**Sintaxis y Semántica de los Lenguajes**

**Investigación de lenguajes: Rust y C#**

**Alison Salazar, Guido Biotti, Santiago Hamamura**

**Universidad Tecnológica Nacional, Facultad Regional Buenos Aires**

**1. Abstract.**

En el marco de la cátedra Sintaxis y Semántica de los Lenguajes, a cargo de la profesora Ing. Roxana Leituz, se seleccionaron dos lenguajes de programación cualesquiera comparables, para investigar y presentar sus Backus-Naur Form (BNF), junto con una breve historia de ambos. Luego, se midieron las diferencias de rendimiento entre estos mediante dos algoritmos a elección del grupo. La conclusión, en el presente documento, consta del informe de comparación de rendimiento (Benchmark) y la justificación de los resultados. Se encuentran adjuntos dos archivos (.pdf) con las BNF de los respectivos lenguajes y el código fuente utilizado para las pruebas.

**2. Primer Lenguaje: Rust**

Rust es un lenguaje multiparadigma, principalmente imperativo, y de medio-bajo nivel como C. Nació como proyecto personal de un empleado de Mozilla, Graydon Hoare, durante el 2006 [1]. Tres años después, en el 2009, la compañía decidió que el sistema tenía la suficiente robustez, se involucró y comenzó a patrocinarlo. Un año después, se comenzó a migrar del compilador inicial a uno auto-hospedado en rust, y en el 2011 se compiló exitosamente la primera versión del que está en uso actual, denominado *rustc.* Para entonces el proyecto se mantenía a puertas cerradas. En 2012 se publicó una pre-alfa (no estable), anunciando al lenguaje en desarrollo como “fuertemente tipado, con foco en la seguridad de memoria y la concurrencia” [2], principios que hasta día de hoy se mantienen y caracterizan al proyecto Rust. La primera versión estable, numerada 1.0, se publicó recién en 2015 [3]. Entre los grandes proyectos actuales que involucran Rust están componentes para el motor de búsqueda del navegador web de Mozilla, Firefox, en colaboración con Samsung. También algunos sistemas operativos y componentes relacionados emplean Rust, como

por ejemplo, un file system denominado Stratis, empleado por la distribución de Linux, Fedora. En general, los proyectos utilizando Rust requieren una manipulación fiable y sencilla de recursos de bajo nivel.

**3. Segundo Lenguaje: C#**

El siguiente lenguaje a tratar es C#, nacido en 2002 por mano del ingeniero Danés, Anders Hejlsberg, bajo contrato de Microsoft.

En los 90 predominó el lenguaje de programación Java, desarrollado por Sun Microsystems, e innovando por su capacidad de ejecutarse en diferentes sistemas operativos gracias a su sistema de compilación intermedia a bytecode [4]. Microsoft, con la misión de mantenerse vigente, y dado que la licencia de Java lo permitía, desarrolló una nueva IDE junto con un lenguaje, todo en base a la Java Virtual Machine (JVM). La IDE fue Visual J++, y su lenguaje J++. Se buscaba tener una sintaxis lo más parecida posible a la de su lenguaje padre, para así atraer desarrolladores Java, con la diferencia de que J++ solo funcionaría en el sistema operativo Windows. Esto último rompía con el lema de Java, “Write once, Run anywhere”, principio que se defendía legalmente en un conjunto de *tests* de compatibilidad con distintos sistemas operativos que las implementaciones sobre la JVM debían aprobar [5]. El conflicto resultó en una demanda millonaria por parte de Sun Microsystems contra Windows en 1997, junto con la discontinuación del proyecto J++ como tal.

Microsoft, tras insistentes negociaciones, contrató a Heljsberg en 1996, quien por entonces trabajaba para la empresa Borland, siendo responsable del compilador e IDE para Pascal, Turbo Pascal [6], junto con otros proyectos de éxito. El primer gran proyecto a cargo de Heljsberg en Microsoft, tras la discontinuación de Visual J++, fue la creación desde cero de un entorno de desarrollo exclusivo para Windows, llamado .NET Framework, conteniendo un compilador llamado “csc.exe”(C Sharp Compiler) y un lenguaje de programación bautizado como C#. Se trataba de un lenguaje que, al igual que J++, era sumamente parecido a Java [7]. Microsoft estaba decidido en que era la vía para atraer usuarios. C# era compilado por csc en un esquema de traducción intermedia a bytecode (llamado Código MSIL en .NET Framework) y luego a código máquina en un esquema “Just-In-Time”. Resultaba ser el mismo esquema de funcionamiento de Java sobre la JVM. Recién en 2016, se lanza .NET Core como un framework multiplataforma, admitiendo que C# pueda compilar sobre distintos sistemas operativos, y sustituyendo csc por *Roslyn*, el compilador vigente hasta el día de hoy. La estrategia de compilación y ejecución de C# se mantiene idéntico.

C# es multiparadigma, principalmente orientado a objetos. Dispone de un colector de basura para optimizar el uso de memoria. Por formar parte del framework multipropósito .NET, se trata de un lenguaje que suele presentarse en todo tipo de proyectos. Desde aplicaciones de escritorio o web, pasando por videojuegos y hasta inteligencia artificial y machine learning.

**4. Primer Algoritmo: Multiplicación de Matrices**

La primer prueba consta de tres bucles anidados para lograr la multiplicación de dos matrices dadas de 3x3 con valores numéricos enteros, así y obtener una tercer matriz resultado. Un pseudocódigo apropiado para describir el proceso es:

1. Definir matriz1[3][3] y matriz2[3][3]

2. Definir matriz3[3][3] resultado

3. Para i de 0 a 2:

Para j de 0 a 2:

matriz3[i][j] = 0

Para k de 0 a 2:

matriz3[i][j]+=matriz1[i][k] \* matriz2[k][j]

4. Mostrar Resultado

**5. Segundo Algoritmo: Ordenamiento Burbuja**

La segunda prueba consiste en realizar un ordenamiento de tipo burbuja sobre un vector de 10 números enteros, ordenados de mayor a menor, exigiendo que el vector resultado esté ordenado de menor a mayor. Así, estamos induciendo el mayor tiempo de tardado posible del algoritmo. Un pseudocódigo apropiado es:

1. Definir arreglo[10], orden de mayor a menor

2. Para i de 0 a 9:

Para j de 0 a 8-i:

Si arreglo[j] > arreglo[j+1], entonces:

Intercambiar(arreglo[j], arreglo[j+1])

3. Mostrar arreglo

Se trata de la versión “base” del algoritmo de ordenamiento burbuja, sin lógica de corte inteligente, propia del algoritmo *burbuja optimizado*.

**6. Benchmark**

Se procede a detallar los resultados de las pruebas de rendimiento (Fig. 1). Las implementaciones particulares de ambos algoritmos se encuentran adjuntos en forma de código fuente junto al presente documento. La versión del compilador de Rust, *Rustc*, corresponde con la 1.71.0. La versión del compilador integrado de .NET Core SDK, *Roslyn*, corresponde con la 7.0.109.

|  | Rust | C# |
| --- | --- | --- |
| Matrices |  |  |
| Ordenamiento |  |  |

*Fig.1. Tabla de Rendimiento*

**7. Interpretación de Resultados**

**Referencias**

1.Web.archive-rust-faq:

<https://web.archive.org/web/20160609195720/https://www.rust-lang.org/faq.html#project>

2.Reddit thread, rust pre-release: <https://www.reddit.com/r/programming/comments/opgxd/mozilla_and_the_rust_community_release_rust_01_a/>

3.Rust official release <https://blog.rust-lang.org/2015/05/15/Rust-1.0.html>

4.Java-bytecode <https://www.infoworld.com/article/2077233/bytecode-basics.html>

5. Sun Microsystems suing Microsoft <https://caselaw.findlaw.com/court/us-9th-circuit/1260682.html#:~:text=Sun%20filed%20suit%20against%20Microsoft,that%20failed%20Sun's%20compatibility%20tests>.

6.Turbo-Pascal. <https://www.seabrookewindows.com/9MgKzyqMd/>

7.C#-first-release-comments. <https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history>