Department of Computer Science
University of Pretoria

Programming Languages
COS 333

Practical Lab Experience 2: Ada, C and Fortran

16 August 2016

1 Objectives

This practical lab experience aims to achieve the following general learning objective:

- To gain and consolidate some experience writing programs and scripts in several different imperative languages, including: Fortran, C and Ada;
- To consolidate a variety of basic concepts related to imperative programming languages, as presented in the prescribed textbook for this course.

2 Plagiarism Policy

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.ais.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then click the *Plagiarism* link). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

3 Submission Instructions

Upload all practical-related files as a single tar-gzip archive (named `s99999999.tgz`, where `99999999` is your student number), to the appropriate assignment upload on the course website. Multiple uploads are allowed, but only the last one will be marked. The deadline is **Monday, 29 August 2016, at 23:30**. The archive must include all the program source files that you have written. The practical will be assessed during the practical sessions in the week of 29 August 2016. The uploaded files must be the ones that you demonstrate during your practical session.

4 Background Information

For this practical, you will be writing programs in Fortran 2008, C and Ada 2012. You will have to compare these languages in terms of their support for different concepts related to data types, control structures and subprograms. To do this, you will have to write short programs to demonstrate how each language handles the concept under consideration. These programs do not need to be long, but must demonstrate the concept adequately, and allow you to describe the language's support (or lack of support) for the feature.

Your Fortran 2008 program should compile using the `gfortran` compiler, your Ada program should compile using `GNAT` (the GNU Ada Compiler extension for the GNU Compiler Collection), while your C program should

compile with the `gcc` compiler. Support for all three languages is provided under Linux as part of the GNU Compiler Collection. If you decide to use another compiler, **make sure that you test your programs using these compilers**.

The course website contains documentation related to the Fortran language [3] and the `gfortran` compiler [4]. The course website also contains documentation related to the Ada language [1]. Documentation for the `GNAT` compiler is available online [2]. For C, you are referred to the texts and references for previous courses related to the C++ language, since all of the concepts you will need are a subset of the C++ language.

Note that that your Fortran program code must comply with the `gfortran` compiler's fixed format. Adherence to the fixed format will contribute to your mark. For the C implementation, you may not use any C++ language features (i.e. compile the program as a C program, not a C++ program).

5 Practical Tasks

You will need to write programs that generate matrices. Each program must create and initialise a 5×5 matrix of the basic form:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \\ 26 & 27 & 28 & 29 & 30 \end{pmatrix}$$

and obtain an integer between 1 and 10 from the user (specified either from the command line or via an input stream). The program must then make two subprogram calls (use either a function, subroutine, or procedure, depending on which is the most appropriate), one for populating the matrix and one for displaying parts of the matrix.

The first subprogram must populate the matrix based on the user's input. You must use a switch or case statement to implement the following rules. The populated values should be computed (not hard coded), so that the rules will work for any matrix values.

- If the user's number is 2, then the the sum of each row in the matrix needs to be calculated and each element in the row needs to be changed to that total, as follows:

$$\begin{pmatrix} 15 & 15 & 15 & 15 & 15 \\ 40 & 40 & 40 & 40 & 40 \\ 90 & 90 & 90 & 90 & 90 \\ 115 & 115 & 115 & 115 & 115 \\ 140 & 140 & 140 & 140 & 140 \end{pmatrix}$$

- If the user's number is 1 or 3, the sum of each column needs to be calculated and each element in the column needs to be changed to that total as above.

- If the user's number is in the range 4-7 (including 4 and 7), then the sum of both diagonals needs to be calculated and each element in the those diagonals needs to be repalced with that total, as follows:

$$\begin{pmatrix} 160 & 2 & 3 & 4 & 160 \\ 6 & 160 & 8 & 160 & 10 \\ 16 & 17 & 160 & 19 & 20 \\ 21 & 160 & 23 & 160 & 25 \\ 160 & 27 & 28 & 29 & 160 \end{pmatrix}$$

- If the user's number is in the range 8-10 (including 8 and 10), then each even number in matrix needs to be replaced with that number squared and each odd number replaced with that number cubed.

- If the user's number is not in this range then the matrix should be left as is.

Consider using an automatic array initialisation mechanism to initialise the basic structure of the array, after which the matrix elements can be modified to produce the required matrices. Your programs must display the matrix, in the general format described above, after this subprogram has been called. Also note that this subprogram should ONLY populate the matrix and not dispaly it

For the second subprogram the user must specify any value in the matrix and the subprogram must find and return the entire row and entire column of the first instance that value is found in (start the search in the

top left corner of the matrix, and search right, row by row), as two one-dimensional arrays. It must also print out these arrays. Note that if value is not found the output should be the entire matrix.

Beyond these requirements, you may code the remainder of the program as you see fit. Bear in mind that this program must be implemented in all three of the aforementioned languages, namely: Fortran 2008, Ada 2012, and C. Marks will be awarded for the sophistication of the language features you use (we will specifically look at the features of the switch or case statement you use, the mechanism you use for array initialisation, whether rectangular arrays are used instead of jagged arrays, and whether array slices are used to access the user-specified row). Note that all these language features are not present in all three languages, and you should therefore use appropriate language constructs for each language.

Include a comment at the top of each program, explaining how the program should be compiled and executed, and any special requirements that the program has.

6 Marking

Each of the programs will count 5 marks for a total of 15 marks. Both the implementation and the correct execution of the programs will be taken into account. Your program code will be marked during the practical sessions in the week of 29 August 2016. Make sure that you upload your complete program code to the appropriate assignment upload slot, and include comments at the top of each program, explaining how your program should be compiled and executed, as well as any special requirements that your program has.

References

[1] AdaCore. GNAT reference manual, 2015.

[2] AdaCore. GNAT user's guide for native platforms, 2016. `http://docs.adacore.com/gnat_ugn-docs/html/gnat_ugn/gnat_ugn.html`.

[3] PGI Compilers and Tools. PGI Fortran reference: Version 2015, 2016.

[4] The `gfortran` team. Using GNU Fortran: For GCC version 4.5.0, 2008.