Shawn Kaplan (0966499) April 6, 2020

Fun with Languages - Reflection Report

For this assignment, I opted to create the Russian Peasant Multiplication algorithm. Overall, this was an amazing experience and helped me confirm my beliefs on the advantages and disadvantages of the languages we learned throughout the course. I found that the struggles I faced while working with Fortran, Ada, and Cobol in their respective units helped build a strong foundation to get me through this assignment quickly, as I spent less than two hours on all of the coding and testing.

The first step I took after reading the instructions was converting the pseudocode into C, as this is the language I am most comfortable with. I found that the pseudocode given was nearly identical to the code that had to be written, so making the recursive function was extraordinarily simple. Once I had the recursion down, I was able to fully understand the algorithm, so I quickly made the iterative version and started testing. After a few minutes of giving the program different inputs, I was satisfied with my C code and started on my next favourite language, Ada.

When writing the Ada program, I followed a similar approach to the one I took with C, except this time I had functional C code to essentially copy/paste into the Ada file and make minor syntax changes. This was by-far the fastest part of the assignment for me and reaffirmed my love for Ada. Once I was done testing this program, I moved on to Fortran.

While Fortran is my least favourite language of the ones we learned this semester, it is syntactically similar to Ada and C, so I figured it would be easy to port prior code. For the most part, this turned out to be true, except Fortran's syntax for recursion differs from the other two languages. Further, learning how to properly print numbers with strings in a single line was a challenge, but I found a simple workaround and used multiple print statements as I had grown tired of Fortran already. Working with this language was by far the most time-consuming part of the assignment and I was glad to move on to Cobol after brief testing.

Writing the Cobol program was largely just a matter of determining how big to make the variables; handling the math logic was nothing compared to assignment 3, so it mostly went by without a hitch. I found this power to decide the size of any variable to be both a gift and a curse. The lack of categorization between types of numbers (long, float, int, double, etc.) made the process of handling numerical input daunting at first, but once I realized that I actually had more freedom dealing with massive numbers in Cobol than the other languages, I just gave each number 30 digits and called it a day.

Overall, my previous thoughts on Fortran, Ada, and Cobol were only reassured by this assignment. I found Ada to be, by far, the most enjoyable language in this course, and it is one that I am open to learning more in the future. Cobol, while uniquely interesting with its variable declaration and surprisingly high-level syntax, feels limited in other ways, like its lack of functions and inability to perform recursion. Fortran, for some reason, just does not sit well with me; input/output formatting was a nightmare to learn, especially compared to the other 3 languages, which made concatenating numbers with strings very simple.

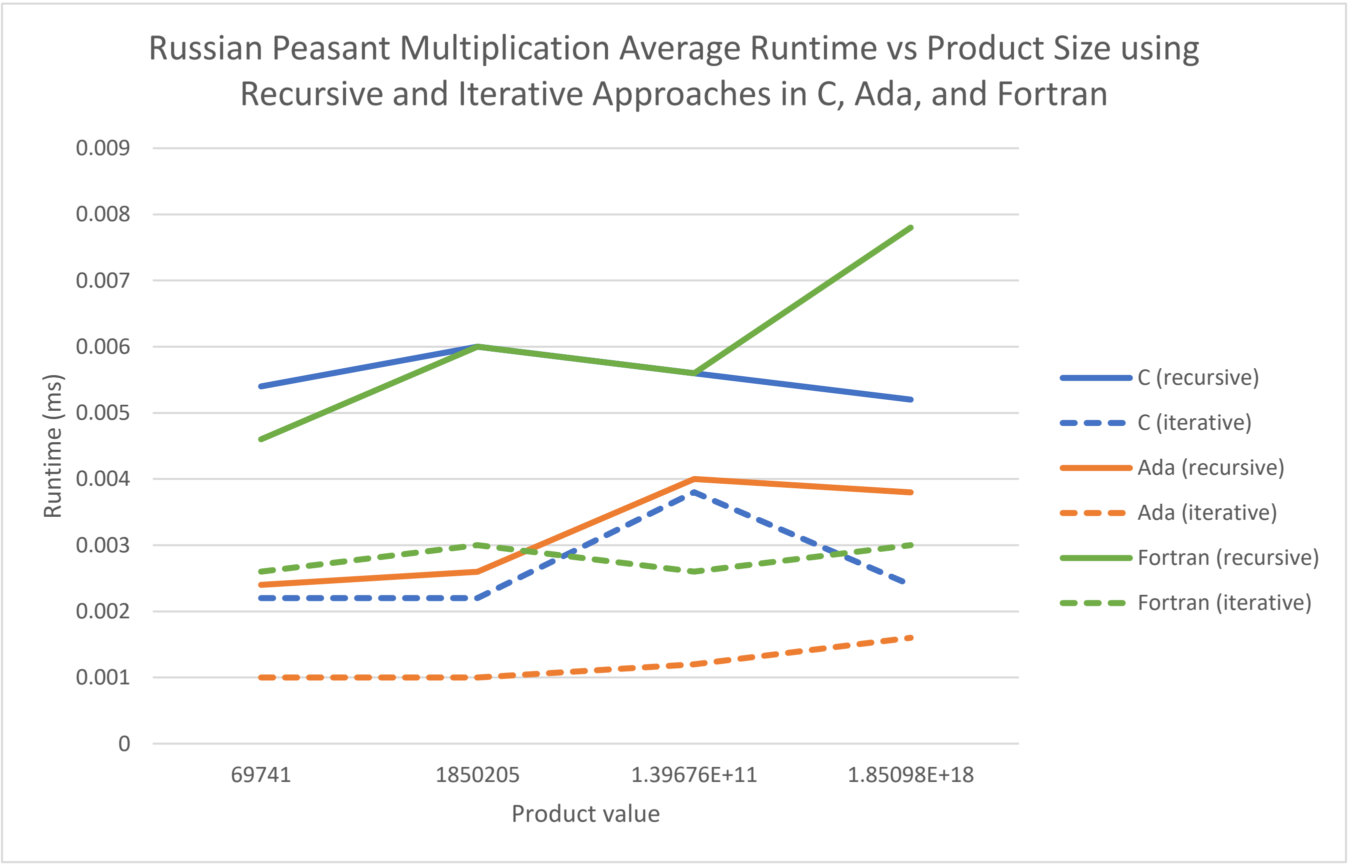
NOTE: I already wrote the Cobol program before the announcement that it is not needed for this assignment. While the code is included in this submission, it is not included in testing, which is shown below.

Testing – Results and Analysis

In order to test C, Ada, and Fortran's recursive and iterative approaches to the Russian Peasant Multiplication algorithm, I made a simple script to run each function 5 times per input-set (multiplier and multiplicand), using 4 different input-sets of different sizes, with products ranging from 70,000 to 1.85e18. I then used Excel to organize the data and calculate the average runtime of each function for each product. The results are shown below:

	AVERAGE RUNTIMES					
Product	C (recursive)	C (iterative)	Ada (recursive)	Ada (iterative)	Fortran (recursive)	Fortran (iterative)
69741	0.0054	0.0022	0.0024	0.001	0.0046	0.0026
1850205	0.006	0.0022	0.0026	0.001	0.006	0.003
1.397E+11	0.0056	0.0038	0.004	0.0012	0.0056	0.0026
1.851E+18	0.0052	0.0024	0.0038	0.0016	0.0078	0.003

The table was then graphed to make it easier to see the trends in the data:



Looking at the above graph, it is apparent that the iterative function of each language is significantly faster than its recursive counterpart, with the recursive functions being 2-3x slower in almost every run. We can also see that Ada's iterative function has the fastest runtime out of all the languages, and its recursive function is faster than the other recursive functions. On the slower end, we see that Fortran's recursive algorithm has peak runtime when given the largest inputs, while the other languages' recursive functions see little to no change in their runtimes.

Though the amount of data collected for this mini experiment is limited, it is clear that iteratively performing Russian Peasant Multiplication is much faster than doing it recursively. It is also clear that Ada runs this algorithm at a higher speed than C and Fortran across a wide range of product sizes. This comes as a surprise to me because Fortran is supposed to be a superior language for number processing and mathematical calculations and C is supposed to compile directly into highly efficient machine code. I suspect compiling with different flags may increase the speed of some of these programs, though I am not sure how much of an impact it would have on the overall results.

Dear Professor Wirth,

I would like to thank you for giving us a wonderful class and being such a resourceful, passionate instructor. This was easily one of the best courses I've taken in my four years at UoG. I hope you are faring well in the current crisis and I wish you a safe, enjoyable summer.

Regards,
Shawn Kaplan
