AVANCES: INVESTIGACIÓN EN INGENIERÍA • ISSN: 1794-4953 (Impreso) • ISSN: 2619-6581 (En línea) • Vol. 14 • DOI: doi.org/10.18041/avances.v14i1 • pp 63 - 76 (2017)

Metodología algorítmica para construir funciones que resuelvan cálculos basados en procesos simples usando programación funcional

Algorithmic Methodology to build functions to solve processes with simple calculations using two programming paradigms

Omar Iván Trejos Buriticá[1]
[1] *Universidad Tecnológica de Pereira, Pereira, Colombia, omartrejos@utp.edu.co*

Fecha de recepción: 06/09/2017
Fecha de aceptación del artículo: 28/10/2017

DOI: doi.org/10.18041/1794-4953/avances.1.1284

Cómo citar:
Trejos Buriticá, O. (2017). Metodología algorítmica para construir funciones que resuelvan cálculos basados en procesos simples usando Programación Funcional. AVANCES: Investigación en Ingeniería, 14, 62-75. doi.org/10.18041/1794-4953/avances.1.1284.

Resumen

El presente artículo presenta una metodología algorítmica para construir funciones que resuelvan cálculos, basándose en procesos simples usando sumas y restas apoyándose en dos paradigmas de programación: la programación funcional y la programación imperativa, usando los procesos recursivos y cíclicos, respectivamente, que cada una provee. El propósito de esta investigación es demostrar lo sencillo que es resolver algunos problemas muy conocidos a partir de las operaciones más simples que se conocen y para las cuales el computador está acondicionado a calcular, de forma que los estudiantes conciban la solución a un problema a través de sus propios conocimientos para un aprendizaje significativo y con sentido. En la parte metodológica, se les hizo una presentación a los estudiantes de lo que se iba a realizar, los propósitos, los alcances y los logros que se esperaban para que, de una forma comparativa, pudieran formularse soluciones algorítmicas e implementarse en dos paradigmas de programación. Se encontró que los alumnos confieren suprema importancia a la relación entre el nuevo conocimiento que se les explica y sus nexos con los conocimientos ya adquiridos y que sea cual fuere la solución a implementar, a ésta le subyacen modelos que son independientes de la tecnología. La investigación concluye que cuando los estudiantes conocen la metodología con la cual se va a abordar un nuevo conocimiento, la siguen rigurosamente de la mano del docente, la codifican y comprueban la efectividad de sus resultados, el aprendizaje se alcanza más fácilmente.

Palabras claves: Algoritmo, Metodología, Programación de computadores, Programación funcional, Programación imperativa

Abstract

This article presents an algorithmic methodology to construct functions to solve calculations based on simple processes using addition and subtraction, on two programming paradigms: functional programming and imperative programming using recursive and cyclical processes, respectively, each provides. The purpose of this research is to demonstrate how easy it is to solve some well-known problems from the simplest operations that are known and for which the computer is conditioned to calculate, so that students devise a solution to a problem through their own knowledge for meaningful learning and sense. In the methodological part, we made a presentation to students of what was to be held, the purpose, scope and achievements expected so that a comparative basis, could be formulated an algorithmic solution and implemented in two programming paradigms. We found that students give a especial importance to the relationship between new knowledge that is explained and its links with the already acquired knowledge and that whatever the solution to be implemented, and it has models that are independent of technology. The research concludes that when students know the methodology which will address new knowledge, strictly follow the teacher's hand, the coding and check the effectiveness of their learning outcomes more easily reached.

Keywords: Algorithm, Computer programming, functional programming, imperative programming, Methodology.

1. Introducción

En términos de la programación de computadores, el concepto de función constituye uno de los pilares más importantes para la comprensión de los diferentes paradigmas de programación según Trejos (2000) [1]. No en vano la función es la esencia de la programación funcional, es el concepto que simplifica la concepción de soluciones imperativas y que se convierte la función en parte constitutiva de la definición de una clase bajo el nombre de método [2], por citar tan solo tres paradigmas de programación, tal vez, los más comunes.

El presente artículo encuentra su justificación en el hecho de buscar tanto aplicaciones cada vez más prácticas y cercanas del estudiante al concepto de función como la implementación de éstas como camino de solución de problemas matemáticos, reto metodológico que tiene el docente de programación de computadores para aproximar a sus alumnos a ese punto de convergencia que conforman las matemáticas y la programación y permitir que éstos lo conciban como fuente tanto de problemas computables como de soluciones implementables según Trejos, Significado y Competencias (2013) [3].

Uno de los objetivos de la enseñanza de la programación de computadores, dentro del contexto de un proceso de formación profesional para ingenieros de sistemas, radica en proveer de herramientas lógicas y tecnológicas a los estudiantes para que puedan encontrar, diseñar, implementar, monitorear, retroalimentar y optimizar soluciones que resuelvan problemas computables de manera que sea el mismo estudiante quien las diseñe, las conciba y las implemente [4]. De otra parte, se pretende con la enseñanza de la programación que los estudiantes

puedan capitalizar al máximo los recursos que provee la tecnología computacional para resolver problemas transversales a otras ciencias y otras áreas de conocimiento [5]. El reto que tiene el docente de programación de hoy radica en poder encontrar soluciones óptimas, fundamentadas e implementables tanto a problemas computables como buscar caminos para encontrar, por vía de la programación, aproximaciones para resolver problemas no computables. Puede advertirse que una de las competencias que el docente debe infundir en sus estudiantes es la capacidad para reconocer un problema computable y encontrar ágilmente por lo menos una solución posible, así como reconocer que un problema no es computable y, aun así, aproximarse a su posible solución por el camino de la programación de computadores [6].

El problema que abordar en este artículo consiste en plantear una metodología simple que se aproxime a la construcción de funciones simples y que se base en operaciones simples (la reduplicación de la palabra "simple" es intencional) para resolver problemas heredados de la matemática, pero con gran impacto en la computación aprovechando los recursos que provee el concepto de función y su relevancia dentro de la programación funcional. La temática puede destacarse como importante dado que en la medida en que el estudiante encuentre, apropie, asimile y aplique el concepto de función tanto en su formalización matemática como en su aplicación tecnológica desde la programación de computadores, podrá simplificar el camino para aproximarse y apropiar por lo menos tres paradigmas de programación: paradigma funcional, paradigma imperativo y paradigma orientado a objetos según Trejos Buriticá [7], y, de esta forma, podrá articularse con recursos que estos paradigmas proveen tal que la programación se convierta en una gran herramienta para su desempeño profesional como ingeniero de sistemas. Adicional a esto, el tema es relevante pues provee, una vez más, al estudiante del gran sustento teórico que simplifica la concepción, el diseño y la implementación de soluciones algorítmicas como es el concepto de función.

Se ha acudido a fuentes de carácter bibliográfico representadas en libros y artículos especializados que abordan temas similares tanto desde lo conceptual como desde lo investigativo y que tienen íntima relación con la búsqueda permanente de estrategias que posibiliten para el estudiante de ingeniería de sistemas la apropiación de conceptos que catapulten su lógica de solución de problemas computables desde una óptica tanto conceptual como práctica; conceptual en lo que compete a las matemáticas y práctica en lo que corresponde a la programación de computadores. De la misma manera se ha acudido a encontrar en las fuentes bibliográficas los fundamentos de algoritmia que subyacen a una solución implementada a partir de funciones, los conceptos que giran alrededor de las operaciones básicas y la génesis tanto de dichos conceptos como de las operaciones en mención y los elementos que fundamentan la programación funcional como espacio excelso de aplicación del concepto de función dentro de las posibilidades que brinda el lenguaje de programación DrRacket.

Lo innovador que se puede encontrar en este artículo gira alrededor de la búsqueda de soluciones simple a problemas simples que posiblemente ya está resuelto (la pala-

bra "simple" se ha utilizado de repetido de manera intencional) pero que encuentra en el concepto de función, a partir de su conceptualización e interpretación matemática de base, una forma innovadora de obtener resultados aprovechando las características modernas de la tecnología computacional y propiciando un espacio como ejercicio algorítmico para los estudiantes.

Los algoritmos presentados en este artículo, debido a que se fundamentan en el concepto de la recursividad y su respectiva implementación, están sujetos a las limitantes que provea la tecnología computacional puntual de la máquina en donde se ejecuten dado que la recursión, como insumo algorítmico, tiene un costo de máquina que debe tenerse en cuenta, aunque se sale de las fronteras de análisis y contenido del presente artículo. De la misma manera tienen su efecto en el planteamiento y ejecución de una metodología como la que se propone en este artículo, los antecedentes de programación que tengan los estudiantes, aunque no son determinantes para capitalizar su efectividad.

El objetivo de este artículo consiste plantear una posible solución metodológica a la búsqueda y resolución de enunciados con los cuales está familiarizado el estudiante, y su respectiva implementación en un lenguaje funcional (como en este caso DrScheme). Los enunciados puntuales a resolver y la metodología utilizada para proponer, evaluar, analizar, implementar y ejecutar la solución se presentan en el ítem Metodología de este artículo junto con un análisis breve de algunas opiniones de los estudiantes que dan fe del impacto de ésta sobre ellos. La metodología se utilizó en el desarrollo del curso Programación I de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira durante el I y II semestre de 2015 y el I semestre de 2016. El contenido de dicha asignatura corresponde al estudio, apropiación, aplicación, asimilación y retroalimentación del paradigma de programación funcional.

En términos de metodología de la investigación, la hipótesis se orienta hacia responder la pregunta ¿es posible plantear una propuesta algorítmica que permita construir funciones que faciliten la resolución de cálculos simples y que acudan tanto a operaciones básicas como a los recursos que la programación funcional provee?

La respuesta está contenida en el presente artículo y es un subproducto del Proyecto Código 6-16-13 "Desarrollo de un modelo metodológico para el aprendizaje de la programación en ingeniería de Sistemas basado en Aprendizaje Significativo, Aprendizaje por Descubrimiento y el Modelo 4Q de preferencias de pensamiento" aprobado por la Vicerrectoría de Investigaciones, Innovación y Extensión de la Universidad Tecnológica de Pereira.

2. Marco teórico

Cuando se habla del término "Algoritmo" se hace referencia a una secuencia específica de pasos que, bajo un enfoque determinado, permite lograr objetivos cuantificables, comparables y verificables en relación con un enunciado o una situación que exige una solución según Brassard & Bratley (2006) [8]. Un algoritmo es el paso previo a la construcción de un programa dado que el algoritmo es el conjunto de pasos que, conceptualmente, resuelven un problema mientras que un programa es un algoritmo escrito en los términos de un lenguaje de programa-

ción, es decir, es el mismo conjunto de pasos pero que, en vez de pasos, tiene instrucciones entendibles y ejecutables por el computador.

El algoritmo es una de las herencias que deja el pensamiento lógico matemático que, a lo largo de su historia, ha procurado establecer conjuntos lógicos de pasos que posibilitan la resolución de problemas y que son independientes del contexto asociado al problema lo cual permite que se pueda ver como una solución general y no puntual [5]. Puede ser, hasta cierto punto, un planteamiento conceptual que resuelve de manera general una situación problema, el programa (por su naturaleza propia asociada a las características de un lenguaje de programación) podría considerarse como la solución específica sin desconocer que, en sí mismo, también tiene elementos que generalizan el uso de los recursos de cara a la implementación de la solución.

Un paradigma de programación es un enfoque a partir del cual se puede visualizar el camino para resolver un problema, esto es, una forma de mirar el problema para así mismo encontrarle una determinada solución en consonancia con Trejos Buriticá (2013) [4]. Un paradigma de programación es la expresión matemática que posteriormente se cristaliza con los lenguajes de programación y que está asociada a ellos de alguna manera. El paradigma declarativo ha dado como uno de sus productos la programación funcional que en su más simple definición es un enfoque que posibilita la construcción de soluciones a problemas computables basándose en el aprovechamiento máximo del concepto de función, de sus características y de las posibilidades que éstas brindan para simplificar y resolver un problema según Van Roy (2008) [9].

En la programación funcional tres elementos son de gran importancia y constituyen su base lógica de trabajo: a) el concepto de función a partir del cual una solución se busca sobre la base de una disgregación del objetivo principal de forma que, resolviendo pequeños objetivos, en conjunto se pueda resolver el problema inicial; b) la recursividad definida, en lo conceptual, como la característica que tiene una función de poder llamarse a sí misma y, en lo tecnológico, como la posibilidad que brinda un lenguaje de programación de que así sea permitiendo que dichos llamados tengan un final finito y, a través de dicha recursión, se logren objetivos puntuales y c) la estrategia de desarrollo "divide and conquer" cuyo objetivo consiste en que el programador no resuelva un objetivo general sino que lo subdivida en pequeños objetivos y la solución al problema consista en la solución y conjunción de dichos pequeños objetivos de acuerdo con Van Roy (2008) [10].

Cuando se habla de operaciones simples en programación se habla de cálculos basados exclusivamente en sumas y restas que no involucran ninguna función de librería que implemente otro tipo de funciones. La suma se constituye en la operación más simple por excelencia y si se busca la etimología de la palabra "computador" se encuentra que proviene del griego "computare" que significa "sumar" es decir, un computador es, eminentemente, una máquina que suma según Crilly (2011) [11]. Gracias a los aportes que realizan los métodos numéricos, la resolución de muchos problemas (que en las matemáticas obedecen a mecanismos de aplicación de reglas algebraicas) en los computadores se reducen a un conjunto de sumas de elementos que, al final, obtienen el mismo resultado. Tal es el caso del cálculo de las funciones trigonométricas y

de otras funciones a partir de la aplicación de métodos como el de Newton Raphson acorde con Chapra & Canale (2010) [12].

En este sentido, este es un camino para conferirle significado y sentido al conocimiento, apropiación, asimilación, aplicación y retroalimentación de la programación de computadores y que permite el encuentro entre la teoría del aprendizaje significativo y la enseñanza de la programación. Como aprendizaje significativo se entiende la teoría que sostiene que cualquier aprendizaje es mucho más efectivo si tiene significado y sentido y que en cualquier proceso de aprendizaje lo más importante es lo que el estudiante ya sabe según Ausubel (1986) [13]. El aprendizaje significativo destaca el conocimiento previo y el nuevo conocimiento como los dos pilares para que el aprendizaje se cause, es decir, a partir de las relaciones que se puedan establecer entre ambos bien sean relaciones de complemento, de sustitución, de optimización o de actualización.

El conocimiento previo es el conjunto de saberes que le van quedando a un estudiante y que son producto de sus interacciones con los tres contextos en los cuales se mueve: el contexto del aula, el contexto escolar y el contexto extraescolar. El nuevo conocimiento se puede definir como ese saber que aún no le ha llegado al estudiante, al cual no ha podido acceder o simplemente que no ha podido estudiar e intentar asimilarlo y apropiarlo de acuerdo con Bruner (1963) [14]. La palabra "nuevo" en el concepto "nuevo conocimiento" no siempre implica lo propiamente nuevo acorde con el significado estricto de la palabra. "Nuevo" también significa aquello que se desconoce y a lo cual, por los medios convencionales disponibles, no se ha podido acceder. El tercer componente del aprendizaje significativo, como elemento propio de la teoría expuesta, es la actitud del estudiante y en esto tiene que ver mucho la labor del docente dado que se fundamenta en la motivación y la disposición que tenga el estudiante para aprender y en la capacidad que desarrolle el estudiante para establecer relaciones entre el conocimiento previo y el nuevo conocimiento.

En lo que se refiere al aprendizaje por descubrimiento se puede decir, basado en su teoría asociada, que cuando se descubre el proceso de aprendizaje en el cual tal acción esté inmersa se hace más efectivo y adquiere de inmediato significado para el aprendiz según Bruner (1969) [15]. El descubrimiento también va asociado con la posibilidad de explorar el contexto en el cual el conocimiento es útil o podría ser útil, experimentar con diferentes variantes, ensayar formas disímiles que apunten hacia lo mismo y equivocarse dentro del contexto de la búsqueda de solidificarlo. El descubrimiento es, pues, un recurso didáctico al cual el docente puede acudir dentro de un proceso de aprendizaje para potencializarlo y simplificar el camino de apropiación, asimilación, aplicación y retroalimentación de dicho conocimiento de acuerdo con Blanco Rivero & Silva Sanchez (2009) [16].

En este sentido debe advertirse que la enseñanza de la programación de computadores es un área más en la cual la aplicación de la teoría del aprendizaje significativo y la teoría del aprendizaje por descubrimiento y pueden ser soportes para que se diseñen actividades que propendan por una asimilación más simple con una componente de aplicación más directa teniendo en cuenta las facilidades modernas de acceder a un computador y hacer efectivo las diferentes

formas de utilización y aprovechamiento de dicha tecnología computacional en tratándose de resolución de problemas.

3. Metodología

La metodología didáctica utilizada para la resolución de problemas simples se ajusta a la forma estándar de solución de un problema computable. Esta metodología se ha socializado, compartido y aplicado con los estudiantes del curso Programación I de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira durante los semestres I y II del año 2015 y durante los semestres I y II del año 2016. En la Tabla 1 se presentan las fases que involucra esta metodología que ha sido, por definición, la metodología tradicional que se utiliza en el aula sin embargo en la práctica pocas veces se pone en escena de manera tan explícita y tan definida como en el desarrollo de la presente investigación.

Tabla 1. Metodología utilizada para resolver un problema computable.

Fase	*Nombre*	*Descripción*
1	Planteamiento del Problema	Se enuncia el problema que debe resolverse previendo que sea un problema computable y se define el objetivo a alcanzar
2	Formulación del Algoritmo	Se abre un espacio de discusión y análisis para que los estudiantes propongan posibles soluciones al problema planteado
3	Prueba de Escritorio	Se revisa "en frío" la propuesta algorítmica escogida de forma que se pueda verificar, en el papel (o en el tablero), que el algoritmo resuelve el problema
4	Revisión y ajuste del Algoritmo	Se hacen los ajustes pertinentes para que el algoritmo sea fácilmente implementable en un lenguaje de programación
5	Codificación	Se escoge un lenguaje de programación y se codifica el algoritmo convirtiéndose en programa. En este caso, por razones de contenido de la asignatura, se acudió al lenguaje de programación DrScheme
6	Digitación	Se transcribe el programa en un Ambiente Integrado de Desarrollo (Integrated Development Environment – IDE)
7	Compilación	Se le hace una revisión sintáctica al programa por parte del compilador y se realizan las correcciones pertinentes
8	Ejecución	Se ejecuta el programa para obtener los resultados a través del computador
9	Verificación de Resultados	Se verifica que dichos resultados satisfacen el objetivo inicial

Fuente: Elaboración propia.

En lo referente al plan general metodológico aplicado en el aula de clases, la Tabla 2 presenta las actividades que se adoptaron con miras a facilitar el proceso pedagógico.

Tabla 2. Plan general metodológico.

Ítem	*Nombre*	*Descripción*
1	Exposición de la Metodología	Se les realiza a los estudiantes una exposición detallada de los pasos que se van a desarrollar en cumplimiento de la aplicación de la metodología
2	Desarrollo de la Metodología	Se desarrolla la metodología contando con la opinión activa y participativa de los estudiantes en todo momento
3	Discusión sobre la Metodología	Una vez resuelto el problema desde su planteamiento hasta la verificación de los resultados, se abre un espacio de discusión para escuchar la opinión de los estudiantes

Ítem	*Nombre*	*Descripción*
4	Evaluación sobre la Metodología	Se realiza una evaluación escrita de tipo cuantitativo que permita verificar el nivel de apropiación del conocimiento adquirido
5	Retroalimentación de laMetodología	Se abre un espacio de opinión escrita que posibilite una verificación cualitativa del proceso y que retroalimente la metodología utilizada

Fuente: Elaboración propia.

En referencia con la metodología adoptada se formularon los siguientes enunciados:

a. Construir una función que permita calcular una multiplicación por el método de sumas sucesivas

b. Construir una función que permita calcular el cociente entero de una división por el método de restas sucesivas

c. Construir una función que permita obtener el residuo de una división apoyándose en otra función que calcule el cociente entero por el método de restas sucesivas

d. Construir una función que permita calcular el resultado de elevar una base a una potencia apoyándose en otra función que calcule una multiplicación por el método de sumas sucesivas

Para la construcción de estas funciones se ha acudido al concepto fundamental que inspira dichos operadores con el ánimo de lograr tres propósitos: a) buscar una solución original y propia por parte de los estudiantes, b) aprovechar la capacidad de proceso de la tecnología computacional moderna y c) buscar nuevas formas de solución a problemas que siendo muy familiares para los estudiantes pocas veces se les busca soluciones alternas.

La Tabla 3 presenta los algoritmos a los problemas planteados con su respectiva codificación en lenguaje DrScheme.

Tabla 3. Algoritmos a problemas planteados.

Enunciado	*Algoritmo*	*Codificación DrScheme*
Construir una función que permita calcular una multiplicación por el método de sumas sucesivas	Función multiplica Argumentos: Op1 → 1º operador Op2 → 2º operador Cont → contador Inicio Si Cont = Op2 Retorne Op1 Sino Calcule Op1 + Lo que retorne la Función multiplica Con argumentos Op1 Op2 Cont+1 Fin Si Fin Función	;; Función que realiza la multiplicación (define (multiplica op1 op2 cont) (if (= cont op2) op1 (+ op1 (multiplica op1 op2 (+ cont 1))))))
Construir una función que permita calcular el cociente entero de una división por el método de restas sucesivas	Función división Argumentos: Dividendo → 1º Op Divisor → 2º Op Inicio Si divisor > dividendo Retorne 0 Sino Sume 1 + Lo que retorne la Función division Con argumentos (dividendo – divisor) Divisor Fin Si Fin	;; Función que realiza la división recursiva (define (division dividendo divisor) (if (> divisor dividendo) 0 (+ 1 (division (- dividendo divisor) divisor))))

Enunciado	*Algoritmo*	*Codificación DrScheme*
Construir una función que permita obtener el residuo de una división apoyándose en otra función que calcule el cociente entero por el método de restas sucesivas	Función residuo Argumentos: Residuo → 1º operando Dividendo → 2º Operan. Divisor → 3º operando Inicio Reste Dividendo – El producto de Divisor Por Lo que retorne la Función division Con Argumentos Dividendo Divisor Fin Función	;; Función que calcula el residuo de una ;; división (define (residuo dividendo divisor) (- dividendo (* (division dividendo divisor) divisor)))
Construir una función que permita calcular el resultado de elevar una base a una potencia apoyándose en otra función que calcule una multiplicación por el método de sumas sucesivas	Funcion PotSuma Argumentos: B → Base E → Exponente Cont → Contador Aux → Auxiliar Inicio Si Cont = E Retorne (multiplica B Aux 1) Sino Llamar función PostSuma Con argumentos B E Cont+1 (multiplica B Aux 1) Fin Si Fin Función	;; Funcion que calcula la potencia basada en ;; sumas ;; b -> Base, e -> Exponente, ;; cont -> Contador de iteraciones ;; aux -> Argumento que simula la potenciación (define (potsuma b e cont aux) (if (= cont e) (multiplica b aux 1) (+ 0 (potsuma b e (+ cont 1) (multiplica b aux 1))))) ;; La función multiplica es la 1ª función de esta tabla

Fuente: Elaboración propia.

Debe anotarse que se presenta en la Tabla 3 (en su 3ª columna) el código de las diferentes funciones; sin embargo, este código debe complementarse con otras funciones que validan problemas como el signo o la relación de causalidad entre los operandos.

4. Resultados

La Tabla 4 presenta los resultados obtenidos con la ejecución de los programas.

Tabla 4. Resultados obtenidos.

Función	*Condiciones*	*Resultado*
Multiplicación basada en sumas sucesivas	Se requiere alimentar tres argumentos: *op1* y *op2* que corresponden a los multiplicandos y el argumento *cont* que corresponde al valor 1 para que en ese valor comiencen las sumas sucesivas. El problema del signo se resuelve con otra función. Los valores deben ser positivos y enteros	> (multiplica 5 4 1) 20 > (multiplica 30 25 1) 750 > (multiplica 15 13 1) 195
División basada en restas sucesivas	Se requiere alimentar dos argumentos: *dividendo* y *divisor*, cuyos nombres hablan por sí solos. Las consideraciones de signo y relaciones de causalidad entre los operandos se resuelven con otras funciones. Los valores deben ser positivos y enteros	> (división 20 5) 4 > (división 100 15) 6 > (división 200 18) 11

Función	*Condiciones*	*Resultado*
Potencia basada en multiplicaciones sucesivas (función del ítem 1)	En esta función se requieren cuatro argumentos para que su llamado sea efectivo: la base que se quiere elevar, la potencia a la cual se quiere elevar y dos valores 1 que corresponden a los argumentos que ayudan a que la potencia se calcule basado en sumas sucesivas, es decir, en la función *multiplica* del ítem 1 de esta tabla	> (potsuma 3 4 1 1) 81 > (potsuma 5 3 1 1) 125 > (potsuma 6 6 1 1) 46656
Residuo de una división	Para calcular el residuo de una división (módulo) sin acudir a las funciones de librería solo se necesita utilizar de manera eficiente la función división (del ítem 2) pues ésta calcula el cociente de la división, luego fácilmente podemos inferir la forma de obtener el residuo tal como se ve en los ejemplos	> (- 200 (* (división 200 18)18)) 2 > (- 10 (* (división 10 3) 3)) 1 > (- 20 (* (división 20 7) 7)) 6

Fuente: Elaboración propia.

A partir de estas soluciones, se realizó una prueba escrita tal que se pudiera tener una aproximación a los conceptos asimilados. La prueba versaba sobre dos problemas similares a los planteados. La Tabla 5 presenta un resumen de los resultados cuantitativos de la prueba escrita realizada a los estudiantes.

Tabla 5. Resultados cuantitativos.

Año	*Sem*	*Cant Estuds*	*Prom*	*Nota Mas Baja*	*Nota Mas Alta*
2015	I	22	3.8	3.6	4.3
	II	25	4.1	3.6	4.4
2016	I	19	4.4	4.0	4.6
	II	20	4.6	4.4	4.7

Fuente: Elaboración propia.

Finalizado todo el proceso metodológico, se abrió un espacio para que los estudiantes opinaran acerca de lo que se había realizado. Un resumen de las opiniones se presenta en la Tabla 6 teniendo en cuenta que se considera una *Opinión Favorable* (OpFav) aquella que elogia o exalta la metodología y todo el proceso y una *Opinión Desfavorable* (OpDesf) aquella que no lo hace, que es altamente discreta en su texto o aquella que la descalifica.

Tabla 6. Resultados cualitativos.

Año	*Sem*	*Cant Estuds*	*OpFav*	*OpDesf*
2015	I	22	19	3
	II	25	23	2
2016	I	19	18	1
	II	20	20	0

Fuente: Elaboración propia.

Debe anotarse que, por razones prácticas, se han expuestos los resultados cuantitativos de las evaluaciones como pauta de análisis de los datos recogidos. Por otra parte, en estos procesos siempre se espera que las opiniones favorables sean mayoritarias, aunque no siempre se cumple pues procesos que no satisfagan al estudiante se ven reflejadas en las opiniones más aun cuando éstas pueden ser anónimas.

5. Discusión

Se ha acudido a una metodología, como lo muestra la Tabla 1, que posibilita la construcción sistemática, organizada y secuencia de una solución implementable a un problema computable. Esta metodología permite que se pueda formular, en tres partes, el algoritmo que posteriormente se ha de convertir en un programa, aunque se admite que otras metodologías podrían obtener resultados similares, pero habría que realizar una investigación que lo compruebe.

En la primera parte se plantea la necesidad de realizar una lluvia de ideas de forma que se pueda llegar a una propuesta lógica que resuelva el problema planteado. En esta parte se realizan pruebas en frío, es decir, en el escritorio de manera que se pueda llegar a una aproximación de la forma como funcionaría el algoritmo en caso de que se implementara en un sistema computacional. Esta parte pretende involucrar plenamente al estudiante y conferirle la posibilidad de opinar, de escuchar y de ser escuchado, puertas que no siempre la exposición magistral posibilita.

En la segunda parte, la metodología acude a la necesidad de utilizar la tecnología computacional para reconstruir la solución planteada desde la algoritmia; pero esta vez, a partir de los recursos, posibilidades y limitantes que ofrece un lenguaje de programación. Para el caso de este artículo se ha utilizado un lenguaje heredado del paradigma declarativo cuyo núcleo principal se basa en la programación funcional. La implementación de estos mismos algoritmos en lenguajes fundamentados en otros paradigmas hace más interesante la comparación entre unos y otros.

Se parte de la solución en lo algorítmico, pero se plantea en términos puramente tecnológicos dado que lo que el computador ejecuta es un programa y por esa razón debe transformarse el algoritmo (que es una solución lógica escrita en términos humanos) en el código correspondiente (que es la misma solución escrita en términos tecnológicos). En este proceso es posible que se incorporen cambios en la lógica de la solución debido a las características y recursos que provee el lenguaje de programación sin que se afecte la lógica general de solución del problema lo cual deja de presenta la discusión acerca de la necesidad del algoritmo frente a la implementación a nivel de código puesto que, es posible, que uno de los dos pueda sustituir al otro.

En términos formales, el algoritmo posibilita que una solución pueda ser implementada en cualquier lenguaje de programación sin embargo la lógica que subyace a cualquier codificación invita a plantear la idea de que el mismo programa podría ser su algoritmo al tiempo.

El plan general metodológico que se expone en la Tabla 2 es la propuesta que se le presenta a los estudiantes de manera que ellos puedan involucrarse y hacerse partícipes de todo el proceso para que no solo puedan mirarlo con ojos críticos sino también que puedan capitalizar cada avance que se tengan en él y que ellos tengan a su respectivo ritmo. Presentarles a los estudiantes el plan metodológico abre las posibilidades de que ellos opinen y expongan criterios que pueden incorporarse para refinar dicha metodología.

Cabe anotar que las tres últimas fases del plan general metodológico corresponden a los espacios desde donde el estudiante puede tomar mayor actividad en su participación. La discusión les abre la posibilidad de que ellos opinen sobre la solución que se ha planteado tanto desde lo puramente algorítmico como desde lo tecnológico, es

decir, a nivel de su implementación en un lenguaje de programación. La evaluación, desde esta óptica, se constituye en un factor motivacional y en un instrumento de aprendizaje. Como factor motivacional los estudiantes, según sus propias opiniones, sienten por primera vez el gusto de sentirse evaluados dado que ellos mismos pueden verificar el avance en su proceso de aprendizaje a partir de las fases en las cuales está inmerso y de la metodología para resolver un problema computacional.

Como herramienta de aprendizaje, el profesor puede contar con una evaluación objetiva, heredada de la práctica que emerge a partir de la apropiación y asimilación de lo conceptual, lo cual es altamente necesario en procesos de formación tecnológica. La última fase del plan general metodológico implica un tiempo dedicado a la retroalimentación en el cual los estudiantes pueden manifestar sus coincidencias y divergencias con todo el proceso realizado, con cada una de las fases del plan general y con cada paso de la metodología de manera que, a partir de sus propias opiniones, se puedan reconocer elementos que permitan que en una siguiente oportunidad se refine la forma como se intenta llegar al conocimiento de los alumnos por este camino.

El código de las funciones planteadas tiene tres características que los hace innovadores y altamente creativos y que vale la pena destacar: a) son códigos muy simples de entender, b) son códigos muy fáciles de implementar y c) son códigos muy fáciles de probar. Estos tres factores permiten que se puedan encontrar aplicaciones en donde estas soluciones tan simples puedan ser de fácil aplicación y de fácil extrapolación. Se ha procurado, a pesar del costo computacional que implica el concepto de recursión, plantear soluciones lo más simples posibles de forma que puedan ser muy entendibles por los estudiantes y de tal manera que ellos puedan concebir que, una solución algorítmica, no requiere tener un nivel de complejidad en su concepción así tenga determinado nivel de complejidad (computacional) en su implementación.

Cabe anotar que se ha capitalizado el hecho de que la multiplicación es un conjunto de sumas sucesivas, la división es un conjunto de restas sucesivas y la potenciación es un conjunto de multiplicaciones sucesivas que a su vez son sumas sucesivas, razón por la cual puede concebirse la potenciación también como un conjunto definido y estructurado de sumas sucesivas. Para el caso del cálculo del residuo de una división (valor que siempre es entero) se ha implementado basado en el cálculo del cociente pues si éste se obtiene a partir de restas sucesivas, el módulo o residuo es igual al valor que queda restando luego que se han realizado todas las restas posibles. Por esta razón, el módulo o residuo se obtiene basado en la función que calcula el resultado del cociente en una división entera. Pudo haberse implementado una función independiente que calculara tanto el residuo como la potenciación, pero por su naturaleza se consideró, a juicio del autor de este artículo, que bien podía implementarse a partir de las funciones de multiplicación y división ya construidas.

En relación con los resultados obtenidos con los programas, que se muestran en la Tabla 4, los resultados son los esperados, es decir, los que también se obtendrían si se utilizaran los operadores de multiplicación (*) y de división (/) que para tal fin tiene previsto cualquier lenguaje de programación independiente de la notación que utilice. En la tabla se especifican los argumentos que se requieren en lo que respecta a su significado o sentido especialmente aquellos argumentos que necesitan algún tipo de valor de inicio para que el resultado sea

el esperado. Esto se supera porque para llamar las funciones que se presentan en este artículo se han diseñado funciones de interfaz que facilitan la tarea al usuario o sea que le solicitan solamente los valores naturales (u obvios). Esto implica que, por ejemplo, para la multiplicación el usuario solo tiene que digitar los dos valores que tiene que multiplicar; cualquier otro valor de apoyo que se requiera en la ejecución de los programas se resuelve internamente de manera que para el usuario se constituye en procesos transparentes y, finalmente, es el mismo usuario el que recibe el resultado que esperaba.

Esto anterior implica que, contando con la codificación completa del programa, además de lo explicado para la multiplicación, para la división se necesitan solamente los valores del dividendo y el divisor, para la potenciación solo se requiere los valores de la base y del exponente y para el cálculo del módulo (o residuo) solamente se requiere los mismos valores que para la división. En este artículo, por razones prácticas, se han omitido las demás funciones de apoyo y solo se han presentado las que resuelven puntualmente el problema.

En referencia con los resultados cuantitativos obtenidos en la evaluación que se le realizó a los estudiantes que se basaba en el contenido de este artículo, como objeto de aprendizaje, y que buscaba tener una aproximación numérica que cuantificara lo aprendido, se nota que los promedios, sobre un valor mínimo de 1,0 y un valor máximo de 5,0, es notoriamente favorable sin desconocer que en cada semestre, a partir de la retroalimentación realizada por los mismos estudiantes, el valor promedio va aumentando. La relación entre las notas más bajas y las notas más altas y su análisis con el valor promedio en cada semestre (dentro de este proceso específico) demuestra que cada vez son más los estudiantes que obtienen notas altas y menos los que obtienen notas bajas pues cada vez el promedio aumenta y se aproxima a los valores altos. Todo esto partiendo de que la evaluación escrita ha cumplido con un principio de objetividad que permite que estos resultados tengan la confiabilidad que la investigación requiere.

Ahora bien, en referencia con los resultados cualitativos las opiniones de los estudiantes han permitido retroalimentar todo el proceso al punto que se puede considerar ese como un factor altamente incidente en el hecho de que cada vez el promedio del salón es más alto. De la misma manera la Tabla 6 da fe de que cada semestre las opiniones favorables aumentan mientras que, en su proporción, las opiniones desfavorables disminuyen. En este caso se ha presentado algo que bien podría ser una coincidencia, aunque no necesariamente sea la única razón para que esto suceda y consiste en el hecho de que cada vez los estudiantes con opiniones desfavorables son menos disminuyendo de manera lineal de 1 en 1 llegando en el II semestre de 2016 a 0 estudiantes con opiniones desfavorables lo cual quiere decir que todos los estudiantes se articularon con el proceso metodológico y que encontraron en él la respuesta a sus expectativas.

6. Conclusiones

El desarrollo e implementación de un programa desde una perspectiva funcional implica la adopción de una metodología que no solo posibilite el seguimiento por parte de los estudiantes, sino que, al mismo tiempo, los haga partícipes activos de los procesos de aprendizaje. La presentación de la metodología que se adopte hace que el mismo estudiante tenga una visión crítica y proactiva que permita retroalimentar dicha metodología y que le posibilite ca-

pitalizar cada paso y cada avance que se dé en ella. Para el docente, tener un plan general metodológico le permite visualizar sus avances y las posibles fallas que, oportunamente, se puedan corregir de cara a buscar, cada vez, nuevos y mejores caminos para la apropiación, asimilación y aplicación de nuevos conocimientos.

El análisis estadístico tanto en lo cuantitativo como en lo cualitativo es un camino muy eficiente que permite retroalimentar este tipo de procesos y que posibilita un camino para mejorar, semestre a semestre, la aplicación de metodologías y la búsqueda de solución de problemas como los que se plantean en este artículo. Lo cuantitativo permite aproximarse, bajo la lupa de evaluaciones objetivas, a lo aprendido por el estudiante y a la apropiación del nuevo conocimiento. En cuanto a lo cualitativo, la opinión de los estudiantes, su análisis y su uso como mecanismo de retroalimentación permite que se refinen las estrategias y actividades asociadas a las fases metodológicas y al proceso de investigación en sí. Apoyarse en instrumentos y herramientas que permitan un buen análisis estadístico permite que se tenga una perspectiva más clara de lo que dicen los datos al margen de las expectativas que, al respecto, tenga el investigador. A los datos se les debe dar la oportunidad de que hablen por sí solos.

7. Referencias

[1]Trejos Buriticá, O. (2000). La Esencia de la Lógica de Programación. Pereira: Papiro.

[2] Deitel & Deitel. (2013). C++ Programming. New York: Prentice Hall.

[3] Trejos Buriticá, O. (2013). Significado y Competencias. Pereira: Papiro.

[4] Kaasboll, J. (1999). Exploring didacti models for programming. Oslo: Universidad de Oslo.

[5] Shoup, V. (2008). A computational introduction to number theory and algebra. Cambridge: Cambridge University Press.

[6] Polya, G. (1989). Cómo plantear y resolver problema. México D. F. : Editorial Trillas.

[7] Trejos Buriticá, O. (2013). Estrategia metodológica para aproximar los paradigmas funcional, imperativo y orientado a objetos en ingeniería de sistemas a partir de aprendizaje significativo. (U. L. Colombia, Ed.) Avances en Ingeniería, 49-63.

[8] Brassard, G., & Bratley, P. (2006). Fundamentos de Algoritmia. Madrid: Prentice Hall.

[9] Van Roy, P. (2008). Concepts, Techniques and Models of Computer Programming. Estocolmo: Université catholique de Louvain.

[10] Van Roy, P. (2008). Techniques and methods in programming computer. Louvaine: University Press.

[11] Crilly, T. (2011). Grandes cuestiones matemáticas. Barcelona: Ariel Editorial.

[12] Chapra, S., & Canale, R. (2010). Métodos numéricos para ingenieros. México: McGrawHill Educación.

[13] Ausubel, D. (1986). Sicología Educativa: Un punto de vista cognoscitivo. Ciudad de México: Trillas.

[14] Bruner, J. S. (1963). El proceso de la Educación. Ciudad de México: Editorial Hispanoamericana.

[15] Bruner, J. S. (1969). Hacia un teoría de la instrucción. Ciudad de México: Hispanoamericana.

[16] Blanco Rivero, L., & Silva Sanchez, E. (2009). Herramientas pedagógicas para el profesor de Ingeniería. Bogotá: Lemoine Editores.