Programming in Ada 95
Step-by-Step

Equivalence For-Loop – While-Loop

<table>
<thead>
<tr>
<th>For Loop</th>
<th>While Loop</th>
</tr>
</thead>
</table>
| for I in 1..10 loop  
  Put(I);  
end loop; | I := 1;  
while I <= 10 loop  
  Put(I);  
  I := I + 1;  
end loop; |
| for I in reverse 1..10 loop  
  Put(I);  
end loop; | I := 10;  
while I >= 1 loop  
  Put(I);  
  I := I - 1;  
end loop; |

The main difference is that you need to declare the I used in the While-Loop.

Generating triangles with a single symbol character

**Problem:** You are asked to create a procedure that will output a triangle to the screen.

Let’s see what your procedure should display.

<table>
<thead>
<tr>
<th>Left</th>
<th>Right</th>
<th>Center</th>
</tr>
</thead>
</table>
| *******  
  ******  
  *****  
  ****  
  ***  
  **  
  * | *******  
  ******  
  *****  
  ****  
  ***  
  **  
  * | *  
  ***  
  *****  
  ******* |

Now, what have we to do for the left alignment?

First of all, we should start creating the structure of our Ada program.
with Ada.Text_IO; use Ada.Text_IO;

procedure left_stars is
begin
end left_stars;

We need a constant that will contain the character used to create the triangle, let’s name it Symbol, and another constant that define the number of lines of our triangle, let’s name it N_Lines.

with Ada.Text_IO; use Ada.Text_IO;

procedure left_stars is
    Symbol : constant character := '*';
    N_Lines : constant := 7;
begin
end left_stars;

We need two loops. One for printing each line, and the other for printing the correct number of stars according to the index of the line.

The first loop displays the lines of stars. Hence, we can use a For-Loop that goes from 1 to the number of lines, i.e. from 1 to N_Lines.

with Ada.Text_IO; use Ada.Text_IO;

procedure left_stars is
    Symbol : constant character := '*';
    N_Lines : constant := 7;
begin
    for I in 1..N_Lines loop
        end loop;
end left_stars;
Next, the second loop should display the stars. We will have $N_{\text{Lines}}$ of stars. First line has $(N_{\text{Lines}} + 1 - \text{Index of the line})$ times the Symbol character. Where $\text{Index of the line}$ is the $I$ (that goes from 1 to $N_{\text{Lines}}$) of the outer loop. During the first loop, it’s equal to 1, then 2, 3, ..., 10.

```ada
with Ada.Text_IO; use Ada.Text_IO;

procedure left_stars is
  Symbol : constant character := '*';
  N_Lines : constant := 7;
  begin
    for I in 1..N_Lines loop
      for J in 1..(N_Lines + 1 - I) loop
        end loop;
      end loop;
    end loop;
  end left_stars;
```

And what should the inner loop do? Only displaying one Symbol character on the screen and let the cursor on the same line to get the next Symbol character close to the previous one. Be careful, we want our lines of symbol characters displayed each time on a new line. That’s why we add a command to go to a new line after the inner loop.

```ada
with Ada.Text_IO; use Ada.Text_IO;

procedure left_stars is
  Symbol : constant character := '*';
  N_Lines : constant := 7;
  begin
    for I in 1..N_Lines loop
      for J in 1..(N_Lines + 1 - I) loop
        Put(Symbol);
        end loop;
      New_Line;
      end loop;
    end loop;
  end left_stars;
```
Now it is easy to adapt your procedure in order to align your triangle on the right. Start by renaming your procedure.

```ada
with Ada.Text_IO; use Ada.Text_IO;
procedure right_stars is
    Symbol : constant character := '*';
    N_Lines : constant := 7;
begin
    for I in 1..N_Lines loop
        for J in 1..(N_Lines + 1 - I) loop
            Put(Symbol);
            end loop;
        New_Line;
        end loop;
end right_stars;
```

The only stuff to add is some blank spaces before starting to display the stars. How many blank spaces per line? 0 for the first line, 1 for the second, ... \((\text{Index	extunderscore of	extunderscore the	extunderscore line} - 1)\) for the \text{Index	extunderscore of	extunderscore the	extunderscore line}\text{-st} line of stars. Thus, create another For-Loop that goes from 1 to \((I - 1)\) before displaying the stars.

```ada
with Ada.Text_IO; use Ada.Text_IO;
procedure right_stars is
    Symbol : constant character := '*';
    N_Lines : constant := 7;
begin
    for I in 1..N_Lines loop
        for J in 1..(I - 1) loop
            Put(' ');  
        end loop;
        for J in 1..(N_Lines + 1 - I) loop
            Put(Symbol);
            end loop;
        New_Line;
        end loop;
end right_stars;
```

We may use the same name of variable for the loop because they are at the same level of indentation.
For the center alignment, you can easily determine the number of stars for the $I$-st line: $(I - 1) \times 2 + 1$

The rest of the code is easy to find.

```ada
with Ada.Text_IO; use Ada.Text_IO;

procedure center_stars is
    Symbol : constant character := '*';
    N_Lines : constant := 4;

begin
    for I in 1..N_Lines loop
        for J in 1..(N_Lines - I) loop
            Put(' ');
        end loop;
        for J in 1..((I – 1) * 2 + 1) loop
            Put(Symbol);
        end loop;
        New_Line;
    end loop;
end center_stars;
```

You can now try to adapt the previous procedures to get these outputs...

<table>
<thead>
<tr>
<th>Left</th>
<th>Right</th>
<th>Center</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*******</td>
</tr>
<tr>
<td>**</td>
<td>**</td>
<td>******</td>
</tr>
<tr>
<td>***</td>
<td>***</td>
<td>*****</td>
</tr>
<tr>
<td>****</td>
<td>****</td>
<td>****</td>
</tr>
<tr>
<td>*****</td>
<td>*****</td>
<td>**</td>
</tr>
<tr>
<td>******</td>
<td>******</td>
<td>*</td>
</tr>
<tr>
<td>*******</td>
<td>*******</td>
<td></td>
</tr>
</tbody>
</table>