Nested Loops

In this exercise you will use "nested" DO loops to study how variations in altitude and airspeed affect the position of where our bomb lands.

1. Problem:

The FAA only requires that new pilots be able to demonstrate that they can hold an altitude to ± 100 feet and an airspeed to ± 10 knots. Besides this, there may be errors of this magnitude in the aircraft instruments. We would like to see how a variation of the initial airspeed and/or altitude of the bomb changes the final position where it lands. One way to do this is to simply repeat the simulations with many different values for the initial position and velocity. This is sometimes refered to as performing a "parametric study."

Modify your previous program (or write a new one) to compute the trajectory for the bomb many times, with many different starting conditions, as follows: Your program should still read an altitude and airspeed as before, but now it should compute the impact position for exactly 9 altitudes between 100 feet above the given altitude down to 100 feet below the given altitude. For each of these 9 altitudes, the program should compute the impact for exactly 5 initial airspeeds, from 10 knots below the given airspeed to 10 knots above the given airspeed. The loop over airspeeds should therefore run "faster" than the loop over altitudes; that is, for each altitude you should repeat the calculation for all possible airspeeds, then move to the next altitude.

If the altitude read by your program is less than 100 feet then you should make the lowest altitude out of the 9 be exactly zero, and still consider exactly 9 altitudes from 100 feet above the specified altitude down to zero.

As before, you should use a time step of 0.1 seconds or less, and if necessary you should assume that the time of fall will be 30.0 seconds or less. If a drop takes longer than 30.0 seconds your program should not finish the drop, and it should flag in the output that the drop did not finish. (One easy way to do this is to set the time to 30.0 and the distance to zero or a negative number, but there are other ways as well.)

2. Input:

Your program should first read the reference altitude, in feet, and then the reference airspeed, in knots.

The loop over different airspeeds should use knots, but inside the loop you will want to convert each airspeed to feet per second. You should use a `PARAMETER` statement to define the conversion factor for this calculation (you might call it `KTS2FPS`).

Do not forget to echo the values of all variables you read in.

3. Output:

Use `FORMAT` statements to output the time until impact and the final position of the object (both the x and y coordinates) for each altitude and airspeed, all in an easy to read table with all numbers properly lined up and appropriate headings over each column in the table. On each line you should list the altitude first, then the airspeed, then the time, and then the impact position.

Do not print any intermediate values during your calculations, but only the final positions of the impacts. However, you may find it very useful while testing your program to use a `PRINT` statement to print the whole trajectory of the object. Be sure to remove such "debugging" statements from the final version of your program. They are the equivalent of the scaffolding put up to aid in the construction of a building and then removed when the building is completed.

4. Style Notes – Indenting

Be sure to indent the statements inside each `DO` loop a few more spaces, to make it easier to see which loops are nested inside of other loops.

5. Creating a Web page with HTML

In the previous lesson you learned that any file put in a particular directory can be served out to the World Wide Web. Most web pages are not raw data files or source code files, they are formatted using a set of 'tags' that are collectively known as HTML – HyperText Markup Language. It is very easy to create a simple but functional web page using just a text editor like emacs and knowing a few HTML tags.

The easiest way to learn to use HTML is to start with a simple example. Using emacs you could edit a file called `example.html` in your `www` directory containing the following:

```
<HTML>
<HEAD>
<TITLE>
     Sample HTML web page
</TITLE>
</HEAD>
<BODY>
<H1>
      SAMPLE Web Page
</H1>
This is a sample web page.
<P>
Here is a <a href="http://www.google.com/">link to Google</a>.
<BR>
And here is a picture of the current weather radar
     for New England, from Unisys.com:<br>
     <img src="http://weather.unisys.com/radar/rad_ne.gif">
<P>
Here is an example of pre-formatted text:
<PRE>
        print *, 'Hello, World!'
        stop
        end
</PRE>
</BODY>
</HTML>
```

Tags in HTML are enclosed in angle brackets, like `<TITLE>` or `<P>`. Some tags stand by themselves. The `<P>` tag begins a new paragraph, including a blank line. The `<BR>` tag creates a line break but does not add a blank line. Other tags enclose a bit of text, so there needs to be one tag to open and another to close. The material enclosed between the `<TITLE>` and `</TITLE>` tags is the title of the web page and will appear in the title bar of your web browser window (not in the web page itself). The material enclosed between the `<H1>` and `</H1>` tags is a heading of Level 1, which will be printed in large type and set off from the rest of the text. Notice that the title and heading are slightly different, so that you can see the difference in how they are displayed. Headings can also use `<H2>`, `<H3>`, etc for smaller headings. Don't forget the closing tags for these.

Blanks, empty lines, and extra spaces or indentation don't matter in an HTML file, though they can make the file easier to read. You need to use `<P>` and `<BR>` tags and the like to format the text so that the web page is readable. Tags can be typed in upper or lower case, it makes no difference.

The `<A>` tag starts an "anchor" for a link to another web page. In this case all text up to the closing `</A>` tag will be underlined and displayed in a different color on the web page, and clicking on that text will take you to the URL listed by the `href=` parameter (don't forget the quotes). In the example above the link will take you to the Google search engine.

The `<IMG>` tag inserts an image into the web page. The `src=` parameter is a URL specifying a GIF, JPEG or PNG image file which is to be inserted into the page. You can even enclose the image inside a set of anchor tags so that clicking on the picture takes you somewhere else.

Both the `href=` parameter for a hyperlink and the `src=` parameter for an image can refer to a file in the current directory or in a subdirectory, or to a link to some other web page or image file somewhere else in the world. The ability to link to pages and images all over the Internet is the reason why it's called the "web".

A good way to learn more HTML is to see how other web pages do it. You can look at the HTML file itself from your web browser by using the "View Source" feature.

6. Optional Improvements:

You do not have to do these parts of the exercise, but you can if you would like the extra practice:

- Use a `CHARACTER` variable to indicate that the bomb simulation did not finish.
- Type in the file `example.html` described above in your `www` directory and then try to view it with a web browser like Netscape or Safari.
- Create a web page to display the output table created by your Fortran program. Since the table is already nicely formatted using the FORMAT statement, you can enclose it in `<pre>` and `</pre>` tags, which are for "preformatted" text.
- Start your own "Home" page in a file called `Home.html` in your `www` directory. If you want to put a picture of yourself or of something interesting on the web then just borrow a digital camera and transfer the picture file to your `www` directory.
- You may optionally turn in your sample run (typescript file) via the web by moving or copying it to your web area as a "`.log`" file such as `mavassar09.log`.

7. Reading

You should read about `DO` loops, and in particular about "nested" DO loops.

You should also read about the `PARAMETER` statement, because you will want to use this to convert airspeeds from knots to feet per second.

You can learn a lot more about HTML from various lessons and tutorials on the web. To find one you like just search for "HTML lessons" on Google.