

Joe Student

COMP122

LAB 1: Telephone Number Database and Autodialer

January 1, 1980

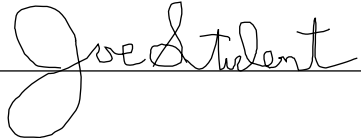
Due Week 16

For: Sr. Professor Wheeler

OPERATIONAL SIGN OFF _____

FINAL SIGN OFF _____

"THIS IS THE ORIGINAL WORK OF JOE STUDENT"



Telephone Number Database and Autodialer - End User Documentation

Introduction:


The purpose of this program is twofold. First, it maintains a complete database of names, addresses, and telephone numbers. In that respect, it is a digital replacement for a pencil-and-paper address book. Second, the program automatically dials telephone numbers and monitors caller ID. With this program, the user can:

- Look up a name or address from his or her personal database.
- Automatically dial numbers from that database at the press of a key.
- Use the caller ID feature of the user's modem to automatically add callers into the program's database.

Starting the Program:

To start the program, insert the provided disk into drive A: and type the following at the DOS prompt:

A:TND

TIP: On a Windows computer, type "  - R" to enter this command.

The program will begin running. The following menu will be presented on the screen:

```
Welcome to the Telephone Autodialer

There are 100 entries in the phone book.

1) Add a number to the phone book
2) Browse the telephone book
3) Search for a name or number in the book
4) Set Caller ID Options
5) Review the last 50 telephone calls
6) Exit the program
```

What do you want to do (1-5)?

This program (which doesn't actually exist) is a little more involved than most of the projects you'll create in this course. Therefore, the documentation you create for projects isn't likely to be quite this long!



When the program first runs, the phone book is empty. It is automatically saved to the disk each time an entry is made. To manually make an entry into the book, enter `[N]`. The following screen will pop up:

```
Manual Number Entry
```

```
Name?  
Telephone Number?  
Area Code?  
Address?  
City?  
State?
```

To enter the number, just answer the questions in sequence. If you mistakenly enter this command by accident, just press the `[Escape]` key.

Browsing the Telephone Book:

Pressing `[B]` will allow you to view the contents of the telephone book. In this mode, the screen looks like this:

```
Database Entry 1  
  
James Brown      (816) 844-1212  
12345 College  
Overland Park  KS  
  
[+] Next          [-] Previous  
[D] Dial          [DEL] Delete  
                  [ESC] Exit to menu
```

The options are shown at the bottom of the display. You can press `[D]` at any time during browsing to dial the number. The `[+]` and `[-]` keys move forward or backward through the database.

You can also jump quickly to all entries that start with a certain letter of the alphabet. For example, typing `[Z]` will cause the screen to show the first entry with a last name starting with "Z."

Press `[Escape]` to exit this mode and return to the main menu.

Searching for Names and Numbers:

Pressing `[S]` will cause the program to enter search mode. The screen will prompt you to enter a name (or fragment) to search for. (You may also enter a telephone number, or portion thereof):

Search Mode

Enter the data to search for, or [ESC] to exit: _____

Upon entering the information, the program will search for the information. The first entry that matches the search criterion will be returned, and the program will revert to *browse* mode as above. Press Escape to return to the main menu when you're finished.

Setting Caller ID Options:

Press 4 to set caller ID options. The following sub-menu will appear. The word "ENABLED" or "DISABLED" will follow each item in the menu. *Press the number corresponding that menu item to toggle the state of that item.* For example, the default settings are as follows:

Caller ID Options

1. Record Incoming Numbers : DISABLED
2. Add Incoming to Database: DISABLED
3. Block Anonymous Callers : DISABLED

Press (1-3) to toggle option, or [ESC] to return to the main menu.

Choose:

Pressing 1 will turn the recording of incoming numbers *on*. Pressing 1 again will turn that option back off.

Press Escape to save the options and return to the main menu.

Reviewing Telephone Calls:

If you've turned on the *Record Incoming Numbers* option in the Caller ID Options screen, the program will keep track of the last 50 calls to your phone line. Calls are held in a FIFO (First-In, First-Out) buffer, so only the last 50 calls are maintained.

To see this information, press 5 at the main menu. The program will open a screen very similar to that of the *browse* function:

Caller 1 (02/15/1986 14:46)

James Brown (816) 844-1212

[+] Next [-] Previous
[D] Dial [ESC] Exit to menu
[A] Add to database

The command keys work identically to those for the *browse* command. The main difference is that typing **[A]** will add this caller's name and telephone number to the permanent database. The address fields will be blank, as caller ID doesn't provide that information.

- The [A]dd to database command will be inactive (grey) if the caller already exists within the database.
- There's no delete command in this mode. The caller's information will automatically be deleted as other caller information is accumulated.

Exiting The Program:

To exit the Telephone Autodialer, press **[G]**. All information entered into the program is saved.

- You can let this program operate minimized under Windows within a DOS window. It will continue to log calls, even if it is minimized.

Sample Run of Telephone Autodialer

Welcome to the Telephone Autodialer

There are no entries in the phone book.

- 1) Add a number to the phone book
- 2) Browse the telephone book
- 3) Search for a name or number in the book
- 4) Set Caller ID Options
- 5) Review the last 50 telephone calls
- 6) Exit the program

What do you want to do (1-5)? 1

Manual Number Entry

Name? Joe Cool
Telephone Number? 844-1212
Area Code? 816
Address? 12345 College Blvd
City? Overland Park
State? KS

Welcome to the Telephone Autodialer

There is 1 entry in the phone book.

- 1) Add a number to the phone book
- 2) Browse the telephone book
- 3) Search for a name or number in the book
- 4) Set Caller ID Options
- 5) Review the last 50 telephone calls
- 6) Exit the program

What do you want to do (1-5)? 2

Database Entry 1

James Brown (816) 844-1212
12345 College
Overland Park KS

[+] Next [-] Previous
[D] Dial [DEL] Delete
 [ESC] Exit to menu

Telephone Autodialer Program Listing

```
//  
// Program Name: Telephone Autodialer  
// Author: Joe Student  
// Date: January 1, 1980  
//  
// Revision history: 1/10/80 -- Fixed overrun p  
//                   1/12/80 -- Modified communicat  
//                   modem compatibility.
```

Your name must appear at the top of all program listings. It's also a good idea to keep a revision history, as well as any other related information.

```
////////////////////////////////////  
// Function: OnPaint()  
// Description: This function responds to the WM_PAINT message that is  
//              generated when a previously-covered portion of the  
//              window is uncovered. The global background bit-map is  
//              also repainted.  
//  
// Parameters passed: None, the OS gives information through CpaintDC  
// Parameters returned: None. The window is repainted.  
////////////////////////////////////
```

```
void CSignalWindow::OnPaint()  
{  
  
//  
// The following instruction forces the ENTIRE client area  
// to always be repainted. It MUST take place before the d  
// of the CPaintDC so that the CPaintDC PAINTSTRUCT will be filled  
// with proper information.  
//  
// This approach is used because calculating the bounding rectangle  
// of signal waveforms might be time consuming, so the compromise  
// here is to always repaint the entire window.  
//  
// Failure to repaint the window properly could leave pieces of  
// the old waveform in the client area, which invalidates the  
// background!  
//
```

Header comment tells what purpose of function is. Many people use the "/" characters to make a box.

```
InvalidateRect(NULL);  
  
//  
// Build a CPaintDC (BeginPaint() called here)  
//  
CPaintDC dc(this); // device context for painting  
  
CRect aRect; // Will hold the rectangle of the client area  
int nResult;  
  
GetClientRect(&aRect); // Get dimensions of our "play" area  
// (Needed for painting the window background)  
  
m_bIsPainting = TRUE; // Let draw routine know we are painting,  
// since it may be called upon during the  
// painting process by a multithreaded  
// application.
```

The body of this program isn't an autodialer, however, it is an example of good commenting style. Note that each *concept* or *thought* is thoroughly explained.

Don't worry if the code looks complicated. The style is what is important!

```
//  
// If we had created a bit-mapped background before, delete it, since  
// we're creating a new one.  
//
```

```
if (m_pBackground)  
    delete m_pBackground;  
  
//  
// Paint the gridlines and produce the scale legend
```

Spacing program statements gives a clean appearance, making reading (and understanding!) much easier.

```

//
PaintNewBackGround( &dc, m_WindowBackgroundColor,
                    m_ScaleColor, aRect );

//
// Copy the window frame client area into the m_pBackground
// bit map (which must be created first in the same size
// dimensions as the window client area)
//
m_pBackground = new CBitmap;
//
// When creating the CBitmap object that will hold the
// image, its characteristics must be made compatible
// DEVICE CONTEXT "memdc", which is forced to be compatible
// the drawing DC by "CreateCompatibleDC()"
//
int nPlanes = dc.GetDeviceCaps( PLANES );
int nBitPixel = dc.GetDeviceCaps( BITSPIXEL );

nResult = m_pBackground->CreateBitmap( m_cx, // width
                                     m_cy, // Height
                                     nPlanes, // Planes
                                     nBitPixel, // nBitCount
                                     NULL); // const void* lpBits

// create an in-memory device context
CDC memdc;
memdc.CreateCompatibleDC(&dc);
memdc.SelectObject( m_pBackground );
//
// Copy our newly-generated background to the bitmap selected into
// the memory device context "memdc"
//
nResult = memdc.BitBlt(
    0,0, // (x,y) of destination
    m_cx,m_cy, // (width,height) of destination
    &dc, // copy FROM this dc
    0,0, // copy FROM this (x,y) in src dc
    SRCCOPY); // copy mode

//
// Try painting the last waveform we were passed. If none,
// don't paint anything, since owner hasn't sent us any waveform
// data yet.
//
if ((m_pOldWaveform != NULL) && m_DisDrawing == FALSE)
{
    m_bIsPainting = FALSE; // Must reset this or drawing routine
                          // won't do anything

    DrawWaveform(&dc, (*m_pOldWaveform),
                 m_DCOffset, m_color, m_FullScale);
}

//
// Let others know that we're done painting so that they can resume
// execution
//
m_bIsPainting = FALSE;
}

```

Related statements (or in this case, parts of a statement) should be indented appropriately and consistently.

The indentation of this block of code (marked by the "{}" characters) makes it clear what belongs -- and what doesn't!