.2.6	abolish(+ Name/Arity)	(190)
4.3	检测数据库	(190)
4.3.1	listing	(190)
4.3.2	current-predicate(- Predicate)	(191)
4.4	保存和恢复数据库	(191)
4.4.1	save	(191)
4.4.2	restore	(192)
4.5	用域进行工作	(192)
4.5.1	creat_world(+ World_name)	(193)
4.5.2	code_world(- Old, + New)	(193)
4.5.3	data_world(- Old, + New)	(194)
4.5.4	what_worlds(- X)	(194)
4.5.5	delete_world(+ World_name)	(194)
第5章 索引数据库的数据		(195)
5.1	b-树	(195)
5.1.1	reordb(+ Tree_name, + Sort_Key, + Term)	(196)
5.1.2	retrieveb(+ Tree_name, ? Sort_Key, ? Term)	(197)
5.1.3	betweenb(+ Tree_name, + Key1, + Key2, + Relation1, + Relation2, - Key, - Term)	(197)
5.1.4	betweenkeysb(+ Tree_name, Key1, + Key2, - Key)	(198)
5.1.5	defineb(+ Tree_name, + Splitsize, + Uniqueness, + Order)	(198)
5.1.6	replaceb(+ Tree_name, Sort_key, + Old_term, + New_term)	(200)
5.1.7	remoneb(+ Tree_name, + Sort_key, + Term)	(200)
5.1.8	nhat_btrees(- Btree)	(200)
5.1.9	removeallb(+ Tree_name)	(200)
5.2	建立 hash 表	(201)
5.2.1	recordh(+ Table_name, + Sort_Key, + Term)	(202)
5.2.2	retrieveh(+ Table_name, ? Sort_key, ? Term)	(202)
5.2.3	defineh(+ Table_name, + HashBuckets)	(203)
5.2.4	removeh(+ Table_name, Sort_key, + Term)	(203)
5.2.5	removeallh(+ Table_name)	(203)
5.3	对b-树和Hash 表使用多重索引	(203)

第 6 章 标准输入输出.....	(205)
6.1 项 I/O.....	(205)
6.1.1 read(- Term)	(206)
6.1.2 write(+ Term)	(206)
6.1.3 writeq(+ Term)	(206)
6.1.4 display(+ Term)	(206)
6.1.5 op(+ Prec, + Assoc, + Op).....	(207)
6.1.6 current_op(? Prec, ? Assoc, ? Op)	(209)
6.1.7 reset_op	(209)
6.2 字符 I/O.....	(209)
6.2.1 geto(- Char)	(210)
6.2.2 get(- Char)	(210)
6.2.3 geto__noecho(Char)	(210)
6.2.4 Keyb(- ASCII, Scan)	(210)
6.2.5 Keyb-Peek(- ASCII, - Scan)	(211)
6.2.6 flush	(212)
6.2.7 skip(+ Char)	(212)
6.2.8 put(+ Char)	(212)
6.2.9 nl	(212)
6.2.10 tab(+ Name).....	(212)
6.3 串 I/O.....	(212)
第 7 章 文件输入输出.....	(214)
7.1 打开和关闭文件	(215)
7.1.1 Create(- Handle, + Filemae).....	(216)
7.1.2 Gopen(- Handle, + Filename, + Access)	(216)
7.1.3 P-open(- Handle, + Filename, + Access).....	(216)
7.1.4 close(+ Handle)	(217)
7.2 文件中移动	(217)
7.2.1 seek(+ Handle, + Offset, + Method, - Newloc)	(217)
7.3 改变输入输出方向	(218)
7.3.1 stdin, stdout, stdinout	(218)
7.3.2 file__list	(219)
7.4 标准 Prolog 输入输出	(219)
7.4.1 see(+ Filename)	(219)
7.4.2 seeing(- Filename).....	(219)
7.4.3 seen	(220)
7.4.4 see__h(+ Handle)	(220)
7.4.5 tell(+ Filename)	(220)

7.4.6	telling(- Filename)	(220)
7.4.7	told	(220)
7.4.8	tell__h(Hanolle)	(220)
第8章 低级 I/O		(221)
8.1	in(+ Port, - Byte)	(221)
8.2	out(+ Port, + Byte)	(221)
第9章 字符串		(222)
9.1	管理字符串	(224)
9.1.1	string-search	(224)
9.1.2	substring(+ Instring, + N, + Length, - Outstring)	(224)
9.1.3	nth__char(+ N, + String, - Char)	(225)
9.1.4	string__length(+ String, - Length)	(225)
9.1.5	concat	(225)
9.2	转换字符串	(226)
9.2.1	string__term(?String, ?Term)	(226)
9.2.2	atom__string(?Atom, ?String)	(226)
9.2.3	int__text(?Integer, ?Text)	(226)
9.2.4	float__text(?Float, ?Text, ?Format)	(227)
9.2.5	ilst__text(?List, ?String)	(227)
9.3	字符串输入输出	(227)
9.3.1	read__string	(227)
9.3.2	read__line(+ Handle, - Line)	(228)
第10章 窗口		(229)
10.1	建立和修改窗口	(229)
10.2	回送窗口信息	(230)
10.3	current__window(- Old, ?New)	(231)
10.4	hide__window(- Current, + New)	(231)
10.5	what__windows(- Name)	(231)
10.6	resiz__window(+ Rows, + Columns)	(232)
10.7	move__window(+ Rows, + Columns)	(232)
10.8	relabel__window(+ Lable)	(232)
10.9	recolor__window(+ Window__attr, + Border__attr)	(232)
10.10	store__windows	(233)
10.11	refresh	(233)
10.12	delete__window(+ Name)	(233)
10.13	上托窗口	(233)