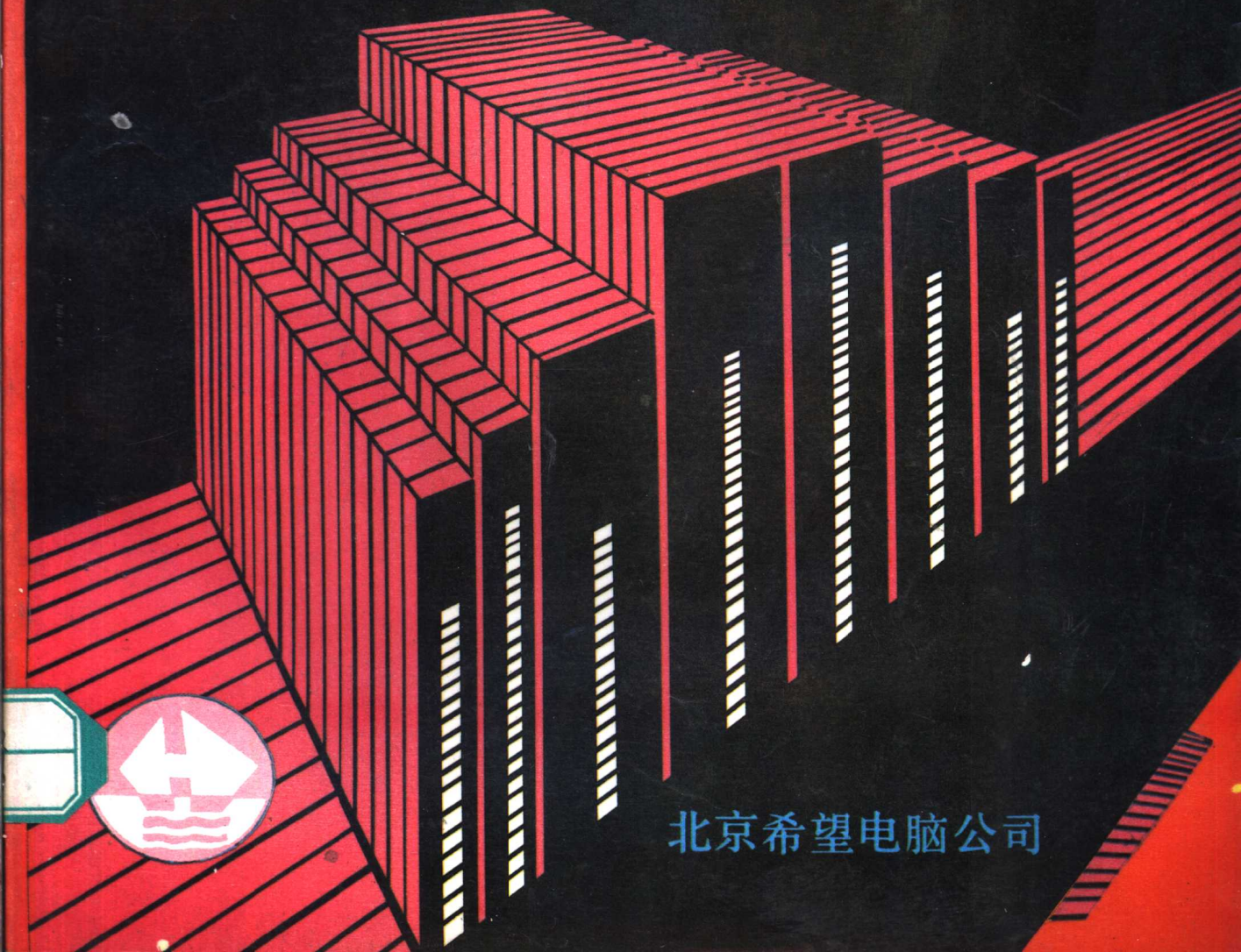
HOPE

4·20

C语言数据库函数库

Code Base 用户指南



北京希望电脑公司

C语言数据库函数库

Code Base 用户指南

（V4.20版）

张成龙 丁素文 译　　王德华 校

北京希望电脑公司

一九九一年七月

译 者 序

数据库是目前应用最为广泛的信息处理技术。在我国，使用最多的数据库管理系统是dBASE Ⅲ关系数据库，广大用户利用dBASE Ⅲ开发出了许多信息管理系统。但在使用dBASE Ⅲ进行实际系统开发时，我们不难发现，dBASE Ⅲ的速度慢、代码效率低、灵活性差，不利于作进一步的开发。

为了克服以上不足，开发出更方便、高效的管理信息系统，许多用户希望改用其他数据库甚至高级语言，但是又存在两个方面的顾虑：一是原来用dBASE Ⅲ积累多年的数据是否可以继续使用；二是原来开发应用程序的思想、乃至程序是否可以加以利用。

在这里，我们特别向这部分高级用户推荐美国Sequiter软件公司推出的C语言数据库函数库——CodeBase。

Code Base是一个面向数据库和屏幕管理的C语言函数库。Code Base与dBASE兼容，它实现了dBASE Ⅲ的全部功能，并可以使用与dBASE兼容的其它任何产品。此外，Code Base还支持多用户。

Code Base可以直接存取dBASE、Clipper或Foxbase的数据库文件，因此，原来的数据库文件均可保留。由于Code Base与dBASE兼容，所以，原来用dBASE Ⅲ开发应用程序的思想仍可以采用。

Code Base完整地实现了数据库管理功能，包括记录的追加、插入和删除，字段的定义、存取和替换，过滤操作，文件的打开、关闭与建立，数据库和索引文件的锁定与解锁，MEM型文件处理，以及数据库的整理、扩展和索引等。此外，Code Base还提供了强大的屏幕管理功能，包括窗口、菜单和读取操作。

目前，Code Base直接支持Turbo C、Quick C、Microsoft C、Zortech C++，Watcom C386和Sco Unix或Xenix操作系统。由于Code Base提供完整的源代码和测试代码，用户可以自己建库，经调试使其正常工作，因此，Code Base也可用于其它编译器或操作系统。

Code Base使用的命名约定与dBASE相同，因而易于掌握。任何人，只要熟悉C程序设计语言并了解一些基本的数据库知识，学习Code Base都不会感到困难。

本书是《C语言数据库函数库Code Base用户指南》。书中讲述了数据库的基本概念，详细说明了Code Base的安装、调试和使用方法。本书还分类列出并描述了Code Base所提供的二百多个函数，对于每个例子均给出了示范程序。

本书的出版得到了北京希望电脑公司资料部秦人华经理的大力支持和帮助，承蒙北京计算机学院王德华老师审阅了全部译稿，在此，一并向他们表示诚挚的谢意。

译者于北京

一九九一年七月

目　录

第一章　Code Base 概述

Code Base 是一个面向数据库和屏幕管理的 C 语言函数库，Code Base 与 dBASE 的兼容性使它可以用于任何其它与 dBASE 兼容的产品。

Code Base 使用与 dBASE 相同的命名约定，因而易于掌握。任何人，只要熟悉 C 程序设计语言并了解一些基本的数据库知识，学习 Code Base 都不会感到困难。

所有学习 Code Base 的用户都应先阅读本章。快速参考部分将告诉你哪些例程是最急需的。第二章"例程详解"中各例程前的说明非常重要，我们建议您在使用例程前应先阅读相应的说明。

打算在多用户环境下使用 Code Base 的用户需要阅读，"关于多用户"一章。

本书重点讨论了 Database(数据库)，Field(字段)，index File(索引文件)，windowing(窗口)，Menuning(菜单)和 get(读取)等例程。这些例程是开发数据库应用程序最重要的例程。

Browse 例程教你建立用户浏览和屏幕编辑，因而在此之前又需要首先学会 get 例程和一些 windowing 例程。

对关系和过滤感兴趣的读者不妨参考"扩展例程"。

1.1　基本术语

下面是一些 Code Base 手册中所用术语的定义。

数据库

数据库是一个数据文件。例如，通讯录数据库包含了一些人的姓名和地址。

记录

数据库信息由记录构成。在通讯录数据库中，每个记录包含一个人的信息。

记录缓冲区

对每个数据库，Code Base 专门为一个记录的信息分配一个存储区，称为记录缓冲区。许多高级 Code Base 例程，例如 d4go 自动使用这些存储缓冲区。记录缓冲区的指针可用 f4record 例程获得。Code Base 内部代码描述了记录缓冲区的格式。

记录号

数据库中的每个记录都被赋予一个从 1 开始的顺序记录号。当前记录号对应于记录缓冲区中的记录。当前记录号可用"d4recno"例程获得。

字段

记录由字段组成，一个数据库字段描述每个数据库记录的一个信息类。通讯录数库中的字段可以是：姓名、住址、城市和电话号码。例如，NAME、ADDRESS、CITY 和 PHONE_NUM 是对应的字段名。

字段可以是字符型、数字型、浮点型、逻辑型、日期型或存储型。字段的类型很重要，因为它确定了字段所存信息的种类，字段名和字段类型是在创建数据库生成时定义的。

索引文件

在不同的时刻可能需要按不同的方式查询数据库，例如，通讯录数据库有时要求按 LAST_NAME(姓)字段排序，有时又要求按 ADDRESS(地址)字段排序。如果每提出一种新

的要求都重新排一次序，这对大型数据库需要很长时间。

因此，有必要建立索引文件，永久地保存排好序的信息。索引文件为一个数据库保存一种排序结果。索引文件采用 B+ 树结构，从而能够快速有效地查找排序信息。

索引表达式

索引文件还含有表明数据按什么要求排序的信息，称为索引表达式。例如，LAST_NAME 是一个表达式，表明该索引文件是按 LAST_ NAME 排序的。

引用号

Code Base 几乎处处都要用到引用号，每个 Code Base 使用的数据库、索引文件、字段、入口、窗口、菜单项都有一个对应的引用号。引用号是许多 Code Base 例程的参数。

1.2 数据库的基本概念

下面介绍了一些本手册涉及的数据库基本概念，其余假定读者已经熟悉。

使用 Code Base 可能要保存和存取多达几百兆字节的信息，这些信息以数据库文件的形式保存在磁盘上，每一时刻，全部有效信息中只有一部分被处理。

几个数据库文件可能同时打开，但某一时刻仅有一个被选中，数据库例程仅作用于这个被选文件。打开、关闭数据库要占用许多机时，而在打开的数据库中选择一个数据库则可在瞬间完成。

每个打开的数据库都有一个称为记录缓冲区的内存区。记录缓冲区的精确格式在附录中说明，即在 Code Base 内部代码一节中描述，许多高级数据库管理例程都使用记录缓冲区。例如，例程 d4go 把一个数据库记录从磁盘读到记录缓冲区，类似地，例程 d4write 和 d4append 直接把数据库缓冲区的内容写到数据库文件中去。

存取或修改记录缓冲区应使用字段例程。用字段例程修改记录缓冲区时，所作的修改自动写到磁盘中。

每个数据库可能有一组打开的索引文件，每个索引文件对应于一个特定的数据库。数据库被修改后，相关的索引文件自动被修改，每个数据都可以有自己的“被选”索引文件。高级数据库管理例程，例如 d4seek 和 d4skip 使用属于当前被选数据库的当前被选索引文件。低级索引文件例程，例如 i4seek 和 i4skip 一般不直接使用。

存储文件用于存储可变长的正文，为提高效率，每个字段在数据库记录中占据固定空间。一个存储型字段在数据库记录中占 10 个字节，但在存储文件中占有可变的存储空间。除了文件扩展名以 DBF 变为 DBT，存储文件与相应的数据库文件同名，一个特定的数据库文件的所有字段使用同一个存储文件。

1.3 启动

第一次启动 Code Base 时，需要进行以下工作：

1. 制作工作盘；
2. 仔细阅读 README 文件，以获得最新信息；
3. 阅读“编译器说明”(下节)；
4. 安装 Code Base；
5. 运行 d4learn，以检验 Code Base 安装情况。阅读文档并参阅相应的示范程序。

6. 阅读“创建 Code Base 应用程序”一节。

7. 用 Code Base 编程，如遇到问题，可参阅“常见问题”一节。

1.3.1 编译器说明

目前，Code Base 直接支持 Turbo C、Quick C、Microsoft C、Zortech C++、Watcom C386 和 SCO Unix 或 Xenix 操作系统。当使用其它编译器或操作系统时，由于 Code Base 提供完整的源代码和测试代码，因此，用户可以自己建库，经调试使其正常工作。测试代码放在磁盘目录\TEST 中，有关文件请参阅\TEST\TEST. DOC。

同时提供的还有两个预先建立的库 T4. LIB 和 M4. LIB，分别用于 Turbo C 和 Microsoft C。M4. LIB 还可以用于 Quick C 2.0 或更高版本。这些库都是用大内存模式建立的，使用 Code Base 屏幕管理，dBASE 的索引文件和 DOS 操作系统进行浮点仿真。如果用户采用不同的编译程序、操作系统或需要不同配置，请查阅附录 D。

1.3.2 Turbo C 环境下使用 Code Base

为了在 Turbo C 交互环境下使 Code Base 库，需要一个项目文件。D4LEARN. PRJ 是一个 project 样本文件，用于创建程序 D4LEARN. EXE。D4LEARN. PRJ 文件包含下面两行信息：

```
D4LEARN. C
T4. LIB
```

Turbo C 栈容量的缺省值为 3000 字节。如果索引文件很大，为保险起见，应在用户程序顶部增加一行代码以把栈扩展到最大：

```
extern unsigned  _stklen = 10000 ;
```

另外，在磁盘目录\TC 下有一个 TURBOC. CFG 文件，该文件给定 TCC 命令行的编译选项，包含如下信息：

```
-LC:\TC\LIB -IC:\TC\INCLUDE -ml -c -DTURBO -N
```

这行信息描述了库目录、include 目录、大模式、说明只编译，定义了 TURBO 开关。TURBO 开关用于说明将使用任何 Turbo C 专用源代码。Code Base 源码在 #ifdef 和 #ifndef 预处理程序中使用 TURBO 开关指向编译程序。

在磁盘目录\TC 下有一个批文件 C4TC. BAT，它用于编译和链接小的 Code Base 应用程序。

1.3.3 Quick C 环境下使用 Code Base

为在 Quick C 交互环境下使用 Code Base 库，需要一个 MAKE 文件。通过 Quick C 窗口选择项，增加两个文件 D4LEARN. C 和 M4. LIB。这两个文件帮助建立 make 文件并编译和连接 D4LEARN. EXE 程序。

必须说明大的存储模型，分配额外的栈空间，栈空间约为 10K 最为保险。在 Quick C 交互环境下，有菜单选项。

Quick C 的命令行编译程序 QCL. EXE 的使用方法与 CL. EXE 相同，CL. EXE 是 Microsoft C 编译程序，请参考下节“Microsoft C 环境下使用 Code Base”。

1.3.4 Microsoft C 环境下使用 Code Base

运行命令行编译程序 CL.EXE 进行编译，要求说明大的存储模式和附加的栈空间，例如：

```
CL /AL D4LEARN.C /link M4.LIB /STACK:10000
```

关于编译和连接选择项的详细解释，请参考《Microsoft C 编译器用户指南》。在磁盘目录\MSC 下，有一个批文件 C4MSC.BAT，用于编译和链接小的 Code Base 应用程序。

1.3.5 Zortech C++和 Watcom C 386 环境下使用 Code Base

应使用大模式、分配附加的栈空间。

1.3.6 Code Base 的安装

使用 Code Base，仅需要拷贝专用的库文件、头文件和配置信息。

① 从适当库盘中拷贝三个预建库之一。如果没有适用的库可选择，则参考附录 D 的建库部分。

Turbo C 的例子：

```
COPY A:\TC\T4.LIB C:TC\LIB
```

其中，COPY 命令把与 Turbo C 兼容的 Code Base 库从 A:盘传送到 C:盘目录\TC\LIB 下。

Microsoft C (Quick C)的例子：

```
COPY A:\MSC\M4.LIB C:\MSC
```

该命令把与 Microsoft C 兼容的 Code Base 库从 Code Base 盘目录\MSC 中拷贝到 C:\MSC 中，如果 Microsoft C 在 OS/2 环境下使用，请参考附录 D 的建库部分。

② 从 Code Base 盘目录\H 下拷贝头文件。注意 Turbo C 头文件与 Microsoft C 和 Quick C 的头文件相同。

例子：

```
COPY A:\H\*.* C:\TC\INCLUDE
```

上述命令把所有的 Code Base 头文件盘拷贝到 Turbo C 的 Include 目录下。这些头文件是使用 Code Base 必不可少的。

③ 如果打算使用 Code Base 源文件，需要拷贝 Code Base 盘目录\SOURCE 的内容。

例子：

```
COPY  A:\SOURCE\*.C  C:\MSC
```

1.3.7 运行示范程序 d4learn

程序 d4learn.exe 存在磁盘目录\EXAMPLES 下，用于帮助学习 Code Base。请使用下拉菜单选择 Code Base 例程，当 d4learn 成为例程参数时，这个例程开始执行并把结果显示出来。

在磁盘目录\EXAMPLES 下，另外还有一些示范程序，请您参阅文件 EXAMPLES\EXAMPLES.DOC。

1.3.8 创建 Code Base 应用程序

在 Code Base 应用程序源文件中，经常要用到下述 #include 伪指令：

```
#include <d4base.h>
#include <w4.h>
```

如果 Code Base 头文件不是放在 include 目录中，而是在 Source 目录中，则使用下述 #include指令：

```
#include "d4base.h"
#include "w4.h"
```

文件 d4base.h 为 Code Base 数据管理例程定义结构和模式；w4.h 为 Code Base 屏幕管理例程定义结构和模式。

例子：

```
/* 文件 d4example.c 中有一个类似的例子 */
#include "d4base.h"
#include "w4.h"

#ifdef TURBO
    extern unsigned  _stklen = 10000;
#endif

main()
{
    int j;      long ref;

    /* 初始化 Code Base 并清除屏幕 */
    d4init();
    w4clear(-1);

    /* 打开数据库 */
    if (d4use("d4learn") < 0) exit(1);

    w4(0,5,"D4LEARN.DBF Field Names:");
    for (j=1; j<= f4num_fields(); j++)
    {
        ref = f4j_ref(j);

        /* 显示字段名 */
        w4(j,8,f4name(ref));
    }
```

```
    w4cursor( j,0 ) ;

    w4exit( 0 ) ;
}
```

1.4 常见问题

常见问题主要包括四个方面，即编译、连接、调试和理解。

① 编译问题

在编译一个 Code Base 源程序或开关例子文件时，用户有时会遇到编译错误，通常是由于忘记设置适当的编译开关所致。例如，Turbo C 用户经常忘记设置 TURBO 条件编译开关。不同的编译程序往往使用不同的头文件，采用不同的运行库，Code Base 用条件编译开关来解决这些差别。

遇到复杂的编译应用程序代码，要尽量予以简化。把出错的代码行转换成测试文件。先删去测试程序中通不过编译的部分。这种办法往往能准确地把问题分离出来。

② 连接问题

大多数模糊的连接错误，象 undefined symbol_fcvt87 或 fixup overflow 都是由编译的不一致引起的。例如，修改了存储模式而没有对目标模块进行重新编译，或库文件中的目标模块也许没有建立与应用程序相同的存储模式。另一种编译不一致的方法是对一些编译过程进行浮点仿真，而对另一些编译过程使用不同的浮点数。这种一致的编译规则对库文件中的目标模块也同样适用，即库文件中的目标模块与应用程序用同样的方式编译。

另一个引起外部代码未定义的原因是忘记连接适当的库或目标模块，也可能是调用了不存在的例程。

第二个字符是数字 4 的所有外部过程名有可能是 Code Base 定义的外部过程。Code Base 外部过程名都遵循这一约定。Code Base 内部使用的外部变量大部分在文件“d4init. C”中定义，有时也会有一个未定义的 Code Base 变量，这是由于没有调用初始化 Code Base 的例程。例如，如果仅调用 d4seek，变量 v4cur_base 就成为未定义的外部过程。如果在调用 d4seek 之前调用了 d4use，则 v4cur_base 就被定义了。这是因为 d4seek 引用了一个包含在文件 d4init. c 中的初始化例程，并且与 d4init. c 对应的目标模块变成应用程序的一部分。变量 v4cur_base 是在文件 d4init. c 中说明的。

③ 调试问题

如果 Code Base 程序不象予期的那样工作，则可用下列方法确定问题之所在。

1. 让编译程序查错，使用完全原型的最大警告级编译。
2. 分离问题

把应用程序中不能正确工作的部分分隔出来，并独立地运行这部分程序。

3. 先发现的问题先解决

研究首先发现的问题，第二个问题有可能是由第一个问题而引起的。

4. 确定出错点

如果一个正在运行的例程中止了运行，工作存储区有可能被破坏，这时要确定程序在何处中止运行，这一点可能就是错误发生之处。