

COP 3223: C Programming Spring 2009

File Processing In C – Part 2

Instructor : Dr. Mark Llewellyn
 markl@cs.ucf.edu
 HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cop3223/spr2009/section1>

School of Electrical Engineering and Computer Science
University of Central Florida



More File I/O

- Let's look carefully at Practice Problem #1 from the File Processing Part 1 notes.
- In this program you were to create an input file named "pracprob1.dat", where the first line contained the number of integers that were to be entered by the user, rather than have the program already have determined this number.
- While this obviously makes the program more versatile, since it works with any size input, it also means that the data file contains more than one "kind" of data. In other words, the first line of the file has a different meaning to the program than do any of the other lines.
- Remember that C itself does not impose any structure on the contents of the file, the programmer must do that.



```
4
5 #include <stdio.h>
6
7 int main()
8 {
9     FILE *inFilePtr; //pointer to the input file
10    int i; //loop counter
11    int limit; //first line in file indicating how many numbers
12    int valueRead; //value of integer read from file
13    int sum = 0; //holds sum of first i integers
14
15    //open file
16    if ( (inFilePtr = fopen("pracprob1.dat", "r") ) == NULL )
17        printf("Sorry the file could not be opened\n");
18    //end file open if stmt
19    else {
20        fscanf(inFilePtr, "%d", &limit);
21        for (i = 1; i <= limit; i++) {
22            fscanf(inFilePtr, "%d", &valueRead);
23            sum = sum + valueRead;
24            printf("The value read was: %d. The sum so far is: %d\n", valueRead, sum);
25        } //end for stmt
26        fclose(inFilePtr); //close file
27    } //end else
28
29    printf("\n\n");
30    system("PAUSE");
31    return 0;
32 } //end main function
```

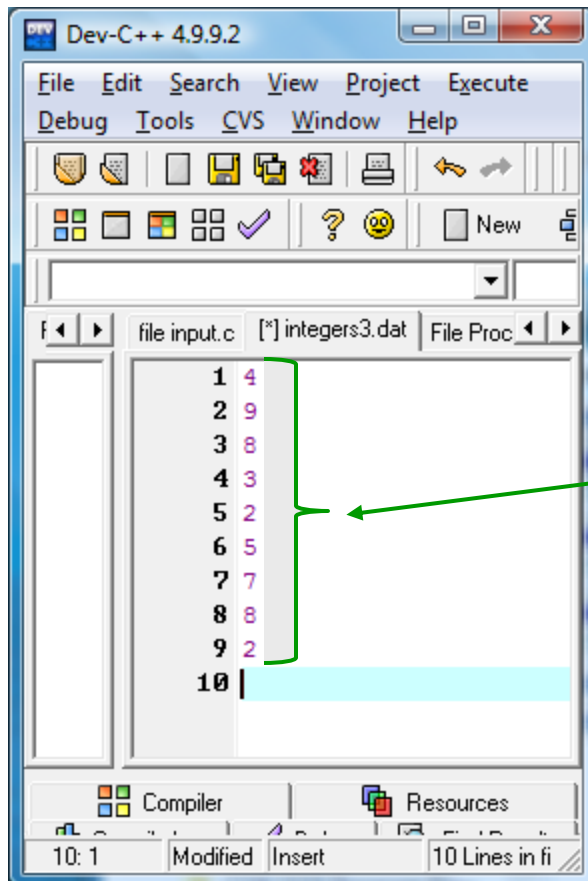
The first value read from the file represents the number of integers to be read from the rest of the file.

All of the rest of the values in the file represent "user" entered integer values to be summed.



More File I/O

- Given the file structure that is assumed by the program on page 3, what would happen if the following input file were used as the input to the program?



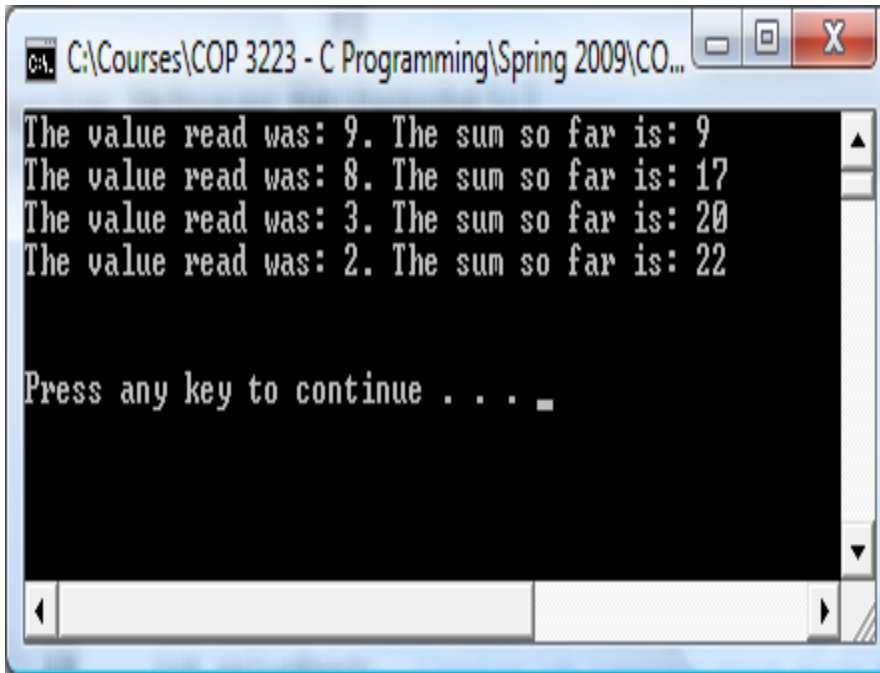
```
1 4
2 9
3 8
4 3
5 2
6 5
7 7
8 8
9 2
10 |
```

The sum of these numbers is 48. Will the program produce this result?



More File I/O

- The answer is no! Since the first number in the file was a 4, this was read into the variable named `limit` and the `for` loop on line 21 executed four times, which means that only the values 9, 8, 3, and 2 were read from the file. The remaining values of 5, 7, 8, and 2 were never read by the program!



```
C:\Courses\COP 3223 - C Programming\Spring 2009\CO...
The value read was: 9. The sum so far is: 9
The value read was: 8. The sum so far is: 17
The value read was: 3. The sum so far is: 20
The value read was: 2. The sum so far is: 22

Press any key to continue . . . _
```



Detecting End-Of-File

- Suppose that our input file consists only of integer values that we want to sum. In other words, the first number is not indicating how many numbers are in the file, but is just one of the numbers that we want to sum.
- Since the program has no way of knowing in advance how many values will be read from the file to sum, two things become obvious: (1) we can't use a `for` loop, since we don't know how many times it will need to be executed, and (2) we need a way to be able to eventually stop the loop when the program has read all of the numbers in the input file.
- The solution to (1) is that we must use either a `while` or `do...while` stmt to control the reading from the file.



Detecting End-Of-File

- The solution to (2) is that we need a way to determine if we have seen all the numbers in the file. This is known as detecting the **end-of-file**.
- In C, the function used to detect the end-of-file is named `feof` and has the following form:

```
feof ( fileptr )
```

- The `feof` function returns a non-zero value if the end-of-file is true for the file referenced by `fileptr`, otherwise the function returns 0.
- Think of the end-of-file marker as a “special” character in the file that you can’t see but appears right after the very last actual value in the file. The next example illustrates using `feof`.



```
3
4 #include <stdio.h>
5
6 int main()
7 {
8     FILE *inFilePtr; //declare input file pointer
9     int sum = 0; //hold running sum
10    int value; //input value to be summed
11
12    if ( (inFilePtr = fopen("numbers.dat","r") ) == NULL ) {
13        printf("Sorry, the file could not be opened\n");
14    }
15    else {
16        fscanf(inFilePtr, "%d", &value); //get first number from file
17        while ( !feof(inFilePtr) ){
18            sum += value;
19            printf("The value was: %d. The running sum is: %d\n", value, sum);
20            fscanf(inFilePtr, "%d", &value); //get next number from file
21        } //end while stmt
22    } //end else
23
24    fclose(inFilePtr); //close input file
25    printf("\n\n");
26    system("PAUSE");
27    return 0;
28 } //end main function
```




```
1 2
2 4
3 6
4 8
5 10
6 12
7 14
8 16
9 18
10 20
11 22
12 24
13 26
14 28
15 30
16
```

```
The value was: 2. The running sum is: 2
The value was: 4. The running sum is: 6
The value was: 6. The running sum is: 12
The value was: 8. The running sum is: 20
The value was: 10. The running sum is: 30
The value was: 12. The running sum is: 42
The value was: 14. The running sum is: 56
The value was: 16. The running sum is: 72
The value was: 18. The running sum is: 90
The value was: 20. The running sum is: 110
The value was: 22. The running sum is: 132
The value was: 24. The running sum is: 156
The value was: 26. The running sum is: 182
The value was: 28. The running sum is: 210
The value was: 30. The running sum is: 240

Press any key to continue . . . _
```

Think of the end-of-file marker as being right after the 0 in the 30 on the last line of the file.



Detecting End-Of-File

- Notice in the program on page 8 that the first line from the file is read before the while statement is entered and thus, the first test for the end-of-file occurs after we have already read one integer from the file.
- We are making the assumption that there is at least one integer value in the input file. If this is not true, would our program would fail? (Try it!)
- Answer: No, it will not fail, since the `fscanf` call on line 16 will actually read the end-of-file marker (it has an integer representation (as do all characters) and thus when the call to `feof` occurs in the conditional expression of the while statement in line 17, it will return true and since `not true` is false, the loop will not execute.



Detecting End-Of-File

- What would happen if we redid the program on page 8 using a `do...while` loop instead of a `while` loop, but making no other changes. Would it still work correctly? (Try this one too.)

- Answer: Yes, assuming that there is at least one actual value in the file, otherwise, the end-of-file marker will be read, but the `feof` test will not occur until after we've gone through the body of the loop the first time and actually produced a sum, whose value will be useless.



```
1 //using a while loop and testing for end-of-f
2 //February 2, 2009   Written by: Mark Llevel
3
4 #include <stdio.h>
5
6 int main()
7 {
8     FILE *inFilePtr; //declare input file po
9     int sum = 0; //hold running sum
10    int value; //input value to be summed
11
12    if ( (inFilePtr = fopen("numbers.dat","r") ) == NULL ) {
13        printf("Sorry, the file could not be opened\n");
14    }
15    else {
16        fscanf(inFilePtr, "%d", &value); //get first number from file
17        do{
18            sum += value;
19            printf("The value was: %d. The running sum is: %d\n", value, sum);
20            fscanf(inFilePtr, "%d", &value); //get next number from file
21        } while ( !feof(inFilePtr)); //end do...while stmt
22    } //end else
23
24    fclose(inFilePtr); //close input file
25    printf("\n\n");
26    system("PAUSE");
27    return 0;
28
```

```
C:\Courses\COP 3223 - C Programming\Spring ...
The value was: 2. The running sum is: 2
The value was: 4. The running sum is: 6
The value was: 6. The running sum is: 12
The value was: 8. The running sum is: 20
The value was: 10. The running sum is: 30
The value was: 12. The running sum is: 42
The value was: 14. The running sum is: 56
The value was: 16. The running sum is: 72
The value was: 18. The running sum is: 90
The value was: 20. The running sum is: 110
The value was: 22. The running sum is: 132
The value was: 24. The running sum is: 156
The value was: 26. The running sum is: 182
The value was: 28. The running sum is: 210
The value was: 30. The running sum is: 240
Press any key to continue . . . _
```



A Little Case Study

- A bank has a number of customer accounts. Each account has a balance, which is either positive, zero, or negative (overdrawn). Let's assume that an account number is a 5-digit integer.
 1. We want to create a program that will ask the user to enter a series of account numbers and their corresponding balances and write this information to a file.
 2. Then we want to create a second program that will allow a user to print listings of all the accounts that have either positive, zero, or negative balances. We want the second program to be “menu-driven” where we give the user a choice of options to be performed by the program.



A Little Case Study – Part 1

- To solve the first part of our problem, let's assume that we'll ask the user to first tell the program how many accounts and balances they will enter.
- Once we have this number, we can run a counted loop (a for statement) to read the account number and corresponding balance for each account to be entered.
- Once each account number and balance have been entered, the program will write the values to an output file. Let's call this file "accounts.dat".
- Let's do this first step now:



```
8 int main()
9 {
10     int accountNumber; //user account number
11     float balance; //user account balance
12     int howMany; //number of user accounts to be entered
13     int i; //loop counter
14     FILE *outFilePtr; //output file pointer
15
16     if ( ( outFilePtr = fopen("accounts.dat", "w") ) == NULL ) {
17         printf("Sorry, the file couldn't be created\n");
18     } //end if
19     else {
20         printf("How many accounts will you enter? ");
21         scanf("%d", &howMany);
22         printf("\n");
23         //enter account information
24         for (i = 1; i <= howMany; i++) {
25             printf("Enter a 5-digit account number and balance: ");
26             scanf("%5d%f", &accountNumber, &balance);
27             printf("\n");
28             fprintf(outFilePtr, "%5d %8.2f \n", accountNumber, balance);
29         } //end for stmt
30         //all account information is entered, so close file
31         fclose(outFilePtr);
32     } //end else
33
34     printf("\n\n");
35     ...
```



```
C:\Courses\COP 3223 - C Programming\Spring 2009\COP 3223 Program Fi...
How many accounts will you enter? 8
Enter a 5-digit account number and balance: 11111 156.98
Enter a 5-digit account number and balance: 22222 0.00
Enter a 5-digit account number and balance: 33333 -78.25
Enter a 5-digit account number and balance: 44444 1254.00
Enter a 5-digit account number and balance: 55555 -65.10
Enter a 5-digit account number and balance: 66666 23.10
Enter a 5-digit account number and balance: 77777 0.00
Enter a 5-digit account number and balance: 88888 456.33
Press any key to continue . . .
```

User interaction with program 1

```
Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug
Tools CVS Window Help
New Insert
case study part 1.c accounts.dat
1 11111 156.98
2 22222 0.00
3 33333 -78.25
4 44444 1254.00
5 55555 -65.10
6 66666 23.10
7 77777 0.00
8 88888 456.33
9
Compiler Resources
1: 1 Insert 9 Lines in file
```

The contents of the output file "accounts.dat" after user entered the data



A Little Case Study – Part 2

- To solve the second part of our problem, we need to create a menu of options for the user. Let's assume that option 1 will be to print accounts with positive balances, option 2 accounts with zero balances, option 3 accounts with negative balances, and option 4 will be to quit the program.
- The program needs to display the menu once and then continue to loop until the user enters option 4 to stop the program.
- We'll need to be able to scan the file from top to bottom repeatedly. So far, we've only opened a file, moved through it from top to bottom once, and then closed the file. If you need to move through the file again, use the `rewind` function, whose format is `rewind(fileptr);`



A Little Case Study – Part 2

- When you attempt to solve problems similar to this, you should always use a systematic and methodical approach.
1. Sketch out the main objectives of your solution. In this case (1) creating the menu and (2) listing the different options given to the user.
 2. Proceed in a systematic fashion with your coding. Thus, first write the code that creates the menu and asks the user for their choice and test that you are correctly reading in the value entered by the user. Then write the code for each option separately and test each case before moving on to other cases. By systematic coding and testing you will be able to narrow down where errors occur and accomplish more than attempting to put everything into code and once and having to figure out where things are going wrong.



```
7 |
8 #include <stdio.h>
9
10 int main()
11 {
12     int userOption; //user's choice for option
13     int i; //loop counter
14     int accountNumber; //account number
15     float balance; //account balance
16     FILE *inFilePtr; //input file pointer
17
18     if ( (inFilePtr = fopen("accounts.dat", "r")) == NULL ) {
19         printf("Sorry, couldn't open input file\n");
20     }
21     else {
22         //display menu
23         printf("Select option from menu\n");
24         printf("Enter 1 to list accounts with positive balances\n");
25         printf("Enter 2 to list accounts with zero balances\n");
26         printf("Enter 3 to list accounts with negative balances\n");
27         printf("Enter 4 to quit program\n");
28         printf("\nYour selection: ");
29         scanf("%d", &userOption);
30         printf("\n\n");
31
```



```
32     while ( userOption != 4) {
33         fscanf(inFilePtr, "%5d%f", &accountNumber, &balance);
34         switch (userOption) {
35             case 1:
36                 printf("Accounts with positive balances\n");
37                 printf("-----\n");
38                 //read file until eof
39                 while ( !feof(inFilePtr) ) {
40                     if ( balance > 0 ) {
41                         printf("%5d%8.2f\n", accountNumber, balance);
42                     }//end if
43                     fscanf(inFilePtr, "%5d%f", &accountNumber, &balance);
44                 }//end while stmt
45                 break; //end case 1
46
47             case 2:
48                 printf("Accounts with zero balances\n");
49                 printf("-----\n");
50                 //read file until eof
51                 while ( !feof(inFilePtr) ) {
52                     if ( balance == 0 ) {
53                         printf("%5d %8.2f\n", accountNumber, balance);
54                     }//end if
55                     fscanf(inFilePtr, "%5d%f", &accountNumber, &balance);
56                 }//end while stmt
57                 break; //end case 2
```



```
58
59     case 3:
60         printf("Accounts with negative balances\n");
61         printf("-----\n");
62         //read file until eof
63         while ( !feof(inFilePtr) ) {
64             if ( balance < 0 ) {
65                 printf("%5d %8.2f\n", accountNumber, balance);
66             } //end if
67             fscanf(inFilePtr, "%5d%f", &accountNumber, &balance);
68         } //end while stmt
69         break; //end case 3
70     } //end switch statement
71
72     rewind(inFilePtr); //return file pointer to top of file
73     printf("\nYour selection: ");
74     scanf("%d", &userOption);
75     printf("\n");
76     } //end while stmt
77     printf("Thank you!\n\n");
78     fclose(inFilePtr); //close the input file
79 } //end else
80     system("PAUSE");
81     return 0;
82 } //end main function
83
```



```
C:\Courses\COP 3223 - C Programming\Spring 2009\COP 3223 P...
Select option from menu
Enter 1 to list accounts with positive balances
Enter 2 to list accounts with zero balances
Enter 3 to list accounts with negative balances
Enter 4 to quit program

Your selection: 1

Accounts with positive balances
-----
11111  156.98
44444 1254.00
66666   23.10
88888  456.33

Your selection: 2

Accounts with zero balances
-----
22222    0.00
77777    0.00

Your selection: 3

Accounts with negative balances
-----
33333  -78.25
55555  -65.10

Your selection: 4

Thank you!

Press any key to continue . . . _
```

User interaction with program 2



Practice Problems

1. Modify the case study program part 1 on page 15 so that it reads the account information from an input file where there is no indication of how many accounts are listed in the file.
2. Modify the case study program part 2 so that the output of the various cases is written to both the screen and to an output file named `"account results.dat"`.
3. Modify the case study program part 2 so that the menu is reprinted after each option selected by the user has finished producing output.

