Lecture C2: The Spider Adventure

Response to 'Muddiest Part of the Lecture Cards'

(63 respondents)

1) ***Not sure what we will do for homework in this class***? (1 student)
Homework are/will be published under the "homework" link on the unified course web page.
There will be a few specific exercises based on every lecture in CP.

2) ***Need more information on how to call packages, what syntax to use for Ada constructs, is Ada case sensitive, …*** ? (and similar questions) (7 students)
The fall term has 17 CP lectures. More or less all of them will introduce new Ada constructs and/or show Ada code. The first two lectures of CP, C1 and C2, have been of an introductory sort, showing you examples rather than the exact details. Starting with lecture C3, we will take it step by step with all the details, and I will tell you more than you ever wanted to know about Ada :)

There is something called the **Ada Style Guide**, it can for example be found via the following link. The style guide is divided into sections that map to the major decisions that each programmer must make when creating high-quality, reliable, reusable, and portable Ada software. Some overlap exists in the sections because not all programming decisions can be made independently.

3) ***In the Hello_Semantic example, if numbers had been placed within quotation marks, would the program have worked***? (1 student)
Yes, it would then have produced the output: **12345We hope you enjoy studying Ada**

Another way could have been to include the following context clause: **with Ada.Integer_Text_Io;** and change the Put statement to be: **Ada.Integer_Text_IO.Put(Item => 12345);** The result would have been very similar to the one shown above.

4) ***Why does the spider program run so slowly***? (1 student)
Spider runs slowly because of the way both the screen and spider packages are defined. We will come back and revisit the call flow in spider when we talk about packages in detail.

5) ***What is difference between a procedure and a function***? (1 student)
Functions can only accept inputs and generate a single output. Procedures on the other hand can both accept inputs and generate multiple outputs.

6) ***What keys do we use to produce a list file***? (1 student)

Alt + F2, or via menu **Compile --> Compile to Listing** in AdaGIDE.

7) ***What was the first file shown for the spider program***? (1 student)
That was the specification for the Spider package. Name of file is **spider.ads**

8) ***What other languages will we use other than Ada***? (1 student)
You will use Ada 95 and a machine language, which is presented in Appendix C in the Brookshear book.

9) ***What are debugging techniques for run-time and logic errors***? (1 person)
Run-time errors are typically caught by adding exception handling to your code.
Logic errors on the other hand are caught by means of 1) code reviews during development; 2) testing the code once it has been developed.

10) ***What is, and how is a FOR LOOP used***? (and similar loop questions) (6 students)
Different loop constructs will in detail be covered in lectures 4, 10 and 11.

11) ***Where do we find out which commands a package contains***? (and other package questions) (6 students)
Use a text editor of your choice, or the AdaGIDE tool to open the files filenam.ads and filename.adb. An Ada package is divided into two parts; the interface/specification (filename.ads), which can be seen as a 'table of contents' for the set of resources the package provides. The other part of a package is the implementation/body (filename.adb), which contains the actual code segments for all the operations in the package.

For example a statement like "Spider.Step;" as shown in the examples shown in class means that we want to execute the procedure Step that is available via the Spider package. Don't forget that for us to be able to use routines from a package we need to include the context clause "WITH Spider;" in the beginning of our application/implementation. Take a look in files Spider.ads and Spider.adb for exact details.

```
-- Following four lines are copied from spider.ads The details of procedure
-- step can be found in spider.adb
PROCEDURE Step;
-- Precondition: None
-- Postcondition: Spider takes one step forward in the direction it is facing.
-- Raises: Hit_the_Wall if spider tries to step into a wall.
```

Yes, you will later during CP have to write your own packages.

A detailed presentation of packages will be given in lecture C9.

12) ***Is Ada object oriented or procedural***? (1 student)

Ada is a procedural language. The latest version of Ada, called “Ada 95’ is also an object-oriented language. The former standard of Ada, called “Ada 83” was not object-oriented. “Ada 95” is the first object-oriented programming language to become standardized. The ISO/ANSI standard (ISO-8652:1995) is the latest standard for Ada which was accepted in February 1995.

13) ***What is an example in Ada code of a run-time error***? (1 student)
Run-time errors are called exceptions in Ada. The most common kind of exception relates to the range of variables. A range error occurs when a program tries to save an inappropriate value in a variable. This could happen if the program itself computes a result that is out of range for the variable in which it should be saved, or a user of the program enters a value via the keyboard that is an out-of range value.

14) ***Can a program be executed on any machine after it is compiled and linked***? (1 student)
No, you will need a specific compiler for your type of machine (UNIX, Mac, PC, ...), if you want to compile code on your machie to run on another platform, you will need to use a 'cross compiler'.

15) ***What is gnatcc, system libs, and gnalb***? (1 student)
Take a look at this.

The entire process of converting Ada95 code into an executable program is shown below

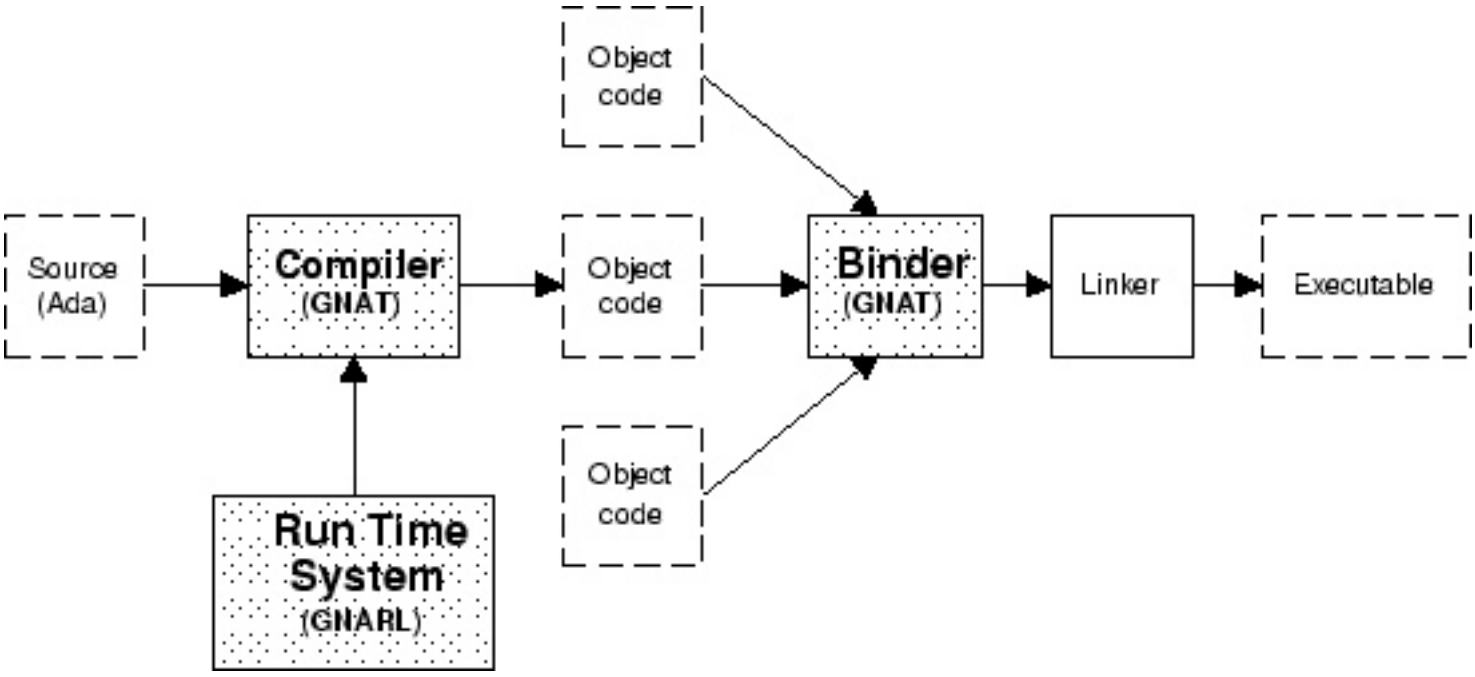

Figure 1. GNU Ada Translator (GNAT) Process *

It has three main parts: the *Compiler*, the *Run Time System* and the *Binder* (the GNU linker for the target operating system is always reused).

We will address the pieces in reverse order:

The GNAT binder verifies the consistency of the objects and determines a valid order of elaboration (initialization) for the objects (from the same or different languages) that are to be assembled into an executable file. Following this sketch, the next sections of this chapter describe each part of the compiler. This is summarized in slide 5 as “gnatbl”

All Ada language constructs are translated into calls to functions/ procedures in the GNARL (Gnu Ada Run-Time Library). This translation step is summarized as “system libs” in slide 5.

GCC (GNU Compiler Collection) is a collection of compilers for the C, C++, Objective-C, Ada, Fortran, Java and treelang languages. The Ada compiler of the gcc is refered to as gnatcc in slide 5.

* Figure From Javier Miranda – A Detailed Description of the GNU Ada Run Time.

16) ***What parts of a program should be compiled***? (1 student)
All parts of a program has to be compiled. After each change to your source code you will have to recompile the code. Packages only has to be compiled, they should not be linked.

17) ***What commands does Ada understand***? (1 student)
Information about all existing Ada commands and how they should be used can be found in the online Ada 95 standard: the Ada 95 Reference Manual. Another document well worth taking a look at is the Ada 95 Rationale which describes the overall scope and objectives of the Ada 95 standard, and gives an overview of its main technical features. Both of these documents should also be available on the Feldman CD.

18) ***The screen files on the Feldman CD did not work on my XP machine, will the updates on CP web page fix that***? (1 student)
That is my intention. Please let me know if the new screen package does not solve your problems.

19) ***"No mud"*** (28 students)
Good :)