Chapter 8 – Files

This chapter about files is pretty useful. Sometimes when you do calculations, you may want to write the information to your C: drive or a floppy disk. You can write c++ code to do this.

To be able to use certain keywords we have to include the header file fstream.h

Here is a simple example that writes some numbers to a file.

```
1  #include<iostream.h>
2  #include<fstream.h>
3
4  int main()
5  {
6  fstream myfile;
7  int i;
8
9  myfile.open("c:\\test.txt",ios::out);
10
11 for(i=0; i< 10; i++)
12 {
13 myfile << i << endl;
14 }
15
16 myfile.close();
17
18 return 0;
19 }
```

Example 8a

Check your c drive for this file!

We have to declare a variable of type fstream to represent the file. You can call it whatever you like. Here we called it myfile.

The line myfile.open("c:\test.txt",ios::out); tells the computer that we are writing data **OUT** to a file and not reading in data.

"myfile << i << endl;" is similar to "cout << i << endl;". However nothing is printed on the screen, like when cout is used. Instead the data is printed to a file. When we are finished writing to the file, we close it with "myfile.close();"

Note: When you are writing to sub directories or folders you have to use TWO backslashes.

Now let's read in the data we have just written to the file. We will use an array to read in and store the data.

```
1  #include<iostream.h>
2  #include <fstream.h>
3
4  int main()
5  {
6  fstream myfile;
7  int i,arr[10];
8
9  myfile.open("c:\\test.txt",ios::in);
10
11 for(i=0; i<10; i++)
12 {
13 myfile >> arr[i];
14 cout << arr[i] << " ";
15 }
16
17 myfile.close();
18
19 return 0;
20 }
```

Example 8b

This program will read the contents of the file we created in the first example.
Take a look at the line "myfile.open("c:\\test.txt",ios::in);". This time there is an *in* instead of an *out*, as we are reading data into our program.

Now look at "myfile >> arr[i];". Notice that the arrows are pointing inwards similar to "cin".

Suppose we wanted to add more data to our file. We could use "myfile.open("c:\\test.txt",ios::out);" again but what this will do is overwrite the data that is already there. What we need to do is append or add the data on. We can do this by using the line "myfile.open("c:\\test.txt",ios::app);"
Notice that the "out" is replaced by "app".

If we know how many lines of data are in a file we can use a “for loop” to read in the data as in the previous programs. But often we do not know in advance how many lines there are. In that case we need a way of detecting when we have reached the end of the file.
The following program uses the function eof() which detects the end of the file.

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  int main()
5  {
6  fstream myfile;
7  int i;
8
9  myfile.open("c:\\test.txt",ios::in);
10
11 myfile >> i;
12
13 while(myfile.eof() == 0)
14 {
15 cout << i << " ";
16 myfile >>i;
17 }
18
19 myfile.close();
20
21 return 0;
22 }
```

Example 8c

The function eof() returns 1 if the end of file is reached, otherwise it returns a 0. This is why we write code to read in data “while(myfile.eof()==0)”.

Open notepad and make a list of 10 names (one on each line) and save it as names.txt on your c drive. The next program will read in the list of names and print them on the screen.

The names can be 14 characters long leaving one space for the end of string indicator, '\0'.

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  int main()
5  {
6  fstream myfile;
7  int i;
8  char names[10][15];
9
10 myfile.open("c:\\names.txt",ios::in);
11
12 for(i=0; i<10; i++)
13 {
14 myfile >>names[i];
15 cout << names[i] << endl;
16 }
17
18 myfile.close();
19
20 return 0;
21 }
```

Example 8d

Here we use a two-dimensional array to store the names, 10 names down by up to 14 letters across.