

目 录

第一部分 输入技术及实用程序

1.1	Turbo C 多功能控制输入函数	(3)
1.2	可编辑的输入函数	(7)
1.3	磁盘数据的录入与备份程序	(9)
1.4	对 Turbo C 2.0 键盘输入功能的扩充	(11)
1.5	实现中文系统下键盘变速的技巧	(12)
1.6	在程序中转换数据录入方式的程序	(14)
1.7	在 Windows 3.1 中文版中增加五笔字型输入方法	(16)
1.8	PROTEL 电子 CAD 软件包汉字输入的实现及其绘图功能的开发 应用	(18)
1.9	一个鼠标器的应用程序	(20)
1.10	通过安装特殊事件处理程序来扩展鼠标驱动程序的功能	(21)
1.11	以彩色画面为背景的三层菜单生成程序	(27)
1.12	简易实用的中文菜单程序	(30)
1.13	应用软件系统的功能菜单设计及程序示例	(33)

第二部分 图形、图像处理技术及实用程序

2.1	VGA 图形控制器直接编程技术	(39)
2.2	位图的 Super VGA 显示技术及一种位图的 TVGA 方式下快速 显示算法	(44)
2.3	文本方式下的图形显示	(49)
2.4	不同显示卡下正确设置光标类型的程序	(51)
2.5	用 C 语言实现双列目录显示	(53)
2.6	为 Turbo C 增加区域处理函数	(56)
2.7	基于 VGA 图形控制器编程的快速图形保存、恢复与打印	(59)
2.8	用 Turbo C 实现 EGA/VGA 移屏和分屏	(62)
2.9	用 EGA/VGA 寄存器实现屏幕特技	(64)
2.10	屏幕特技显示——虚屏模拟	(66)
2.11	用 TCTOOLS 实现按钮和图形鼠标	(69)
2.12	图形明暗层次自然过渡的处理方法及显示程序	(71)
2.13	屏幕随意作图程序的实现	(72)
2.14	用 Turbo C 编写的股市行情走势图程序	(75)
2.15	美术字和复杂图形的制作及在程序图形界面中的应用	(76)
2.16	在图形编辑中精确定位的“十字架”设计方法	(81)

1.17	一套自成系统的图形截取、编辑系统	(84)
1.18	C 语言 curses 函数的使用方法及程序示例	(91)
2.19	构造 C 语言可变参数个数的函数及应用实例	(95)
2.20	在 XENIX 下利用 CGI 软件包作图的程序	(98)
2.21	VGA 卡的高效模块化图形程序设计及实例	(102)
2.22	在工作站上用 C+XWindow 进行图形程序设计实例	(106)
2.23	用 Turbo C 2.0 实现堆栈式图形画面的存取	(109)
2.24	用 Turbo C 的 Putimage 函数实现动画显示	(111)
2.25	用 Borland C++ 2.0 改进 outtext()和 settextstyle()函数	(112)
2.26	用 C++实现的图形窗口程序包	(115)
2.27	C++中类的概念在建立弹出式图形窗口中的应用	(118)
2.28	C++虚文件数组及应用	(121)
2.29	在 EXE 文件中嵌入图形驱动程序的方法	(123)
2.30	Turbo C++图形函数库的连接	(127)
2.31	用 C 语言快速地给 Auto CAD 点变量赋值	(129)
2.32	用 C 语言直接存取 DWG 文件的方法	(130)
2.33	用 Turbo C(V2.0)编写的“多边形填充”程序	(131)
2.34	Auto CAD 与 Microstation 图形数据交换方法	(135)
2.35	显示速度随机可变、随机切换分屏和打印、可预先指定起始行的一 个有益尝试	(138)
2.36	改变 VGA 的 16 种显示颜色	(140)
2.37	用集群方法进行颜色选择的实例	(142)
2.38	不使用鼠标 ICON 图符用户界面的程序设计	(148)
2.39	一个功能完整实用的肖像图符编辑程序	(152)
2.40	采用聚类方法对彩色图像进行色彩压缩及演示程序	(158)
2.41	C 语言多维动态数组在图像处理与矩阵运算中的应用及实例程序	(163)
2.42	PCX 图像压缩及还原方法	(165)
2.43	用 C 语言实现 PCX 文件向 BMP 文件格式转换	(170)
2.44	VGA 高质量黑白图像的快速显示程序	(174)
2.45	在微机上处理黑白像片的方法	(177)
2.46	推镜头显示效果的实现	(179)
2.47	TIF 图像文件格式分析及显示程序	(181)
2.48	LZW 压缩算法的实现及 GIF 图像显示/压缩存储中的压缩还 原程序	(189)
2.49	TIFF 图像文件简便解读程序	(194)
2.50	BMP 图像文件的格式分析与显示	(197)
2.51	汉字系统下显示 MSP 图像文件	(201)
2.52	UNIX system V 3.2.2 以上版本中图像显示的方法	(203)
2.53	矢量量化彩色图像的快速显示源程序	(207)
2.54	Windows 标像文件的格式和结构以及显示程序	(208)

第三部分 汉字处理技术及实用程序

3.1	C 语言图形状态下的汉字显示	(217)
3.2	在 CEGA 卡的微机上实现图形汉字显示	(218)
3.3	在无中文操作系统支撑下的汉字显示方法	(220)
3.4	能直接在西文 DOS 下显示汉字的实用程序	(223)
3.5	谈西文状态下显示汉字	(225)
3.6	在西文 DOS 下实现汉字的放大显示	(227)
3.7	Turbo C 在非中文系统下显示和放大汉字的实现	(228)
3.8	西文方式下彩色汉字的显示、放大与旋转程序	(230)
3.9	西文 Turbo C 中多种汉字字体字型的平滑显示	(232)
3.10	西文 DOS 下的中西文文本阅读程序	(235)
3.11	在西文状态下汉字标题的制作方法	(238)
3.12	汉字的放大、旋转和倾斜	(241)
3.13	旋转汉字显示的快速算法及 C 程序	(243)
3.14	汉字勾边处理新算法及其实现	(247)
3.15	利用 2.13 高点阵字库实现屏幕汉字放大	(254)
3.16	同 C 语言完成矢量汉字字库数据的读取、分析及汉字显示	(257)
3.17	在汉字系统环境下单字节边框与汉字同屏显示的实现	(260)
3.18	Turbo C 中使用汉字	(263)
3.19	为 C 增加一个显示汉字串的函数	(266)
3.20	基于汉卡的汉字直接写屏技术	(267)
3.21	用 C 语言显示空心汉字的方法	(268)
3.22	图形方式下汉字系统的使用	(270)
3.23	最新立体文件格式分析和调用方法	(271)
3.24	在 C 语言下汉字快速显示及旋转的实现	(277)
3.25	动态字库驻留 VRAM 高端	(278)
3.26	PC SHELL 7.0 中 FNT 字型库的利用	(281)
3.27	WPS 文本页码自动编辑程序	(283)
3.28	忘记密码后金山 WPS 文件的恢复方法	(289)
3.29	自动调整文本文件的 C 程序	(290)
3.30	WPS 硬、软回车换行符之间的相互转换程序	(291)
3.31	将硬回车换成软回车的方法	(292)
3.32	SPT 和 BMP 文件格式分析及其转换实用程序	(293)
3.33	用 Turbo C 2.0 连接文件时常遇到的问题	(297)

第四部分 打印实用程序

4.1	用 C 语言检测打印针的状态	(301)
4.2	用 Turbo C 语言实现屏幕图形压缩存储及打印	(302)
4.3	XENIX 系统下终端打印机的使用	(306)

4.4	彩色图像的伪灰度打印程序的实现	(308)
4.5	在 24 针打印机上打印人物像片的方法	(313)
4.6	Turbo C 屏幕图像的打印	(316)
4.7	条形码及打印程序设计示例	(318)
4.8	Auto CAD 打印机驱动程序的编程方法	(323)
4.9	面向对象报表打印输出仿真	(327)

第五部分 文件查询、删除、拷贝实用程序

5.1	按属性搜索文件的程序	(335)
5.2	找出某类文件并存入 FoxBASE 数据库的方法	(336)
5.3	DOS 目录属性超级管理方法及程序设计	(338)
5.4	超级文件子目录属性查询工具 SATTRIB	(339)
5.5	对 PEL 命令的增强和扩充	(341)
5.6	用 Turbo C 实现文书文件的自由删除	(345)
5.7	全盘查询和删除文件的实用程序	(347)
5.8	短小精悍的磁盘文件粉碎机	(349)
5.9	用 Turbo C 编写的删除目录树程序	(350)
5.10	修改、查看及删除子目录名的程序	(351)
5.11	保密文件的安全删除实用程序	(353)
5.12	物理存储位置不变的文件移动	(356)
5.13	拷贝前的空间测试程序	(357)
5.14	在 XENIX 下磁带备份和恢复的实用程序	(358)
5.15	屏幕像素的块拷贝	(361)
5.16	DOS 下库结构的直接拷贝	(363)
5.17	拷贝特大文件的方法	(364)
5.18	多用户系统中实现设备资源共享	(365)

第六部分 病毒检测、消除实用程序

6.1	流行计算机病毒惯用伎俩综析	(370)
6.2	一个实用的病毒检测程序	(376)
6.3	“5978”病毒及其清除程序	(378)
6.4	“627”病毒的检查及清除	(383)
6.5	“1741”病毒的分析、检测和消除	(385)
6.6	“1741”病毒的检测及其清除	(387)
6.7	“1759”病毒的分析与清除	(389)
6.8	“1465”病毒的特点及清除方法	(394)
6.9	一种新型的计算机病毒——V300E	(397)
6.10	Auto-Copy 病毒的检测与清除	(400)
6.11	CHAIRMAN 病毒及清除程序	(403)
6.12	“新世纪”病毒的清除	(409)

6.13	预防“定时炸弹”病毒的 Turbo C 程序	(413)
6.14	一种新的软件保护技术——程序自杀	(416)
6.15	让硬盘具有自动清除引导型病毒的能力	(417)
6.16	用 Turbo C 编制的能消除十种病毒的程序	(420)
6.17	病毒防御实用程序三例	(440)
6.18	1824 病毒的检测和消除	(444)

第七部分 加解密方法及实用程序

7.1	磁盘中的应用程序防拷贝加密程序	(452)
7.2	用 C 语言的位操作对文件进行加密	(454)
7.3	AMI CMOS 口令的加密原理和解密程序	(455)
7.4	FC-LCK 的加锁原理和修改、恢复口令标志字的程序	(458)
7.5	成批文件的一种简易加密方法	(460)
7.6	在 XENIX 系统任意目录下多个文件的加密	(462)
7.7	一个较完美的硬盘加锁程序	(463)
7.8	DOS 系统下批处理程序的加密程序	(468)
7.9	记录 UNIX/XENIX 系统的“黑名单”	(469)
7.10	对 XENIX 系统关机命令的改进	(472)
7.11	用 C 语言读取磁盘文件位置链表的方法	(473)
7.12	读取 WPS 文件密码的程序	(476)
7.13	找回 WPS 文件中被忘记的密码	(477)
7.14	动态伪随机序列加密法的实现及程序	(478)

第八部分 其它实用程序

8.1	自动计算 Turbo C TSR 程序的驻留长度的方法	(483)
8.2	有选择性的内存清理程序	(485)
8.3	内存文件恢复简法	(486)
8.4	CMOS 内容的保存与恢复	(487)
8.5	C 程序利用 BIOS 访问扩展内存的方法	(489)
8.6	扩展内存与扩充内存的比较及使用	(492)
8.7	面向对象的程序设计实例	(504)
8.8	Borland C++ 的可动态派遣虚拟表的应用	(507)
8.9	基于 C++ 的包容类库的实现	(509)
8.10	用 C 语言实现 dBASE III (FoxBASE) 数据向 ORACLE 的自动转换	(512)
8.11	FoxBASE 与 ORACLE 之间数据转载的构思及程序实现	(515)
8.12	Borland C++ 调用汇编语言的方法和例程	(521)
8.13	将小写字母转换成大写字母的一个实用程序	(528)
8.14	数字金额转换为大写金额的 C 程序	(529)
8.15	EXE2BIN 命令递过程的实现	(531)

8.16	用 C 语言的函数指针实现宏命令的方法	(533)
8.17	进入子目录的实用程序	(536)
8.18	对 C 语言的 FILE 结构剖析	(538)
8.19	对 Turbo C 与流相联的缓冲区的几个附注	(542)
8.20	伪变量的应用实例	(546)
8.21	简化型 OCI 的设计与实现	(550)
8.22	用 OWL 开发 Windows 应用程序的方法	(552)
8.23	用 Turbo C 编写 BASIC 串操作函数	(556)
8.24	一种后台驱动程序	(559)
8.25	用 Turbo C 编写的利用欧几里得算法判断 d_1, d_2 是否互质及最大 公约数的程序	(560)
8.26	改进多用户系统中的 DOS 盘操作命令	(561)
8.27	FoxBASE+源程序结构检测程序 Fox-CHECK	(563)
8.28	FoxBASE 数据库文件完整性诊断及程序设计	(566)
8.29	用 TurboC 语言直接实现数据库及其管理	(569)
8.30	用 Turbo C 操作 BDF 数据库的方法与程序	(572)
8.31	为 FoxBASE 系统增加一个实用程序过程分解器	(578)
8.32	面向文件的 DBMS 设计性能的测试方法及程序设计	(580)
8.33	生成过程文件的实用程序	(584)
8.34	清除旧记录压缩数据库文件的一种方法	(585)
8.35	PC 机和 FOXBORO 控制器间的通信	(588)
8.36	一种美化 C++ 程序的方法	(592)
8.37	硬盘维护工具 DISKBR 原理及实用程序	(598)
8.38	IDE 硬盘参数的测定	(603)
8.39	测定硬盘柱面数、磁头数、每磁道扇区数的 C 程序	(608)
8.40	UNIX/XENIX C 语言程序的调整方法	(610)
8.41	反编译 FoxBASE+ 伪编译文件的源程序	(613)

前 言

C 语言以简洁的表达方式,高级语言的编程形式,低级语言的操作方式,具有灵活性强、移植性好、目标代码质量高和丰富的函数调用,为设计高质量的、人机界面友好的程序提供了理想的工具。所以愈来愈受到程序开发者的青睐。其应用范围日趋广泛,应用水平不断提高,应用成果、开发经验、编程技巧和实用程序层出不穷。

为此,我们从近期众多的 C 语言应用开发经验中精选出 200 例,经加工、分类,汇编成册,供广大计算机用户,特别是 C 语言编程者、爱好者学习选用。本书虽无全面的技术理论论述,但一事一议、短小精悍,实用价值大。初学者可即取即用,或稍加修改即可用,节约开发时间,提高工作效率。有经验的设计者也能从中获取有益的启示,给自己的思路打开联想的天窗,从而编制出更完美的程序。

全书分为八部分,内容包括输入技术和实用程序;图形、图像处理技术及实用程序;汉字处理及实用程序;打印实用程序;文件查询、删除、拷贝实用程序;病毒检测、消除实用程序;加解密方法及实用程序;还有其它众多方面的实用程序。

由于有些文章的内容本身具有多重属性,只能按其主要属性予以归类。另外作者众多,水平经验有差别,文章风格各异,则便于交流经验、开拓思路、采众家之长,达到推陈出新之目的。

参加编审工作的还有金秋、海燕、乔晖、林海、靖涛。同时每篇文章都署有原作者。

编 者

1994 年 6 月

第一部分

输入技术及实用程序

1.1 Turbo C 多功能控制输入函数

戴林清

笔者为 Turbo C 编写了一个多功能控制输入函数 `get()`。该函数具有以下功能：

1. 良好的编辑界面,显示数据初值,使用←和→键灵活改变光标位置,退格键和 Del 键删除左边或当前字符,Ins 键随时作插入和修改切换。

2. 较强的控制能力,可使用字符检测、集合检测以及格式修饰功能,在字符检测下滤去过所有非法字符,在集合检测下识别输入字符的类型,在格式修饰下自动生成格式符,编辑时不予干扰,同时对输入进行逐位集合检测。

3. 配备了全屏幕编辑基础条件,使用光标键及有关控制键可实时结束编辑,函数返回中断键值,在此基础上稍作加工就能实现全屏幕编辑。

4. 杜绝了半个汉字。

以下是函数的调用方式:

```
int get (int r,int c,char *s,int l,char *ss);
```

各参数意义见源程序中注释。需要说明的是控制串 SS,根据 SS 首字符的取值不同,编辑将采取不同的控制方式。

1. 首字母为“&”时,为字符检测方式。“&”后的所有字符组成合法字符集,编辑时只有合法字符集中的字符才被接收,其余皆视为非法。

2. 首字母为“@”时,为集合检测方式。“@”后的那个字符为集合符,其取值范围及意义如下:

A—只允许字母 N—允许字母和数值

9—只允许数字 X—允许所有字符

编辑时将作类型判别,与集合符意义不符的键值将不予理睬。

3. 首字母非“&”或“@”时,为格式修饰方式。此时 SS 为格式串,其长度应与编辑数据长度相等,不等时程序会自动补齐或截尾。格式串中字符可取任意字符,并约定:如果格式串中某位不是 2 中所述集合符,则该位字符称为格式符,表示被编辑数据的对应位字符只能取该格式符,并且无须编辑,程序在编辑前会自动生成该位字符,而在以后的编辑中总是跳过它;如果格式串中某位是上述集合符,表示被编辑数据对应位字符需要编辑,且取值只能取相应类型的字符。

格式修饰是程序设计的关键,程序中采用连续结构链来记录被格式符分割的各个编辑区间的首尾位置,在编辑过程中随时记录光标所在区间,在此基础上完成光标在各编辑区间的跳动,并使删除等操作的作用范围限定在本区间内。

读者只需将所列函数文件编译后,用库管理程序加入相应库,就能象调用其它 C 函数一

样直接调用函数 get(), 以下是调用实例。

```
#include<stdio.h>
#include<conio.h>
int get (int r, int c, char * s, int l, char * ss);
main()
{ char s1[9], s3[9], s3[9];
  sl[8]=0; memset(s1, '\'', 8);
  strcpy(s2, s1); strcpy(s3, s1);
  textattr(15); clrscr();
  directvideo=0; /* 取消直接写视频 */
  get(4, 9, s1, 8, "8abede"); /* 字符检测 */
  get (5, 9, s2, 8, "@N"); /* 集合检测 */
  get(6, 9, s3, 8, "99/99/99"); /* 格式修饰 */
}
```

本文所述 get() 函数只能直接编辑字符型数据, 对数值型数据则需在编辑前用有关函数转换成字符串, 构造相应合法字符集, 使用字符检测方式进行编辑, 编辑完毕再转换成原来类型的数值。此外, 表面上用 get() 编辑时, 编辑区域不能超出一行, 但只要对参数 S 和 l 进行合理组合, 即可达成多行编辑效果。下例是对长 150 的字符串 t 进行三行编辑:

```
get(4, 9, t, 50, ""); get (, 5, 9, &t[50], 50, "");
get(6, 9, &t[100], 50, "");
源程序在 Turbo C2.0 下编译通过, 少量改动即可移植到 C 的其它产品下。
```

源程序清单:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<bios.h>
#include<ctype.h>
#define HOME 327 /* Home 键 */
#define END 335 /* End 键 */
#define LEFT 331 /* ← 键 */
#define RIGHT 333 /* → 键 */
#define INS 338 /* Ins 键 */
#define DEL 339 /* Del 键 */
#define UP 328 /* ↑ 键 */
#define DOWN 336 /* ↓ 键 */
#define ESC 27 /* Esc 键 */
#define ENTER 13 /* 回车键 */
#define BACK 8 /* 退格键 */
typedef struct /* 区间结构 */
{ int i, lo, hi;
  int e; /* 本区间起始位置 */
  int e; /* 本区间终止位置 */
} inv;
int inkey(void) /* 接收键值函数 */
{ int i, lo, hi;
  i=bioskey(0); /* 接收键值 */
  lo=i&0x00ff; /* 低字节 */
  hi=(i&0xff00)>>8; /* 高字节 */
  return((lo-0)? (hi+256):lo);
}
int get (int r, int c, char * s, int l, char * ss)
/* r: 行; c: 列; s: 数据串; l: 长; ss: 控制串 */
{ inv * b; /* 连续结构 */
  int n, i, j, k, bno, bmax;
  int p=0; /* 相对编辑位置 */
  int ins=0; /* 插入状态标志 */
  int chan=0; /* 消除半个汉字标记 */
  int loop=1; /* 继续编辑标志 */
  unsigned char * t; /* 编辑暂存区 */
  unsigned char cc; /* 接收后半半个汉字 */
  char * set; /* 指向控制区 */
  char f=1; /* 为 1 表示 set 为格式串 */
  unsigned char u;
  int first, end; /* 可编辑字符头尾序号 */
  if (ss[0]!='&') { strlen(ss);
    else i=1+1;
    set=(char *) malloc(i); /* 申请控制串空间 */
    if (! set) return 0;
    set [i-1]=0; /* 控制区终止符 */
    if(ss[0]!='&') /* 为字符检测时 */
      memcpy(set, &ss[1], i-1); /* 合法字符集 */
  }
}
```

```

f=0; } /*置合法字符集标志*/
else if (ss[0]!='@'){ /*为集合检测时*/
u=toupper(ss[1]); /*集合符转大写*/
memset(set,u,1); /*格式串各位赋值*/
else if (ss[0] != 0) { /*为格式修饰时*/
memcpy(set,ss,1); /*直接拷贝至set*/
i=strlen(ss) /*长度不够时反面以X补齐*/
if(i<1)memset(&set[i],'X',1-i);
else memset(set,'X',1); /*ss为空串时*/
if (f){ /*有格式串时求各区间首尾序号*/
if(strchr("ANX9",set[0])){j=1;k=1;
else{j=0;k=0}
i=1;
do {
if (j)
while (i<1&&strchr("ANX9",set[i++]);
else
while (i<1&&1 strchr("ANX9",set[i++]);
if (i==1)break;
if(i) j++;
} while(1);
if(i) k) return 0; /*不存在可编辑区间时*/
b=(inve *)malloc(k * sizeof(inve));
if(! b)return 0;
bmax=k-1; /*最大区间序号(从0起算)*/
i=0;k=0;
while(i<1&&1 strchr("ANX9",set[i]))i++;
j=1;
do{
if(j){
(b+k)->f=i;
while(i<1&&1 strchr("ANX9",
set[i]))i++;
(b+k)->e=i-1,k++;
else while(i<1&&1 strchr("ANX9",
set[i]))i++;
if (i==1)break;
j++;
}while(1);
}
else{
b=(inve *)malloc (sizeof(inve));
if (! b)return 0;
bmax=0;b->f=0;b->e=1-1;
first=b->f; /*第一个可编辑字符*/
end=(b+bmax)->e; /*最后一个可编辑字符*/
p=first;bno=0; /*当前编辑位置所在区间号*/
/*
t=(unsigned char *)malloc(1+1);
if (! t)return 0;
t[1]='\0'; memcpy(t,s);
if (f)for (i=0;i<1;i++)
if (! strchr("ANX9",set[i]))
t[i]=set[i];
gotoxy(c,r);
for(i=0;i<1;i++)putch(t[i]);
do {
chan=0;
gotoxy(c+p,r) /*光标定位到当前编辑位置*/
switch(n=inkey()){ /*接收键值并处理*/
case HOME: /*Home到首*/
if(p==first)loop=0 /*已为首则退出*/
/*
p=first;bno=0;break;
case END: /*End到尾位置*/
if (p==end)loop=0 /*已为尾则退出*/
/*
p=end;bno=bmax;break;
case INS:ins=! ins;break; /*Ins键切换*/
/*
case LEFT: /*←键左移*/
if(p==first){loop=0;break;}
if(p>(b+bno)->f){p--;
if (t[p]>0xa0)p--; /*
汉字要左移2格*/
else {__bno--; p=(b+bno)->e;
if (t[p]>0xa0)p--; /*
移到前半汉字处*/
}
break;
case RIGHT: /*→键右移*/
if(p==end){loop=0;break;}
if(p<(b+bno)->e){p++;
if(t[(p-1)>0xa0)p++;
/*汉字要右移2格*/
else {bno++; p=(b+bno)->f;

```

```

break;
case BACK: /* 退格键 */
if (p==first) { /* 已在首位时退出编辑 */
* /
loop=0;in=LEFT;break;}
if (p>(b+bno)->f){i=1;
if (t[p-1]0xa0)i++; /* 汉字要退 2 格 */
* /
memmove (&t[p-1],&t[p],(b+bno)->
e-p+1);
memset (&t[(b+bno)->e-i+1],'',i);
p--i; /* 整体前移并调整编辑位置 */
gotoxy(c+p,r); /* 重显更新部分 */
for (i=p;i<=(b+bno)->e;
i++)putch(t[i]);
}
else {bno--; p=(b+bno)->e;
if (t[p]>0xa0)p--; /* 移到前半汉字处 */
* /
break;
case DEL: /* 删除光标处字符 */
if (p==(b+bno)->e){t[p]='';putch
('');}
else {i=1;
if (t[p]>0xa0)i++ /* 汉字要删 2 格 */
memmove (&t[p],&t[p+1],(b+bno)->e
-p-i+1);
memset (&t[(b+bno)->e-i+1],'',);
for (i=p;i<=(b+bno)->e; i++)
putch(t[i]);
}
break;
case UP: /* 上移键 ↑ */
case DOWN: /* 下移键 ↓ */
case ESC: /* Esc 放弃编辑内容 */
case ENTER: loop=0;break; /* 回车结束 */
* /
default;
if(n>255)continue; /* 不睬无效键 */
u=(unsigned char)n;
if (t[i]&0x1f stchr (set,u)) continue;
else if (t switch (set[p]) {
case 'A': if (t isalpha (u)) continue;
break;

```

```

case 'N': if (t isalnum(u)) continue;
break;
case '9': if (t isdigit(u)) continue;
break; }
i=1;
if(u>0xa0){ /* 是汉字时 */
if (p==(b+bno)->e||
(is&0x01&t[p+1] != 'X'))
continue; /* 末尾 1 位无法输 */
cc=(unsigned char)inkey();
i++; /* 继续接收后半 */
if ((t[ins]&t[p]<0xa0&t[p+1]>0xa0){
memmove (&t[p+2],&t[p+1],(b+
bno)->e-p-1);
chan=1;}
} /* 在汉字处输入字符要删除后半汉字 */
else if ((t[ins]&t[p]>0xa0){
memcpy (&t[p+1],&t[p+2],(b+bno)
->e-p-1);
t[(b+bno)->e]='';chan=1;}
if (ins) /* 插入时后部后移 */
memmove (&t[p+1],&t[p],(b+bno)->e
-p-i+1);
t[p]=u; /* 存放字符 */
if (i>1)t[p+1]=cc; /* 保存后半汉字 */
j=0; /* 统计汉字占位数以消除半个汉字 */
* /
for (k=p;k<=(b+bno)->e;k++)
if (t[k]>0xa0)j++;
if (j%2) t [(b+bno) -> e] = '';
* / 清尾半个 */
if (ins||chan k=(b+bno)->e;
else k=p-i-1;
gotoxy(c+p,r); /* 重显 */
for (j=p;j<=k;j++)putch(t[j]);
p+=i; /* 改变当前编辑位置 */
if (p>end) {loop=0;c=ENTER;}
* / 编辑完 */
else if (p>(b+bno)->e){
bno++; p=(b+bno)->f;
break;
}
} while (loop);
gotoxy(c,r);

```

```

if (n! =ESC) memcpy(s,t,l); /* 认可编辑
*/
else for (i=0;i<1;i++) putch(s[i]);
}
free(t); free(set); /* 释放暂存区 */
return n; /* 返回中断编辑的键值 */

```

1.2 可编辑的输入函数

付 辉

笔者在用 TURBO C 2.0 编程时,深感 C 所提供的标准输入函数的不完善。表现在无编辑能力、控制能力弱、界面差。为此笔者编制了输入函数(int get (int x, int y, int attr, int len, char * old str),较好地解决了这一问题。使用函数 get(),能按指定的属性和字段长度进行输入。具有仿全屏编辑功能,越界报警。能够在汉字系统下随意录入、删除、插入汉字。具有消除“半个汉字”的功能,键入回车键确认所得、键入 ESC 键保留默认字符串。函数的返回值为中断编辑的键值。

例如:

```

...
char * str = “可编辑的输入函数实现”;
...
get (10, 10, BLUE * 16+RED, 30,str);
...

```

运行时,以指定属性(蓝底红色)在(10,10)处,显示字符串“可编辑的输入函数实现”,长度 30,不足处以空格补足。可运用全屏编辑功能对其操作。ENTER 确认,ESC 取消操作。

此函数在浪潮 286—12 上运行通过,有兴趣的同行不妨一试。

程序清单:

```

#include<conio.h>
#include<mem.h>
#include<alloc.h>
#define HOME 327
#define END 335
#define LEFT 331
#define RIGHT 333
#define INS 338
#define DEL 339
#define ESC 27
#define ENTER 13
#define BACK 8
int get (int x, int y, int attr, int len, char *
oldstr)/ * oldstr 缺省串 */
(unsigned char * newstr; /* 编辑串 */
unsigned char cc, uu;
int oldlen, loop=1, n, p=0, i, ins=1,j,k; /
* p 当前编辑位置 ins 插入标志 */
newstr=(unsigned char *) malloc (len+1);
oldlen=strlen(oldstr);
memcpy(newstr, oldstr, oldlen);
memset (newstr+oldlen, '', len-oldlen);
* (newstr+oldlen) = 0; /* 复制缺省串入编辑串
*/
textattr (attr); /* 指定字符串显示属性 */
gotoxy (x,y); /* 指定显示位置 */
cprintf ("%s", newstr);
if (oldlen! =0 _P=oldlen-1 /* 如果缺省为空
串,校验编辑位置 */
else p=0

```

```

gotoxy(x+p,y);
do
{
    n=inkey();
    switch(n) {
        case HOME: p=0; gotoxy(x,y); break;
        /* 返回首位置 */
        case END: p=len-1; gotoxy(x+len,y);
        break; /* 返回尾位置 */
        case LEFT:
            if (p<=0) printf(" \007");
            else {p--;
                if (p!=0)
                    if (* (newstr+p)<0xa0) p--;
                /* 汉字左移两位 */
                gotoxy(x+p,y);
            }
            break;
        case RIGHT:
            if (p>=len-1) printf(" \007");
            /* 越界警报 */
            else {p++;
                if (p!=len-1)
                    if (* (newstr+p)>0xa0) p++;
                gotoxy(x+p,y); /* 汉字右移两位 */
            }
            break;
        case BACK:
            if (p<=0) printf(" \007");
            else
                {i=1;
                    if ((p-i)!=0)
                        if (* (newstr+p-1)>0xa0) i++;
                    /* 汉字退两格 */
                    memmove(newstr+p-i, newstr+p, len-
                        p);
                    memset(newstr+len-i, ' ', i);
                    p-=i;
                    gotoxy(x,y);
                    printf("%s", newstr);
                    gotoxy(x+p,y);
                }
            break;
        case DEL:
            if (p>len-1) printf(" \007");
            else { if (p==len-1) * (newstr+len-
                -1)=' ';
                else {i=1;

```

```

                    if ((p-i)!=0)
                        if (* (newstr+p)>0xa0)
                            i++; /* 汉字时删两位 */
                    memmove(newstr+p, newstr+p+
                        i, len-p);
                    memset(newstr+len-i, ' ', i);
                    gotoxy(x,y);
                    printf("%s", newstr);
                    gotoxy(x+p,y);
                }
            break;
        case INS: ins=1; break;
        case ESC:
        case ENTER: loop=0; break;
        default:
            if (n>255) continue;
            i=1; uu= (unsigned char)n;
            if (n>0xa0)
                {
                    cc= (unsigned char) inkey();
                    i++;
                    if (! ins && * (newstr+p) <0xa0
                        && * (newstr+p+1) >0xa0)
                        memmove(newstr+p+2, newstr+p+1,
                            1);
                    else if (! ins && * (newstr+p) >
                        0xa0)
                        {memcpy(newstr+p+1, newstr
                            +p+2, len-p);
                            * (newstr+len-1)=' '; }
                    if (ins)
                        memmove((newstr+p+i),
                            (newstr+p), (len-p));
                        * (newstr+len)=0;
                        * (newstr+p)=uu;
                        if (i>1) * (newstr+p+1)=cc;
                        j=0;
                        for (k=p; k<len; k++)
                            if (( * (newstr+k)>0xa0) j++;
                            if (j%2) * (newstr+len-1)=' ';
                        /* 删除半个汉字 */
                        gotoxy(x,y);
                        printf("%s", newstr);
                        gotoxy(x+p,y);
                        p+=i;
                        if (p>=len) {printf(" \007"); loop=

```

```

0; }
break; }
}while (loop);
if (n==ENTER) strcpy (oldstr, newstr);
/* 只有回车时才确认编辑成功 */
free (newstr);
return(n);

int inkey (void)
{ int ch, low, hig;
ch=bikey (0);
low =ch & 0x00ff;
hig =(ch& 0xff00)>>8;
return (( low ==0)? hig+256,low);
}

```

1.3 磁盘数据的录入与备份程序

赵广慧 孙海荣 张 莉

XENIX 多用户操作系统下的 INFORMIX—ESQL/C, 作为关系数据库嵌套 C 语言结构的系统软件工具, 不仅对软件设计编程十分方便实用, 而且具有强大的数据处理能力, 对数据库的操纵既灵活又快捷, 很受软件编程人员的喜爱。笔者利用这套工具编制的报表管理软件, 整体结构紧凑, 其中的磁盘数据录入与备份两个功能, 操作使用简便, 对于报表数据的传递、汇总起到了省时省力、准确可靠的良好作用。我们愿提供这两段程序的源码, 希望能够在此与同行们交流切磋技术。本程序的编译方法在此就不介绍了, 头文件、主控函数及一些防错排错处理都做了简略。程序清单:

```

/*
INFORMTX _ESQL/C
DLJH BBGL, EC
*/
bbgl_clr()
{int _hh, xhh;
char fm [5], name1 [22], hcop [100], hcv2 [50],
name [22];
struct stat buf;
int x=10,y=8;
strcpy(fname, "bbgl_clr");
strcpy (fm, "zjphh");
strcpy(name1, "/usr/tmp/lzjphh");
plot(x,y,"下级行号: [ ] [ ]");
loop1: readint (x+14,y,& xhh,3,3);

¥select fs2hm into ¥ ts2rec. fs2hm from fs2
where fs2hh=¥xhh;
if (sqlca.sqlcode==
SQL NOTFOUND)
{ plot (x+20,y+7, "此行号不存在,重
输!!");
beep(5);
getchar ();
plot (x+20,y+7," ");
goto loop1; }
plot (x+26,y, ts2rec. fs2hm);
hh=xhh;
if (hh<10)
sprintf (name, "/usr/tmp/%s0%1d", fm, hh);
if (hh <100 && hh>=10)
sprintf (name, " /usr /tmp /%s0%1d", fm, hh);
sprintf (hcv2, "tar xv2 %s 2>/dev/null 1>/dev/
null", name);
sprintf (hcop, "cp %s %s 2>/dev/null 1>/dev/
null", name, name1);
plot(20,11,"请插入该行的报表盘,按任一键继
续...");
getchar();
system(hcv2);
plot (20,13,"正在运行请稍候.....");
}

```

```

system ("cp/dev/null/tmp/zjerr");
system(hcop);
¥delete from trd1 where frd1hh=¥xhh;
system ("dhload-d bbg1-c/usr/bin/load_dfm\
-1/tmp/zjerr 1>dev/null 2>/dev/null");
¥update frd1 set frd1hh=¥xhh
where frd1hh=99;
'plot (30,16," 磁盘完毕!");
hh=xhh;
getchar();
return; }
bbg1_hbf()
{¥int hh, xhh;
char fm[5], name1[22], sev2[50];
int x=10, y=8;
strcpy (fname, "bbg1_hbf");
strcpy(fm, "zjphh");
plot(x,y,"行号: [ ] [ ]");
loop: readint (x+14,y,8*xhh,3,3);
if (trd1rec. frd1hh==0)
{plot (x+26,y,"本行汇总表"),FFSTD;
¥select fc1bh into ¥felrec. fc1bh
from fc1;
if (sqlca. sqlcode<0)
warning (sqlca. sqlcode, pname, mname, fname,
"select fc1");
hh=felrec. fc1bh;
if (xhh==999)
{plot(x+26,y,"总行报表");
hh=xhh;
if (xhh!=999 && xhh=0)
{¥select fs2hm into ¥ts2rec. fs2hm from fs2
where fs2hh=¥xhh;
if (sqlca. sqlcode ==SQLNOTFOUND)
{plot (x+20, y+7, "此行号不存在!");

```

```

beep (5);
getchar ();
plot (x+20,y+7," ");
goto loop; }
plot (x+26,y,ts2rec,fs2hm);
hh=xhh; }
if (hh<10)
sprintf (name1, "/usr/tmp/%s01d", fm, hh);
if (hh<100 && hh>=10)
sprintf (name1, "/usr/tmp/%s%2d", fm, hh);
plot (20,10,"生成文本文件...");
¥update frd1 set frd1hh=99 where frd1hh= $
xhh;
unload ("select frd1sxh, frd1kmb, frd1qcj, frd1qcd,
frd1j, frd1d, / frd1zhbh, frd1hh from frd1 where frd1-
hh=99", name1);
¥update frd1 set frd1hh=¥xhh where frd1hh=99;
if (sqlca. sqlcode<0)
warning (sqlca. sqlcode, pname, mname,
fname," UPDATE FRD1");
scpp (31,11); printf (" % .21s", name1);
plot (20,13,"插入高密度软盘,按任一键拷贝");
wgetch();
sprintf (scv2,"tar cv2 %s 2>/dev/null", name1);
flag=system(scv2);
if (flag)
plot(20,14,"软盘写错误,请换盘重来!");
else
{plot (20,14,"拷贝成功!,按任意键返回");
getchar();
return; } }
load_dfm
FILE "/usr/tmp/lzjphh" DRLTMITER" | "8;
insert into frd1
values (f01,f02,f03,f04,f05,f06,f07,f08);

```