



Introduction to Programming

Tutorial Task 4.1: File Handling

Overview

This program will allow you to demonstrate simple reading and writing from a file using Pascal in the Lazarus IDE.

Purpose: Develop a simple Pascal program that reads and writes from a file and uses loops and arrays. **NB: You can work on this task over Weeks 4 and 5. In Week 4 work on reading and writing records, and add the array in Week 5.**

Task: Create your own file reading program using the Lazarus IDE (or the Bash shell). Submit to Doubtfire when complete.

Time: This task should be started in your fourth lab class and submitted for feedback before the start of week 6.

Resources: Swinburne CodeCasts ([YouTube Channel](#), [iTunesU](#))

- [Branching with if statements](#)
- [Repeating code with loops](#)
- [Using arrays to work with multiple values](#)
- [Dynamically changing the size of an array](#)
- Free Pascal Wiki: http://wiki.freepascal.org/File_Handling_In_Pascal
- Free Pascal: <http://www.freepascal.org/docs-html/rtl/system/filefunctions.html>
- Syntax Videos
 - [Repeat](#), [Compound Statement](#), [If Statement](#), [While](#), [Case](#)

Submission Details

You must submit the following files to Doubtfire:

- ReadFromFile source code (ReadFromFile.pas)
- Screenshot of the Terminal showing the execution of your ReadFromFile program.

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

- The program must read the required details from the file, stores the read data in an array and then prints out each line to the terminal.
- 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.

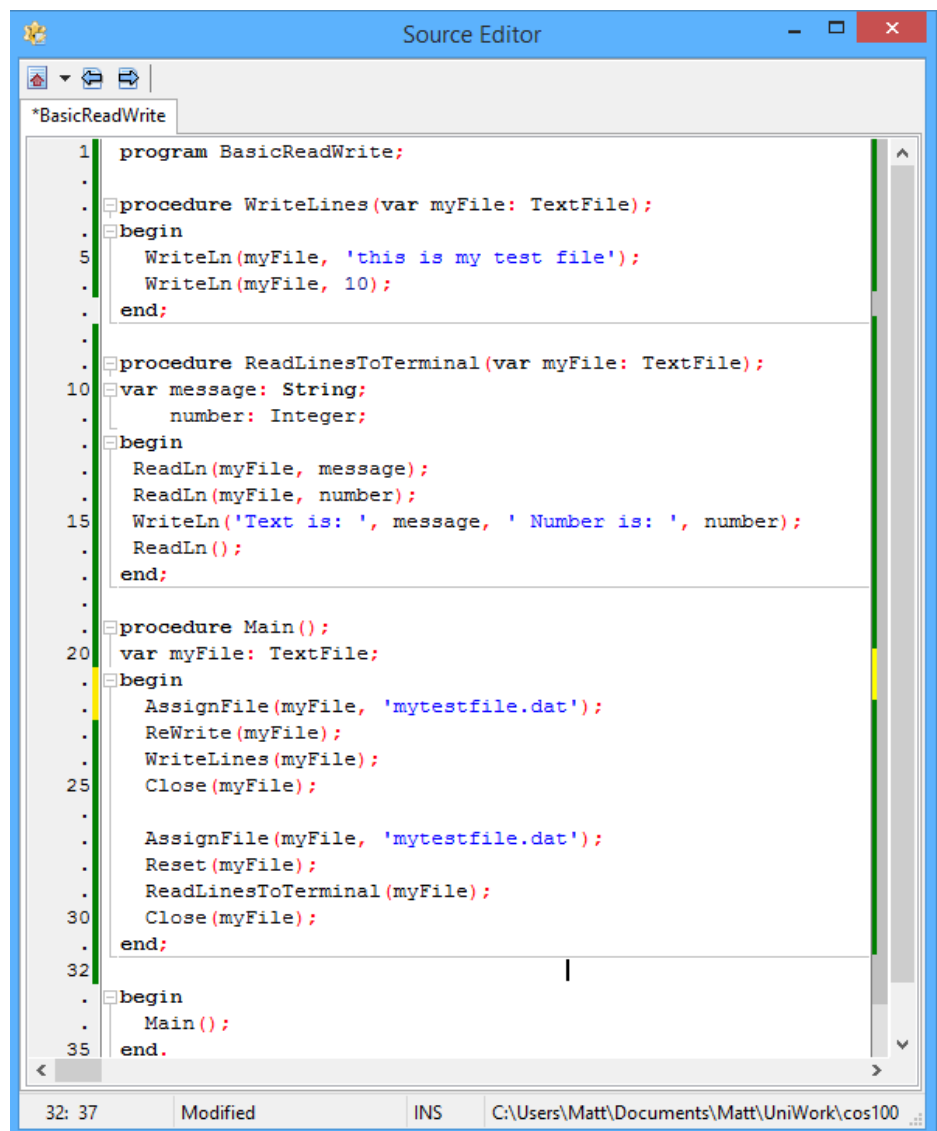
Instructions

Files allow you to store data persistently. In this task you will write a simple file reading program to read multiple lines using a loop, store each line as an element in an array, then print the lines to the terminal screen.

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

- Open a file and read in the number of lines in the file.
- Loop according to the number of lines to read in each line.
- Store each line in an array.
- Print out the lines from the array to the terminal.
- Request the user to "Press Enter to Continue".
- Read a blank line.

1. Open **Lazarus** and select "Project", "New project" then "Simple Program".
2. Declare the example BasicReadWrite program as provided in this Task's resources.
3. Compile and run the example BasicReadWrite program.
(NB: you can opt to do this in the Bash shell if you wish)



```
1  program BasicReadWrite;
.
.  procedure WriteLines(var myFile: TextFile);
.  begin
5   WriteLn(myFile, 'this is my test file');
.   WriteLn(myFile, 10);
.  end;
.
.  procedure ReadLinesToTerminal(var myFile: TextFile);
10 var message: String;
.   number: Integer;
.  begin
.   ReadLn(myFile, message);
.   ReadLn(myFile, number);
15  WriteLn('Text is: ', message, ' Number is: ', number);
.   ReadLn();
.  end;
.
.  procedure Main();
20 var myFile: TextFile;
.  begin
.   AssignFile(myFile, 'mytestfile.dat');
.   Rewrite(myFile);
.   WriteLines(myFile);
25  Close(myFile);
.
.   AssignFile(myFile, 'mytestfile.dat');
.   Reset(myFile);
.   ReadLinesToTerminal(myFile);
30  Close(myFile);
.  end;
.
32 begin
.   Main();
35 end.
```

4. Look at the code in the Resources for examples of reading and writing from files.
5. Write your own FileReadWrite.pas program which:
 - Reads an Integer from the first line of the file. This integer should indicate the number of text lines that follow.
 - Your program should then loop reading one line after another into an array until all the text lines have been read in (maybe write a function for this).
 - Calls a procedure PrintArray() (which you must write – see the pseudocode below) passing in the array and the number of lines. The procedure should print out to the terminal all the text lines that you read from the file.
 - Prompts the user to press enter to continue.
6. You will need to create a text file (**mytestfile.dat**) with a number on the first line (to indicate the following number of text lines) and then lines of text. Use notepad or Sublime Text to create the file.

Procedure: Main

Variables:

- numberOfLines (to store an Integer value read from the file)
- lineArray[20] (to store String values read from the file)

Steps:

- 1: Open the file
2. Read in the number of text lines in the file.
3. Loop reading each line of line text into the lineArray until all the lines have been read.
- 4: Call the procedure printArray(count: Integer, lines: LineArray)
- 5: Print to the terminal: 'The file contained: '
6. Loop printing to the terminal each element of the array on a new line.
- 7: Print to the terminal the message 'Press Enter to Continue'
- 8: Read in a blank line.

7. If using Lazarus select "Run" then "Run" to compile and run your program, otherwise use fpc in the Bash window.

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

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

1. Use [Sketch](#) (or your preferred screenshot program) to take a screenshot of the Terminal, as this is one of the things you will need to submit.
2. If using Lazarus save the document (using Lazarus "Save As") 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).
3. Login to Doubtfire, and locate Tutorial Task 4.1
4. Change the status of the task to **Ready To Mark**
5. Upload your completed Hello World code and the screenshot.
6. 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.