NORTHERN ILLINOIS UNIVERSITY

A Microsoft Windows-Based Information Search and Retrieval Tool: INFOTOOL

A Thesis Submitted to the
University Honors Program
In Partial Fulfillment of the
Requirements of the Baccalaureate Degree
With University Honors
Department of
Electrical Engineering

by
Todd E. Toles
DeKalb, Illinois
May 13, 1995

Honors Thesis Abstract
Thesis Submission Form

Author:	Todd E. Toles		
Thesis Title:	A Microsoft Windows-Based Information Search and Retrieval Tool: INFOTOOL		
Advisor:	Dr. Gerald Miller	Advisors Dept:	Electrical Engineering
Discipline:	Electrical Engineering	Year:	1995
Page Length:	157	Bibliography:	Yes
Illustrated:	Yes	Published:	No
Copies:	Hard Copy, Diskette		

Abstract:

This project implements a Microsoft Windows-based information search and retrieval tool that supports mapping a single item to multiple keywords. It supports up to 70 locations per keyword and a description field which describes the item. It is fully compliant with the Multiple Document Interface, Dynamic Data Exchange, and "Point and Click" standards.

Student name: Todd Toles

Approved by: Gerald Miller

Department of: Electrical Engr

Date: 5/11/95

Table of Contents

Introduction 1

Background
- The World Wide Web(WWW) 2
- Database design 5
- Basics of windows programming 7

Design
- Design constraints 9
- Equipment 10
- Structured design 11

Results
- Walk-through of the program 16

Discussion
- Troubles with this project 22
- Suggestions for additional work in this area 24

Conclusion 25

Acknowledgments 26

References 27

Appendices
- A. The main program
  - Inftlapp.cpp A-1
  - Inftlapp.h A-11
- B. The resource scripts
  - Inftlapp.rc B-1
  - Inftlapp.rh B-13
- C. The keyword database class
  - Keyword.cpp C-1
  - Keyword.h C-34
- D. The actions(add,delete,search) file
  - Actions.cpp D-1
  - Actions.h D-20
- E. The places database class
  - Places.cpp E-1

Places.h	E-10
F. The dialogs	
Dialogs.cpp	F-1
Dialogs.h	F-4
G. The about box dialog	G-1
H. The MDI (Multiple Document Interface) child class	H-1
I. The MDI Client class	I-1
J. The Infotlapp program definition file	J-1

List of Figures

1. The Data Flow Diagram (DFD) 11
2. The Control Flow Diagram (CFD) 12
3. Screen capture of the INFOTOOL main screen 16
4. Screen capture of the add dialog box 17
5. Screen capture of the search dialog box 18
6. Screen capture of the search results dialog box 19
7. Screen capture of the description dialog box 20
8. Screen capture of the delete dialog box 21

Abstract

This project implements a Microsoft Windows-based information search and retrieval tool thats supports mapping a single item to multiple keywords. It supports up to 70 locations per keyword and a description field which describes the item.

Introduction

"Now where did I put that?" This is an oft-heard expression around many offices. The problem of managing information and pointers to information is an age old problem; however, with the development of the computer this problem has both been helped and hindered. The inter-networking of computers has made more information accessible to more people, but the computer is also an excellent tool to store and manage information. INFOTOOL is designed to help users better manage their information and be more productive.

This report will provide a background to the various areas of technology that will be used in INFOTOOL, convey the structured design of this package, report on the successful conclusion of this project, discuss the problems that were encountered during the execution of this project, and finally provide suggestions for the improvement of this program.

Background

The World Wide Web

The Internet can be viewed as a vast repository of information just waiting to be tapped. Many tools have been developed to aid in the search and retrieval of this information—among them ftp, gopher, veronica, jughead, and archie. The information search and retrieval tool that has received great attention lately is the World Wide Web concept.

The World Wide Web (WWW) concept was first presented by a group at CERN, the European Laboratory for Particle Physics located in Geneva, Switzerland. They viewed it as a "way to link and access information of various kinds as a web of nodes in which the user can browse at will."(Proposal.html) Initially they wanted to use the WWW to link various dissimilar databases. Their project was approved in October of 1990 (History.html).

The software used for the Web consists of two parts—the client and the server. The server holds the information to be retrieved and the client presents the information that is received from the server to the user.

The beauty of the WWW concept is that it is both platform and software independent. Clients and servers exist for most major hardware platforms, including Sun workstations, MS-Windows based machines, Vax stations, and VM machines. However the most significant advantage is the software independence. Rather than being tied to a

proprietary authoring format, the WWW can use many different types of file formats and retrieval methods. Methods include

1) http. The team and CERN developed this format to serve as the base format for the WWW. It supports links which will be discussed later.
2) ftp. The client can open a ftp session and retrieve a file for you.
3) telnet. The client can open a telnet session to a remote host.
4) gopher. The client can connect to existing gopher servers. It can send gopher commands to manipulate the gopher server.
5) viewers. The software can be configured to invoke various file viewers based on the information requested. Examples of this include calling up a MPEG viewer for a movie file, calling up a GIF/JPEG viewer to view a still picture, and calling up a .WAV/.AU player to listen to a sound clip.

This ability to support multiple software types is a great asset to the WWW suite of software. It can tie all forms of information together.

Another important advantage of the WWW concept is the use of links. A link is a pointer to related information. For example, if this paper was on the WWW (in .html format) and the reader did not know what a .WAV/.AU player was, the reader could click on the phase ".WAV/.AU" and have supporting information explaining the player brought onto the screen. Links are denoted by either using a blue font or putting the text into reverse-video.

The WWW is a milestone in the development of the Internet. By bringing all facets of information into an easy-to-retrieve form, it helps to make the Internet accessible to the non-computer literate person.

Database Design

To manage this information effectively there needs to be a structure that will hold information. This structure is called a database. A database is a means of organizing and managing information so that it can be quickly and easily retrieved. To do this the information is organized into fields and records. A field is a smallest group of information that can be individually addressed; a record is a group of related fields. For example, in the sample below the entire group would be called one record and there would be three fields in that record -- name, address, and phone number.

Name: John Q. Public
Address: 555 Foo Lane
Phone: 815-555-1212

Great care must be taken in database design to ensure that the correct fields are chosen. For example if you wanted to use only the first name in the above example you would have great difficulty in doing so. However, if you designed the database as follows, you could easily pick up the first name.

First Name: John
Middle Initial: Q.
Last Name: Public
Address: 555 Foo Lane
Phone: 815-555-1212

Thus in designing the databases for INFOTOOL great care was taken in designing the database so that the above problems would be avoided.

Basics of Windows Programming

To understand Microsoft Windows programming the reader firsts need to be familiar with a few terms.

Microsoft Windows programming is a complex task and many methods have been used to design Windows programs. The programming language C++ and its use of classes provide an effective solution to the problems of Windows programming. A class is defined as a structure than contains data and the routines to process that data. This follows the model that is most prevalent in real-life -- that data and routines that process the data are inter-related. This, however, is different from many programming languages in which the data and routines that modify this data have a ancillary relationship.

Microsoft Windows is a non-preemptive multi-tasking system, which is defined as an operating system where the OS does not have the ability to interrupt a running process. Thus the program may be competing with other programs for the use of the processor. A program can also set up multiple windows of its own that can communicate with each other. All of these programs operating at the same time must have a method communicating with each other. This method is called message passing. Message passing is a way for one program (or a segment of a program) to let another program to start a process or modify a currently operating process. The code fragment below is an example of a message:

```
DEFINE_RESPONSE_TABLE1(TAddDialog, TDialog)
 EV_COMMAND(IDHELP, CmHelp),
END_RESPONSE_TABLE;
```

When the user is looking at the add dialog box and presses the radio button labeled help a message of IDHELP is generated. The code fragment above tells Windows to then pass control to a function called CmHelp.

A dialog box is a fundamental component of Windows programming. It is a method for a user to provide input to the program of for the program to provide output to the user. A dialog box is a very versatile way of presenting information. In fact most of the communication in INFOTOOL is done with dialog boxes.

A control is a component that is usually included in a dialog box. A control is a way for a user to communicate his wishes to the program. Types of controls include pushbuttons, radio buttons, edit boxes, and static text items. Also of the preceding controls are used in INFOTOOL in one way or another. For example the results of the search are created using an edit box that the program inserts the results into. Then if the user chooses to double-clink on an item that he would like the description of, the edit box is used as input to determine which description to call up.

Design

Design Constraints

The constraints on this program were limited. I did not implement a 32-bit Windows application, but other than that this program supports most of the features that are available in Windows.

Also due to the inability of a program to perform as its specifications claimed that it would, I was unable to implement the html record searching. See the Discussion section in this report for further details.

Equipment

This project was developed using Microsoft Windows for Workgroups 3.11 as the operating system and Borland C++ 4.02 for the C++ compiler. All development work was done on a Gateway 2000 Pentium P5-75 with 16 Megabytes of RAM. The use of the Pentium made the development of this program easier due to the fact that a compile of this program took only around three minutes versus the ten to fifteen minutes that it would have taken if I used a CEET lab machine. Borland C++ includes a series of classes called ObjectWindows which is defined to make programming in Windows easier.

Design

This program was designed by using structured software techniques. I used both PSPECs (Process Specifications), DFDs (Data Flow Diagrams), and CFDs (Control Flow Diagrams).

Control Flow Diagram

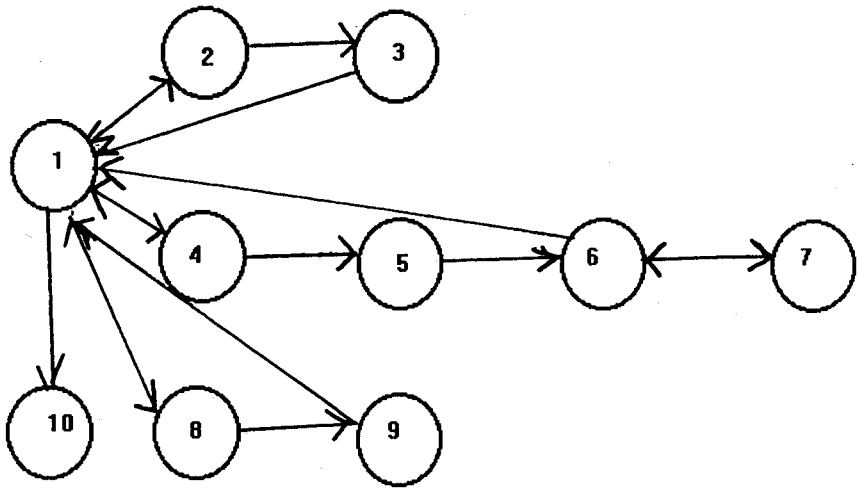


Figure 1

Data Flow Diagram

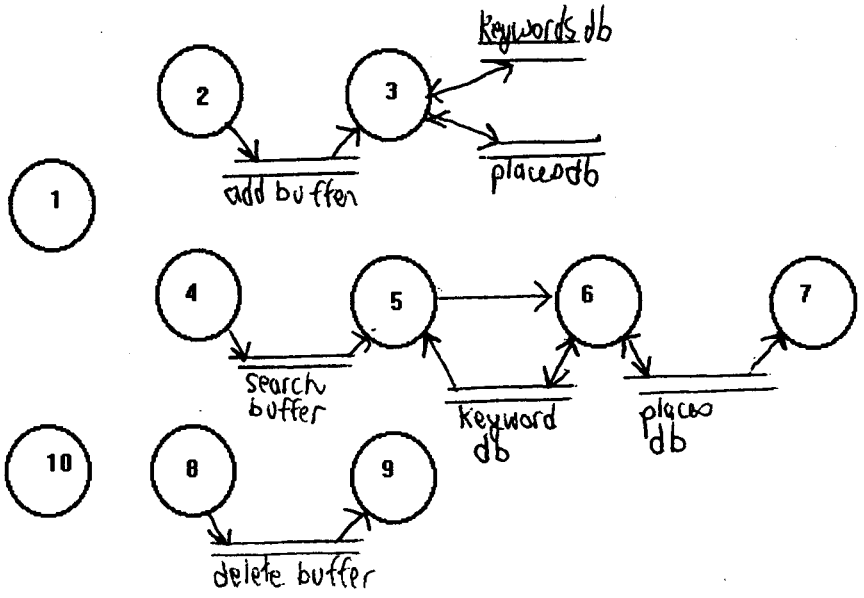


Figure 2

The diagrams above are numbered. The numbers correspond with the numbers associated with the PSPECS.

PSPECS

1 - The **main program** process accepts no input. It creates a window, initializes the menu bar, and waits for user interaction. The process evaluates and validates the user's input and then passes control to one of the following processes: **add dialog, search dialog, delete dialog,** or **exit**. The outputs are messages that indicate that one of the preceding processes is to run.

2 - The **add dialog** process accepts input from the user. The process first creates a dialog box asking for up to nine keywords, a location, a description, and an action. There are two actions possible -- OK and Cancel. If the user selects Cancel, then he is returned to the **main program**. If the user selects OK, then control is transferred to the **add** process. The keywords, the location, and the description are then passed as a buffer to the **add** process.

3 - The **add** process accepts as input as buffer in which the keywords, the location, and the description are described. It opens the places database and reads the last record to determine the last place number used. It then appends a record to the places database consisting of the place number, the place name, and the associated description. The process then closes the places database and opens the keywords database. For every keyword that is not blank (NULL), the process searches the keywords database to see if the keyword is already in the database. If the keyword is already in the database, then the program adds the places location to that keyword's record and increments the number of places field by one. If the keyword is not in the database, then the program creates a new keyword record and places the keyword and the place number into the new record. It also sets the number of places field to one. Then process then closes all open databases and returns control to the **main program**.

4 - The **search dialog** process accepts input from the user. The process first creates a dialog box asking for a keyword to be searched for and an action. There are two actions possible - OK and Cancel. If the user selects Cancel, then he is returned to the **main program.** If the user selects OK, then control is transferred to the **search** process. The keyword to be searched for is passed as a buffer to the **search** process.

5 - The **search** process accepts as input a buffer in which the keyword for be searched for is passed. It then opens the keyword database and searches for the closest matching keyword. It then retrieves the number of places associated with the keyword and then passes both the keyword and the number of places associated with that keyword to the **search results dialog**.

6 - The **search results dialog** process takes as input a keyword and the number of places associated with that keyword. In then creates a dialog box and inserts the number of places specified into the dialog box. It does this by picking up the place number from the keyword record in the keyword database and then searching for it in the places database. The process asks the user to either choose OK or double-click on a place to retrieve the associated description. If the user selects OK then control is passed back to the **main program** and no output is passed. If the user double-clicks on a place then control is passed to the **description dialog** process and the place selected is passed to this process.

7 - The **description dialog** process takes as input a location. The process then searches the places database for a match. When it finds one it then displays the associated description in a message box. When the user selects OK to continue, control is then passed back to the **search results dialog.**

8 - The **delete dialog** process accepts no input. The process first creates a dialog box asking for an item to be deleted and an action. There are two actions possible - OK and Cancel. If the user selects Cancel, then he is returned to the **main program**. If the user selects Cancel, then control is transferred to the **delete** process. The item to be deleted is passed to the **delete** process by using a buffer.

9 - The **delete** process accepts an input buffer which holds the item to be deleted. It then pops-up a message box explaining that this feature is not yet implemented. It then returns control to the **main program**

10 - The **exit** process accepts no input. When it is passed control it closes all open windows, finishes all i/o, and then returns control to Windows.

Results

Being as this is a software project it is difficult to show results. The program responds to all of the design elements except for two. They are the html record connections and then ability to delete a record. I will discuss these problems later on in my paper.

Walk-Through of Program

When INFOTOOL is first started, the user is presented with the following screen:

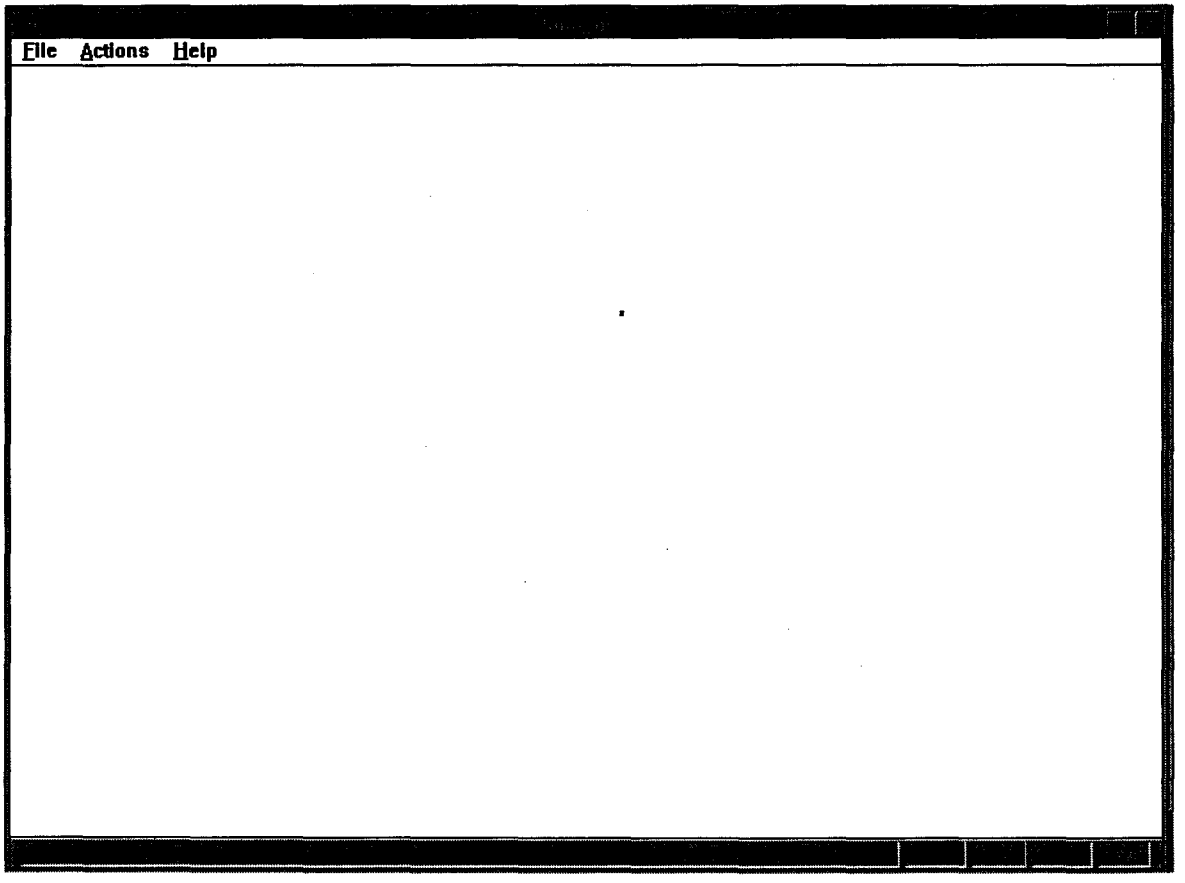


Figure 1

The user then should select actions which will present him with three selections

1) Add
2) Search
3) Delete

If the user selects "Add", then he is presented with the following screen.

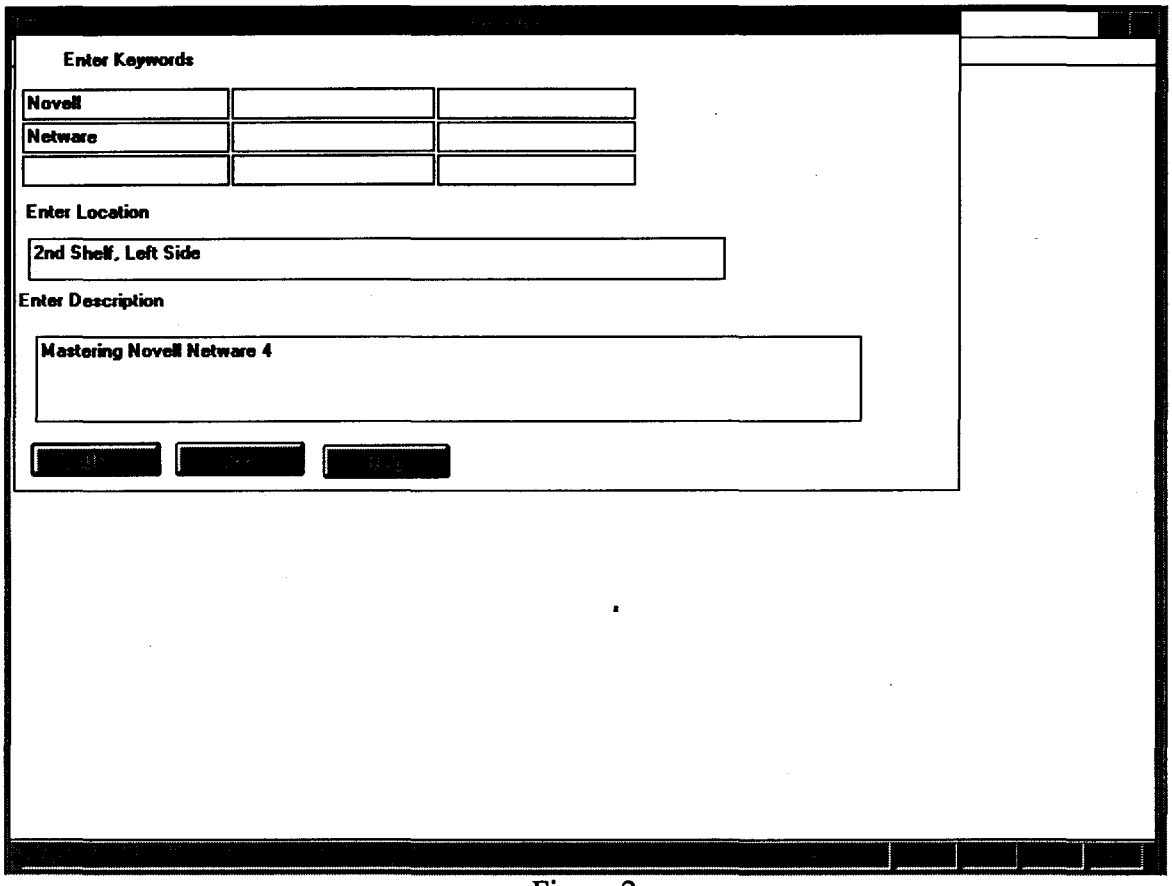


Figure 2

The user can then enter up to nine keywords, an item location, and a description. Then the user clicks on OK and the location, description, and keywords are added to the databases.

If the user selects search, he is presented with the following screen:

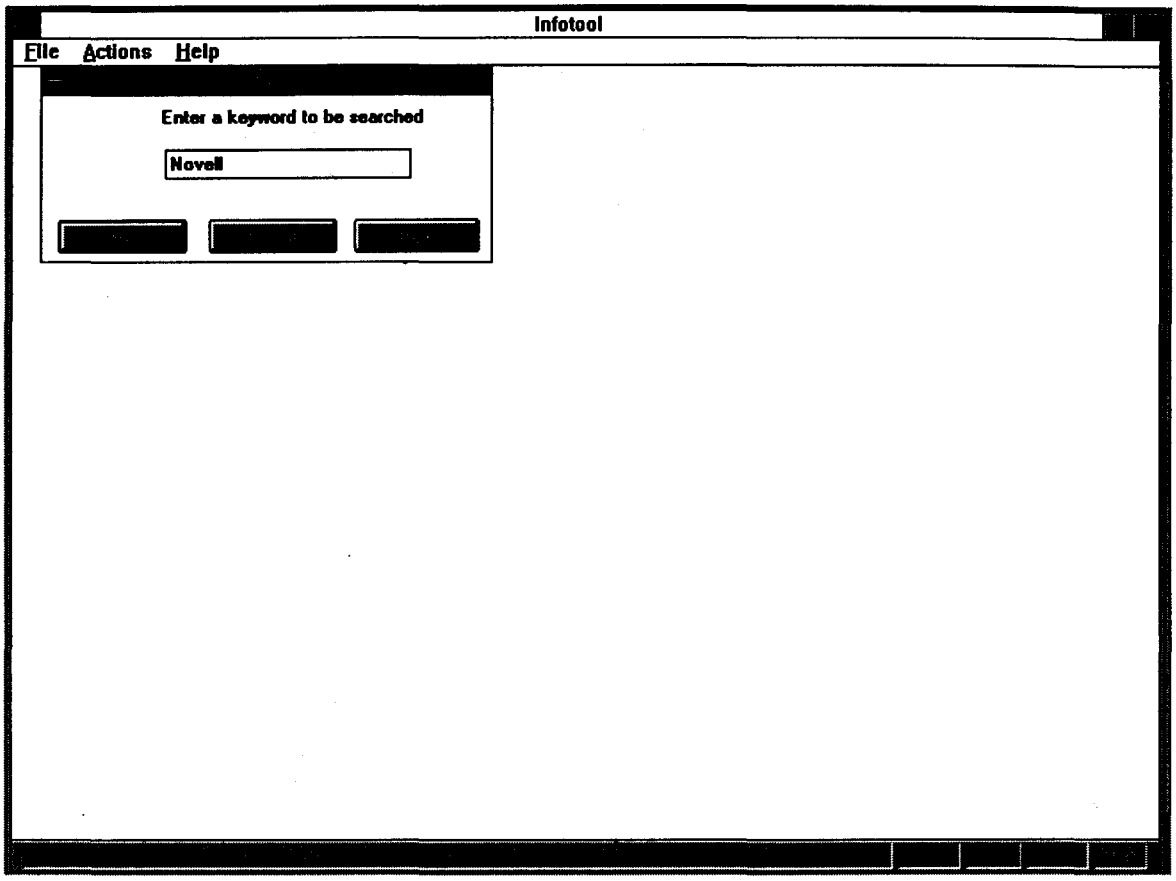


Figure 3

The user is prompted to enter a search keyword and click on OK. When he does this then the program finds the closest match and displays a screen similar to the following:

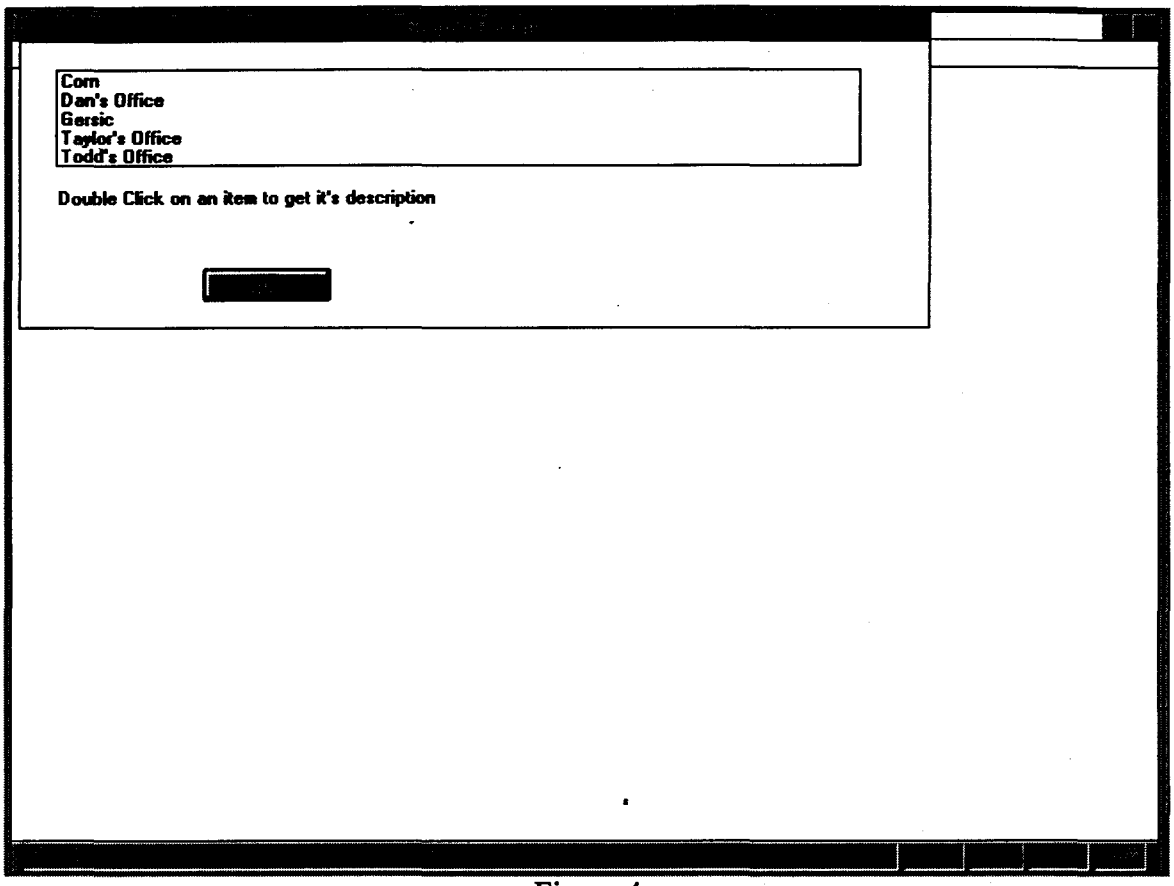


Figure 4

From here the user can either click on OK to close the dialog box or click on a location to bring up a description of that item. If the user clicks on a location, then the following screen is displayed

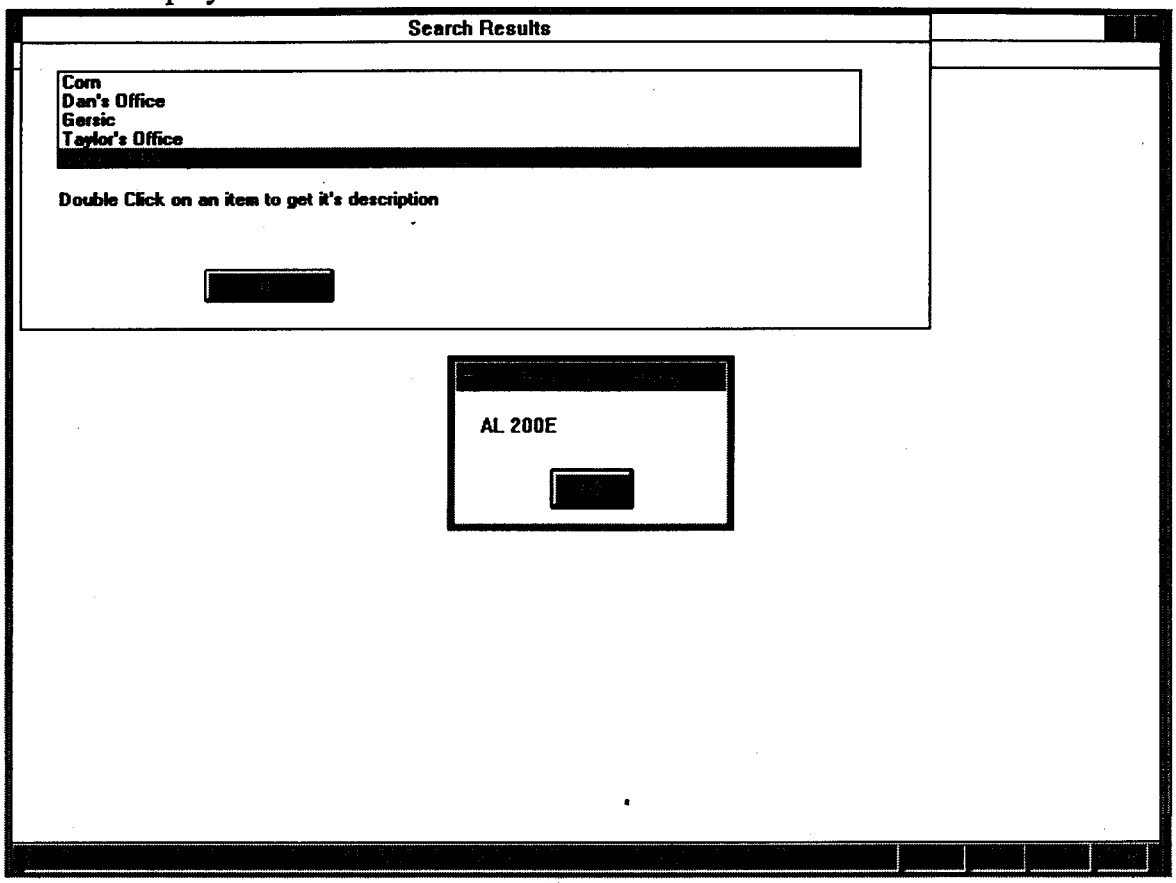


Figure 5

If the user chooses delete then the following screen is displayed:

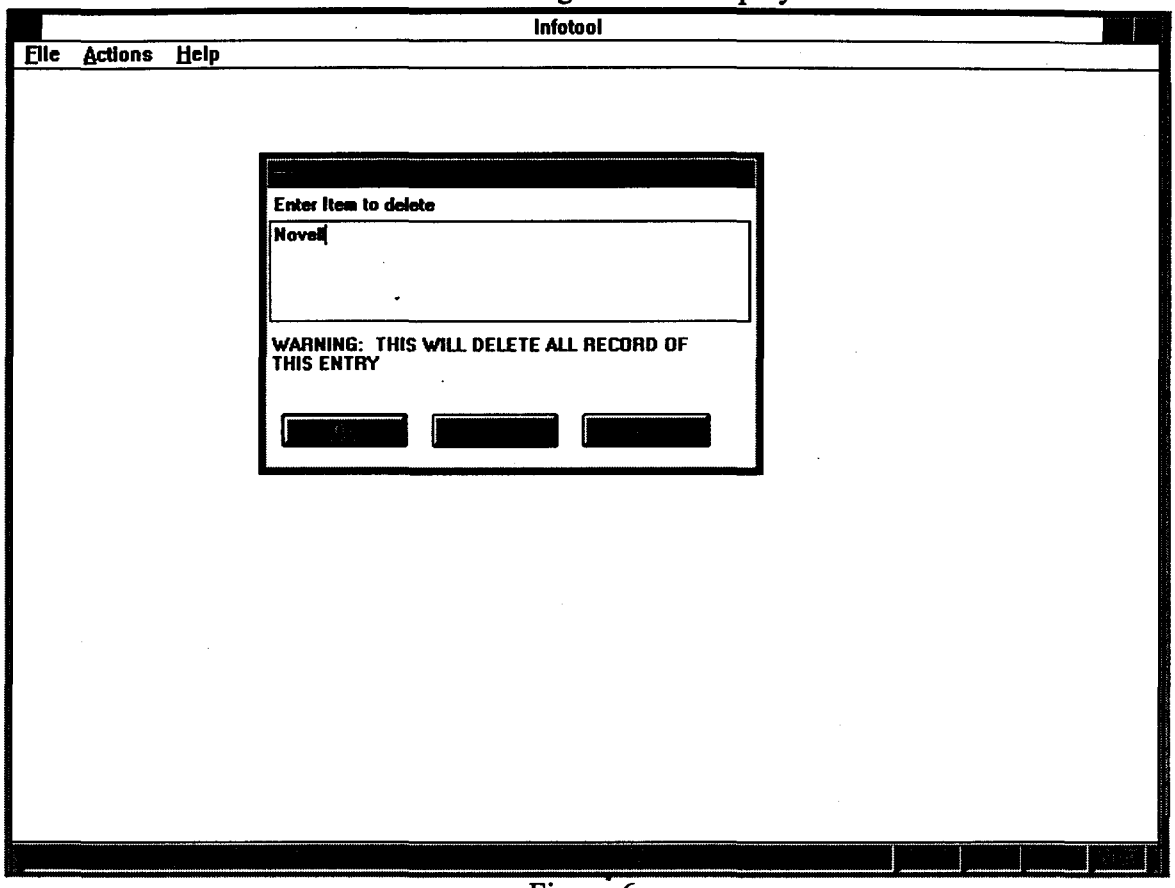


Figure 6

However, in this version nothing is done with this information. The user is just told that this function will be implemented in Version 2.0.

Discussion

Troubles with this Project

This was my first introduction to programming for the Microsoft Windows operating system. I spent many days struggling to make Windows communicate correctly with my program. This problem was exacerbated by the fact that the documentation included with Borland C++ was sketchy in many areas and erroneous in a few areas.

An example of the errors that were included in the documentation were in the area of communicating with listboxes. The printed documentation indicated one method of implementing this, the on-line documentation indicated another method, and the references that I consulted suggested another method of implementing this. However, the documentation that was included with the compiler, both on-line and printed, was incorrect. The correct way to implement this was to use the method that was indicated by the outside reference. The following code fragment show the correct way to do this:

```
TDeleteDialog::TDeleteDialog(TDecoratedMDIFrame * frame, TDeleteBuffer *
  delete_buffer) : TDialog(frame, DELETE_DIALOG)
{
 //Creates a new edit control
 new TEdit(this, IDC_EDIT1, 101);
 //Sets up the transfer buffer so that we can get data from the control
 SetTransferBuffer(delete_buffer);
}
```

However in both sets of sets of documentation supplied by Borland, the makers of the C++ compiler, neglected to include the SetTransferBuffer statement. This function enables the passing of the input from the control to a structure that can then be processed.

I also planned to implement a method to let the user store html records in the database and later recall the records and then retrieve the information. To retrieve the information I intended on using a WWW browse called Air Mosaic by Spry, Inc. I was going to pass Air Mosaic the html record via Dynamic Data Exchange (DDE) and have it retrieve the record. Spry claimed that Version 1.1 of Air Mosaic supported DDE; however, this was not the case. In talking to a Software Engineer at Spry, I was informed that Version 1.1 did not correctly implement the DDE specification; however, a new version would be forthcoming that would correct this problem. Thus I could not make my program drive Air Mosaic into retrieving the information for me and I was forced to leave this additional feature out until Spry corrected this problem.

Suggestions for Additional Work in this Area

Currently there is no method for deleting an item once it has been entered into the database. This will cause problems because the user of this software will eventually want to remove an item from the database and he will be unable to do so. This function will be implemented in Version 2.0 of this software package.

Also there is no method for the user to print a list of locations that match a specified keyword other than to "drag and drop" the information into a package, such as Microsoft Word, that can then print the output. This function will be implemented in Version 2.0 of this software package.

Originally I had intended to allow the user to add html records to the database as items. However I was not able to correctly make this work with the web browses that I had available so this will probably be implemented in Version 3.0 of this software package.

Conclusion

This project acquainted me to the trials and tribulations surrounding programming for the Microsoft Windows operating system. While Microsoft Windows is very user-friendly for the user, it is extremely unfriendly to the programmer. However I was able to complete my defined goals in a reasonable fashion.

I plan to further develop this program by implementing the suggestions that I outlined above and possibly release this software to the world as freeware.

Acknowledgments

I would like to thank the following people:

- My parents, John and Linda Toles, for being there for me for whatever I needed. From financing my education to doing my wash when I did not have the time you were always there for me

- The staff of Grants Fiscal Administration for being understanding when I called into work because I was busy working on my project and for allowing me the use of the facilities to complete this project.

- Microsoft Development Network for providing me with a copy of the Microsoft Development Network CDs.

References

Berners-Lee,T & R. Cailliau. "WorldWideWeb: Proposal for a HyperText Project." <http://info.cern.ch/hypertext/www/Project.html>

Franks, Mike. "Building An Internet Information Server." Network Computing. March 01, 1994 5:3 p.164.

"History of WWW Development at CERN." <http://info.cern.ch/hypertext/www/History.html>

Microsoft Developer Network. Development Library: October 1994 Edition

Microsoft Developer Network. Development Platform: October 1994 Edition

Pappas, Chris H and William H. Murray. Borland C++ Handbook. Berkeley, California: McGraw-Hill, 1994.

Perry, Paul J. Your Borland C++ Consultant. Indianapolis, In: Sams Publishing, 1993.

Swan, Tom. Mastering Window Programming with Borland C++. Indianapolis, In: Sams Publishing, 1994.

van den Bout, Theo. The CS-libraries 1.6.b March 1995.

Appendix A BEGIN INFTLAPP.CPP

```
/*  Project Infotool
    HBT Consulting
    Copyright _ 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftlapp.cpp
    AUTHOR:       Todd E. Toles


       OVERVIEW
    ========
       Source file for implementation of InfotoolApp (TApplication).
*/


#include <owl\owlpch.h>
#include <owl\applicat.h>
#include <owl\framewin.h>
#include <cstring.h>
#include <owl\radiobut.h>
#include <owl\button.h>
#include <owl\window.h>
#include <owl\groupbox.h>
#include <owl\checkbox.h>
#include <owl\dialog.h>
#include <owl\gdiobjec.h>       //For TIcon class
#include <owl\dc.h>             //For dc
#pragma hdrstop

#include <dir.h>

#include "inftlapp.h"
#include "inftmdic.h"
#include "inftmdi1.h"
#include "inftlabd.h"                          // Definition of about dialog.


//
// Generated help file.
//
const char HelpFileName[] = "infotool.hlp";
const WORD ID_RBUTTON1 = 102;
const WORD ID_RBUTTON2 = 103;
const WORD ID_GROUPBOX = 104;

// Drag / Drop support:
TFileDrop::TFileDrop (char* fileName, TPoint& p, BOOL inClient, TModule*
module)
{
```

```
        char    exePath[MAXPATH];

    exePath[0] = 0;
    FileName = strcpy(new char[strlen(fileName) + 1], fileName);
       Point = p;
       InClientArea = inClient;

       Icon = (WORD)FindExecutable(FileName, ".\\", exePath) <= 32 ? 0 :
::ExtractIcon(*module, exePath, 0);

       // Use a question mark if couldn't get the icon from the executable.
       //
    if ((WORD)Icon <= 1) {  // 0=no icons in exe,  1=not an exe
        Icon = LoadIcon(0, (WORD)Icon == 1 ? IDI_APPLICATION : IDI_QUESTION);
        DefIcon = TRUE;
    } else
              DefIcon = FALSE;
}

TFileDrop::~TFileDrop ()
{
    delete FileName;
       if (!DefIcon)
              FreeResource(Icon);
}

const char *TFileDrop::WhoAmI ()
{
  return FileName;
}


//{{InfotoolApp Implementation}}

//{{DOC_VIEW}}
DEFINE_DOC_TEMPLATE_CLASS(TFileDocument, TEditView, DocType1);
//{{DOC_VIEW_END}}

//{{DOC_MANAGER}}
DocType1 __dvt1("All Files (*.*)", "*.*", 0, "dbf", dtAutoDelete |
dtUpdateDir);
//{{DOC_MANAGER_END}}

//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(InfotoolApp, TApplication)
//{{InfotoolAppRSP_TBL_BEGIN}}
    EV_OWLVIEW(dnCreate, EvNewView),
    EV_OWLVIEW(dnClose,  EvCloseView),
    EV_COMMAND(CM_HELPABOUT, CmHelpAbout),
    EV_COMMAND(CM_HELPCONTENTS, CmHelpContents),
    EV_COMMAND(CM_ADDITEM, CmAddItem),
```

```
    EV_COMMAND(CM_DELETE, CmDeleteItem),
    EV_COMMAND(CM_SEARCH, CmSearchItem),
    EV_COMMAND(CM_HELPUSING, CmHelpUsing),
    EV_WM_DROPFILES,
    EV_WM_WININICHANGE,
//{{InfotoolAppRSP_TBL_END}}
END_RESPONSE_TABLE;


//////////////////////////////////////////////////////////
// InfotoolApp
// =====
//
InfotoolApp::InfotoolApp () : TApplication("Infotool")
{
       HelpState = FALSE;
       ContextHelp = FALSE;
       HelpCursor = 0;

       Printer = 0;
       Printing = FALSE;

       DocManager = new TDocManager(dmMDI | dmMenu);

}


InfotoolApp::~InfotoolApp ()
{
       if (Printer)
                  delete Printer;
}


BOOL InfotoolApp::CanClose ()
{
       BOOL result = TApplication::CanClose();

       //
    // Close the help engine if we used it.
       //
    if (result && HelpState)
        MainWindow->WinHelp(HelpFileName, HELP_QUIT, 0L);

       return result;
}


//////////////////////////////////////////////////////////
// InfotoolApp
// =====
// Application intialization.
//
void InfotoolApp::InitMainWindow ()
```

```
{
    TDecoratedMDIFrame* frame = new TDecoratedMDIFrame(Name, MDI_MENU, *(new
InfotoolMDIClient), TRUE);

    // Override the default window style for the main window.
       frame->Attr.Style |= WS_BORDER | WS_CAPTION | WS_CLIPCHILDREN |
WS_MAXIMIZEBOX | WS_MINIMIZEBOX | WS_SYSMENU | WS_THICKFRAME | WS_VISIBLE;
       frame->Attr.Style &= ~(WS_CHILD);

    nCmdShow = (nCmdShow != SW_SHOWMINNOACTIVE) ? SW_SHOWNORMAL : nCmdShow;

    //
    // Assign ICON w/ this application.
       //
    //frame->SetIcon(this, IDI_MDIAPPLICATION);
    frame->SetIcon(this, ICON_PROG);

       //
       // Menu associated with window and accelerator table associated with
table.
    //
       frame->AssignMenu(MDI_MENU);

    //
       // Associate with the accelerator table.
    //
    frame->Attr.AccelTable = MDI_MENU;



    TStatusBar *sb = new TStatusBar(frame, TGadget::Recessed,

      TStatusBar::CapsLock        |
                                    TStatusBar::NumLock         |
                                    TStatusBar::ScrollLock      |
                                    TStatusBar::Overtype);
    frame->Insert(*sb, TDecoratedFrame::Bottom);

       MainWindow = frame;

}


////////////////////////////////////////////////////////
// InfotoolApp
// =====
// Response Table handlers:
//
void InfotoolApp::EvNewView (TView& view)
{
    TMDIClient *mdiClient = TYPESAFE_DOWNCAST(MainWindow->GetClientWindow(),
TMDIClient);
    if (mdiClient) {
```

```
        InfotoolMDIChild* child = new InfotoolMDIChild(*mdiClient, 0,
view.GetWindow());

              // Associate ICON w/ this child window.
        child->SetIcon(this, IDI_DOC);

              child->Create();
    }
}


void InfotoolApp::EvCloseView (TView&)
{
}
//Responds to the menu choice Actions|Search
//Creates a dialog box, gets a search key, searches based on that key
//then returns the results
void InfotoolApp::CmSearchItem()
{
  memset(&search_buffer,0,sizeof(search_buffer)); //Sets up the buffer
  //Creates the Dialog
  TSearchDialog* searchDialog = new TSearchDialog(0,&search_buffer);
  //Executes the Dialog
  if(searchDialog->Execute() == IDOK)
  {
    //Creates the Dialog
    TSearchResponse* cdialog = new TSearchResponse(0, search_buffer);
    //Executes the Dialog
    if(cdialog->Execute()==MB_OK)
    {
       //If we needed to do any processing.  It would go here
    }
  }
}
//Responds to the menu choice Actions|Delete
//Creates a dialog box, gets a delete key, and then deletes the item
void InfotoolApp::CmDeleteItem()
{
  //Allocates memory for the buffer
  memset(&delete_buffer,0,sizeof(delete_buffer));
  //Creates the Dialog
  TDeleteDialog * deleteDialog = new TDeleteDialog(0,&delete_buffer);
  //Executes the Dialog
  if(deleteDialog->Execute() == IDOK)
  {
     // Commented out until Version 2.0
     //delete_data(delete_buffer);
     MessageBox(0,"This function will be added in Version 2","SORRY!",MB_OK |
MB_ICONSTOP);
  }
}

//Responds to the menu item actions|add
//Creates a dialog box, gets the data, and then adds it to the databases
```

```
void InfotoolApp::CmAddItem ()
{
  //Allocates memory for the buffer
  memset(&add_buffer,0,sizeof(add_buffer));
  //Creates the Dialog
  TAddDialog * addDialog = new TAddDialog(0,&add_buffer);
  //Executes the Dialog
  if(addDialog->Execute() == IDOK)
  {
     //Calls the processing function
     add_data(add_buffer);
  }
}


//////////////////////////////////////////////////////////
// InfotoolApp
// =====
// Menu Help Contents command
void InfotoolApp::CmHelpContents ()
{
    //
    // Show the help table of contents.
       //
    HelpState = MainWindow->WinHelp(HelpFileName, HELP_CONTENTS, 0L);
}


//////////////////////////////////////////////////////////
// InfotoolApp
// =====
// Menu Help Using Help command
void InfotoolApp::CmHelpUsing ()
{
    //
    // Display the contents of the Windows help file.
    //
    HelpState = MainWindow->WinHelp(HelpFileName, HELP_HELPONHELP, 0L);
}


//////////////////////////////////////////////////////////
// InfotoolApp
// ===========
// Menu Help About infotool.exe command
void InfotoolApp::CmHelpAbout ()
{
       //
    // Show the modal dialog.
    //
       InfotoolAboutDlg(MainWindow).Execute();
}
```

```
void InfotoolApp::InitInstance ()
{
    TApplication::InitInstance();

       // Accept files via drag/drop in the frame window.
       MainWindow->DragAcceptFiles(TRUE);
}


void InfotoolApp::EvDropFiles (TDropInfo drop)
{
    // Number of files dropped.
    int totalNumberOfFiles = drop.DragQueryFileCount();

    TFileList* files = new TFileList;

    for (int i = 0; i < totalNumberOfFiles; i++) {
        // Tell DragQueryFile the file interested in (i) and the length of
your buffer.
              int     fileLength = drop.DragQueryFileNameLen(i) + 1;
        char    *fileName = new char[fileLength];

              drop.DragQueryFile(i, fileName, fileLength);

        // Getting the file dropped. The location is relative to your client
coordinates,
        // and will have negative values if dropped in the non client parts of
the window.
              //
        // DragQueryPoint copies that point where the file was dropped and
returns whether
        // or not the point is in the client area.  Regardless of whether or
not the file
        // is dropped in the client or non-client area of the window, you will
still receive
        // the file name.
              TPoint  point;
        BOOL    inClientArea = drop.DragQueryPoint(point);
              files->Add(new TFileDrop(fileName, point, inClientArea, this));
    }

    // Open the files that were dropped.
    AddFiles(files);

       // Release the memory allocated for this handle with DragFinish.
    drop.DragFinish();
}


void InfotoolApp::AddFiles (TFileList* files)
{
    // Open all files dragged in.
    TFileListIter fileIter(*files);
```

```
    while (fileIter) {
        TDocTemplate* tpl = GetDocManager()->MatchTemplate(fileIter.Current()-
>WhoAmI());
        if (tpl)
            tpl->CreateDoc(fileIter.Current()->WhoAmI());
        fileIter++;
       }
}


BOOL InfotoolApp::ProcessAppMsg (MSG& msg)
{
    if (msg.message == WM_COMMAND) {
        if (ContextHelp || (GetKeyState(VK_F1) < 0)) {
            ContextHelp = FALSE;
            MainWindow->WinHelp(HelpFileName, HELP_CONTEXT, msg.wParam);
            return TRUE;
        }
    } else
              switch (msg.message) {
        case WM_KEYDOWN:
                        if (msg.wParam == VK_F1) {
                                // If the Shift/F1 then set the help cursor and
turn on the modal help state.
                                if (::GetKeyState(VK_SHIFT) < 0) {
                    ContextHelp = TRUE;
                    HelpCursor = ::LoadCursor(MainWindow->GetModule()-
>GetInstance(), MAKEINTRESOURCE(IDC_HELPCURSOR));
                                        ::SetCursor(HelpCursor);
                    return TRUE;        // Gobble up the message.
                } else {
                    // If F1 w/o the Shift key then bring up help's main
index.
                    MainWindow->WinHelp(HelpFileName, HELP_INDEX, 0L);
                                        return TRUE;        // Gobble up the
message.
                }
                        } else {
                if (ContextHelp && (msg.wParam == VK_ESCAPE)) {
                    if (HelpCursor)
                        ::DestroyCursor(HelpCursor);
                    ContextHelp = FALSE;
                                        HelpCursor = 0;
                                        MainWindow->SetCursor(0, IDC_ARROW);
                    return TRUE;    // Gobble up the message.
                }
                        }
                        break;

              case WM_MOUSEMOVE:
        case WM_NCMOUSEMOVE:
            if (ContextHelp) {
                ::SetCursor(HelpCursor);
                return TRUE;        // Gobble up the message.
```

```
            }
            break;

        case WM_INITMENU:
                        if (ContextHelp) {
                                ::SetCursor(HelpCursor);
                return TRUE;        // Gobble up the message.
                        }
                        break;

        case WM_ENTERIDLE:
            if (msg.wParam == MSGF_MENU)
                if (GetKeyState(VK_F1) < 0) {
                ContextHelp = TRUE;
                MainWindow->PostMessage(WM_KEYDOWN, VK_RETURN, 0L);
                return TRUE;        // Gobble up the message.
            }
                        break;

              default:
                        ;
              };  // End of switch

    // Continue normal processing.
       return TApplication::ProcessAppMsg(msg);
}


void InfotoolApp::EvWinIniChange (char far* section)
{
    if (lstrcmp(section, "windows") == 0) {
                // If the device changed in the WIN.INI file then the printer
        // might have changed.  If we have a TPrinter (Printer) then
        // check and make sure it's identical to the current device
        // entry in WIN.INI.
        if (Printer) {
                        char printDBuffer[255];
                        LPSTR printDevice = printDBuffer;
            LPSTR devName = 0;
            LPSTR driverName = 0;
                        LPSTR outputName = 0;

            if (::GetProfileString("windows", "device", "", printDevice,
sizeof(printDevice))) {
                                // The string which should come back is
something like:
                //
                //      HP LaserJet III,hppcl5a,LPT1:
                //
                // Where the format is:
                //
                //      devName,driverName,outputName
                //
                devName = printDevice;
```

```
                                while (*printDevice) {
                                        if (*printDevice == ',') {
                                *printDevice++ = 0;
                                                if (!driverName)
                                                        driverName =
printDevice;
                                else
                                    outputName = printDevice;
                            } else
                                printDevice = AnsiNext(printDevice);
                        }

                                if ((Printer->GetSetup().Error != 0)
||
                                        (lstrcmp(devName, Printer-
>GetSetup().GetDeviceName()) != 0)     ||
                                        (lstrcmp(driverName, Printer-
>GetSetup().GetDriverName()) != 0) ||
                                        (lstrcmp(outputName, Printer-
>GetSetup().GetOutputName()) != 0)) {

                                        // New printer installed so get the new
printer device now.
                                        delete Printer;
                                        Printer = new TPrinter;
                                }
                                } else {
                                // No printer installed (GetProfileString
failed).
                                delete Printer;
                                Printer = new TPrinter;
                                }
                        }
                }
}


int OwlMain (int , char* [])
{
        InfotoolApp     App;
        int             result;

        result = App.Run();

        return result;
}
```

```
BEGIN INFTLAPP.H
#if !defined(__inftlapp_h)              // Sentry, use file only if it's not
already included.
#define __inftlapp_h

/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftlapp.h
    AUTHOR:       Todd E. Toles


       OVERVIEW
    ========
    Class definition for InfotoolApp (TApplication).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include <owl\statusba.h>
#include "dialog.h"
#include "actions.h"
#include <owl\dialog.h>
#include <owl\mdi.h>
#include <owl\windowev.h>
#include <owl\framewin.h>
#include <owl\listbox.h>
#include <owl\editview.h>
#include "keyword.h"
#include "places.h"
#include <owl\listview.h>
#include <owl\docmanag.h>
#include <owl\edit.h>
#include <owl\filedoc.h>
#include <owl\printer.h>

//
#include <classlib\bags.h>


#include "inftlapp.rh"            // Definition of all resources.


// TFileDrop class Maintains information about a dropped file, its name, where
it was dropped,
// and whether or not it was in the client area
class TFileDrop {
public:
       operator == (const TFileDrop& other) const {return this == &other;}
```

```
        char*   FileName;
        TPoint  Point;
    BOOL    InClientArea;

        HICON   Icon;
        BOOL    DefIcon;

        TFileDrop (char*, TPoint&, BOOL, TModule* module);
    ~TFileDrop ();

    const char* WhoAmI ();
private:
    //
    // hidden to prevent accidental copying or assignment
        //
        TFileDrop (const TFileDrop&);
        TFileDrop & operator = (const TFileDrop&);
};

typedef TIBagAsVector<TFileDrop> TFileList;
typedef TIBagAsVectorIterator<TFileDrop> TFileListIter;


//{{TApplication = InfotoolApp}}
class InfotoolApp : public TApplication {
private:
        BOOL            HelpState;                          // Has the help
engine been used.
        BOOL            ContextHelp;                        // SHIFT-F1 state
(context sensitive HELP)
        HCURSOR         HelpCursor;                         // Context
sensitive help cursor

private:
        void AddFiles (TFileList* files);
public:
        InfotoolApp ();
        virtual ~InfotoolApp ();

        // Public data members used by the print menu commands and Paint
routine in MDIChild.
        TPrinter        *Printer;                           // Printer support.
        BOOL            Printing;                           // Printing in
progress.
     TAddBuffer      add_buffer;                        // Creates the add
buffer
      TSearchBuffer   search_buffer;                     // Creates the
search_buffer
     TDeleteBuffer   delete_buffer;                     // Creates the
delete_buffer
//{{InfotoolAppVIRTUAL_BEGIN}}
public:
        virtual void InitMainWindow();
```

```
        virtual void InitInstance();
    virtual BOOL CanClose ();
        virtual BOOL ProcessAppMsg (MSG& msg);
//{{InfotoolAppVIRTUAL_END}}

//{{InfotoolAppRSP_TBL_BEGIN}}
protected:
        void EvNewView (TView& view);
        void EvCloseView (TView& view);
        void CmHelpAbout ();
        void CmHelpContents ();
    void CmHelpUsing ();
    void CmAddItem ();
    void CmDeleteItem ();
    void CmSearchItem ();
        void EvDropFiles (TDropInfo drop);
        void EvWinIniChange (char far* section);
//{{InfotoolAppRSP_TBL_END}}
  DECLARE_RESPONSE_TABLE(InfotoolApp);
};    //{{InfotoolApp}}

#endif                                          // __inftlapp_h sentry.
```

Appendix B BEGIN INFTLAPP.RC

```
/*  Main Infotool
    HBT Consulting
    Copyright © 1995. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftlapp.rc
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    All resources defined here.
*/

#if !defined(WORKSHOP_INVOKED)
#include <windows.h>
#endif
#include "inftlapp.rh"

MDI_MENU MENU
{
 POPUP "&File"
 {
  MENUITEM "Print Pre&view...", CM_FILEPRINTPREVIEW, GRAYED
  MENUITEM "&Print...", CM_FILEPRINT, GRAYED
  MENUITEM "P&rint Setup...", CM_FILEPRINTERSETUP, GRAYED
  MENUITEM SEPARATOR
  MENUITEM "E&xit\tAlt+F4", CM_EXIT
 }

 POPUP "&Actions"
 {
  MENUITEM "&Add", CM_ADDITEM
  MENUITEM "&Search", CM_SEARCH
  MENUITEM "&Delete", CM_DELETE
 }

 POPUP "&Help"
 {
  MENUITEM "&Contents", CM_HELPCONTENTS
  MENUITEM "&Using help", CM_HELPUSING
  MENUITEM SEPARATOR
  MENUITEM "&About...", CM_HELPABOUT
 }

}


// Accelerator table for short-cut to menu commands. (include\owl\editfile.rc)
```

```
MDI_MENU ACCELERATORS
{
 VK_DELETE, CM_EDITCUT, VIRTKEY, SHIFT
 VK_INSERT, CM_EDITCOPY, VIRTKEY, CONTROL
 VK_INSERT, CM_EDITPASTE, VIRTKEY, SHIFT
 VK_DELETE, CM_EDITCLEAR, VIRTKEY, CONTROL
 VK_BACK, CM_EDITUNDO, VIRTKEY, ALT
 VK_F3, CM_EDITFINDNEXT, VIRTKEY
}


// Context sensitive help cursor.
IDC_HELPCURSOR CURSOR "help.cur"


//
// Table of help hints displayed in the status bar.
//
STRINGTABLE
{
 -1, "File/document operations"
 CM_MDIFILENEW, "Creates a new document"
 CM_MDIFILEOPEN, "Opens an existing document"
 CM_VIEWCREATE, "Create a new view for this document"
 CM_FILEREVERT, "Reverts changes to last document save"
 CM_FILECLOSE, "Close this document"
 CM_FILESAVE, "Saves this document"
 CM_FILESAVEAS, "Saves this document with a new name"
 CM_FILEPRINT, "Print this document"
 CM_FILEPRINTERSETUP, "Setup this document print characteristics"
 CM_FILEPRINTPREVIEW, "Display full pages as read-only"
 CM_EXIT, "Quits InfotoolApp and prompts to save the documents"
 CM_EDITUNDO -1, "Edit operations"
 CM_EDITUNDO, "Reverses the last operation"
 CM_EDITCUT, "Cuts the selection and puts it on the Clipboard"
 CM_EDITCOPY, "Copies the selection and puts it on the Clipboard"
 CM_EDITPASTE, "Inserts the clipboard contents at the insertion point"
 CM_EDITDELETE, "Deletes the selection"
 CM_EDITCLEAR, "Clear the document"
 CM_EDITFIND -1, "Search/replace operations"
 CM_EDITFIND, "Finds the specified text"
 CM_EDITREPLACE, "Finds the specified text and changes it"
 CM_EDITFINDNEXT, "Finds the next match"
 CM_CASCADECHILDREN -1, "Window arrangement and selection"
 CM_CASCADECHILDREN, "Cascades open windows"
 CM_TILECHILDREN, "Tiles open windows"
 CM_ARRANGEICONS, "Arranges iconic windows along bottom"
 CM_CLOSECHILDREN, "Closes all open windows"
 CM_HELPCONTENTS -1, "Access online help"
 CM_HELPCONTENTS, "Help table of contents"
 CM_HELPUSING, "Help on using online Help"
 CM_HELPABOUT, "About the Infotool application"
 CM_DELETE, "Add an item to the database"
 CM_SEARCH, "Search the database"
```

```
 2, "Delete an entry from the database"
}


//
// OWL string table
//

// EditFile (include\owl\editfile.rc and include\owl\editsear.rc)
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    IDS_CANNOTFIND,         "Cannot find ""%s""."
    IDS_UNABLEREAD,         "Unable to read file %s from disk."
    IDS_UNABLEWRITE,        "Unable to write file %s to disk."
    IDS_FILECHANGED,        "The text in the %s file has changed.\n\nDo
you want to save the changes?"
    IDS_FILEFILTER,         "Text files (*.TXT)|*.TXT|AllFiles (*.*)|*.*|"
END


// Doc/View (include\owl\docview.rc)
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    IDS_DOCMANAGERFILE,     "&File"
    IDS_DOCLIST,            "--Document Type--"
    IDS_VIEWLIST,           "--View Type--"
    IDS_UNTITLED,           "Untitled"
    IDS_UNABLEOPEN,         "Unable to open document."
    IDS_UNABLECLOSE,        "Unable to close document."
    IDS_READERROR,          "Document read error."
    IDS_WRITEERROR,         "Document write error."
    IDS_DOCCHANGED,         "The document has been changed.\n\nDo you want
to save the changes?"
    IDS_NOTCHANGED,         "The document has not been changed."
    IDS_NODOCMANAGER,       "Document Manager not present."
    IDS_NOMEMORYFORVIEW,    "Insufficient memory for view."
    IDS_DUPLICATEDOC,       "Document already loaded."
END


// Printer (include\owl\printer.rc)
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    IDS_PRNON,              " on "
    IDS_PRNERRORTEMPLATE,   "'%s' not printed. %s."
    IDS_PRNOUTOFMEMORY,     "Out of memory"
    IDS_PRNOUTOFDISK,       "Out of disk space"
    IDS_PRNCANCEL,          "Printing canceled"
    IDS_PRNMGRABORT,        "Printing aborted in Print Manager"
    IDS_PRNGENERROR,        "Error encountered during print"
    IDS_PRNERRORCAPTION,    "Print Error"
END
```

```
// Exception string resources (include\owl\except.rc)
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    IDS_OWLEXCEPTION,           "ObjectWindows Exception"
    IDS_UNHANDLEDXMSG,          "Unhandled Exception"
    IDS_OKTORESUME,             "OK to resume?"
    IDS_UNKNOWNEXCEPTION,       "Unknown exception"

    IDS_UNKNOWNERROR,           "Unknown error"
    IDS_NOAPP,                  "No application object"
    IDS_OUTOFMEMORY,            "Out of memory"
    IDS_INVALIDMODULE,          "Invalid module specified for window"
    IDS_INVALIDMAINWINDOW,      "Invalid MainWindow"

    IDS_INVALIDWINDOW,          "Invalid window %s"
    IDS_INVALIDCHILDWINDOW,     "Invalid child window %s"
       IDS_INVALIDCLIENTWINDOW,    "Invalid client window %s"

    IDS_CLASSREGISTERFAIL,      "Class registration fail for window %s"
    IDS_CHILDREGISTERFAIL,      "Child class registration fail for window %s"
    IDS_WINDOWCREATEFAIL,       "Create fail for window %s"
    IDS_WINDOWEXECUTEFAIL,      "Execute fail for window %s"
    IDS_CHILDCREATEFAIL,        "Child create fail for window %s"

    IDS_MENUFAILURE,            "Menu creation failure"
    IDS_VALIDATORSYNTAX,        "Validator syntax error"
    IDS_PRINTERERROR,           "Printer error"

       IDS_LAYOUTINCOMPLETE,       "Incomplete layout constraints specified in
window %s"
    IDS_LAYOUTBADRELWIN,        "Invalid relative window specified in layout
constraint in window %s"

    IDS_GDIFAILURE,             "GDI failure"
    IDS_GDIALLOCFAIL,           "GDI allocate failure"
    IDS_GDICREATEFAIL,          "GDI creation failure"
    IDS_GDIRESLOADFAIL,         "GDI resource load failure"
    IDS_GDIFILEREADFAIL,        "GDI file read failure"
    IDS_GDIDELETEFAIL,          "GDI object %X delete failure"
    IDS_GDIDESTROYFAIL,         "GDI object %X destroy failure"
    IDS_INVALIDDIBHANDLE,       "Invalid DIB handle %X"
END


// General Window's status bar messages. (include\owl\statusba.rc)
STRINGTABLE
BEGIN
    IDS_MODES                   "EXT|CAPS|NUM|SCRL|OVR|REC"
    SC_SIZE,                    "Changes the size of the window"
    SC_MOVE,                    "Moves the window to another position"
    SC_MINIMIZE,                "Reduces the window to an icon"
    SC_MAXIMIZE,                "Enlarges the window to it maximum size"
    SC_RESTORE,                 "Restores the window to its previous size"
    SC_CLOSE,                   "Closes the window"
```

```
        SC_TASKLIST,                "Opens task list"
    SC_NEXTWINDOW,              "Switches to next window"
END


// Validator messages (include\owl\validate.rc)
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    IDS_VALPXPCONFORM          "Input does not conform to picture:\n""%s"""
    IDS_VALINVALIDCHAR         "Invalid character in input"
    IDS_VALNOTINRANGE          "Value is not in the range %ld to %ld."
    IDS_VALNOTINLIST           "Input is not in valid-list"
END


//
// Print Preview speed bar bitmaps
//
APX_PPR_PREVIOUS BITMAP "previous.bmp"
APX_PPR_NEXT BITMAP "next.bmp"
APX_PPR_ONEUP BITMAP "preview1.bmp"
APX_PPR_TWOUP BITMAP "preview2.bmp"


//
// Misc application definitions
//

// MDI document ICON
IDI_DOC ICON "mdichild.ico"

// Application ICON
IDI_MDIAPPLICATION ICON "appldocv.ico"


// About box.
IDD_ABOUT DIALOG 12, 17, 204, 65
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About Infotool"
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT "Version", IDC_VERSION, 2, 14, 200, 8, SS_NOPREFIX
    CTEXT "490 Project - Infotool", -1, 2, 4, 200, 8, SS_NOPREFIX
    CTEXT "", IDC_COPYRIGHT, 2, 27, 200, 17, SS_NOPREFIX
    RTEXT "", IDC_DEBUG, 136, 55, 66, 8, SS_NOPREFIX
    ICON IDI_MDIAPPLICATION, -1, 2, 2, 16, 16
    DEFPUSHBUTTON "OK", IDOK, 88, 48, 28, 12
END


// Printer abort box.
IDD_ABORTDIALOG DIALOG 84, 51, 143, 64
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Print"
```

```
BEGIN
    PUSHBUTTON "Cancel", IDCANCEL, 52, 44, 40, 14, WS_CHILD | WS_VISIBLE |
WS_TABSTOP
    CTEXT "Printing", -1, 0, 4, 144, 8, SS_NOPREFIX
    CTEXT "%s", ID_TITLE, 0, 12, 144, 8, SS_NOPREFIX
    CTEXT "on the %s Printer", ID_DEVICE, 0, 20, 144, 8, SS_NOPREFIX
    CTEXT "connected to %s", ID_PORT, 0, 28, 144, 8, SS_NOPREFIX
END


// TInputDialog class dialog box.
IDD_INPUTDIALOG DIALOG 20, 24, 180, 64
STYLE WS_POPUP | WS_CAPTION | DS_SETFONT
FONT 8, "Helv"
{
 LTEXT "", ID_PROMPT, 10, 8, 160, 10, SS_NOPREFIX
 EDITTEXT ID_INPUT, 10, 20, 160, 12, WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP | ES_AUTOHSCROLL
 DEFPUSHBUTTON "&OK", IDOK, 47, 42, 40, 14
 PUSHBUTTON "&Cancel", IDCANCEL, 93, 42, 40, 14
}


// Horizontal slider thumb bitmap for TSlider and VSlider
(include\owl\slider.rc)
IDB_HSLIDERTHUMB BITMAP PRELOAD MOVEABLE DISCARDABLE
BEGIN
    '42 4D 66 01 00 00 00 00 00 00 76 00 00 00 28 00'
    '00 00 12 00 00 00 14 00 00 00 01 00 04 00 00 00'
    '00 00 F0 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 10 00 00 00 00 00 00 00 00 00 C0 00 00 C0'
    '00 00 00 C0 C0 00 C0 00 00 00 C0 00 C0 00 C0 C0'
    '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
    '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
    '00 00 FF FF FF 00 BB BB 0B BB BB BB B0 BB BB 00'
    '00 00 BB B0 80 BB BB BB 08 0B BB 00 00 00 BB 08'
    'F8 0B BB B0 87 70 BB 00 00 00 B0 8F F8 80 BB 08'
    '77 77 0B 00 00 00 08 F8 88 88 00 88 88 87 70 00'
    '00 00 0F F7 77 88 00 88 77 77 70 00 00 00 0F F8'
    '88 88 00 88 88 87 70 00 00 00 0F F7 77 88 00 88'
    '77 77 70 00 00 00 0F F8 88 88 00 88 88 87 70 00'
    '00 00 0F F7 77 88 00 88 77 77 70 00 00 00 0F F8'
    '88 88 00 88 88 87 70 00 00 00 0F F7 77 88 00 88'
    '77 77 70 00 00 00 0F F8 88 88 00 88 88 87 70 00'
    '00 00 0F F7 77 88 00 88 77 77 70 00 00 00 0F F8'
    '88 88 00 88 88 87 70 00 00 00 0F F7 77 88 00 88'
    '77 77 70 00 00 00 0F F8 88 88 00 88 88 87 70 00'
    '00 00 0F F7 77 78 00 88 77 77 70 00 00 00 0F FF'
    'FF FF 00 88 88 88 80 00 00 00 B0 00 00 00 BB 00'
    '00 00 0B 00 00 00'
END
```

```
// Vertical slider thumb bitmap for TSlider and HSlider
(include\owl\slider.rc)
IDB_VSLIDERTHUMB BITMAP PRELOAD MOVEABLE DISCARDABLE
BEGIN
    '42 4D 2A 01 00 00 00 00 00 00 76 00 00 00 28 00'
       '00 00 28 00 00 00 09 00 00 00 01 00 04 00 00 00'
    '00 00 B4 00 00 00 00 00 00 00 00 00 00 00 00 00'
    '00 00 10 00 00 00 00 00 00 00 00 00 C0 00 00 C0'
    '00 00 00 C0 C0 00 C0 00 00 00 C0 00 C0 00 C0 C0'
    '00 00 C0 C0 C0 00 80 80 80 00 00 00 FF 00 00 FF'
    '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
    '00 00 FF FF FF 00 B0 00 00 00 00 00 00 00 00 0B'
    'B0 00 00 00 00 00 00 00 00 0B 0F 88 88 88 88 88'
    '88 88 88 80 08 88 88 88 88 88 88 88 88 80 0F 77'
    '77 77 77 77 77 77 77 80 08 77 77 77 77 77 77 77'
    '77 80 0F 77 FF FF FF FF FF FF F7 80 08 77 FF FF'
    'FF FF FF FF F7 80 0F 70 00 00 00 00 00 00 77 80'
       '08 70 00 00 00 00 00 00 77 80 0F 77 77 77 77 77'
    '77 77 77 80 08 77 77 77 77 77 77 77 77 80 0F 77'
    '77 77 77 77 77 77 77 80 08 77 77 77 77 77 77 77'
    '77 80 0F FF FF FF FF FF FF FF FF F0 08 88 88 88'
    '88 88 88 88 88 80 B0 00 00 00 00 00 00 00 00 0B'
    'B0 00 00 00 00 00 00 00 00 0B'
END


// Version info.
//
#if !defined(__DEBUG_)
// Non-Debug VERSIONINFO
1 VERSIONINFO LOADONCALL MOVEABLE
FILEVERSION 1, 0, 0, 0
PRODUCTVERSION 1, 0, 0, 0
FILEFLAGSMASK 0
FILEFLAGS VS_FFI_FILEFLAGSMASK
FILEOS VOS__WINDOWS16
FILETYPE VFT_APP
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        // Language type = U.S. English (0x0409) and Character Set = Windows,
Multilingual(0x04e4)
            BLOCK "040904E4"                                    // Matches
VarFileInfo Translation hex value.
        BEGIN
            VALUE "CompanyName", "HBT Consulting\000"
            VALUE "FileDescription", "Infotool for Windows\000"
            VALUE "FileVersion", "1.0\000"
            VALUE "InternalName", "Infotool\000"
            VALUE "LegalCopyright", "Copyright © 1995 by Todd E. Toles. All
Rights Reserved.\000"
            VALUE "LegalTrademarks", "Windows /231 is a trademark of Microsoft
Corporation\000"
            VALUE "OriginalFilename", "Infotool.EXE\000"
```

```
            VALUE "ProductName", "Infotool\000"
            VALUE "ProductVersion", "1.0\000"
        END
       END

    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x04e4, 0x0409        // U.S. English(0x0409) &
Windows Multilingual(0x04e4) 1252
    END

END
#else

// Debug VERSIONINFO
1 VERSIONINFO LOADONCALL MOVEABLE
FILEVERSION 1, 0, 0, 0
PRODUCTVERSION 1, 0, 0, 0
FILEFLAGSMASK VS_FF_DEBUG | VS_FF_PRERELEASE | VS_FF_PATCHED |
VS_FF_PRIVATEBUILD | VS_FF_SPECIALBUILD
FILEFLAGS VS_FFI_FILEFLAGSMASK
FILEOS VOS__WINDOWS16
FILETYPE VFT_APP
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        // Language type = U.S. English (0x0409) and Character Set = Windows,
Multilingual(0x04e4)
        BLOCK "040904E4"                           // Matches VarFileInfo
Translation hex value.
        BEGIN
                        VALUE "CompanyName", "HBT Consulting\000"
            VALUE "FileDescription", "Infotool for Windows\000"
            VALUE "FileVersion", "1.0\000"
            VALUE "InternalName", "Infotool\000"
            VALUE "LegalCopyright", "Copyright © 1993. All Rights
Reserved.\000"
            VALUE "LegalTrademarks", "Windows \231 is a trademark of Microsoft
Corporation\000"
            VALUE "OriginalFilename", "Infotool.EXE\000"
            VALUE "ProductName", "Infotool\000"
            VALUE "ProductVersion", "1.0\000"
            VALUE "SpecialBuild", "Debug Version\000"
            VALUE "PrivateBuild", "Built by Todd E. Toles\000"
        END
       END

    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x04e4, 0x0409        // U.S. English(0x0409) &
Windows Multilingual(0x04e4) 1252
    END

END
```

```
#endif
ICON_PROG ICON
{
 '00 00 01 00 01 00 20 20 10 00 00 00 00 00 E8 02'
 '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
 '00 00 01 00 04 00 00 00 00 00 80 02 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 80 00 00 80 00 00 00 80 80 00 80 00'
 '00 00 80 00 80 00 80 80 00 00 80 80 80 00 C0 C0'
 'C0 00 00 00 FF 00 00 FF 00 00 00 FF FF 00 FF 00'
 '00 00 FF 00 FF 00 FF FF 00 00 FF FF FF 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF EF'
 '8E 37 FF EF 75 D7 FF EF 75 D7 FF EF 75 D7 FF EF'
 '75 D7 FF EF 8E 37 FF EF FF F7 FF EF FF F7 FF 83'
 'FF F7 FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FB 75'
 'C7 FF FB 75 BB FF FB 75 BB FF FB 75 BB FF FB 35'
 'BB FF FB 4C C7 FF FB FD FF FF FB FD FF FF FB FE'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
}
ICON_ADD_ENTRY ICON
{
```

```
 '00 00 01 00 01 00 20 20 10 00 00 00 00 00 E8 02'
 '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
 '00 00 01 00 04 00 00 00 00 00 80 02 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 80 00 00 80 00 00 00 80 80 00 80 00'
 '00 00 80 00 80 00 80 80 00 00 80 80 80 00 C0 C0'
 'C0 00 00 00 FF 00 00 FF 00 00 00 FF FF 00 FF 00'
 '00 00 FF 00 FF 00 FF FF 00 00 FF FF FF 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF'
 'FF FF FF FF FF FF FF FF FF 3F FF FF FF DF FC 1B'
 'B5 DF FD FB AD CF FD FB AD B7 FD FB AD B7 FC 39'
 'AD B7 FD FA 64 B7 FD FF EF FF FD FF EF FF FC 1F'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF EF B0'
 'C3 FF EF AE BB FF F0 6E BB FF F7 6E BB FF F7 6E'
 'BB FF FA F0 C3 FF FA FE FB FF FD FE FB FF FD FE'
 'FB FF FF FF FF FF FF FF FF FF FF FF FF FF'
}
ICON_FIND ICON
{
 '00 00 01 00 01 00 20 20 10 00 00 00 00 00 E8 02'
 '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
 '00 00 01 00 04 00 00 00 00 00 80 02 00 00 00 00'
```

```
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 80 00 00 80 00 00 00 80 80 00 80 00'
 '00 00 80 00 80 00 80 80 00 00 80 80 80 00 C0 C0'
 'C0 00 00 00 FF 00 00 FF 00 00 00 FF FF 00 FF 00'
 '00 00 FF 00 FF 00 FF FF 00 00 FF FF FF 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF EF AE C3 FF EF AE'
 'BB FF EF AE BB FF EF AE BB FF E1 A6 BB FF EF A9'
 'C3 FF EF FF FB FF EF FF FB FF E0 BF FB FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
}
SEARCH_DIALOG DIALOG 14, 38, 177, 70
STYLE WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Search"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 6, 52, 50, 14
 PUSHBUTTON "Cancel", IDCANCEL, 65, 52, 50, 14
 PUSHBUTTON "Help", IDHELP, 122, 52, 50, 14
```

```
 EDITTEXT IDSEARCH_ENTRY, 48, 23, 97, 12
 CTEXT "Enter a keyword to be searched", -1, 3, 5, 191, 13
}
ADD_DIALOG DIALOG 4, 9, 373, 193
STYLE WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Add Entry"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 6, 173, 50, 14
 PUSHBUTTON "Cancel", IDCANCEL, 62, 173, 50, 14
 PUSHBUTTON "Help", IDHELP, 120, 174, 50, 14
 EDITTEXT ADD_KW1, 3, 23, 80, 13
 EDITTEXT ADD_KW2, 3, 37, 80, 13
 EDITTEXT ADD_KW3, 3, 51, 80, 13
 EDITTEXT ADD_KW4, 84, 23, 80, 13
 EDITTEXT ADD_KW5, 84, 37, 80, 13
 EDITTEXT ADD_KW6, 84, 51, 80, 13
 EDITTEXT ADD_KW7, 165, 23, 80, 13
 EDITTEXT ADD_KW8, 165, 37, 80, 13
 EDITTEXT ADD_KW9, 165, 51, 80, 13
 LTEXT "Enter Keywords", -1, 18, 6, 191, 13
 EDITTEXT EDIT_ITEM, 5, 86, 276, 18
 EDITTEXT ADD_DESCRIPTION, 8, 128, 327, 36
 LTEXT "Enter Location", -1, 4, 71, 114, 12
 LTEXT "Enter Description", -1, 1, 108, 124, 15
}
DELETE_DIALOG DIALOG 6, 15, 194, 119
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Delete"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 6, 96, 50, 14
 PUSHBUTTON "Cancel", IDCANCEL, 66, 96, 50, 14
 PUSHBUTTON "Help", IDHELP, 126, 96, 50, 14
 LTEXT "Enter Item to delete", -1, 3, 3, 183, 12
 EDITTEXT IDC_EDIT1, 1, 14, 191, 43
 LTEXT "WARNING:  THIS WILL DELETE ALL RECORD OF THIS ENTRY", -1, 2, 63, 183,
19
}
ID_LISTBOX DIALOG 6, 15, 356, 121
STYLE WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Search Results"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDOK, 71, 96, 50, 14
 CONTROL "Results:", IDC_LISTBOX, "LISTBOX", LBS_STANDARD | LBS_HASSTRINGS,
14, 12, 314, 45
 LTEXT "Double Click on an item to get it's description", -1, 14, 62, 153, 20
}
```

```
BEGIN INFTLAPP.RH


//#if !defined(__inftlapp_rh)                    // Sentry use file only if it's
not already included.
//#define __inftlapp_rh

/*  Main Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftlapp.h
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Constant definitions for all resources defined in inftlapp.rc.
*/


//
// IDHELP BorButton for BWCC dialogs.
//
#define IDHELP                  998             // Id of help button


//
// Application specific definitions:
//
#define DIALOG_1  1
#define ID_LISTBOX      4020
#define IDC_LISTBOX     4010
#define DELETE_DIALOG   4007
#define SEARCH_DIALOG   4005
#define IDC_EDIT1       4006
#define ADD_DIALOG      4004
#define EDIT_ITEM       4110
#define ADD_DESCRIPTION 4111
#define ADD_KW9 4109
#define ADD_KW8 4108
#define ADD_KW1 4101
#define ADD_KW2 4102
#define ADD_KW3 4103
#define ADD_KW4 4104
#define ADD_KW5 4105
#define ADD_KW6 4107
#define ADD_KW7 4106
#define IDSEARCH_ENTRY  4002
#define IDHELP_SEARCH   4001
#define ICON_PROG 1
#define ICON_ADD_ENTRY  2
```

```
#define ICON_FIND       3
#define IDI_MDIAPPLICATION      1001            // Application icon
#define IDI_DOC                 1002            // MDI child window icon

#define MDI_MENU                100             // Menu resource ID and
Accelerator IDs
#define CM_DELETE       1010
#define CM_SEARCH       1011
#define CM_ADDITEM      1012

//
// CM_FILEnnnn commands (include\owl\editfile.rh except for
CM_FILEPRINTPREVIEW)
//
#define CM_MDIFILENEW           24331
#define CM_MDIFILEOPEN          24332
#define CM_FILECLOSE            24339
#define CM_FILESAVE             24333
#define CM_FILESAVEAS           24334
#define CM_FILEREVERT           24335
#define CM_VIEWCREATE           24341
#define CM_FILEPRINT            24337
#define CM_FILEPRINTERSETUP     24338
#define CM_FILEPRINTPREVIEW     24340


//
// Window commands (include\owl\windows.rh)
//
#define CM_EXIT                 24310


//
// CM_EDITnnnn commands (include\owl\edit.rh)
//
#define CM_EDITUNDO             24321
#define CM_EDITCUT              24322
#define CM_EDITCOPY             24323
#define CM_EDITPASTE            24324
#define CM_EDITDELETE           24325
#define CM_EDITCLEAR            24326


//
// Search menu commands (include\owl\editsear.rh)
//
#define CM_EDITFIND             24351
#define CM_EDITREPLACE          24352
#define CM_EDITFINDNEXT         24353


//
// Windows menu commands (include\owl\mdi.rh)
//
```

```
#define CM_CASCADECHILDREN      24361
#define CM_TILECHILDREN         24362
#define CM_TILECHILDRENHORIZ    24363
#define CM_ARRANGEICONS         24364
#define CM_CLOSECHILDREN        24365
#define CM_CREATECHILD          24366


//
// Help menu commands.
//
#define CM_HELPCONTENTS         24381
#define CM_HELPUSING            24382
#define CM_HELPABOUT            24389


// Context sensitive help cursor.
#define IDC_HELPCURSOR          24000


//
// About Dialogs
//
#define IDD_ABOUT            22000
#define IDC_VERSION             22001
#define IDC_COPYRIGHT           22002
#define IDC_DEBUG               22003

//
// OWL defined strings
//

// Statusbar
#define IDS_MODES               32530

// EditFile
#define IDS_UNABLEREAD          32551
#define IDS_UNABLEWRITE         32552
#define IDS_FILECHANGED         32553
#define IDS_FILEFILTER          32554

// EditSearch
#define IDS_CANNOTFIND          32540


//
// General & application exception messages (include\owl\except.rh)
//
#define IDS_UNKNOWNEXCEPTION    32767
#define IDS_OWLEXCEPTION        32766
#define IDS_OKTORESUME          32765
#define IDS_UNHANDLEDXMSG       32764
#define IDS_UNKNOWNERROR        32763
#define IDS_NOAPP               32762
```

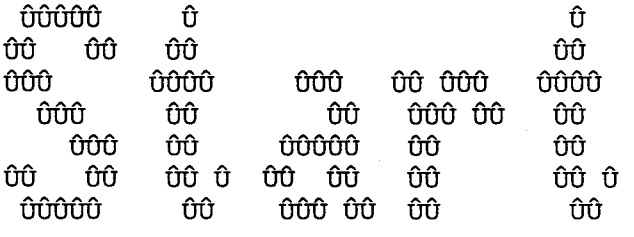
```
#define IDS_OUTOFMEMORY          32761
#define IDS_INVALIDMODULE        32760
#define IDS_INVALIDMAINWINDOW    32759

//
// Owl 1 compatibility messages
//
#define IDS_INVALIDWINDOW        32756
#define IDS_INVALIDCHILDWINDOW   32755
#define IDS_INVALIDCLIENTWINDOW  32754

//
// TXWindow messages
//
#define IDS_CLASSREGISTERFAIL    32749
#define IDS_CHILDREGISTERFAIL    32748
#define IDS_WINDOWCREATEFAIL     32747
#define IDS_WINDOWEXECUTEFAIL    32746
#define IDS_CHILDCREATEFAIL      32745

#define IDS_MENUFAILURE          32744
#define IDS_VALIDATORSYNTAX      32743
#define IDS_PRINTERERROR         32742

#define IDS_LAYOUTINCOMPLETE     32741
#define IDS_LAYOUTBADRELWIN      32740

//
// TXGdi messages
//
#define IDS_GDIFAILURE           32739
#define IDS_GDIALLOCFAIL         32738
#define IDS_GDICREATEFAIL        32737
#define IDS_GDIRESLOADFAIL       32736
#define IDS_GDIFILEREADFAIL      32735
#define IDS_GDIDELETEFAIL        32734
#define IDS_GDIDESTROYFAIL       32733
#define IDS_INVALIDDIBHANDLE     32732


// DocView (include\owl\docview.rc)
#define IDS_DOCMANAGERFILE       32500
#define IDS_DOCLIST              32501
#define IDS_VIEWLIST             32502
#define IDS_UNTITLED             32503
#define IDS_UNABLEOPEN           32504
#define IDS_UNABLECLOSE          32505
#define IDS_READERROR            32506
#define IDS_WRITEERROR           32507
#define IDS_DOCCHANGED           32508
#define IDS_NOTCHANGED           32509
#define IDS_NODOCMANAGER         32510
#define IDS_NOMEMORYFORVIEW      32511
#define IDS_DUPLICATEDOC         32512
```

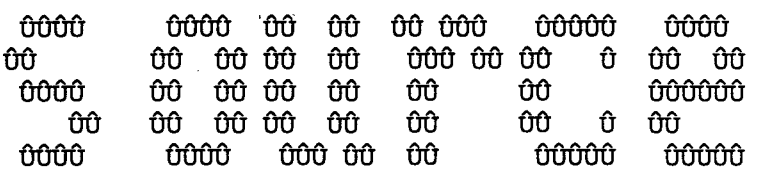
```
//
// Printing error message string resource IDs (include\owl\printer.rh)
//
#define IDS_PRNON               32590
#define IDS_PRNERRORTEMPLATE    32591
#define IDS_PRNOUTOFMEMORY      32592
#define IDS_PRNOUTOFDISK        32593
#define IDS_PRNCANCEL           32594
#define IDS_PRNMGRABORT         32595
#define IDS_PRNGENERROR         32596
#define IDS_PRNERRORCAPTION     32597


//
// Printer abort dialog & control IDs
//
#define IDD_ABORTDIALOG         32599
#define ID_TITLE                101
#define ID_DEVICE               102
#define ID_PORT                 103


//
// Print Preview
//
#define APX_PPR_PREVIOUS        24500
#define APX_PPR_NEXT            24501
#define APX_PPR_ONEUP           24502
#define APX_PPR_TWOUP           24503
#define APX_PPR_CURRPAGE        24504


// TInputDialog DIALOG resource (include\owl\inputdia.rh)
#define IDD_INPUTDIALOG         32514
#define ID_PROMPT               4091
#define ID_INPUT                4090


// TSlider bitmaps (horizontal and vertical) (include\owl\slider.rh)
#define IDB_HSLIDERTHUMB        32000
#define IDB_VSLIDERTHUMB        32001


// Validation messages (include\owl\validate.rh)
#define IDS_VALPXPCONFORM       32520
#define IDS_VALINVALIDCHAR      32521
#define IDS_VALNOTINRANGE       32522
#define IDS_VALNOTINLIST        32523


//#endif         // __inftlapp_rh sentry.
```

Appendix C BEGIN KEYWORDS.CPP

```
////////////////////////////////////////////////////////
//      Implementation file of database class: KEYWORDS.
//
//      Source generated by: CSDBGEN version 1.2.b.
//      Date of generation:  Monday, 1 May 1995.
//      Time of generation:  21:32:47.
//
//
//      The next lines represent the database definition
//      file used as input for CSDBGEN.
//
//
////////////// Start of the .def file /////////////////
/*
class: KEYWORDS
record: KEYWORDS_record
file: it_keywd
field: keyword s 20 Y
field: num_of_items i
field: places_tag_1 i
field: places_tag_2 i
field: places_tag_3 i
field: places_tag_4 i
field: places_tag_5 i
field: places_tag_6 i
field: places_tag_7 i
field: places_tag_8 i
field: places_tag_9 i
field: places_tag_10 i
field: places_tag_11 i
field: places_tag_12 i
field: places_tag_13 i
field: places_tag_14 i
field: places_tag_15 i
field: places_tag_16 i
field: places_tag_17 i
field: places_tag_18 i
field: places_tag_19 i
field: places_tag_20 i
field: places_tag_21 i
field: places_tag_22 i
field: places_tag_23 i
field: places_tag_24 i
field: places_tag_25 i
field: places_tag_26 i
field: places_tag_27 i
field: places_tag_28 i
field: places_tag_29 i
field: places_tag_30 i
field: places_tag_31 i
```

```
field: places_tag_32 i
field: places_tag_33 i
field: places_tag_34 i
field: places_tag_35 i
field: places_tag_36 i
field: places_tag_37 i
field: places_tag_38 i
field: places_tag_39 i
field: places_tag_40 i
field: places_tag_41 i
field: places_tag_42 i
field: places_tag_43 i
field: places_tag_44 i
field: places_tag_45 i
field: places_tag_46 i
field: places_tag_47 i
field: places_tag_48 i
field: places_tag_49 i
field: places_tag_50 i
field: places_tag_51 i
field: places_tag_52 i
field: places_tag_53 i
field: places_tag_54 i
field: places_tag_55 i
field: places_tag_56 i
field: places_tag_57 i
field: places_tag_58 i
field: places_tag_59 i
field: places_tag_60 i
field: places_tag_61 i
field: places_tag_62 i
field: places_tag_63 i
field: places_tag_64 i
field: places_tag_65 i
field: places_tag_66 i
field: places_tag_67 i
field: places_tag_68 i
field: places_tag_69 i
field: places_tag_70 i
*/
/////////////// End of .def file
////////////////////////////////////////////////
/*
```





```
*/
////////////////////////////////////////////////////////////////////////////

#include "keyword.h"

extern unsigned _stklen=7000;  //A large stack is needed


////////////////////////////////// Constructor ////////////////////////////////
KEYWORDS::KEYWORDS(void)
{
    is_open=FALSE;
    current=1;
}

////////////////////////////////// reindex ////////////////////////////////////
void KEYWORDS::reindex(void)
{
    U32 l=current;
    KEYWORDS_record *rp;
    in1.empty();
    for(current=numrec(); current>0; current--)
    {
       rp=(KEYWORDS_record *)db.locate_rec(current);
       in1.insert(rp->_keyword,&current);
    }
    current=l;
}

/////////////// go_to ////////////////////////////////////////////
void KEYWORDS::go_to(long n)
{
    if(order()!=UNSORTED)
    {
      csmess_p(7110,"KEYWORDS");
      return;
    }
    write_rec();
    current=max(min(n,db.numrec()),1);
    read_rec();
}

/////////////// append blank//////////////////////////////////////
void KEYWORDS::append_blank(void)
{
```

```
        append();
        in1.insert(rec._keyword,&current);
}

//////////////////////////////// append ////////////////////////////////////
// This function doesn't update the indexes, which can save some
// disk I/O because you are likely to alter the fields
// immediately after you have appended the record.
// However, if you have an index on a field you don't update, this
// record will NOT appear in that particular index!
// The 'append_blank' function does update all indexes, which
// makes it a safer, but slower option.
//
void KEYWORDS::append(void)
{
    write_rec();
    memset(&rec,0,sizeof(KEYWORDS_record));
    current=db.append_rec(&rec);
    recp=(KEYWORDS_record *)db.locate_rec(current);
    dirty=TRUE;
}

//////////////////////////////// open /////////////////////////////////////
void KEYWORDS::open(void)
{
     if(is_open) return;
     int needs_reindex=FALSE;
     dirty=FALSE;

#ifdef _Windows
     int fre=300/3; //Use 300 Kb for buffers. You may increase this.
#else
     int fre=(int)(coreleft()-100000L)/3/1024;
     fre=max(fre,0);
#endif

     if(!db.open("it_keywd.dbf",fre))
     {
         csmess_disp("FATAL: Can't open database it_keywd.dbf.");
         exit(1);
     }
     if(db.lengthrec()!=sizeof(KEYWORDS_record))
     {
         csmess_disp("FATAL: wrong record size.\n\rProbably wrong or old
database file.");
         db.close();
         exit(1);
     }
     if(!file_exist("it_key01.idx"))
     {
       in1.multiple_keys(TRUE);
       in1.define("it_key01.idx",KEYWORD_LENGTH+1,sizeof(long));
       needs_reindex=TRUE;
     }
```

```
     in1.open("it_key01.idx",fre*2);

     if(needs_reindex) reindex();
     is_open=TRUE;
     if(numrec()==0) append_blank();
     else            read_rec();

     order(UNSORTED);
}

////////////////////////////////// close ////////////////////////////////////
void KEYWORDS::close(void)
{
     if(!is_open) return;
     write_rec();
     db.close();
     in1.close();
     is_open=FALSE;
}

////////////////////////////////// define ///////////////////////////////////
void KEYWORDS::define(void)
{
     db.define("it_keywd.dbf",sizeof(KEYWORDS_record));
     in1.multiple_keys(TRUE);
     in1.define("it_key01.idx",KEYWORD_LENGTH+1,sizeof(long));
}

////////////////////////////////// pack /////////////////////////////////////
void KEYWORDS::pack(void)
{
     write_rec();
     db.pack();
     reindex();
     if(numrec()==0) append_blank();
     top();
}

////////////////////////////////// skip /////////////////////////////////////
int  KEYWORDS::skip0(int delta)
{
     long old_current=current;
     current=max(min(current+delta,db.numrec()),1);
     return current-old_current;
}

int  KEYWORDS::skip(int delta)
{
     int rc;
     write_rec();
     rc=(this->*skip_fun)(delta);
     read_rec();
     return rc;
}
```

```
//////////////////////////////// order ////////////////////////////////////
void KEYWORDS::order(int nr)
{
     switch(nr)
     {
         case 0:         //Unsorted
                  bof_fun   =&KEYWORDS::bof0;
                  eof_fun   =&KEYWORDS::eof0;
                  skip_fun  =&KEYWORDS::skip0;
                  top_fun   =&KEYWORDS::top0;
                  bottom_fun=&KEYWORDS::bottom0;
                  search_fun=&KEYWORDS::search0;
                  break;
         case 1:         //Index on field keyword
                  bof_fun   =&KEYWORDS::bof1;
                  eof_fun   =&KEYWORDS::eof1;
                  skip_fun  =&KEYWORDS::skip1;
                  top_fun   =&KEYWORDS::top1;
                  bottom_fun=&KEYWORDS::bottom1;
                  search_fun=&KEYWORDS::search1;
                  break;
         default: return; // Function called with wrong parameter.
     }
     iOrder=nr;
     top();
}
//////////////////////////////writing record  ////////////////////////////////
void KEYWORDS::write_rec2(void)
{
    if(strcmp(recp->_keyword,rec._keyword))
    {
      in1.delet(recp->_keyword,&current);
      in1.insert(rec._keyword,&current);
    }
    db.write_rec(current,&rec);
    dirty=FALSE;
}

//////////////////////////////// export ////////////////////////////////////
int  KEYWORDS::export(char *s)
{
     FILE *fo=fopen(s,"w");
     if(fo==NULL) return FALSE;

     write_rec();
     fprintf(fo,"class:  KEYWORDS");
     fprintf(fo,"\nrecord: KEYWORDS_record");
     fprintf(fo,"\nfile:   it_keywd");
     fprintf(fo,"\nfield:  keyword s 20 Y");
     fprintf(fo,"\nfield:  num_of_items i  ");
     fprintf(fo,"\nfield:  places_tag_1 i  ");
     fprintf(fo,"\nfield:  places_tag_2 i  ");
     fprintf(fo,"\nfield:  places_tag_3 i  ");
```

```
fprintf(fo,"\nfield:  places_tag_4 i  ");
fprintf(fo,"\nfield:  places_tag_5 i  ");
fprintf(fo,"\nfield:  places_tag_6 i  ");
fprintf(fo,"\nfield:  places_tag_7 i  ");
fprintf(fo,"\nfield:  places_tag_8 i  ");
fprintf(fo,"\nfield:  places_tag_9 i  ");
fprintf(fo,"\nfield:  places_tag_10 i  ");
fprintf(fo,"\nfield:  places_tag_11 i  ");
fprintf(fo,"\nfield:  places_tag_12 i  ");
fprintf(fo,"\nfield:  places_tag_13 i  ");
fprintf(fo,"\nfield:  places_tag_14 i  ");
fprintf(fo,"\nfield:  places_tag_15 i  ");
fprintf(fo,"\nfield:  places_tag_16 i  ");
fprintf(fo,"\nfield:  places_tag_17 i  ");
fprintf(fo,"\nfield:  places_tag_18 i  ");
fprintf(fo,"\nfield:  places_tag_19 i  ");
fprintf(fo,"\nfield:  places_tag_20 i  ");
fprintf(fo,"\nfield:  places_tag_21 i  ");
fprintf(fo,"\nfield:  places_tag_22 i  ");
fprintf(fo,"\nfield:  places_tag_23 i  ");
fprintf(fo,"\nfield:  places_tag_24 i  ");
fprintf(fo,"\nfield:  places_tag_25 i  ");
fprintf(fo,"\nfield:  places_tag_26 i  ");
fprintf(fo,"\nfield:  places_tag_27 i  ");
fprintf(fo,"\nfield:  places_tag_28 i  ");
fprintf(fo,"\nfield:  places_tag_29 i  ");
fprintf(fo,"\nfield:  places_tag_30 i  ");
fprintf(fo,"\nfield:  places_tag_31 i  ");
fprintf(fo,"\nfield:  places_tag_32 i  ");
fprintf(fo,"\nfield:  places_tag_33 i  ");
fprintf(fo,"\nfield:  places_tag_34 i  ");
fprintf(fo,"\nfield:  places_tag_35 i  ");
fprintf(fo,"\nfield:  places_tag_36 i  ");
fprintf(fo,"\nfield:  places_tag_37 i  ");
fprintf(fo,"\nfield:  places_tag_38 i  ");
fprintf(fo,"\nfield:  places_tag_39 i  ");
fprintf(fo,"\nfield:  places_tag_40 i  ");
fprintf(fo,"\nfield:  places_tag_41 i  ");
fprintf(fo,"\nfield:  places_tag_42 i  ");
fprintf(fo,"\nfield:  places_tag_43 i  ");
fprintf(fo,"\nfield:  places_tag_44 i  ");
fprintf(fo,"\nfield:  places_tag_45 i  ");
fprintf(fo,"\nfield:  places_tag_46 i  ");
fprintf(fo,"\nfield:  places_tag_47 i  ");
fprintf(fo,"\nfield:  places_tag_48 i  ");
fprintf(fo,"\nfield:  places_tag_49 i  ");
fprintf(fo,"\nfield:  places_tag_50 i  ");
fprintf(fo,"\nfield:  places_tag_51 i  ");
fprintf(fo,"\nfield:  places_tag_52 i  ");
fprintf(fo,"\nfield:  places_tag_53 i  ");
fprintf(fo,"\nfield:  places_tag_54 i  ");
fprintf(fo,"\nfield:  places_tag_55 i  ");
fprintf(fo,"\nfield:  places_tag_56 i  ");
fprintf(fo,"\nfield:  places_tag_57 i  ");
```

```
fprintf(fo,"\nfield:  places_tag_58 i  ");
fprintf(fo,"\nfield:  places_tag_59 i  ");
fprintf(fo,"\nfield:  places_tag_60 i  ");
fprintf(fo,"\nfield:  places_tag_61 i  ");
fprintf(fo,"\nfield:  places_tag_62 i  ");
fprintf(fo,"\nfield:  places_tag_63 i  ");
fprintf(fo,"\nfield:  places_tag_64 i  ");
fprintf(fo,"\nfield:  places_tag_65 i  ");
fprintf(fo,"\nfield:  places_tag_66 i  ");
fprintf(fo,"\nfield:  places_tag_67 i  ");
fprintf(fo,"\nfield:  places_tag_68 i  ");
fprintf(fo,"\nfield:  places_tag_69 i  ");
fprintf(fo,"\nfield:  places_tag_70 i  ");

if(ferror(fo)) { fclose(fo); return FALSE; }

KEYWORDS_record *recp;
for(long l=numrec();l>0;l--)
{
   recp=( KEYWORDS_record * )db.locate_rec(l);
   fprintf(fo,"\n%c",12);
   fprintf(fo,"\n%s",recp->_keyword);
   fprintf(fo,"\n%d",recp->_num_of_items);
   fprintf(fo,"\n%d",recp->_places_tag_1);
   fprintf(fo,"\n%d",recp->_places_tag_2);
   fprintf(fo,"\n%d",recp->_places_tag_3);
   fprintf(fo,"\n%d",recp->_places_tag_4);
   fprintf(fo,"\n%d",recp->_places_tag_5);
   fprintf(fo,"\n%d",recp->_places_tag_6);
   fprintf(fo,"\n%d",recp->_places_tag_7);
   fprintf(fo,"\n%d",recp->_places_tag_8);
   fprintf(fo,"\n%d",recp->_places_tag_9);
   fprintf(fo,"\n%d",recp->_places_tag_10);
   fprintf(fo,"\n%d",recp->_places_tag_11);
   fprintf(fo,"\n%d",recp->_places_tag_12);
   fprintf(fo,"\n%d",recp->_places_tag_13);
   fprintf(fo,"\n%d",recp->_places_tag_14);
   fprintf(fo,"\n%d",recp->_places_tag_15);
   fprintf(fo,"\n%d",recp->_places_tag_16);
   fprintf(fo,"\n%d",recp->_places_tag_17);
   fprintf(fo,"\n%d",recp->_places_tag_18);
   fprintf(fo,"\n%d",recp->_places_tag_19);
   fprintf(fo,"\n%d",recp->_places_tag_20);
   fprintf(fo,"\n%d",recp->_places_tag_21);
   fprintf(fo,"\n%d",recp->_places_tag_22);
   fprintf(fo,"\n%d",recp->_places_tag_23);
   fprintf(fo,"\n%d",recp->_places_tag_24);
   fprintf(fo,"\n%d",recp->_places_tag_25);
   fprintf(fo,"\n%d",recp->_places_tag_26);
   fprintf(fo,"\n%d",recp->_places_tag_27);
   fprintf(fo,"\n%d",recp->_places_tag_28);
   fprintf(fo,"\n%d",recp->_places_tag_29);
   fprintf(fo,"\n%d",recp->_places_tag_30);
   fprintf(fo,"\n%d",recp->_places_tag_31);
```

```
            fprintf(fo,"\n%d",recp->_places_tag_32);
            fprintf(fo,"\n%d",recp->_places_tag_33);
            fprintf(fo,"\n%d",recp->_places_tag_34);
            fprintf(fo,"\n%d",recp->_places_tag_35);
            fprintf(fo,"\n%d",recp->_places_tag_36);
            fprintf(fo,"\n%d",recp->_places_tag_37);
            fprintf(fo,"\n%d",recp->_places_tag_38);
            fprintf(fo,"\n%d",recp->_places_tag_39);
            fprintf(fo,"\n%d",recp->_places_tag_40);
            fprintf(fo,"\n%d",recp->_places_tag_41);
            fprintf(fo,"\n%d",recp->_places_tag_42);
            fprintf(fo,"\n%d",recp->_places_tag_43);
            fprintf(fo,"\n%d",recp->_places_tag_44);
            fprintf(fo,"\n%d",recp->_places_tag_45);
            fprintf(fo,"\n%d",recp->_places_tag_46);
            fprintf(fo,"\n%d",recp->_places_tag_47);
            fprintf(fo,"\n%d",recp->_places_tag_48);
            fprintf(fo,"\n%d",recp->_places_tag_49);
            fprintf(fo,"\n%d",recp->_places_tag_50);
            fprintf(fo,"\n%d",recp->_places_tag_51);
            fprintf(fo,"\n%d",recp->_places_tag_52);
            fprintf(fo,"\n%d",recp->_places_tag_53);
            fprintf(fo,"\n%d",recp->_places_tag_54);
            fprintf(fo,"\n%d",recp->_places_tag_55);
            fprintf(fo,"\n%d",recp->_places_tag_56);
            fprintf(fo,"\n%d",recp->_places_tag_57);
            fprintf(fo,"\n%d",recp->_places_tag_58);
            fprintf(fo,"\n%d",recp->_places_tag_59);
            fprintf(fo,"\n%d",recp->_places_tag_60);
            fprintf(fo,"\n%d",recp->_places_tag_61);
            fprintf(fo,"\n%d",recp->_places_tag_62);
            fprintf(fo,"\n%d",recp->_places_tag_63);
            fprintf(fo,"\n%d",recp->_places_tag_64);
            fprintf(fo,"\n%d",recp->_places_tag_65);
            fprintf(fo,"\n%d",recp->_places_tag_66);
            fprintf(fo,"\n%d",recp->_places_tag_67);
            fprintf(fo,"\n%d",recp->_places_tag_68);
            fprintf(fo,"\n%d",recp->_places_tag_69);
            fprintf(fo,"\n%d",recp->_places_tag_70);
            fprintf(fo,"\n"); //Additional linefeed, to avoid trouble!

            if(ferror(fo)) { fclose(fo); return FALSE; }
        }
        return !fclose(fo);
}

//////////////////////////////// import ////////////////////////////////////
int KEYWORDS::import(char *s)
{

        FILE *fr=fopen(s,"r");
        if(fr==NULL) return FALSE;
```

```
#define MAX_NUM_FIELDS  100   //Increase this to allow more fields
#define MAX_FIELD_LEN   500   //Increase this to allow longer fields

      int *finu;
      finu=(int *)malloc(MAX_NUM_FIELDS*sizeof(int));
      if(finu==NULL) { fclose(fr); return FALSE; }

      char *fibu;
      fibu=(char *)malloc(MAX_FIELD_LEN);
      if(fibu==NULL) { fclose(fr); free(finu); return FALSE; }
      *fibu=0;

      char *fipo=fibu+strlen("field:");
      char *cp;
      int  ifieldnr=0;
      int  ofieldnr;

      memset(finu,0,MAX_NUM_FIELDS*sizeof(int));

      fgets(fibu,MAX_FIELD_LEN,fr);
      while(!strchr(fibu,12))
      {
        strlwr(fibu);
        notabs(fibu);
        trim_string(fibu);
        if(strstr(fibu,"field:"))
        {
           ifieldnr++;
           trim_string(fipo);
           if((cp=strchr(fipo,' '))!=NULL) *cp=0;
           if     (!strcmp(fipo,"keyword"))     ofieldnr=1;
           else if(!strcmp(fipo,"num_of_items"))      ofieldnr=2;
           else if(!strcmp(fipo,"places_tag_1"))      ofieldnr=3;
           else if(!strcmp(fipo,"places_tag_2"))      ofieldnr=4;
           else if(!strcmp(fipo,"places_tag_3"))      ofieldnr=5;
           else if(!strcmp(fipo,"places_tag_4"))      ofieldnr=6;
           else if(!strcmp(fipo,"places_tag_5"))      ofieldnr=7;
           else if(!strcmp(fipo,"places_tag_6"))      ofieldnr=8;
           else if(!strcmp(fipo,"places_tag_7"))      ofieldnr=9;
           else if(!strcmp(fipo,"places_tag_8"))      ofieldnr=10;
           else if(!strcmp(fipo,"places_tag_9"))      ofieldnr=11;
           else if(!strcmp(fipo,"places_tag_10"))     ofieldnr=12;
           else if(!strcmp(fipo,"places_tag_11"))     ofieldnr=13;
           else if(!strcmp(fipo,"places_tag_12"))     ofieldnr=14;
           else if(!strcmp(fipo,"places_tag_13"))     ofieldnr=15;
           else if(!strcmp(fipo,"places_tag_14"))     ofieldnr=16;
           else if(!strcmp(fipo,"places_tag_15"))     ofieldnr=17;
           else if(!strcmp(fipo,"places_tag_16"))     ofieldnr=18;
           else if(!strcmp(fipo,"places_tag_17"))     ofieldnr=19;
           else if(!strcmp(fipo,"places_tag_18"))     ofieldnr=20;
           else if(!strcmp(fipo,"places_tag_19"))     ofieldnr=21;
           else if(!strcmp(fipo,"places_tag_20"))     ofieldnr=22;
           else if(!strcmp(fipo,"places_tag_21"))     ofieldnr=23;
           else if(!strcmp(fipo,"places_tag_22"))     ofieldnr=24;
```

```
        else if(!strcmp(fipo,"places_tag_23"))      ofieldnr=25;
        else if(!strcmp(fipo,"places_tag_24"))      ofieldnr=26;
        else if(!strcmp(fipo,"places_tag_25"))      ofieldnr=27;
        else if(!strcmp(fipo,"places_tag_26"))      ofieldnr=28;
        else if(!strcmp(fipo,"places_tag_27"))      ofieldnr=29;
        else if(!strcmp(fipo,"places_tag_28"))      ofieldnr=30;
        else if(!strcmp(fipo,"places_tag_29"))      ofieldnr=31;
        else if(!strcmp(fipo,"places_tag_30"))      ofieldnr=32;
        else if(!strcmp(fipo,"places_tag_31"))      ofieldnr=33;
        else if(!strcmp(fipo,"places_tag_32"))      ofieldnr=34;
        else if(!strcmp(fipo,"places_tag_33"))      ofieldnr=35;
        else if(!strcmp(fipo,"places_tag_34"))      ofieldnr=36;
        else if(!strcmp(fipo,"places_tag_35"))      ofieldnr=37;
        else if(!strcmp(fipo,"places_tag_36"))      ofieldnr=38;
        else if(!strcmp(fipo,"places_tag_37"))      ofieldnr=39;
        else if(!strcmp(fipo,"places_tag_38"))      ofieldnr=40;
        else if(!strcmp(fipo,"places_tag_39"))      ofieldnr=41;
        else if(!strcmp(fipo,"places_tag_40"))      ofieldnr=42;
        else if(!strcmp(fipo,"places_tag_41"))      ofieldnr=43;
        else if(!strcmp(fipo,"places_tag_42"))      ofieldnr=44;
        else if(!strcmp(fipo,"places_tag_43"))      ofieldnr=45;
        else if(!strcmp(fipo,"places_tag_44"))      ofieldnr=46;
        else if(!strcmp(fipo,"places_tag_45"))      ofieldnr=47;
        else if(!strcmp(fipo,"places_tag_46"))      ofieldnr=48;
        else if(!strcmp(fipo,"places_tag_47"))      ofieldnr=49;
        else if(!strcmp(fipo,"places_tag_48"))      ofieldnr=50;
        else if(!strcmp(fipo,"places_tag_49"))      ofieldnr=51;
        else if(!strcmp(fipo,"places_tag_50"))      ofieldnr=52;
        else if(!strcmp(fipo,"places_tag_51"))      ofieldnr=53;
        else if(!strcmp(fipo,"places_tag_52"))      ofieldnr=54;
        else if(!strcmp(fipo,"places_tag_53"))      ofieldnr=55;
        else if(!strcmp(fipo,"places_tag_54"))      ofieldnr=56;
        else if(!strcmp(fipo,"places_tag_55"))      ofieldnr=57;
        else if(!strcmp(fipo,"places_tag_56"))      ofieldnr=58;
        else if(!strcmp(fipo,"places_tag_57"))      ofieldnr=59;
        else if(!strcmp(fipo,"places_tag_58"))      ofieldnr=60;
        else if(!strcmp(fipo,"places_tag_59"))      ofieldnr=61;
        else if(!strcmp(fipo,"places_tag_60"))      ofieldnr=62;
        else if(!strcmp(fipo,"places_tag_61"))      ofieldnr=63;
        else if(!strcmp(fipo,"places_tag_62"))      ofieldnr=64;
        else if(!strcmp(fipo,"places_tag_63"))      ofieldnr=65;
        else if(!strcmp(fipo,"places_tag_64"))      ofieldnr=66;
        else if(!strcmp(fipo,"places_tag_65"))      ofieldnr=67;
        else if(!strcmp(fipo,"places_tag_66"))      ofieldnr=68;
        else if(!strcmp(fipo,"places_tag_67"))      ofieldnr=69;
        else if(!strcmp(fipo,"places_tag_68"))      ofieldnr=70;
        else if(!strcmp(fipo,"places_tag_69"))      ofieldnr=71;
        else if(!strcmp(fipo,"places_tag_70"))      ofieldnr=72;
        else ofieldnr=0;
        finu[ifieldnr]=ofieldnr;
    }
    fgets(fibu,MAX_FIELD_LEN,fr);
}
```

```
for(;;)
{
  if(!strchr(fibu,12))
  {
     ifieldnr++;
     if(ifieldnr<MAX_NUM_FIELDS)
       switch(finu[ifieldnr])
       {
          case 0:  break;
          case 1:
                   fibu[KEYWORD_LENGTH]=0;
                   keyword(fibu);
                   break;
          case 2:
                   num_of_items(atoi(fibu));
                   break;
          case 3:
                   places_tag_1(atoi(fibu));
                   break;
          case 4:
                   places_tag_2(atoi(fibu));
                   break;
          case 5:
                   places_tag_3(atoi(fibu));
                   break;
          case 6:
                   places_tag_4(atoi(fibu));
                   break;
          case 7:
                   places_tag_5(atoi(fibu));
                   break;
          case 8:
                   places_tag_6(atoi(fibu));
                   break;
          case 9:
                   places_tag_7(atoi(fibu));
                   break;
          case 10:
                   places_tag_8(atoi(fibu));
                   break;
          case 11:
                   places_tag_9(atoi(fibu));
                   break;
          case 12:
                   places_tag_10(atoi(fibu));
                   break;
          case 13:
                   places_tag_11(atoi(fibu));
                   break;
          case 14:
                   places_tag_12(atoi(fibu));
                   break;
```

```
case 15:
        places_tag_13(atoi(fibu));
        break;
case 16:
        places_tag_14(atoi(fibu));
        break;
case 17:
        places_tag_15(atoi(fibu));
        break;
case 18:
        places_tag_16(atoi(fibu));
        break;
case 19:
        places_tag_17(atoi(fibu));
        break;
case 20:
        places_tag_18(atoi(fibu));
        break;
case 21:
        places_tag_19(atoi(fibu));
        break;
case 22:
        places_tag_20(atoi(fibu));
        break;
case 23:
        places_tag_21(atoi(fibu));
        break;
case 24:
        places_tag_22(atoi(fibu));
        break;
case 25:
        places_tag_23(atoi(fibu));
        break;
case 26:
        places_tag_24(atoi(fibu));
        break;
case 27:
        places_tag_25(atoi(fibu));
        break;
case 28:
        places_tag_26(atoi(fibu));
        break;
case 29:
        places_tag_27(atoi(fibu));
        break;
case 30:
        places_tag_28(atoi(fibu));
        break;
case 31:
        places_tag_29(atoi(fibu));
        break;
case 32:
        places_tag_30(atoi(fibu));
        break;
```

```
case 33:
        places_tag_31(atoi(fibu));
        break;
case 34:
        places_tag_32(atoi(fibu));
        break;
case 35:
        places_tag_33(atoi(fibu));
        break;
case 36:
        places_tag_34(atoi(fibu));
        break;
case 37:
        places_tag_35(atoi(fibu));
        break;
case 38:
        places_tag_36(atoi(fibu));
        break;
case 39:
        places_tag_37(atoi(fibu));
        break;
case 40:
        places_tag_38(atoi(fibu));
        break;
case 41:
        places_tag_39(atoi(fibu));
        break;
case 42:
        places_tag_40(atoi(fibu));
        break;
case 43:
        places_tag_41(atoi(fibu));
        break;
case 44:
        places_tag_42(atoi(fibu));
        break;
case 45:
        places_tag_43(atoi(fibu));
        break;
case 46:
        places_tag_44(atoi(fibu));
        break;
case 47:
        places_tag_45(atoi(fibu));
        break;
case 48:
        places_tag_46(atoi(fibu));
        break;
case 49:
        places_tag_47(atoi(fibu));
        break;
case 50:
        places_tag_48(atoi(fibu));
        break;
```

```
case 51:
        places_tag_49(atoi(fibu));
        break;
case 52:
        places_tag_50(atoi(fibu));
        break;
case 53:
        places_tag_51(atoi(fibu));
        break;
case 54:
        places_tag_52(atoi(fibu));
        break;
case 55:
        places_tag_53(atoi(fibu));
        break;
case 56:
        places_tag_54(atoi(fibu));
        break;
case 57:
        places_tag_55(atoi(fibu));
        break;
case 58:
        places_tag_56(atoi(fibu));
        break;
case 59:
        places_tag_57(atoi(fibu));
        break;
case 60:
        places_tag_58(atoi(fibu));
        break;
case 61:
        places_tag_59(atoi(fibu));
        break;
case 62:
        places_tag_60(atoi(fibu));
        break;
case 63:
        places_tag_61(atoi(fibu));
        break;
case 64:
        places_tag_62(atoi(fibu));
        break;
case 65:
        places_tag_63(atoi(fibu));
        break;
case 66:
        places_tag_64(atoi(fibu));
        break;
case 67:
        places_tag_65(atoi(fibu));
        break;
case 68:
        places_tag_66(atoi(fibu));
        break;
```

```
                case 69:
                        places_tag_67(atoi(fibu));
                        break;
                case 70:
                        places_tag_68(atoi(fibu));
                        break;
                case 71:
                        places_tag_69(atoi(fibu));
                        break;
                case 72:
                        places_tag_70(atoi(fibu));
                        break;
            }
        }
        else
        {
           ifieldnr=0;
           append_blank();
        }
        if(feof(fr)) break;
        fgets(fibu,MAX_FIELD_LEN,fr);
        cp=fibu+(max(1,strlen(fibu))-1);
        if(*cp=='\n') *cp=0;  //removing the line feed
      }


      fclose(fr);
      free(fibu);
      free(finu);

#undef MAX_NUM_FIELDS
#undef MAX_FIELD_LEN

 return TRUE;
}

//////////////////////////////// export to dBASE compatible file. ///////////
int KEYWORDS::to_DBASE(char *s)
{

      char bufje[12];
      if(!is_open) return FALSE;

      write_rec();
      FILE *fo=fopen(s,"wb");
      if(fo==NULL) return FALSE;

      int i;

      DATE d_upda;
      d_upda.sem_jul(db.sj_updated());
      fputc(03,fo);

      fputc(d_upda.year()%100,fo);
```

```
      fputc(d_upda.month(),fo);
      fputc(d_upda.day(),fo);

      long nr_record=numrec();
      fwrite(&nr_record,sizeof(long),1,fo);
      putw(2338,fo);          //Header length
      putw(376,fo);           //Length of data record
      for(i=0;i<20;i++) fputc(0,fo); // 20 dummy bytes


// Writing definition of field keyword to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"KEYWORD");
      fwrite(bufje,11,1,fo);
      fputc('C',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(20,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field num_of_items to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"NUM_OF_ITE");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_1 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_2 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_3 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
```

```
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_4 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_5 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_6 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_7 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_8 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
```

```
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_9 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_10 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_11 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_12 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_13 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_14 to dbase file header.
      memset(bufje,0,11);
```

```
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_15 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_16 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_17 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_18 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_19 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
```

```
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_20 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_21 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_22 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_23 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_24 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes
```

```
// Writing definition of field places_tag_25 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_26 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_27 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_28 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_29 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_30 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
```

```
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_31 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_32 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_33 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_34 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_35 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
```

```
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_36 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_37 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_38 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_39 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_40 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_41 to dbase file header.
      memset(bufje,0,11);
```

```
        strcpy(bufje,"PLACES_TAG");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
        fputc(5,fo);
        fputc(0,fo);
        for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_42 to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACES_TAG");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
        fputc(5,fo);
        fputc(0,fo);
        for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_43 to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACES_TAG");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
        fputc(5,fo);
        fputc(0,fo);
        for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_44 to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACES_TAG");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
        fputc(5,fo);
        fputc(0,fo);
        for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_45 to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACES_TAG");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
        fputc(5,fo);
        fputc(0,fo);
        for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_46 to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACES_TAG");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
```

```
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_47 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_48 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_49 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_50 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_51 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes
```

```
// Writing definition of field places_tag_52 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_53 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_54 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_55 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_56 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_57 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
```

```
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_58 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_59 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_60 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_61 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_62 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
```

```
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_63 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_64 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_65 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_66 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_67 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_68 to dbase file header.
      memset(bufje,0,11);
```

```
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_69 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field places_tag_70 to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"PLACES_TAG");
      fwrite(bufje,11,1,fo);
      fputc('N',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(5,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes


      fputc(13,fo);  //Field terminator
      fputc(0,fo);

// By now we have written the definition of the
// record structure to the file header.
// From here on we will export the records.

     KEYWORDS_record *recp;
     for(long l=numrec();l>0;l--)
     {

        if(ferror(fo)) { fclose(fo); return FALSE; }
        recp=(KEYWORDS_record *)db.locate_rec(l);

        if(db.is_delet(l)) fputc(42,fo);
        else               fputc(32,fo);

////////////////////// writing field keyword ////////////
        fprintf(fo,"%-20s",recp->_keyword);
////////////////////// writing field num_of_items ////////////
        fprintf(fo,"%5d",recp->_num_of_items);
////////////////////// writing field places_tag_1 ////////////
        fprintf(fo,"%5d",recp->_places_tag_1);
////////////////////// writing field places_tag_2 ////////////
        fprintf(fo,"%5d",recp->_places_tag_2);
```

```
////////////////////// writing field places_tag_3 ////////////
        fprintf(fo,"%5d",recp->_places_tag_3);
////////////////////// writing field places_tag_4 ////////////
        fprintf(fo,"%5d",recp->_places_tag_4);
////////////////////// writing field places_tag_5 ////////////
        fprintf(fo,"%5d",recp->_places_tag_5);
////////////////////// writing field places_tag_6 ////////////
        fprintf(fo,"%5d",recp->_places_tag_6);
////////////////////// writing field places_tag_7 ////////////
        fprintf(fo,"%5d",recp->_places_tag_7);
////////////////////// writing field places_tag_8 ////////////
        fprintf(fo,"%5d",recp->_places_tag_8);
////////////////////// writing field places_tag_9 ////////////
        fprintf(fo,"%5d",recp->_places_tag_9);
////////////////////// writing field places_tag_10 ///////////
        fprintf(fo,"%5d",recp->_places_tag_10);
////////////////////// writing field places_tag_11 ///////////
        fprintf(fo,"%5d",recp->_places_tag_11);
////////////////////// writing field places_tag_12 ///////////
        fprintf(fo,"%5d",recp->_places_tag_12);
////////////////////// writing field places_tag_13 ///////////
        fprintf(fo,"%5d",recp->_places_tag_13);
////////////////////// writing field places_tag_14 ///////////
        fprintf(fo,"%5d",recp->_places_tag_14);
////////////////////// writing field places_tag_15 ///////////
        fprintf(fo,"%5d",recp->_places_tag_15);
////////////////////// writing field places_tag_16 ///////////
        fprintf(fo,"%5d",recp->_places_tag_16);
////////////////////// writing field places_tag_17 ///////////
        fprintf(fo,"%5d",recp->_places_tag_17);
////////////////////// writing field places_tag_18 ///////////
        fprintf(fo,"%5d",recp->_places_tag_18);
////////////////////// writing field places_tag_19 ///////////
        fprintf(fo,"%5d",recp->_places_tag_19);
////////////////////// writing field places_tag_20 ///////////
        fprintf(fo,"%5d",recp->_places_tag_20);
////////////////////// writing field places_tag_21 ///////////
        fprintf(fo,"%5d",recp->_places_tag_21);
////////////////////// writing field places_tag_22 ///////////
        fprintf(fo,"%5d",recp->_places_tag_22);
////////////////////// writing field places_tag_23 ///////////
        fprintf(fo,"%5d",recp->_places_tag_23);
////////////////////// writing field places_tag_24 ///////////
        fprintf(fo,"%5d",recp->_places_tag_24);
////////////////////// writing field places_tag_25 ///////////
        fprintf(fo,"%5d",recp->_places_tag_25);
////////////////////// writing field places_tag_26 ///////////
        fprintf(fo,"%5d",recp->_places_tag_26);
////////////////////// writing field places_tag_27 ///////////
        fprintf(fo,"%5d",recp->_places_tag_27);
////////////////////// writing field places_tag_28 ///////////
        fprintf(fo,"%5d",recp->_places_tag_28);
////////////////////// writing field places_tag_29 ///////////
        fprintf(fo,"%5d",recp->_places_tag_29);
```

```
////////////////////// writing field places_tag_30 ////////////
        fprintf(fo,"%5d",recp->_places_tag_30);
////////////////////// writing field places_tag_31 ////////////
        fprintf(fo,"%5d",recp->_places_tag_31);
////////////////////// writing field places_tag_32 ////////////
        fprintf(fo,"%5d",recp->_places_tag_32);
////////////////////// writing field places_tag_33 ////////////
        fprintf(fo,"%5d",recp->_places_tag_33);
////////////////////// writing field places_tag_34 ////////////
        fprintf(fo,"%5d",recp->_places_tag_34);
////////////////////// writing field places_tag_35 ////////////
        fprintf(fo,"%5d",recp->_places_tag_35);
////////////////////// writing field places_tag_36 ////////////
        fprintf(fo,"%5d",recp->_places_tag_36);
////////////////////// writing field places_tag_37 ////////////
        fprintf(fo,"%5d",recp->_places_tag_37);
////////////////////// writing field places_tag_38 ////////////
        fprintf(fo,"%5d",recp->_places_tag_38);
////////////////////// writing field places_tag_39 ////////////
        fprintf(fo,"%5d",recp->_places_tag_39);
////////////////////// writing field places_tag_40 ////////////
        fprintf(fo,"%5d",recp->_places_tag_40);
////////////////////// writing field places_tag_41 ////////////
        fprintf(fo,"%5d",recp->_places_tag_41);
////////////////////// writing field places_tag_42 ////////////
        fprintf(fo,"%5d",recp->_places_tag_42);
////////////////////// writing field places_tag_43 ////////////
        fprintf(fo,"%5d",recp->_places_tag_43);
////////////////////// writing field places_tag_44 ////////////
        fprintf(fo,"%5d",recp->_places_tag_44);
////////////////////// writing field places_tag_45 ////////////
        fprintf(fo,"%5d",recp->_places_tag_45);
////////////////////// writing field places_tag_46 ////////////
        fprintf(fo,"%5d",recp->_places_tag_46);
////////////////////// writing field places_tag_47 ////////////
        fprintf(fo,"%5d",recp->_places_tag_47);
////////////////////// writing field places_tag_48 ////////////
        fprintf(fo,"%5d",recp->_places_tag_48);
////////////////////// writing field places_tag_49 ////////////
        fprintf(fo,"%5d",recp->_places_tag_49);
////////////////////// writing field places_tag_50 ////////////
        fprintf(fo,"%5d",recp->_places_tag_50);
////////////////////// writing field places_tag_51 ////////////
        fprintf(fo,"%5d",recp->_places_tag_51);
////////////////////// writing field places_tag_52 ////////////
        fprintf(fo,"%5d",recp->_places_tag_52);
////////////////////// writing field places_tag_53 ////////////
        fprintf(fo,"%5d",recp->_places_tag_53);
////////////////////// writing field places_tag_54 ////////////
        fprintf(fo,"%5d",recp->_places_tag_54);
////////////////////// writing field places_tag_55 ////////////
        fprintf(fo,"%5d",recp->_places_tag_55);
////////////////////// writing field places_tag_56 ////////////
        fprintf(fo,"%5d",recp->_places_tag_56);
```

```
//////////////////////// writing field places_tag_57 ////////////
        fprintf(fo,"%5d",recp->_places_tag_57);
//////////////////////// writing field places_tag_58 ////////////
        fprintf(fo,"%5d",recp->_places_tag_58);
//////////////////////// writing field places_tag_59 ////////////
        fprintf(fo,"%5d",recp->_places_tag_59);
//////////////////////// writing field places_tag_60 ////////////
        fprintf(fo,"%5d",recp->_places_tag_60);
//////////////////////// writing field places_tag_61 ////////////
        fprintf(fo,"%5d",recp->_places_tag_61);
//////////////////////// writing field places_tag_62 ////////////
        fprintf(fo,"%5d",recp->_places_tag_62);
//////////////////////// writing field places_tag_63 ////////////
        fprintf(fo,"%5d",recp->_places_tag_63);
//////////////////////// writing field places_tag_64 ////////////
        fprintf(fo,"%5d",recp->_places_tag_64);
//////////////////////// writing field places_tag_65 ////////////
        fprintf(fo,"%5d",recp->_places_tag_65);
//////////////////////// writing field places_tag_66 ////////////
        fprintf(fo,"%5d",recp->_places_tag_66);
//////////////////////// writing field places_tag_67 ////////////
        fprintf(fo,"%5d",recp->_places_tag_67);
//////////////////////// writing field places_tag_68 ////////////
        fprintf(fo,"%5d",recp->_places_tag_68);
//////////////////////// writing field places_tag_69 ////////////
        fprintf(fo,"%5d",recp->_places_tag_69);
//////////////////////// writing field places_tag_70 ////////////
        fprintf(fo,"%5d",recp->_places_tag_70);
    }
    fputc(26,fo);  //End of File
    fclose(fo);

 return TRUE;
}
```

```
BEGIN KEYWORDS.H

#include "ctype.h"
#include "csa.h"
#include "csdb.h"

#define KEYWORD_LENGTH   20

////////// Indexes to be used with the 'order()' function.////////////
#define UNSORTED   0
#define KEYWORD_INDEX    1

typedef struct
{
   char    _keyword[KEYWORD_LENGTH+1];
   int     _num_of_items;
   int     _places_tag_1;
   int     _places_tag_2;
   int     _places_tag_3;
   int     _places_tag_4;
   int     _places_tag_5;
   int     _places_tag_6;
   int     _places_tag_7;
   int     _places_tag_8;
   int     _places_tag_9;
   int     _places_tag_10;
   int     _places_tag_11;
   int     _places_tag_12;
   int     _places_tag_13;
   int     _places_tag_14;
   int     _places_tag_15;
   int     _places_tag_16;
   int     _places_tag_17;
   int     _places_tag_18;
   int     _places_tag_19;
   int     _places_tag_20;
   int     _places_tag_21;
   int     _places_tag_22;
   int     _places_tag_23;
   int     _places_tag_24;
   int     _places_tag_25;
   int     _places_tag_26;
   int     _places_tag_27;
   int     _places_tag_28;
   int     _places_tag_29;
   int     _places_tag_30;
   int     _places_tag_31;
   int     _places_tag_32;
   int     _places_tag_33;
   int     _places_tag_34;
   int     _places_tag_35;
   int     _places_tag_36;
   int     _places_tag_37;
   int     _places_tag_38;
```

```
    int     _places_tag_39;
    int     _places_tag_40;
    int     _places_tag_41;
    int     _places_tag_42;
    int     _places_tag_43;
    int     _places_tag_44;
    int     _places_tag_45;
    int     _places_tag_46;
    int     _places_tag_47;
    int     _places_tag_48;
    int     _places_tag_49;
    int     _places_tag_50;
    int     _places_tag_51;
    int     _places_tag_52;
    int     _places_tag_53;
    int     _places_tag_54;
    int     _places_tag_55;
    int     _places_tag_56;
    int     _places_tag_57;
    int     _places_tag_58;
    int     _places_tag_59;
    int     _places_tag_60;
    int     _places_tag_61;
    int     _places_tag_62;
    int     _places_tag_63;
    int     _places_tag_64;
    int     _places_tag_65;
    int     _places_tag_66;
    int     _places_tag_67;
    int     _places_tag_68;
    int     _places_tag_69;
    int     _places_tag_70;
} KEYWORDS_record;

class KEYWORDS
{
 protected:

    KEYWORDS_record rec;
    KEYWORDS_record *recp;

    long current;
    int  dirty;
    int  is_open;
    int  iOrder;


    int  (KEYWORDS::*bof_fun)(void);
    int  (KEYWORDS::*eof_fun)(void);
    int  (KEYWORDS::*skip_fun)(int delta);
    void (KEYWORDS::*top_fun)(void);
    void (KEYWORDS::*bottom_fun)(void);
    int  (KEYWORDS::*search_fun)(void *k);
```

```
   TBASE  db;
   BTREEa in1;          //Index on field keyword


   int bof0(void) { return (current==1); }
   int bof1(void) { return in1.tBOF(); }

   int eof0(void) { return (current==db.numrec()); }
   int eof1(void) { return in1.tEOF(); }

   void top0(void)      { current=1; }
   void top1(void)      { in1.min_dat(&current); }

   void bottom0(void)   { current=db.numrec(); }
   void bottom1(void)   { in1.max_dat(&current); }

   int  search0(void * )      { return TRUE; }
   int  search1(void *k)      { return in1.search_dat_ge(k,&current); }

   int  skip0(int delta);
   int  skip1(int delta)      { return in1.skip_dat(delta,&current); }

 public:

 ////////////////////////////////// class constructor
//////////////////////////
   KEYWORDS(void);

 ////////////////////////////////// class destructor
//////////////////////////
   ~KEYWORDS(void)  { close(); }

 ////////////////////////////////// current record number
//////////////////////
   long curr_rec(void)   { return current; }

 ////////////////////////////////// define
//////////////////////////////////////
   void define(void);

 ////////////////////////////////// open & close
//////////////////////////////
   void open(void);
   void close(void);

 ////////////////////////////////// delete
//////////////////////////////////////
   int  is_delet(void)  { return db.is_delet(current); }
   void undelet(void)   { db.undelet(current); }
   void delet(void)     { db.delet(current); }

   int  is_delet(long n) { return db.is_delet(n); }
   void undelet(long n)  { db.undelet(n); }
   void delet(long n)    { db.delet(n); }
```

```
   ////////////////////////////////// number of records
  ////////////////////////////
     long numrec(void)          { return db.numrec(); }

   ////////////////////////////////// import/export
  ////////////////////////////////
     int  import(char *s);
     int  export(char *s);
     int  to_DBASE(char *s);

   ////////////////////////////////// read/write current record
  ////////////////////
     void write_rec2(void);
     void write_rec(void) { if(dirty) write_rec2(); }

     void read_rec(void)
     {
          recp=(KEYWORDS_record *)db.locate_rec(current);
          rec=*recp;
     }

   ////////////////////////////////// reindexing
  ////////////////////////////////////
     void reindex(void);

   ////////////////////////////////// append
  ////////////////////////////////////////
     void append(void);         //Indexes are NOT updated.
     void append_blank(void);   //Indexes ARE updated.

   ////////////////////////////////// data in header
  ////////////////////////////////
     int data_2_header(void *p,U16 size) { return db.data_2_header(p,size); }
     int header_2_data(void *p,U16 size) { return db.header_2_data(p,size); }
     U16 max_data_in_header(void)        { return db.max_data_in_header();  }

   ////////////////////////////////// pack
  ////////////////////////////////////////
     void pack(void);

   ////////////////////////////////// (change) active index
  ////////////////////////
     void order(int nr);
     int  order(void)  { return iOrder; }

   ////////////////////////////////// testing begin/end
  ////////////////////////////
     int  tBOF(void)        { return (this->*bof_fun)(); }
     int  tEOF(void)        { return (this->*eof_fun)(); }

   ////////////////////////////////// relocating
  ////////////////////////////////////
```

```
   int  skip(int del);
   void bottom(void)      { write_rec(); (this->*bottom_fun)(); read_rec(); }
   void top(void)         { write_rec(); (this->*top_fun)(); read_rec(); }
   void search(void *k)   { write_rec(); if(!(this->*search_fun)(k)) bottom();
read_rec(); }
   void go_to(long n);

 ////////////////////////reading fields
//////////////////////////////////////
    char * keyword(void)          { return rec._keyword; }
    int    num_of_items(void)     { return rec._num_of_items; }
    int    places_tag_1(void)     { return rec._places_tag_1; }
    int    places_tag_2(void)     { return rec._places_tag_2; }
    int    places_tag_3(void)     { return rec._places_tag_3; }
    int    places_tag_4(void)     { return rec._places_tag_4; }
    int    places_tag_5(void)     { return rec._places_tag_5; }
    int    places_tag_6(void)     { return rec._places_tag_6; }
    int    places_tag_7(void)     { return rec._places_tag_7; }
    int    places_tag_8(void)     { return rec._places_tag_8; }
    int    places_tag_9(void)     { return rec._places_tag_9; }
    int    places_tag_10(void)    { return rec._places_tag_10; }
    int    places_tag_11(void)    { return rec._places_tag_11; }
    int    places_tag_12(void)    { return rec._places_tag_12; }
    int    places_tag_13(void)    { return rec._places_tag_13; }
    int    places_tag_14(void)    { return rec._places_tag_14; }
    int    places_tag_15(void)    { return rec._places_tag_15; }
    int    places_tag_16(void)    { return rec._places_tag_16; }
    int    places_tag_17(void)    { return rec._places_tag_17; }
    int    places_tag_18(void)    { return rec._places_tag_18; }
    int    places_tag_19(void)    { return rec._places_tag_19; }
    int    places_tag_20(void)    { return rec._places_tag_20; }
    int    places_tag_21(void)    { return rec._places_tag_21; }
    int    places_tag_22(void)    { return rec._places_tag_22; }
    int    places_tag_23(void)    { return rec._places_tag_23; }
    int    places_tag_24(void)    { return rec._places_tag_24; }
    int    places_tag_25(void)    { return rec._places_tag_25; }
    int    places_tag_26(void)    { return rec._places_tag_26; }
    int    places_tag_27(void)    { return rec._places_tag_27; }
    int    places_tag_28(void)    { return rec._places_tag_28; }
    int    places_tag_29(void)    { return rec._places_tag_29; }
    int    places_tag_30(void)    { return rec._places_tag_30; }
    int    places_tag_31(void)    { return rec._places_tag_31; }
    int    places_tag_32(void)    { return rec._places_tag_32; }
    int    places_tag_33(void)    { return rec._places_tag_33; }
    int    places_tag_34(void)    { return rec._places_tag_34; }
    int    places_tag_35(void)    { return rec._places_tag_35; }
    int    places_tag_36(void)    { return rec._places_tag_36; }
    int    places_tag_37(void)    { return rec._places_tag_37; }
    int    places_tag_38(void)    { return rec._places_tag_38; }
    int    places_tag_39(void)    { return rec._places_tag_39; }
    int    places_tag_40(void)    { return rec._places_tag_40; }
    int    places_tag_41(void)    { return rec._places_tag_41; }
    int    places_tag_42(void)    { return rec._places_tag_42; }
    int    places_tag_43(void)    { return rec._places_tag_43; }
```

```
    int    places_tag_44(void)  { return rec._places_tag_44; }
    int    places_tag_45(void)  { return rec._places_tag_45; }
    int    places_tag_46(void)  { return rec._places_tag_46; }
    int    places_tag_47(void)  { return rec._places_tag_47; }
    int    places_tag_48(void)  { return rec._places_tag_48; }
    int    places_tag_49(void)  { return rec._places_tag_49; }
    int    places_tag_50(void)  { return rec._places_tag_50; }
    int    places_tag_51(void)  { return rec._places_tag_51; }
    int    places_tag_52(void)  { return rec._places_tag_52; }
    int    places_tag_53(void)  { return rec._places_tag_53; }
    int    places_tag_54(void)  { return rec._places_tag_54; }
    int    places_tag_55(void)  { return rec._places_tag_55; }
    int    places_tag_56(void)  { return rec._places_tag_56; }
    int    places_tag_57(void)  { return rec._places_tag_57; }
    int    places_tag_58(void)  { return rec._places_tag_58; }
    int    places_tag_59(void)  { return rec._places_tag_59; }
    int    places_tag_60(void)  { return rec._places_tag_60; }
    int    places_tag_61(void)  { return rec._places_tag_61; }
    int    places_tag_62(void)  { return rec._places_tag_62; }
    int    places_tag_63(void)  { return rec._places_tag_63; }
    int    places_tag_64(void)  { return rec._places_tag_64; }
    int    places_tag_65(void)  { return rec._places_tag_65; }
    int    places_tag_66(void)  { return rec._places_tag_66; }
    int    places_tag_67(void)  { return rec._places_tag_67; }
    int    places_tag_68(void)  { return rec._places_tag_68; }
    int    places_tag_69(void)  { return rec._places_tag_69; }
    int    places_tag_70(void)  { return rec._places_tag_70; }

 ////////////////////////writing fields
////////////////////////////////////.
 // Note: when writing strings there are NO checks on the
 //       length. A string which is longer then the definition
 //       of the field indicates, will OVERWRITE other data!!
 //
    void   keyword(char *s)     { strcpy(rec._keyword,s); dirty=TRUE; }
    void   num_of_items(int i)  { rec._num_of_items=i; dirty=TRUE; }
    void   places_tag_1(int i)  { rec._places_tag_1=i; dirty=TRUE; }
    void   places_tag_2(int i)  { rec._places_tag_2=i; dirty=TRUE; }
    void   places_tag_3(int i)  { rec._places_tag_3=i; dirty=TRUE; }
    void   places_tag_4(int i)  { rec._places_tag_4=i; dirty=TRUE; }
    void   places_tag_5(int i)  { rec._places_tag_5=i; dirty=TRUE; }
    void   places_tag_6(int i)  { rec._places_tag_6=i; dirty=TRUE; }
    void   places_tag_7(int i)  { rec._places_tag_7=i; dirty=TRUE; }
    void   places_tag_8(int i)  { rec._places_tag_8=i; dirty=TRUE; }
    void   places_tag_9(int i)  { rec._places_tag_9=i; dirty=TRUE; }
    void   places_tag_10(int i)  { rec._places_tag_10=i; dirty=TRUE; }
    void   places_tag_11(int i)  { rec._places_tag_11=i; dirty=TRUE; }
    void   places_tag_12(int i)  { rec._places_tag_12=i; dirty=TRUE; }
    void   places_tag_13(int i)  { rec._places_tag_13=i; dirty=TRUE; }
    void   places_tag_14(int i)  { rec._places_tag_14=i; dirty=TRUE; }
    void   places_tag_15(int i)  { rec._places_tag_15=i; dirty=TRUE; }
    void   places_tag_16(int i)  { rec._places_tag_16=i; dirty=TRUE; }
    void   places_tag_17(int i)  { rec._places_tag_17=i; dirty=TRUE; }
    void   places_tag_18(int i)  { rec._places_tag_18=i; dirty=TRUE; }
```

```
    void   places_tag_19(int i)  { rec._places_tag_19=i; dirty=TRUE; }
    void   places_tag_20(int i)  { rec._places_tag_20=i; dirty=TRUE; }
    void   places_tag_21(int i)  { rec._places_tag_21=i; dirty=TRUE; }
    void   places_tag_22(int i)  { rec._places_tag_22=i; dirty=TRUE; }
    void   places_tag_23(int i)  { rec._places_tag_23=i; dirty=TRUE; }
    void   places_tag_24(int i)  { rec._places_tag_24=i; dirty=TRUE; }
    void   places_tag_25(int i)  { rec._places_tag_25=i; dirty=TRUE; }
    void   places_tag_26(int i)  { rec._places_tag_26=i; dirty=TRUE; }
    void   places_tag_27(int i)  { rec._places_tag_27=i; dirty=TRUE; }
    void   places_tag_28(int i)  { rec._places_tag_28=i; dirty=TRUE; }
    void   places_tag_29(int i)  { rec._places_tag_29=i; dirty=TRUE; }
    void   places_tag_30(int i)  { rec._places_tag_30=i; dirty=TRUE; }
    void   places_tag_31(int i)  { rec._places_tag_31=i; dirty=TRUE; }
    void   places_tag_32(int i)  { rec._places_tag_32=i; dirty=TRUE; }
    void   places_tag_33(int i)  { rec._places_tag_33=i; dirty=TRUE; }
    void   places_tag_34(int i)  { rec._places_tag_34=i; dirty=TRUE; }
    void   places_tag_35(int i)  { rec._places_tag_35=i; dirty=TRUE; }
    void   places_tag_36(int i)  { rec._places_tag_36=i; dirty=TRUE; }
    void   places_tag_37(int i)  { rec._places_tag_37=i; dirty=TRUE; }
    void   places_tag_38(int i)  { rec._places_tag_38=i; dirty=TRUE; }
    void   places_tag_39(int i)  { rec._places_tag_39=i; dirty=TRUE; }
    void   places_tag_40(int i)  { rec._places_tag_40=i; dirty=TRUE; }
    void   places_tag_41(int i)  { rec._places_tag_41=i; dirty=TRUE; }
    void   places_tag_42(int i)  { rec._places_tag_42=i; dirty=TRUE; }
    void   places_tag_43(int i)  { rec._places_tag_43=i; dirty=TRUE; }
    void   places_tag_44(int i)  { rec._places_tag_44=i; dirty=TRUE; }
    void   places_tag_45(int i)  { rec._places_tag_45=i; dirty=TRUE; }
    void   places_tag_46(int i)  { rec._places_tag_46=i; dirty=TRUE; }
    void   places_tag_47(int i)  { rec._places_tag_47=i; dirty=TRUE; }
    void   places_tag_48(int i)  { rec._places_tag_48=i; dirty=TRUE; }
    void   places_tag_49(int i)  { rec._places_tag_49=i; dirty=TRUE; }
    void   places_tag_50(int i)  { rec._places_tag_50=i; dirty=TRUE; }
    void   places_tag_51(int i)  { rec._places_tag_51=i; dirty=TRUE; }
    void   places_tag_52(int i)  { rec._places_tag_52=i; dirty=TRUE; }
    void   places_tag_53(int i)  { rec._places_tag_53=i; dirty=TRUE; }
    void   places_tag_54(int i)  { rec._places_tag_54=i; dirty=TRUE; }
    void   places_tag_55(int i)  { rec._places_tag_55=i; dirty=TRUE; }
    void   places_tag_56(int i)  { rec._places_tag_56=i; dirty=TRUE; }
    void   places_tag_57(int i)  { rec._places_tag_57=i; dirty=TRUE; }
    void   places_tag_58(int i)  { rec._places_tag_58=i; dirty=TRUE; }
    void   places_tag_59(int i)  { rec._places_tag_59=i; dirty=TRUE; }
    void   places_tag_60(int i)  { rec._places_tag_60=i; dirty=TRUE; }
    void   places_tag_61(int i)  { rec._places_tag_61=i; dirty=TRUE; }
    void   places_tag_62(int i)  { rec._places_tag_62=i; dirty=TRUE; }
    void   places_tag_63(int i)  { rec._places_tag_63=i; dirty=TRUE; }
    void   places_tag_64(int i)  { rec._places_tag_64=i; dirty=TRUE; }
    void   places_tag_65(int i)  { rec._places_tag_65=i; dirty=TRUE; }
    void   places_tag_66(int i)  { rec._places_tag_66=i; dirty=TRUE; }
    void   places_tag_67(int i)  { rec._places_tag_67=i; dirty=TRUE; }
    void   places_tag_68(int i)  { rec._places_tag_68=i; dirty=TRUE; }
    void   places_tag_69(int i)  { rec._places_tag_69=i; dirty=TRUE; }
    void   places_tag_70(int i)  { rec._places_tag_70=i; dirty=TRUE; }

};
```

Appendix D BEGIN ACTIONS.CPP

```
#include "inftlapp.h"

Places places_db;                    //Makes the database classes
KEYWORDS keywords_db;

// add_data adds data to the keywords and places database
// It uses the database classes places_db and keywords_db
// It takes as input an input buffer (add_buffer) that is
// generated from the add_input dialog
void add_data(TAddBuffer  add_buffer)
{
  int next_place;                   //Holds the next_place number
  int result;                       //Used to hold the result of a strcpy
  char nothing[21] =
"\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0\x0";
  int temp;
  places_db.open();                  //Opens the databases
  keywords_db.open();
  keywords_db.order(KEYWORD_INDEX); //Sets search to use the keyword
                                     //As the key
  places_db.order(PLACE_NUMBER_INDEX); //Sets search to use the place_number
                                       //as the key
  places_db.bottom();                //Goes to bottom of db
  next_place = places_db.place_number() + 1; //Sets next_place
  places_db.append_blank();           //Adds a new record
  places_db.place_number(next_place); //sets place_number to next_place
  places_db.place(add_buffer.item);   //sets place to the item
  places_db.description(add_buffer.description); //adds the description
  //This checks to see if there was a keyword entered in the first
  //keyword control.
  if (strcmp(add_buffer.add_kw1,nothing)!=0)
  {
    keywords_db.search(add_buffer.add_kw1);  //Looks to see if we already
                                             //have this keyword in the db
    //We need to do this because the database class chooses the closest
    //match, which may not be an exact match.
    result = strcmp(keywords_db.keyword(),add_buffer.add_kw1);
    //If we do have the keyword, then add the place_number to the list
    //of place numbers associated with this keyword.  If not, then
    //we need to add a new keyword record.
    if (result == 0)
    {
      temp = keywords_db.num_of_items(); //Gets the number of items before
      temp ++;                           //Adds one
      keywords_db.num_of_items(temp);    //Stores the new number of items
      if (keywords_db.num_of_items() != 71) //Checks for too many items
      {
        update_places(next_place, temp);  //Adds the place to the keyword rec
      }
      else
```

```
    {
      // Let the user know that we couldn't add this item to the keyword
      // Record.  However, still tries to add the rest.
      MessageBox(0,"There are too many entries for this keyword. Please
         try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
      temp = keywords_db.num_of_items(); //Resets the number of items
      temp --;                           //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw1);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
  }
}
//This checks to see if there was a keyword entered in the second
//keyword control.
if (strcmp(add_buffer.add_kw2,nothing)!=0)
{
  keywords_db.search(add_buffer.add_kw2);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw2);
  //If we do have the keyword, then add the place_number to the list
  //of place numbers associated with this keyword.  If not, then
  //we need to add a new keyword record.
  if (result == 0)
  {
    temp = keywords_db.num_of_items(); //Gets the number of items before
    temp ++;                           //Adds one
    keywords_db.num_of_items(temp);    //Stores the new number of items
    if (keywords_db.num_of_items() != 71) //Checks for too many items
    {
      update_places(next_place, temp);  //Adds the place to the keyword rec
    }
    else
    {
      // Let the user know that we couldn't add this item to the keyword
      // Record.  However, still tries to add the rest.
      MessageBox(0,"There are too many entries for this keyword. Please
         try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
      temp = keywords_db.num_of_items(); //Resets the number of items
      temp --;                           //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
```

```
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw2);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
  }
}
//This checks to see if there was a keyword entered in the third
//keyword control.
if (strcmp(add_buffer.add_kw3,nothing)!=0)
{
  keywords_db.search(add_buffer.add_kw3);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw3);
  //If we do have the keyword, then add the place_number to the list
  //of place numbers associated with this keyword.  If not, then
  //we need to add a new keyword record.
  if (result == 0)
  {
    temp = keywords_db.num_of_items(); //Gets the number of items before
    temp ++;                           //Adds one
    keywords_db.num_of_items(temp);    //Stores the new number of items
    if (keywords_db.num_of_items() != 71) //Checks for too many items
    {
      update_places(next_place, temp);  //Adds the place to the keyword rec
    }
    else
    {
      // Let the user know that we couldn't add this item to the keyword
      // Record.  However, still tries to add the rest.
      MessageBox(0,"There are too many entries for this keyword. Please
         try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
      temp = keywords_db.num_of_items(); //Resets the number of items
      temp --;                           //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw3);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
  }
}
//This checks to see if there was a keyword entered in the fourth
//keyword control.
if (strcmp(add_buffer.add_kw4,nothing)!=0)
{
```

```
  keywords_db.search(add_buffer.add_kw4);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw4);
  //If we do have the keyword, then add the place_number to the list
  //of place numbers associated with this keyword.  If not, then
  //we need to add a new keyword record.
  if (result == 0)
  {
    temp = keywords_db.num_of_items(); //Gets the number of items before
    temp ++;                           //Adds one
    keywords_db.num_of_items(temp);    //Stores the new number of items
    if (keywords_db.num_of_items() != 71) //Checks for too many items
    {
      update_places(next_place, temp);  //Adds the place to the keyword rec
    }
    else
    {
      // Let the user know that we couldn't add this item to the keyword
      // Record.  However, still tries to add the rest.
      MessageBox(0,"There are too many entries for this keyword. Please
         try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
      temp = keywords_db.num_of_items(); //Resets the number of items
      temp --;                           //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw4);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
  }
}
//This checks to see if there was a keyword entered in the fifth
//keyword control.
if (strcmp(add_buffer.add_kw5,nothing)!=0)
{
  keywords_db.search(add_buffer.add_kw5);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw5);
  //If we do have the keyword, then add the place_number to the list
  //of place numbers associated with this keyword.  If not, then
  //we need to add a new keyword record.
  if (result == 0)
  {
    temp = keywords_db.num_of_items(); //Gets the number of items before
    temp ++;                           //Adds one
```

```
      keywords_db.num_of_items(temp);     //Stores the new number of items
      if (keywords_db.num_of_items() != 71) //Checks for too many items
      {
        update_places(next_place, temp);  //Adds the place to the keyword rec
      }
      else
      {
        // Let the user know that we couldn't add this item to the keyword
        // Record.  However, still tries to add the rest.
        MessageBox(0,"There are too many entries for this keyword. Please
           try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
        temp = keywords_db.num_of_items(); //Resets the number of items
        temp --;                           //To 70
        keywords_db.num_of_items(temp);
      }
    }
    else
    {
      keywords_db.append_blank(); //Add a new keyword record
      keywords_db.num_of_items(1); //Set the number of items to 1
      //Copy the keyword to the keyword database
      strcpy(keywords_db.keyword(),add_buffer.add_kw5);
      keywords_db.places_tag_1(next_place); //Put the item tag in the
                                            //keyword db
    }
  }
  //This checks to see if there was a keyword entered in the sixth
  //keyword control.
  if (strcmp(add_buffer.add_kw6,nothing)!=0)
  {
    keywords_db.search(add_buffer.add_kw6);  //Looks to see if we already
                                             //have this keyword in the db
    //We need to do this because the database class chooses the closest
    //match, which may not be an exact match.
    result = strcmp(keywords_db.keyword(),add_buffer.add_kw6);
    //If we do have the keyword, then add the place_number to the list
    //of place numbers associated with this keyword.  If not, then
    //we need to add a new keyword record.
    if (result == 0)
    {
      temp = keywords_db.num_of_items(); //Gets the number of items before
      temp ++;                           //Adds one
      keywords_db.num_of_items(temp);    //Stores the new number of items
      if (keywords_db.num_of_items() != 71) //Checks for too many items
      {
        update_places(next_place, temp);  //Adds the place to the keyword rec
      }
      else
      {
        // Let the user know that we couldn't add this item to the keyword
        // Record.  However, still tries to add the rest.
        MessageBox(0,"There are too many entries for this keyword. Please
           try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
        temp = keywords_db.num_of_items(); //Resets the number of items
```

```
      temp --;                                  //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw6);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
  }
}
//This checks to see if there was a keyword entered in the seventh
//keyword control.
if (strcmp(add_buffer.add_kw7,nothing)!=0)
{
  keywords_db.search(add_buffer.add_kw7);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw7);
  //If we do have the keyword, then add the place_number to the list
  //of place numbers associated with this keyword.  If not, then
  //we need to add a new keyword record.
  if (result == 0)
  {
    temp = keywords_db.num_of_items(); //Gets the number of items before
    temp ++;                           //Adds one
    keywords_db.num_of_items(temp);    //Stores the new number of items
    if (keywords_db.num_of_items() != 71) //Checks for too many items
    {
      update_places(next_place, temp);  //Adds the place to the keyword rec
    }
    else
    {
      // Let the user know that we couldn't add this item to the keyword
      // Record.  However, still tries to add the rest.
      MessageBox(0,"There are too many entries for this keyword. Please
         try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
      temp = keywords_db.num_of_items(); //Resets the number of items
      temp --;                                  //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw7);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
```

```
  }
}
//This checks to see if there was a keyword entered in the eighth
//keyword control.
if (strcmp(add_buffer.add_kw8,nothing)!=0)
{
  keywords_db.search(add_buffer.add_kw8);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw8);
  //If we do have the keyword, then add the place_number to the list
  //of place numbers associated with this keyword.  If not, then
  //we need to add a new keyword record.
  if (result == 0)
  {
    temp = keywords_db.num_of_items(); //Gets the number of items before
    temp ++;                           //Adds one
    keywords_db.num_of_items(temp);    //Stores the new number of items
    if (keywords_db.num_of_items() != 71) //Checks for too many items
    {
      update_places(next_place, temp);  //Adds the place to the keyword rec
    }
    else
    {
      // Let the user know that we couldn't add this item to the keyword
      // Record.  However, still tries to add the rest.
      MessageBox(0,"There are too many entries for this keyword. Please
         try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
      temp = keywords_db.num_of_items(); //Resets the number of items
      temp --;                           //To 70
      keywords_db.num_of_items(temp);
    }
  }
  else
  {
    keywords_db.append_blank(); //Add a new keyword record
    keywords_db.num_of_items(1); //Set the number of items to 1
    //Copy the keyword to the keyword database
    strcpy(keywords_db.keyword(),add_buffer.add_kw8);
    keywords_db.places_tag_1(next_place); //Put the item tag in the
                                          //keyword db
  }
}
//This checks to see if there was a keyword entered in the ninth
//keyword control.
if (strcmp(add_buffer.add_kw9,nothing)!=0)
{
  keywords_db.search(add_buffer.add_kw9);  //Looks to see if we already
                                           //have this keyword in the db
  //We need to do this because the database class chooses the closest
  //match, which may not be an exact match.
  result = strcmp(keywords_db.keyword(),add_buffer.add_kw9);
  //If we do have the keyword, then add the place_number to the list
```

```
    //of place numbers associated with this keyword.  If not, then
    //we need to add a new keyword record.
    if (result == 0)
    {
      temp = keywords_db.num_of_items(); //Gets the number of items before
      temp ++;                           //Adds one
      keywords_db.num_of_items(temp);    //Stores the new number of items
      if (keywords_db.num_of_items() != 71) //Checks for too many items
      {
        update_places(next_place, temp);  //Adds the place to the keyword rec
      }
      else
      {
        // Let the user know that we couldn't add this item to the keyword
        // Record.  However, still tries to add the rest.
        MessageBox(0,"There are too many entries for this keyword. Please
           try again", "Error: Too Many", MB_OK | MB_ICONSTOP);
        temp = keywords_db.num_of_items(); //Resets the number of items
        temp --;                           //To 70
        keywords_db.num_of_items(temp);
      }
    }
    else
    {
      keywords_db.append_blank(); //Add a new keyword record
      keywords_db.num_of_items(1); //Set the number of items to 1
      //Copy the keyword to the keyword database
      strcpy(keywords_db.keyword(),add_buffer.add_kw9);
      keywords_db.places_tag_1(next_place); //Put the item tag in the
                                            //keyword db
    }
  }
keywords_db.close();  //Closes the databases and flushes their buffers
places_db.close();
}

// This function deletes an item from the database
// It is not totally implimented yet, but will be in Version 2.0
// It uses the keyword_db and places_db.  It takes as input
// an item to be deleted (from delete_buffer).  delete_buffer is set
// by the delete_dialog control
void delete_data(TDeleteBuffer delete_buffer)
{
  keywords_db.open(); //Opens the database
  places_db.close();
places_db.order(PLACE_NUMBER_INDEX);  //Sets the search order to place_number
keywords_db.order(KEYWORD_INDEX);     //Sets the search order to keyword
// The delete code will go here
keywords_db.close();  // Closes the databases
places_db.close();

}

// This function takes the place_number (in next_place) and the next
```

```
// available place_tag (in temp) that is located in keywords_db.
// The databases have already been opened so we don't need to open them again
void update_places(int next_place, int temp)
{
  switch(temp) { //We do switch to determine which place_tag field to use
     case 2:
       keywords_db.places_tag_2(next_place); //Sets the tag to the place
                                             //number that we created
       break;  //Break out of switch construct
     case 3:
       keywords_db.places_tag_3(next_place);
       break;
     case 4:
       keywords_db.places_tag_4(next_place);
       break;
     case 5:
       keywords_db.places_tag_5(next_place);
       break;
     case 6:
       keywords_db.places_tag_6(next_place);
       break;
     case 7:
       keywords_db.places_tag_7(next_place);
       break;
     case 8:
       keywords_db.places_tag_8(next_place);
       break;
     case 9:
       keywords_db.places_tag_9(next_place);
       break;
     case 10:
       keywords_db.places_tag_10(next_place);
       break;
     case 11:
       keywords_db.places_tag_11(next_place);
       break;
     case 12:
       keywords_db.places_tag_12(next_place);
       break;
     case 13:
       keywords_db.places_tag_13(next_place);
       break;
     case 14:
       keywords_db.places_tag_14(next_place);
       break;
     case 15:
       keywords_db.places_tag_15(next_place);
       break;
     case 16:
       keywords_db.places_tag_16(next_place);
       break;
     case 17:
       keywords_db.places_tag_17(next_place);
       break;
```

```
case 18:
  keywords_db.places_tag_18(next_place);
  break;
case 19:
  keywords_db.places_tag_19(next_place);
  break;
case 20:
  keywords_db.places_tag_20(next_place);
  break;
case 21:
  keywords_db.places_tag_21(next_place);
  break;
case 22:
  keywords_db.places_tag_22(next_place);
  break;
case 23:
  keywords_db.places_tag_23(next_place);
  break;
case 24:
  keywords_db.places_tag_24(next_place);
  break;
case 25:
  keywords_db.places_tag_25(next_place);
  break;
case 26:
  keywords_db.places_tag_26(next_place);
  break;
case 27:
  keywords_db.places_tag_27(next_place);
  break;
case 28:
  keywords_db.places_tag_28(next_place);
  break;
case 29:
  keywords_db.places_tag_29(next_place);
  break;
case 30:
  keywords_db.places_tag_30(next_place);
  break;
case 31:
  keywords_db.places_tag_31(next_place);
  break;
case 32:
  keywords_db.places_tag_32(next_place);
  break;
case 33:
  keywords_db.places_tag_33(next_place);
  break;
case 34:
  keywords_db.places_tag_34(next_place);
  break;
case 35:
  keywords_db.places_tag_35(next_place);
  break;
```

```
case 36:
  keywords_db.places_tag_36(next_place);
  break;
case 37:
  keywords_db.places_tag_37(next_place);
  break;
case 38:
  keywords_db.places_tag_38(next_place);
  break;
case 39:
  keywords_db.places_tag_39(next_place);
  break;
case 40:
  keywords_db.places_tag_40(next_place);
  break;
case 41:
  keywords_db.places_tag_41(next_place);
  break;
case 42:
  keywords_db.places_tag_42(next_place);
  break;
case 43:
  keywords_db.places_tag_43(next_place);
  break;
case 44:
  keywords_db.places_tag_44(next_place);
  break;
case 45:
  keywords_db.places_tag_45(next_place);
  break;
case 46:
  keywords_db.places_tag_46(next_place);
  break;
case 47:
  keywords_db.places_tag_47(next_place);
  break;
case 48:
  keywords_db.places_tag_48(next_place);
  break;
case 49:
  keywords_db.places_tag_49(next_place);
  break;
case 50:
  keywords_db.places_tag_50(next_place);
  break;
case 51:
  keywords_db.places_tag_51(next_place);
  break;
case 52:
  keywords_db.places_tag_52(next_place);
  break;
case 53:
  keywords_db.places_tag_53(next_place);
  break;
```

```
case 54:
  keywords_db.places_tag_54(next_place);
  break;
case 55:
  keywords_db.places_tag_55(next_place);
  break;
case 56:
  keywords_db.places_tag_56(next_place);
  break;
case 57:
  keywords_db.places_tag_57(next_place);
  break;
case 58:
  keywords_db.places_tag_58(next_place);
  break;
case 59:
  keywords_db.places_tag_59(next_place);
  break;
case 60:
  keywords_db.places_tag_60(next_place);
  break;
case 61:
  keywords_db.places_tag_61(next_place);
  break;
case 62:
  keywords_db.places_tag_62(next_place);
  break;
case 63:
  keywords_db.places_tag_63(next_place);
  break;
case 64:
  keywords_db.places_tag_64(next_place);
  break;
case 65:
  keywords_db.places_tag_65(next_place);
  break;
case 66:
  keywords_db.places_tag_66(next_place);
  break;
case 67:
  keywords_db.places_tag_67(next_place);
  break;
case 68:
  keywords_db.places_tag_68(next_place);
  break;
case 69:
  keywords_db.places_tag_69(next_place);
  break;
case 70:
  keywords_db.places_tag_70(next_place);
  break;
 default:
   break;
}
```

```
}
// Sets up a Message for when the user double clicks on the item
DEFINE_RESPONSE_TABLE1(TSearchResponse, TDialog)
  EV_LBN_DBLCLK(IDC_LISTBOX, SelectAnItem),
END_RESPONSE_TABLE;

// The class destrcutor
TSearchResponse::~TSearchResponse()
{
}
// The class constructor
// First Parameter: The Window that we are working with
// Second Parameter: The search_buffer that holds the item to be search for
// The Dialog created is the one defined in intlapp.rc item ID_LISTBOX
TSearchResponse::TSearchResponse(TDecoratedMDIFrame * frame, TSearchBuffer
  search_buffer) : TDialog(frame, ID_LISTBOX)
{
  int number;           // The number of items in the database
                        // that are slugged to the specified keyword
  int t1;               //The integer to be searched for
  keywords_db.open();   //Opens the databases
  places_db.open();
  places_db.order(PLACE_NUMBER_INDEX);  //Sets the index order to place_number
  keywords_db.order(KEYWORD_INDEX);     //Sets the index order to keyword
  keywords_db.search(search_buffer.search_string); //Does the search
  number = keywords_db.num_of_items();  //Gets the number of items associated
                                        //With that keyword
  listbox1 = new TListBox(this, IDC_LISTBOX); //Constructs a listbox control
                                              //with the parameters coming
                                              //from IDC_LISTBOX in
                                              //inftlapp.rc
  listboxdata = new TListBoxData();           //Creates a transfer buffer
  listbox1->EnableTransfer();                 //Tells listbox1 to
                                              //Look for transfers
  switch (number)                             //Switches on number of
  {                                           //items to be displayed
  //NOTE: this is a fall-through switch construct as we don't use
  //      any breaks.
  case 70:
    t1 = keywords_db.places_tag_70();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case 69:
    t1 = keywords_db.places_tag_69();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case 68:
    t1 = keywords_db.places_tag_68();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case 67:
    t1 = keywords_db.places_tag_67();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
```

```
case 66:
  t1 = keywords_db.places_tag_66();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 65:
  t1 = keywords_db.places_tag_65();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 64:
  t1 = keywords_db.places_tag_64();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 63:
  t1 = keywords_db.places_tag_63();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 62:
  t1 = keywords_db.places_tag_62();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 61:
  t1 = keywords_db.places_tag_61();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 60:
  t1 = keywords_db.places_tag_60();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 59:
  t1 = keywords_db.places_tag_59();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 58:
  t1 = keywords_db.places_tag_58();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 57:
  t1 = keywords_db.places_tag_57();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 56:
  t1 = keywords_db.places_tag_56();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 55:
  t1 = keywords_db.places_tag_55();
     places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 54:
  t1 = keywords_db.places_tag_54();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 53:
  t1 = keywords_db.places_tag_53();
```

```
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 52:
  t1 = keywords_db.places_tag_52();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 51:
  t1 = keywords_db.places_tag_51();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 50:
  t1 = keywords_db.places_tag_50();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 49:
  t1 = keywords_db.places_tag_49();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 48:
  t1 = keywords_db.places_tag_48();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 47:
  t1 = keywords_db.places_tag_47();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 46:
  t1 = keywords_db.places_tag_46();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 45:
  t1 = keywords_db.places_tag_45();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 44:
  t1 = keywords_db.places_tag_44();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 43:
  t1 = keywords_db.places_tag_43();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 42:
  t1 = keywords_db.places_tag_42();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 41:
  t1 = keywords_db.places_tag_41();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 40:
  t1 = keywords_db.places_tag_40();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
```

```
case 39:
  t1 = keywords_db.places_tag_39();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 38:
  t1 = keywords_db.places_tag_38();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 37:
  t1 = keywords_db.places_tag_37();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 36:
  t1 = keywords_db.places_tag_36();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 35:
  t1 = keywords_db.places_tag_35();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 34:
  t1 = keywords_db.places_tag_34();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 33:
  t1 = keywords_db.places_tag_33();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 32:
  t1 = keywords_db.places_tag_32();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 31:
  t1 = keywords_db.places_tag_31();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 30:
  t1 = keywords_db.places_tag_30();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 29:
  t1 = keywords_db.places_tag_29();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 28:
  t1 = keywords_db.places_tag_28();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 27:
  t1 = keywords_db.places_tag_27();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 26:
  t1 = keywords_db.places_tag_26();
```

```
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 25:
  t1 = keywords_db.places_tag_25();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 24:
  t1 = keywords_db.places_tag_24();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 23:
  t1 = keywords_db.places_tag_23();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 22:
  t1 = keywords_db.places_tag_22();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 21:
  t1 = keywords_db.places_tag_21();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case  20:
  t1 = keywords_db.places_tag_20();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 19:
  t1 = keywords_db.places_tag_19();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 18:
  t1 = keywords_db.places_tag_18();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 17:
  t1 = keywords_db.places_tag_17();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 16:
  t1 = keywords_db.places_tag_16();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 15:
  t1 = keywords_db.places_tag_15();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 14:
  t1 = keywords_db.places_tag_14();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
case 13:
  t1 = keywords_db.places_tag_13();
  places_db.search(&t1);
  listboxdata->AddString(places_db.place());
```

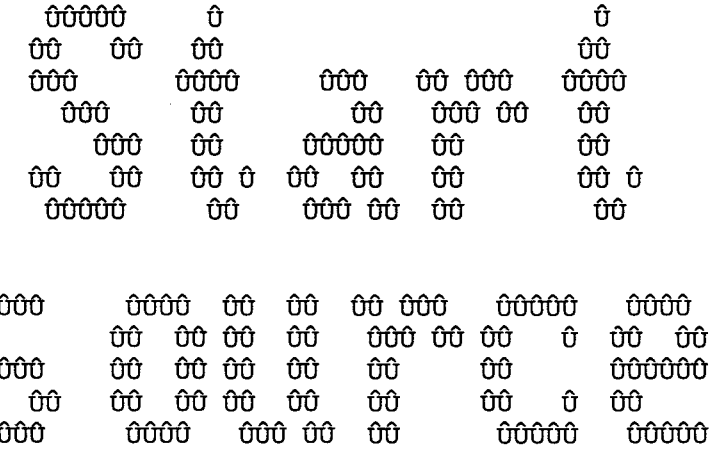
```
  case 12:
    t1 = keywords_db.places_tag_12();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case 11:
    t1 = keywords_db.places_tag_11();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  10:
    t1 = keywords_db.places_tag_10();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  9:
    t1 = keywords_db.places_tag_9();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  8:
    t1 = keywords_db.places_tag_8();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  7:
    t1 = keywords_db.places_tag_7();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  6:
    t1 = keywords_db.places_tag_6();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  5:
    t1 = keywords_db.places_tag_5();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  4:
    t1 = keywords_db.places_tag_4();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  3:
    t1 = keywords_db.places_tag_3();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case  2:
    t1 = keywords_db.places_tag_2();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  case 1:
    t1 = keywords_db.places_tag_1();
    places_db.search(&t1);
    listboxdata->AddString(places_db.place());
  default:
    break;
  }
  SetTransferBuffer(listboxdata);
}
```

```
// This function responds to the user's double click
// It searches the database to find the description associated
// with the place that was double clicked.  It then pops up
// a messagebox with the description.
void
TSearchResponse::SelectAnItem()
{
  int i = listbox1->GetSelIndex();     //gets the selected index
  places_db.order(PLACE_INDEX);        //Search order setup
  keywords_db.order(KEYWORD_INDEX);
  if (i >= 0)
  {
        char selection[21];
          //Gets the Selected string
          listbox1->GetSelString(selection,sizeof(selection));
          //Finds the description associated with the string
        places_db.search(selection);
          //Does a messagebox of the item's description
        MessageBox(places_db.description(),"Item's Description", MB_OK);
  }
}
```

```
BEGIN ACTIONS.H

#include "inftlapp.h"
#include <owl\listbox.h>
#include <owl\mdi.h>
#include <owl\mdichild.h>
#include <owl\decmdifr.h>

void add_data(TAddBuffer add_buffer);

void delete_data(TDeleteBuffer delete_buffer);

void update_places(int next_place, int temp);

class TSearchResponse: public TDialog {
public:
  TSearchResponse(TDecoratedMDIFrame * frame, TSearchBuffer search_buffer);
  ~TSearchResponse();
  TListBox* listbox1;           //Defines a listbox construct
  TListBoxData* listboxdata;    //Defines a listboxdata construct
protected:
  void SelectAnItem();
DECLARE_RESPONSE_TABLE(TSearchResponse);
};
```

Appendix E BEGIN PLACES.CPP

```
////////////////////////////////////////////////////////
//      Implementation file of database class: Places.
//
//      Source generated by: CSDBGEN version 1.2.b.
//      Date of generation:  Monday, 1 May 1995.
//      Time of generation:  21:31:59.
//
//
//      The next lines represent the database definition
//      file used as input for CSDBGEN.
//
//
/////////////// Start of the .def file /////////////////
/*
class: Places
record: Places_Record
file: it_places
field: place_number i Y
field: place s 100 Y
field: description s 500
*/
/////////////// End of .def file
////////////////////////////////////////////
/*
```



```
*/
////////////////////////////////////////////////////////////////////////////

#include "places.h"

//extern unsigned _stklen=7000;  //A large stack is needed
```

```
//////////////////////////////// Constructor ////////////////////////////////
Places::Places(void)
{
    is_open=FALSE;
    current=1;
}

//////////////////////////////// reindex ////////////////////////////////////
void Places::reindex(void)
{
    U32 l=current;
    Places_Record *rp;
    in1.empty();
    in2.empty();
    for(current=numrec(); current>0; current--)
    {
       rp=(Places_Record *)db.locate_rec(current);
       in1.insert(&rp->_place_number,&current);
       in2.insert(rp->_place,&current);
    }
    current=l;
}

/////////////// go_to ////////////////////////////////////////////
void Places::go_to(long n)
{
    if(order()!=UNSORTED)
    {
      csmess_p(7110,"Places");
      return;
    }
    write_rec();
    current=max(min(n,db.numrec()),1);
    read_rec();
}

/////////////// append blank/////////////////////////////////////////
void Places::append_blank(void)
{
       append();
       in1.insert(&rec._place_number,&current);
       in2.insert(rec._place,&current);
}

//////////////////////////////// append ////////////////////////////////////
// This function doesn't update the indexes, which can save some
// disk I/O because you are likely to alter the fields
// immediately after you have appended the record.
// However, if you have an index on a field you don't update, this
// record will NOT appear in that particular index!
// The 'append_blank' function does update all indexes, which
// makes it a safer, but slower option.
```

```
//
void Places::append(void)
{
    write_rec();
    memset(&rec,0,sizeof(Places_Record));
    current=db.append_rec(&rec);
    recp=(Places_Record *)db.locate_rec(current);
    dirty=TRUE;
}

////////////////////////////////// open ////////////////////////////////////////
void Places::open(void)
{
     if(is_open) return;
     int needs_reindex=FALSE;
     dirty=FALSE;

#ifdef _Windows
     int fre=300/5; //Use 300 Kb for buffers. You may increase this.
#else
     int fre=(int)(coreleft()-100000L)/5/1024;
     fre=max(fre,0);
#endif

     if(!db.open("it_place.dbf",fre))
     {
         csmess_disp("FATAL: Can't open database it_place.dbf.");
         exit(1);
     }
     if(db.lengthrec()!=sizeof(Places_Record))
     {
         csmess_disp("FATAL: wrong record size.\n\rProbably wrong or old
database file.");
         db.close();
         exit(1);
     }
     if(!file_exist("it_pla01.idx"))
     {
       in1.multiple_keys(TRUE);
       in1.define("it_pla01.idx",sizeof(int),sizeof(long));
       needs_reindex=TRUE;
     }
     in1.open("it_pla01.idx",fre*2);

     if(!file_exist("it_pla02.idx"))
     {
       in2.multiple_keys(TRUE);
       in2.define("it_pla02.idx",PLACE_LENGTH+1,sizeof(long));
       needs_reindex=TRUE;
     }
     in2.open("it_pla02.idx",fre*2);

     if(needs_reindex) reindex();
     is_open=TRUE;
```

```
     if(numrec()==0) append_blank();
     else            read_rec();

     order(UNSORTED);
}

////////////////////////////////// close ////////////////////////////////////
void Places::close(void)
{
     if(!is_open) return;
     write_rec();
     db.close();
     in1.close();
     in2.close();
     is_open=FALSE;
}

////////////////////////////////// define ///////////////////////////////////
void Places::define(void)
{
     db.define("it_place.dbf",sizeof(Places_Record));
     in1.multiple_keys(TRUE);
     in1.define("it_pla01.idx",sizeof(int),sizeof(long));
     in2.multiple_keys(TRUE);
     in2.define("it_pla02.idx",PLACE_LENGTH+1,sizeof(long));
}

////////////////////////////////// pack /////////////////////////////////////
void Places::pack(void)
{
     write_rec();
     db.pack();
     reindex();
     if(numrec()==0) append_blank();
     top();
}

////////////////////////////////// skip /////////////////////////////////////
int  Places::skip0(int delta)
{
     long old_current=current;
     current=max(min(current+delta,db.numrec()),1);
     return current-old_current;
}

int  Places::skip(int delta)
{
     int rc;
     write_rec();
     rc=(this->*skip_fun)(delta);
     read_rec();
     return rc;
}
```

```
////////////////////////////////// order ////////////////////////////////////
void Places::order(int nr)
{
     switch(nr)
     {
         case 0:        //Unsorted
                  bof_fun   =&Places::bof0;
                  eof_fun   =&Places::eof0;
                  skip_fun  =&Places::skip0;
                  top_fun   =&Places::top0;
                  bottom_fun=&Places::bottom0;
                  search_fun=&Places::search0;
                  break;
         case 1:        //Index on field place_number
                  bof_fun   =&Places::bof1;
                  eof_fun   =&Places::eof1;
                  skip_fun  =&Places::skip1;
                  top_fun   =&Places::top1;
                  bottom_fun=&Places::bottom1;
                  search_fun=&Places::search1;
                  break;
         case 2:        //Index on field place
                  bof_fun   =&Places::bof2;
                  eof_fun   =&Places::eof2;
                  skip_fun  =&Places::skip2;
                  top_fun   =&Places::top2;
                  bottom_fun=&Places::bottom2;
                  search_fun=&Places::search2;
                  break;
         default: return; // Function called with wrong parameter.
     }
     iOrder=nr;
     top();
}
////////////////////////////writing record  ////////////////////////////////
void Places::write_rec2(void)
{
    if(recp->_place_number!=rec._place_number)
    {
      in1.delet(&recp->_place_number,&current);
      in1.insert(&rec._place_number,&current);
    }
    if(strcmp(recp->_place,rec._place))
    {
      in2.delet(recp->_place,&current);
      in2.insert(rec._place,&current);
    }
    db.write_rec(current,&rec);
    dirty=FALSE;
}

////////////////////////////////// export ////////////////////////////////////
int  Places::export(char *s)
{
```

```
    FILE *fo=fopen(s,"w");
    if(fo==NULL) return FALSE;

    write_rec();
    fprintf(fo,"class:  Places");
    fprintf(fo,"\nrecord: Places_Record");
    fprintf(fo,"\nfile:   it_places");
    fprintf(fo,"\nfield:  place_number i Y");
    fprintf(fo,"\nfield:  place s 100 Y");
    fprintf(fo,"\nfield:  description s 500  ");

    if(ferror(fo)) { fclose(fo); return FALSE; }

    Places_Record *recp;
    for(long l=numrec();l>0;l--)
    {
       recp=( Places_Record * )db.locate_rec(l);
       fprintf(fo,"\n%c",12);
       fprintf(fo,"\n%d",recp->_place_number);
       fprintf(fo,"\n%s",recp->_place);
       fprintf(fo,"\n%s",recp->_description);
       fprintf(fo,"\n"); //Additional linefeed, to avoid trouble!

       if(ferror(fo)) { fclose(fo); return FALSE; }
    }
    return !fclose(fo);
}

//////////////////////////////// import ////////////////////////////////////
int Places::import(char *s)
{

     FILE *fr=fopen(s,"r");
     if(fr==NULL) return FALSE;


#define MAX_NUM_FIELDS  100   //Increase this to allow more fields
#define MAX_FIELD_LEN   500   //Increase this to allow longer fields

     int *finu;
     finu=(int *)malloc(MAX_NUM_FIELDS*sizeof(int));
     if(finu==NULL) { fclose(fr); return FALSE; }

     char *fibu;
     fibu=(char *)malloc(MAX_FIELD_LEN);
     if(fibu==NULL) { fclose(fr); free(finu); return FALSE; }
     *fibu=0;

     char *fipo=fibu+strlen("field:");
     char *cp;
     int  ifieldnr=0;
     int  ofieldnr;

     memset(finu,0,MAX_NUM_FIELDS*sizeof(int));
```

```
fgets(fibu,MAX_FIELD_LEN,fr);
while(!strchr(fibu,12))
{
  strlwr(fibu);
  notabs(fibu);
  trim_string(fibu);
  if(strstr(fibu,"field:"))
  {
     ifieldnr++;
     trim_string(fipo);
     if((cp=strchr(fipo,' '))!=NULL) *cp=0;
     if     (!strcmp(fipo,"place_number"))      ofieldnr=1;
     else if(!strcmp(fipo,"place")) ofieldnr=2;
     else if(!strcmp(fipo,"description")) ofieldnr=3;
     else ofieldnr=0;
     finu[ifieldnr]=ofieldnr;
  }
  fgets(fibu,MAX_FIELD_LEN,fr);
}



for(;;)
{
  if(!strchr(fibu,12))
  {
     ifieldnr++;
     if(ifieldnr<MAX_NUM_FIELDS)
       switch(finu[ifieldnr])
       {
          case 0:  break;
          case 1:
                   place_number(atoi(fibu));
                   break;
          case 2:
                   fibu[PLACE_LENGTH]=0;
                   place(fibu);
                   break;
          case 3:
                   fibu[DESCRIPTION_LENGTH]=0;
                   description(fibu);
                   break;
       }
  }
  else
  {
     ifieldnr=0;
     append_blank();
  }
  if(feof(fr)) break;
  fgets(fibu,MAX_FIELD_LEN,fr);
  cp=fibu+(max(1,strlen(fibu))-1);
  if(*cp=='\n') *cp=0;  //removing the line feed
```

```
        }


        fclose(fr);
        free(fibu);
        free(finu);

#undef MAX_NUM_FIELDS
#undef MAX_FIELD_LEN

 return TRUE;
}

////////////////////////////////// export to dBASE compatible file. ///////////
int Places::to_DBASE(char *s)
{

        char bufje[12];
        if(!is_open) return FALSE;

        write_rec();
        FILE *fo=fopen(s,"wb");
        if(fo==NULL) return FALSE;

        int i;

        DATE d_upda;
        d_upda.sem_jul(db.sj_updated());
        fputc(03,fo);

        fputc(d_upda.year()%100,fo);
        fputc(d_upda.month(),fo);
        fputc(d_upda.day(),fo);

        long nr_record=numrec();
        fwrite(&nr_record,sizeof(long),1,fo);
        putw(130,fo);           //Header length
        putw(606,fo);           //Length of data record
        for(i=0;i<20;i++) fputc(0,fo); // 20 dummy bytes


// Writing definition of field place_number to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACE_NUMB");
        fwrite(bufje,11,1,fo);
        fputc('N',fo);
        for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
        fputc(5,fo);
        fputc(0,fo);
        for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field place to dbase file header.
        memset(bufje,0,11);
        strcpy(bufje,"PLACE");
```

```
      fwrite(bufje,11,1,fo);
      fputc('C',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(100,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes

// Writing definition of field description to dbase file header.
      memset(bufje,0,11);
      strcpy(bufje,"DESCRIPTIO");
      fwrite(bufje,11,1,fo);
      fputc('C',fo);
      for(i=0;i<4;i++) fputc(0,fo); // 4 dummy bytes
      fputc(500,fo);
      fputc(0,fo);
      for(i=0;i<14;i++) fputc(0,fo); // 14 dummy bytes


      fputc(13,fo);  //Field terminator
      fputc(0,fo);

// By now we have written the definition of the
// record structure to the file header.
// From here on we will export the records.

     Places_Record *recp;
     for(long l=numrec();l>0;l--)
     {

        if(ferror(fo)) { fclose(fo); return FALSE; }
        recp=(Places_Record *)db.locate_rec(l);

        if(db.is_delet(l)) fputc(42,fo);
        else               fputc(32,fo);

////////////////////// writing field place_number /////////////
        fprintf(fo,"%5d",recp->_place_number);
////////////////////// writing field place /////////////
        fprintf(fo,"%-100s",recp->_place);
////////////////////// writing field description /////////////
        fprintf(fo,"%-500s",recp->_description);
     }
     fputc(26,fo);  //End of File
     fclose(fo);

 return TRUE;
}
```

```
BEGIN PLACES.H
#include "ctype.h"
#include "csa.h"
#include "csdb.h"

#define PLACE_LENGTH     100
#define DESCRIPTION_LENGTH     500

////////// Indexes to be used with the 'order()' function.////////////
#define UNSORTED   0
#define PLACE_NUMBER_INDEX     1
#define PLACE_INDEX      2

typedef struct
{
   int     _place_number;
   char    _place[PLACE_LENGTH+1];
   char    _description[DESCRIPTION_LENGTH+1];
} Places_Record;

class Places
{
 protected:

   Places_Record rec;
   Places_Record *recp;

   long current;
   int  dirty;
   int  is_open;
   int  iOrder;


   int  (Places::*bof_fun)(void);
   int  (Places::*eof_fun)(void);
   int  (Places::*skip_fun)(int delta);
   void (Places::*top_fun)(void);
   void (Places::*bottom_fun)(void);
   int  (Places::*search_fun)(void *k);

   TBASE  db;
   BTREEi in1;          //Index on field place_number
   BTREEa in2;          //Index on field place


   int bof0(void) { return (current==1); }
   int bof1(void) { return in1.tBOF(); }
   int bof2(void) { return in2.tBOF(); }

   int eof0(void) { return (current==db.numrec()); }
   int eof1(void) { return in1.tEOF(); }
   int eof2(void) { return in2.tEOF(); }
```

```
   void top0(void)       { current=1; }
   void top1(void)       { in1.min_dat(&current); }
   void top2(void)       { in2.min_dat(&current); }

   void bottom0(void)    { current=db.numrec(); }
   void bottom1(void)    { in1.max_dat(&current); }
   void bottom2(void)    { in2.max_dat(&current); }

   int  search0(void * )       { return TRUE; }
   int  search1(void *k)       { return in1.search_dat_ge(k,&current); }
   int  search2(void *k)       { return in2.search_dat_ge(k,&current); }

   int  skip0(int delta);
   int  skip1(int delta)       { return in1.skip_dat(delta,&current); }
   int  skip2(int delta)       { return in2.skip_dat(delta,&current); }

public:

 ////////////////////////////////// class constructor
//////////////////////////
   Places(void);

 ////////////////////////////////// class destructor
//////////////////////////
   ~Places(void)  { close(); }

 ////////////////////////////////// current record number
//////////////////////
   long curr_rec(void)   { return current; }

 ////////////////////////////////// define
//////////////////////////////////////
   void define(void);

 ////////////////////////////////// open & close
//////////////////////////////
   void open(void);
   void close(void);

 ////////////////////////////////// delete
//////////////////////////////////
   int  is_delet(void)  { return db.is_delet(current); }
   void undelet(void)   { db.undelet(current); }
   void delet(void)     { db.delet(current); }

   int  is_delet(long n) { return db.is_delet(n); }
   void undelet(long n)  { db.undelet(n); }
   void delet(long n)    { db.delet(n); }

 ////////////////////////////////// number of records
//////////////////////////
   long numrec(void)          { return db.numrec(); }
```

```
 ////////////////////////////////// import/export
//////////////////////////////
   int  import(char *s);
   int  export(char *s);
   int  to_DBASE(char *s);

 ////////////////////////////////// read/write current record
//////////////////

   void write_rec2(void);
   void write_rec(void) { if(dirty) write_rec2(); }

   void read_rec(void)
   {
        recp=(Places_Record *)db.locate_rec(current);
        rec=*recp;
   }

 ////////////////////////////////// reindexing
//////////////////////////////////
   void reindex(void);

 ////////////////////////////////// append
////////////////////////////////////
   void append(void);        //Indexes are NOT updated.
   void append_blank(void);  //Indexes ARE updated.

 ////////////////////////////////// data in header
//////////////////////////////
   int data_2_header(void *p,U16 size)' { return db.data_2_header(p,size); }
   int header_2_data(void *p,U16 size) { return db.header_2_data(p,size); }
   U16 max_data_in_header(void)        { return db.max_data_in_header();  }

 ////////////////////////////////// pack
////////////////////////////////////
   void pack(void);

 ////////////////////////////////// (change) active index
//////////////////////
   void order(int nr);
   int  order(void)  { return iOrder; }

 ////////////////////////////////// testing begin/end
//////////////////////////
   int  tBOF(void)       { return (this->*bof_fun)(); }
   int  tEOF(void)       { return (this->*eof_fun)(); }

 ////////////////////////////////// relocating
//////////////////////////////////
   int  skip(int del);
   void bottom(void)     { write_rec(); (this->*bottom_fun)(); read_rec(); }
   void top(void)        { write_rec(); (this->*top_fun)(); read_rec(); }
   void search(void *k)  { write_rec(); if(!(this->*search_fun)(k)) bottom();
read_rec(); }
```

```
   void go_to(long n);

 ////////////////////////reading fields
//////////////////////////////////////
    int    place_number(void)    { return rec._place_number; }
    char * place(void)           { return rec._place; }
    char * description(void)     { return rec._description; }

 ////////////////////////writing fields
//////////////////////////////////////
 // Note: when writing strings there are NO checks on the
 //       length. A string which is longer then the definition
 //       of the field indicates, will OVERWRITE other data!!
 //
    void   place_number(int i)  { rec._place_number=i; dirty=TRUE; }
    void   place(char *s)       { strcpy(rec._place,s); dirty=TRUE; }
    void   description(char *s) { strcpy(rec._description,s); dirty=TRUE; }

};
```

Appendix F BEGIN DIALOG.CPP

```
#include "inftlapp.h"

// Sets up the messages that the Search Dialog will respond to
DEFINE_RESPONSE_TABLE1(TSearchDialog, TDialog)
 EV_COMMAND(IDHELP, CmHelp),
END_RESPONSE_TABLE;

// Constructor
// Parameter 1: The frame that we are using
// Parameter 2: the search_buffer that will hold the control's ouput
// Paramter  3: A TDialog class that uses the dialog SEARCH_DIALOG
//              defined in inftlapp.rc
TSearchDialog::TSearchDialog(TDecoratedMDIFrame * frame, TSearchBuffer *
  search_buffer) : TDialog(frame, SEARCH_DIALOG)
{
 new TEdit(this, IDSEARCH_ENTRY, 21);  //Creates a new edit control
 SetTransferBuffer(search_buffer);     //Sets up a transfer to search_buffer
}

//Destructor
//Destroys the dialog box
TSearchDialog::~TSearchDialog()
{
}

// This creates the help screen if the user clicks on the help button
// when the search dialog is active.
void TSearchDialog::CmHelp()
{
  string msg;
  string nl('\n');

  msg += "This function will search the database and find" + nl;
  msg += "the closest match to the give keyword.  It will" + nl;
  msg += "then bring up a listing of the data that it finds";
  // Creates a pop-up message box
  MessageBox(msg.c_str(), "Searching Help", MB_OK | MB_ICONINFORMATION);
}

// Sets up the messages that the add dialog will respond to.
DEFINE_RESPONSE_TABLE1(TAddDialog, TDialog)
 EV_COMMAND(IDHELP, CmHelp),
END_RESPONSE_TABLE;

// Constructor
// Parameter 1: the frame that we are using
// Parameter 2: the add buffer that will hold the control's output
// Parameter 3: a TDialog class that uses the dialog ADD_DIALOG
//              defined in inftlapp.rc
TAddDialog::TAddDialog(TDecoratedMDIFrame * frame, TAddBuffer *
```

```
  add_buffer) : TDialog(frame, ADD_DIALOG)
{
 //Creates a series of edit controls.  The input is limited to 21 chars.
 new TEdit(this, ADD_KW1,21);
 new TEdit(this, ADD_KW2,21);
 new TEdit(this, ADD_KW3,21);
 new TEdit(this, ADD_KW4,21);
 new TEdit(this, ADD_KW5,21);
 new TEdit(this, ADD_KW6,21);
 new TEdit(this, ADD_KW7,21);
 new TEdit(this, ADD_KW8,21);
 new TEdit(this, ADD_KW9,21);
 //Creates a 101 char edit control.
 new TEdit(this, EDIT_ITEM,101);
 //Creates a 501 char edit control.
 new TEdit(this, ADD_DESCRIPTION,501);
 // Sets up the transfer buffer so that we can get data from the controls
 SetTransferBuffer(add_buffer);
}

//Destructor
//This function destroys the window
TAddDialog::~TAddDialog()
{
}

//This function responds to the help button in the ADD_DIALOG dialog box.
void TAddDialog::CmHelp()
{
  string msg;
  string nl('\n');

  msg += "This function will add the location and the" + nl;
  msg += "keywords assocated with it to the database";
  MessageBox(msg.c_str(), "Adding Help", MB_OK | MB_ICONINFORMATION);
}

//Sets up the DeleteDialog message reponses
DEFINE_RESPONSE_TABLE1(TDeleteDialog, TDialog)
 EV_COMMAND(IDHELP, CmHelp),
END_RESPONSE_TABLE;

// Constructor
// Creates the delete dialog
// Parameter 1: the frame that we are using
// Parameter 2: the delete buffer that holds the output of the control
// Parameter 3: A TDialog class that uses the dialog DELETE_DIALOG
//              found in inftlapp.rc
TDeleteDialog::TDeleteDialog(TDecoratedMDIFrame * frame, TDeleteBuffer *
  delete_buffer) : TDialog(frame, DELETE_DIALOG)
{
 //Creates a new edit control
 new TEdit(this, IDC_EDIT1, 101);
 //Sets up the transfer buffer so that we can get data from the control
```

```
 SetTransferBuffer(delete_buffer);
}

//Destructor
//Destroys the dialog box
TDeleteDialog::~TDeleteDialog()
{
}

//Responds if the user clicks on help in the Delete Dialog
void TDeleteDialog::CmHelp()
{
  string msg;
  string nl('\n');

  msg += "The function will delete the location and the keywords" + nl;
  msg += "associated with the location." + nl + nl;
  msg += "WARNING:  This will remove all trace of this location";
  MessageBox(msg.c_str(), "Deleting Help", MB_OK | MB_ICONSTOP);
}
```

BEGIN DIALOG.H

```
// This is the Search Buffer.  It holds the output of the control SEARCH_EDIT
struct TSearchBuffer
{
       char search_string[21];
 };

// This is the add buffer
struct TAddBuffer
{
    char add_kw1[21],        // Maps to Control ADD_KW1
         add_kw2[21],        // Maps to Control ADD_KW2
         add_kw3[21],        // Maps to Control ADD_KW3
         add_kw4[21],        // Maps to Control ADD_KW4
         add_kw5[21],        // Maps to Control ADD_KW5
         add_kw6[21],        // Maps to Control ADD_KW6
         add_kw7[21],        // Maps to Control ADD_KW7
         add_kw8[21],        // Maps to Control ADD_KW8
         add_kw9[21],        // Maps to Control ADD_KW9
         item[101],          // Maps to Control ITEM
         description[501];   // Maps to Control ITEM_DESCRIPTION
};

// This is the delete buffer.  It holds the output of the control DELETE_EDIT
struct TDeleteBuffer
{
    char delete_item[101];
};

class TSearchDialog: public TDialog {
public:
      TSearchDialog(TDecoratedMDIFrame * frame, TSearchBuffer* search_buffer);
      ~TSearchDialog();
protected:
   void CmHelp();
DECLARE_RESPONSE_TABLE(TSearchDialog);
};

class TAddDialog: public TDialog {
public:
   TAddDialog(TDecoratedMDIFrame * frame, TAddBuffer* add_buffer);
   ~TAddDialog();
protected:
   void CmHelp();
DECLARE_RESPONSE_TABLE(TAddDialog);
};

class TDeleteDialog: public TDialog {
public:
   TDeleteDialog(TDecoratedMDIFrame * frame, TDeleteBuffer* delete_buffer);
   ~TDeleteDialog();
protected:
```

```
   void CmHelp();
DECLARE_RESPONSE_TABLE(TDeleteDialog);
};
```

Appendix G BEGIN INFTLABD.CPP

```
/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftlabd.cpp
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Source file for implementation of InfotoolAboutDlg (TDialog).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include <owl\static.h>

#include <ver.h>

#include "inftlapp.h"
#include "inftlabd.h"


// Reading the VERSIONINFO resource.
class ProjectRCVersion {
public:
    ProjectRCVersion (TModule *module);
    virtual ~ProjectRCVersion ();

    BOOL GetProductName (LPSTR &prodName);
    BOOL GetProductVersion (LPSTR &prodVersion);
    BOOL GetCopyright (LPSTR &copyright);
    BOOL GetDebug (LPSTR &debug);

protected:
    LPBYTE      TransBlock;
    void FAR    *FVData;

private:
    // Don't allow this object to be copied.
    ProjectRCVersion (const ProjectRCVersion &);
    ProjectRCVersion & operator =(const ProjectRCVersion &);
};


ProjectRCVersion::ProjectRCVersion (TModule *module)
{
```

```
    char    appFName[255];
    DWORD   fvHandle;
    UINT    vSize;

    FVData = 0;

    module->GetModuleFileName(appFName, sizeof(appFName));
    DWORD dwSize = GetFileVersionInfoSize(appFName, &fvHandle);
    if (dwSize) {
        FVData  = (void FAR *)new char[(UINT)dwSize];
        if (GetFileVersionInfo(appFName, fvHandle, dwSize, FVData))
            if (!VerQueryValue(FVData, "\\VarFileInfo\\Translation", (void
FAR* FAR*)&TransBlock, &vSize)) {
                delete FVData;
                FVData = 0;
            }
    }
}


ProjectRCVersion::~ProjectRCVersion ()
{
    if (FVData)
        delete FVData;
}


BOOL ProjectRCVersion::GetProductName (LPSTR &prodName)
{
    UINT    vSize;
    char    subBlockName[255];

    wsprintf(subBlockName, "\\StringFileInfo\\%08lx\\%s", *(DWORD
*)TransBlock, (LPSTR)"ProductName");
    return FVData ? VerQueryValue(FVData, subBlockName, (void FAR*
FAR*)&prodName, &vSize) : FALSE;
}


BOOL ProjectRCVersion::GetProductVersion (LPSTR &prodVersion)
{
    UINT    vSize;
    char    subBlockName[255];

    wsprintf(subBlockName, "\\StringFileInfo\\%08lx\\%s", *(DWORD
*)TransBlock, (LPSTR)"ProductVersion");
    return FVData ? VerQueryValue(FVData, subBlockName, (void FAR*
FAR*)&prodVersion, &vSize) : FALSE;
}


BOOL ProjectRCVersion::GetCopyright (LPSTR &copyright)
{
    UINT    vSize;
```

```
    char    subBlockName[255];

    wsprintf(subBlockName, "\\StringFileInfo\\%08lx\\%s", *(DWORD
*)TransBlock, (LPSTR)"LegalCopyright");
    return FVData ? VerQueryValue(FVData, subBlockName, (void FAR*
FAR*)&copyright, &vSize) : FALSE;
}


BOOL ProjectRCVersion::GetDebug (LPSTR &debug)
{
    UINT    vSize;
    char    subBlockName[255];

    wsprintf(subBlockName, "\\StringFileInfo\\%08lx\\%s", *(DWORD
*)TransBlock, (LPSTR)"SpecialBuild");
    return FVData ? VerQueryValue(FVData, subBlockName, (void FAR*
FAR*)&debug, &vSize) : FALSE;
}


//{{InfotoolAboutDlg Implementation}}


//////////////////////////////////////////////////////////
// InfotoolAboutDlg
// ==========
// Construction/Destruction handling.
InfotoolAboutDlg::InfotoolAboutDlg (TWindow *parent, TResId resId, TModule
*module)
    : TDialog(parent, resId, module)
{
    // INSERT>> Your constructor code here.
}


InfotoolAboutDlg::~InfotoolAboutDlg ()
{
    Destroy();

    // INSERT>> Your destructor code here.
}


void InfotoolAboutDlg::SetupWindow ()
{
    LPSTR prodName, prodVersion, copyright, debug;

    // Get the static text who's value is based on VERSIONINFO.
    TStatic *versionCtrl = new TStatic(this, IDC_VERSION, 255);
    TStatic *copyrightCtrl = new TStatic(this, IDC_COPYRIGHT, 255);
    TStatic *debugCtrl = new TStatic(this, IDC_DEBUG, 255);

    TDialog::SetupWindow();
```

```
    // Process the VERSIONINFO.
    ProjectRCVersion applVersion(GetModule());

    // Get the product name, product version and legal copyright strings.
    applVersion.GetProductName(prodName);
    applVersion.GetProductVersion(prodVersion);
    applVersion.GetCopyright(copyright);

    // IDC_VERSION is the product name and version number, the initial value
of IDC_VERSION is
    // the word Version (in whatever language) product name VERSION product
version.
    char    buffer[255];
    char    versionName[128];
    versionCtrl->GetText(versionName, sizeof(versionName));
    wsprintf(buffer, "%s %s %s", prodName, versionName, prodVersion);
    versionCtrl->SetText(buffer);

    copyrightCtrl->SetText(copyright);

    // Only get the SpecialBuild text if the VERSIONINFO resource is there.
    if (applVersion.GetDebug(debug))
        debugCtrl->SetText(debug);
}
```

```
BEGIN INFTLABD.H

#if !defined(__inftlabd_h)              // Sentry, use file only if it's not
already included.
#define __inftlabd_h

/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftlabd.h
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Class definition for InfotoolAboutDlg (TDialog).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include "inftlapp.rh"                  // Definition of all resources.


//{{TDialog = InfotoolAboutDlg}}
class InfotoolAboutDlg : public TDialog {
public:
    InfotoolAboutDlg (TWindow *parent, TResId resId = IDD_ABOUT, TModule
*module = 0);
    virtual ~InfotoolAboutDlg ();

//{{InfotoolAboutDlgVIRTUAL_BEGIN}}
public:
    void SetupWindow ();
//{{InfotoolAboutDlgVIRTUAL_END}}
};    //{{InfotoolAboutDlg}}


#endif                                  // __inftlabd_h sentry.
```

Appendix H BEGIN INFTMDI1.CPP

```
/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftmdi1.cpp
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Source file for implementation of InfotoolMDIChild (TMDIChild).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include "inftlapp.h"
#include "inftmdi1.h"

#include <stdio.h>


//{{InfotoolMDIChild Implementation}}


//
// Build a response table for all messages/commands handled
// by InfotoolMDIChild derived from TMDIChild.
//
DEFINE_RESPONSE_TABLE1(InfotoolMDIChild, TMDIChild)
//{{InfotoolMDIChildRSP_TBL_BEGIN}}
    EV_WM_GETMINMAXINFO,
//{{InfotoolMDIChildRSP_TBL_END}}
END_RESPONSE_TABLE;


//////////////////////////////////////////////////////////
// InfotoolMDIChild
// ==========
// Construction/Destruction handling.
InfotoolMDIChild::InfotoolMDIChild (TMDIClient &parent, const char far *title,
TWindow *clientWnd, BOOL shrinkToClient, TModule *module)
    : TMDIChild (parent, title, clientWnd, shrinkToClient, module)
{
    // INSERT>> Your constructor code here.

}
```

```
InfotoolMDIChild::~InfotoolMDIChild ()
{
    Destroy();

    // INSERT>> Your destructor code here.

}


//
// Paint routine for Window, Printer, and PrintPreview for an TEdit client.
//
void InfotoolMDIChild::Paint (TDC& dc, BOOL, TRect& rect)
{
    InfotoolApp *theApp = TYPESAFE_DOWNCAST(GetApplication(), InfotoolApp);
    if (theApp) {
        // Only paint if we're printing and we have something to paint,
otherwise do nothing.
        if (theApp->Printing && theApp->Printer && !rect.IsEmpty()) {
            // Use pageSize to get the size of the window to render into.  For
a Window it's the client area,
            // for a printer it's the printer DC dimensions and for print
preview it's the layout window.
            TSize   pageSize(rect.right - rect.left, rect.bottom - rect.top);

            HFONT   hFont = (HFONT)GetClientWindow()->GetWindowFont();
            TFont   font("Arial", -12);
            if (hFont == 0)
              dc.SelectObject(font);
            else
              dc.SelectObject(TFont(hFont));

            TEXTMETRIC  tm;
            int fHeight = (dc.GetTextMetrics(tm) == TRUE) ? tm.tmHeight +
tm.tmExternalLeading : 10;

            // How many lines of this font can we fit on a page.
            int linesPerPage = MulDiv(pageSize.cy, 1, fHeight);
            if (linesPerPage) {
                TPrintDialog::TData &printerData = theApp->Printer-
>GetSetup();

                int maxPg = 1;

                // Get the client class window (this is the contents we're
going to print).
                TEdit *clientEditWindow = 0;
                TListBox *clientListWindow = 0;

                clientEditWindow = TYPESAFE_DOWNCAST(GetClientWindow(),
TEdit);
                if (clientEditWindow)
```

```
                    maxPg = ((clientEditWindow->GetNumLines() / linesPerPage)
+ 1.0);
                else {
                    clientListWindow = TYPESAFE_DOWNCAST(GetClientWindow(),
TListBox);
                    if (clientListWindow)
                        maxPg = ((clientListWindow->GetCount() / linesPerPage)
+ 1.0);
                }

                // Compute the number of pages to print.
                printerData.MinPage = 1;
                printerData.MaxPage = maxPg;

                // Do the text stuff:
                int     fromPage = printerData.FromPage == -1 ? 1 :
printerData.FromPage;
                int     toPage = printerData.ToPage == -1 ? 1 :
printerData.ToPage;
                char    buffer[255];
                int     currentPage = fromPage;

                while (currentPage <= toPage) {
                    int startLine = (currentPage - 1) * linesPerPage;
                    int lineIdx = 0;
                    while (lineIdx < linesPerPage) {
                        // If the string is no longer valid then there's
nothing more to display.
                        if (clientEditWindow) {
                            if (!clientEditWindow->GetLine(buffer,
sizeof(buffer), startLine + lineIdx))
                                break;
                        }
                        if (clientListWindow) {
                            if (clientListWindow->GetString(buffer, startLine
+ lineIdx) < 0)
                                break;
                        }
                        dc.TabbedTextOut(TPoint(0, lineIdx * fHeight), buffer,
lstrlen(buffer), 0, NULL, 0);
                        lineIdx++;
                    }
                    currentPage++;
                }
            }
        }
    }
}


void InfotoolMDIChild::EvGetMinMaxInfo (MINMAXINFO far& minmaxinfo)
{
    InfotoolApp *theApp = TYPESAFE_DOWNCAST(GetApplication(), InfotoolApp);
    if (theApp) {
```

```
        if (theApp->Printing) {
            minmaxinfo.ptMaxSize = TPoint(32000, 32000);
            minmaxinfo.ptMaxTrackSize = TPoint(32000, 32000);
            return;
        }
    }
    TMDIChild::EvGetMinMaxInfo(minmaxinfo);
}
```

```
BEGIN INFTMDI1.H

#if !defined(__inftmdi1_h)              // Sentry, use file only if it's not
already included.
#define __inftmdi1_h

/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftmdi1.h
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Class definition for InfotoolMDIChild (TMDIChild).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include <owl\editfile.h>
#include <owl\listbox.h>

#include "inftlapp.rh"            // Definition of all resources.


//{{TMDIChild = InfotoolMDIChild}}
class InfotoolMDIChild : public TMDIChild {
public:
    InfotoolMDIChild (TMDIClient &parent, const char far *title, TWindow
*clientWnd, BOOL shrinkToClient = FALSE, TModule* module = 0);
    virtual ~InfotoolMDIChild ();

//{{InfotoolMDIChildVIRTUAL_BEGIN}}
public:
    virtual void Paint (TDC& dc, BOOL erase, TRect& rect);
//{{InfotoolMDIChildVIRTUAL_END}}
//{{InfotoolMDIChildRSP_TBL_BEGIN}}
protected:
    void EvGetMinMaxInfo (MINMAXINFO far& minmaxinfo);
//{{InfotoolMDIChildRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(InfotoolMDIChild);
};    //{{InfotoolMDIChild}}


#endif                                          // __inftmdi1_h sentry.
```

Appendix I BEGIN INFTMDIC.CPP

```
/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftmdic.cpp
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Source file for implementation of InfotoolMDIClient (TMDIClient).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include <dir.h>

#include "inftlapp.h"
#include "inftmdic.h"
#include "inftmdi1.h"
#include "apxprint.h"
#include "apxprev.h"


//{{InfotoolMDIClient Implementation}}


//
// Build a response table for all messages/commands handled
// by InfotoolMDIClient derived from TMDIClient.
//
DEFINE_RESPONSE_TABLE1(InfotoolMDIClient, TMDIClient)
//{{InfotoolMDIClientRSP_TBL_BEGIN}}
    EV_COMMAND(CM_FILEPRINT, CmFilePrint),
    EV_COMMAND(CM_FILEPRINTERSETUP, CmFilePrintSetup),
    EV_COMMAND(CM_FILEPRINTPREVIEW, CmFilePrintPreview),
    EV_COMMAND_ENABLE(CM_FILEPRINT, CmPrintEnable),
    EV_COMMAND_ENABLE(CM_FILEPRINTERSETUP, CmPrintEnable),
    EV_COMMAND_ENABLE(CM_FILEPRINTPREVIEW, CmPrintEnable),
    EV_WM_DROPFILES,
//{{InfotoolMDIClientRSP_TBL_END}}
END_RESPONSE_TABLE;


//////////////////////////////////////////////////////////
// InfotoolMDIClient
```

```
// ===========
// Construction/Destruction handling.
 InfotoolMDIClient::InfotoolMDIClient ()
 : TMDIClient ()
{
    // Change the window's background color
    SetBkgndColor(RGB(0xff, 0xff, 0xff));

    ChildCount = 0;

    // INSERT>> Your constructor code here.

}


 InfotoolMDIClient::~InfotoolMDIClient ()
{
    Destroy();

    // INSERT>> Your destructor code here.

}


////////////////////////////////////////////////////////
// InfotoolMDIClient
// ===========
// MDIClient site initialization.
void InfotoolMDIClient::SetupWindow ()
{
    // Default SetUpWindow processing.
    TMDIClient::SetupWindow ();

    // Accept files via drag/drop in the client window.
    DragAcceptFiles(TRUE);
}


////////////////////////////////////////////////////////
// InfotoolMDIClient
// ==========
// Menu File Print command
void InfotoolMDIClient::CmFilePrint ()
{
    //
    // Create Printer object if not already created.
    //
    InfotoolApp *theApp = TYPESAFE_DOWNCAST(GetApplication(), InfotoolApp);
    if (theApp) {
        if (!theApp->Printer)
            theApp->Printer = new TPrinter;

        //
        // Create Printout window and set characteristics.
```

```
        //
        APXPrintOut printout(theApp->Printer, Title, GetActiveMDIChild(),
TRUE);

        theApp->Printing = TRUE;

        //
        // Bring up the Print dialog and print the document.
        //
        theApp->Printer->Print(GetActiveMDIChild()->GetClientWindow(),
printout, TRUE);

        theApp->Printing = FALSE;
    }
}


////////////////////////////////////////////////////////
// InfotoolMDIClient
// ==========
// Menu File Print Setup command
void InfotoolMDIClient::CmFilePrintSetup ()
{
    InfotoolApp *theApp = TYPESAFE_DOWNCAST(GetApplication(), InfotoolApp);
    if (theApp) {
        if (!theApp->Printer)
            theApp->Printer = new TPrinter;

        //
        // Bring up the Print Setup dialog.
        //
        theApp->Printer->Setup(this);
    }
}


////////////////////////////////////////////////////////
// InfotoolMDIClient
// ==========
// Menu File Print Preview command
void InfotoolMDIClient::CmFilePrintPreview ()
{
    InfotoolApp *theApp = TYPESAFE_DOWNCAST(GetApplication(), InfotoolApp);
    if (theApp) {
        if (!theApp->Printer)
            theApp->Printer = new TPrinter;

        theApp->Printing = TRUE;

        PreviewWindow *prevW = new PreviewWindow(Parent, theApp->Printer,
GetActiveMDIChild(), "Print Preview", new TLayoutWindow(0));
        prevW->Create();

        GetApplication()->BeginModal(GetApplication()->MainWindow);
```

```
        // We must destroy the preview window explicitly.  Otherwise, the
window will not be destroyed until
        // it's parent the MainWindow is destroyed.
        prevW->Destroy();
        delete prevW;

        theApp->Printing = FALSE;
    }
}


//////////////////////////////////////////////////////
// InfotoolMDIClient
// ==========
// Menu enabler used by Print, Print Setup and Print Preview.
void InfotoolMDIClient::CmPrintEnable (TCommandEnabler &tce)
{
    if (GetActiveMDIChild()) {
        InfotoolApp *theApp = TYPESAFE_DOWNCAST(GetApplication(),
InfotoolApp);
        if (theApp) {
            // If we have a Printer already created just test if all is okay.
            // Otherwise create a Printer object and make sure the printer
      // really exists and then delete the Printer object.
            if (!theApp->Printer) {
                theApp->Printer = new TPrinter;

                tce.Enable(theApp->Printer->GetSetup().Error == 0);
            } else
                tce.Enable(theApp->Printer->GetSetup().Error == 0);
        }
    } else
        tce.Enable(FALSE);
}


void InfotoolMDIClient::EvDropFiles (TDropInfo)
{
    Parent->ForwardMessage();
}
```

```
BEGIN INFTMDIC.H

#if !defined(__inftmdic_h)              // Sentry, use file only if it's not
already included.
#define __inftmdic_h

/*  Project Infotool
    HBT Consulting
    Copyright © 1993. All Rights Reserved.

    SUBSYSTEM:    infotool.exe Application
    FILE:         inftmdic.h
    AUTHOR:       Todd E. Toles


    OVERVIEW
    ========
    Class definition for InfotoolMDIClient (TMDIClient).
*/


#include <owl\owlpch.h>
#pragma hdrstop

#include <owl\opensave.h>

#include "inftlapp.rh"            // Definition of all resources.


//{{TMDIClient = InfotoolMDIClient}}
class InfotoolMDIClient : public TMDIClient {
public:
    int                     ChildCount;                 // Number of child
window created.

    InfotoolMDIClient ();
    virtual ~InfotoolMDIClient ();

    void OpenFile (const char *fileName = 0);

private:
    void LoadTextFile ();

//{{InfotoolMDIClientVIRTUAL_BEGIN}}
protected:
    virtual void SetupWindow ();
//{{InfotoolMDIClientVIRTUAL_END}}

//{{InfotoolMDIClientRSP_TBL_BEGIN}}
protected:
    void CmFilePrint ();
    void CmFilePrintSetup ();
    void CmFilePrintPreview ();
```

```
    void CmPrintEnable (TCommandEnabler &tce);
    void EvDropFiles (TDropInfo);
//{{InfotoolMDIClientRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(InfotoolMDIClient);
};    //{{InfotoolMDIClient}}


#endif                                          // __inftmdic_h sentry.
```

Appendix J BEGIN INFTLAPP.DEF

```
;------------------------------------------------
;   Main Infotool
;   HBT Consulting
;   Copyright © 1993. All Rights Reserved.
;
;   SUBSYSTEM:    infotool.exe Module Defintion File
;   FILE:         inftlapp.def
;   AUTHOR:       Todd E. Toles
;
;------------------------------------------------

NAME Infotool

DESCRIPTION 'Infotool Application - Copyright © 1993. All Rights Reserved.'
EXETYPE     WINDOWS
CODE        PRELOAD MOVEABLE DISCARDABLE
DATA        PRELOAD MOVEABLE MULTIPLE
HEAPSIZE    4096
STACKSIZE   8192
```