Algoritmo Recursivo diferente para hallar elementos de la serie de Fibonacci usando Programación Funcional

A new recursive algorithm to find Fibonacci´s Series elements using Functional Programming

Omar Ivan Trejos Buriticá, PhD

[1] *Ing de Sistemas, PhD Ciencias de la Educación, Universidad Tecnológica de Pereira, omartrejos@utp.edu.co*

Fecha de recepción: 14/07/2014 Fecha de aceptación del artículo: 10/02/2015

Resumen

Este artículo presenta un algoritmo que, usando recursividad, permite hallar los elementos de la Serie de Fibonacci utilizando programación funcional y bajo una lógica algorítmica diferente a la solución que, tradicionalmente, se ha presentado a este problema. La propuesta de solución es una puesta en escena de una solución innovadora que pretende darle significado al concepto de función y recursividad dentro de ciclo de formación en programación de ingenieros de sistemas. Se ha acudido a la programación funcional por su gran cercanía con la notación matemática y se ha planteado de forma sencilla y fácil de comprender a la luz del lenguaje de programación Scheme.

Palabras clave

Algoritmo, recursión, Fibonacci, programación funcional.

Abstract

This article shows an algorithm that, using recursion, let us find the elements of the Fibonacci Series using functional programming and an algorithmic solution different from the traditional solution for this problem. This is a new solution that brings a meaning to the concept of function and recursion in the computer programming in Systems Engineering formation cycle. We use functional programming because it is very close to math and it is a simple-and-easy algorithm.

Keywords

Algorithm, recursion, Fibonacci, Functional Programming

1. Introducción

Los problemas que plantean las matemáticas constituyen una fuente de enunciados casi inagotable para áreas como la programación de computadores. Con el tiempo, se han difundido determinadas soluciones a (también y valga la redundancia) determinados problemas y se han popularizado de una manera que prácticamente la academia se ha olvidado de buscar otras posibles soluciones o, al menos, de preguntarse si es posible encontrar soluciones alternas a dichos problemas.

El propósito de este artículo es presentar una solución alterna, diferente, al problema (ya resuelto en la programación) de encontrar los elementos de la serie de Fibonacci con bajas restricciones por el tamaño de los datos dado que, si es una serie progresiva, los datos son acumulativos y su solución requiere que se puedan manipular guarismos cada más amplios y menos restrictivos.

Lo novedoso de este artículo radica en que se aparta de la solución tradicional que aparece en los libros y en los sitios de programación para encontrar los elementos de la serie de Fibonacci y plantea una nueva forma de encontrarlos aprovechando, como es natural, la potencialidad de la recursión simple dentro del marco de la programación funcional.

Desde el punto de vista académico, toda búsqueda de nuevas formas de solución algorítmica a problemas tradicionales abre puertas para que el pensamiento, la lógica, la algoritmia, la programación y las matemáticas converjan para que el estudiante encuentre un espacio tanto de significado como de descubrimiento. Espacio de significado porque encuentra mucho sentido al conocimiento adquirido con la convergencia denotada y espacio de descubrimiento porque el estudiante puede palpar que dichas soluciones están al alcance de sus propios conocimientos yendo hacia soluciones simples en términos lógicos a pesar de ser recursivas.

La programación de computadores ha sido un espacio de conocimiento y academia desde donde se han planteado diferentes soluciones a problemas de la matemática por diferentes autores sin embargo pocas veces se encuentra que programación y matemáticas converjan en el mismo documento ya que, con frecuencia, el enfoque computacional utilizado por las matemáticas se queda en planteamientos abiertos o la minucia matemática se queda corta a la hora de implementar los algoritmos. Este artículo es una muestra de la relación tan cercana que existe entre las dos áreas de conocimiento, al fin y al cabo, la programación es (en muchos casos) la instrumentalización de las matemáticas aunque sean pocos los documentos que den fe de ello.

La limitante que se puede tener en la implementación de este tipo de soluciones, por ser procesos progresivos acumulativos, radica en el tamaño de los tipos de datos. Esa es la razón por la cual se ha optado por recurrir a la programación funcional y específicamente al lenguaje Scheme en el entorno DrRacket versión 5.1 pues involucra un manejo de datos insospechado y demasiado útil, tal como se demostrará en la parte de aplicación en el numeral de Metodología.

El objetivo general de este artículo va en el sentido de plantear una solución nueva y diferente al tradicional problema de generar, hasta un tope definido por el usuario, los elementos de la serie de Fibonacci con pocas restricciones de almacenamiento en los números generados. Para elaborar el artículo se debió acudir a los conceptos propios de la teoría del aprendizaje significativo, teoría del aprendizaje por descubrimiento, la Serie de Fibonacci, el concepto de Función, la programación funcional, el planteamiento del algoritmo recursivo tradicional y las restricciones en tipos de datos.

En cuanto a la metodología usada, primeramente se estudió la naturaleza y formulación matemática de la serie de Fibonacci, luego se acudió a la consulta de la solución tradicional que se ha planteado para resolver este problema en los libros de programación y finalmente se ideó una solución que fuera completamente diferente a la tradicional pero que, al tiempo, cumpliera con el objetivo de resolver el problema dentro de los parámetros de eficiencia y complejidad de la solución tradicional.

Este artículo es uno de los productos del proyecto de investigación "Análisis pedagógico, instrumental y conceptual de algunos paradigmas de programación como contenido de la asignatura Programación I del programa Ingeniería de Sistemas y Computación" tramitado ante la Vicerrectoría de Investigaciones, Innovación y Extensión de la Universidad Tecnológica de Pereira.

A manera de hipótesis se plantea en este artículo que, recorriendo los caminos de la algoritmia, siempre es posible encontrar una solución a los problemas que la matemática provee independiente de las soluciones tradicionales que los libros ya hayan encontrado y que esta solución puede moverse dentro de unos linderos de eficiencia y complejidad razonables.

El artículo se ajusta a la metodología IMRYD, es decir, se presenta una introducción seguida de una teoría relevante para el artículo; posteriormente se explica la metodología utilizada tanto en su descripción como en su aplicación, se plantean unos resultados obtenidos y a partir de allí se abre una discusión y unas conclusiones. Finalmente se relaciona la bibliografía utilizada como base de consulta para este artículo.

2. Teoria

2.1 Aprendizaje significativo

La teoría del aprendizaje significativo constituye la base de los procesos de formación moderna si se tiene en cuenta que razones naturales obligan al cerebro a buscar significado a todo aquello que captan sus sentidos, incluyendo los conceptos, teorías, métodos y formulaciones que se derivan de los procesos de aprendizaje. El aprendizaje significativo es un modelo a través del cual se le da alta relevancia al significado basado en tres fundamentos: el conocimiento previo, el nuevo conocimiento y la actitud del estudiante que, a su vez, tiene dos pilares: la motivación del estudiante y la capacidad de relacionar el conocimiento previo con el nuevo conocimiento [1].

Según lo planteado por el Dr. David Paul Ausubel, la base del aprendizaje se resume en el concepto de significado y lo más importante dentro de un proceso de aprendizaje es lo que el alumno ya sabe pues es lo que le permite establecer una relación entre lo que conceptúa y lo que vive. El conocimiento que se califica como "nuevo" es un conocimiento que proviene de las teorías, los conceptos y los planteamientos que no se conocían, no se sabían, no se había accedido a ellos o no se habían aprendido lo cual implica asimilación y adecuación.

Las relaciones entre el conocimiento previo y el nuevo conocimiento pueden ser de complementación, de suplantación, de renovación o de innovación [2] de forma que el conocimiento previo sea el vehículo que le conceda significado al nuevo conocimiento. Según el autor de la teoría, el Dr. Ausubel, el ser humano aprende mucho más fácil todo aquello que tiene significado para él, es decir, todo aquello que puede servir como puente entre lo vivido y lo concebido.

2.2 Aprendizaje por descubrimiento

Corresponde a la teoría que resume el planteamiento del Dr. Jerome Seymour Bruner según el cual un proceso de aprendizaje involucra tres fases: por un lado se describe la adquisición de la información nueva, por otro lado está el proceso de manipulación del conocimiento para adecuarlo a nuevas aplicaciones y a la comprobación de la manera como se ha adecuado dicha información constituye la tercera de las fases.

Con la unión de estos tres procesos se despliegan las habilidades cognitivas de alto nivel aunque aparecen implicaciones de orden secundario: en la adquisición se debe tener en cuenta lo que se conoce como la confiabilidad de las fuentes, por otra parte debe considerarse también la capacidad de aplicar dicha transformación en la relación con el conocimiento tal que permita aplicarlo a nuevas tareas. En la evaluación es necesario que existan unos elementos de juicio suficientemente sólidos que permitan la verificación en la transformación del conocimiento adquirido y su pertinencia dentro de un determinado contexto de necesidades.

En esta teoría, la experimentación tiene un espacio de gran importancia y, con ella, el descubrimiento como el gran fundamento a través del cual el estudiante es capaz de construir su propio conocimiento. Según el Dr. Bruner, es claro que el ser humano puede aprender mucho más fácil todo aquello que, para él, tiene significado y de esta forma se concede especial importancia a la exploración, a la experimentación, al ensayo y al error como los elementos que facilitan el camino para el aprendizaje en tiempos modernos [3].

En el aprendizaje por descubrimiento se destaca la capacidad comunicativa como una de las aristas de gran importancia dentro de la teoría. El aula sigue siendo un entorno de aprendizaje aunque, en nuestros tiempos, no es el único. Por lo tanto, considera el autor que un buen marco de premios y castigos, siendo bien diseñado, es el camino para que el aprendizaje se dé dentro de un espacio motivacional suficiente como para que el alumno quiera aprender [4]. La línea divisoria entre los castigos y los premios es bastante delgada por eso requiere particular cuidado.

La naturaleza misma del cerebro le concede a lo insólito y a la fascinación un espacio de gran importancia [5]. El descubrimiento pocas veces

está distanciado de estos dos factores que son los que motivan al estudiante a encontrar soluciones, respuestas, formulaciones y planteamientos que resuelvan las situaciones que se les plantea. El hecho de que el estudiante tenga la sensación de que ha descubierto algo y que los elementos que han intervenido han sido la fascinación y lo insólito lleva a pensar que el proceso de aprendizaje está servido.

Todo lo insólito tiene un sentido especial hacia el conocimiento que no siempre es controlable, especialmente cuando se vive entre las relaciones entre seres humanos y nuevas tecnologías de información y comunicación. Para Bruner, descubrimiento, fascinación e insólito parecieran ser caras de un mismo solido.

2.3 Serie de Fibonacci

En las matemáticas, la llamada Serie de Fibonacci (más apropiado, sucesión de Fibonacci) consiste en una serie infinita que se compone de números enteros y que esta construida sobre la siguiente premisa: se inicia con los números 0 y 1 y, sucesivamente, se van sumando los dos últimos números para generar el siguiente [6]. De esta manera la serie de Fibonacci estará conformada por los números:

0 1 1 2 3 5 8
13 21 34 55 89 …

En donde se puede notar que cada número, exceptuando los dos primeros, es el resultado de sumar los dos anteriores. A cada uno de los elementos de esta sucesión se les conoce con el nombre de números de Fibonacci.

Esta sucesión de números fue planteada por el matemático Leonardo de Pisa (quien se hacía llamar Fibonacci) en el continente Europeo en la primera mitad del siglo XIII. En ciencias de la computación tiene grandes aplicaciones así como en las matemáticas y en la llamada Teoría de Juegos [7]. La lógica de la serie de Fibonacci aparece en una gran cantidad de situaciones de la naturaleza, en el mundo de la biología y de la química, en el surgimiento de los ramas de los arboles, en la forma como están dispuestas las hojas en un tallo, en los pétalos de las flores y en las caracteristicas geométricas de diferentes sólidos.

La inquietud acerca de la progresión de estos números en sucesión le surge al matemático Fibonacci a partir de notar el comportamiento de la reproducción en los conejos y la manera como estos se van multiplicando. La progresión de la serie es completamente exponencial y, al graficarla, se pueden realizar otras observaciones de orden matemático que se salen del contexto de este artículo. La relación entre dos números sucesivos de la serie de Fibonacci se aproxima al número áureo [8] y algunas composiciones musicales de alto nivel han incorporado en su estructura esta serie; así sucede con compositores como Bartok y Messiaen.

Matemáticamente la serie se define de la siguiente forma. La serie de Fibonacci F es una sucesión de números enteros F_i que cumplen con las siguientes condiciones:

$F_0 = 0$
$F_1 = 1$
$F_n = F_{n-1} + F_{n-2}$

Es de anotar que esta definición corresponde plenamente a la matemática discreta. La sucesión de Fibonacci, por sus características, encarna algunas propiedades que han sido muy útiles en el desarrollo de aplicaciones en diferentes áreas. Entre ellas se pueden contar:

- El cociente entre dos elementos consecutivos de la serie varía de manera permanente pero tiende a estabilizarse en el número áureo
- Un número natural puede ser escrito mediante la suma de un número determinado y diferente de términos de la sucesión de Fibonacci
- Cada elemento F_n de la serie es igual a promediar el término que se encuentra dos posiciones antes (F_{n-2}) y el término que se encuentra inmediatamente después (F_{n+1})
- El resultado de sumar los primeros n elementos de la serie origina el número que ocupa la posición n+2 quitándole el valor 1

- Si se calcula el máximo común divisor de dos números de Fibonacci, se obtiene otro número de la misma serie de Fibonacci
- Si un número F_p de la serie de Fibonacci es igual a un valor x y el valor x es un número primo, entonces el valor p también será un número primo

Se han encontrado una gran cantidad de identidades sobre la serie de Fibonacci, sobre los elementos que la componen y sobre la relación entre dichos elementos y otros conjuntos de números caracterizados por la matemática, sin embargo dichas identidades se salen de los objetivos de este artículo pero serán tratados computacionalmente en otros artículos del mismo autor.

2.4 El concepto de Función

Un programa que se quiera construir bajo la perspectiva "divide y vencerás" ha de apoyarse en el concepto de función que puede definirse, en su concepción más simplificada, como el conjunto de instrucciones que se identifica con un nombre (único o no único), que puede recibir argumentos o parámetros (dependiendo del paradigma) y que puede retornar valores bien sea de manera automática o de manera explícita tal como la programación funcional y la programación estructurada lo permiten, respectivamente [9].

Este concepto de función es el camino para simplificar el proceso de construcción de una solución posibilitando que un objetivo complejo, por la vía de la atomización, se puede subdividir en pequeños objetivos, cada uno más simple, y que posteriormente se pueda resolver el problema original a través del enlace de dichos pequeños objetivos. Precisamente es la función la unidad que permite resolver un pequeño objetivo.

El paradigma funcional (que se basa en la matemática derivada del cálculo Lambda de Church) se basa en la función como elemento primordial para que matemática y programación encuentren un lugar común en donde converger. La función no solo es la base nuclear de la programación funcional, también es la gran herramienta para capitalizar toda la potencialidad de la programación estructurada y es el concepto que subyace a los métodos en la programación estructurada.

2.5 Programación Funcional

La programación funcional es un paradigma de programación que destaca la gran importancia que tiene el concepto de función en la concepción, diseño y desarrollo de soluciones simples y fácilmente mantenibles, suficiente y apropiadamente atomizadas y rápidamente entendibles. La programación funcional permite hacer efectiva la filosofía "divide and conquer" (divide y vencerás) para llegar a la simplificación de programas que, en su concepción general, podrían ser de alta complejidad [10].

Dentro del marco de la programación funcional, los programas son simplemente colecciones de funciones simples que, debidamente enlazadas, permiten resolver problemas complejos y atender algorítmicamente objetivos generales a partir del cumplimiento riguroso de sus objetivos específicos atomizados. En la programación funcional, todo gira alrededor del concepto de función que se constituye en su unidad de trabajo más importante.

Alonzo Church, profesor de la Universidad de Princeton y gran apasionado por las soluciones teóricas que la matemática abstracta provee, se dedicó a plantear los fundamentos de una aritmética que llamó el cálculo Lambda (λ Calculus) apoyado en el poder que la tecnología computacional podía brindar. Esta nueva concepción matemática fue apoyada y cristalizada por el matemático Alan Turing quien sentó las bases de la computación moderna.

La programación funcional concede gran énfasis a la función [11] a diferencia de la programación estructurada que enfatiza en los cambios de estado de las variables. A pesar de que los lenguajes de programación funcional tienen una perspectiva mucho más académica, la industria ya ha comenzado a aceptar este tipo de lenguajes, este paradigma y el concepto de función como base para desarrollos tecnológicos. El paradigma funcional, como filo-

sofía de programación, puede ser capitalizado y aprovechado al máximo en los lenguajes no funcionales, tales como los lenguajes estructurados y los lenguajes totalmente orientados a objetos.

2.6 Algoritmo recursivo tradicional

Los libros de programación, con mucha frecuencia, citan la serie de Fibonacci y la generación de sus elementos como uno de los ejemplos excelsos de programación especialmente cuando se trata de aplicar el concepto de recursividad (característica que tiene una función de llamarse a sí misma). La solución recursiva que, tradicionalmente, se ha conocido es la siguiente:

Algoritmo Version Recursiva

```
1  Funcion Fibo ( x )
2  Si x < 2 entonces
3       Retorne x
4  Sino
5       Retorne (Fibo (x – 1) + Fibo (x – 2))
6  Fin Si
7  Fin Funcion
```

Es observable que esta función acude a la recursividad como mecanismo de cálculo de los elementos de la serie. Para facilitar una breve explicación se han numerado las líneas del código. En la línea 1 se observa la declaración abstracta de la función según la cual ésta se llama Fibo y recibe un argumento valor.

En la línea 2 se establece un condicional según el cual si el valor almacenado en x es menor que 2 entonces que se retorne el mismo valor x y, en la línea 4, reza que si esta condición no es verdadera, es decir, si el valor almacenado en x no es menor que 2 entonces que retorne la suma de calcular lo que retorne la función enviándole como argumento el valor anterior de x (x – 1) mas lo que retorne esa misma función Fibo enviándole como argumento el valor trasanterior de x (x – 2).

Como se ha explicado este es el algoritmo que tradicionalmente aparece en los libros de programación y en las definiciones recursivas de la serie de Fibonacci de los libros de matemáticas. Precisamente lo que se busca en este artículo es presentar una opción diferente a la tradicional de manera que, sin desconocer la utilidad de la recursividad, se pueda obtener un resultado con un nivel de rendimiento, eficiencia y complejidad similar al del algoritmo tradicional.

2.7 Restricciones en tipos de datos

Debe tenerse en cuenta que se ha escogido el lenguaje Scheme y su entorno DrRacket debido a que las limitaciones en el tratamiento de datos enteros primitivos en lenguajes de programación más comerciales resulta ser un inconveniente cuando se trata de este tipo de algoritmos. Si bien es cierto que desde lo puramente matemático no existen restricciones en la longitud de los datos de una serie, en lo computacional estas restricciones incorporan una serie de fronteras que no existen en los planteamientos teóricos.

Ha sido precisamente ese valor agregado el que ha permitido que se escoja el lenguaje Scheme que, por sus caracteristicas tan próximas a la notación y la concepción matemática, posibilita un tratamiento más libre de los datos y, en este caso, permite conocer la progresión de la sucesión de Fibonacci allende las fronteras que los otros lenguajes posibilitan.

Si bien el algoritmo puede ser implementado en cualquier otro lenguaje y desde cualquier paradigma, resulta ser de gran utilidad acudir al lenguaje Scheme y a la programación funcional como mecanismo de acceso a soluciones que impliquen cálculos progresivos en donde los límites de almacenamiento sean menos restrictivos.

Por tomar un ejemplo se cita el caso del lenguaje de programación C en el cual para almacenar datos enteros de más de 20 dígitos debe acudirse a los algoritmos y recursos que proveen las estructuras de datos (tales como las listas simples) para poder realizar cálculos de manera apropiada dado que sus tipos de datos primitivos no proveen estas facilidades de almacenamiento.

Scheme como lenguaje de programación tiene una frontera que va mucho más allá de lo convencional y por tanto el tratamiento de datos de más de 20 dígitos termina siendo un tratamiento natural dentro del contexto teórico, matemático y algorítmico propio de la programación funcional.

3. Metodologia

3.1 Descripción

El algoritmo se describe desde la óptica del código propio del lenguaje de programación Scheme y, a la par, dentro de los estándares que implica la utilización de seudocódigo como comentario anexo a cada línea lógica del programa. Es de anotar que en este lenguaje la aparición del signo punto y coma (;) repetido representa el inicio de una línea de comentario, es decir, de una línea que no es procesado por el compilador.

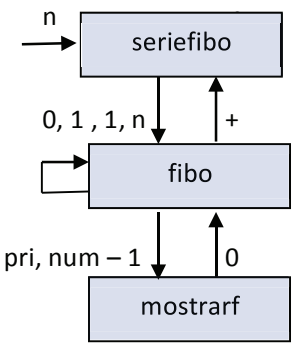
```
;;============================
;; GENERACION SERIE DE FIBONACCI
;; Genera serie Fibonacci hasta el n-esimo termino
;; para un n recibido como argumento
;;============================
;; Función que muestra un elemento de la serie
;; con su respectivo número con un ordinal
;; ****************************************
(define (mostrarf numf num)  ;; Definición función
 (display «Elemento «)  ;; Título
 (display (+ num 1))    ;; Ordinal elemento serie
 (display «o.....»)     ;; Título
 (display «Num Fibonacci «)    ;; Título
 (display numf)         ;; Muestra elemento serie
 (newline)              ;; Nueva línea
 0                      ;; Retorno de la función
 )                      ;; Fin función mostrarf
;; Función que genera la serie de Fibonacci
;; ****************************************
;; pri    = 1o elemento serie (siempre es igual a 0)
;; seg    = 2o elemento serie (siempre es igual a 1)
;; num    = Contador de los elementos de la serie
;; tope   = LimSup hasta donde se genera la serie
;; ****************************************
*******
(define (fibo pri seg num tope) ;; Definición función
;; Si se superó al tope solicitado por el usuario
 (if (> num tope)
 0                  ;; retornar 0
;; Sino sumar lo que retorne la función mostrarf
;; mas lo que retorne la misma función fibo
 (+ (mostrarf pri (- num 1))
 (fibo seg (+ pri seg) (+ num 1) tope)))
 )                  ;; Fin función fibo
;; Función que inicia el proceso recibiendo como
;; argumento el ordinal máximo del elemento
;; de la serie de Fibonacci
;; ****************************************
;; Esta función tiene por objetivo independizar al
;; usuario de los requerimientos de la función fibo a
;; nivel de argumentos
;;****************************************
(define (seriefibo n)     ;; Definición función
 (fibo 0 1 1 n)  ;; Llamado a la función fibo
 )        ;; Fin función seriefibo
```

3.2 Aplicación

Se presenta en la figura 1 el esquema funcional de este programa con el ánimo de que, a partir de él, se interpreten tanto el uso de las funciones como sus interrelaciones y sus retornos.

La potencialidad del concepto de función en medio del paradigma de programación funcional radica en la manera como se atomiza un objetivo y se resuelve, vía diseño e implementación de funciones eficientes, a partir de las conexiones que se establezcan entre ellas.

Figura 1. Esquema Funcional



Fuente: El autor

Como se puede observar en la figura 1, la función seriefibo recibe el número ordinal n que el usuario ordena usar para conocer el n-ésimo término de la serie de Fibonacci. Eso significa que la función seriefibo es la función que se debe llamar desde la línea de comandos. Esta función invoca a la función fibo a la cual le envía en primera instancia los valores iniciales de la serie y el tope hasta donde debe llegar. La función recursiva fibo llama a la función mostrarf que se encarga de mostrar cada uno de los elementos de la serie de Fibonacci. Cuando ha terminado, esta función mostrarf retorne el valor 0. Al finalizar la función fibo se retorna la suma de los valores retornados por la función mostrarf y como esta retorne siempre un valor 0 entonces la suma será igual a 0 siempre. Esta técnica se usa como comodín para que se pueda implementar una función operativa recursiva bajo los parámetros algorítmicos que la misma recursividad implica.

4. Resultados

La parte más interesante de este algoritmo está en el rendimiento de su ejecución pues, cuando se invoca a la función que oficia como primera, se obtienen los resultados que se presentan a continuación. Si se invoca la función seriefibo con argumento 4 indicando que se quieren conocer los primeros cuatro elementos de la serie de Fibonacci, los resultados son:

> (seriefibo 4)

Elemento 1o.....Num Fibonacci 0

Elemento 2o.....Num Fibonacci 1

Elemento 3o.....Num Fibonacci 1

Elemento 4o.....Num Fibonacci 2

Si se invoca la función seriefibo con argumento 13 indicando que se quieren conocer los primeros trece elementos de la serie de Fibonacci, los resultados son:

> (seriefibo 13)

Elemento 1o.....Num Fibonacci 0

Elemento 2o.....Num Fibonacci 1

Elemento 3o.....Num Fibonacci 1

Elemento 4o.....Num Fibonacci 2

Elemento 5o.....Num Fibonacci 3

Elemento 6o.....Num Fibonacci 5

Elemento 7o.....Num Fibonacci 8

Elemento 8o.....Num Fibonacci 13

Elemento 9o.....Num Fibonacci 21

Elemento 10o.....Num Fibonacci 34

Elemento 11o.....Num Fibonacci 55

Elemento 12o.....Num Fibonacci 89

Elemento 13o.....Num Fibonacci 144

Si se quiere conocer el elemento 1000º de la serie de Fibonacci, entonces se invoca la función seriefibo de la siguiente manera (por razones prácticas se muestran solo los últimos tres resultados):

.

.

.

Elemento 998o.....Num Fibonacci

1026106236203326233660492672924522213266855812060212427776462290569940798254671148827285946888745795908773311924256407785074365766118082732679853917775891982813511440749936979646564952426675539110499009912 0377

Elemento 999o.....Num Fibonacci

1660274766245209704954180047289770183494805119838482806235855309191857371770117020106551018559589860510409473691887927846223301598102952299783631123261876053919903676539979992673143323971886037334508837505424 9

Elemento 1000o.....Num Fibonacci

26863810024485359386146727202142923967616609318986952340123175997617981700247881689338369654483356564191827856161443356312976673642210350324634850410377680367334151172899169723197082763985615764450078474174626

Es notoria la facilidad de manejo de datos numéricos, en cuanto a la cantidad de dígitos, con la cual trabaja Scheme. Esto lo hace mucho más cercano a la matemática conceptual y menos atado a las restricciones tecnológicas computacionales que limitan todos los demás lenguajes de programación.

De otra parte el desarrollo de operaciones entre números enteros, prácticamente sin límite de extensión, permite que se puedan manejar datos de una extensión casi inimaginable lo cual aproxima mucho más la programación de computadores a las bases conceptuales de la matemática.

5. Discusión

Las matemáticas constituyen uno de los grandes pilares de la ingeniería y en la formación de ingenieros el área de las ciencias básicas constituye ese espacio de conocimiento desde donde se proveen las herramientas para interpretar, modelar, describir, analizar, optimizar e intervenir el mundo que nos rodea. Hasta el momento parecieran ser las matemáticas ese camino que permita conformar, en los futuros ingenieros, el pensamiento científico con las implicaciones que ello conlleva. La pregunta podría ser ¿están cumpliendo las matemáticas ese papel en los procesos de formación en ingeniería? la evaluación de su papel es muy importante dado que desde ellas se heredan las herramientas para que los ingenieros tengan el sustento teórico y conceptual que posibilita la interpretación del mundo desde una óptica científica.

En un sentido muy aproximado, y más orientado para ingenieros de sistemas aunque no lejano a las demás ingenierías, está la programación de computadores que se constituye en otro gran puente de comunicación entre las necesidades de la sociedad y las facilidades que las tecnologías modernas proveen. La programación de computadores busca sembrar en el ingeniero la lógica computacional como camino algorítmico para resolver problemas en donde los computadores, y sus dispositivos y servicios asociados, tengan cabida.

La posible distancia que se presenta entre los procesos de formación en ingeniería y el pensamiento científico derivado de las matemáticas radica en lo difuso que es el significado de las matemáticas y sus aplicaciones en ingeniería. Si bien es cierto que todo en ingeniería gira alrededor de las matemáticas, también lo es que por un lado los estudiantes conocen unos métodos y unas teorías derivadas de las matemáticas y que, por otro lado, conocen la base teórica y aplicativa de su área disciplinar y que la relación entre unos y otros no es tan directa ni tan concreta como debería ser.

En muchos casos, las matemáticas dejan de ser ese gran bastión a partir del cual el futuro ingeniero forma su pensamiento científico para constituirse simplemente en un conjunto de asignaturas inconexas o con débil conexión con el área disciplinar. La programación de computadores, en el caso de la ingeniería de sistemas, es el área de conocimiento que cristaliza el significado de las matemáticas y que provee los elementos de juicio para encontrarle aplicabilidad a partir del pensamiento algorítmico.

De allí podemos inferir que la teoría del aprendizaje significativo tiene mucho para aportarle al desarrollo del pensamiento científico toda vez que, entre la programación como camino algorítmico de solución de problemas y las matemáticas como fundamento para la constitución del pensamiento científico, se encuentre, se aplique, se evalúe y se retroalimente el significado del conocimiento a partir de la complementación de ambas áreas.

De la misma forma podría decirse que, partiendo del sustento teórico que provee la teoría del aprendizaje por descubrimiento, lo insólito y lo fascinante podrían ser elementos que apoyan la convergencia entre matemáticas y programación, estableciendo caminos más llanos para adquirir, asimilar, apropiar, aplicar y evaluar el conocimiento aplicado

alrededor de estos temas. Es de anotar que las nuevas tecnologías son expresiones electrónicas que afianzan lo fascinante en los jóvenes de hoy.

Este artículo presenta, precisamente, una forma de concederle significado a las matemáticas a partir de la programación y, al mismo tiempo, posibilita la incorporación de lo insólito y fascinante, derivado de la teoría del aprendizaje por descubrimiento, como elemento que hace que programación y matemáticas sean una sola y generen motivación en el futuro ingeniero.

6. Conclusiones

Siempre es posible encontrar un camino para que las ciencias básicas, mayormente representadas por las matemáticas, encuentren un significado que facilite su aprendizaje y su aplicación

La programación de computadores se constituye en un espacio excelso que permite conceder significado a la aplicación y apropiación de los conceptos y teorías derivados de la matemática

La motivación por el aprendizaje constituye uno de los grandes bastiones de soporte de la teoría del aprendizaje significativo y se apoya en el hallazgo efectivo del significado del conocimiento

Lo insólito y lo fascinante hacen que la motivación emerja casi de manera automática y que el cerebro automáticamente quiera encontrar, asimilar, apropiar y evaluar nuevos conocimientos

Siempre es posible encontrar nuevas formas de resolver problemas matemáticos desde la programación, al margen de las soluciones tradicionales que se exponen en los libros

La teoría de números es un excelente espacio teórico para aplicar los conceptos algorítmicos que la programación de computadores provee

La programación funcional puede constituirse en una subárea de la programación que facilita mucho más la solución de problemas derivados de las matemáticas

Poder trabajar con números enteros sin límites de almacenamiento posibilita la formulación de algoritmos más generales y más entendibles

Referencias

Ausubel, P. (2010). *The acquisition and retention of knowledge*. Publisher: Springer Netherlands. NY, USA. ISBN-13: 9789048155361.

Medina, J. (2010). *Los 12 principios del cerebro*. Grupo Editorial Norma. Santafé de Bogotá. ISBN: 9789584528971

Bruner, J. (1969). *Hacia una teoría de la instrucción*. Manuales Uteha, No. 373, México, Editorial Hispanoamericana. ISBN: 84-7112-179-4

Bruner, J. (1991). *Actos de significado, Más allá de la revolución cognitiva*. Madrid, Alianza Editorial. ISBN: 9788420648125

Small, G. (2010). *El cerebro digital*. Editorial Urano. Barcelona. ISBN 10: 8479537159

Mora, W. (2010). *Introducción a la teoría de números*. Escuela de Matemática. Instituto Tecnológico de Costa Rica. ISBN: 978-9968-641-11-1

Ivorra, C. (2010). *Teoría de Números*. Universidad Politécnica de Valencia. España. Editorial Sanz y Torres, S. L. ISBN 84-88667-00-0

Devlin, K. (2002). *El lenguaje de las matemáticas. MaNon Troppo*, Ediciones Robinson. Barcelona. España. ISBN: 84-95601-30-3

Trejos, O. (2000). *La esencia de la lógica de programación*. Centro Editorial Universidad de Caldas. Manizales, Colombia. ISBN: 958-33-1125

Van Roy, P. (2003). *Concepts, techniques and models of Computer Programming*. Universidad Católica de Lovaine. Swedish Institute of Computer Science. Switzerland. ISBN 0-262-22069-5

Rivadera, G. (2008). *La programación funcional: un poderoso paradigma*. Cuadernos de la Facultad No. 3. Universidad Católica de Salta, UCASAL, Buenos Aires (Argentina). ISSN: 1852-7094.