Subject #8: More about Input and Output

- Create a file named **numbers.dat** containing about a dozen real numbers, positive and negative, in separate lines. Write a word (not a number!) in some of the lines.
- Write, compile and run the following program

```
#include <stdio.h>

main()
{
    char infile[40] = "numbers.dat" ;
    char garbage[100] ;
    FILE *infp, *outfp ;
    int n ;
    float val ;
    int rc ;

    infp  = fopen(infile, "r") ;
    outfp = fopen("output.dat", "w") ;
    if (infp == NULL  ||  outfp == NULL) {
        fprintf(stderr, "Problem opening file\n") ;
        exit(1) ;
    }

    for (n = 1 ; (rc = fscanf(infp, "%f", &val)) != EOF ; n++)
        if (rc == 1)      /* A good        line */
            fprintf(outfp, "%3d \t %+7.3f \n", n, val) ;
        else {            /* A problematic line */
            fprintf(stderr, "Problem at line %d\n", n) ;
            fscanf(infp, "%s", garbage) ;
        }

    fclose(infp ) ;
    fclose(outfp) ;
}
```

- Look at the file **output.dat** and verify that you understand the above program.
- The type representing a character ("letter") in C is *char*. A *string* (a "word" or a "line") is composed of characters and is not a basic type. There can be *string variables* (like **infile** in our program) and string constants (between double quotes, like `"output.dat"` in our program).
- The **fopen** command returns a *file pointer*, which is of type **FILE *** . If it fails to open the file, it returns **NULL** (which is actually 0). The first argument of **fopen** is the name of the file (a string), and its second one is the mode. `"r"` is read, `"w"` is write, `"a"` is append, `"r+"` is read and write.
- The **exit** command terminates the execution of the program. It should be given an integer argument, which is the exit status of the program. Usually, a zero value signals that all is well, while non-zero values signify various problems.

- The **fscanf** command is the file version of **scanf**. It has an extra first variable (a file pointer) telling it where to read from. Those functions return the number of variables successfully matched, or **EOF** if *End Of File* is encountered. This allows for reading an unknown number of lines, and for checking that the lines are in the desired format. In matching the format, white space is ignored. Each time variables are read successfully, **fscanf** advances the file pointer past the read text in the file so that it points at the next part to be read.
- The variable `n` holds the number of lines read. This number is updated by the `n++` part in the **for** statement.
- When a line not containing a number is encountered, **scanf** does not read it and the file pointer is not advanced — we still need to get rid of it, and we reread it to the string **garbage**. As **garbage** is really a string, we don't need to put a `&` before it in **fscanf**.
- The **fprintf** command is the file version of **printf**.
- There are three special file pointers (defined in the header file). **stdin** is the standard input. fscanf(stdin, ...) is identical to scanf(...). **stdout** is the standard output. fprintf(stdout, ...) is identical to printf(...). **stderr**, the standard error, is usually used to print error messages on the screen. When the standard output is *redirected* (e.g., `a.out > output.dat`), the standard error is still attached to the screen.
- The **fclose** command should be used when we don't need the file pointer anymore, although files are automatically treated properly at the end of the program.
- We have used *formatted output* in our program. Let's explain the *escape sequence* %+7.3f we have used. %f means, of course, the variable is of type *float*. 7.3 means that at least 7 characters will be used to represent the variable, and the precision is 3 digits after the decimal point. The + means that a sign is always printed.
- The escape sequence %e prints a *float* in an exponential notation. The escape sequence %g prints it in the more natural of the forms %f and %e.
- The escape sequences needed to print and read the non-numerical types are
  - %c for *char*s
  - %s for strings.
- A character constant is given between single quotes. For example, **'a'** is the letter **a**. Some characters require a backslash to be represented — for example, the *newline* character constant is written as **'\n'**, and **'\t'** is the *tab* character constant.
- There are many more options for clever and nice-looking formatting...
- The format in a function like **printf** is a string. It does not have to be a string *constant* — if the same format is used over and again, it is more convenient to put it in a string *variable*!

Classwork: Computing a weighted average grade. Write a program which reads the file **grades.dat**, calculates the average grade, weighted by the number of hours, and prints it with a precision of two digits after the decimal point. Make sure you allow enough space in the format even for the average grade 100 (however unreasonable) ...
The file **grades.dat** contains the grade of a course in the first column and the number of hours in the second column. For example:

80 3	70 2
60 4	95 3
75 4	85 2
90 3	100 3