Name: Travis Stop

Professor: Dr. Michael Oudshoorn

Class: CSC4510 (Programming Language Design and Implementation)

Date: 11/13/2020

Part 1: Generic Mechanisms

At first glance, the generic programming mechanisms in C++ seem to be much simpler to implement than in Ada. There were a total of three lines of code that needed to be adjusted in the case of C++ in order to make a functional generic exchange sort subprogram. Generic programming in Ada is much more involved. First there needs to be a generic procedure/function declaration, followed by its implementation, followed by it’s instantiation in a “driver” program. In C++, An abstract function can be ready to use with any data type as soon as it’s declared and defined.

One interesting thing i noticed about generic programming in Ada is the ability to restrict the use of a procedure to a specific set of data types or “subtypes”. This seems to give the programmer a level of control over how abstract functions are used that is not as possible in C++ to my knowledge.

Part 2: Natural Language in Describing Programming Mechanisms

Natural language was not very useful to me in describing the syntax of ada, let alone the generic programming mechanisms of Ada. I found it much more useful to look at example programs in Ada and see how adjustments to their syntax altered the behavior of the program. I think the biggest issue with using natural language to describe programming languages in general is that it’s hard to describe something as abstract and precise as Ada generic programming mechanisms using an ambiguous language like English. One thing that was difficult to figure out was how to create an array that was of a size determined by user input. There were few examples on the internet of how to create an array in Ada without populating it with values, so I had to almost guess the syntax.