Appendix

IV

A Phone Book

After reading the book, one would like to apply the gained knowledge to some practical work. This case study, *Phone Book* is a simple application that incorporates most of the features of C that you have learnt in the book. I will discuss each of the features of this application between the code, although most of the code has comments and you by now would have become quite an expert in writing code in C.

The *Phone Book* application provides features like storing a room/phone number into a database, sorting, finding the room number/phone number etc.

I will refer to any particular line of code during explanation using the line number against each line. Well then let's begin,

```
 1  /*************************************************************************
 2  Program: Phone Book (phoneb.c)
 3  Programmer: Jude D'souza
 4  TESTED/COMPILED - Windows 2000 Pro/Borland Turbo C++ 3.0
 5  *************************************************************************/
 6  #include <stdio.h>
 7  #include <conio.h>
 8  #include <stdlib.h>
 9  #include <graphics.h>
10  #include <ctype.h>
11  #include <string.h>
12  #define MAXDB 500 /* Maximum number of Entries in the phone book */
13
14
15  /* List of Globe variables */
16  int i; /*globe index*/
17  long int phone[MAXDB+1];
18  int room[MAXDB+1];
19  /* phone_tmp & room_tmp array's are temp storage used for delete recovery */
20  long int phone_tmp[MAXDB+1];
21  int room_tmp[MAXDB+1];
22  void AddEntry(int, long int);
23  int add_count=0; /* master counter for entries added */
24  int current_e_add; /* counter for current entries added within a giventime.*/
25  int DeleteEntry(int, long int);
26  int FindPhone(long int);
27  int FindRoom(int);
```

```
28 int phone_found,room_found;
29 int del_entry; /* counts del entry at a given time */
30 int tot_del_entry=0; /* master del counter */
31 int ListAll(void);
32 int SortAllEntries(char);
33 int GeTotalEntries(void);
34 int chkstrdig (char str[], int range);
35 char menu(void);
36 void LoadDB(void); /* load database from file function */
37 void exitmenu(void);
38 void drawscreen(void);
39 void refreshscreen(void);
40
41 char dbload[80]; /* loaded database */
42
43 void main(void)
44 {
45 char iroom[80],iphone[80],add_quit;
46 char option,sortopt,exit_opt; /* menu,sort and exit option*/
47 int phone_check,room_check,delete_check,sort_check,list_check;
48 int iroom_search,iroom_del;
49 int int_iroom,total_entries;
50 int error_iphone,error_iroom; /* used to check inputs error's  */
51 long int longint_iphone;
52 long int iphone_search;
53 long int iphone_del;
54
55 /* Init while no valid database file is loaded program will work in RAM! */
56 strcpy(dbload, "No database file loaded (RAM MODE!).");
57
58 /* MAIN MENU */
59 do
60  {
61   do
62    { option = menu();
63      if (option == '1') /* AddEntry Option */
64      {  current_e_add=0; /*init current entries added to zero.*/
65   for (i=add_count; i < MAXDB; i++)
66   {   clrscr();
67       refreshscreen();
68       drawscreen();
69       gotoxy(1,4);
70       printf(">> Add Entry <<");
71       gotoxy(1,25);
72       cprintf("Please Add Your Entry, leave blank to quit to Main Menu");
73       gotoxy(1,6);
```

```
74        printf("Enter Room  Number[%3d]: ",i+1);
75        gets(iroom);
76
77        if (iroom[0] == '\0' ) /* user hits enter - quits */
78        {  gotoxy(1,25);
             cprintf("You chose to quit: Entry %d was not added to the
79  database.",i+1);
             getch();
             break;
82        }
83        printf("Enter Phone Number[%3d]: ",i+1);
84        gets(iphone);
85
86        if (iphone[0] == '\0') /* user hits enter - quits */
87        {  gotoxy(1,25);
             cprintf("You chose to quit: Entry %d was not added to the
88    database.",i+1);
             getch();
             break;
91        }
92        /* check the string for valid inputs */
93        error_iroom = chkstrdig(iroom,4);
94        error_iphone = chkstrdig(iphone,8);
95        /* loop's while room input error (out of range/character) */
96        while(error_iroom != 0)
97        { if (error_iroom == -1)
98          {  clrscr();
99             refreshscreen();
100            drawscreen();
101            gotoxy(1,4);
102            printf(">> Add Entry <<");
103            gotoxy(1,25);
               cprintf("Error: Room  Number - out of Range, Your entry was greater
104   than 4 digits. ");
105            gotoxy(1,6);
106            printf("Renter Room  Number[%3d]: ",i+1);
107            gets(iroom);
108         }
109         if (error_iroom == -2)
110         {  clrscr();
111            refreshscreen();
112            drawscreen();
113            gotoxy(1,4);
114            printf("*** Add Entry ***");
115            gotoxy(1,25);
               cprintf("Error: Room  Number - Character(s) detected, character(s)
```

```
116  are not allowed.");
117           gotoxy(1,6);
118           printf("Renter Room  Number[%3d]: ",i+1);
119           gets(iroom);
120        }/* checks string room input if valid */
121        error_iroom = chkstrdig(iroom,4);
122       }/*loop's while phone input error (out of range/character) */
123       while(error_iphone !=0)
124       { if (error_iphone == -1)
125         {  clrscr();
126           refreshscreen();
127           drawscreen();
128           gotoxy(1,4);
129           printf(">> Add Entry <<");
130           gotoxy(1,25);
              cprintf("Error: Phone Number - out of Range, Your entry was greater
131  than 8 digits. ");
132           gotoxy(1,6);
133           printf("Room  Number[%3d] Entry: %s",i+1,iroom);
134           gotoxy(1,7);
135           printf("Renter Phone Number[%3d]: ",i+1);
136           gets(iphone);
137         }
138         if (error_iphone == -2)
139         {  clrscr();
140           refreshscreen();
141           drawscreen();
142           gotoxy(1,4);
143           printf(">> Add Entry <<");
144           gotoxy(1,25);
              cprintf("Error: Phone Number - Character(s) detected, character(s)
145  are not allowed.");
146           gotoxy(1,6);
147           printf("Room  Number[%3d] Entry: %s",i+1,iroom);
148           gotoxy(1,7);
149           printf("Renter Phone Number[%3d]: ",i+1);
150           gets(iphone);
151        }/* checks phone input valid */
152        error_iphone = chkstrdig(iphone,8);
153       }
154       /* no room or phone input error - addentry */
155       if (error_iroom == 0 && error_iphone == 0)
156       {  int_iroom = atoi(iroom); /* converts string to int */
157          longint_iphone = atol(iphone); /* converts string to long int */
158          current_e_add++;
159          AddEntry(int_iroom,longint_iphone);
```

```
160       }
161    }
162    if (add_count == MAXDB) /* database full */
163    {  gotoxy(1,25);
164       cprintf("\aDatabase is full!: %d entries were added, ",add_count);
165       cprintf("that is the Maximum No. I can hold.");
166       getch();
167    }
168        }
169        else
170        if (option == '2') /* DeleteEntry option */
171        {  del_entry = 0; /* Initialize del_entry counter zero */
172        clrscr();
173        refreshscreen();
174        drawscreen();
175        gotoxy(1,4);
176        printf(">> Delete Entry <<");
177        gotoxy(1,6);
178        printf("Enter room  number to delete: ");
179        scanf("%d",&iroom_del);
180        flushall(); /* clears buffer */
181
182        printf("Enter phone number to delete: ");
183        scanf("%ld",&iphone_del);
184        flushall();
185
186        delete_check = DeleteEntry(iroom_del,iphone_del);
187
188        if (delete_check == 0)/*successfully found or deleted entries display*/
189        {  gotoxy(1,25);
           cprintf("Successful: There are currently %d entries in the database,
190    ",add_count);
191           cprintf("deleted %d.",del_entry);
192           getch();
193        }
194        if (delete_check == -1) /* error: does not delete if db not found */
195        {  gotoxy(1,25);
           cprintf("Error: The Room No./Phone No. Your looking for was Not Found.
196    ");
197           getch();
198        }
199
200     }
201     else
202     if (option == '3') /* FindPhone Option */
203     {  phone_found = 0; /*initialize phone no. found to zero */
```

```
204         clrscr();
205         refreshscreen();
206         drawscreen();
207         gotoxy(1,4);
208         printf(">> Find Room Number <<");
209
210         gotoxy(1,6);
211         printf("Enter the phone number to search for: ");
212         scanf("%ld",&iphone_search);
213         flushall(); /* clears buffer */
214
215         phone_check = FindPhone(iphone_search);
216
217         if (phone_check == 0) /* return = 0 Phone found */
218         {  gotoxy(1,25);
               cprintf("Successful: There are currently %d entries in the database,
219   ",add_count);
               /* phone_found(globe), counts phone no. found(within FindPhone
               function */
220            printf("found %d.",phone_found);
221            getch();
222         }
223         if (phone_check == -1) /* return = -1 Phone not found */
224         {  gotoxy(1,25);
225            cprintf("Error: The Phone No. Your looking for was Not Found.");
226            getch();
227         }
228         }
229         else
230         if (option == '4') /* FindRoom Option */
231         {  room_found = 0;  /* initialize room no. found to zero */
232            clrscr();
233            refreshscreen();
234            drawscreen();
235            gotoxy(1,4);
236            printf(">> Find Phone Number <<");
237
238            gotoxy(1,6);
239            printf("Enter the room number to search for: ");
240            scanf("%d",&iroom_search);
241            flushall();
242
243            room_check = FindRoom(iroom_search);
244
245            if (room_check == 0) /* return = 0 Room found */
246            {  gotoxy(1,25);
247
```

```
                cprintf("Successful: There are currently %d entries in the data
                base,
",add_count);
                /* room_found is globe it counts room no. found in FindRoom
                function */
                cprintf("found %d.",room_found);
                getch();
            }
            if (room_check == -1) /* return = -1 Room was not found */
            {  gotoxy(1,25);
               cprintf("Error: The Room No. Your looking for was Not Found.");
               getch();
            }

           }
           else
           if (option == '5') /* ListAll option */
           {  clrscr();
              refreshscreen();
              drawscreen();
              gotoxy(1,4);
              printf(">> ListAll <<\n\n");

              list_check = ListAll();

              if (list_check == 0) /* return 0 if entries are in database */
              {  gotoxy(1,25);
                 cprintf("List Sucuessful");
                 getch();
              }
              if (list_check == -1) /* return -1 - emptylist */
              {
                  gotoxy(1,25);
                  cprintf("Empty List");
                  getch();
         }
           }
           else
           if (option == '6') /* Getotalentries option */
           {  total_entries = GeTotalEntries();
gotoxy(1,25);
cprintf("There are currently %d entries stored in the
Database.",total_entries);
            getch();
          }
          else
```

```
290      if (option == '7') /* Sort Option */
291      {  clrscr();
292        refreshscreen();
293        drawscreen();
294        gotoxy(1,4);
295        printf(">> Sort All Entries <<");
296        gotoxy(1,6);
297        printf("Press 'A' to sort database in [A]scending order");
298        gotoxy(1,7);
299        printf("Press 'D' to sort database in [D]escending order.");
300        gotoxy(1,9);
301        printf("Note: Database is sorted by phone no. entries.");
302        sortopt = getch();
303        flushall();
304
305        sort_check = SortAllEntries(sortopt);
306        getch();
307        if (sort_check == 0) /* return = 0 - entries, in db & was sorted */
308        {  gotoxy(1,25);
           cprintf("Database was successfully Sorted.
309  ");
310           getch();
311        }
312        if (sort_check == -1) /* return = -1 - if db is empty */
313        {  gotoxy(1,25);
314           cprintf("Database was not sorted - Database is empty!");
315           getch();
316        }
317      }
318      else
319      if (option == '8') /* Load Database from file option */
320      {  clrscr();
321         refreshscreen();
322         drawscreen();
323         gotoxy(1,4);
324         printf(">> Load Database <<");
325         LoadDB();
326     }
327      else
328      if (option == '9') /* exit option */
329      {  gotoxy(1,25);
         cprintf("Do you really want to exit?, Press 'Y' to confirm, anykey to
330  cancel");
331        exit_opt = getch();
332         flushall();
333         if (exit_opt == 'y' || exit_opt == 'Y')
```

```
334         {  clrscr();
335            refreshscreen();
336            drawscreen();
337            gotoxy(1,4);
338            printf(">> Exit To system <<\n\n");
339            exitmenu();
340         }
341       }
342        else /* user presses an invalid key display msg error */
343       {  gotoxy(1,25);
344          cprintf("Error: Invalid option! Select an option between 1 and 9");
345          getch();
346          flushall(); /* clears buffer */
347       }
348     }while  (option > '9' || option < '1' );
349   }while (option != '`'); /* unlimated loop */
350  }
```

The function main() display a console menu which has the following options:

- Add Entry
- Delete Entry
- Find room number
- Find phone number
- List all entries
- Display total entries in database
- Sort entries
- Load database from file
- Exit

Each of these menu options calls the appropriate function for performing its designated operation.

```
351  /*-------------------------------------------------------------------------
352       AddEntry Function
353       ----------------
354       Does not return any values it is used to added valid inputs(only) into
355       the database and display the entries which was added.
356       * A valid inputs are positive numbers only.
357       1. Room no. input with less than or equal to 4 digits only.
358       2. Phone no. input with less than or equal to 8 digits only.
359       3. Input of Zero for room no. or phone no. inputs is invalid.
360  -------------------------------------------------------------------------*/
361  void AddEntry(int r, long int p)
362  {
363       room[i] = r;  /* store r(room) input into db */
364       phone[i] = p; /* store r(room) input into db */
365       add_count++;  /* keeps track of total entries added. */
```

```
        printf("\nRoom No.  [%-4d]\nPhone No. [%-8ld]\n%d entries
366 added.",r,p,current_e_add);
367        getch();
368 }
369 /*-------------------------------------------------------------------------

370        DeleteEntry function
371        ————————
372        Used to delete entrys in the database.
373        Returns 0 if room no. & phone no. was found in the database.
374        Returns -1 if room no. & phone no. is not found in the database.
375
376        Note: Auto-Recovery was implemented into this function but was never
377        used. room_tmp and phone_tmp arrays contain the deleted data
378        which maybe used for recovery.
379 -------------------------------------------------------------------------*/
380 int DeleteEntry(int r, long int p)
381 {
382 int  k,x,del_found_flag=-1,loop_mov_stop,loop_mov,count_del=0;
383 char del_me; /* Variable to confirm delete */
384
385     for(k=0; k < add_count; k++)
386     {  if (add_count != 0) /* checks if database is not empty */
387        {  if (r == room[k] && p == phone[k])
388        {  gotoxy(1,8);
389           printf("Match Found: \n");
           printf("Room No. [%-4d]\tPhone No. [%-8ld] was found in record No.
390 [%3d  ]\n",room[k],phone[k],k+1);
391           del_found_flag = 0; /* when found, set's del_found_flag=0 */
392           gotoxy(1,25);
           cprintf("Delete record [%3d  ]?, Press 'Y' to confirm, anykey to
393 cancel.",k+1);
394           del_me = getch();
395           flushall();
396           if (del_me == 'y' || del_me == 'Y')
           {  room_tmp[tot_del_entry] = room[k]; /* tmp array storage for room
397 found */
398           room[k] = -1; /* marks -1 for deleted */
399           phone_tmp[tot_del_entry] = phone[k];
400           phone[k] = -1;
401           del_entry++; /* counter for deleted entry */
402           tot_del_entry++; /* counter for temp storage */
403          }
404        }
405        }
```

```
406       }
407       if (add_count !=0) /* if database is not empty process with delete */
408       {/* keeps looping while move up position is not = to deleted entry */
409          for (x=0; x < del_entry; x++)
410          {   for (k=0; k < add_count; k++)
411           {/* When -1 is found it moves everything by one */
412              if (room[k] == -1 && phone[k] == -1)
413              {  loop_mov_stop=0;
414              loop_mov =0;
415              count_del++;
416              /* loop_mov_stop calculates moves needed */
417              loop_mov_stop = add_count-(k+1);
418              while (loop_mov_stop != loop_mov)
419              {      room[k+loop_mov] = room[(k+1)+loop_mov];
420                     phone[k+loop_mov] = phone[(k+1)+loop_mov];
421                     loop_mov++; /* counter for move */
422              }
423              }
424           }
425          }
426       }
427       /* Calcalates total entry */
428       add_count = add_count - del_entry;
429
430       if (del_found_flag == 0) /* flag is 0 when delete entry input was found
*/
431       { return(0);  } /* return sucessful */
432       Else
433       { return(-1); } /* return not found */
434    }
```

Let's take a closer look at how the DeleteEntry function works. To make things easier let,
Room =1,2,3,4,5,6,7,8,9,10
Phone=1,2,3,4,5,6,7,8,9,10

Ten entries in the database with the digits from 1 to 10 both having the same values entered. Now if the user requests Room/Phone "4" to be deleted, the delete entry function will find the digit "4" in both Room and Phone matching the user's request.

Find -> is done within a for loop until add_count number is reached, Add_count is the counter for the number of entries added (Line 385). If it finds the digit '4' it asks the user if he/she wants to delete the current entry in the record.

This is what happens when the user selects 'Yes',

1) Copy that current entry to a temp location (Lines 397, 399).
2) Then a '-1' is copied on top of the location where digit '4' was found overwriting it (marking it has been deleted) (Lines 398, 400). Tot_del_entry and del_entry is incremented by one each time this is done (Lines 401,402).

3) Another for loop is nested within the for, used to find '-1's' marked for deleted, it loops for the no. of entries that has been deleted (Lines 409, 412). Calculation of the move up stop position is done on line 417.

4) Then using the while loop (Line 418) everything is moved up by one position. At the end of the while loop (Line 428), the number of records that exist after deletion has been done is calculated.

```
435   /*-------------------------------------------------------------------------
436       FindPhone function
437       ------------------
438       Used to search for a phone number in the database.
439
440       Returns 0 if phone no. was found.
441       Returns -1 if phone no. is not found.
442   -------------------------------------------------------------------------*/
443   int FindPhone(long int p)
444   {
445   int k, phone_found_flag= -1;
446       gotoxy(1,8);
447       for(k=0; k < add_count; k++)
448       {  if (add_count != 0) /* if database is not empty then run a search */

449          {  if (k != 0 && (k%15) == 0)
450         {  gotoxy(1,8); /* moves cursor to beginning when screen filled */
451            getch();
452         }
453         if (p == phone[k])
          {  printf("Phone No. [%-8ld] was found in record No. [%3d  ]\tRoom No.
454   [%-4d]\n",phone[k],k+1,room[k]);
455            phone_found++;
456            phone_found_flag = 0;
457         }
458          }
459       }
460       if (phone_found_flag == 0) /* flag is 0 if record was found */
461       { return(0);  } /* return sucessful */
462       else
463       { return(-1); } /* return not found */
464   }
465   /*-------------------------------------------------------------------------
466       FindRoom function
467       -----------------
468       Used to search for a Room number in the database.
469
470       Returns 0 if room no. was found.
471       Returns -1 if room no. is not found.
472   -------------------------------------------------------------------------*/
```

```
473    int FindRoom(int r)
474    {
475    int k, room_found_flag=-1;
476
477        gotoxy(1,8);
478        for(k=0; k < add_count; k++)
479        {  if (add_count != 0) /* if database is not empty then run a search */

480           {  if (k != 0 && (k%15) == 0)
481         {  gotoxy(1,8); /* moves cursor to beginning when screen filled */
482            getch();
483         }
484         if (r == room[k])
            {  printf("Room No. (%-4d) was found in record No. [%3d  ]\tPhone No.
485    (%-81d)\n",room[k],k+1,phone[k]);
486            room_found++;
487            room_found_flag = 0;
488         }
489          }
490        }
491
492        if (room_found_flag == 0)
493        { return(0);  } /* return sucessful */
494        else
495        { return(-1); } /* return not found */
496    }
```

The FindRoom and FindPhone function simply match the input parameter to the function with the room and phone array and prints out the values if a match is found.

```
497    /*-------------------------------------------------------------------------
498       ListAll Function
499       ----------------
500       Used for displaying data entered into the database.
501       returns -1 if list is empty.
502       returns 0 if sucessful (database contains valid entries)
503    -------------------------------------------------------------------------*/
504    int ListAll(void)
505    {
506    int k;
507        gotoxy(1,6);
508        for (k=0; k < add_count; k++)
509        {
510       if (k != 0 && (k%17) == 0)
511       {  gotoxy(1,6); /* moves cursor to beginning when screen filled */
512          getch();
```

```
513         }
514         /* double checks - it will not print out delete entries(-1) */
515         if (room[k] != -1 && phone[k] != -1)
516         {  printf("Room Number [%3d  ]: %-4d\t",k+1,room[k]);
517            printf("Phone Number[%3d  ]: %-8ld\n",k+1,phone[k]);
518         }
519         }
520         if (add_count == 0)
521         {  return(-1); } /* Empty List */
522         else
523         {  return(0);  } /* Successful */
524     }
```

The ListAll function iterates through the room and phone array and displays the phone book entries to the user.

```
525     /*-------------------------------------------------------------------------
526        GeTotalEntries function
527        ----------------------
528        Used to return total entries added to database.
529     -------------------------------------------------------------------------*/
530     int GeTotalEntries(void)
531     {
        /* This function is not required as it does nothing but returns the total
532     entries that have been added.*/
533
534        return(add_count);
535     }
536     /*-------------------------------------------------------------------------
537        SortAllEntries function
538        ----------------------
539        Sort is done with the use of bubble sort.
540        returns 0 if sort was successful.
541        returns -1 if database is empty.
542     -------------------------------------------------------------------------*/
543
544     int SortAllEntries(char sel)
545     {
546     int k,room_str_tmp,sortalldone;
547     long int phone_str_tmp;
548         if (add_count !=0) /* if list not empty continue with sort */
549         {  do
550            { sortalldone=0;
551         for (k = 0; k < add_count; k++)
552         {   /* sort in ascending order */
               if ((phone[k] > phone[k + 1])&&(sel == 'a' || sel == 'A')&&(k !=
```

```
553     add_count -1))
                {   phone_str_tmp = phone[k]; /* stores previous array to
554     phone_str_tmp */
                 phone[k] = phone[k + 1]; /* copys next array to the previous
555     array before it */
                 phone[k + 1] = phone_str_tmp; /* Previous array is copyed to next
556     array */
557              /* same process is done here but with room no. */
558              room_str_tmp = room[k];
559              room[k] = room[k + 1];
560              room[k + 1] = room_str_tmp;
561              sortalldone =1; /* sets to 1 if sort is done */
562             }
563             /* same method used here but sorts in decending order */
                if ((phone[k] < phone[k + 1])&&(sel == 'd' || sel == 'D')&&(k !=
564     add_count -1))
565             {   phone_str_tmp = phone[k];
566              phone[k] = phone[k + 1];
567              phone[k + 1] = phone_str_tmp;
568              room_str_tmp = room[k];
569              room[k] = room[k + 1];
570              room[k + 1] = room_str_tmp;
571              sortalldone =1;
572             }
573         }
574            }while (sortalldone);
575        }
576
577        if ((sel == 'a' || sel == 'A')&&add_count !=0)
578        {   gotoxy(1,25);
579             printf("You have chosen to sort the database in [A]scending order. ");
580             return(0);    /* sucessfully sorted */
581        }
582        else
583        if ((sel == 'd' || sel == 'D')&&add_count !=0)
584        {  gotoxy(1,25);
585            printf("You have chosen to sort the database in [D]ecending order. ");
586           return(0);     /* sucessfully sorted */
587        }
588        else
589        if ((sel != 'a' || sel != 'A' || sel != 'd' || sel != 'D')&&add_count !=0)
590        {  gotoxy(1,12);
591            printf("Invalid option - database was not sorted!");
592        }
593        else
594        { return(-1); } /* list empty */
595     }
```

The SortAllEntries uses the bubble sort method for sorting the phone book entries. The bubble sorting method used a temporary variable to swap the values. Sorting is done both in the ascending as well as the descending order. For the ascending order, each time the previous value is greater than the next value, the two values are swaped. For the sescending order, the same principal is used, only when the previous value is less than the next then the value is swapped. This entire process happens within a for loop.

```
596    /*-------------------------------------------------------------------------
597       Chkstrdig(Check string for digits)
598       ---------------------------------
599       Used to check string inputs, and lenght.
600       returning:
601       -1 = if string lenght is out of range.
602       -2 = if a char is found within the string.
603       0 = if successful
604
605       NOTE:
606       Addentry specifed that Room allows for 4 digits or less.
607       Phone allowing 8 digits or less.
608    -------------------------------------------------------------------------*/
609    int chkstrdig (char str[], int range)
610    {
611    int lenght=0,k;
612        lenght = strlen(str); /* get lenght of string */
613
614        if (lenght > range)
615        {    return(-1);} /* out of range */
616        if (lenght <= range) /* string is in range */
617        {  for (k=0; k < lenght; k++)
618           {   if (isdigit(str[k]) == 0)
619             { return(-2); } /* char detected return error msg */
620           }
621        return(0); /* successful - input string was in range and was digits */
622        }
623    }
624    /*-------------------------------------------------------------------------
625       Load up database from file function
626       -----------------------------------
627       Used to load a file into the database.
628    -------------------------------------------------------------------------*/
629    void LoadDB(void)
630    {
631    int count,nofilen,dbfilecount=0;
632    char finphone[80];
```

```
633    char finroom[80];
634    int error_junk;
635
636    long int l_finphone;
637    int i_finroom;
638
639    FILE *f1;
640         gotoxy(1,6);
641         printf("Enter the filename of the database: ");
642         gotoxy(1,7);
643         printf("Example: c:\\mydbfile.txt");
644         gotoxy(37,6);
645         gets(dbload);
646         flushall(); /* clears all buffers */
647         f1 = fopen (dbload,"r"); /* open file for reading */
648         if (f1==NULL)
649         {  gotoxy(1,25);
650       fprintf(stderr,"There was an error reading your database file!");
651       strcpy(dbload, "No database file loaded (RAM MODE!).");
652       getch();
653       exit;
654         }
655         else
656         {    for (count=0; count < MAXDB; count++)
657       {    if (!feof(f1)) /* if not end of file continue to input data */
658            { fscanf(f1,"%s\t%s\n",&finroom,&finphone);
659              /* saves info to room, phone array */
660            error_junk = chkstrdig(finroom,4);
661            error_junk = chkstrdig(finphone,8);
662              if (error_junk == -1 || error_junk == -2)
663              {  printf("Sorry that was an invalid database\n");
664                 printf("Now working in RAM MODE!");
665                 getch();
666                 strcpy(dbload, "No database file loaded (RAM MODE!).");
667            break;
668          }
669              i_finroom = atoi(finroom); /* converts string to int */
670              l_finphone = atol(finphone); /* converts string to long int */
671              room[count] = i_finroom;
672              phone[count] = l_finphone;
673              dbfilecount++; /* counts no. of records stored in file */
674            }
675       }
676            if (error_junk ==0)
677            {
678          gotoxy(1,25);
```

```
679          printf("Database %s, was successfully loaded!",dbload);
680          getch();
681          /* copys no. of records in file into master counter*/
682          add_count = dbfilecount;
683            }
684          }
685          fclose(f1);
686     }
```

The LoadDB function loads the phone book entries from a flat file. The file is opened on line 647 using fopen and the data is loaded into the room and phone arrays (lines 656...675).

```
687     /*****************************************************************************

688        MAIN function
689        ------------
690        Menu, ExitMenu, drawscreen and refreshscreen.
691     *****************************************************************************/
692     /*---------------------------------------------------------------------------
693        Menu function
694        ------------
695        Display valid options on the screen
696     ---------------------------------------------------------------------------*/
697     char menu(void)
698     {
699     char optrtn;
700          clrscr();
701          window(1,1,80,25); /*Set position and screen mode*/
702          refreshscreen();
703          drawscreen();
704          gotoxy(1,4);
705          printf("[1] - Add entry\n");
706          printf("[2] - Delete entry\n");
707          printf("[3] - Find room number\n");
708          printf("[4] - Find phone number\n");
709          printf("[5] - List all entries\n");
710          printf("[6] - Display total entries in database\n");
711          printf("[7] - Sort entries\n");
712          printf("[8] - Load database from file\n");
713          printf("[9] - Exit");
714          gotoxy(1,25);
715          cprintf("Please select an option between 1 and 9.");
716          gotoxy(1,15);
717          printf("Database loaded: %s",dbload);
718          gotoxy(1,14);
```

```
719         printf("Select an option: ");
720         optrtn = getch();
721         return optrtn;
722     }
723     /*-------------------------------------------------------------------------
724        ExitMenu function
725        -----------------
726        While exiting to system, askes user if he/she wants to save
727        database into a file.
728     -------------------------------------------------------------------------*/
729     void exitmenu(void)
730     {    char filename[20],save_opt;
731          int k;
732          FILE *f1;
733          gotoxy(1,6);
734          printf("Do You want to Save database before exiting? ");
735          gotoxy(1,25);
736          cprintf("Press 'Y' to confirm, anykey to cancel.");
737          save_opt = getch();
738          flushall();
739          if (save_opt == 'y' || save_opt == 'Y')
740          {  gotoxy(1,8);
741        printf("Please Enter the path and filename to save to:");
742        gotoxy(1,10);
743        printf("Example: c:\\mydbfile.txt");
744        gotoxy(48,8); /* move cursor back to line 8 */
745        gets(filename);
746        flushall();
747        f1 = fopen (filename,"a"); /*open file for appending mode */
748        if (f1== NULL)
749        {  gotoxy(1,12);
750           fprintf(stderr, "Error opening file %s.",filename);
751           gotoxy(1,25);
752           cprintf("Database was not saved!
753     ");
754           getch();
755           exit;
756        }
757        else
758        {   for (k=0; k < add_count; k++)
759            {   fprintf(f1, "%d\t%ld\n",room[k],phone[k]);}
760             fclose(f1);
761             gotoxy(1,25);
762             cprintf("Database was successfully saved in %s",filename);
763             getch();
            }
```

```
764         }
765         else
766         {   gotoxy(1,25);
                cprintf("Database was not saved!
767     ");
768         getch();
769         }
770         clrscr();
771         gotoxy(23,10);
772         printf("Thank you for using this program");
773         gotoxy(23,11);
774         printf("Coded by: Jude D'souza!");
775         gotoxy(23,13);
776         printf("Email: shrewdjackal@yahoo.com");
777         getch();
778         exit(0);
779     }
780     /*-------------------------------------------------------------------------
781        Drawscreen function
782        --------------------
783        Draws program header.
784     -------------------------------------------------------------------------*/
785     void drawscreen(void)
786     {
787         gotoxy(1,1);
              cprintf("-------------------------------------------------------------
788     ------------");
789         gotoxy(1,2);
            cprintf("                              *** PHONE BOOK ***
790     ");
791         gotoxy(1,3);
              cprintf("-------------------------------------------------------------
792     ------------");
793     }
794     /*-------------------------------------------------------------------------
795        Refreshscreen function
796        ----------------------
797        used to refresh colour display.
798     -------------------------------------------------------------------------*/
799     void refreshscreen(void)
800     {    clrscr();
801         textcolor(WHITE);
802         textbackground(BLACK);
803         gotoxy(1,25);
            cprintf("
```

```
804     ");
805          clrscr();
806          textcolor(WHITE);
807          textbackground(BLUE);
808          gotoxy(1,25);
             cprintf("
809     ");
810     }
811
812     /* EOF */
```

The above functions are used to draw the menu and the exit message on the screen. The ExitMenu function performs the task of saving the data to a flat file before closing the application.

I hope the above case study has been useful to you and will enable you to write applications in C. You could work upon this *Phone book* application and incorporate more features. Try using link lists and binary trees to store the Phone/Room numbers instead of arrays. Remember the saying 'Practice makes perfect'. Happy programming...