

HOPE

Turbo系列语言混合编程



- Turbo Prolog 与其它语言的接口
- Turbo C 与其它语言的接口
- Turbo Pascal 与其它语言的接口
- Turbo Basic 与其它语言的接口
- 混合程序的调试
- 混合编程的参考资料



中国科学院希望高级电脑技术公司

TP312
L362

354175

Turbo 系列语言混合编程

李 文 编

中国科学院希望高级电脑公司

序 言

JS204/8

Borland 公司推出众多风靡世界的 Turbo 系列语言, Turbo Pascal、Turbo C、Turbo Prolog、Turbo Basic、Turbo C++。这些年轻的语言以其优良的性能和用户界面,很快获得世界各地用户的欢迎。Borland 率先使用的集成环境、联机帮助与热键驱动已当今软件的用户界面的标准。独有的内部调试器使编程和调试的效率倍增,开发周期骤缩。此外, Borland 辅助开发实用程序 Turbo Debugger、Turbo Profiler、MAKE、TLINK、TLIB、TCREF、GREP、TOUCH、OBJXREF 和 TC(P)HELP 等,更使 Turbo 系列语言大放光彩。形成了编程(集成环境)、调试(内部源级调试器和 Turbo Debugger)、运行(集成环境中模拟)、剖视(Turbo Profiler)和工程管理(MAKE、Project 性能)一体化的良好环境。

Turbo 系列语言都有其解决问题的方向,如果综合各自的优点,进行混合编程,那么编程水平就将更上一层楼。

现在世界软件市场异彩纷呈, Microsoft 公司开发的语言 Microsoft C、QuickC、Microsoft Pascal、Quick Pascal、Microsoft Fortran 和 Microsoft Basic、Quick Basic 也很受用户欢迎。如果能超越 Borland 和 Microsoft 公司的障碍,把两者的优点结合在一起编程,那将无往而不胜。

Turbo C、Turbo Pascal、Turbo Prolog 等高级语言与汇编语言接口能直接使用 BIOS、DOS 的功能、直接对串行口、视频、游戏棒与鼠标等硬件进行存取、能进行内存驻留程序、设备驱动程序编程,提高了处理的速度(特别是图形处理速度),扩展了控制。

高级语言之间的接口能充分发挥各种语言的独特优势,使编程更具灵活性;高级语言与数据库管理语言的接口能减少繁琐的数据项管理(数据项插入、删除、检索、排序);高级语言与 BIOS 和 DOS 的接口扩展了高级语言的低级功能,减少使用汇编带来的繁琐的负担。

顾名思义,接口就是联络,搭起一座进行资源共享的桥梁。

接口的要素在于参数传递协议、函数返回协议、寄存器协议、数据的内部格式的差异、不同编译器生成的 OBJ 的兼容性(特别是相应的汇编语言的兼容性)、生成的 OBJ 文件所用的汇编级的选项的差别、启动代码的主次、重复的屏幕输入/输出和文件操作等相同功能的扬弃和内存管理的处理等。

本书针对以上难题进行探讨,介绍了混合编程的基础,混合编程的技巧,混合编程的调试,混合编程的工程管理等。虽然如此,但是接口论属于高级编程的领域,加上此专题的资料不多,因此本书只是结合长期经验的一次尝试。

编者

目 录

序言

第一部分 Turbo Prolog 与其它语言的接口	1
第一章 Turbo Prolog 与其它语言接口的约定	1
§ 1.1 声明外部谓词	1
§ 1.2 调用约定和参数压栈顺序	1
§ 1.3 命名约定	2
第二章 Turbo Prolog 与 Turbo C 的接口	3
§ 2.1 Turbo Prolog 调用 Turbo C 过程	3
§ 2.1.1 说明外部谓词	3
§ 2.1.2 建立 C 函数源程序	3
§ 2.1.3 Turbo C 编译选项和连接	3
§ 2.1.4 动态存贮分配	4
§ 2.1.5 传递复合对象到其它语言的程序	5
§ 2.1.6 例子	6
§ 2.2 Turbo C 调用 Turbo Prolog	9
第三章 Turbo Prolog 与 Turbo Assembler 的接口	12
§ 3.1 声明外部谓词	12
§ 3.2 调用约定和参数压栈	12
§ 3.3 命名约定	12
§ 3.4 编写汇编语言谓词	13
§ 3.4.1 例: 实现 double 谓词	16
§ 3.5 用多重流模式实现谓词	18
§ 3.6 从汇编函数调用 Turbo Prolog 谓词	19
§ 3.7 表和函子	21
第四章 Turbo Prolog 与 MS-Fortran 4.0 的接口	25
§ 4.1 系统设置	25
§ 4.2 Turbo Prolog 调用 MS-Fortran 过程	25
§ 4.2.1 在 Prolog 中说明外部谓词	25
§ 4.2.2 定义 Fortran 子程序并建立源程序	25
§ 4.2.2.1 命名约定	26
§ 4.2.2.2 参数约定	26
§ 4.2.2.3 屏幕输出	26
§ 4.2.3 连接步骤	27
§ 4.2.4 例子	27
§ 4.3 Fortran 调用 Turbo Prolog	29

§ 4.4 常用接口例程序、预处理程序的建立以及 Fortran 库的改造	30
§ 4.4.1 常用接口例程序的建立	30
§ 4.4.2 预处理程序	31
§ 4.4.3 Fortran 库的改造	33
第五章 Turbo Prolog 访问 dBASE III 数据文件	39
§ 5.1 Prolog 事实与 dBASE III 记录	39
§ 5.2 dBASE III 中 DBF 的存贮结构	39
§ 5.3 把 DBF 记录转换成 Turbo Prolog 事实	40
§ 5.4 利用 Turbo Prolog 工具库访问 dBASE III 数据文件	40
§ 5.4.1 一次读出 dBASE III 文件的所有记录	41
§ 5.4.2 一次读出一个 dBASE III 记录	42
第六章 Turbo Prolog 与 DOS 系统级的接口	49
§ 6.1 访问 DOS	49
§ 6.1.1 system/1	49
§ 6.1.2 system/3	49
§ 6.1.3 envsymbol/2	50
§ 6.1.4 date/3 和 time/4	50
§ 6.1.5 comline/1	51
§ 6.2 访问硬件: 低级支撑	52
§ 6.2.1 bios/3 和 bios/4	52
§ 6.2.2 ptr_dword/3	53
§ 6.2.3 membyte/3 和 memword/3	53
§ 6.2.4 port_byte/2	53
§ 6.3 例子:	54
第二部分 Turbo C 与其它语言的接口	56
第七章 Turbo C 与汇编的接口	56
§ 7.1 在 Turbo C 中使用嵌入式汇编	56
§ 7.1.1 嵌入式汇编如何工作	58
§ 7.1.1.1 Turbo C 如何知道使用嵌入式汇编模式	61
§ 7.1.1.2 激活 Turbo Assembler 处理嵌入式汇编	62
§ 7.1.1.3 Turbo C 在何处汇编嵌入式汇编码	62
§ 7.1.1.4 将-1 开关用于 80186/80286 指令	63
§ 7.1.2 嵌入式汇编语句的格式	64
§ 7.1.2.1 嵌入式汇编中的分号	64
§ 7.1.2.2 嵌入式汇编中的注解	64
§ 7.1.2.3 访问结构/联合的元素	65
§ 7.1.3 嵌入式汇编示例	67
§ 7.1.4 嵌入式汇编的限制	70
§ 7.1.4.1 内存和地址操作数限制	70

§ 7.1.4.2 嵌入式汇编中缺少隐含的自动变量大小	71
§ 7.1.4.3 必须保存寄存器	73
§ 7.1.4.3.1 保存调用函数和寄存器变量	73
§ 7.1.4.3.2 抑制内部寄存器变量	73
§ 7.1.5 嵌入式汇编码相对于纯 C 代码的缺点	73
§ 7.1.5.1 降低了可移植性和可维护性	73
§ 7.1.5.2 降低了编译速度	73
§ 7.1.5.3 仅可由 TCC 使用	74
§ 7.1.5.4 损失了优化能力	74
§ 7.1.5.5 限制了对错误的反跟踪	74
§ 7.1.5.6 调试限制	74
§ 7.1.5.7 用 C 开发而用嵌入式汇编编译最终代码	75
§ 7.2 在 Turbo C 中调用 Turbo Assembler 函数	75
§ 7.2.1 Turbo C 与 Turbo Assembler 的接口机制	76
§ 7.2.1.1 内存模式和段	76
§ 7.2.1.1.1 简化的段伪指令与 Turbo C	76
§ 7.2.1.1.2 过时风格的段伪指令与 Turbo C	78
§ 7.2.1.1.3 段缺省: 何时需要装载段?	80
§ 7.2.1.2 公共量和外部量	83
§ 7.2.1.2.1 下划线	83
§ 7.2.1.2.2 大小写字母的意义	84
§ 7.2.1.2.3 标号类型	84
§ 7.2.1.2.4 远类型的外部量必须在任何段之外	85
§ 7.2.1.3 链接器命令行	87
§ 7.2.2 Turbo Assembler 与 Turbo C 的交互性	87
§ 7.2.2.1 参数传递	87
§ 7.2.2.2 保存寄存器	93
§ 7.2.2.3 返回值	94
§ 7.2.3 从 Turbo C 中调用 Turbo Assembler 函数	95
§ 7.2.4 Pascal 调用约定	98
§ 7.3 在 Turbo Assembler 中调用 Turbo C	99
§ 7.3.1 链入 C 的启动码	99
§ 7.3.2 确保已正确设置了段	100
§ 7.3.3 执行调用	100
§ 7.3.4 在 Turbo Assembler 调用 Turbo C 函数	101
第八章 Turbo C 与 DOS、BIOS 的接口	104
§ 8.1 寄存器	104
§ 8.2 中断	105
§ 8.2.1 使用 DOS 中断的注意事项	105
§ 8.3 利用功能调度器实现中断	105
§ 8.4 使用 BIOS 中断	145

§ 8.5 小结	152
----------------	-----

第三部分 Turbo Pascal 与其它语言的接口153

第九章 Turbo Pascal 与汇编语言的接口153

§ 9.1 扩展 Turbo Pascal153

§ 9.1.1 嵌入代码153

§ 9.2 嵌入指令155

§ 9.3 外部过程156

§ 9.3.1 外部函数156

§ 9.3.2 使用全程数据和过程158

§ 9.3.3 使用 Turbo Assembler160

§ 9.4 嵌入代码与外部过程的比较163

§ 9.5 使用 Turbo Debugger163

第十章 再论 Turbo Pascal 与汇编的接口169

§ 10.1 Turbo Pascal 内存映象169

§ 10.1.1 程序段前缀169

§ 10.1.2 代码段169

§ 10.1.3 全局数据段170

§ 10.1.4 堆栈170

§ 10.1.5 堆171

§ 10.2 Turbo Pascal 中寄存器的用法171

§ 10.3 近调用还是远调用 ?171

§ 10.4 与 Turbo Pascal 共享信息171

§ 10.4.1 \$L 编译伪指令和外部子程序171

§ 10.4.2 PUBLIC 伪指令: 使 Turbo Pascal 可利用 Turbo Assembler 的信息 172

§ 10.4.3 EXTRN 伪指令: 使 Turbo Assembler 可利用 Turbo Pascal 的信息 173

§ 10.4.3.1 使用 EXTRN 对象的限制175

§ 10.4.4 使用段定位175

§ 10.4.5 无效代码的消除176

§ 10.5 Turbo Pascal 参数传递约定176

§ 10.5.1 值参176

§ 10.5.1.1 标量类型176

§ 10.5.1.2 实型177

§ 10.5.1.3 单精度、双精度、扩展的和复合型: 8087 类型177

§ 10.5.1.4 指针177

§ 10.5.1.5 串177

§ 10.5.1.6 记录和数组177

§ 10.5.1.7 集合177

§ 10.5.2 变量参数178

§ 10.5.3 栈的维护178

§ 10.5.4 存取参数	178
§ 10.5.4.1 使用 BP 寄存器寻址堆栈	178
§ 10.5.4.1.1 ARG 伪指令	179
§ 10.5.4.1.2 .MODEL 和 Turbo Pascal	180
§ 10.5.4.1.3 使用另一个基址或变址寄存器	180
§ 10.6 Turbo Pascal 中的函数结果	181
§ 10.6.1 标量函数结果	181
§ 10.6.2 实型函数结果	181
§ 10.6.3 8087 函数结果	181
§ 10.6.4 串函数结果	181
§ 10.6.5 指针函数结果	181
§ 10.7 为局部数据分配空间	181
§ 10.7.1 分配私有静态存储区	181
§ 10.7.2 分配动态存储区	182
§ 10.8 由 Turbo Pascal 调用汇编语言子程序的例子	183
§ 10.8.1 通用 16 进制转换子程序	183
§ 10.8.2 交换两个变量	186
§ 10.8.3 扫描 DOS 环境	189
第十一章 Turbo Pascal 与 DOS 和 BIOS 的接口	194
§ 11.1 8088 寄存器	194
§ 11.2 DOS 单元	195
§ 11.3 寄存器集	195
§ 11.4 磁盘驱动功能调用	197
§ 11.4.1 报告磁盘空闲空间	197
§ 11.4.2 读取和设置文件属性	199
§ 11.4.3 目录列表	203
§ 11.4 视频功能调用	207
§ 11.4.1 报告当前视频模式	207
§ 11.4.2 设置光标大小	208
§ 11.4.3 从屏幕读字符	210
§ 11.5 时间和日期功能	211
§ 11.5.1 获取系统日期	211
§ 11.5.2 设置系统日期	213
§ 11.5.3 获取和设置系统时间	214
§ 11.5.4 获取和设置文件的时间和日期	216
§ 12.6 报告换档键状态	222
§ 11.6 Turbo Pascal DOS 单元	224
§ 11.6.1 DOS 单元常量	224
§ 11.6.2 DOS 单元数据类型	225
§ 11.6.2.1 DateTime 类型	226

§ 11.6.2.2 SearchRec 类型	226
§ 11.6.3 DosError 变量	226
§ 11.6.4 DOS 单元过程与函数	227
§ 11.6.4.1 中断支持子程序	227
§ 11.6.4.2 日期和时间例程	227
§ 11.6.4.3 磁盘和文件例程	227
§ 11.6.5 进程例程	227
第四部分 Turbo Basic 与其它语言的接口	240
第十二章 Turbo Basic 与 Turbo Assembler 的接口	240
§ 12.1 传递参数	240
§ 12.1.1 不在当前数据段的变量	242
§ 12.1.2 什么类型的调用?	242
§ 12.2 弹出堆栈	243
§ 12.3 为 Turbo Basic 创建一个汇编程序	243
§ 12.4 调用一个在线汇编子程序	243
§ 12.5 在内存中安装一个 Turbo Basic 子程序	245
§ 12.5.1 隐藏串	246
§ 12.5.2 绝对调用(CALL ABSOLUTE)	247
§ 12.5.2.1 到一固定内存位置作 CALL ABSOLUTE	247
§ 12.5.2.2 到内存不定位置作 CALL ABSOLUTE	248
§ 12.5.2.3 CALL ABSOLUTE 的其他问题	249
§ 12.6 调用中断	249
§ 12.7 样本程序	250
第五部分 混合编程程序的调试	253
第十三章 Turbo Debugger 调试的一个快速示例	253
§ 13.1 演示程序	253
§ 13.2 使用 Turbo Debugger	254
§ 13.2.1 菜单(The menus)	254
§ 13.2.2 状态行(The status line)	254
§ 13.2.3 窗口(The windows)	255
§ 13.3 使用 C 演示程序	256
§ 13.3.1 设置断点(Setting breakpoints)	257
§ 13.3.2 利用监视(Using watches)	257
§ 13.3.3 考察简单的 C 数据对象	257
§ 13.3.4 考察复杂的 C 数据的对象	259
§ 13.3.5 改变 C 数据值	259
§ 13.4 使用 Pascal 示例程序	260
§ 13.4.1 设置断点(Setting breakpoints)	261

§ 13.4.2 使用监视(Using watches)	262
§ 13.4.3 考察简单的 Pascal 数据对象	262
§ 13.4.4 考察复杂的 Pascal 数据对象	263
§ 13.4.5 改变 Pascal 数据值	264
第十四章 启动 Turbo Debugger	266
§ 14.1 准备待调试的程序	266
§ 14.1.1 准备 Turbo C 程序	266
§ 14.1.2 准备 Turbo Pascal 程序	266
§ 14.1.3 准备 Turbo 汇编程序	266
§ 14.1.4 准备 Microsoft 程序	267
§ 14.2 运行 Turbo Debugger	267
§ 14.3 命令行选择项	267
§ 14.3.1 装载配置文件(-c)	268
§ 14.3.2 显示更新方式(-d)	268
§ 14.3.3 获取帮助(-h 与 -?)	268
§ 14.3.4 进程 ID 转换(-i)	268
§ 14.3.5 击键记录(-k)	268
§ 14.3.6 汇编模式启动(-l)	268
§ 14.3.7 设置堆大小(-m)	268
§ 14.3.8 鼠标器支持(-p)	269
§ 14.3.9 远程调试(-r)	269
§ 14.3.10 源代码处理(-s)	269
§ 14.3.11 视频硬件(-v)	269
§ 14.3.1.2 覆盖池大小(-y)	270
§ 14.4 配置文件	270
§ 14.5 选项菜单	270
§ 14.5.1 语言命令	271
§ 14.5.2 宏菜单	271
§ 14.5.2.1 创建(Create)	271
§ 14.5.2.2 停止记录(Stop Recording)	271
§ 14.5.2.3 删除(Remove)	271
§ 14.5.2.4 全清>Delete All)	271
§ 14.5.3 显示选择命令	271
§ 14.5.3.1 显示切换	271
§ 14.5.3.2 整数格式	272
§ 14.5.3.3 屏幕行数	272
§ 14.5.3.4 制表键大小	272
§ 14.5.4 源命令路径	272
§ 14.5.5 保存选择项命令	272
§ 14.5.6 恢复选择项命令	273

§ 14.6 在 Turbo Debugger 中运行 DOS	273
§ 14.7 返回 DOS	273
第六部分 混合编程的参考资料	275
附录 A TASM 命令行参考	275
§ A.1 在 DOS 中启动 Turbo Assembler	275
§ A.2 命令行选择项	277
附录 B 混合编程实用程序	287
§ B.1 独立的 MAKE 实用程序	287
§ B.1.1 一个快速示例	287
§ B.1.1.1 创建一个 make 文件	288
§ B.1.1.2 使用一个 make 文件	289
§ B.1.1.3 步进	289
§ B.1.2 创建 make 文件	290
§ B.1.2.1 Make 文件的组成	290
§ B.1.3 使用 MAKE	301
§ B.1.3.1 命令行语法	301
§ B.1.3.2 中止 MAKE 的说明	302
§ B.1.3.3 BUILTINS.MAK 文件	302
§ B.1.3.4 MAKE 是如何查找 make 文件的	302
§ B.1.3.5 TOUCH 实用程序	302
§ B.1.3.6 MAKE 命令行选择项	303
§ B.1.4 MAKE 出错信息	303
§ B.1.4.1 致命错	303
§ B.1.4.2 一般错	304
§ B.2 Turbo Link	305
§ B.2.1 调用 TLINK	305
§ B.2.2 使用应答文件	306
§ B.2.3 TLINK 选择项	307
§ B.2.3.1 /x,/m,/s 选择项	307
§ B.2.3.2 /l 选择项	308
§ B.2.3.3 /i 选择项	308
§ B.2.3.4 /n 选择项	309
§ B.2.3.5 /c 选择项	309
§ B.2.3.6 /d 选择项	309
§ B.2.3.7 /e 选择项	309
§ B.2.3.8 /t 选择项	309
§ B.2.3.9 /v 选择项	309
§ B.2.3.10 /s 选择项	310
§ B.2.4 一些限制	310

§ B.2.5 出错消息	310
§ B.2.5.1 致命错	310
§ D.2.5.2 非致命错	311
§ B.2.5.3 警告	312
§ B.3 TLIB; Turbo 库管理员	312
§ B.3.1 使用目标模块库的优点	312
§ B.3.2 TLIB 命令行的组成	313
§ B.3.2 操作表(Operations)	313
§ B.3.3 使用应答文件	315
§ B.3.4 改进的操作:/c 选择项	315
§ B.3.5 例子	315
§ B.3.6 创建一扩展词典: /E 选择项	316
§ B.4 GREP: 一种文件查找实用程序	316
§ B.4.1 GREP 选择项	316
§ B.4.1.1 优先级次序	317
§ B.4.2 查找串	318
§ B.4.2.1 正则表达式中的操作符	318
§ B.4.3 文件说明	318
§ B.4.4 带说明的例子	318
§ D.5 OBJXREF:目标模块交叉引用实用程序	321
§ D.5.1 OBJXREF 命令行	321
§ D.5.1.1 命令行选择项	321
§ D.5.2 应答文件	322
§ D.5.2.1 自由形式的应答文件	322
§ D.5.2.2 连接器应答文件	322
§ D.5.2.3 /D 命令	323
§ D.5.2.4 /O 命令	323
§ D.5.2.5 /N 命令	323
§ D.5.3 OBJXREF 报告样本	323
§ D.5.3.1 按公用名报告(/RP)	324
§ D.5.3.2 按模块报告(/RM)	325
* § D.5.3.3 按引用报告(/RR)(缺省方式)	325
§ D.5.3.4 按外部引用报告(/RX)	325
§ D.5.3.5 按模块长度报告(/RS)	326
§ D.5.3.6 按类报告 (/RC)	326
§ D.5.3.7 按未引用符号名报告 (/RV)	326
§ D.5.3.8 冗长报告 (/RV)	327
§ D.5.4 使用 OBJXREF 的例子	327
§ D.5.5 OBJXREF 出错信息和警告	327
§ D.5.5.1 出错信息	327

§ D.5.5.2 警告	327
§ D.6 TCREF: 源模块交叉引用实用程序	328
§ D.6.1 应答文件	328
§ D.6.2 与 TLINK 的兼容	328
§ D.6.2.1 开关	329
§ D.6.2.2 全局(或连接器级)报告	329
§ D.6.2.3 局部(或模块级)报告	329
附录 C 嵌入汇编的助记符与机器码对照表	330

第一部分 Turbo Prolog 与其它语言的接口

Turbo Prolog 是基于逻辑程序设计的语言。从 1.0 版、1.1 版到 2.0 版,已有了很大的发展,成为开发专家系统等知识库系统的有力工具。

尽管 Turbo Prolog 在许多方面是一个非常好的工具,但是仍然需要使用其它语言。例如用 Pascal 或 Fortran 语言很容易完成数值积分运算,用汇编语言实现中断处理及低层操作更好。另外,如果用其它语言开发的程序已经解决了某些问题,这些工作不应被抛弃和否定。因此, Turbo Prolog 允许 Turbo Prolog 程序与其它语言连接。下面将介绍 Turbo Prolog 与其它语言的接口约定、 Turbo Prolog 与 Turbo C 的接口、 Turbo Prolog 与 MS-Fortran 4.0 的接口、 Turbo Prolog 与汇编语言的接口。此外,还介绍了 Turbo Prolog 访问 dBASE III 文件的方法,以及 Turbo Prolog 与 DOS 系统级的接口谓词。

第一章 Turbo Prolog 与其它语言接口的约定

在调用其它语言写的子程序和函数前,需要在 Turbo Prolog 中把它们说明成外部谓词,还应了解正确的调用约定和参数压栈顺序,以及如何命名外部谓词的不同流变体。下面三节分别叙述这三个问题。

注意:当提到 Turbo Prolog 时都是指 1.0 或更高版本。

§ 1.1 声明外部谓词

Turbo Prolog 允许通过使用一个全局谓词(global predicates)声明与其他语言进行接口。在声明后面附加一个语言说明,以便使 Turbo Prolog 能知道这个全局谓词是用哪一种语言实现的。

global predicates

add(integer,integer,integer) - (i,i,o),(i,i,o) language asm

scanner(string, token) - (i,o) language Pascal

Turbo Prolog 使接口语言明确化能简化活动记录和参数格式、调用和返回约定、段定义、链接和初始化等问题。

§ 1.2 调用约定和参数压栈顺序

8086 系列处理器使程序员能够对近子例程调用和远子例程调用进行选择。Turbo Prolog 创建大存贮模式程序,并要求所有对子例程的调用和从子例程返回都是远调用。

Turbo Prolog 支持多种调用约定,包括 C、Pascal 和 Assembler 等等。当使用 C 调用约定与子例程接口时,参数以反序压入堆栈,返回后栈指针自动调整;当与其他语言接口时,参数以正常次序入栈,被调函数负责从栈中取走参数。

在许多 8086 系列的语言编译程序中,可以选择 16 位指针或 32 位指针,16 位指针指

向的是默认段，而 Turbo Prolog 总是使用 32 位指针来访问所有内存。

Turbo Prolog 的类型以下列方式实现：

整型	2字节
实型	8字节(IEEE格式)
字符型	1字节(压入堆栈时用2字节)
字符串型	4字节 双字指针指向一个以NULL结尾的串
符号型	4字节 双字指针指向一个记录。

输出参数以指向某地址的 32 位指针形式压入堆栈，返回值必须赋给指针指向的单元。对输入参数，其值直接入栈，并且参数的大小取决于它的类型。

§ 1.3 命名约定

在 Turbo Prolog 中，一个谓词可以有多个类型的变体和多个输入输出流变体，每个类型和流变体有其各自的子程序。为了调用这些不同过程，必须为每一过程赋以唯一的名字，这是通过从 0 开始对具有相同谓词名的不同子程序向上进行编号来实现的。

例如，给出如下声明：

```
global predicates
```

```
add(integer, integer, integer) - (i,i,o),(i,i,i) language asm
```

第一个变体（流模式为(i,i,o)）被命名为 add_0，第二个变体（流模式为(i,i,i)）命名为 add_1。

Turbo Prolog 还允许程序员为全局谓词声明一个明确的名字，这是通过在声明后跟"as public name"来实现的。下例中，全局谓词 Pred 将用 my_pred 而不是 pred_0 来声明：

```
global predicates
```

```
pred(integer, integer) - (i,o) language asm as "my_pred"
```

这种方法在用户命名只有一种流模式的谓词时采用。如果存在多种流模式，则用户只能为每个变体声明一个名字。以 add 谓词为例，谓词定义可能如下：

```
global predicats
```

```
add(integer,integer,integer) - (i,i,o) language asm as "doadd"
```

```
add(integer,integer,integer) - (i,i,i) language asm as "add_check"
```

第一个变体（流模式为(i,i,o)）被命名为 doadd，第二个变体（流模式为(i,i,i)）被命名为 add_check。注意这种命名方法需要分别对各变体进行声明。

第二章 Turbo Prolog 与 Turbo C 的接口

§ 2.1 Turbo Prolog 调用 Turbo C 过程

§ 2.1.1 说明外部谓词

在 Turbo Prolog 主模块中, 应说明被调用的 C 函数为全局谓词。下列的 Turbo Prolog 程序说明了如何访问一个用 Turbo C 编写的过程 double。可用如下的 Turbo Prolog 语句调用该过程:

```
double(MyInputInteger, MyOutVar)
```

在调用前 MyInputInteger 约束为一个整数值, 调用后 MyOutVar 被约束为该值的两倍。

```
global predicates
double(integer,integer) - (i,o) language C
goal
double(5,N),
write("5 doubled is ",N).
```

在包含调用 double 的 Turbo Prolog 程序中, 必须在全局谓词段中指明实现该过程的语言。

§ 2.1.2 建立 C 函数源程序

在 C 函数源程序中, 必须按 Turbo Prolog 命名约定为 C 函数命名, 即 C 函数名就是 Prolog 的谓词名紧接一下划线和与一流模式对应的整数。

对于上例的 double 有如下的 C 程序:

```
void double_0(int in, int *out)
{
    *out = in + in;
}
```

§ 2.1.3 Turbo C 编译选项和连接

为了连接 Turbo Prolog 与 Turbo C 模块, 必须带有下列选择项编译 C 源程序

(1) 用 TC.EXE 编译:

- <1> Options/Compiler/Model/Large
- <2> Options/Compiler/Optimization/Jump optimization ... On
- <3> Options/Compiler/Code generation/Generate underbars ... Off

(2) 用 TCC.EXE 编译:

```
-ml -o -u- -z -c
```

(3) 连接程序模块:

Turbo Prolog 和 Turbo C 应按下列通用命令行连接:

`tlink init <pOBS> <cOBS> <.sym>,[exe],[map],[usr] + prolog[ +emu +mathl +cl]`

该命令行各个参数的意义如下表所示:

参 数	功 能
tlink	调用TLINK, 即Borland的Turbo连接程序
init	Turbo Prolog的初始化文件
pOBS	Turbo Prolog .OBJ模块
cOBS	Turbo C .OBJ模块
.sym	Turbo Prolog的符号文件须为第一个逗号前的最后一个文件
exe	(可选)可执行文件名
map	(可选)map文件名
usr	(可选)用户库文件列表
prolog	指PROLOG.LIB
emu	(可选)C浮点仿真库EMU.LIB
mathl	(可选)C大模式数学库MATHL.LIB
cl	(可选)C大模式库CL.LIB

假设 PDOUB.PRO 已经编译成.OBJ, 连接 PDOUB.PRO 和 CDOUB.C 的正确命令为:

```
tcc -ml -o -u- -r- -c cdoub
```

```
tlink init pdoub cdoub pdoub.sym,pdoub,,prolog
```

如不用 TLINK, 可创建一个工程定义文件 CPROJ.PRJ, 在工程文件中给出两个模块名:

```
pdoub
cdoub.obj
```

注: 需给 C 目标文件一个扩展名, 如果在 C 文件名后不加扩展名, Turbo Prolog 的编译程序将给出错误信息, 系统不能成功地进行连接。

创建工程定义文件后要做的是将 Prolog 模块说明为工程的一部分, 如果 C 模块已经编译成 CDOUB.OBJ, 只要选择 Compile/EXE file(程序 PDOUB.PRO 必须已装入 Turbo Prolog 编辑器中), Turbo Prolog 系统将自动完成所有连接。如果需要用到外部库文件, 可以在 O/L/Libraries 菜单项中指明。

```
project "cproj"
global predicates
double(integer, integer) - (i,o) language c
goal
double(5,N),
write("5 doubled is ", N).
```

§ 2.1.4 动态存贮分配