Reporting

WEEK
3

An application that tracks data is great. Being able to view information on the screen makes the application useful; however, most applications require the capability to access the information in a printed format. Today you will:

- ☐ Add additional menus to the *Record of Records!* program to support reporting.
- ☐ Discuss the concept behind reporting.
- ☐ Do prototypes for a few reports.
- ☐ Add reports that access a single file.
- ☐ Work with reports that access multiple files.
- ☐ Work with a report that only prints selected records.
- ☐ Understand the importance of providing flexibility for reporting.

Is Reporting Important?

On the entry and edit windows, you can only see records one at a time. Many times this is inadequate. Additionally, there are times when you will want to share the information that is stored in your database with others. Reporting enables you to view or print your data in forms different from the form that you see it in on the entry and edit screen.

Before you start creating reports, you should do a bit of preplanning just as you did with the entry and edit screens. With the entry and edit screens, you created a specification. With reporting, you could also create a specification. On simpler reports, a specification may not be needed. What is almost always needed before coding a report is a prototype.

Prototyping Reports

While specifications aren't always required, a prototype is almost always required. If you are developing an application for others, you will be able to present them with a prototype. If they suggest changes, you can change the prototype. You wouldn't have to change the code because you would not have started it yet. Changing the prototype is much easier than changing the code.

The *Record of Records!* application will have several reports. There will be three different reports listing data from each file. These will be followed by a single report that will be accessible in multiple ways.

Simple List Reports

Three of the four reports will simply list specific information out of each database. These lists are good for verifying that each database contains the information that you expect. In addition, lists can serve many other uses. For example, by printing a list of medium codes and their descriptions, you can use it when entering Musical Items. You could print a list from the Group file to show others what groups you listen to. A list of your musical items would be good to place in a safe place for insurance purposes. Lists are a common type of report.

Note: A list is a report that simply lists fields from a file. Lists generally follow the format of one record per line. You will see this in the prototypes presented later.

Before being able to create these reports, you should look at the prototypes. Following will be the prototypes for the Medium Code List and the Group Information Code List. Creating a prototype for the Musical Items List will be left for you to do in an exercise at the end of today.

The List of Medium Codes Prototype

The first prototype presented is the List of Medium Codes prototype.

```
            List of Medium Codes
                 99/99/99


Code    Description
----    -----------

 XX     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XX     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 AL     Album
 CD     Compact Disc

            *** End of Report ***
```

There are several features about this prototype you should be aware of. First is the date that is present under the heading. It is always good to include the current date on a report. This enables the person using the report to know when the information was printed. If an old date is on the report, then there is a good chance the information is out-of-date.

This prototype shows a great deal more. The title and column headings are presented exactly as they will appear, which includes their line spacing. The data is then presented. First, several Xs are presented followed by two examples of "real" data. The purpose of the Xs is to show exactly where the data will be. This includes the size of the data.

One final comment should be made about the prototype. The date is displayed with 9s instead of Xs. This is because the date is composed of numbers. The code and description is composed of Xs because they are alphanumeric fields. In creating your prototypes, you should use 9s for numeric fields and Xs for alphanumerics.

Tip: Not all reports will print every field. In addition, some longer fields may be truncated (that is, chopped off). The number of Xs should signify how may digits or characters should be printed.

The Group Information List's Prototype

The Group file contains more information than will fit on a single line. Because of this, you must be a little more creative in creating the prototype.

```
                List of Group Information
                       99/99/99


                                 Date
Group                            Formed      Type of Music
-------------------------      --------    --------------------

XXXXXXXXXXXXXXXXXXXXXXXXX     99/99/99     XXXXXXXXXXXXXXXXXXX

                    Members:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


XXXXXXXXXXXXXXXXXXXXXXXXX     99/99/99     XXXXXXXXXXXXXXXXXXX

                    Members:   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                               XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
                              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                              XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

COWBOY JUNKIES                01/01/87    Alternative

                    Members:  Alan Anton
                              Margo Timmins
                              Michael Timmins
                              Peter Timmins

Violent Femmes                01/01/83    Alternative

                    Members:  Gordon Gano
                              Brian Ritchie
                              Victor DeLorenzo


                    *** End of Report ***
```

This prototype follows the same constructs as the previous prototpye. Xs and 9s mark the location of the fields. Because all the fields don't fit on a single line, part of the data falls on the following lines. You should notice that the additional lines are indented several spaces so that it is easy to follow where new records start.

Expert Tip: By including a couple of "real" records on the prototype, it is often easier to get a better idea of how the report will really look.

Note: The Musical Items List will be prototyped in an exercise at the end of the day. There will also be an exercise asking you to create this report.

Complex Reports

In addition to the three list reports that will be added to the application, a fourth report will also be added. This report will be much more complex. The complexity of the report will make it much more useful.

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX                           Today: 99/99/99

Group:       XXXXXXXXXXXXXXXXXXXXXXXXX

Group Desc:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
             XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
             XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Type of Music:  XXXXXXXXXXXXXXXXXXXX

Medium:  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Date Purchased: 99/99/99

Cost:   $999.99
Value:  $999.99

SONGS:

Track     Song Title                                 Time
----     ----------------------------------------   ----
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99
 99       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   99:99


 ---------------------- PAGE BREAK --------------------------


Walking on the Moon                                      Today: 99/99/99

Group:       Philippe Kahn

Group Desc:  Jazz Music by the Owner of Borland International. Features
             several famous people


Type of Music:  Jazz
```

```
Medium:  Compact Disc

Date Purchased: 12/13/92

Cost:   $ 14.95
Value:  $342.00

SONGS:

Track     Song Title                                Time
----     ----------------------------------------   ----
 01       OOPs!                                     05:06
 02       Interlude                                 00:00
 03       Walkin' on the Moon                       07:28
 04       Interlude                                 00:00
 05       Epistrophy                                04:35
 06       Interlude                                 00:00
 07       S, E and L                                04:10
 08       All's Well That Ends                      08:12
 09       Interlude                                 00:00
 10       This Masquerade                           05:48
 11       Interlude                                 00:00
 12       Ralph's Piano Waltz                       05:45
 13       Interlude                                 00:00
 14       Calor                                     08:07
 15       Interlude                                 00:00
 16       Better Days                               03:49
 17       Interlude                                 00:00
 18       Mopti                                     05:46
 19       Interlude                                 00:00
 20       Silence                                   05:52
```


As you can see, this report is formatted differently from that of the previous lists. In addition, this report contains information from not only the Musical Items file, but also the Group file and the Medium Code file. The formatting of this report has been changed so that only one musical item will be printed per page. The complexity of this report will require a slightly different approach to produce than the lists.

If you printed this layout for every record that could possibly be in your database, you could use a great deal of paper. Because of this, you will want to add selection criteria. Later today, you will add an option that will allow this report to be printed for either all of the records in the file or for only a single musical item.

There are some other issues involved with this report that were not present in the previously prototyped lists. What do you do if you print a musical item that does not have a corresponding medium code or group? In these cases, you may be missing some information that is expected based on the prototype. This is a situation that should be addressed at the time you do the prototypes. In the case of the Musical Items report,

you will print UNKNOWN in the medium code description, and you will leave the group information blank. In addition, if you don't print the group information, you should suppress the blank lines from the report. Sound complicated? Later today, you will see the code necessary to complete this report.

Expert Tip: Because of the complexity of the Musical Items Information report, you may want to create a specification. The specification would contain information such as the selection criteria and what to do if some of the data is missing.

Creating the Reports

Before creating the reports, you will first need to set up the main application to receive them. Based on what was presented earlier, you should have an idea of what will need to be added to the *Record of Records!* application. There already was a reporting menu option included in the Main menu. Because you know what reports need to be added, you should be able to include a reporting menu. A menu should be created to contain the following options:

1. Detailed Information
2. List of Musical Items
3. List of Groups
4. List of Medium Codes

These options are listed in the order that they are most likely to be used. The users of the application are expected to use the Detailed Information Report (on Musical Items) the most. They are expected to use the List of Medium Codes the least. Listing 19.1 is the code necessary to create this menu.

As stated earlier, the Detailed Report will have an option. The users will be able to print all of the musical items, or they will be able to select a specific musical item to print. To provide your users with this option, a third menu will be included accessible from the Detailed Information option. This menu will contain two reporting options, All Items and One Item. Additionally, it will contain a Return option, which will remove the menu. The code for this menu is also included in Listing 19.1.

Type

Listing 19.1. REC_RPTG.C. The reporting menu code.

```
1:  /*=========================================================
2:   * Filename: REC_RPTG.c
3:   *           RECORD OF RECORDS - Version 1.0
4:   *
5:   * Author:   Bradley L. Jones
6:   *           Gregory L. Guntle
7:   *
8:   * Purpose:  The Reporting menus.
9:   *
10:  * Note:     Assumes 80 columns by 25 columns for screen.
11:  *=========================================================*/
12: #include <string.h>
13: #include <stdio.h>
14: #include <ctype.h>
15: #include "tyac.h"
16: #include "records.h"
17:
18: /*--------------------*
19:  *     prototypes     *
20:  *--------------------*/
21: #include "recofrec.h"
22:
23: int do_detail_album_menu(void);
24:
25: /*=========================================================*
26:  *                    main()                               *
27:  *=========================================================*/
28:
29: int do_reporting(void)
30: {
31:   int  rv       = 0;
32:   int  cont     = TRUE;
33:   int  menu_sel = 0;
34:   char *saved_screen = NULL;
35:
36:
37:   char *rpt_menu[10] = {
38:           "1. Detailed Information", "1Dd",
39:           "2. List of Musical Item", "2Mm",
40:           "3. List of Groups      ", "3Gg",
41:           "4. List of Medium Codes", "4Mm",
42:           "5. Return to Main Menu ", "5RrEeQq" };
43:
44:   char MENU_EXIT_KEYS[MAX_KEYS] = {F3, F10, ESC_KEY};
45:
46:   while( cont == TRUE )               /* loop in temp menu */
47:   {
```

continues

Listing 19.1. continued

```
48:      saved_screen = save_screen_area(10, 21, 28, 58 );
49:
50:      rv = display_menu(12, 30, DOUBLE_BOX, rpt_menu, 10,
51:                        MENU_EXIT_KEYS, &menu_sel, NO_LR_ARROW,
52:                        SHADOW);
53:
54:      switch( rv )
55:      {
56:         case ENTER_KEY: /* accept selection */
57:         case CR:
58:               switch( menu_sel )
59:               {
60:                  case 1:  /* Menu option 1 */
61:                           cursor_on();
62:                           do_detail_album_menu();
63:                           break;
64:
65:                  case 2:  /* Menu option 2 */
66:                           cursor_on();
67:  //                         list_musical_items();
68:                           display_msg_box("Report 2...",
69:                              ct.help_fcol, ct.help_bcol );
70:                           break;
71:
72:                  case 3:  /* Menu option 3 */
73:                           cursor_on();
74:  //                         list_groups();
75:                           display_msg_box("Report 3...",
76:                              ct.help_fcol, ct.help_bcol );
77:                           break;
78:
79:                  case 4:  /* Reporting */
80:                           cursor_on();
81:  //                         list_medium_codes();
82:                           display_msg_box("Report 4...",
83:                              ct.help_fcol, ct.help_bcol );
84:                           break;
85:
86:                  case 5:  /* exit */
87:                           cont = FALSE;
88:                           break;
89:
90:                  default: /* continue looping */
91:                           boop();
92:                           break;
93:               }
94:               break;
95:
96:        case ESC_KEY: rv = ENTER_KEY;  /* don't exit totally */
```

```
97:                     cont = FALSE;     /* exit */
98:                     break;
99:
100:       case F3:     /* exiting */
101:                    cont = FALSE;
102:                    break;
103:
104:       case F10:    /* action bar */
105:                    rv = do_main_actionbar();
106:
107:                    if( rv == F3 )
108:                        cont = FALSE;
109:
110:                    break;
111:
112:       default:     boop();
113:                    break;
114:     }
115:   }
116:   restore_screen_area(saved_screen);
117:
118:   return(rv);
119: }
120:
121: /*-----------------------------------------------*
122:  * Detailed Musical Item Report                  *
123:  *-----------------------------------------------*/
124:
125: int do_detail_album_menu(void)
126: {
127:   int  rv       = 0;
128:   int  cont     = TRUE;
129:   int  menu_sel = 0;
130:   char *saved_screen = NULL;
131:
132:
133:   char *album_menu[6] = {
134:           "1. All Items", "1Aa",
135:           "2. One Item ", "2Oo",
136:           "3. Return   ", "3RrEeQq" };
137:
138:   char MENU_EXIT_KEYS[MAX_KEYS] = {F3, F10, ESC_KEY};
139:
140:   while( cont == TRUE )               /* loop in temp menu */
141:   {
142:     saved_screen = save_screen_area(13, 19, 35, 55 );
143:
144:     rv = display_menu(14, 40, DOUBLE_BOX, album_menu, 6,
145:                       MENU_EXIT_KEYS, &menu_sel, NO_LR_ARROW,
146:                       SHADOW);
```

continues

Listing 19.1. continued

```
147:
148:     switch( rv )
149:     {
150:        case ENTER_KEY: /* accept selection */
151:        case CR:
152:              switch( menu_sel )
153:              {
154:                 case 1:  /* Menu option 1 */
155:                          cursor_on();
156: //                         music_rpt(0);
157:                          display_msg_box("Do All of em....",
158:                              ct.help_fcol, ct.help_bcol );
159:                          break;
160:
161:                 case 2:  /* Menu option 2 */
162:                          cursor_on();
163: //                         music_rpt(1);
164:                          display_msg_box("Do only one...",
165:                              ct.help_fcol, ct.help_bcol );
166:                          break;
167:
168:                 case 3:  /* Exit menu */
169:                          cont = FALSE;
170:                          break;
171:
172:                 default: /* continue looping */
173:                          boop();
174:                          break;
175:              }
176:              break;
177:
178:       case ESC_KEY: rv = ENTER_KEY;  /* so don't exit clear out */
179:                     cont = FALSE;    /* exit */
180:                     break;
181:
182:       case F3:      /* exiting */
183:                     cont = FALSE;
184:                     break;
185:
186:       case F10:     /* action bar */
187:                     rv = do_main_actionbar();
188:
189:                     if( rv == F3 )
190:                         cont = FALSE;
191:
192:                     break;
193:
194:       default:      boop();
195:                     break;
```

```
196:      }
197:    }
198:    restore_screen_area(saved_screen);
199:
200:    return(rv);
201: }
202:
203: /*=========================================================*
204:  *                      end of listing                     *
205:  *=========================================================*/
```

Output

Analysis

In order for the *Record of Records!* application to access these new menus, you will also need to make some changes to RECOFREC.C and RECOFREC.H. The following prototypes should be added to RECOFREC.H.

```
/*------------------------------*
 * Prototypes for Reporting     *
 *------------------------------*/

int  do_reporting(void);
```

```
void list_medium_codes(void);
void list_groups(void);
void list_musical_items(void);
void music_rpt(int);
void setup_today(void);          /* in listmed.c */
```

The changes to RECOFREC.C involve changing case 4 in the `main()` function to the following:

```
case 4:  /* Reporting */
         do_reporting();
         break;
```

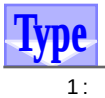
Once you have changed these files, you can recompile. Listing 19.1 has been set up so that comments are printed when the reporting options are selected. You can see this in lines 68, 75, 82, 157, and 164. The actual reports will each be created in its own source file. The call to the functions are included in both Listing 19.1 and the RECOFREC.H file if you made all the changes mentioned. As you create the reports, you will be able to remove the comments from the call and delete the message.

The rest of this listing should not need explanation. The Reporting menu operates like the other menus you have seen. You should notice that this menu was called by the Main menu. In addition, option one in lines 60 to 63 calls a third menu. From the output, you can see that the menus do not completely cover each other. This helps the users see where they are in the menus.

Expert Tip: When presenting more than one menu on the screen at the same time, you should avoid completely covering a lower menu. This helps the users see where they are.

Creating the List of Medium Codes

The list of medium codes will be created in a function called `list_medium_codes()`, which will be created in a file called LISTMED.C. This is presented in Listing 19.2.



Listing 19.2. LISTMED.C. The list of medium codes.

```
1:   /*=========================================================
2:    * Filename: listmed.c
3:    *
4:    * Author:   Bradley L. Jones
5:    *           Gregory L. Guntle
```

```
6:    *
7:    * Purpose:  Lists the information in the medium DB.
8:    *
9:    *=========================================================*/
10:
11:  #include <stdio.h>
12:  #include <string.h>
13:  #include "tyac.h"
14:  #include "records.h"
15:
16:  /*--------------------*
17:   *  Global Variables  *
18:   *--------------------*/
19:  extern FILE *idx_fp;      /* Main File ptr to data file */
20:  extern FILE *db_fp;       /* Data file */
21:  extern nbr_records;       /* Total # of rec for mediums */
22:  extern int start_rec;
23:  extern int disp_rec;
24:
25:
26:  /*------------------*
27:   * Defined constants*
28:   *------------------*/
29:
30:  #define MEDIUM_IDX   "MEDIUMS.IDX"
31:  #define MEDIUM_DBF   "MEDIUMS.DBF"
32:
33:  /*------------------------*
34:   * structure declarations *
35:   *------------------------*/
36:
37:  extern MEDIUM_REC medium;
38:
39:  /*--------------------*
40:   *  Prototypes        *
41:   *--------------------*/
42:
43:  #include "recofrec.h"
44:
45:  void setup_today(void);
46:  void print_med_hdr(void);
47:  int process_med_list(void);
48:
49:  char today[8];
50:
51:  /*=========================================================*
52:   *   list_medium_codes()                                   *
53:   *=========================================================*/
54:
55:  int list_medium_codes(void)
```

continues

Listing 19.2. continued

```
56:  {
57:    int rv=NO_ERROR;
58:
59:    /* Open both Index and DBF file */
60:    if ( (rv = open_files(MEDIUM_IDX, MEDIUM_DBF)) == 0 )
61:    {
62:       /* Are there any records to process ? */
63:       if (nbr_records == 0)
64:       {
65:         display_msg_box("No medium records to process",
66:            ct.err_fcol, ct.err_bcol);
67:       }
68:       else
69:       {
70:         setup_today();
71:         print_med_hdr();
72:         rv = process_med_list();
73:       }              /* End ELSE - records to process */
74:
75:       rv = close_files();
76:    }                 /* End No Errors on Opening Files */
77:    return(rv);
78:  }
79:
80:  /*=======================================================*
81:   *   setup_today()                                       *
82:   *=======================================================*/
83:  void setup_today()
84:  {
85:    int year, month, day;
86:    char hold_date[6];
87:
88:    current_date(&month, &day, &year);
89:    convert_str(hold_date, month, PACK_ZERO);
90:    strncpy(today, hold_date+3, 2);
91:    today[2] = '/';
92:    convert_str(hold_date, day, PACK_ZERO);
93:    strncpy(today+3, hold_date+3, 2);
94:    today[5] = '/';
95:    if (year >1900)
96:       year-=1900;
97:    convert_str(hold_date, year, PACK_ZERO);
98:    strncpy(today+6, hold_date+3, 2);
99:    today[8] = '\0';
100:
101: }
102:
103: /*=======================================================*
```

```
104:  *   print_med_hdr()                                        *
105:  *=========================================================*/
106: void print_med_hdr()
107: {
108:   fprintf(stdprn,"\n\r\t\tList of Medium Codes\n\r");
109:   fprintf(stdprn,"\t\t    %s\n\r\n\r", today);
110:
111:   fprintf(stdprn,"\tCode\tDescription\n\r");
112:   fprintf(stdprn,"\t----\t----------\n\r");
113: }
114:
115: /*=========================================================*
116:  *   process_med_list()                                    *
117:  *=========================================================*/
118: int process_med_list()
119: {
120:   int rv = NO_ERROR;
121:   int done = FALSE;
122:   int srch_rec = start_rec;
123:   MEDIUM_INDEX temp;
124:
125:   while (rv == NO_ERROR && !done)
126:   {
127:     /* Get INDEX */
128:     rv = get_rec(srch_rec, idx_fp, sizeof(temp),
129:          sizeof(int)*2, (char *)&temp);
130:     if (rv == NO_ERROR)
131:     {
132:       /* Get the data record */
133:       rv = get_rec(temp.data, db_fp, sizeof(medium),
134:              0, (char *)&medium);
135:       if (rv == NO_ERROR)
136:       {
137:         /* Print the data */
138:         fprintf(stdprn, "\t%-2s\t%-35s\n\r",
139:             medium.code, medium.desc);
140:         srch_rec = temp.next;
141:         /* Check for end of list */
142:         if (srch_rec == 0)
143:         {
144:            done = TRUE;
145:            fprintf(stdprn, "\f");
146:         }
147:       }        /* End of NO_ERROR - from DBF */
148:     }          /* End of NO_ERROR - from INDEX */
149:   }            /* End WHILE */
150:   return(rv);
151: }
```

Note: The `current_date()` function was taken from Listing 8.2 in Day 8. This function should be added to your TYAC.H library if it isn't already there.

Output

```
List of Medium Codes
            02/20/94

Code    Description
----    ----------
cd      Compact Disc
cs      Cassette Tape
```

Note: Your output will look like the prototype. The specific data printed will be dependent upon the data in your files.

Analysis

This listing starts in the same manner as most of the listings in the *Record of Records!* application. In lines 1 to 9, comments are included giving information on the listing. In lines 11 to 14, the necessary header files are included. The TYAC.H header is included so that your library routines are available. The RECORDS.H header is included for the record layouts that will be needed to hold the medium information. Lines 19 to 23 include the external declarations that you should be very familiar with by now. These are the declarations that are needed to access the databases.

Lines 30 and 31 create the same defined constants that you saw in the Medium Code listing for the entry and edit screen. These constants, `MEDIUM_IDX` and `MEDIUM_DBF`, contain the name of the medium code index file and data file. Line 37 contains an external declaration for a medium code structure. Lines 43 to 47 contain prototypes for the functions included later in the listing.

Expert Tip: The defined constants for the Medium Code files are included in more than one file. Because of this, it would be better to place the constants in a header file. This header file could then be included where needed. In addition, if the file names defined in the constants change, then only one area of the code needs to be modified instead of several.

The first line of unfamiliar code should be line 49. A character array called `today` is declared in line 49. This global array will be used to hold the current date.

Line 55 begins the `list_medium_code()` function, which is the purpose of this listing. Before printing the report, you need to open the file. Line 60 calls the `open_file()` function to open the medium code files. If there wasn't an error, then line 63 checks the global variable, `nbr_records`, that was filled when the file was opened. If there aren't any records in the file, then a message is displayed in line 65; otherwise the report is processed. Line 70 calls a function to set up the current date, line 71 prints the first header on the report, and then line 72 processes the detailed information. Once the processing is done, or if there weren't any records, the files are then closed (line 75). Line 77 ends the function by returning the value in `rv`.

Setting up the current date occurs in lines 80 to 101 in the `setup_today()` function. The `current_date()` function, which was presented on Day 8, is used to get the current month, day, and year.

One other function is used that has not been covered. This is the `convert_str()` function. This function is used to modify a number into a string. The code for this function is presented in Listing 19.3.

Lines 103 to 113 contain the function, `print_med_hdr()`. It is a common practice to create a function that prints only the header to a report. This function can then be called whenever needed by the function that prints the detailed information. Lines 108 to 112 contain several `fprintf()` statements that display information to `stdprn`.

Review Tip: The `fprintf()` function is just like `printf()` except that it has an additional parameter. The first parameter of `fprintf()` is a pointer to where you want the printed information to go. This can be a file such as the medium file or one of the standard I/O streams. The most common standard I/O streams are `stdout`, `stdprn`, and `stderr`. The `stdout` stream is for standard output. This is almost always the screen. The `stdprn` stream is for standard print. This is almost always a printer. The `stderr` stream is for standard errors. This is also almost always the screen.

The processing of the detail information in the report starts on line 118 with the `process_med_list()` function. A `while` loop in lines 125 to 149 will execute as long as all the records are not processed and as long as no errors occur.

Line 128 will read an index record from the index file. This will initially be the starting record. If the read is successful, then the corresponding data record will be read in line 133. If this read is successful, then line 138 prints the detail line containing the medium code and description. Line 140 sets the `srch_rec` to the next sorted record in the index file by using the `next` pointer. If the next index record to be read is zero, then the end of the file has been reached. In this case, line 144 sets the `done` flag to `TRUE` so that the printing loop will end. Additionally, line 145 does a form feed on the report.

Note: This report does not take page size into consideration. It assumes that the number of records that will be printed will not go over one page. In the next listing, this assumption is not made.

The `convert_str()` function was used in Listing 19.2. This function is presented in Listing 19.3. You should add this function to your TYAC.LIB library.

Listing 19.3. The `convert_str()` function.

```
1:   /* ----------------------------------------------------------
2:    * Program: CONVSTR.C
3:    * Authors: Bradley L. Jones
4:    *          Gregory L. Guntle
5:    *
6:    * Purpose: Converts an integer variable to a string
7:    *
8:    * Enter with: rec_disp - char * to place new string
9:    *             rec_nbr  - integer to convert
10:   *             flag  - Whether to front-end it with 0's
11:   *
12:   * Returns: N/A
13:   *==========================================================*/
14:
15:  #include <stdlib.h>  /* Required by itoa */
16:  #include <string.h>
17:  #include <mem.h>
18:
19:  #define PACK_ZERO    1
20:  #define NOPACK       0
21:
22:  void convert_str(char *disp_buff, int nbr, int pack_flag)
23:  {
24:    char hold_convert[18];           /* Required by itoa */
25:
26:    /* Null out area to store new string */
```

```
27:    memset(disp_buff, '\0', sizeof(disp_buff));
28:
29:    /* Convert integer to string */
30:    itoa(nbr, hold_convert, 10);
31:
32:    /* Should the nbr be front-packed with 0's */
33:    if ( pack_flag == PACK_ZERO )    /* Front pack # with 0's ? */
34:    {
35:      memset(disp_buff, '0', 5-strlen(hold_convert));
36:      disp_buff[5-strlen(hold_convert)] = '\0';
37:    }
38:
39:    /* Place convert number in passed parm */
40:    strcat(disp_buff, hold_convert);
41:
42:  }
```

Analysis This function simply converts an integer to a string. Using a defined constant, the string may or may not be packed with zeros. These defined constants, PACK_ZERO and NOPACK, should already be in your TYAC.H header file.

Creating the List of Group Codes

The List of Groups is similar to the List of Medium Codes. The `list_groups()` function is presented in the RPT_GRPS.C file. This is presented in Listing 19.4.

Type **Listing 19.4. RPT_GRPS.C. The List of Groups.**

```
1:   /*============================================================
2:    * Filename: listgrps.c
3:    *
4:    * Author:   Bradley L. Jones
5:    *           Gregory L. Guntle
6:    *
7:    * Purpose:  Lists the information in the groups DB.
8:    *
9:    *============================================================*/
10:
11:  #include <stdio.h>
12:  #include <string.h>
13:  #include "tyac.h"
14:  #include "records.h"
15:
16:  /*--------------------*
17:   *  Global Variables  *
18:   *--------------------*/
```

continues

Listing 19.4. continued

```
19:  extern FILE *idx_fp;      /* Main File ptr to data file */
20:  extern FILE *db_fp;       /* Data file */
21:  extern nbr_records;       /* Total # of rec for mediums */
22:  extern int start_rec;
23:  extern int disp_rec;
24:
25:  int line_ctr;             /* Line ctr */
26:  int page_ctr;             /* Page ctr */
27:
28:  /*------------------*
29:   * Defined constants*
30:   *------------------*/
31:  #define GROUPS_IDX   "GROUPS.IDX"
32:  #define GROUPS_DBF   "GROUPS.DBF"
33:
34:  /*-----------------------*
35:   * structure declarations *
36:   *-----------------------*/
37:  extern GROUP_REC groups;
38:
39:  /*--------------------*
40:   *  Prototypes        *
41:   *--------------------*/
42:  #include "recofrec.h"
43:
44:  int  list_groups(void);
45:  void print_grp_hdr(void);
46:  int  process_groups(void);
47:  void print_group(void);
48:
49:  void setup_date(void);
50:  extern char today[8];
51:
52:  /*=======================================================*
53:   *   list_groups()                                       *
54:   *=======================================================*/
55:
56:  int list_groups(void)
57:  {
58:    int rv=NO_ERROR;
59:
60:    /* Open both Index and DBF file */
61:    if ( (rv = open_files(GROUPS_IDX, GROUPS_DBF)) == 0 )
62:    {
63:       /* Are there any records to process ? */
64:       if (nbr_records == 0)
65:       {
66:         display_msg_box("No groups records to process",
67:            ct.err_fcol, ct.err_bcol);
```

```
68:       }
69:       else
70:       {
71:         setup_today();
72:         print_grp_hdr();
73:         rv = process_groups();
74:       }              /* End ELSE - records to process */
75:
76:       rv = close_files();
77:     }                /* End No Errors on Opening Files */
78:     return(rv);
79:   }
80:
81:   /*=========================================================*
82:    *   print_grp_hdr()                                       *
83:    *=========================================================*/
84:   void print_grp_hdr()
85:   {
86:     fprintf(stdprn,"\n\r\t\tList of Groups\n\r");
87:     fprintf(stdprn,"\t\t   %s\n\r\n\r", today);
88:
89:     fprintf(stdprn,"                                Date\n\r");
90:     fprintf(stdprn,"Group                           ");
91:     fprintf(stdprn,"Formed      Type of Music\n\r");
92:
93:     fprintf(stdprn,"------------------------    ");
94:     fprintf(stdprn,"--------    ------------\n\r");
95:   }
96:
97:   /*=========================================================*
98:    *   process_groups()                                      *
99:    *=========================================================*/
100:  int process_groups()
101:  {
102:    int rv = NO_ERROR;
103:    static int done = FALSE;
104:    static int srch_rec;
105:    static GROUP_INDEX temp;
106:
107:    line_ctr = 0;
108:    page_ctr = 1;
109:
110:    srch_rec = start_rec;
111:    while (rv == NO_ERROR && !done)
112:    {
113:      /* Get INDEX */
114:      rv = get_rec(srch_rec, idx_fp, sizeof(temp),
115:           sizeof(int)*2, (char *)&temp);
116:      if (rv == NO_ERROR)
117:      {
```

continues

Listing 19.4. continued

```
118:       /* Get the data record */
119:       rv = get_rec(temp.data, db_fp, sizeof(groups),
120:              0, (char *)&groups);
121:       if (rv == NO_ERROR)
122:       {
123:         print_group();
124:         srch_rec = temp.next;
125:         /* Check for end of list */
126:         if (srch_rec == 0)
127:         {
128:            done = TRUE;
129:            fprintf(stdprn, "\f");
130:         }
131:         else
132:            done = FALSE;
133:       }        /* End of NO_ERROR - from DBF */
134:     }          /* End of NO_ERROR - from INDEX */
135:   }            /* End WHILE */
136:   return(rv);
137: }
138:
139: /*=======================================================*
140:  *   print_group()                                       *
141:  *=======================================================*/
142: void print_group()
143: {
144:   int i;
145:   char hold_date[8];
146:
147:   if (line_ctr+9 > 55)
148:   {
149:      fprintf(stdprn,"\f");    /* New page */
150:      print_grp_hdr();         /* Reprint header */
151:      fprintf(stdprn,"%-25s",groups.group);
152:      fprintf(stdprn,"    %-8s    %-20s\n\r",
153:              hold_date, groups.music_type);
154:      line_ctr = 6;
155:      page_ctr++;
156:   }
157:   else
158:      line_ctr+=9;
159:
160:   /* Build the date first */
161:   strncpy(hold_date, groups.date_formed.month, 2);
162:   hold_date[2] = '/';
163:   strncpy(hold_date+3, groups.date_formed.day, 2);
164:   hold_date[5] = '/';
165:   strncpy(hold_date+6, groups.date_formed.year, 2);
```

```
166:   hold_date[8] ='\0';
167:
168:   fprintf(stdprn,"%-25s",groups.group);
169:   fprintf(stdprn,"    %-8s    %-20s\n\r",
170:        hold_date, groups.music_type);
171:
172:   fprintf(stdprn,"\n\r\t\t   Members:  %-30s\n\r",
173:     groups.member[0]);
174:
175:   for (i=1; i<=5; i++)
176:   {
177:     fprintf(stdprn,"\t\t             %-30s\n\r",
178:        groups.member[i]);
179:   }
180:
181:   fprintf(stdprn,"\n\r");
182: }
```

Output

```
                List of Groups
                   02/20/94

                                 Date
Group                            Formed      Type of Music
-------------------------     --------    ------------
Violent Femmes                   01/01/87    Alternative

                      Members:   Gordon Gano
                                 Brian Ritchie
                                 Victor DeLorenzo



Yanni                            09/11/93     Instrumental

                      Members:   Yanni
                                 two
                                 three
                                 four
                                 five
                                 six
```

Note: Your output will look like the prototype. The specific data printed will be dependent upon the data in your files.

Analysis This listing should look similar to the Listing 19.2. Because these two listings are so similar, only the differences need an explanation. You should have expected some of the differences. For example, lines 31 and 32 contain defined constants for the group file names instead of the medium files.

The main function, `list_groups()`, in lines 52 to 79 is almost identical to the `list_medium_codes()` function presented earlier. The only difference is that functions related to groups, instead of medium codes, are called.

The List of Groups header information is different from that used by the List of Medium Codes report. Lines 81 to 95 contain the `print_grp_header()` function. This function simply prints several lines of information exactly as it was presented in the prototype.

Processing the group records is also very similar to processing the medium code records. There are two main differences. The first is that the detail information is printed in a separate function called `print_group()`. The information printed for each group is much more complex than what was printed for the medium code. To help keep the code easy to follow, the printing of the detail information was placed in its own function.

The second difference is that the report will perform page breaks. In lines 107 and 108, two additional variables have are used, `line_ctr` and `page_ctr`. In line 147, an `if` statement checks to see if there is room on the page to print another record. If there is not, then line 149 does a form feed using `fprintf()`. Line 150 then calls `print_grp_hdr()` to print the header information on the new page. The `line_ctr` is then reset to six in line 154, and the page counter is incremented by one in line 155.

If there was enough room to print the header, then line 158 increments the line counter by nine. This is the number of lines that are printed for an individual group. With the header check completed, the detail information is formatted and printed. The report cycles through each record until all the records have been printed.

Note: The `line_ctr` was set to six instead of zero because of the header information that was printed.

Note: Most reports contain from 55 to 60 lines on a page.

Creating the Detailed Information Report

The Detailed Information Report will differ from the list reports in several ways. The main difference will be that the Detailed Information Report will use all three databases instead of just one. Listing 19.5 presents the Detailed Information Report.

Listing 19.5. RPT_DETL.C. The detailed Musical Item Report.

```
1:  /*=========================================================
2:   * Filename: listdalb.c
3:   *
4:   * Author:   Bradley L. Jones
5:   *           Gregory L. Guntle
6:   *
7:   * Purpose:  Lists detailed information from the Albums area.
8:   *
9:   * Returns:  0 = No Errors
10:  *          <0 = No records in DB
11:  *          >0 = File I/O Error
12:  *
13:  *=========================================================*/
14:
15: #include <stdio.h>
16: #include <string.h>
17: #include "tyac.h"
18: #include "records.h"
19:
20: /*------------------*
21:  * Global Variables *
22:  *------------------*/
23:
24: extern FILE *idx_fp;      /* Main File ptr to data file */
25: extern FILE *db_fp;       /* Data file */
26: extern FILE *song_fp;     /* Pointer for songs */
27: extern nbr_records;       /* Total # of rec for albums */
28: extern int start_rec;
29: extern int disp_rec;
30:
31: extern ALBUM_REC albums;
32: extern SONG_REC songs[7];
33: extern MEDIUM_REC medium;
34: extern GROUP_REC groups;
35:
36: FILE *alb_idx_fp;         /* Needed when opening multiple DB */
37: FILE *alb_db_fp;
```

continues

Listing 19.5. continued

```
38:  FILE *alb_songs_fp;
39:  FILE *med_idx_fp;
40:  FILE *med_db_fp;
41:  FILE *grp_idx_fp;
42:  FILE *grp_db_fp;
43:
44:  int alb_nbr_recs;
45:  int alb_start_rec;
46:  int grp_nbr_recs;
47:  int grp_start_rec;
48:  int med_nbr_recs;
49:  int med_start_rec;
50:  int med_fnd;
51:  int grp_fnd;
52:
53:  /*------------------*
54:   * Defined constants*
55:   *------------------*/
56:
57:  #define ALBUMS_DBF   "ALBUMS.DBF"
58:  #define ALBUMS_IDX   "ALBUMS.IDX"
59:  #define SONGS_DBF    "SONGS.DBF"
60:
61:  #define GROUPS_IDX   "GROUPS.IDX"
62:  #define GROUPS_DBF   "GROUPS.DBF"
63:
64:  #define MEDIUM_IDX   "MEDIUMS.IDX"
65:  #define MEDIUM_DBF   "MEDIUMS.DBF"
66:
67:
68:  #define ALBUMS       00
69:  #define GROUPS       01
70:  #define MEDIUM       02
71:
72:  #define TO_GLOBAL    00
73:  #define FROM_GLOBAL  01
74:
75:  /*------------------------*
76:   * structure declarations *
77:   *------------------------*/
78:
79:  GROUP_INDEX  grp_idx;
80:  GROUP_REC    grp_dbf;
81:  MEDIUM_INDEX med_idx;
82:  MEDIUM_REC   med_dbf;
83:  ALBUM_INDEX  alb_idx;
84:  ALBUM_REC    alb_dbf;
85:  SONG_REC     alb_songs[7];
86:
```

```
87:  /*--------------------*
88:   *  Prototypes        *
89:   *--------------------*/
90:
91:  #include “recofrec.h”
92:
93:  int  open_all_files(void);
94:  void print_song_hdr(void);
95:  int  process_details(char *);
96:  void print_album_details(void);
97:  void switch_globals(int, int);
98:
99:  void setup_date(void);
100: extern char today[8];
101:
102: int get_info(void);
103:
104: /*=========================================================*
105:  *   list_albums()                                         *
106:  *=========================================================*/
107:
108: int music_rpt(char *title)
109: {
110:   int rv=NO_ERROR;
111:
112:   setup_today();
113:   rv = open_all_files();
114:
115:   if (rv == NO_ERROR)
116:   {
117:     rv = process_details(title);
118:
119:     /* Closing ALBUMS */
120:     switch_globals(ALBUMS, TO_GLOBAL);
121:     rv = close_files();
122:
123:     /* Closing GROUPS */
124:     switch_globals(GROUPS, TO_GLOBAL);
125:     rv = close_files();
126:
127:     /* Closing MEDIUM */
128:     switch_globals(MEDIUM, TO_GLOBAL);
129:     rv = close_files();
130:
131:   }
132:   else
133:   if (rv < 0)
134:   {
135:     display_msg_box(“No albums records to process”,
136:          ct.err_fcol, ct.err_bcol);
```

continues

Listing 19.5. continued

```
137:   }
138:   return(rv);
139: }
140:
141: /*=======================================================*
142:  *   open_all_files()                                     *
143:  *=======================================================*/
144:
145: int open_all_files()
146: {
147:   int rv = NO_ERROR;
148:
149:   /* Open ALBUMS Index and DBF file */
150:   if ( (rv = open_files(ALBUMS_IDX, ALBUMS_DBF)) == 0 )
151:   {
152:     /* Are there any records to process ? */
153:     if (nbr_records == 0)
154:     {
155:       rv = -1;
156:     }
157:     else
158:     {
159:       switch_globals(ALBUMS, FROM_GLOBAL);
160:       if( (alb_songs_fp = fopen(SONGS_DBF, "r+b" )) == NULL )
161:       {
162:          rv = READ_ERROR;
163:       }
164:     }
165:
166:
167:     if (rv == NO_ERROR)
168:     {
169:       /* Now Open GROUPS Index and DBF file */
170:       if ( (rv = open_files(GROUPS_IDX, GROUPS_DBF)) == 0 )
171:       {
172:         switch_globals(GROUPS, FROM_GLOBAL);
173:       }
174:     }
175:
176:
177:     if (rv == NO_ERROR)
178:     {
179:       /* Open MEDIUM Index and DBF file */
180:       if ( (rv = open_files(MEDIUM_IDX, MEDIUM_DBF)) == 0 )
181:       {
182:         switch_globals(MEDIUM, FROM_GLOBAL);
183:       }
184:     }
185:   }
```

```
186:
187:   return(rv);
188: }
189:
190: /*=======================================================*
191:  *   print_song_hdr()                                     *
192:  *=======================================================*/
193: void print_song_hdr()
194: {
195:   fprintf(stdprn,"SONGS:\n\r\n\r");
196:   fprintf(stdprn,"Track    Song Title");
197:   fprintf(stdprn,"                                    Time\n\r");
198:   fprintf(stdprn,"----    “);
199:   fprintf(stdprn,"----------------------------------------");
200:   fprintf(stdprn,"   ----\n\r");
201: }
202:
203: /*=======================================================*
204:  *   process_details()                                   *
205:  *=======================================================*/
206: int process_details(char *title)
207: {
208:   static int rv = NO_ERROR;
209:   static int done = FALSE;
210:   static int srch_rec;
211:
212:   if(strlen(title) == 0)     /* Do all of them */
213:   {
214:     /* Albums Idx will drive report */
215:     srch_rec = alb_start_rec;
216:
217:     while (rv == NO_ERROR && !done)
218:     {
219:       /* GET ALBUM INFORMATION */
220:       rv = get_rec(srch_rec, alb_idx_fp, sizeof(alb_idx),
221:            sizeof(int)*2, (char *)&alb_idx);
222:
223:       if (rv == NO_ERROR)
224:       {
225:         /* Get the Albums data record */
226:         rv = get_rec(alb_idx.data, alb_db_fp, sizeof(alb_dbf),
227:                0, (char *)&alb_dbf);
228:
229:         if (rv == NO_ERROR)
230:         {
231:           rv = get_info();
232:           if (rv == NO_ERROR)
233:           {
234:             srch_rec = alb_idx.next;
235:             print_album_details();
```

continues

Listing 19.5. continued

```
236:             /* Check for end of list */
237:             if (srch_rec == 0)
238:             {
239:               done = TRUE;
240:             }
241:           }      /* End of NO_ERROR - from DBF */
242:         }
243:       }          /* End of NO_ERROR - from INDEX */
244:     }            /* End WHILE */
245:   }
246:   else    /* Do search for specific record */
247:   {
248:     switch_globals(ALBUMS, TO_GLOBAL);
249:
250:     rv = search_alb_rec(title);
251:
252:     if (rv <=0 )  /* No error or Did not find rec */
253:     {
254:       if (rv == NO_ERROR)     /* Found it */
255:       {
256:         /* Transfer into working area */
257:         memcpy(&alb_dbf, &albums, sizeof(albums) );
258:
259:         rv = get_info();
260:
261:         if (rv == NO_ERROR)
262:         {
263:            print_album_details();
264:         }
265:       }
266:       else  /* Must not have found rec */
267:       {
268:         rv = 0;             /* Reset - not really an error */
269:         fprintf(stdprn,"Title:  %s  not found!\f",
270:                 title);
271:       }
272:     }
273:   }
274:   return(rv);
275: }
276:
277: /*=======================================================*
278:  *   get_info()                                          *
279:  *                                                       *
280:  *  Loads all the relevant information from the other    *
281:  *  files.                                               *
282:  *=======================================================*/
283: int get_info()
284: {
```

```
285:   int rv = NO_ERROR;
286:
287:   /* Get Songs */
288:   rv = get_rec(alb_idx.song, alb_songs_fp,
289:             sizeof(alb_songs), 0,
290:             (char *)&alb_songs);
291:
292:   if (rv == NO_ERROR)
293:   {
294:     /* Search for GROUP information */
295:     switch_globals(GROUPS, TO_GLOBAL);
296:
297:     rv = search_grp_rec(alb_dbf.group);
298:
299:     if (rv <=0 )  /* No error or Did not find rec */
300:     {
301:       if (rv == NO_ERROR)
302:       {
303:         /* Transfer into working area */
304:         memcpy(&grp_dbf, &groups, sizeof(groups) );
305:         grp_fnd = TRUE;
306:       }
307:       else  /* Must not have found rec */
308:       {
309:         grp_fnd = FALSE;     /* Set flag */
310:         rv = NO_ERROR;       /* Reset - not really an error */
311:       }
312:     }
313:
314:     /* Search for MEDIUM information */
315:     switch_globals(MEDIUM, TO_GLOBAL);
316:     rv = search_med_rec(alb_dbf.medium_code);
317:
318:     if (rv <=0 )  /* No error or Did not find rec */
319:     {
320:       if (rv == NO_ERROR)
321:       {
322:          /* Transfer into working area */
323:          memcpy(&med_dbf, &medium, sizeof(medium) );
324:          med_fnd = TRUE;
325:       }
326:       else  /* Must not have found rec */
327:       {
328:          med_fnd = FALSE;     /* Set flag */
329:          rv = NO_ERROR;       /* Reset - not really an error */
330:       }
331:     }
332:   }
333:   return(rv);
334: }
```

continues

Listing 19.5. continued

```
335:
336: /*=======================================================*
337:  *   print_album_details()                                *
338:  *=======================================================*/
339: void print_album_details()
340: {
341:   int i;
342:   char hold_items[8];
343:
344: /* ----------------------------------------------*/
345:
346:   /* Display ALBUM TITLE */
347:   fprintf(stdprn,"%-30s",alb_dbf.title);
348:   fprintf(stdprn,"\t\t\t\t\t%8s\n\r\n\r", today);
349:
350: /* ----------------------------------------------*/
351:
352:   /* Display GROUP-related Information */
353:   fprintf(stdprn,"Group:       %-25s\n\r\n\r",
354:        alb_dbf.group);
355:
356:   fprintf(stdprn,"Group Desc:  ");
357:   if (grp_fnd)
358:   {
359:     fprintf(stdprn,"%-60s\n\r",grp_dbf.info[0]);
360:     for (i = 1; i<3; i++)
361:     {
362:       fprintf(stdprn,"             %-60s\n\r",
363:               grp_dbf.info[i]);
364:     }
365:     fprintf(stdprn,"\n\r");
366:
367:   }
368:   else
369:   {
370:     fprintf(stdprn, "Unknown\n\r\n\r\n\r\n\r");
371:   }
372:
373: /* ----------------------------------------------*/
374:
375:   /* Display Types of Music */
376:   fprintf(stdprn, "Type of Music:  ");
377:   if (grp_fnd)
378:   {
379:     fprintf(stdprn,"%-20s\n\r\n\r",grp_dbf.music_type);
380:   }
381:   else
382:   {
383:     fprintf(stdprn,"Unknown\n\r\n\r");
```

```
384:   }
385:
386: /* ---------------------------------------------*/
387:
388:   /* MEDIUM Information */
389:   fprintf(stdprn, “Medium:  “);
390:   if (med_fnd)
391:   {
392:     fprintf(stdprn,”%-35s\n\r\n\r”,med_dbf.desc);
393:   }
394:   else
395:   {
396:     fprintf(stdprn,”Unknown\n\r\n\r”);
397:   }
398:
399: /* ---------------------------------------------*/
400:
401:   /* ALBUM Information */
402:
403:   /* Setup date purchased */
404:   strncpy(hold_items, alb_dbf.date_purch.month, 2);
405:   hold_items[2] = ‘/’;
406:   strncpy(hold_items+3, alb_dbf.date_purch.day, 2);
407:   hold_items[5] = ‘/’;
408:   strncpy(hold_items+6, alb_dbf.date_purch.year, 2);
409:   hold_items[8] =’\0';
410:   fprintf(stdprn,”Date Purchased:  %-8s\n\r\n\r”, hold_items);
411:
412:   /* Setup formatting for COST */
413:   strncpy(hold_items, alb_dbf.cost, 3);
414:   hold_items[3]=’.’;
415:   strncpy(hold_items+4, alb_dbf.cost+3, 2);
416:   hold_items[6] = ‘\0’;
417:   fprintf(stdprn,”Cost:   $%6s\n\r”, hold_items);
418:
419:   /* Setup formatting for VALUE */
420:   strncpy(hold_items, alb_dbf.value, 3);
421:   hold_items[3]=’.’;
422:   strncpy(hold_items+4, alb_dbf.value+3, 2);
423:   hold_items[6] = ‘\0’;
424:   fprintf(stdprn,”Value:  $%6s\n\r\n\r”, hold_items);
425:
426: /* ---------------------------------------------*/
427:
428:   /* SONGS Information */
429:
430:   print_song_hdr();
431:
432:   for (i=0; i<7; i++)
433:   {
```

continues

Listing 19.5. continued

```
434:     fprintf(stdprn," %2d      %-40s   %2s:%2s\n\r",
435:             i+1, alb_songs[i].title,
436:             alb_songs[i].minutes,
437:             alb_songs[i].seconds);
438:   }
439:
440:   fprintf(stdprn,"\n\r");
441:   fprintf(stdprn,"\f");
442: }
443:
444:
445: /*=======================================================*
446:  *   switch_globals()                                    *
447:  *=======================================================*/
448: void switch_globals(int whichone, int whichway)
449: {
450:
451:  switch(whichone)
452:  {
453:    case ALBUMS:
454:                 if (whichway == TO_GLOBAL)
455:                 {
456:                   idx_fp = alb_idx_fp;
457:                   db_fp  = alb_db_fp;
458:                   nbr_records = alb_nbr_recs;
459:                   start_rec   = alb_start_rec;
460:                 }
461:                 else    /* Take FROM_GLOBAL */
462:                 {
463:                   alb_idx_fp = idx_fp;
464:                   alb_db_fp  = db_fp;
465:                   alb_nbr_recs  = nbr_records;
466:                   alb_start_rec = start_rec;
467:                 }
468:                 break;
469:
470:
471:    case GROUPS: if (whichway == TO_GLOBAL)
472:                 {
473:                   idx_fp = grp_idx_fp;
474:                   db_fp  = grp_db_fp;
475:                   nbr_records = grp_nbr_recs;
476:                   start_rec   = grp_start_rec;
477:                 }
478:                 else    /* Take FROM_GLOBAL */
479:                 {
480:                   grp_idx_fp = idx_fp;
481:                   grp_db_fp  = db_fp;
482:                   grp_nbr_recs  = nbr_records;
```

```
483:                   grp_start_rec = start_rec;
484:                 }
485:                 break;
486:
487:    case MEDIUM: if (whichway == TO_GLOBAL)
488:                 {
489:                   idx_fp = med_idx_fp;
490:                   db_fp  = med_db_fp;
491:                   nbr_records = med_nbr_recs;
492:                   start_rec   = med_start_rec;
493:                 }
494:                 else    /* Take FROM_GLOBAL */
495:                 {
496:                   med_idx_fp = idx_fp;
497:                   med_db_fp  = db_fp;
498:                   med_nbr_recs  = nbr_records;
499:                   med_start_rec = start_rec;
500:                 }
501:                 break;
502:  }
503: }
```

Output

```
Walking on the Moon                                    Today: 02/20/94

Group:       Philippe Kahn

Group Desc:  Jazz Music by the Owner of Borland International.
             Features several famous people


Type of Music:  Jazz

Medium:  Compact Disc

Date Purchased: 12/13/92

Cost:   $ 14.95
Value:  $342.00

SONGS:

Track     Song Title                                    Time
----     ----------------------------------------    ----
 01       OOPs!                                        05:06
 02       Interlude                                    00:00
 03       Walkin' on the Moon                          07:28
 04       Interlude                                    00:00
 05       Epistrophy                                   04:35
 06       Interlude                                    00:00
 07       S, E and L                                   04:10
```

```
08      All's Well That Ends                    08:12
09      Interlude                               00:00
10      This Masquerade                         05:48
11      Interlude                               00:00
12      Ralph's Piano Waltz                     05:45
13      Interlude                               00:00
14      Calor                                   08:07
15      Interlude                               00:00
16      Better Days                             03:49
17      Interlude                               00:00
18      Mopti                                   05:46
19      Interlude                               00:00
20      Silence                                 05:52
```

Note: Your output will look like the prototype. The specific data printed will be dependent upon the data in your files.

As you can see, a detailed report has the potential of requiring as much code as an entry and edit screen. The outcome of this listing is a detailed report that contains information out of all three data files in the *Record of Records!* system. With the coding that you have already done in this book, you should be able to follow a majority of this listing on your own. There are just a few areas that may need explanation.

Because the I/O functions used a single set of global variables, there is some complexity in the listing. In fact, a function was necessary to swap the global I/O variables between files. This function, called `switch_globals()`, is presented in lines 445 to 503. Variables are set up to hold each database's global information (see lines 36 to 51). When each of the files is opened, it will call the `switch_globals()` function to copy the generic global values to the file specific areas. As each file is needed, the file-specific variables will be copied back to the global areas.

To call this report, you pass `music_rpt()` a string. If the string is of zero length, then all of the records in the albums database are printed. If you pass a string value, then only that particular title is printed. To give the user the ability to enter a specific string, you need to modify the REC_RPTG.C listing.

Two changes need to be made. The first change is to the `case` statement for printing an individual title with the detail report. This was lines 161 to 166 of Listing 19.1. You should replace this case with the following:

```
case 2:  /* Menu option 2 */
         cursor_on();
```

```
rv = get_alb_selection(alb_ttl);
if( rv == NO_ERROR )
{
  if( strlen(alb_ttl) != 0 )
  {
     music_rpt(alb_ttl);
  }
  else
  {
     display_msg_box("No title entered.",
        ct.err_fcol, ct.err_bcol );
  }
}
break;
```

As you can see, you will call a function called `get_alb_selection()` to determine which title should be printed. The parameter being passed is a character array of 31 characters that needs to be declared at the beginning of the `do_detail_album_menu()` function. If the return value from the `get_alb_selection()` function is not an error, then a string may have been entered. The next line uses `strlen()` to ensure that a string has been entered. If you pass a zero length string to `music_rpt()`, you would get a listing of all the musical items. Because that is a separate menu option, you want to print an error here.

The second change to the REC_RPTG.C listing is the addition of the new `get_alb_selection()` function. Listing 19.6 presents an updated REC_RPTG.C listing with the new changes and function added. You should also notice that the prototype for this function was added to the beginning of the listing.

Type

Listing 19.6. REC_RPTG.C with the `get_alb_selection()` function.

```
1:  /*=========================================================
2:   * Filename: REC_RPTG.c
3:   *           RECORD OF RECORDS - Version 1.0
4:   *
5:   * Author:   Bradley L. Jones
6:   *           Gregory L. Guntle
7:   *
8:   * Purpose:  The Reporting menus.
9:   *
10:  * Note:     Assumes 80 columns by 25 columns for screen.
11:  *=========================================================*/
12:
13: #include <stdio.h>
14: #include <string.h>
15: #include <ctype.h>
```

continues

Listing 19.6. continued

```
#include “tyac.h”
#include “records.h”

/*--------------------*
 *     prototypes     *
 *--------------------*/
#include “recofrec.h”

int do_detail_album_menu(void);
int get_alb_selection(char *);

/*=========================================================*
 *                      main()                             *
 *=========================================================*/

int do_reporting(void)
{
  int  rv       = 0;
  int  cont     = TRUE;
  int  menu_sel = 0;
  char *saved_screen = NULL;


  char *rpt_menu[10] = {
          “1. Detailed Information”, “1Dd”,
          “2. List of Musical Item”, “2Mm”,
          “3. List of Groups      “, “3Gg”,
          “4. List of Medium Codes”, “4Mm”,
          “5. Return to Main Menu “, “5RrEeQq” };

  char MENU_EXIT_KEYS[MAX_KEYS] = {F3, F10, ESC_KEY};

  while( cont == TRUE )                /* loop in temp menu */
  {
    saved_screen = save_screen_area(10, 21, 28, 58 );

    rv = display_menu(12, 30, DOUBLE_BOX, rpt_menu, 10,
                      MENU_EXIT_KEYS, &menu_sel, NO_LR_ARROW,
                      SHADOW);

    switch( rv )
    {
       case ENTER_KEY: /* accept selection */
       case CR:
             switch( menu_sel )
             {
                case 1:  /* Menu option 1 */
                         cursor_on();
                         do_detail_album_menu();
```

```
65:                             break;
66:
67:                     case 2:  /* Menu option 2 */
68:                             cursor_on();
69:                             list_musical_items();
70:                             break;
71:
72:                     case 3:  /* Menu option 3 */
73:                             cursor_on();
74:                             list_groups();
75:                             break;
76:
77:                     case 4:  /* Reporting */
78:                             cursor_on();
79:                             list_medium_codes();
80:                             break;
81:
82:                     case 5:  /* exit */
83:                             cont = FALSE;
84:                             break;
85:
86:                     default: /* continue looping */
87:                             boop();
88:                             break;
89:                 }
90:                 break;
91:
92:         case ESC_KEY: rv = ENTER_KEY;  /* so don't exit clear out */
93:                       cont = FALSE;    /* exit */
94:                       break;
95:
96:         case F3:      /* exiting */
97:                       cont = FALSE;
98:                       break;
99:
100:        case F10:     /* action bar */
101:                      rv = do_main_actionbar();
102:
103:                      if( rv == F3 )
104:                          cont = FALSE;
105:
106:                      break;
107:
108:        default:      boop();
109:                      break;
110:     }
111:   }
112:   restore_screen_area(saved_screen);
113:
114:   return(rv);
```

continues

Listing 19.6. continued

```
115: }
116:
117: /*-----------------------------------------------*
118:  * Detailed Musical Item Report                   *
119:  *-----------------------------------------------*/
120:
121: int do_detail_album_menu(void)
122: {
123:   int  rv       = 0;
124:   int  cont     = TRUE;
125:   int  menu_sel = 0;
126:   char *saved_screen = NULL;
127:   char alb_ttl[31];
128:
129:
130:   char *album_menu[6] = {
131:           "1. All Items", "1Aa",
132:           "2. One Item ", "2Oo",
133:           "3. Return   ", "3RrEeQq" };
134:
135:   char MENU_EXIT_KEYS[MAX_KEYS] = {F3, F10, ESC_KEY};
136:
137:   while( cont == TRUE )                /* loop in temp menu */
138:   {
139:     saved_screen = save_screen_area(13, 19, 35, 55 );
140:
141:     rv = display_menu(14, 40, DOUBLE_BOX, album_menu, 6,
142:                       MENU_EXIT_KEYS, &menu_sel, NO_LR_ARROW,
143:                       SHADOW);
144:
145:     switch( rv )
146:     {
147:        case ENTER_KEY: /* accept selection */
148:        case CR:
149:              switch( menu_sel )
150:              {
151:                 case 1:  /* Menu option 1 */
152:                          cursor_on();
153:                          music_rpt(""); /* empty string for all*/
154:                          break;
155:
156:                 case 2:  /* Menu option 2 */
157:                          cursor_on();
158:                          rv = get_alb_selection(alb_ttl);
159:                          if( rv == NO_ERROR )
160:                          {
161:                            if( strlen(alb_ttl) != 0 )
162:                            {
163:                               music_rpt(alb_ttl);
```

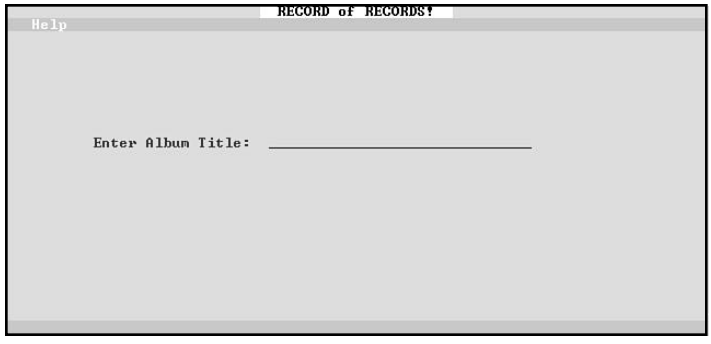
```
164:                            }
165:                            else
166:                            {
167:                               display_msg_box("No title entered.",
168:                                  ct.err_fcol, ct.err_bcol );
169:                            }
170:                         }
171:                         break;
172:
173:                case 3:  /* Exit menu */
174:                         cont = FALSE;
175:                         break;
176:
177:                default: /* continue looping */
178:                         boop();
179:                         break;
180:             }
181:             break;
182:
183:       case ESC_KEY: rv = ENTER_KEY;  /* so don't exit clear out */
184:                     cont = FALSE;    /* exit */
185:                     break;
186:
187:       case F3:      /* exiting */
188:                     cont = FALSE;
189:                     break;
190:
191:       case F10:     /* action bar */
192:                     rv = do_main_actionbar();
193:
194:                     if( rv == F3 )
195:                         cont = FALSE;
196:
197:                     break;
198:
199:       default:      boop();
200:                     break;
201:     }
202:   }
203:   restore_screen_area(saved_screen);
204:
205:   return(rv);
206: }
207:
208: /*-----------------------------------------------*
209:  * Get title for reporting.                      *
210:  *-----------------------------------------------*/
211:
212: int get_alb_selection(char *title)
213: {
```

continues

Listing 19.6. continued

```
214:     int  rv = NO_ERROR;
215:     char *saved_screen = NULL;
216:
217:     static char fexit_keys[ 5 ] = { F3, ESC_KEY, CR_KEY,
218:                                     ENTER_KEY, NULL };
219:     static char *exit_keys = fexit_keys;
220:     getline( SET_EXIT_KEYS, 0, 4, 0, 0, 0, exit_keys );
221:
222:     /*** setup colors and default keys ***/
223:     getline( SET_DEFAULTS, 0, 0, 0, 0, 0, 0 );
224:     getline( SET_NORMAL, 0, ct.fld_fcol, ct.fld_bcol,
225:                 ct.fld_high_fcol, ct.fld_high_bcol, 0 );
226:     getline( SET_UNDERLINE, 0, ct.fld_fcol, ct.fld_bcol,
227:                 ct.fld_high_fcol, ct.fld_high_bcol, 0 );
228:     getline( SET_INS, 0, ct.abar_fcol, ct.abar_bcol, 24, 76, 0 );
229:
230:
231:     saved_screen = save_screen_area(8, 22, 10, 70 );
232:
233:     box(8, 22, 10, 70, BLANK_BOX, FILL_BOX,
234:                        ct.bg_fcol, ct.bg_bcol);
235:
236:     getline( CLEAR_FIELD, 0, 31, 0, 0, 0, title );
237:
238:     write_string("Enter Album Title:",
239:                 ct.fld_prmpt_fcol, ct.fld_prmpt_bcol, 10, 10 );
240:
241:     rv = getline( GET_ALPHA, 0, 10, 30, 0, 30, title);
242:
243:     if( rv == ESC_KEY || rv == F3 )
244:     {
245:        getline( CLEAR_FIELD, 0, 31, 0, 0, 0, title );
246:        rv = 1;
247:     }
248:     else
249:     {
250:        rv = 0;
251:     }
252:
253:     restore_screen_area(saved_screen);
254:     return( rv );
255: }
256:
257: /*=======================================================*
258:  *                    end of listing                     *
259:  *=======================================================*/
```

Output



Analysis

The `get_alb_selection()` function uses the same code that you used to create an entry and edit screen. Lines 217 to 228 set up the `getline()` function. The only exit keys that are used are the Escape key, the Enter or Carriage Return key, and the F3 key. In line 231, the familiar `save_screen_area()` function is used to save off the menus that are displayed on the screen. Line 233 then erases the menus from the screen using the `box()` function to clear with the background colors.

Line 236 begins the process of getting the title. The field that the title will be placed in is cleared. Line 238 displays the prompt to the user. Line 241 calls `getline()` to retrieve that actual data. If the user exits the field with the escape key or the F3 key, then line 245 clears the field and sets the return value to an error. In any other circumstance, the return value is set to zero. Once complete, the screen is restored to the menus and control is returned to the calling function.

Building Flexibility into the Reporting

You have created several reports. In the Detailed Information Report, you used a single report to print either all or a single musical item. Being able to limit the scope of what prints in a report is often a requirement. Many times the scope is limited by using a box similar to the one presented to obtain the musical item title. While this box contained only a single field, it could have contained several.

In addition to limiting the scope of what prints, you may also choose to change whether the report prints to paper or the screen. Based on this choice, you will need to do additional formatting. You will also need to add control for the screen because only 25 lines can be displayed.

Most printers will enable you to format reports with different fonts. These fonts may be as simple as condensed and regular-sized text. In addition, bolding and underlining are also often used in reports. Many of these features require specific information to be sent to the computer. There are a multitude of features that could be added to reports; however, many of them require printer-specific controls.

Summary

Today, you were presented with information on adding printed reports to your applications. Several list reports were presented along with a detailed report. The list reports simply list information from a single file, whereas the detailed report was much more complex. Before beginning the coding on a report, you should create a prototype. A prototype gives you the ability to see what the report will look like before you take the time to code it. It is easier to change a prototype than it is to modify the code.

Q&A

Q How important is reporting in an application?

A Most people prefer to obtain data from a report rather than try to view it from the entry and edit. Most entry and edit screens will only allow one record to be viewed at a time. A report allows multiple records to be seen at the same time.

Q How many reports should be provided?

A You should provide your users with whatever reports are needed. Often, by adding a few selection criteria—such as all or one—you can provide fewer reports and still meet the needs of the users.

Workshop

The Workshop provides quiz questions to help you solidify your understanding of the material covered and exercises to provide you with experience in using what you've learned.

Quiz

1. What is a benefit of prototyping a report?
2. What is a list?
3. What are some of the benefits to listing information from a file?
4. Why are 9s used on a prototype?
5. Why are Xs used on a prototype?
6. Why is it good to include the current date on a report?
7. Why is it good to include “real” data on a report prototype?
8. How many lines can generally print on a page?
9. Using the name data presented below, what would be printed using the following prototype?

   The prototype:

   ```
   Last name    First name    Middle Initial
   XXXXXXX, XXXXX X.
   ```

   The name data:

Last name	*First name*	*Middle name*
Foster	MaryBeth	Elizabeth
Bell	Mike	James
Livenstone	Angela	

Exercises

1. Modify the List of Groups Report. Don’t print lines that contain blank members.
2. Create a prototype for a Musical Item List. This list should contain the title and the group name.
3. Write the code to create the Musical Item List that was prototyped in Exercise 2.
4. **ON YOUR OWN:** Create additional reports and include them in the *Record of Records!* application.
5. **ON YOUR OWN:** Create your own reports for your own application.