15

The User Interface: Action Bars

WEEK
3

On Day 14, you added menus to your application. This helped you produce a front end to your application. In addition, it helped to provide boxes that listed selections instead of requiring data entry. Today, you'll expand on your menus by moving into action bars. Today you will:

- ☐ Learn what action bars are.
- ☐ See how action bars work in conjunction with accelerator keys.
- ☐ See how to add an action bar to your main menu.
- ☐ See how to add an action bar to your entry and edit screens.

What Are Action Bars?

Action bars are menus containing options for various actions that your program can perform. An action bar is generally placed at the top of the screen on the line just below the title. Figure 15.1 presents an action bar menu.

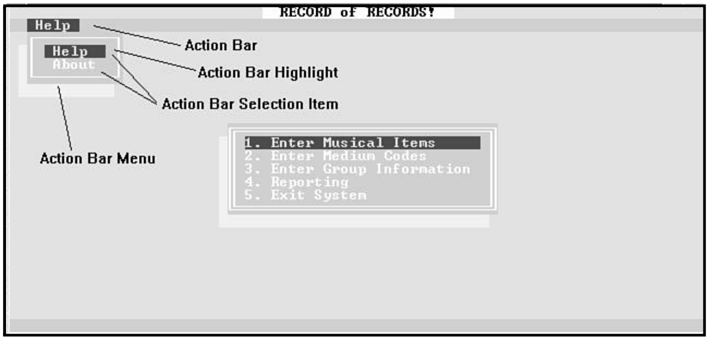


Figure 15.1. *An action bar.*

As you can see, the File action bar item has a single option called Exit. When this option is selected, the exiting routine will be called.

Many action bar items also have accelerator keys. An *accelerator key* is a keyboard key that automatically causes an action to occur. If an action bar option has an accelerator key, it's usually listed in the action bar menu. In Figure 15.1, you can see that the Exit action has an accelerator key of F3.

Standards in Action Bars

Most people choose to follow the same format and naming conventions in their action bars. This provides a consistency for the user between applications. The naming of action bar selections can even cross application types. For instance, a word processor may have an action bar with a selection called File that allows files to be opened, closed, printed, and more. A database program could also have a File action bar item with these same options even though the file types are completely different. The actions that these applications are performing are the same.

There are several action bar items that are prevalent in applications. File, Edit, View, Options, and Help are a few action bar items; however, they aren't the only ones that can be used. For example, a word processor may also include action bar items such as Insert, Format, Tools, and Table. However, keep in mind that File, Edit, View, Options, and Help are common in many different types of applications. Each of these action bar menu items has certain types of actions that should be categorized in their action bar menus. You should create a new action bar item only when an item won't fit into one of these.

The File Action Bar Item

The File option should be used for actions that apply to an entire file. If your application is going to enable the user to choose which file they are working with, then possible options would be those listed in Table 15.1.

Table 15.1. Possible selection items for the File action bar item's menu.

Item	Purpose
New	Open a new file
Open...	Open an existing file
Close	Close the current file, but do not exit
Save	Save the current file
Save as...	Save the current file as a new name
Print	Print the current files information
Exit	Exit the current file

If any of these actions apply to your application, you should use the item name presented in the table. This will help your application be consistent with other applications.

Note: In the *Record of Records!* application, a default file is being used. The only File option that will be used is Exit.

Tip: On an entry and edit screen with an action bar, you should always provide a File action bar option with at least the option to Exit.

The Edit Action Bar Item

The Edit option contains actions that modify the current information that is being presented. The options that are presented here may vary depending on the type of application. In working with a document in a word processor, the Edit options may involve moving textual information around. This would be selection items such as Copy, Cut, and Paste. In a database application, the Edit options may involve manipulations to the current record. This could include Clear, Add, and Delete. Table 15.2 presents many common selection items for an Edit action bar menu.

Table 15.2. Possible selection items for the Edit action bar item's menu.

Item	Purpose
Undo	Counteract the last action
Cut	Remove a portion of the current screen
Copy	Copy a portion of the current screen
Paste	Insert the last Cut or Copied portion of the screen
Clear	Remove a portion or all of the screen
Add	(Database) Add the current information to the database as a new record

Item	Purpose
Update	(Database) Update the corresponding database record with the current screen information
Delete	(Database) Delete the database record that corresponds to the current screen record (Non-database) Remove a portion of the screen

As you can see from Table 15.2, different selection items may be presented based on the type of application; however, the overriding functionality is the same. For example, deleting is always removing something. You should remember that the wording of the options in your action bar menu should correspond to those presented in the table if they are the same actions.

The View Action Bar Item

The View action bar item contains selections that enable you to look in different ways at information from the current file. Again, different applications are going to provide different view options. A word processor may have selection items such as Normal and Page layout. A database application may have items such as Next and Previous.

In addition to options that display data, there may also be options that customize the interface of your application. The View action bar menu may include options as setting the tab stops in a word processing application, customizing the colors used in the screen interface, and setting up accelerator keys. Table 15.3 contains the View options that you will use in the *Record of Records!* application.

Table 15.3. Selection items for the View action bar item's menu for *Record of Records!*

Item	Purpose
Find...	Find a specific record and display
Next	Find the next record and display
Previous	Find the previous record and display

As you can see, each of the items in Table 15.3 presents a different record for viewing. You may be wondering why these options aren't in the File action bar menu because they manipulate information in the file. The reason is because they are working at a record level not a file level.

The Options Action Bar Item

The Options action bar item is used to present selection items that customize the user interface. These are options that change what is or isn't displayed to the user. An example of an Option selection item would be an option to turn the command line at the bottom of the screen on and off. This option doesn't change what the application does, it only changes what the user sees. Actions that customize the use of the application fall under the View option.

An example of an Option action versus a View option is in order:

If there is an option to customize the accelerator keys that are displayed at the bottom of the screen, then it falls in the View options because it is a customization of functionality.

If there is an option to display or not display the accelerator keys, then it falls under Options because this is a customization to what the user sees, not how to do it.

Warning: Most programs confuse the use of View and Option.

The Help Action Bar Item

The Help option contains actions that are informational. Although you may think that Help is as simple as just presenting some information, there is actually more to it. The Help option will generally contain several options. Table 15.4 presents the selection items commonly included on the Help action bar menu.

Table 15.4. Possible selection items for the Help action bar item's menu.

Item	Purpose
Help for help...	Presents information on using help
Extended help...	Presents general information about the current screen

Item	Purpose
Keys help...	Presents information on accelerator keys
Help index...	Presents an index for the help information
Tutorial...	Presents a connection to a tutorial program
About...	Presents general information about the application

Note: Detailed information on each of these selections is presented on Day 16 when the use of Help is covered in detail.

Other Action Bar Items

While the action bar items presented are the most common, they aren't the only options. All applications that have action bars should have a Help option. A File option should also be included in most cases because it can provide a means of exiting the application. Other options may include those that were mentioned earlier or application-specific options.

Expert Tip: Always use the action bar items previously listed instead of making up your own names. This will provide consistency with other applications.

DO **DON'T**

DO follow the standard names presented here when creating your action bars.

DON'T make up your own names for standard functions. This may confuse the user of your application.

A Simple Action Bar in Action

In the *Record of Records!* application that you've been developing, there are several action bars. On the main menu is a simple action bar that contains only a single selection item, Help. In most cases, there will be more than a single item; however, by having only a single item, it makes for a good introductory demonstration.

The *Record of Records!* application has followed the standard of setting the F10 key to access the action bar. When you set up the new main menu on Day 14, the F10 key was included as an exit key. If fact, in line 104 of Listing 14.4, a function call was included in the case for F10. At that time, you were told to comment the line out. The action bar functionality is like any other selected functionality in that once a key is pressed, the action bar processes can be performed separately. The `do_main_actionbar()` function can be created in a separate source file that is linked with the rest of the *Record of Records!* files, or it can be added to the end of the RECOFREC.C listing.

Listing 15.1 presents the main menu's action bar function in a separate listing. In addition to incorporating this listing, you need to also make a few other subtle changes. First, you need to uncomment the action bar lines in the RECORREC.C listing (approximately lines 104 to 107). You should also include a `do_main_actionbar()` prototype in the RECOFREC.H header file.

Type

Listing 15.1. MMNUABAR.C. The main menu's action bar functions.

```
/*==========================================================
 * Filename: mmnuabar.c
 *           RECORD OF RECORDS - Version 1.0
 *
 * Author:   Bradley L. Jones
 *
 * Purpose:  Action bar for main menu.  This is a single
 *           menu. Functions for selections will need to
 *           be added later.
 *
 * Return:   Return value is either key used to exit menu
 *           with the exception of F10 which is returned
 *           as Enter to re-display main menu.
 *==========================================================*/

#include <stdio.h>
#include “tyac.h”
#include “records.h”

```

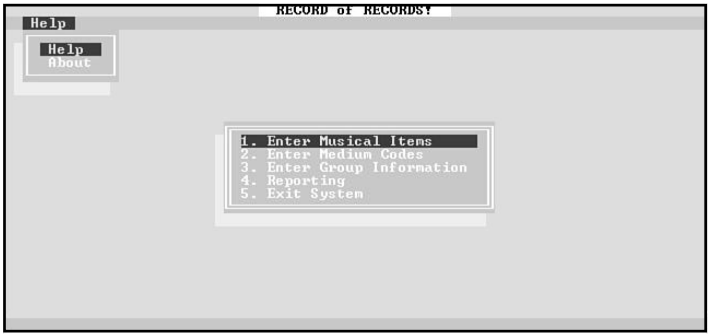
```
/*--------------------*
 *     prototypes     *
 *--------------------*/
#include “recofrec.h”

static void do_something(char *msg);

/*--------------------------------------------*/

int do_main_actionbar( void )
{
  int   rv       = 0;
  int   menu_sel = 0;

  char *help_menu[4] = {
          “ Help  “, “1Hh”,
          “ About “, “2Aa” };

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

  write_string( “ Help “,
                ct.menu_high_fcol, ct.menu_high_bcol, 1, 2);
  rv = display_menu( 3, 4, SINGLE_BOX, help_menu, 4,
                    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 */
                               do_something(
                                  “Main Menu Help...”);

                               break;

                      case 2:  /* Menu option 2 */
                               do_something(
                                  “An about box...”);

                               break;

                      default: /* continue looping */
                               boop();
                               break;
                    }

```

continues

Listing 15.1. continued

```
                    rv = ENTER_KEY;
                    break;

      case F3:      /* exiting */
      case ESC_KEY:
                    break;

      case F10:     /* action bar */
                    rv = ENTER_KEY;
                    break;

      default:      boop();
                    break;
    }
  cursor_on();
  return(rv);
}

/*------------------------------------------------------*
 *   Generic function - temporary                       *
 *------------------------------------------------------*/
static void do_something(char *msg)
{
  display_msg_box( msg, ct.help_fcol, ct.help_bcol );
}
```

Output



Analysis

This code should look similar to the code in the menus that you created on Day 14. In essence, an action bar with only one item is simply a menu that is displayed near the top of the screen.

Because the action bar is being created before all the actions are developed, a generic function will be used to fill the holes. For example, Help is not covered until Day 16.

Instead of calling a function to do help, a generic function called `do_something()` has been created. The `do_something()` function is prototyped in line 31 as a `static void` function. Because the `static` modifier is used, only this listing will be able to use the `do_something()` function that is included in lines 87 to 93. Other listings can have their own versions of the `do_something()` function. In this listing, the `do_something()` function simply displays a message that it receives as a parameter.

Tip: While not done in this book, if a function isn't going to be used by any source files other than the current one, you should add the `static` modifier. This prevents any other source files from accessing the function.

The main function begins in line 29. A variable is declared for the return value and also for the action bar menu selection. In lines 34 to 44, you should notice that a menu is being created in almost the same manner as those shown on Day 14. The one exception is in line 40. The action bar word, " `Help` " is highlighted by using the `write_string()` function and the highlight colors.

Line 46 reacts to the returned value from the menu. The returned value should be one of the exit keys or Enter. Because this is a single menu action bar, the left and right arrows are turned off. The `switch` statement in line 46 causes the appropriate action to occur. If the Enter key was pressed, then the user made a selection. A second `switch` statement routes the processing based on which action bar item the user selected (lines 48 to 70). In lines 53 and 59, the `do_something()` function is called. When you learn about adding help to your application on Day 16, you'll want to replace these function calls with calls to appropriate routines.

If F3 or Esc was pressed, you want to pass the key's value back to the main menu. When the main menu's F10 case receives the values back, you'll want to check for the F3 key. It is assumed that if the F3 key is pressed on the main menu's action bar, the user is ready to exit the program. Following is the F10 case in the main menu (RECOFREC.C):

```
case F10:       /* action bar */
                rv = do_main_actionbar();

                if( rv == F3 )
                    cont = FALSE;

                break;
```

The F10 case, in lines 76 of Listing 15.1, is also unique. If the user presses F10 on the action bar, then line 77 changes it to the Enter key. Because of this translation, there will be no need to worry about the main menu reacting to the F10 key, which it originally used to display the action bar. The final result of this is that the F10 key will toggle between the main menu and the action bar.

The last two lines of the function are straightforward. The cursor is turned back on with the `cursor_on()` function because the previous menu turned it off. Because control is going to another menu, it isn't necessary to turn the cursor back on; however, not all action bars return to menus. Line 84 returns the value in `rv`, which is either the exit key from the action bar or the Enter key.

Note: The functions called by the action bar will be the same functions that could be accessed via accelerator keys. For example, selecting Exit off of the main menu's File action bar is the same as pressing F3. To ensure that the functionality is the same, you should place it in a function of its own. This function should then be called from both the F3 case in the main menu and the F3 case in the File action bar. This becomes more important in the entry and edit screen action bars.

A Multi-Menu Action Bar

The action bar presented in Listing 15.1 is an exception in that only one action bar option existed. Most action bars will have at least two options. If you have two items, the functionality becomes a little more complicated; however, the functionality is still not too complex.

Two major differences exist in a multi-option action bar from that of the single-option action bar already presented. The first difference is in the use of the left and right arrows. In a multi-option action bar, the left and right arrows should move the cursor from one action bar menu to the next. Consider an action bar with a File option followed by an Edit option. If the File option menu is currently displayed and the user presses the right arrow key, then the File option menu should be closed and the Edit options menu opened. Each press of the right arrow key should move one action bar menu to the right. If you are on the last menu, then you should circle back around to the first menu. The left key should do the same, except it should move one menu to the left for each press.

The second change from the single menu should already be obvious from the discussion on the right and left arrow keys. With a multi-option action bar, you need to keep track of which Action bar option is current. Listing 15.3 presents the code for the action bar in the Medium Codes screen of the *Record of Records!* application. Because most of the functions that the action bar calls have not yet been developed, a generic `do_something()` function has been used. The days following will begin to fill in these functions with Help routines, File routines, and more.

Before entering Listing 15.3, you need to make a few minor changes to some of your other listings. You should include the prototype for the Medium Code screen's action bar function in the RECOFREC.H header file. Listing 15.2 contains a RECOFREC.H header file with prototypes for all three entry and edit screens action bars.

Listing 15.2. RECOFREC.H with the action bar function prototypes.

```
/*==========================================================
 * Filename: RECofREC.H
 *
 * Author:   Bradley L. Jones & Gregory L. Guntle
 *
 * Purpose:  Header file for RECORD of RECORDS! application
 *           This contains the function prototypes needed
 *           by more than one source file.
 *=========================================================*/

#ifndef __RECOFREC_H
#define __RECOFREC_H

/*----------------------------*
 * Prototypes from recofrec.c  *
 *----------------------------*/

void draw_borders(char *);

int  do_medium_screen(void);
int  do_albums_screen(void);
int  do_groups_screen(void);
void display_msg_box( char *, int, int );
char yes_no_box( char *, int, int );
int  zero_fill_field( char *, int );

int  do_main_actionbar(void);


```

continues

Listing 15.2. continued

```
/*------------------------------*
 * Prototypes for medium screen *
 *-------------------------------*/

int  do_medium_actionbar(void);
void display_medium_help(void);


/*------------------------------*
 * Prototypes for groups screen *
 *-------------------------------*/

int  do_groups_actionbar(void);
void display_groups_help(void);

/*------------------------------*
 * Prototypes for albums screen *
 *-------------------------------*/

int  do_albums_actionbar(void);
void display_albums_help(void);


#endif
/*========================================================*
 *                    end of header                       *
 *=========================================================*/
```

As you can see, the action bar function for the Medium Code screen will be called `do_medium_actionbar()`. A call to this function needs to be added to the Medium Code screen's listing, MEDIUM.C. You also need to add F10 as an exit key in the `get_medium_input_data()` function. Following is the new set-up lines for `getline()` followed by the new F10 case that should be added to MEDIUM.C:

```
/*  Set up exit keys.  */
static char fexit_keys[ 14 ] = { F1, F3, F4, F10,
                 ESC_KEY, PAGE_DN, PAGE_UP, CR_KEY,
                 TAB_KEY, ENTER_KEY, SHIFT_TAB,
                 DN_ARROW, UP_ARROW, NULL };

static char *exit_keys = fexit_keys;
getline( SET_EXIT_KEYS, 0, 13, 0, 0, 0, exit_keys );
```

The new F10 case:

```
 case F10:        /* action bar */
                  rv = do_medium_actionbar();
```

```
if( rv == F3 )
{
   if( (yes_no_box( “Do you want to exit?”,
         ct.db_fcol, ct.db_bcol )) == ‘Y’ )
   {
       loop = FALSE;
   }
}

position = 0;
break;
```

One other modification also needs to be made. In the function that draws the screen, `draw_medium_screen()`, the addition of drawing the action bar options needs to be made. This is a simple call to `write_string` as follows:

```
write_string( “ File   Edit   Search   Help”,
      ct.abar_fcol, ct.abar_bcol, 1, 2);
```

Once these modifications have been made, you’re ready to create the Medium Code screen’s action bar. Listing 15.3 contains all you need. It is followed by the Screen prints of each of the four menus.

Listing 15.3. MEDMABAR.C. The action bar for the Medium Code screen.

```
/*==========================================================
 * Filename: medmabar.c
 *          RECORD OF RECORDS - Version 1.0
 *
 * Author:   Bradley L. Jones
 *
 * Purpose:  Action bar for medium screen.  This will
 *           contain multiple menus. The functions called
 *           by the menu selections may not be available
 *           until later days.
 *
 * Return:   Return value is either key used to exit a menu
 *           with the exception of F10 which is returned
 *           as Enter to re-display main menu.  In each menu
 *           the left and right keys will exit and move
 *           control to the right or left menu.
 *==========================================================*/


#include <stdio.h>

```

continues

Listing 15.3. continued

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

/*--------------------*
 *     prototypes     *
 *--------------------*/

#include “recofrec.h”

int do_medium_menu1( void );
int do_medium_menu2( void );
int do_medium_menu3( void );
int do_medium_menu4( void );

static void do_something( char * );

/*----------------------------*
 *  medium screen action bar  *
 *----------------------------*/

int do_medium_actionbar( void )
{
  int  menu  = 1;
  int  cont  = TRUE;
  int  rv    = 0;
  char *abar_text ={“ File   Edit   Search   Help “};


  while( cont == TRUE )
  {
     write_string(abar_text, ct.abar_fcol, ct.abar_bcol, 1, 2);

     switch( menu )
     {

        case 1:  /* file menu */
                 write_string( “ File “, ct.menu_high_fcol,
                      ct.menu_high_bcol, 1, 2);

                 rv = do_medium_menu1();
                 break;

        case 2:  /* edit menu */
                 write_string( “ Edit “, ct.menu_high_fcol,
                      ct.menu_high_bcol, 1, 9);

                 rv = do_medium_menu2();
                 break;

        case 3:  /* search menu */
                 write_string( “ Search “, ct.menu_high_fcol,
                      ct.menu_high_bcol, 1, 16);

                 rv = do_medium_menu3();
                 break;

        case 4:  /* Help menu */
                 write_string( “ Help “, ct.menu_high_fcol,
                      ct.menu_high_bcol, 1, 25);

                 rv = do_medium_menu4();
                 break;

        default: /* error */
                 cont = FALSE;
                 break;
     }

     switch( rv )
     {
        case LT_ARROW:  menu--;
                        if( menu < 1 )
                            menu = 4;
                        break;

        case RT_ARROW:  menu++;
                        if( menu > 4 )
                            menu = 1;
                        break;

        default:        cont = FALSE;
                        break;
     }
  }
  write_string(abar_text, ct.abar_fcol, ct.abar_bcol, 1, 2);
  cursor_on();
  return(rv);
}


/*--------------------*
 *  do_menu 1  (File)  *
 *--------------------*/

int do_medium_menu1( void )
{
  int   rv       = 0;
  int   menu_sel = 0;
```

continues

Listing 15.3. continued

```
  char *saved_screen = NULL;

  char *file_menu[2] = { “ Exit  <F3> “, “1Ee” };
  char exit_keys[MAX_KEYS] = {F3, F10, ESC_KEY};

  saved_screen = save_screen_area( 0, 10, 0, 40 );

  rv = display_menu( 3, 4, SINGLE_BOX, file_menu, 2,
                    exit_keys, &menu_sel, LR_ARROW, SHADOW);

  switch( rv )
  {
     case ENTER_KEY: /* accept selection */
     case CR:
                     rv = F3;
                     break;

     case F3:       /* exiting */
     case ESC_KEY:
     case LT_ARROW: /* arrow keys */
     case RT_ARROW:
                     break;

     case F10:      /* exit action bar */
                     rv = ENTER_KEY;
                     break;

     default:        boop();
                     break;
  }
  restore_screen_area( saved_screen );

  return(rv);
}

/*--------------------*
 *  do_menu 2  (Edit)  *
 *--------------------*/

int do_medium_menu2( void )
{
  int   rv       = 0;
  int   menu_sel = 0;
  char *saved_screen = NULL;

  char *edit_menu[8] = {
          “ New          “, “1Nn”,
          “ Add     <F4> “, “2Aa”,
          “ Change  <F5> “, “3Cc”,
        “ Delete  <F6> “, “4Dd” };

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

saved_screen = save_screen_area( 1, 10, 8, 40 );

rv = display_menu( 3, 11, SINGLE_BOX, edit_menu, 8,
                   exit_keys, &menu_sel, LR_ARROW, SHADOW);

switch( rv )
{
   case ENTER_KEY: /* accept selection */
   case CR:
                   switch( menu_sel )
                   {
                     case 1:   /* Clear the screen */
                               do_something( “CLEARING...”);
                               break;


                     case 2:   /* Add a record */
                               do_something( “Adding...”);

                               break;

                     case 3:   /* Update the current record */
                               do_something( “Updating...”);

                               break;

                     case 4:   /* Deleting the current record */
                               do_something( “Deleting...”);

                               break;

                     default:  /* continue looping */
                               boop();
                               break;
                   }

                   rv = ENTER_KEY;
                   break;

    case F3:       /* exiting */
    case ESC_KEY:
    case LT_ARROW: /* arrow keys */
    case RT_ARROW:
                   break;

```

continues

Listing 15.3. continued

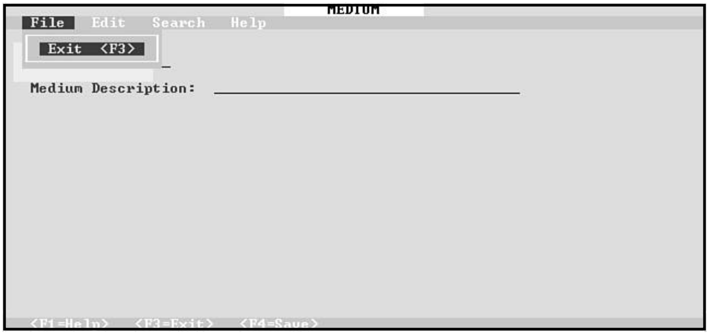
```
      case F10:      /* action bar */
                     rv = ENTER_KEY;
                     break;

      default:       boop();
                     break;
  }
  restore_screen_area( saved_screen );

  return(rv);
}

/*----------------------*
 *  do menu 3  (Search)  *
 *----------------------*/

int do_medium_menu3( void )
{
  int   rv       = 0;
  int   menu_sel = 0;
  char *saved_screen = NULL;

  char *search_menu[6] = {
          “ Find...        “, “1Ff”,
          “ Next      <F7> “, “2Nn”,
          “ Previous  <F8> “, “3Pp” };

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

  saved_screen = save_screen_area( 1, 10, 0, 60 );

  rv = display_menu( 3, 18, SINGLE_BOX, search_menu, 6,
                     exit_keys, &menu_sel, LR_ARROW, SHADOW);

  switch( rv )
  {
     case ENTER_KEY: /* accept selection */
     case CR:
                     switch( menu_sel )
                     {
                       case 1:  /* Do find dialog */
                                do_something( “Find...”);

                                break;

                       case 2:  /* Next Record */
                                do_something( “Next...”);

                                break;

                       case 3:  /* Previous Record */
                                do_something( “Previous...”);

                                break;

                       default: /* shouldn’t happen */
                                boop();
                                break;
                     }

                     rv = ENTER_KEY;
                     break;

      case F3:       /* exiting */
      case ESC_KEY:
      case LT_ARROW: /* arrow keys */
      case RT_ARROW:
                     break;

      case F10:      /* action bar */
                     rv = ENTER_KEY;
                     break;

      default:       boop();
                     break;
  }
  restore_screen_area( saved_screen );

  return(rv);
}

/*--------------------*
 *  do menu 4  (Help)  *
 *--------------------*/

int do_medium_menu4( void )
{
  int    rv       = 0;
  int    menu_sel = 0;
  char *saved_screen = NULL;

  char *help_menu[4] = {
          “ Help  <F2> “, “1Hh”,
          “ About      “, “2Ee” };

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

```

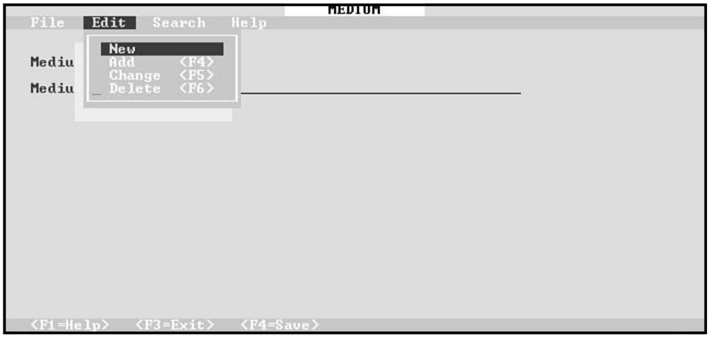
continues

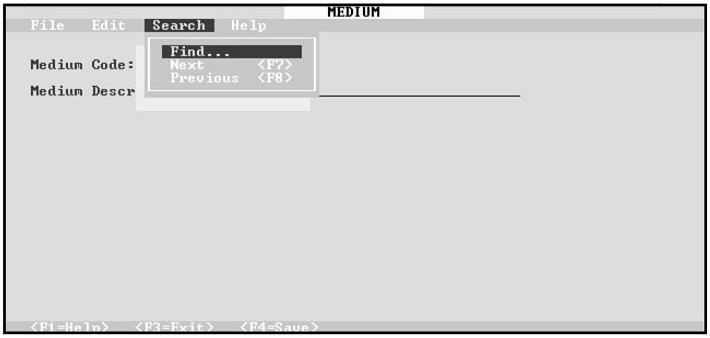
Listing 15.3. continued

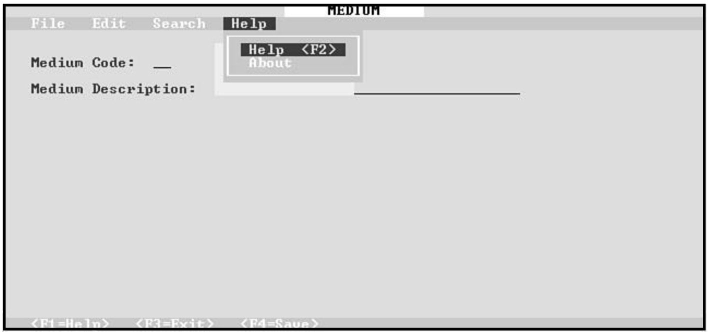
```
  saved_screen = save_screen_area( 1, 10, 0, 60 );

  rv = display_menu( 3, 27, SINGLE_BOX, help_menu, 4,
                     exit_keys, &menu_sel, LR_ARROW, SHADOW);

  switch( rv )
  {
     case ENTER_KEY: /* accept selection */
     case CR:
                     switch( menu_sel )
                     {
                       case 1:  /* Extended Help */
                                display_medium_help();

                                break;

                       case 2:  /* About box */
                                do_something( “About box...”);

                                break;

                       default: /* continue looping */
                                boop();
                                break;
                     }

                     break;

      case F3:       /* exiting */
      case ESC_KEY:
      case LT_ARROW: /* arrow keys */
      case RT_ARROW:
                     break;

      case F10:      /* action bar */
                     rv = ENTER_KEY;
                     break;

      default:       boop();
                     break;
  }
  restore_screen_area( saved_screen );

  return(rv);
}


/*------------------------------------------------------*
 *   Generic function - temporary                       *
 *-------------------------------------------------------*/
static void do_something(char *msg)
{
  display_msg_box( msg, ct.help_fcol, ct.help_bcol );
}
```

Output









Medium Code: __

Medium Description: ____________________

Analysis While this is a long listing, don't be intimidated by it. Most of it is identical to Listing 15.1. You'll see this as the listing is explained.

Listing 15.3 starts out like most other listings. Lines 20 to 23 include the appropriate headers. Line 29 includes the RECOFREC.H header file, which contains the prototypes for the `do_medium_actionbar()` function (see Listing 15.2). Lines 31 to 34 contain four additional prototypes. These prototypes are for each of the action bar options for the Medium Code screen. While I choose to use numbers, it may be better to use descriptive names. For example, `do_medium_menu1()` will present the File action bar menu. This could be named `do_medium_abar_file()`. You can choose whatever name you feel most comfortable with.

Line 36 contains a prototype for the `do_something()` function. Again, the `static` modifier was used so that this source file's function would be separate from all the other source file's `do_something()` functions. When you complete the functionality of the action bar on later days, you will want to remove the `do_something()` function.

Line 42 begins the `do_medium_actionbar()` function. Four variables are set up to be used with the action bar. The first is `menu`. This variable keeps track of which menu is currently being used. Figure 15.2 presents how the numbers relate to the Medium Code screen's action bar.

As you can see, File is 1, Edit is 2, Search is 3, and Help is 4. Because the File menu is highlighted when you first use the action bar, the default value for `menu` is 1.

The remaining variables are easier to understand. A flag, `cont`, is needed to know when to exit the action bar. The variable `rv` is declared to hold a return value. The last variable is a string called `abar_text`. Because the text for the action bar will be written several times, it has been consolidated into a single string.

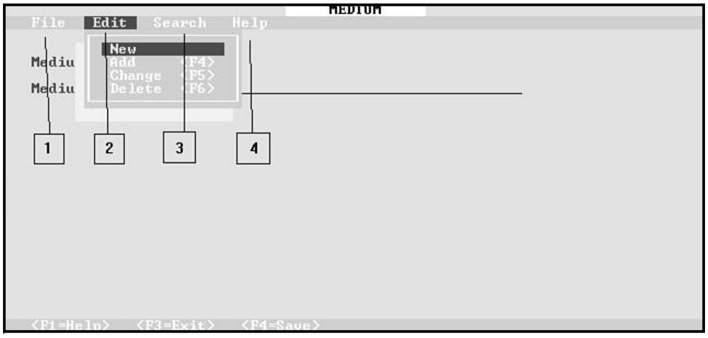


Figure 15.2. *The value of* `menu` *for the Medium Code Screen's action bar.*

Lines 50 to 105 contain a `while` statement that keeps processing the action bar as long as the flag, `cont`, is `TRUE`. The first step is to redraw the action bar across the top of the screen. The first time into the action bar this is redundant; however, each iteration of the `while` following needs the action bar redrawn to overwrite the previous highlighted option. A `switch` beginning in line 54 routes the processing to the current menu. This will be a value from 1 to 4. As you can see by the cases, the processing for each menu is nearly identical. The first step is to highlight the corresponding action bar item by rewriting it in the highlight colors. For menu 1, File is rewritten, for menu 2, Edit, for menu 3, Search, and for menu 4, Help.

Once the action bar item is highlighted, then the corresponding action bar selection items menu is displayed. Each action bar items menu is in a separate function because of its customized options. Each of these functions is formatted in the same manner as the menu in Listing 15.1. The only major difference is that the left and right arrows are enabled. If either of these arrow keys is pressed, they are returned to the calling case in lines 57 to 83.

Each menu handles the selection from the user. When completed, control returns to the `do_medium_actionbar()` case. A `switch` statement, in lines 90 to 104, evaluates the exit key that was used by each menu. If the left arrow is used, then the current menu is decremented (line 92). If the previous menu was 1, then the last menu will be made current. The right arrow key works the same way except that, instead of decrementing the current menu, it is incremented. If the last menu had been current, then the first menu is made current. With the current menu reset, the next iteration of the `while` loop is called.

If any other key had been used to exit the menu, then processing is done. The continue flag, `cont`, is set to `FALSE` so that the `while` loop will end. Before returning to the calling program, the action bar is redrawn without any highlights and the cursor is turned on. Once completed, the last key used is returned to the calling function in MEDIUMS.C.

Expert Tip: In *Record of Records!*, after an action bar selection item is executed, control is returned to the main screen. You could return control to the action bar by removing the default case in lines 85 to 87—or at least line 86.

The rest of this listing contains the four action bar menus, each in its own function. Each of these menus is similar to Listing 15.1 presented earlier with the exceptions already noted.

Summary

Today's material expanded upon yesterday's. Once you understand menus, you're ready to add action bars to your applications. An action bar is the menu across the top of a screen that contains several menus with actions that can be performed. Today, you were presented with several standard action bar options and the standard selection items that fall within them. Common action bar options include File, Edit, View, Options, and Help. When naming your action bar items, you should try to use standard names rather than make up your own. After covering the action bar naming standards, examples of action bars were created. First, a single menu action bar was developed. This was followed by an action bar for the Medium Code screen.

Q&A

Q Is it okay to make up your own names for action bar items?

A While many people believe that making up their own names for action bar items is better than using the standard names, this isn't really true. You may believe that you are creating names that are more descriptive, and you may be correct. However, by using names consistent with most other applications, your users will be more comfortable with the application. In addition, they may actually have a better indication of the action than your seemingly more descriptive name.

Q Are all the standard action bar names presented in today's tables?

A Absolutely not. There are a multitude of standard names. I would suggest looking at several other applications to get an understanding of many of the standard action bar item names. In addition, you can contact companies such as Microsoft Corporation and IBM in regard to standards that they suggest.

Q Does the order of action bar items matter?

A It's best to follow an order when presenting action bar items. The order that the items were presented in this chapter is the standard order. This is File, Edit, View, Options, and then Help. If you need to add additional items, they should be placed between Options and Help. The Help option should always be on the far right.

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 an action bar?
2. What are the most common action bar items?
3. What types of selections should be allowed in a File action bar menu?
4. What types of selections should be allowed in an Edit action bar menu?
5. What types of selections should be allowed in a View action bar menu?
6. What types of selections should be allowed in an Options action bar menu?
7. What types of selections should be allowed in a Help action bar menu?
8. Is a Help option necessary?
9. What do the left and right arrows do on an action bar?
10. Why are action bars called action bars rather than menu bars?

Exercises

1. **ON YOUR OWN:** Review several commercial applications to see what names they use in their action bars. If possible, look at different operating environments or systems also. Many of the action bar names cross over from DOS, OS/2, and Windows applications.
2. Create an action bar function for the Group Information screen.
3. **ON YOUR OWN:** Create an action bar function for the Musical Items screens.