Conditional Execution – the `IF` Statement

In this exercise you will use the `IF` and `IF/THEN` statements in Fortran to change the flow of control of a program, allowing it to make decisions. You will also learn something about directories in Unix.

In the previous exercise your program computed the weight and position of the center of gravity (C.G.) of a small airplane, but it was up to the user (presumably the pilot) to determine whether these were acceptable. Using the `IF` statement you can have your program warn the pilot if the total weight is too much, and/or if the C.G. position is unacceptable. You can also make your program more flexible by having it accept any number of pairs of data, until the user enters some special set of values to tell it to terminate data entry.

1. Problem

Write a new Fortran program which will read in arm and weight values for *any* number of stations. Once the total weight and C.G. position have been computed and printed, the program should warn the user if the weight is too high, or if the C.G. is not in the proper range.

1.1. Input

Read the station arm and weight (in that order, as a pair of numbers) for any number of stations, and add the weight and moment to running sums. Terminate input only when the entered values for *BOTH* the arm and weight are zero. Don't forget to echo all input values.

1.2. Output

Print the gross weight of the loaded aircraft and the position of the center of gravity, as before. If the weight is at *or above* the maximum allowed (1600 pounds) then print a warning message. If the C.G. is not *within* the proper range (31.5 inches to 37.5 inches) print a warning message. The weight and C.G. range limits for a typical small aircraft are shown in Fig. 1.

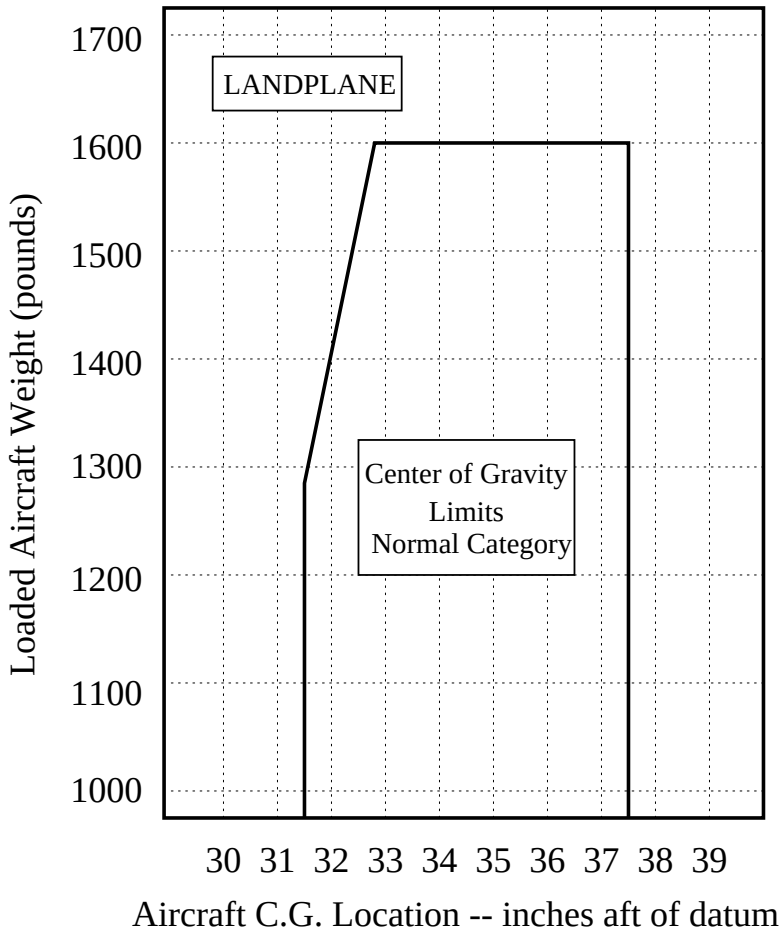

Figure 1: Weight and C.G. limits for a typical small aircraft.

2. Optional Improvements:

You do not have to make these improvements to your program, but you can do so if you really want to make it work nicely:

- Print the total weight and C.G. position after each pair of data points is entered. This is a test You should still print the total weight and C.G. position after the last data points are entered.

- Allow negative weights (which would correspond to removing a previously loaded weight) but print a warning if the total weight becomes negative.

- Print a special warning if the weight is on *or above* the diagonal part at the left of the C.G. limits shown in Fig. 1. You may assume that this line passes through the points (31.5,1280) and (32.9,1600). For the C.G. you have computed you can calculate the maximum allowed weight from the equation of this line. If the actual weight is above this value then print the error message.

3. Style Notes – Indenting

Although Fortran statements can begin in any column between 7 and 72, you can make your programs much more easy to read and understand if you follow some simple guidelines for indenting. In the case of the `IF` statement you can easily set off the parts of your program which are only executed conditionally if you indent by a couple of spaces everything between the `IF/THEN` statement and the `ELSE` or `ENDIF` statements. The `IF`, `ELSE` and `ENDIF` statements themselves should not be indented, but should line up with each other. If you nest another `IF/THEN` block within an `IF` statement then you should indent the inside statements a few more spaces, to show that they are contained within the other conditional material.

4. Directories in Unix

All computer operating systems let you organize the storage of your files into related sets of files. In Unix a collection of files stored together is called a "directory".‡ Unix has several simple commands for creating and manipulating directories. An important concept of Unix is that of the "current" directory: any Unix command which does not refer to a file explicitly by the name of the directory in which it lives will reference that file relative to the "current" directory, whatever that may be at the time the command is issued.

For the purposes of this course it would be very useful to create a directory called "`fortran`" in which to keep your exercises and other course materials. You could also create other directories for other projects you are working on. The Unix command to create a directory is `‘mkdir‘` ("make directory"), so to create this subdirectory of your home directory you should give the Unix command:

```
% mkdir fortran
```

If you then give the `‘ls‘` command to list all the files in the "current" directory, you will see a "file" called `fortran`, but it is actually the directory you just created. You can then make this directory the "current" directory by giving the `‘cd‘` ("change directory") command:

```
% cd fortran
```

If your command prompt is more than just the "`%`" it may now contain the current directory name. In any case, you can always find out what the "current" directory is with the `‘pwd‘` ("print working directory") command:

```
% pwd
/home/mavassar/fortran
```

‡ On Windows PC's and Apple Macintosh computers a directory is called a "folder" – it's the same idea.

The command shell also remembers the current directory in an environment variable called `PWD`, which you can display with the `echo` command:

```
% echo $PWD
/home/mavassar/fortran
```

No matter what the current directory happens to be, you can return to your default directory, your "home" directory, by giving the '`cd`' command with no argument, as:

```
%  cd
```

When you are in your home directory you can move files to your `fortran` subdirectory with the Unix '`mv`' ("move") command. Simply name the file(s) you wish to move, followed by the name of the directory you wish to move them to. You can verify that they have been moved correctly with the '`ls`' command with the directory name as an argument:

```
%  mv mavassar04.f  mavassar05.f  fortran
%  ls fortran
mavassar04.f  mavassar05.f
```

As with most Unix commands, the '`mv`' command says nothing if it completes with no errors. If you use '`ls`' to look at your home directory you will see that the files you moved are no longer there. They are now in your `fortran` subdirectory. The next time you log-in and want to work on one of these files, or a new exercise, you should first make the `fortran` subdirectory the current directory, with the command:

```
%  cd fortran
```

If you ever need to remove a directory the command to do so is '`rmdir`'. However, it will refuse to remove a directory which is not empty.

5. Reading

You should read in your book about conditional execution and the `IF`, `THEN`, `ELSE`, and `ENDIF` statements.

You can also learn more about the '`mkdir`', '`cd`', '`mv`', and '`rmdir`' commands by reading the on-line manual pages for those commands (*i.e.*, type '`man mkdir`' or '`man mv`').