



# Introduction to Programming

## Tutorial Task 3.1: Name Tester

### Overview

Control flow enables you to easily add conditions and loops to your programs. In this task you will create a small program that uses conditions and loops to output custom messages to users.

- Purpose:** Learn to use the control flow statements within a program.
- Task:** Create a program that tests a user's name and echoes a custom message.
- Time:** This task should be completed before the start of week 4.
- Resources:**
- Chapter 5 of the Programming Arcana
  - Swinburne CodeCasts ([YouTube Channel](#), [iTunesU](#))
    - [Branching with if statements](#)
    - [Repeating code with loops](#)
  - Syntax Videos
    - [Repeat](#), [Compound Statement](#), [If Statement](#), [While](#), [Case](#)

**Note:** Remember to submit **all tasks** to Doubtfire for assessment. Also make sure you *fix and resubmit* any tasks you did not get signed off last week!

### Submission Details

You must submit the following files to Doubtfire:

- NameTester program source code.
- Answers to questions about control flow.

Make sure that your task has the following in your submission:

- Demonstrates use of Pascal programming convention, including indentation within selection and repetition statements.
- Demonstrates use of an **if** statement to perform selection, and a **while** loop to perform repetition.

## Instructions

Create a small program that will check the user's name and respond with different messages for different people.

1. Download and extract the resources for this task.
2. Open **NameTester.pas** using Sublime Text.
3. Implement a **Main** procedure with the following logic:
  - It reads a name from the user, and displays back a message.
  - Check if the name entered is your name.
  - If the name is *your name*, output the message 'Awesome name!'
  - Otherwise output the silly name message

Procedure: **Main**

Uses: `TerminalUserInput`

Variables:

- name (which stores a String value)

Steps:

1: Assign name, the result of calling `ReadString` with the prompt: 'Please enter your name: '

2: if name is '—add your name here—' then

3:     Output 'Awesome name!'

4: else

5:     Output name, ' is a silly name'

4. Try using the MinGW command line to compile and run your program.

Open a **Terminal** window and compile and run your program:

- Change into the directory with your code using the `cd` command
  - Compile your program using `fpc -S2 NameTester.pas`
  - Run your program using `./NameTester`
5. One message for everyone isn't that much fun. Enhance your Silly Name Tester so that it has custom messages for at least one other person.

**Hint:** Use Chapter 5 of the text to find the syntax diagrams and example code to help you implement this.

Saying that the name is 'silly' will have a much greater effect if we add lots of 'silly's to the output... Add a **Output Silly Name** procedure to your Name Test program. This will output the person's name and ' is a silly silly' ... with 60 'silly's, then ' name'.

Call this new procedure from main when the name is not your name.

The pseudocode for this procedure follows:

Procedure: **OutputSillyName**

-----

Parameter:

- name (the silly name String)

Local Variables:

- i (an integer, used to count the number of loops)

-----

Steps:

1: Make i equal 0

2: Output (staying on the same line) name, ' is a '

3: While i is less than 60

4:     Output (on the same line) ' silly'

5:     Increment i (make i equal i + 1)

6: Output ' name!' (moving to a new line)

**Note:** Indentation is important! Notice that lines 4 and 5 in Listing 2 are indented... this means that they are inside the while loop. Line 6 is not indented so it sits outside the while loop, as the instruction after the loop.

**Hint:** Use **Write** to output without going to a new line.

Get a screenshot of the terminal showing the output of running this program with your name, and with someone else's name.

**Questions** (*answer these using the answer sheet in this tasks resources*):

1. Assume that age has the value 0. List the actions the computer executes when it runs the following code.

```
WriteLn('Message 0');  
  
while age < 5 do  
begin  
    WriteLn('Message 1');  
    age := age + 1;  
end;
```

```
WriteLn('Message 2');
```

2. Assume that age has the value 8. List the actions the computer executes when it runs the following code.

```
WriteLn('Message 0');  
  
while age > 5 do  
begin  
    WriteLn('Message 1');  
    age := age - 2;  
end;  
  
WriteLn('Message 2');
```