261: 22nd November
C Programming

So far, the programs that we have written have not done anything too flash. They have multiplied matrices and vectors or calculated determinents, which is stuff that we could do in a fraction of the time with a pen and paper.

However, the reason that we wrote them was as an exercise in turning mathematical techniques that we take for granted, into computer algorithms. This can be much harder than it might seem, at first.

One of the reasons that our matrix multiplication program isn't very useful is that the values of the matrix and the vectors are written into the program code, so we can only use it specifically to calculate

$$\begin{pmatrix} 1.2 & 2.3 & 3.4 & 4.3 \\ 4.1 & 5.0 & 6.0 & 3.4 \\ 7.1 & 8.1 & 9.9 & 5.7 \\ 3.2 & 2.6 & 2.6 & 2.8 \end{pmatrix} \begin{pmatrix} 1.5 \\ 4.3 \\ 2.8 \\ 3.6 \end{pmatrix}$$

We need to extend our programming skills so that we can write more flexible code. We must look at ways of creating a program that can multiply any matrix and vector combination. We can do this by introducing file-handling.

Files

We want to be able to modify our matrix-vector mulitplication program, so that it opens two files and retrieves the values for the matrix and vector from them. When the program has calculated the product, it will then write the value of the new vector to a third file.

Declaration

Firstly, we will need three variables to label the files. These variables will be useful when we need to distinguish which file we want to read from and which we want to write to.

In the declaration section of `main`, ie the section where we declare our `int`, `float` and `doubles`, we must declare three more variables, `mfile`, `yfile` and `xfile`.

```
FILE *mfile;
FILE *xfile;
FILE *yfile;
```

Unfortunately, it's not as simple as these three having the `FILE` type. The star means that the variable contains a 'pointer', which is like a grid reference, and tells us where, in the computer, the file can be found. However, this shouldn't inconvenience us. In the commands that we will use, we can treat the variables as if they represented each file directly.

Opening Files

Before we can read from a file or write to it, we have to open it. To do this, we use the `fopen` command. Let's say that our matrix is in a file called `matrixfile` and that our x-vector is in a file called `xvectfile`, then the following commands will open these files and create or open a third file, `yvectfile`, depending on whether or not it already exists.

```
mfile=fopen(``matrixfile'', ``r'');
xfile=fopen(``xvectfile'', ``r'');
yfile=fopen(``yvectfile'', ``w'');
```

When we open a file, we have to state how it's going to be used. The `` `r' `` means that we will only read from the file and the `` `w' `` that we will only write to it. If we try to read from a file opened with `` `w' `` or write to a file opened with `` `r' ``, then the program will throw up an error message.

Usually we only need to open each file once, so the `fopen` commands usually appear at the start of the `main` routine, after the declarations.

Reading From Files

To get data from the files we use the `fscanf` command. This reads a combination of variables from a file in much that same way that `printf` prints a combination to the screen. To retrieve an integer from the matrix file and give its value to a variable called `a11`, we would use the command

```
fscanf(mfile, ``%d'', &a11);
```

To read a `float`, we would replace `%d` with `%f` and to read a `double`, we would use `%lf`.

Writing to Files

Writing is achieved with the `fprintf` command. Again, we need to include `yfile` in the command, so it is clear which file we mean to write to. Apart from that, the form is very similar to the `printf` command.

To write the double, `y1`, to the file represented by `yfile`, we would use the command

```
fprintf(yfile, ``%d'', y1);
```

Closing Files

Finally, before the program ends, we should close all the files. This lets the computer tidy up any internal shuffling that it had to do to access them. The command to close `mfile` would be

```
fclose(mfile);
```

As with the `fopen` command, `fclose` is usually only used once for each file. Typically, `fclose` commands are placed at the end of the `main` routine, before the `return` line.

File Format

To read a file using these commands, the file should comprise just a list of numbers. So, for example, to represent a $3 * 3$ matrix, we would just need a file with 9 entries, one at the beginning of each line.

Question

Write a program that reads in a $4 * 4$ upper diagonal matrix, A, and a $4 * 1$ vector, y and calculates x, where $Mx = y$.