Ada Fundamentals with GNAT: Workshop 5

1. Generic Stack package

Objective: This problem illustrates generic units.

Define a generic package Generic\_Stack\_Pkg as a generalization of Int\_Stack\_Pkg from Workshop 4, problem 1; i.e., Generic\_Stack\_Pkg should have a formal type parameter for the element type. Declare Int\_Stack\_Pkg as a “top-level” instantiation of this generic, with Integer as the element type, and test it with the same main procedure as for Workshop 4.

2. Generic random permutations

Objective: This problem illustrates generic units

In Workshop 2, problem 3 you implemented a package for generating random permutations of arrays of Float elements. The same algorithm can be used for any type that has an assignment operation. Define a generic package, Generic\_Array\_Handling, that takes appropriate formal type parameters, and revise the main procedure accordingly.

3. Generic math integration

Objective: This problem illustrates generic units.

a) Generalizing to any floating-point type

In Workshop 3, problem 4 you implemented an Integrate function that took a run-time parameter identifying which function to integrate. This function was required to take a parameter of type Float and to return a value of type Float. However, the algorithm was not specific to type Float, and should work for any floating-point type. Define a generic version of Integrate\_Pkg, named Generic\_Integrate\_Pkg, with a formal floating-point type parameter. Instantiate this with the type Float to produce Integrate\_Pkg, and test it with the main procedure.

b) Passing a function as a generic formal parameter

Instead of passing the function as a run-time parameter, you can alternatively express Integrate as a generic function, where the function to integrate is a formal subprogram parameter to the generic. Replace Integrate\_Pkg with a generic function, Generic\_Integrate, as follows:

**generic  
 type** Floating **is digits** <>;  
 **with function** Func( X : Floating ) **return** Floating;  
**function** Generic\_Integrate( X\_Low, X\_High : Floating;  
 N : Positive )  
 **return** Floating;  
*-- Integrates Func from X\_Low to X\_High, with N intervals*

Implement the body of this generic function (by updating the corresponding function from problem 3a) and revise the main program accordingly. Instantiate the generic function either at the “top level” or in the declarations of the main procedure.

What are the advantages and disadvantages of the two approaches (function as run-time parameter versus function as generic parameter)?

4. Defeating Ada’s strong typing

Objective: This problem illustrates some of Ada’s low-level features

Define a procedure that declares local Integer and Float variables at the same address. Assign a floating-point value to the Float variable and then display the Integer. Feel very guilty about doing this, and promise only to use this technique when you absolutely have to.

5. Counting lines and line length

Objective: This problem illustrates text file handling

[File handling was not covered in the lecture; if you would like to do this exercise please refer to Ada.Text\_IO in the Ada reference manual or in the Barnes textbook (§23.6).]

Write a program that prompts the user for the name of a text file and then processes the file and displays the following information:

* The number of lines
* A table of line length frequencies

For example, if the text file contents are:

This is line 1  
This is line 2  
And this is the last line

Then the program should display the following output:

Number of lines: 3  
Length Frequency  
 14 2  
 25 1

For simplicity, you can assume that no line in the file is longer than 100 characters.