COP 3223: C Programming
Spring 2009

File Processing In C – Part 1

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

File I/O In C

- In C, the term stream means any source of input or any destination for output. All of the programs that we've developed up to this point have obtained their input from a single stream (the keyboard) and have written all their output to a single stream (the screen via a command prompt window).
- Large programs may have several streams active at any given time, both for input and output.
- While streams can be associated with virtually any I/O device that is connected to your computer, we'll focus only on streams that deal with files (for now at least).

File I/O In C

- C views each file simply as a sequential stream of bytes. There are two basic types of files in C, text files and binary files. For now, we will consider only text files.
- When a file is opened, a stream is associated with the file.
- Three files and their associated streams are automatically opened when program execution begins – the standard input, the standard output, and the standard error.
- Streams provide communication channels between files and programs.
- For example, the standard input stream enables a program to read input data from the keyboard. The standard output stream enables a program to print data on the terminal screen.

File I/O In C

- Opening a file returns a pointer to a FILE structure, which is defined in `<stdio.h>` that contains information the computer needs to access and process the file.
- The standard input, standard output, and standard error streams are accessed using file pointers `stdin`, `stdout`, and `stderr`.
- The standard library `<stdio.h>` provides many different functions for reading and writing data from streams (files). We've been using `scanf` and `printf` for some time now and you should be quite familiar with these functions.
- The `scanf` function reads from the `stdin` stream while the `printf` function writes to the `stdout` stream.
-

How To Create A File Pointer

- Before you can read or write to a file (stream) in C, you must define a pointer to that file. This is done as follows in C:

```
FILE  *filePtr;
```

where `filePtr` is simply a variable name of your choosing.

- NOTE: All the statement above does is declares the file pointer…it does nothing else!

How To Initialize A File Pointer

- In order to properly "initialize" a file pointer, it must be set to "point" to a particular file.
- To do this you must specify two things: (1) the name of the file, and (2) the mode in which the file is to be opened.
  - (1) is either a relative or absolute file name. A relative file name is in the current directory and an absolute file name contains the full pathname to the file.
  - (2) The mode is typically one of `r,` `w`, or `a`. `"r"` opens the file for reading only. `"w"` opens the file for writing. `"a"` opens the file for appending (writing to the end of the file). We'll see more modes later.
- This is done using the `fopen` function whose general format is: `fopen("`*`filename`*`", "`*`mode`*`");`

How To Initialize A File Pointer

Example:

```
#include <stdio.h>
int main()
{ FILE firstInputFilePtr;
  FILE secondInputFilePtr;
  FILE outputFilePtr;

  firstInputFilePtr = fopen("mydata.dat", "r");
  secondInputFilePtr = fopen("C:/data/myfile.txt", "r");
  outputFilePtr = fopen("myoutput.dat", "w");

  . . .
```

Initializes the file pointer named `firstInputFilePtr` to the file named `"mydata.dat"` in the current working directory. File is opened for reading.

Initializes the file pointer named `secondInputFilePtr` to the file named `"myfile.txt"` in the directory whose path is `"C:/data/"`. File is opened for reading.

Initializes the file pointer named `outputFilePtr` to the file named `"myoutput.dat"` in the current working directory. File is opened for writing.

How To Initialize A File Pointer

COMMON PROGRAMMING ERROR

Windows programmers beware when using absolute file names in a call to `fopen`. Standard convention is to use backslashes in file pathnames. However, consider the following case:

```
     fopen ("C:\data\testfile.txt", "r");
```

In C the `\t` in the string will be interpreted as a character escape corresponding to a tab.

The way around this is to use the forward slash as separators in file path names. In this case, the correct version of the `fopen` call shown above would be:

```
        fopen ("C:/data/testfile.txt", "r");
```

How To Initialize A File Pointer

GOOD PROGRAMMING PRACTICE

Whenever you attempt to open a file, by initializing the file pointer to that file, C returns the pointer if the call to `fopen` is successful and returns the value NULL if the call is not successful. Therefore, it is always a good practice to test the value returned by the call to `fopen`. This is commonly done as follows:

```
if ( (filePtr  = fopen(“filename”, “mode”) ) == NULL ) {
   printf(“File could not be opened\n”);
}
else {
   . . .
}
```

Note that if the mode of the file is set to “r”, then the file must already exist before it can be opened. If the file mode is either “w” or “a”, then it does not need to exist before it can be opened (the file will be created dynamically). However, if the mode is “w” and the file pre-exists the `fopen`, the previous contents of the file are lost.

How To Close A File

- To close a file use the function `fclose` which takes a single argument which is a pointer to a file as in:

```
fclose ( filePointer );
```

- Typically you should close a file as soon as the program is finished using the file.

GOOD PROGRAMMING PRACTICE

Most operating systems will close any open files that are associated with an application whenever the application terminates even if the application did not explicitly close the files. However, it is good programming practice to always explicitly close any open files used by the program before the program terminates. Best practice is to close a file as soon as it is known that the program will not reference the file again. This will free up resources both for your program and others that may be running on the same system.

How To Write To A File

- You are already quite familiar with the `printf` function that we have been using to write the output of our programs to the `stdout` stream. Recall that the format for a call to the `printf` function has this form:

```
printf(“control string”, other arguments);
```

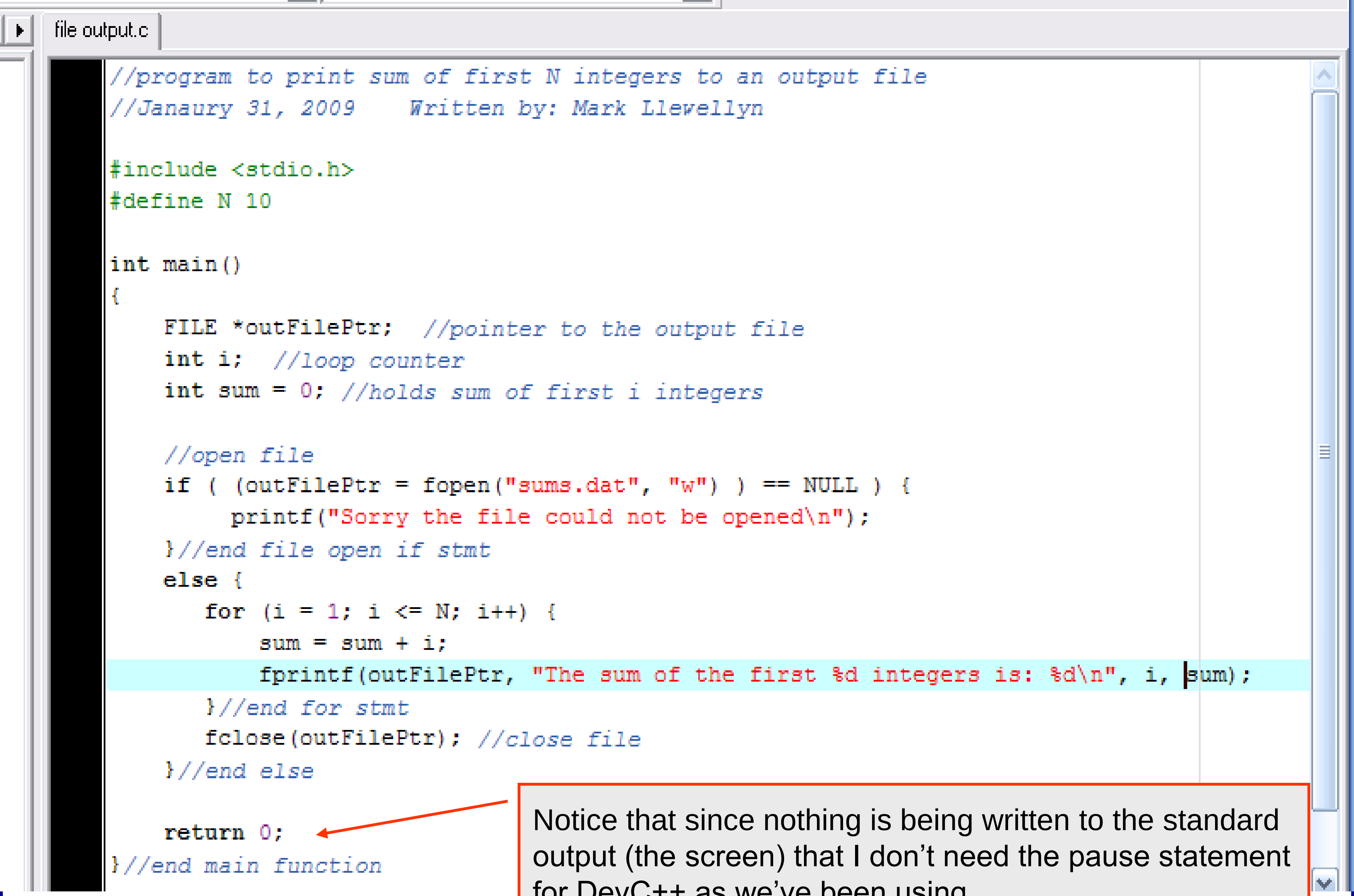
- To write to any file (stream) other than stdout; use the `fprintf` function which has the following form:

```
fprintf( fileptr, “control string”, other arguments);
```

- Notice that the format of a call to `fprintf` is essentially the same as that of `printf`, with the exception that `fprintf` requires a pointer to the file to which you are writing. The `printf` function did not require this argument because it defaults to the `stdout` stream.

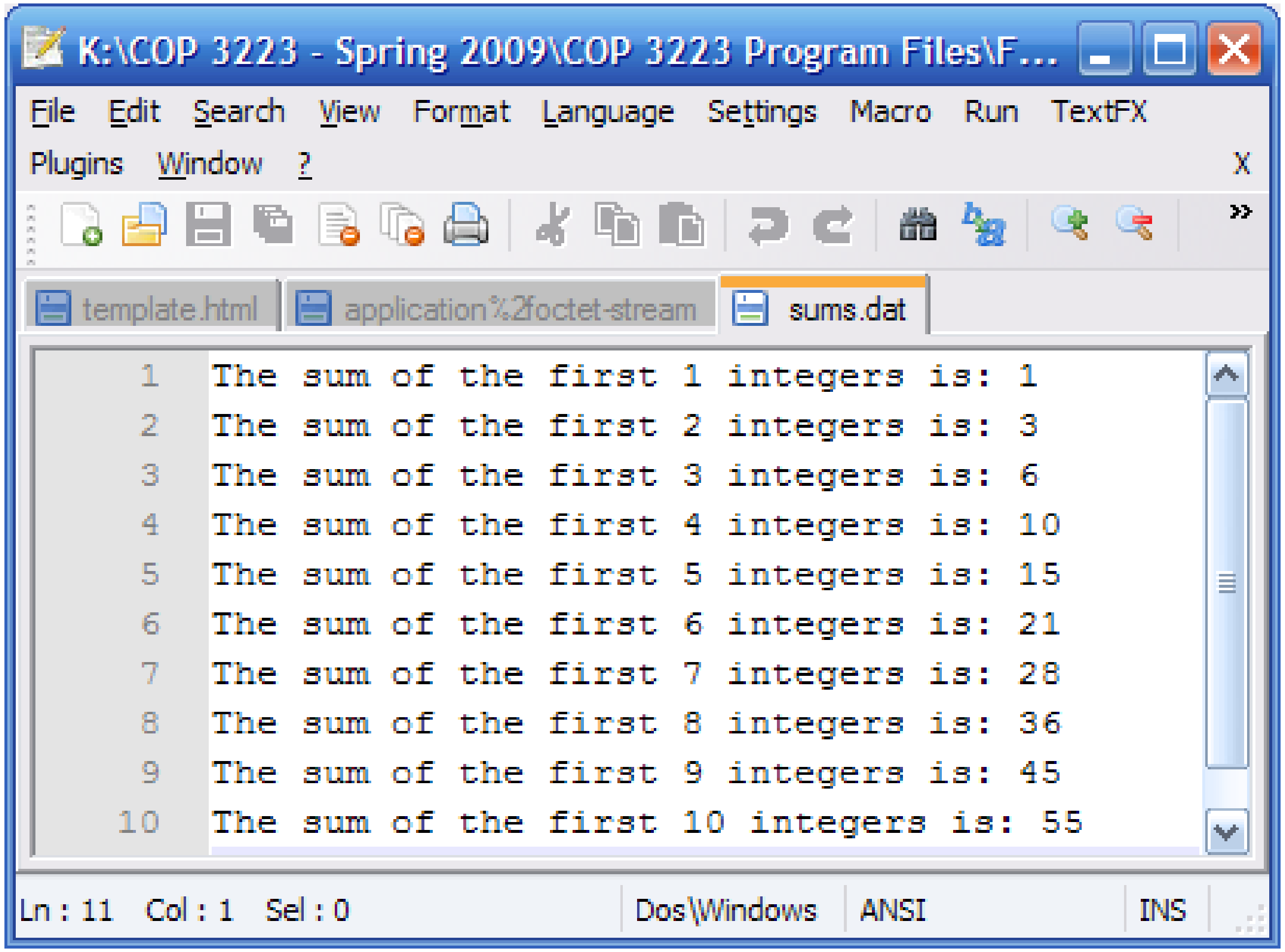
Putting Everything Together So Far

- Let’s return to one of our standard examples of printing the sum for each of the first 10 integers.
- We written this program several different ways so far using different types of repetition structures, but always the output has been to the terminal screen.
- This time, we’ll create an output file and send our output to that file rather than to the screen.
- The program is on the next page.


file output.c

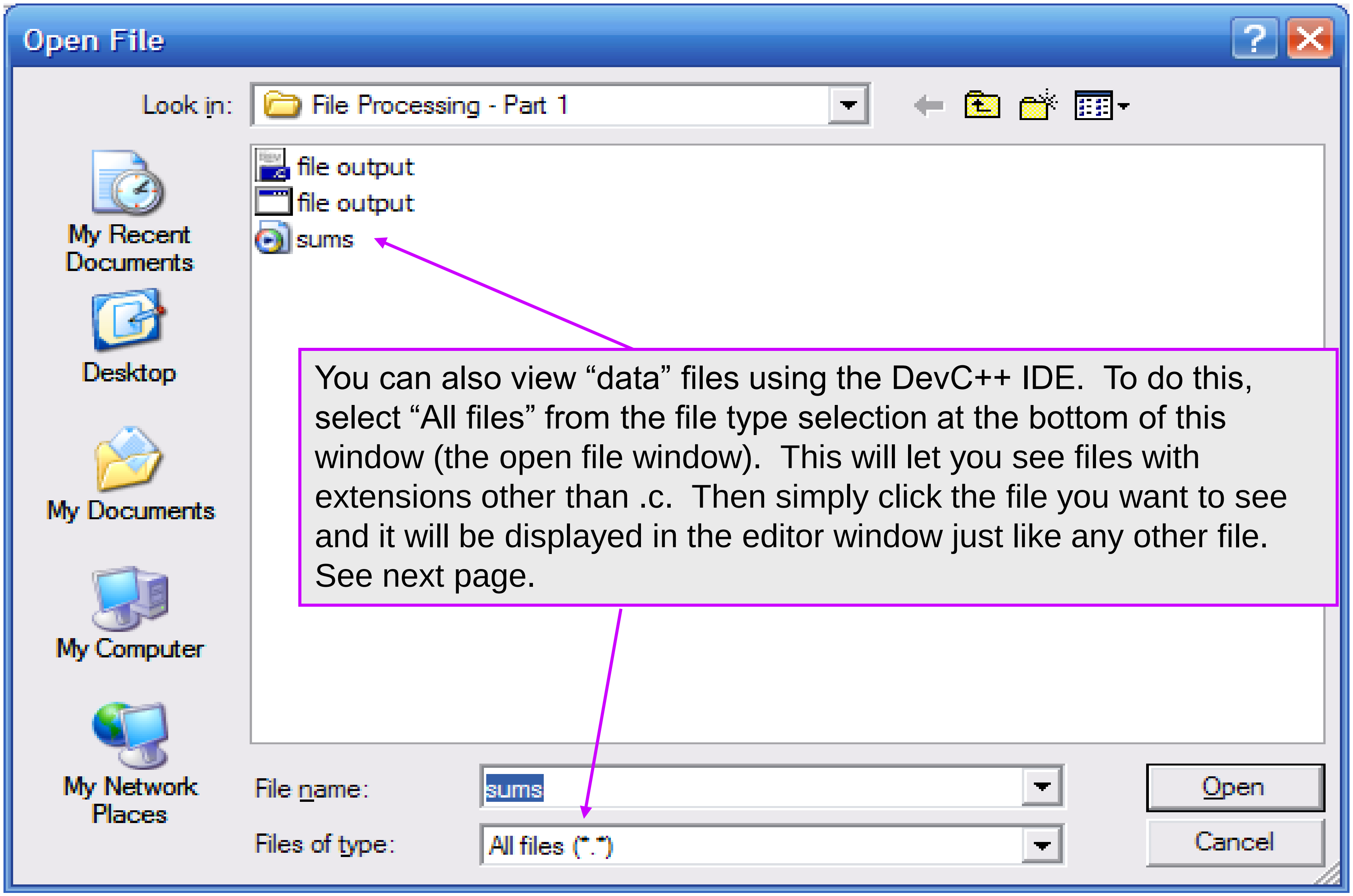
```c
//program to print sum of first N integers to an output file
//Janaury 31, 2009    Written by: Mark Llewellyn

#include <stdio.h>
#define N 10

int main()
{
    FILE *outFilePtr;  //pointer to the output file
    int i;  //loop counter
    int sum = 0; //holds sum of first i integers

    //open file
    if ( (outFilePtr = fopen("sums.dat", "w") ) == NULL ) {
         printf("Sorry the file could not be opened\n");
    }//end file open if stmt
    else {
       for (i = 1; i <= N; i++) {
           sum = sum + i;
           fprintf(outFilePtr, "The sum of the first %d integers is: %d\n", i, sum);
       }//end for stmt
       fclose(outFilePtr); //close file
    }//end else

    return 0;
}//end main function
```

Notice that since nothing is being written to the standard output (the screen) that I don't need the pause statement for DevC++ as we've been using.


K:\COP 3223 - Spring 2009\COP 3223 Program Files\F...
File Edit Search View Format Language Settings Macro Run TextFX
Plugins Window ?
template.html
application%2foctet-stream
sums.dat
Ln : 11 Col : 1 Sel : 0
Dos\Windows
ANSI
INS

```
The sum of the first 1 integers is: 1
The sum of the first 2 integers is: 3
The sum of the first 3 integers is: 6
The sum of the first 4 integers is: 10
The sum of the first 5 integers is: 15
The sum of the first 6 integers is: 21
The sum of the first 7 integers is: 28
The sum of the first 8 integers is: 36
The sum of the first 9 integers is: 45
The sum of the first 10 integers is: 55
```

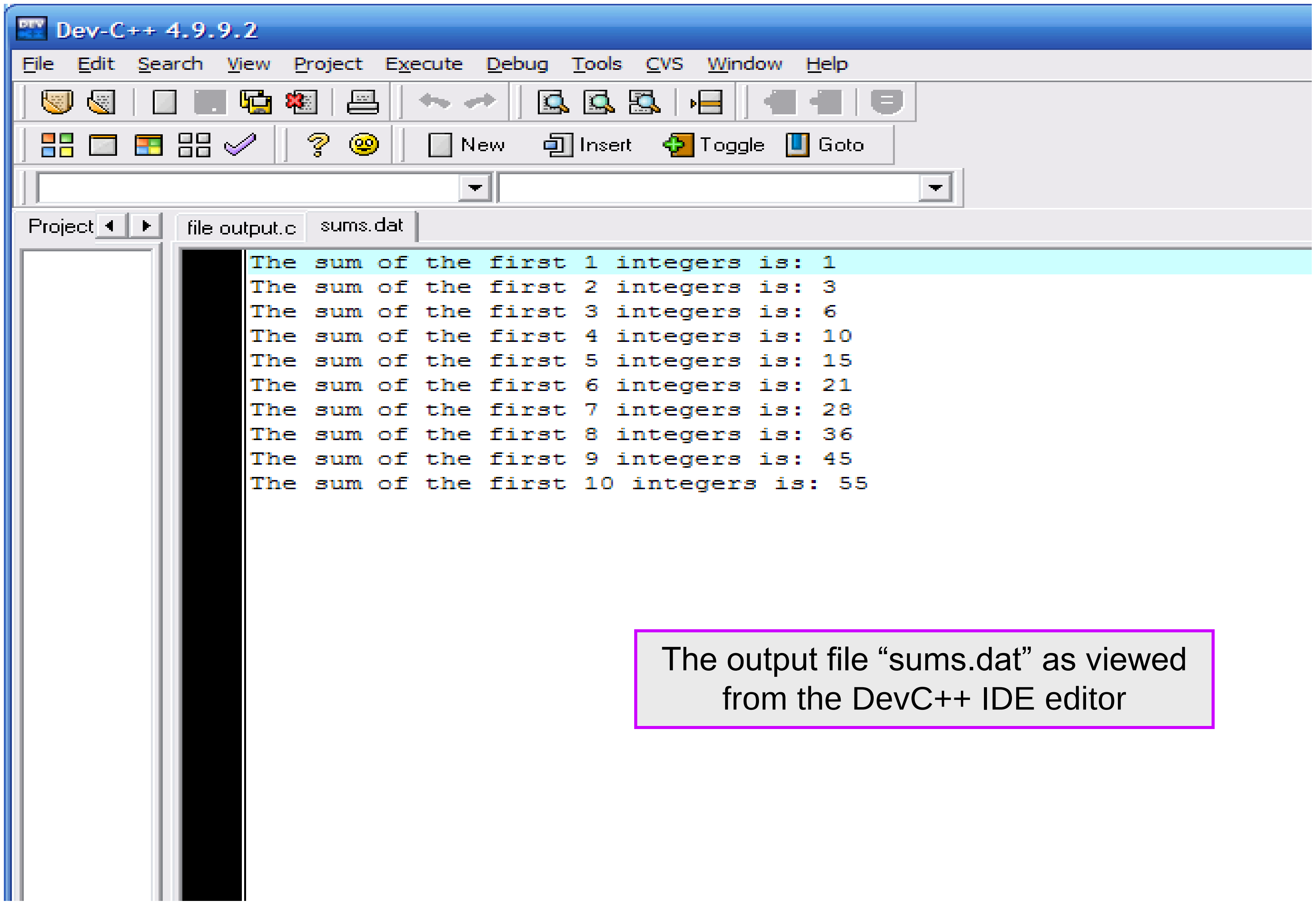
The output file named “sums.dat” as viewed using the Notepad++ editor.



File name: sums

Files of type: All files (*.*)

You can also view “data” files using the DevC++ IDE. To do this, select “All files” from the file type selection at the bottom of this window (the open file window). This will let you see files with extensions other than .c. Then simply click the file you want to see and it will be displayed in the editor window just like any other file. See next page.


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project
file output.c
sums.dat

```
The sum of the first 1 integers is: 1
The sum of the first 2 integers is: 3
The sum of the first 3 integers is: 6
The sum of the first 4 integers is: 10
The sum of the first 5 integers is: 15
The sum of the first 6 integers is: 21
The sum of the first 7 integers is: 28
The sum of the first 8 integers is: 36
The sum of the first 9 integers is: 45
The sum of the first 10 integers is: 55
```

The output file “sums.dat” as viewed from the DevC++ IDE editor

How To Read From A File

- Reading from user defined files operates basically the same as reading from the `stdin` stream. Recall that the format for a call to the `scanf` function has this form:

```
scanf(“control string”, other arguments);
```

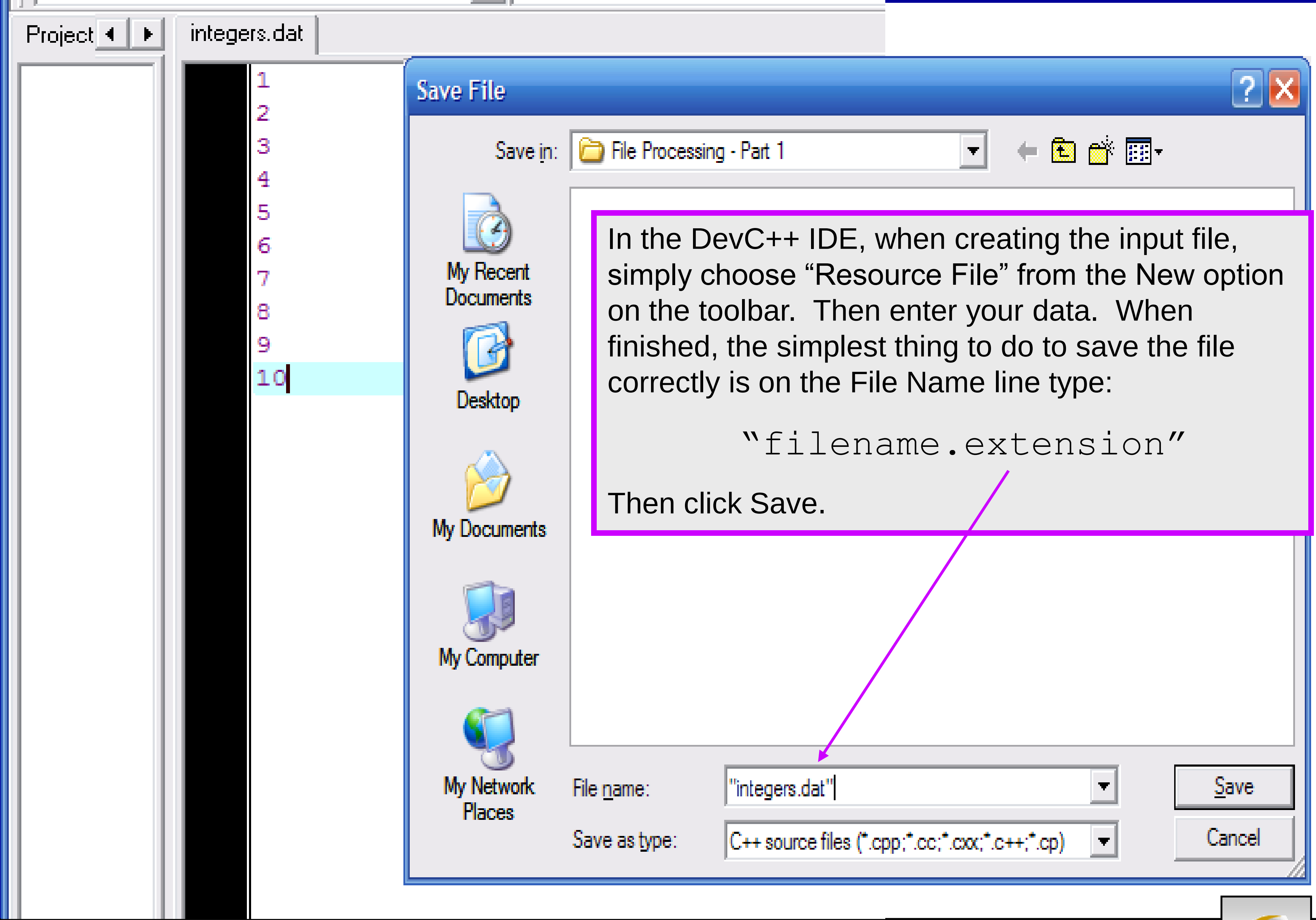
- To read from any file (stream) other than stdin; use the `fscanf` function which has the following form:

```
fscanf( fileptr, “control string”, other arguments);
```

- Notice that the format of a call to `fscanf` is essentially the same as that of `scanf`, with the exception that `fscanf` requires a pointer to the file from which you are reading. The `scanf` function did not require this argument because it defaults to the `stdin` stream.

How To Read From A File

- To see a simple example program that reads from a file, let's once again redo our sum of integers program.
- This time we'll read the integer values that are to be summed from a file.
- First, we need to create this file. You can use any text editor that you choose to create data files. I'll just use DevC++ for this example, and you'll probably typically do the same since you've already got this environment loaded while you're working on your C programs.
- Let's create the file with one integer on each line of the file and name the file "`integers.dat`".



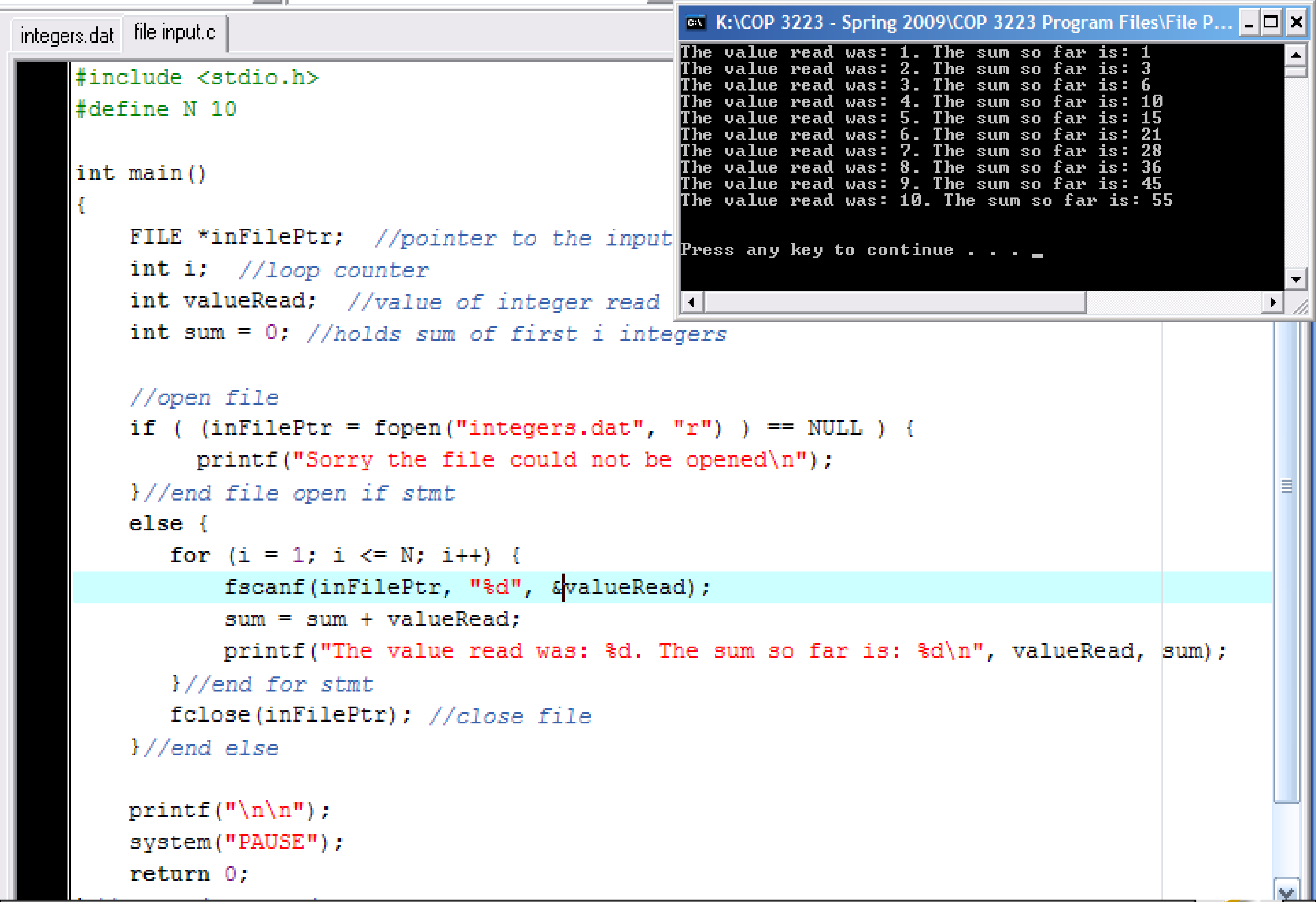
File name: "integers.dat"

Save as type: C++ source files (*.cpp;*.cc;*.cxx;*.c++;*.cp)

In the DevC++ IDE, when creating the input file, simply choose “Resource File” from the New option on the toolbar. Then enter your data. When finished, the simplest thing to do to save the file correctly is on the File Name line type:

```
“filename.extension”
```

Then click Save.

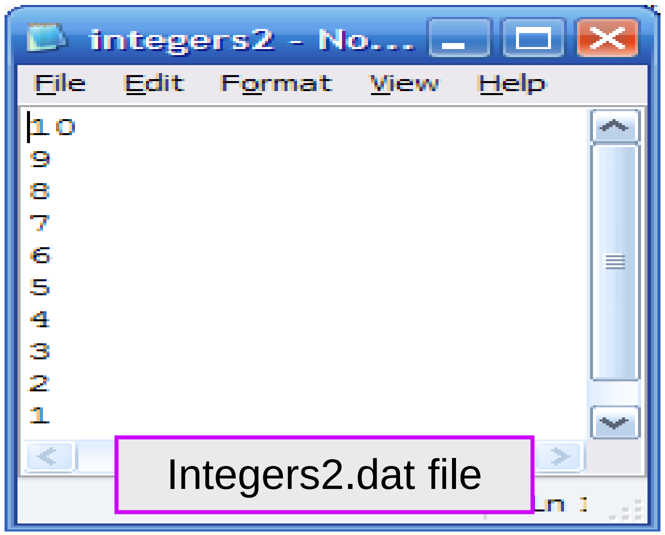


```c
#include <stdio.h>
#define N 10

int main()
{
    FILE *inFilePtr;   //pointer to the input
    int i;   //loop counter
    int valueRead;   //value of integer read
    int sum = 0; //holds sum of first i integers

    //open file
    if ( (inFilePtr = fopen("integers.dat", "r") ) == NULL ) {
         printf("Sorry the file could not be opened\n");
    }//end file open if stmt
    else {
       for (i = 1; i <= N; i++) {
           fscanf(inFilePtr, "%d", &valueRead);
           sum = sum + valueRead;
           printf("The value read was: %d. The sum so far is: %d\n", valueRead, sum);
       }//end for stmt
       fclose(inFilePtr); //close file
    }//end else

    printf("\n\n");
    system("PAUSE");
    return 0;
```

```
The value read was: 1. The sum so far is: 1
The value read was: 2. The sum so far is: 3
The value read was: 3. The sum so far is: 6
The value read was: 4. The sum so far is: 10
The value read was: 5. The sum so far is: 15
The value read was: 6. The sum so far is: 21
The value read was: 7. The sum so far is: 28
The value read was: 8. The sum so far is: 36
The value read was: 9. The sum so far is: 45
The value read was: 10. The sum so far is: 55


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

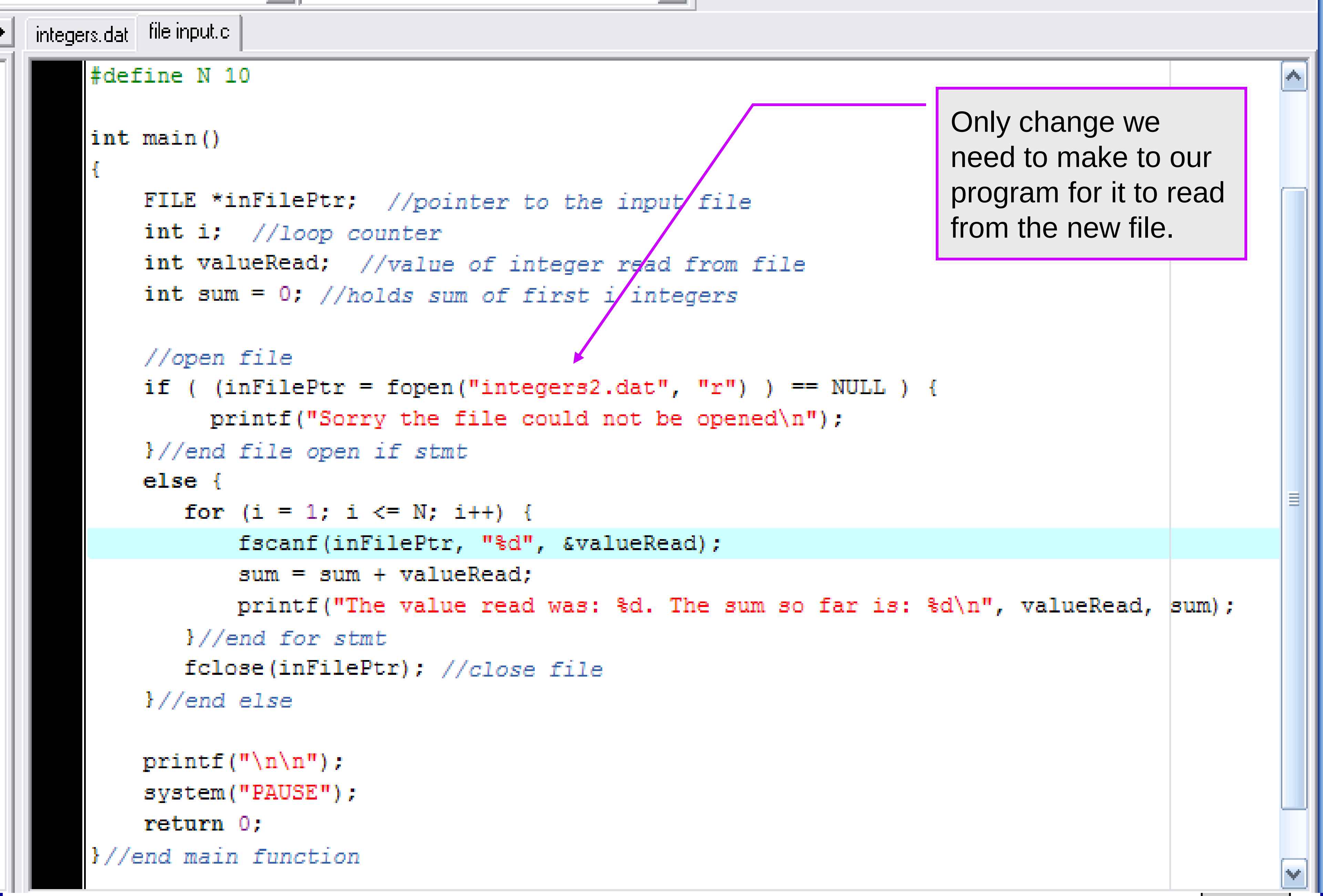
How To Read From A File

- Let's create a second input file named "integers2.dat", where the integer numbers from 1 to 10 are placed in the file in descending order.
- Once the input file is created, we'll modify our program to read from this new file.



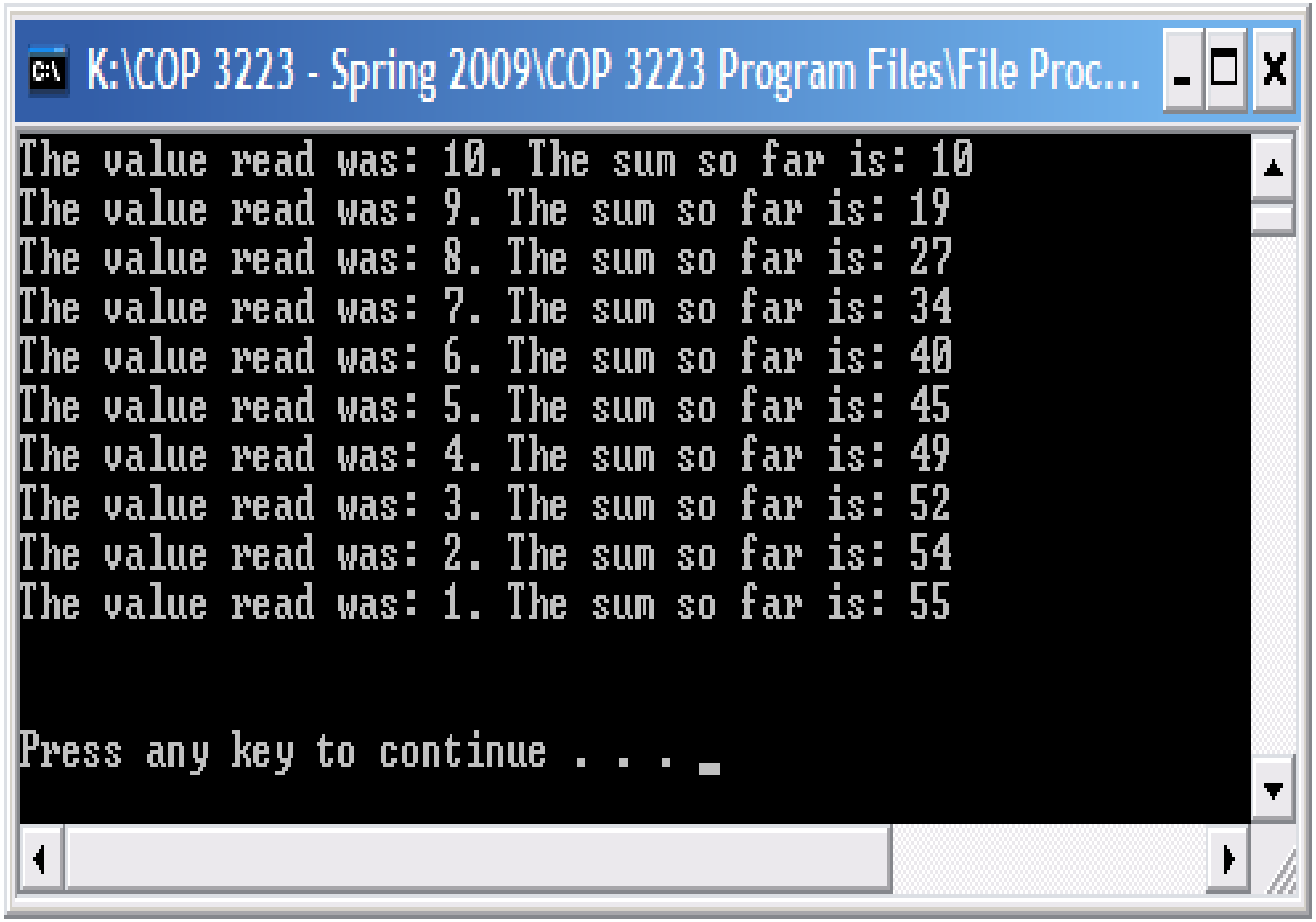
```
10
9
8
7
6
5
4
3
2
1
```

Integers2.dat file



```
#define N 10

int main()
{
    FILE *inFilePtr;  //pointer to the input file
    int i;  //loop counter
    int valueRead;  //value of integer read from file
    int sum = 0; //holds sum of first i integers

    //open file
    if ( (inFilePtr = fopen("integers2.dat", "r") ) == NULL ) {
        printf("Sorry the file could not be opened\n");
    }//end file open if stmt
    else {
       for (i = 1; i <= N; i++) {
           fscanf(inFilePtr, "%d", &valueRead);
           sum = sum + valueRead;
           printf("The value read was: %d. The sum so far is: %d\n", valueRead, sum);
       }//end for stmt
       fclose(inFilePtr); //close file
    }//end else

    printf("\n\n");
    system("PAUSE");
    return 0;
}//end main function
```

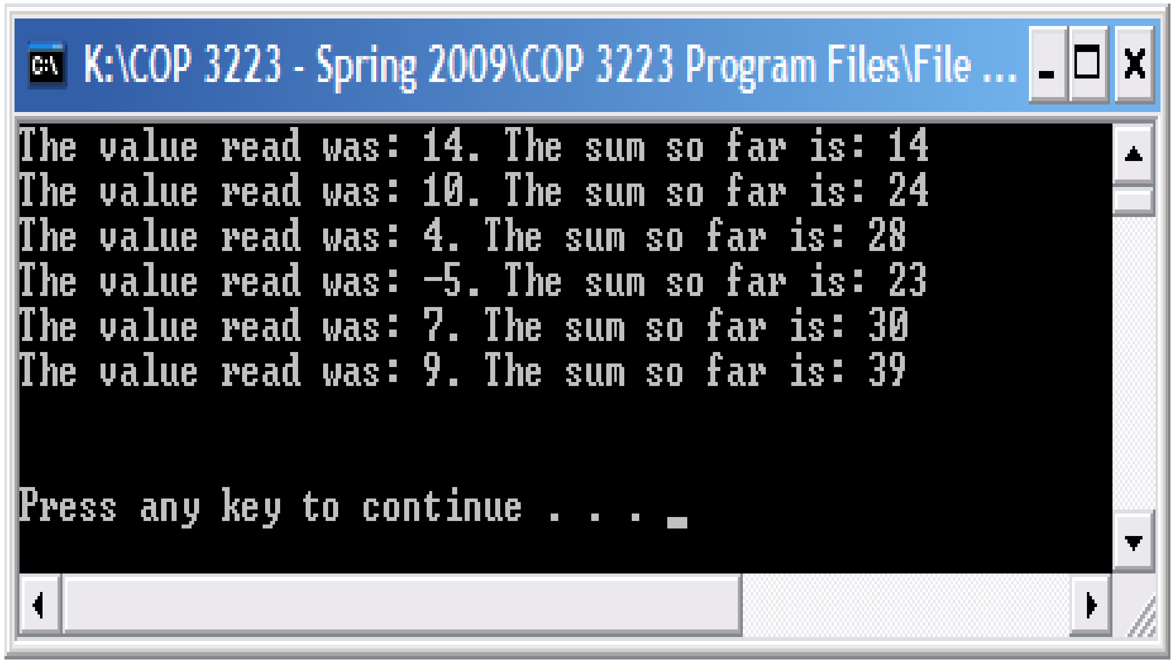
Only change we need to make to our program for it to read from the new file.


K:\COP 3223 - Spring 2009\COP 3223 Program Files\File Proc...

```
The value read was: 10. The sum so far is: 10
The value read was: 9. The sum so far is: 19
The value read was: 8. The sum so far is: 27
The value read was: 7. The sum so far is: 34
The value read was: 6. The sum so far is: 40
The value read was: 5. The sum so far is: 45
The value read was: 4. The sum so far is: 49
The value read was: 3. The sum so far is: 52
The value read was: 2. The sum so far is: 54
The value read was: 1. The sum so far is: 55



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

Practice Problems

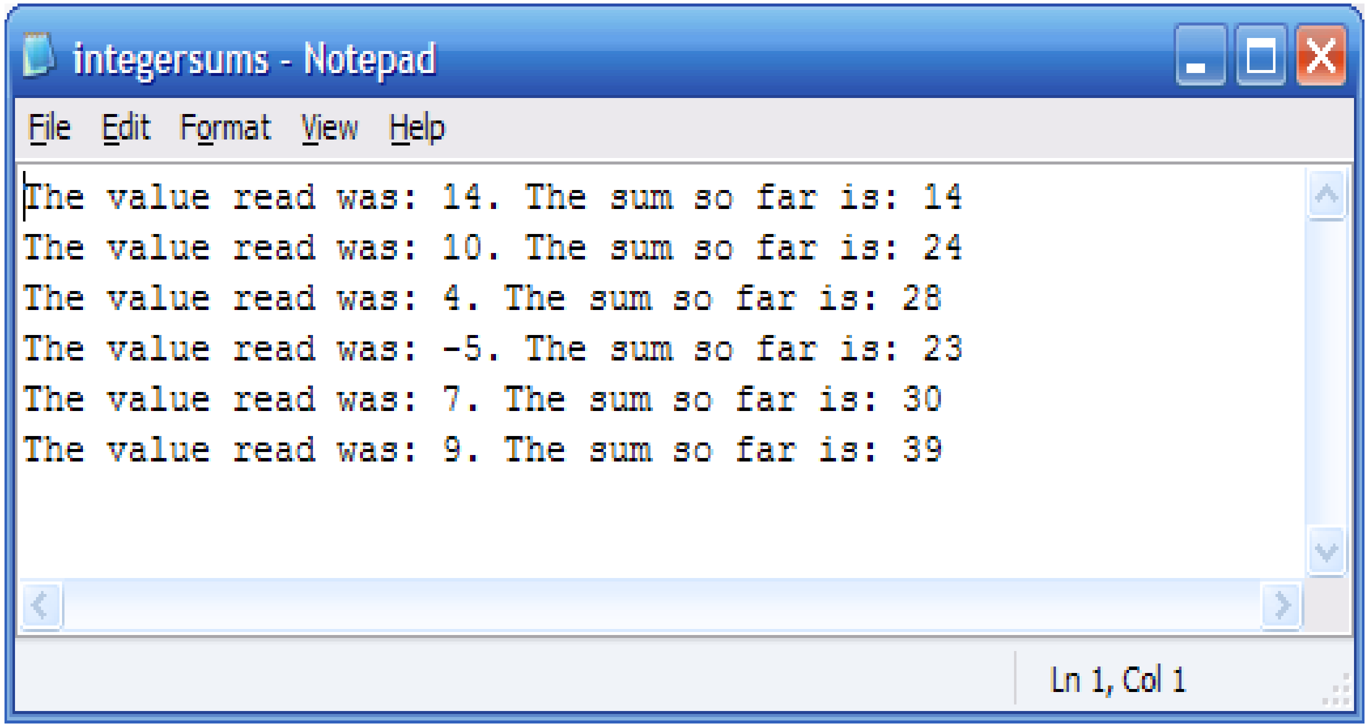
1. Modify the program on page 20 so that the first line of the file the program reads from contains an integer that is the number of integers that will be read from the file. Example: 3 5 6 7, tells the program to read 3 integers from the file and those numbers are 5, 6, and 7 in this case. As before, the program is to produce the running sum of the numbers read from the file.



```
The value read was: 14. The sum so far is: 14
The value read was: 10. The sum so far is: 24
The value read was: 4. The sum so far is: 28
The value read was: -5. The sum so far is: 23
The value read was: 7. The sum so far is: 30
The value read was: 9. The sum so far is: 39


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

Practice Problems

2. Modify the program you wrote for Practice Problem 1 so that the output generated by the program is written to an output file named "`integersums.dat`".



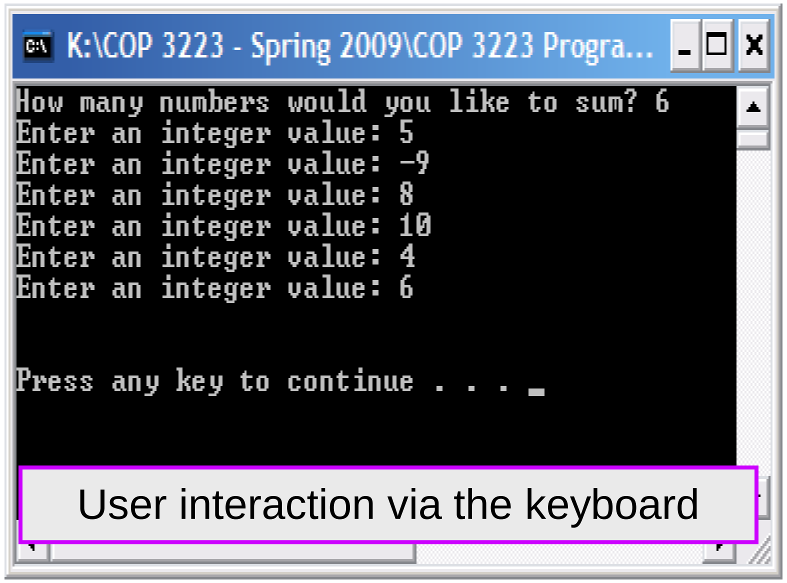
```
The value read was: 14. The sum so far is: 14
The value read was: 10. The sum so far is: 24
The value read was: 4. The sum so far is: 28
The value read was: -5. The sum so far is: 23
The value read was: 7. The sum so far is: 30
The value read was: 9. The sum so far is: 39
```

Practice Problems

3. Construct a C program that in the first part of the program asks the user how many integers they would like to enter and then read in that number of integers and write the integers read in to an output file called `“entered numbers.dat”`. Then the second part of the program reads the `“entered numbers.dat”` file and produces a running sum of the numbers in that file and writes the output to a file named `“pracprob3.dat”`.

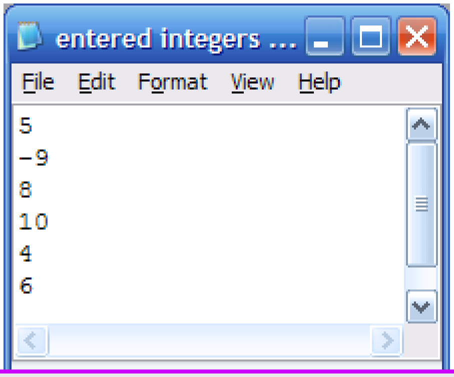
   See the next page screen shots of the various files and screens the user should see or generate.

Practice Problems



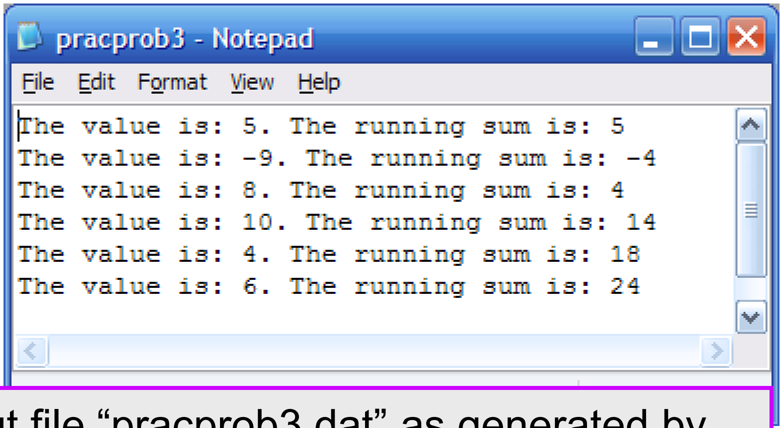
```
How many numbers would you like to sum? 6
Enter an integer value: 5
Enter an integer value: -9
Enter an integer value: 8
Enter an integer value: 10
Enter an integer value: 4
Enter an integer value: 6


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

User interaction via the keyboard



```
5
-9
8
10
4
6
```

The "`entered integers.dat`" file created by the first part of the program



```
The value is: 5. The running sum is: 5
The value is: -9. The running sum is: -4
The value is: 8. The running sum is: 4
The value is: 10. The running sum is: 14
The value is: 4. The running sum is: 18
The value is: 6. The running sum is: 24
```

The output file "pracprob3.dat" as generated by the second part of the program