The Mailing List Application

This appendix shows a complete program that contains most the commands and functions you learned in this book. This program manages a mailing list for your personal or business needs.

When you run the program, you are presented with a menu of choices that guides you through the program's operation. Comments throughout the program offer improvements you might want to make. As your knowledge and practice of C++ improve, you might want to expand this mailing list application into a complete database of contacts and relatives.

Here is the listing of the complete program:

```
// Filename: MAILING.CPP
// * Mailing List Application *
//   ------------------------
//
//   This program enables the user to enter, edit, maintain, and
//   print a mailing list of names and addresses.
//
//   All commands and concepts included in this program are
//   explained throughout the text of C++ By Example.
```

```
//
//
//
// These are items you might want to add or change:
// 1. Find your compiler's clear screen function to
//    improve upon the screen-clearing function.
// 2. Add an entry for the 'code' member to track different
//    types of names and addresses (i.e., business codes,
//    personal codes, etc.)
// 3. Search for a partial name (i.e., typing "Sm" finds
//    "Smith" and "Smitty" and "Smythe" in the file).
// 4. When searching for name matches, ignore case (i.e.,
//    typing "smith" finds "Smith" in the file).
// 5. Print mailing labels on your printer.
// 6. Allow for sorting a listing of names and address by name
//    or ZIP code.


// Header files used by the program:

#include <conio.h>
#include <ctype.h>
#include <fstream.h>
#include <iostream.h>
#include <string.h>

const char FILENAME[] = "ADDRESS.DAT";

// Prototype all of this program's functions.

char get_answer(void);
void disp_menu (void);
void clear_sc (void);
void change_na (void);
void print_na (void);
void err_msg (char err_msg[ ]);
void pause_sc (void);

const int NAME_SIZE = 25;
const int ADDRESS_SIZE = 25;
const int CITY_SIZE = 12;
```

```
const int STATE_SIZE = 3;
const int ZIPCODE_SIZE = 6;
const int CODE_SIZE = 7;

// Class of a name and address
class     Mail
{
private:
   char name[NAME_SIZE]; // Name stored here, should
                         // be Last, First order
   char address[ADDRESS_SIZE];
   char city[CITY_SIZE];
   char state[STATE_SIZE];    // Save room for null zero.
   char zipcode[ZIPCODE_SIZE];
   char code[CODE_SIZE];   // For additional expansion.  You
                           // might want to use this member
                           // for customer codes, vendor codes,
                           // or holiday card codes.
public:
   void pr_data(Mail *item)
   {
   // Prints the name and address sent to it.
      cout << "\nName    : " << (*item).name << "\n";
      cout << "Address: " << (*item).address << "\n";
      cout << "City    : " << (*item).city << "\tState: "
           << (*item).state << "  Zipcode: " << (*item).zipcode
           << "\n";
   }

   void get_new_item(Mail *item)
   {
      Mail temp_item;   // Holds temporary changed input.

      cout << "\nEnter new name and address information below\n(Press the ";
      cout << "Enter key without typing data to retain old "
              "information)\n\n";
      cout << "What is the new name? ";
      cin.getline(temp_item.name, NAME_SIZE);
      if (strlen(temp_item.name)) // Only save new data if user
      { strcpy((*item).name, temp_item.name); } // types something.
      cout << "What is the address? ";
      cin.getline(temp_item.address, ADDRESS_SIZE);
```

```
        if (strlen(temp_item.address))
        {  strcpy((*item).address, temp_item.address); }
        cout << "What is the city? ";
        cin.getline(temp_item.city, CITY_SIZE);
        if (strlen(temp_item.city))
        {  strcpy((*item).city, temp_item.city); }
        cout << "What is the state? (2 letter abbreviation only) ";
        cin.getline(temp_item.state, STATE_SIZE);
        if (strlen(temp_item.state))
        { strcpy((*item).state, temp_item.state); }
        cout << "What is the ZIP code? ";
        cin.getline(temp_item.zipcode, ZIPCODE_SIZE);
        if (strlen(temp_item.zipcode))
        {  strcpy((*item).zipcode, temp_item.zipcode); }
        (*item).code[0] = 0;    // Null out the code member
                                // (unused here).
    }

    void add_to_file(Mail *item);
    void change_na(void);
    void enter_na(Mail *item);
    void getzip(Mail *item);
};

void Mail::change_na(void)
{
// This search function can be improved by using the
// code member to assign a unique code to each person on the
// list.  Names are difficult to search for since there are
// so many variations (such as Mc and Mac and St. and Saint).

    Mail item;
    fstream file;
    int ans;
    int s;   // Holds size of structure.
    int change_yes = 0; // Will become TRUE if user finds
    char test_name[25]; // a name to change.

    cout << "\nWhat is the name of the person you want to change? ";
    cin.getline(test_name, NAME_SIZE);
    s = sizeof(Mail); // To ensure fread() reads properly.
```

```
   file.open(FILENAME, ios::in | ios::out);
   if (!file)
   {
      err_msg("*** Read error - Ensure file exists before "
              "reading it ***");
      return;
   }
   do
   {
      file.read((unsigned char *)&item, sizeof(Mail));
      if (file.gcount() != s)
      {
         if (file.eof())
         { break; }
      }
      if (strcmp(item.name, test_name) == 0)
      {
         item.pr_data(&item);   // Print name and address.
         cout << "\nIs this the name and address to " <<
                 "change? (Y/N) ";
         ans = get_answer();
         if (toupper(ans) == 'N')
         {  break;  } // Get another name.
         get_new_item(&item);  // Enable user to type new
                               // information.
         file.seekg((long)-s, ios::cur);  // Back up a structure.
         file.write((const unsigned char *)(&item),
                    sizeof(Mail));   // Rewrite information.
         change_yes = 1;  // Changed flag.
         break; // Finished
      }
   }
   while (!file.eof());
   if (!change_yes)
   {  err_msg("*** End of file encountered before finding the name ***");}
}

void Mail::getzip(Mail *item) // Ensure that ZIP code
                              // is all digits.
{
   int ctr;
```

```
   int bad_zip;

   do
   {
      bad_zip = 0;
      cout << "What is the ZIP code? ";
      cin.getline((*item).zipcode, ZIPCODE_SIZE);
      for (ctr = 0; ctr < 5; ctr++)
      {
         if (isdigit((*item).zipcode[ ctr ]))
         { continue; }
         else
         {
            err_msg("*** The ZIP code must consist of digits only ***");
         bad_zip = 1;
         break;
         }
      }
   }
   while (bad_zip);
}

void Mail::add_to_file(Mail *item)
{
   ofstream file;

   file.open(FILENAME, ios::app);   // Open file in append mode.
   if (!file)
   {
      err_msg("*** Disk error - please check disk drive ***");
      return;
   }
   file.write((const unsigned char *)(item), sizeof(Mail));
                                    // Add structure to file.
}

void Mail::enter_na(Mail *item)
{
   char ans;
```

```
   do
   {
      cout << "\n\n\n\n\nWhat is the name? ";
      cin.getline((*item).name, NAME_SIZE);
      cout << "What is the address? ";
      cin.getline((*item).address, ADDRESS_SIZE);
      cout << "What is the city? ";
      cin.getline((*item).city, CITY_SIZE);
      cout << "What is the state? (2 letter abbreviation only)";
      cin.getline((*item).state, STATE_SIZE);
      getzip(item); // Ensure that ZIP code is all digits.
      strcpy((*item).code, " ");  // Null out the code member.
      add_to_file(item); // Write new information to disk file.
      cout << "\n\nDo you want to enter another name " <<
              "and address? (Y/N) ";
      ans = get_answer();
   }
   while (toupper(ans) == 'Y');
}

//***************************************************************

// Defined constants
// MAX is total number of names allowed in memory for
// reading mailing list.

const int MAX = 250;
const char BELL = '\x07';

//***************************************************************

int main(void)
{
   char ans;
   Mail item;

   do
   {
      disp_menu();  // Display the menu for the user.
      ans = get_answer();
      switch (ans)
      {
        case '1':
```

```
            item.enter_na(&item);
            break;
         case '2':
            item.change_na();
            break;
         case '3':
            print_na();
            break;
         case '4':
            break;
         default:
            err_msg("*** You have to enter 1 through 4 ***");
            break;
      }
   }
   while (ans != '4');
   return 0;
}

//************************************************************

void disp_menu(void) // Display the main menu of program.
{
   clear_sc(); // Clear the screen.
   cout << "\t\t*** Mailing List Manager ***\n";
   cout << "\t\t    --------------------\n\n\n\n";
   cout << "Do you want to:\n\n\n";
   cout << "\t1. Add names and addresses to the list\n\n\n";
   cout << "\t2. Change names and addresses in the list\n\n\n";
   cout << "\t3. Print names and addresses in the list\n\n\n";
   cout << "\t4. Exit this program\n\n\n";
   cout << "What is your choice? ";
}

//************************************************************

void clear_sc() // Clear the screen by sending 25 blank
                // lines to it.
{
   int ctr; // Counter for the 25 blank lines.

   for (ctr = 0; ctr < 25; ctr++)
```

```
   { cout << "\n"; }
}

//***************************************************************

void print_na(void)
{
   Mail item;
   ifstream file;
   int s;
   int linectr = 0;

   s = sizeof(Mail); // To ensure fread() reads properly.
   file.open(FILENAME);
   if (!file)
   {
      err_msg("*** Error - Ensure file exists before"
              "reading it ***");
      return;
   }
   do
   {
      file.read((signed char *)&item, s);
      if (file.gcount() != s)
      {
         if (file.eof()) // If EOF, quit reading.
         { break; }
      }
      if (linectr > 20) // Screen is full.
      {
         pause_sc();
         linectr = 0;
      }
      item.pr_data(&item); // Print the name and address.
      linectr += 4;
   }
   while (!file.eof());
   cout << "\n- End of list -";
   pause_sc(); // Give user a chance to see names
               // remaining on-screen.
}
```

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

void err_msg(char err_msg[ ])
{
   cout << "\n\n" << err_msg << BELL << "\n";
}

//************************************************************

void pause_sc()
{
   cout << "\nPress the Enter key to continue...";
   while (getch() != '\r')
   { ; }  // Wait for Enter key.
}

//************************************************************

char get_answer(void)
{
   char ans;

   ans = getch();
   while (kbhit())
   { getch(); }
   putch(ans);
   return ans;
}
```