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, 17 August 2015, at 12:00. The archive must include all the program source files that you have written.

4 Background Information

For this practical, you will be writing programs in Fortran 95, 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 subroutines. 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 95 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 your Fortran program will have to comply with this formatting in order to compile successfully. You must use the appropriate compiler settings to enforce this formatting.

5 Practical Tasks

You will need to write programs that analyse poker hands. That is, the programs should take command line input of five strings, each of which represents a card. For example, the command line input might look as follows: “2D 5C 10S 6D 6H” (representing the 2 of diamonds, 5 of clubs, etc.) The representation you use for each card is up to you, provided that suite and number are both represented. The program must then identify whether or not the given hand makes up either: straight, flush, straight flush, or none. The result should simply be printed to screen.

In order to implement this, a number of language features MUST be used. As this practical is centered around procedural programming, there is no need to write your own abstract data type for the cards or the deck. Your program must contain:

- Arrays to represent the provided hand: you must make use of two arrays — one of which represents the card numbers (integer) and a second which represents the suite (character or string) — these can be limited to a capacity of 5.
- Two functions/procedures: the first of which, called “initialise”, takes five parameters, being card input, and populates the aforementioned arrays accordingly. This sets the poker hand. The second, called “analyse”, takes no parameters and makes use of the contents of these arrays to determine what type of poker hand can be found in those five cards (straight, flush, straight flush, or none) and return the result in the form of a string. If a flush is found, the suite of that flush must be printed as well.
- Selection statements: at least one switch or case statement must appear in your code. For example, you may use these in the “analyse” method to determine the suite in which a flush has been found, so as to report the result accordingly.

Beyond these requirements, you may code the remainder of the program as you see fit. Note that the program must be implemented in all three of the aforementioned languages: Ada 2012, Fortran 95, and C.

6 Marking

The uploaded archive will serve as proof of completion for this practical. The practical tasks themselves will be manually marked, as indicated in the course study guide, during the practical sessions of Monday 17 August and Tuesday 18 August. Each of the programs will count 5 marks for a total of 15 marks. Both the implementation (according to the specification given above) and the correct execution of the programs will be taken into account.

References


[1] The details relating to the requirements for a particular type of poker hand can be found online, in the related Wikipedia article, at http://en.wikipedia.org/wiki/List_of_poker_hands.