Brian Jones

9/15/2013

Comp 333

Homework #1: Research Paper on ADA

Ada was developed in 1983 by Dr. Jean Ichbiah’s team at CII-Honeywell-Bull with the intention of being a safer programming language to use for applications that required maximum reliability and repeatable run times. It was enhanced in 1995 under the supervision of Mr. Tucker Taft from Intermetrics to include more features that would allow Ada to become more mainstream and compete with other Programming Languages, especially by adding the option of Garbage Collection to the programs created using Ada (although Garbage Collection was still strictly optional). And after all this time, Ada is still very useful in today’s Programming environment, even gaining more prestige as the Department of Defense requires more and more new contracts to be written in Ada for its ability to be real-time sensitive and its repeatable performance metrics.

There were several features that were standardized in Ada that were somewhat tedious in other programming languages, which greatly reduced coding time and increased reliability and efficiency. Ada allows programmers to constrain variable ranges when they are declared, and throws an immediate run-time error when a value goes out-of-bounds, exceeding the defined range. Ada also determines Generic-Types at compilation time, rather than run-time, thus reducing errors and ensuring repeatable run-times since the program will be compiled with the variable-types hard coded into the program, eliminating the chance of the user-system making a mistake.

Garbage Collection was originally excluded from Ada (since there was no way to guarantee repeatable benchmarks for the program with something as random as Garbage Collection being a possible interruption to the program flow), but in Ada95 it was finally included for many programmers that wanted to use Garbage Collection on things such as GUIs (which are not inherently real-time sensitive and can spare CPU cycles for Garbage Collection). This garbage collection was referred to as “finalization”, and allowed for programmers to “reclaim” certain type-specific constructs.

Ada is especially powerful because it was made with Concurrency (aka “tasks”) and System-Level programming in mind. This provided for a system for sharing data between program modules either implicitly or explicitly (via “rendezvous”, a synchronous data sharing mechanism). Ada makes it easier to manipulate data-records and bit-fields, define data-alignment and size in memory, write Assembly Language code within the program for time-critical code, and even to write custom Interrupt Handlers. Ada was the first language to implement the “Earliest Deadline First” policy, which made it so that whatever “task” had the earliest deadline was required to finish first, rather than allowing Priority-Level Interruptions to interfere with the program flow, thus ensuring consistent run-times.

Ada is a Programming Language that was built off of Algol68, Pascal, C, Smalltalk, Java, and Eiffel, using very strong type-casting to eliminate errors encountered with weaker type-casting (such as by accidentally utilizing pointers as integers in C++, for example). Ada prevents buffer-overruns in arrays, and allows for interfacing to other Programming Languages (such as C++/Java/etc.).

All of these features make Ada very suitable for use in time-critical applications where systems are mission-critical/safety-sensitive (such as on aircraft flight-control and Air Traffic Control systems) and require a very high degree of efficiency and reliability, which is why the Department of Defense primarily employs Ada to write all of their software with, and requires any new contracts to be written in Ada. And with Ada compilers becoming cheaper, or even free, many more Programmers are learning Ada which is bound to only build upon it and make it more popular in more time-critical fields.

# The Ada Program: frequency.adb

1 ***-- frequency.adb: count, display character frequency in standard input***

2

3 **with** [Ada.Text\_IO](http://www.adahome.com/rm95/rm9x-A-10-01.html);

4 **use** Ada;

5

6 **procedure** Frequency **is**

7

8 **subtype** Printable **is** Character **range** ' ' .. '~';

9 Counts : **array** (Printable) **of** Natural := (Printable => 0);

10

11 Char: Character;

12

13 **begin**

14

15 **while** **not** Text\_IO.End\_Of\_File **loop**

16 **while** **not** Text\_IO.End\_Of\_Line **loop**

17 Text\_IO.Get (Item=>Char);

18 **if** Char **in** Printable **then**

19 Counts (Char) := Counts (Char) + 1;

20 **end** **if**;

21 **end** **loop**;

22 Text\_IO.Skip\_Line; ***-- move pointer in input buffer past EOL***

23 **end** **loop**;

24

25 **for** I **in** Counts'Range **loop**

26 **if** Counts(I) > 0 **then**

27 Text\_IO.Put (I & ": ");

28 **for** J **in** 1 .. Counts(I) **loop**

29 Text\_IO.Put ('\*');

30 **end** **loop**;

31 Text\_IO.New\_Line;

32 **end** **if**;

33 **end** **loop**;

34

35 **end** Frequency;

Basically, this program reads characters from a file, and every time a character lies within the range of ‘Printable’ (a Character variable with a range from ‘ ‘ to ‘~’), the program will increment the value found in the Counts array at the element represented by that character. The program will continue to do this until the end-of-file is reached. For example, if the character ‘%’ were read into the program, then Counts(‘%’) would be incremented. This makes it easier to keep track of the character frequencies, since the characters themselves mark the index where they can be found in the Counts array.

Then, the program will loop from 1 to Counts’ Range, looking for any element in Counts(I) that is greater than 0. When found, the program begins processing the data for output, using an ‘\*’ to represent each individual occurrence of each character found in the array.

**Sources**

1. “The Ada Programming Language”, AdaCore, [online] 2011, <http://www.adacore.com/adaanswers/about/ada> (Accessed: 14 September 2013)
2. “Benefits and Features”, AdaCore, [online] 2011, <http://www.adacore.com/adaanswers/benefits-and-features> (Accessed: 14 September 2013)
3. “Ada (Programming Language)”, Wikipedia, [online] 31 August 2013, <http://en.wikipedia.org/wiki/Ada_(programming_language)> (Accessed: 14 September 2013)
4. Ryan Stansifer. (1998 Oct 01). “Notes about the Ada Programming Language”, [online], Available: <http://cs.fit.edu/~ryan/ada/language/index.html> (Accessed: 14 September 2013)
5. Ryan Stansifer. (1998 Oct 01). “The Ada Program: frequency.adb”, [online], Available: <http://cs.fit.edu/~ryan/ada/programs/array/frequency-adb.html> (Accessed: 15 September 2013)
6. Edward G. Okie. (Fall 2013). “Arrays”, [online], Available: <http://www.radford.edu/~nokie/classes/320/Tour/arrays.html> (Accessed: 15 September 2013)
7. “Citing and Referencing”, University of Kansas, [online], <http://www.ittc.ku.edu/~krsna/citing.htm#Website> (Accessed: 15 September 2013)
8. When was ADA Developed? Who developed it?
   1. ADA83 was developed in the early ‘80s by Dr. Jean Ichbiah’s team at CII-Honeywell-Bull
   2. ADA95 enhanced ADA83 in the ‘90s under the supervision of Mr. Tucker Taft from Intermetrics
9. Why was it developed?
   1. Designed for large programs (particularly for embedded-systems) requiring MAXIMUM efficiency and reliability
   2. Highly utilized in the Department of Defense, where systems are mission-critical/safety-sensitive (such as on aircraft flight-control systems) and thus require a very high degree of efficiency and reliability
   3. Highly focused on “real-time critical” programming
10. What new features were added?
    1. Allowing programmers to specify value constraints on variables when they are declared, and attempting to assign an out-of-bounds value would cause an immediate run-time error. This allowed for programmer intent to be very clear when reading source-code, allowing debugging to be easier and more efficient.
       1. Examples:
       2. “type Weekday is (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);”
       3. “subtype Working\_Day is Weekday range Monday .. Friday;”
       4. “type Day\_type is range 1 .. 31;”
    2. Generic-type determination are made at Compilation, rather than Run-Time, thus reducing errors and run-time and ensuring reliability (since the program will be compiled with the types hard-coded, rather than leaving that determination to the user-system)
    3. Garbage-Collection was originally excluded from ADA (since it would interfere with reliable run-time and efficiency metrics), but was introduced in ADA95 as an optional feature called “finalization”, which allowed for the programmer to “reclaim” certain type-specific constructs (for GUI usage, for example, since GUI’s are not typically real-time sensitive).
    4. Concurrency (aka “task”)
       1. Provided a structured, high-level system for sharing data between modules either implicitly or explicitly (via “rendezvous”, a synchronous data sharing mechanism)
    5. Makes it easy to get down to the System level, even allowing easier bit-field manipulation of data-records, define data alignment and size, define data locations in memory, code in explicit assembly-language when time-critical code is required, or even write custom Interrupt Handlers
    6. “Earliest Deadline First”:
       1. Whichever task had the earliest deadline was REQUIRED to finish first, rather than allowing Priority-Level Interruptions to interfere, thus ensuring consistent run-times
    7. STRONG type-checking:
       1. Pointers cannot be accidentally utilized as integers, for example
    8. Prevents buffer-overruns in arrays
    9. Allows interfacing to other Programming Languages (like C++/Java/etc.)
    10. Expanded Compile-Time checks that enhance debugging and reliability factors
11. What languages did it build on? What were the differences between each?
    1. Algol68
    2. Pascal
    3. C++
       1. Stronger type casting in ADA, as well as buffer-overflow protections and bounds checking
    4. Smalltalk
    5. Java
    6. Eiffel
12. Was it a success?
    1. Yes, but further advances are needed to see certification in life-critical systems
13. Who is using ADA now?
    1. The Department of Defense MANDATES use of ADA for all contracts now
14. Where is ADA on the Tiobe Index?
    1. #23
15. How has ADA fared recently?
    1. Doing better and worse
    2. DOD allows for other than ADA now, just as cheaper/free ADA compilers are becoming available, thus simultaneously encouraging more ADA usage but discouraging people to learn it because it is no longer in such high demand.