File Input and Output

There is often a need to access data that is contained in separate files from the code files. This information can be data values, such as a list of names for high scores in a game, operational parameters for the program, temporary data while a program is executed etc. The number of ways to make use of files is limited only by what the programmer would like to do. Data access is such a key aspect of computer programming that there are many tools designed specifically for this purpose. For example, the Grade 12s will use an open source program called mySQL for web apps which need access to live data.

Our work will continue expanding our knowledge of C programming, and there are many different capabilities within C for file operations.

The first thing that is needed is a way for C to get access to the file that is being used. As with our Level Four functions, this is done with a pointer, or specifically, a file pointer, which is declared in a similar manner to other variables.

```
FILE        *fp;            //The type is 'FILE' and fp is 'file pointer'
```

To access a file, it must be opened using the fopen() command. The basic syntax of this command is

```
fp = fopen(const char *filename, const char *mode);
```

Although this looks a bit confusing, essentially it says that to access a file, you need a file name and a mode of opening the file. The mode refers to whether or not the file is to be read, written, added to etc. A list of modes to open files is below:

r	- open for reading
w	- open for writing (file need not exist)
a	- open for appending (file need not exist)
r+	- open for reading and writing, start at beginning
w+	- open for reading and writing (overwrite file)
a+	- open for reading and writing (append if file exists)

Once the access to the file is no longer needed, close the file with the fclose() function.

```
int fclose(FILE *a_file);       //fclose returns zero if the file is closed successfully
```

Again, the use is simpler than the prototype makes it look: fclose(fp);

To write or read data to or from the file, the simplest commands are very similar to those you have already used, and are the fscanf, fprintf, fgetc and fgets. Obviously they have to be a bit different, and the only difference is that you need to include a file to point to for each command.

There is a sample program below to start you exploring file operations. There are various other aspects to file operations which we will not cover; feel free to investigate these on your own.

```
/*
Demonstration program for the use of files in ICS3U

To run this program, a file called data.txt must be located in the same directory
as the code file and must contain a list of integers separated by spaces or an end
of line
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{

	FILE *fp; /*This is the pointer to the text file that will be used*/
	char fileName[20] = "data.txt"; /*The name of the file.. user entered??*/
	int i,theNum;

	printf ("File Manipulation Demonstration!\n\n");

	fp = fopen(fileName,"r+");		/*Open the file to read or write*/

	if(fp == NULL)			/*Don't crash if the file is not found*/
	{
		printf("\n\nError opening file. Program exiting.");
	}
	else
	{
		printf("File %s successfully opened.\n\n",fileName);
		i=0;		/*i will be a counter for the number of numbers*/

		while(!feof(fp)) /*Input data from the file, and echo to the console*/
		{
			fscanf(fp,"%i",&theNum);	/*Read an integer from the file*/
			printf("\n%i",theNum);		/*and echo it to the console*/
			i++;				/*Just counting the files*/
		}

		printf("\nThere were %i files entered from the file.\n",i);
	}

	fclose(fp);				/*Close the file access*/
	return 0;
}
```

Assignment

1. Write a properly structured C program which opens a file called numbers.txt, which contains a list of integer values. This file should contain no more than 100 values. Have the program read the file, count the number of values found, and output this value along with the smallest, largest and average of all the values.

2. Write a properly structured C program which opens a file called names.txt, which contains a list of first and last names for students in a class, stored one per line as first name and then last name. The first value in the file will be an integer identifying how many names are in the file. Have the program read in all of the names and then output them sorted alphabetically by last name.

3. Write a properly structured C program which opens a file called classMarks.txt which contains final marks for a class of up to 30 students. The first line in the file will contain the course code, the second line will contain the number of students, and each subsequent line will contain the lastname, firstname and then final mark for each student. Have the program output the data sorted either by marks from high to low or alphabetically by last name.

4. Adapt your programs from 3 and 4 to allow the sorted data to be written back to the file.