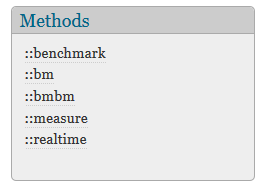
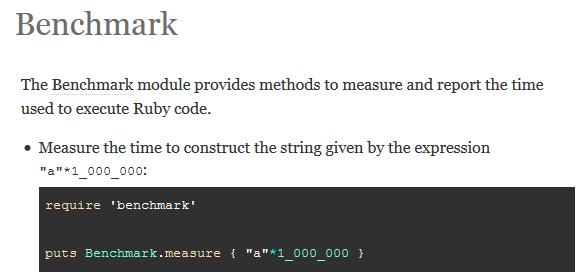


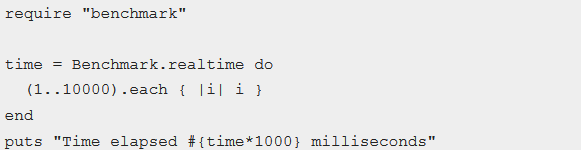
|  |
| --- |
| Optimización |
| Catedrático: Martha Martínez Moreno |
| **Por: Ernesto Álvarez Calderón**  **Jazmín Arlette Carrasco Cruz**  **Ricardo David Rojas Flores** |
|  |

Para esta unidad, se decidió implementar una calculadora en Ruby que fuera capaz de trabajar con expresiones más complejas. Por ejemplo (211\*22)+24. Y una vez implementada, optimizar el código a fin de reducir su tiempo de ejecución y consumo de recursos. En Ruby, la clase encargada de medir estos tiempos de ejecución, es la clase Benchmark, ya integrada en Ruby.





Nostros ocupamos el realtime, que como su nombre lo dice, mide en tiempo real (en segundos) el tiempo de ejecución de cualquier código. El modo de utilizar esta clase es de la siguiente manera.



Donde cualquier pedazo de código que esté entre do y end será examinado. Cabe mencionar que hay operaciones que incrementan considerablemente estos tiempos de ejecución, tales como las operaciones de entrada a través del teclado por métodos como gets. En nuestro caso, dicha calculadora se desarrolló bajo Ruby, a continuación se muestra el programa implementado en Ruby:

**CALC3**

**require 'benchmark'**

**puts "////////////////////////////RUBY CALCULATOR//////////////////////////"**

**puts "Ingresa una expresión"**

**es= gets.chomp**

**t1 = Benchmark.realtime do**

**nums=es.scan /^\(\d+[\+\-\\\*\/]+\d+\)[\+\-\\\*\/]+\d+/**

**puts "El resultado de #{nums} es: "+eval(es).to\_s**

**end**

**puts "/////////////////////////////////////////////////////////////////////"**

**puts "------------------------------------------------"**

**puts "La ejecución tomó "+(t1\*1000).to\_s+" milisegundos"**

Después de un par de optimizaciones podemos observar los cambios que sufre, junto con su tiempo de ejecución.

**CALC2**

**puts "La ejecución tomó "+(t1\*1000).to\_s+" milisegundos"**

**require 'benchmark'**

**puts "////////////////////////////RUBY CALCULATOR//////////////////////////"**

**puts "Ingresa una expresión"**

**es= gets.chomp**

**t1 = Benchmark.realtime do**

**nums=es.scan /^\(\d+[\+\-\\\*\/]+\d+\)[\+\-\\\*\/]+\d+/**

**res= eval(es)**

**puts "El resultado de #{nums} es: #{res}"**

**end**

**puts "/////////////////////////////////////////////////////////////////////"**

**puts "------------------------------------------------"**

**puts "La ejecución tomó "+(t1\*1000).to\_s+" milisegundos"**

**CALC1**

**require 'benchmark'**

**puts "////////////////////////////RUBY CALCULATOR//////////////////////////"**

**puts "Ingresa una expresión"**

**es= gets.chomp**

**t1 = Benchmark.realtime do**

**nums=es.scan /^\(\d+[\+\-\\\*\/]+\d+\)[\+\-\\\*\/]+\d+/**

**print "El resultado de #{nums} es: "**

**print eval(es)**

**print "\n"**

**end**

**puts "/////////////////////////////////////////////////////////////////////"**

**puts "------------------------------------------------"**

Tiempos de ejecución:

|  |  |
| --- | --- |
| Programa | Tiempo de ejecución en tiempo real |
| Calc3 | **0.95 ms** |
| Calc2 | **0.93 ms** |
| Calc1 | **3.42 ms** |

**Calc3:** En este programa el resultado requiere de la conversión explícita de un número enorme. Lo que se traduce en una cadena muy larga, por ende más procesamiento.

**Calc2:** En este método se almacena el resultado en una variable y posteriormente se emplea.

**Calc1:** Es el programa más lento, en este programa se imprime el valor directamente desde la evaluación y utilizando print, un método capaz de leer texto con un formato más rico como el de C.

La diferencia entre estos 2 últimos programas fue mínima, por lo que ambos serían una buena opción.

Sin embargo, contrastando lo anterior expuesto, podemos definir al programa **Calc2** como el más rápido y por lo tanto más optimizado.

**Vídeo: https://youtu.be/pioDPS6Zarw**

Tamaños del programa

C:\Users\HP\AppData\Local\Microsoft\Windows\INetCache\Content.Word\MWSnap001.png

**Rerefencias**

Ruby. (2014). Benchmark. 27 de Octubre del 2017, de Ruby Sitio web: http://ruby-doc.org/stdlib-2.0.0/libdoc/benchmark/rdoc/Benchmark.html