



Introduction to Programming

Pass Task 1.1: Reading and Writing (Hello User)

Overview

This program will allow you to demonstrate simple reading and writing from the terminal window using Pascal.

Purpose: Develop a simple interactive Pascal program.

Task: Create your own Reading and Writing program using the Pascal. Submit to Doubtfire when complete.

Time: This task should be started in your first lab class and submitted for feedback before the start of week 2.

Resources:

- Tutorial 1 Programming Guide
- Swinburne CodeCasts ([YouTube Channel](#), [iTunesU](#))
 - [Series Introduction](#)
 - Install videos for Lazarus (for your machine)
 - [Variables](#)
- Syntax Videos
 - [Introduction](#), [Getting Started](#), [Calling Procedures](#), [Assigning Values to Variables](#), [Declaring Local Variables](#) and [Creating Your Own Procedures](#).
- Chapters 1 and 4 of the Programming Arcana
 - [Assigning Values to Variables](#), [Declaring Local Variables](#), [Calling Functions](#)

Submission Details

You must submit the following files to Doubtfire:

- Hello User source code (HelloUser.pas)
- Screenshot of the Terminal showing the execution of your Hello User program.

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

- The program must read the required details from the user, and respond correctly
- Code must follow the Pascal coding convention used in the unit (layout, and use of case).
- The code must compile and the screenshot show it working.
- Your program must have a procedure for Main.
- Your program must have the indicated local variables, and use them appropriately.

Preparation.

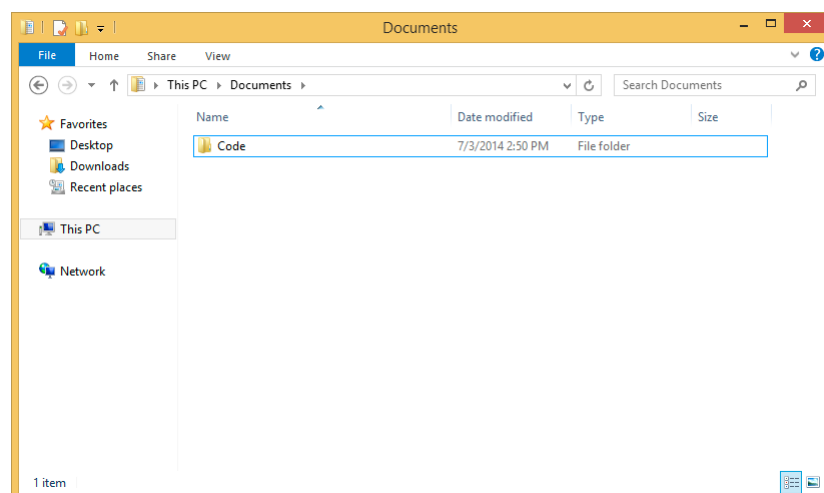
If you have not already done it then you will need to follow the steps to install the tools you will need to use SwinGame in this unit. We will first set these tools up and check they are working by creating, compiling and running the classic '*Hello World*' program that you ran in Tutorial Task 1.1.

1. Install the tools you need to get started. Watch the installation video for your Operating System and *Read Chapter 1 of the Programming Arcana* for full instructions. Ensure that you have:
 - Installed MinGW for Windows, Xcode for Mac, and devtools for Linux
 - Installed the **Free Pascal Compiler** (the compiler)
 - Installed **SublimeText** (the text editor)

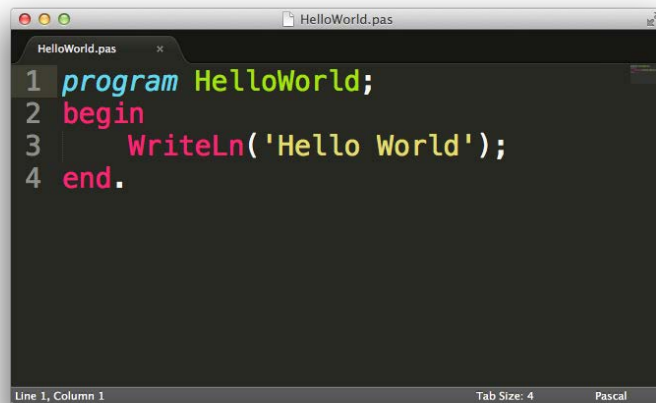
Note: You can skip this step on the computers in the Swinburne lab as this will already be setup.

Hint: If you are using your own Windows machine remember to install MinGW to give you access to the Terminal. See Chapter 1 of the Programming Arcana and the install video for extra help on how to do this.

2. If you don't already have one, make a directory (i.e., a 'folder') to store your code (e.g., *Documents/Code/Lab1*). On a Swinburne computer you may wish to use a directory on your student drive or a USB storage device.
 - Navigate to your *Documents* directory in Finder or File Explorer
 - Right click in the *Documents* directory and select **New Folder**, name it **Code**



3. Open **Sublime Text**, and create a new file.
4. Enter the text for a basic *Hello World* program. It should appear as shown here:

A screenshot of the Sublime Text editor window. The title bar shows 'HelloWorld.pas'. The editor has a dark theme. The code is as follows:

```
1 program HelloWorld;  
2 begin  
3   WriteLn('Hello World');  
4 end.
```

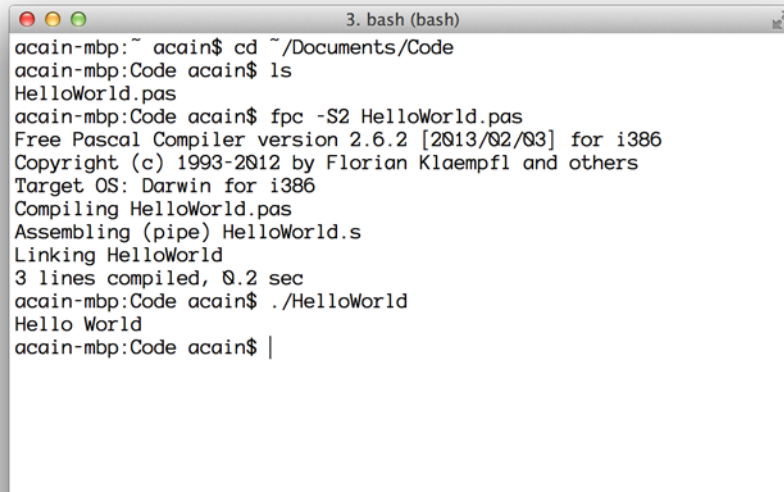
The status bar at the bottom indicates 'Line 1, Column 1', 'Tab Size: 4', and 'Pascal'.

Note: Sublime Text has a number of different colour schemes. Your colours may be different, but Sublime Text should highlight different parts of your code in different colours once you save it as a **.pas** file.

5. Save the file as **HelloWorld.pas** in your code directory.

Note: The pas file contains the *source code* for your program. To make this a *real* program you need to compile it.

6. Open a **Terminal** (MSYS or MinGW shell in Windows), then perform the following commands:
 - Change into the directory containing your code using the **cd** command.
`cd /c/Users/your_user/Documents/Code` on Windows¹ or
`cd ~/Documents/Code` on MacOS or Linux
 - List the files in this directory using the **ls** command
 - Print the working directory using the **pwd** command
 - Compile your program using **fpc -S2 HelloWorld.pas**
 - List the files in this directory using the **ls** command to see the files created by the compile process
 - Run your program using **./HelloWorld**

A terminal window titled '3. bash (bash)' showing a series of commands and their outputs. The user navigates to the directory ~/Documents/Code, lists files, and then compiles a Pascal program named HelloWorld.pas using the Free Pascal Compiler (fpc). The compilation process shows the compiler version (2.6.2), target OS (Darwin for i386), and the steps of compiling, assembling, and linking. Finally, the user runs the compiled program, which outputs 'Hello World'.

Tip: Bash commands (e.g., `cd`, `ls`, `pwd`, `fpc`) do not like spaces in directory or file names (e.g., `My Documents`, or `Hello World.pas`). If you have a space in the name of something you need to add in a reverse slash:

`My\ Documents` and `Hello\ World.pas`

Avoid spaces in the names of your files and folders!

¹ Replace `your_user` with your computer user name

Instructions

Variables allow you to store values that can change in your program. When you declare a local variable you are telling the computer to create a variable for use within the procedure. You must give the variable a name and indicate the type of data it will store. The computer will then set aside enough space for you to store a value of that type for that variable. You can then write values to the variable and read values back.

To explore this topic, we will create a Terminal program that will:

- ask the user to enter their name, age, and the current year,
- calculate the year the user was born, and
- output a greeting message
- Request the user to "Press Enter to Continue"
- Read a blank line.

1. Declare the HelloUser program using the program start code shown below.

```
program HelloUser;  
  
procedure Main();  
var  
    name: String;  
    age: Integer;  
begin  
    WriteLn('Please enter your name: ');  
    ReadLn(name);  
    WriteLn('How old are you this year?: ');  
    ReadLn(age);  
end;  
  
begin  
    Main();  
end.
```

Tip: This code is the basic program template that you can use whenever you create a new program. The *Main* procedure will have the program's main instructions.

2. Extend the code in main so that it does the following:

Procedure: **Main**

Variables:

- name (to store a String value)
- age (to store an Integer value)
- year (to store an Integer value)
- yearBorn (to store an Integer value)

Steps:

- 1: Read the user's name (a String, prompt with 'Please enter your name: ') and store it in the name variable.
- 2: Read the user's age (an Integer, prompt with 'How old are you this year? : ') and store it in the age variable.
- 3: Read the year (an Integer, prompt with 'What year is it? : ') and store it in the year variable.
- 4: Calculate the year the user was born (year - age), and store it in the yearBorn variable
- 8: Output the message, 'Hello ', name, ', you were born in ', then yearBorn
- 9: Output the message "Press enter to continue"
- 10: Read in a blank line.

3. Use Mingw and fpc to compile and run your code (or use Lazarus select "Run" then "Run" to compile and run your program).

Hint: The **WriteLn** procedure can be used to output messages and **ReadLn** will read a line of text.

Note: Don't forget to use camel case when declaring variable names containing more than one word (e.g., use yearBorn (**not** yearborn or YearBorn)).

Tip: The sections titled **Pascal Types** and **Pascal Expression** in Chapter 2 of the Programming Arcana include information on the types of data you can store, and the operations you can perform on them

Now that the Task is complete you can submit it for assessment, which will help prepare it for your portfolio.

7. Use [Skitch](#) (or your preferred screenshot program) to take a screenshot of the Terminal, as this is one of the things you will need to submit.
8. Save the document and backup your work to multiple locations!
 - Once you get things working you **do not** want to lose them.
 - Work on your computer's storage device most of the time... but backup your work when you finish each task.
 - Use **Dropbox** or a similar online storage provider, as well as other locations.
 - Doubtfire is not a Backup of your work, so make sure you keep a copy!
 - A USB key and portable hard drives are good secondary backups... but can be lost/damaged (do not rely upon them).
9. Login to Doubtfire, and locate Pass Task 1.1
10. Change the status of the task to **Ready To Mark**
11. Upload your completed Hello User code and the screenshot.
12. If you check back later Doubtfire will have prepared these as PDFs for your tutor to assess.

You now have another of your first portfolio pieces. This will help demonstrate your learning from the unit.

Note: This is one of the tasks you need to **submit to Doubtfire**. Check the assessment criteria for the important aspect your tutor will check.