Algoritmo para calcular un cociente real con control de cantidad de decimales utilizando dos paradigmas de programación

An algorithm for computing a real quotient with control of decimal part using two paradigms

Omar Iván Trejos Buriticá[1]

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

Fecha de recepción: 08/09/2016 Fecha de aceptación del artículo: 22/12/2016

Resumen

El presente artículo presenta una propuesta algorítmica para calcular un cociente real controlando la cantidad de decimales que se quieran obtener en el resultado a partir de la utilización de procesos cíclicos y recursivos que la programación imperativa y funcional provee respectivamente. El propósito de esta investigación es demostrar qué tan fácil es que los mismos estudiantes logren por sus propios medios resolver problemas que son conocidos, todo en pos de un proceso de aprendizaje efectivo, con significado y sentido. Metodológicamente se les explicó a los estudiantes los pasos que se iban a realizar y se desarrolló, de manera comparativa, una solución algorítmica en dos paradigmas de programación. Se encontró que para los alumnos es de gran importancia encontrar relación directa entre los conocimientos previos y los nuevos conocimientos de forma que unos se vean reflejados en los otros y que pudieran evidenciar que, a la programación de computadores, como expresión tecnológica, subyacen modelos comunes independientes de la tecnología que se involucre. Se concluye que en la medida en que los estudiantes de programación conozcan la metodología que se va a utilizar para resolver un problema, sigan paso a paso dicha metodología, la implementen y puedan comprobar que los resultados satisfacen los requerimientos dentro del marco de unos enunciados que les sean cercanos, el aprendizaje de la programación de computadores será muy simple y efectivo.

Palabras clave

Algoritmo, Números reales, Paradigma de programación, Programación funcional, Programación imperativa

Abstract

This article presents an algorithmic approach for calculating an actual ratio by controlling the number of decimal places using cyclic and recursive processes with imperative and functional programming respectively. The purpose of this research is to demonstrate how easy is that these students achieve on their own solve problems that are known, all in pursuit of an effective learning process, with meaning and sense. Methodologically the students knew the steps that were to be performed and developed, comparatively, an algorithmic solution into two programming paradigms. We found that for students is very important to find a direct relationship between previous knowledge and new knowledge so that some are reflected in each other and that could demonstrate that computer programming, as technological expression underlie independent common models of the technology involved. We conclude that when programming students know the methodology to be used to solve a problem, follow step to that step methodology, implement and verify that the results with the requirements within the framework of some problems that are closed, learning computer programming will be very simple and effective.

Keywords

Algorithm, Functional paradigm, Imperative paradigm, Programming paradigm, Real numbers.

1. Introducción

Uno de los propósitos del Grupo de Investigación en Informática del programa Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira es propender por la búsqueda de soluciones de aquellos problemas que son conocidos y que, teniendo cercana relación con la vida académica del estudiante y con sus conocimientos previos, pocas veces se resuelven de la mano del docente en un formato didáctico que posibilite un acompañamiento efectivo y, además, un aprendizaje con significado y sentido que alcance los objetivos propuestos. Desde lo académico el artículo se justifica en virtud de que son muchos los enunciados que el estudiante puede resolver y que pueden ser insumos para fortalecer y transversalizar el conocimiento desde el área de la programación hacia otras áreas académicas.

Dentro de los cursos de programación, la búsqueda de enunciados y situaciones problema para ser resueltos con los insumos que la programación provee es una de las labores que los docentes de esta área deben hacer permanentemente [4]. Para ello deben recurrir a la inmersión bibliográfica o a la construcción original, ambas producto de sus conocimientos previos con el ánimo de que el estudiante tenga una serie de posibilidades de poner en práctica todo el espectro metodológico que implica programar. Las matemáticas son una fuente casi infinita de enunciados problema muchos de los cuales pueden ser resueltos a través de la programación de computadores [3], cuanto éstos son del tipo de problemas computables, es decir, cuando se pueden resolver a través de la tecnología computacional.

Esto devela una gran cercanía productiva y muy conveniente entre las matemáticas y la programación de computadores sin desconocer que ésta es la expresión tecnológica de aquella en esas situaciones y enunciados en donde puede actuar. Esto implica que todavía existen muchos problemas matemáticos en donde la programación aún no es solución bien por razones del modelo computacional o bien por razones inspiradas en la misma naturaleza del problema [21].

Es de anotar que, precisamente en tiempos modernos, las teorías de programación y los modelos que subyacen a la matemática trabajan para que cada vez sea la programación un camino para implementar soluciones computables y de esa forma ampliar el espectro de aplicación sobre el amplio panorama problémico que provee las matemáticas [5].

El problema a abordar en este artículo acude a la necesidad de encontrar enunciados, heredados de la matemática, que siendo muy familiares para los estudiantes aún no hayan sido resueltos aplicando una metodología específica de solución de problemas computables, acudiendo a un paradigma de programación definido e implementando, a través de las posibilidades que brinda un lenguaje de programación, una solución que sea altamente satisfactoria. La importancia del tema radica en que si bien la programación es un área importante en la formación de ingenieros de sistemas, especialmente en aquellos en los cuales dicha programación es uno de sus perfiles fuertes de formación, no se puede negar la gran importancia que tienen las matemáticas no solo como base para observar, modelar, diseñar, plantear, intervenir, mejorar, optimizar o retroalimentar las realidades que nos rodean sino también para proveer enunciados que puedan ser resueltos óptimamente a través de los insumos que dicha programación provee.

Es allí en donde el docente debe afinar sus estrategias didácticas para que el estudiante encuentre esa relación tan íntima existente entre las matemáticas y la programación desde los enunciados más simples hasta los más complejos y así mismo se lo transmita a sus alumnos de manera que puedan recurrir a ambas áreas, una para que provea problemas, modelos y estructuras conceptuales y la otra para que provea recursos, insumos y posibilidades de solución por el camino de la tecnología computacional [7]. En diferentes revistas especializadas, y cada vez con más ahínco, se ha evidenciado la búsqueda de caminos que, a partir de la investigación, fortalezca la enseñanza y el aprendizaje de la programación de computadores como solución de problemas que se puedan heredar de otras áreas.

Si se quiere acotar lo innovador en este artículo se podrían enunciar tres elementos que se destacan: a) la búsqueda metodológica de una solución a un problema muy cercano a los estudiantes; b) el diseño de una solución original sencilla de entender y sencilla

de implementar y c) el buen aprovechamiento de los recursos cíclicos y recursivos que la programación imperativa y funcional proveen, respectivamente. Si bien no son estos conceptos nuevos, debe anotarse que lo innovador no necesariamente es lo nuevo per se sino aquello en donde lo existente se aplica de forma práctica, simple y, sobre todo, muy útil a partir de encontrar solución a un problema que, siendo tan familiar, no se halla escrito formalmente.

Esta investigación se circunscribe dentro de los límites que la tecnología computacional moderna implica, esto es, en lo que se refiere a la implementación funcional dependerá de las posibilidades de ejecución que la máquina le conceda a la recursividad teniendo en cuenta el costo operativo que ésta implica y, en lo que se refiere a la implementación imperativa, va a depender de los límites establecidos como capacidad máxima binaria de almacenamiento de las variables que constituyen la esencia de este paradigma. No se descarta que la solución, siendo tan simple y sencilla, pueda ser optimizable y mejorable tanto en rendimiento como en consumo de máquina.

Para la implementación de este artículo, se ha requerido acudir a literatura relevante y muy actualizada representada en libros y artículos publicados. Ambas fuentes destacan la importancia de las matemáticas y los modelos y enunciados que proveen, la relevancia de la programación como camino de cristalización de las posibles soluciones a los problemas que plantean las matemáticas cuando éstos son computables, la relación entre matemáticas y programación y la manera efectiva como estos conocimientos pueden servirse a los estudiantes dentro del contexto de un proceso de formación profesional como ingenieros de sistemas.

El objetivo general de esta investigación acude a la solución del problema planteado en unos términos que sean entendibles, comunicables e implementables tal como se demuestra en el contenido del presente artículo. Para la construcción de este artículo se ha requerido acudir a las bases matemáticas del concepto de números reales, sus características y sus implicaciones, la forma como interactúan en procesos iterativos y la idea del infinito como referente a procesos de estas características. De la misma manera se ha tenido que acudir a los fundamentos de la programación funcional y la programación imperativa, dos enfoques de programación que priorizan cada uno aristas diferentes al momento de implementar soluciones y, de la misma forma, se ha debido refinar el concepto de algoritmo y algoritmia, el primero como cristalización de la solución y el segundo como camino conceptual para encontrarla.

La metodología utilizada se explica en detalle en el ítem correspondiente de este artículo, así como el desarrollo paso a paso para el hallazgo de las soluciones. La investigación se realizó durante el II semestre de 2015 y I semestre de 2016 en la asignatura Programación II de la Universidad Tecnológica de Pereira, con el ánimo no sólo de que los estudiantes pudieran conocer un modelo de solución al problema planteado, sino que también pudieran hacer un comparativo entre los dos paradigmas de programación a partir de sus respectivas implementaciones.

Subyace al problema planteado una hipótesis que podría enunciarse en los siguientes términos: ¿es posible capitalizar los modelos conceptuales que brindan las matemáticas, articularlos con los recursos que proveen los paradigmas de programación a través de lenguajes de programación y construir soluciones que resuelvan problemas que, siendo familiares a los estudiantes, son vigentes, necesarios y transversales? La respuesta a esta hipótesis se refleja en el contenido de este artículo el cual se ajusta a los estándares de forma y fondo convencional de artículos de investigación científica y tecnológica.

El presente artículo es un subproducto del Proyecto Código 6-16-13 "Desarrollo de un modelo metodológico para el aprendizaje de la programación imperativa 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.

1. Marco Teórico

Un algoritmo se define como un conjunto de pasos ordenados y secuenciales que permiten lograr un objetivo por un camino apropiadamente sistematizado [7]. Son pasos ordenados debido a que tienen un orden específico, es decir, solo en cierto orden es

como permiten que se alcance el objetivo y son secuenciales porque cada paso debe ejecutarse o realizarse de manera que siempre se realice solo uno, que exista un 1er paso y que exista un último paso, es decir, que se realicen un paso (y solo uno), después de otro [20]. El algoritmo es la forma de simplificar los caminos racionales y deliberativos que el cerebro busca para resolver problemas sin embargo cuando se trata de implementar dichas soluciones, siendo tan amplias en su concepción para el ser humano, es necesario establecer unos elementos paramétricos que permitan definir fronteras lógicas y de ejecución que hagan que la solución no solo sea más concreta sino que, además, sea implementable a través de una máquina que, en tiempos modernos, es el computador [18].

La palabra "algoritmo" es la latinización del nombre del gran difusor del álgebra Al Juarizmi quien a partir de las diferentes interacciones y relaciones que encontró entre los números vistos desde una óptica general comenzó a plantear algunos mecanismos y secuencias de pasos que resolvían problemas, también en términos generales, y que eran comunes a muchas situaciones independientes del contexto y de los valores que intervinieran [15].

Un cociente es el resultado de dividir un número entre otro, es decir, el resultado de repartir una cantidad (dividendo), entre un conjunto de elementos (divisor). Un cociente real es el resultado de dividir un dividendo que no es múltiplo de un divisor razón por la cual el resultado final es un número que tiene una parte entera y una parte decimal [6]. La parte decimal de la parte entera se separa a través de una coma. Un cociente entero equivale a la parte entera de una división, cualquiera que ésta sea, es decir, cuando el dividendo es múltiplo del divisor y cuando no.

El cociente entero también se puede definir como la cantidad de veces completas que un divisor puede ser restado del dividendo. El residuo o módulo es el valor que queda cuando se ha obtenido un cociente entero y el dividendo no es múltiplo del divisor. El residuo o módulo siempre ha de ser menor que el divisor. En las divisiones en las cuales el cociente es exacto [11], el residuo es igual a cero (0). Se conoce como parte decimal (o sencillamente "decimales"), a aquel conjunto de dígitos que, siendo producto de una división no exacta, se encuentran escritos después de la coma que los separa de la parte entera. Cuando se encuentra que un conjunto de dígitos en la parte decimal que se repite se dice que es un número real periódico, por ejemplo: 34,678678678… en donde el periodo de la parte decimal es 678. Debe anotarse que las divisiones cuyos cocientes tienen periodos normalmente son números inconmensurables, es decir, la división nunca finaliza. La cantidad de dígitos de la parte decimal que resulta de una división no exacta determina la precisión del número, es decir, el número 5,75634 es mucho más preciso que 5,75 debido a que aquel tiene más dígitos en la parte decimal que éste.

Un lenguaje de programación es un conjunto de instrucciones que son reconocidas y pueden ser ejecutadas por un computador y que normalmente tienen asociado un ambiente de desarrollo que permite escribir, revisar, compilar, ejecutar y obtener resultados [17]. Normalmente un lenguaje de programación está asociado a un paradigma de programación, es decir, a un enfoque específico para resolver problemas en los cuales la solución pueda ser implementable en un computador. Entre estos paradigmas se encuentra la programación funcional que, como su nombre lo indica, se basa en el aprovechamiento de las características de las funciones para concebir, diseñar, construir y codificar programas de computador [21].

El concepto de función, por lo dicho, es el eje central de la programación funcional y a partir de él se desarrollan tres conceptos que hacen efectivo este enfoque de solución de problemas: a) la estrategia "divide and conquer" que permite atomizar las soluciones a partir de una concepción más simple cada vez atendiendo pequeños objetivos que, en conjunto, solucionan objetivos más complejos [19]; b) el concepto de recursividad que si bien implica un costo computacional alto –que en tiempos modernos se ha ido superando gracias al avance del hardware- también es cierto que se constituye en un excelente recurso para diseñar soluciones simples a problemas de cierta complejidad tanto conceptual como computacional [13], y c) el diseño modular que permite que partes específicas de un programa se codifiquen por aparte y, de esa forma, se haga muy sencillo encontrar los errores lógicos (que no los sintácticos) en el momento en que éstos se presente [14].

Por su parte, otro enfoque de programación lo brinda el paradigma imperativo que basa su perspectiva tecnológica en tres conceptos muy simples, digeribles y manejables: a) la utilidad de tres estructuras básicas que posibilitan, teóricamente, la construcción de cualquier solución que sea computable. Estas tres estructuras corresponden a la secuencia de instrucciones, los condicionales y los procesos cíclicos o iterativos [16]; b) el concepto de estado, que refleja las condiciones iniciales, intermedias y finales de la memoria del computador y que cristaliza la utilización de variables, sus condiciones y sus características como eje central para el desarrollo de un programa concibiéndolo como un conjunto de instrucciones que alteran los contenidos de unas variables que, al final, satisfacen un determinado objetivo y c) la implementación de ciclos de bajo costo computacional como corresponde a los ciclos conceptuales *mientras-que*, *para* y *haga-hasta* y que tienen un equivalente en el lenguaje C++ arista imperativa en los ciclos *while*, *for* y *do-while* [17].

El aprendizaje significativo es una teoría de aprendizaje enunciada por el Dr. David Paul Ausubel que prioriza el significado del conocimiento como base para que éste se haga efectivo en el cerebro del estudiante, es decir, si el conocimiento tiene significado y sentido, el estudiante podrá aprender realmente, siempre y cuando le sean favorables las condiciones que para tal fin se hayan establecido [2]. Lo más importante en un proceso de aprendizaje es lo que el estudiante ya sabe (conocido como conocimiento previo) que, junto con el nuevo conocimiento y la actitud del estudiante, son los tres pilares que se requiere concebir y conceptualizar para que un proceso de aprendizaje sea exitoso. La actitud del estudiante incluye la motivación por aprender y la capacidad de poder establecer interacciones entre el conocimiento previo y el nuevo conocimiento [9].

Por su parte el aprendizaje por descubrimiento es una teoría enunciada por el Dr. Jerome Seymour Bruner quien plantea que todo lo que el ser humano descubre, sea por innovación o por fascinación, queda ubicado en la memoria a largo plazo y le posibilita desarrollar procesos de aprendizaje exitosos [8]. En este sentido se destacan las posibilidades de que el estudiante, en tiempos modernos, pueda explorar, experimentar, ensayar y equivocarse como elementos que allanan el camino hacia el aprendizaje efectivo y que permiten que el descubrimiento se convierta en una forma de acceder al conocimiento con unas bases definidas, pero a partir de sus propios esfuerzos y logros involucrándose en su propio proceso de aprendizaje y estableciendo sus propias metas [1].

2. Metodología

La metodología utilizada en el desarrollo de esta investigación se presenta en la Tabla 1 y está dividida en fases de forma que el estudiante pueda conceptualizar el camino lógico algorítmico para encontrar su solución. Este ha sido planteado, refinado e implementado como parte del contenido de la asignatura Programación II que corresponde al paradigma de programación imperativa y para el cual se ha acudido tanto a su implementación en lenguaje C++ arista imperativa como al lenguaje Scheme (programación funcional) que corresponde al contenido de la asignatura Programación II, requisito para cursar la asignatura Programación I. Es de anotar que la metodología utilizada corresponde a la metodología estándar para la construcción de un algoritmo computable. La metodología se ha utilizado durante los semestres I y II del año 2015 y durante los semestres I y II del año 2016. Por razones institucionales de forma, en este artículo solo se reportan los resultados del año 2015.

Tabla 1. Fases de la Metodología
Fuente: Elaboración propia

Fase	Descripción	Objetivo
1	Planteamiento del Problema	En esta fase se presenta el problema de una manera informal de forma que sea verificable la comprensión del problema por parte de los estudiantes
2	Análisis de la Solución	Se abre un tiempo apropiado y prudente para que los estudiantes planteen diferentes posibles soluciones a manera de *brainstorm**
3	Planteamiento del Algoritmo	Se escoge una de las posibles soluciones atendiendo las facilidades que ésta contenga para ser implementada en un lenguaje de programación
4	Desarrollo de la Prueba de escritorio	Se le hace una prueba de escritorio para verificar su validez y el logro, en el papel, del objetivo propuesto
5	Refinamiento del Algoritmo	Se destina un tiempo para analizar las características del algoritmo y optimizar aquellas partes en donde se puede

6	Implementación	Se codifica el algoritmo sobre la base del concepto de función tanto para la aplicación del paradigma funcional como para el paradigma imperativo. Se acude a la documentación como forma de almacenar la lógica de solución del programa
7	Digitación	Se transcribe en el computador tanto el código en lenguaje funcional DrScheme como en lenguaje imperativo Lenguaje C++ arista imperativa
8	Ejecución	Se ejecuta utilizando el IDE de DrScheme llamado DrRacket y el IDE de Lenguaje C++ llamado DevC++
9	Verificación de Resultados	Se verifican los resultados obtenidos y se comprueba que satisface el objetivo planteado

*Lluvia de ideas

El enunciado problema a resolver es el siguiente: construir un programa que permita calcular el cociente de una división con decimales en el cual se pueda controlar la cantidad de decimales que tenga el resultado. Para la aplicación detallada de esta metodología, tal como se presenta en la Tabla 1, se destinó una semana completa (tres sesiones cada una de dos horas) de manera que se pudieran desarrollar cada una de las fases, especialmente las que incluían participación de los estudiantes, opiniones, conceptos y observaciones, sin ningún afán y siempre buscando que los estudiantes apropiaran, asimilaran y aprendieran la metodología utilizada. La Tabla 2 presenta el algoritmo conceptual seleccionado como el óptimo y la respectiva implementación en lenguaje DrScheme (paradigma funcional) y Lenguaje C++ arista imperativa (paradigma imperativo).

Tabla 2. Algoritmo e Implementación

Descripción	Implementación
Algoritmo	Función DECIMALES Parámetros Entero: Residuo, Divisor, CantDec Variables Locales Real: Resultado=0.0 Entero: Contador=0 Inicio Mientras cantdec > 0 Residuo = (Residuo * 10) módulo divisor CantDec = CantDec - 1 Resultado = Resultado + (Residuo / $10^{Contador}$) Contador = Contador + 1 Fin Mientras Retorne (Resultado) Fin Función DECIMALES

Paradigma Imperativo
 	
Lenguaje C++ (Arista Imperativa)
	float decimales(int residuo, int divisor, int cantdec)
	{ float resultado=0.0;
	int contador=0;
	while(cantdec>0)
	{
	residuo=(residuo*10)%divisor;
	cantdec--;
	resultado=resultado
	+(residuo/pow(10, contador));
	contador++; }
	return(resultado); }

Paradigma Funcional **Lenguaje DrScheme**	(define (decimales residuo divisor cantdec resultado contador) (if (= cantdec 0) resultado (+ 0 (decimales (remainder (* residuo 10) divisor) divisor (- cantdec 1) (+ resultado (/ residuo (expt 10 contador))) (+ contador 1)))))

Fuente: Elaboración colectiva

Al finalizar el proceso completo de implementación, codificación, ejecución y verificación de resultados, se solicitó una opinión de los estudiantes acerca de la metodología aplicada en el tema y además se realizó una evaluación que involucra el tema visto para tener una percepción tanto cualitativa como cuantitativa del aprendizaje del tema.

3. Resultados

Debe anotarse que, para obtener los resultados esperados, además de la función respectiva que se presenta en la Tabla 2, se han codificado adecuadamente instrucciones que permitan aceptar los datos y proceder en consecuencia con la solución del problema. La tabla 3 muestra los resultados obtenidos en cada uno de las soluciones.

Tabla 3. Resultados obtenidos

Paradigma	Ej.*	Resultado Obtenido
Paradigma Funcional **Lenguaje DrScheme** **IDE DrRacket**	1ª	> (interfaz1 1 1 5) DIVISION REAL CON CONTROL DE DECIMALES Dividendo: 20 Divisor: 6 Cant Decim:4 Resultado= 3.2222
	2a	> (interfaz1 1 1 1) DIVISION REAL CON CONTROL DE DECIMALES Dividendo: 20 Divisor: 6 Cant Decim:9 Resultado= 3.222222222
	3ª	> (interfaz1 1 1 1) DIVISION REAL CON CONTROL DE DECIMALES Dividendo: 20 Divisor: 6 Cant Decim:2 Resultado= 3.22
Paradigma Imperativo **Lenguaje C++** **IDE DevC++**	1ª	DIVISION REAL CON CONTROL DE DECIMALES Dividendo: 20 Divisor: 6 Cant Decim:4 Resultado= 3.2222
	2ª	DIVISION REAL CON CONTROL DE DECIMALES Dividendo: 20 Divisor: 6 Cant Decim:9 Resultado= 3.222222222
	3ª	DIVISION REAL CON CONTROL DE DECIMALES Dividendo: 20 Divisor: 6 Cant Decim:2 Resultado= 3.22

*Ejecución

Fuente: Elaboración propia

Se han tomado, como ejemplo, tres ejecuciones en cada lenguaje de programación. En la 1a ejecución se obtiene el resultado de una división con 4 decimales de precisión, en la 2ª ejecución se obtiene el resultado de la misma división con 9 decimales de precisión y en la 3ª ejecución se obtiene el resultado de la misma división, pero esta vez con 2 decimales de precisión. Es de anotar que la cantidad de decimales se mantiene siempre y cuando no correspondan a un valor 0 periódico.

Sobre la base de la metodología utilizada y del problema resuelto, se realizó una evaluación sobre el tema que aprovechara el conocimiento adquirido (nuevo conocimiento), y su relación con el conocimiento previo.

Esta evaluación consistió en 5 ejercicios similares o heredados al problema resuelto y se calificó en el rango 1 a 5 siendo 1 la nota más baja y 5 la nota más alta. Los resultados cuantitativos se presentan en la Tabla 4.

Tabla 4. Resultados cuantitativos de los estudiantes

Año	Sem	Cant. Est.	Prom. Notas	Notas < 3.0	Notas >= 3.0
2015	I	22	4.3	3	19
	II	25	4.5	4	21
2016	I	19	4.4	1	18
	II	20	4.7	0	20

Fuente: Elaboración propia

También se les solicitó a los estudiantes una opinión acerca de la metodología utilizada en el desarrollo del tema, de la presentación que sobre ella se hizo, de la solución al problema y la evaluación realizada.

La Tabla 5 presenta algunas opiniones de los estudiantes al respecto de lo cuestionado como forma de retroalimentación de la metodología.

Tabla 5. Opinión de los estudiantes

Año	Sem	Opinión 1	Opinión 2	Opinión 3
2015	**I**	Excelente la metodología, me gustó mucho la presentación inicial del proceso	Muy claro todo, además del profesor el método (sic) usado es muy entendible	Entendí todo
	II	Muy bueno el proceso,	Es claro que el objetivo	Me parece que esta

		lo preparan a uno muy bien	es que un aprenda, el profe es un verraco (sic)	metodología es tan buena que no depende del profesor
2016	**I**	Todo el proceso me pareció bastante bueno	Creo que no es fácil buscar un buen ejemplo y esta vez se logró	La oportunidad de opinar me parece muy buena
	II	Ojalá todos los profesores hicieran esto	Todo se ve muy fácil con esta metodología	Ojalá se mantenga esta metodología en otros temas y otras materias

Fuente: Elaboración propia

Para facilitar la interpretación de la opinión de los estudiantes, éstos se han tabulado en dos categorías (opiniones favorables y opiniones desfavorables).

Se consideran como opiniones favorables aquellas opiniones que involucran palabras de elogio y exaltación para la metodología utilizada. Son desfavorables aquellas que no cumplen con las expectativas de los estudiantes y que ellos así lo manifiestan. La Tabla 6 presenta los resultados obtenidos.

Tabla 6. Interpretación de la opinión de los estudiantes

Año	Sem	Op. Fav.	Op. Desf.	Total
2015	I	20	2	22
	II	24	1	25
2016	I	19	0	19
	II	19	1	20

Fuente: Elaboración propia

4. Discusión

Tal como se presenta en la Tabla 1, la metodología utilizada corresponde a la forma como se aborda la solución de un problema identificado completamente como computable, es decir, que puede ser resuelto con participación de la tecnología computacional. Si bien esta metodología puede detallarse a un nivel mayor, se ha acudido a la recomendación estándar que se utiliza en estos casos lo cual se traduce en tres partes claramente identificadas: una primera parte puramente humana en donde no se involucra la tecnología computacional, una segunda en donde se cristaliza lo planteado y propuesto en la primera y que requiere la presencia del computador con sus servicios, recursos y herramientas y una tercera parte en la cual se establece una verificación entre los resultados obtenidos y el objetivo propuesto.

Con el ánimo de que los estudiantes sean partícipes activos del proceso, la metodología se socializó para que ellos pudieran ir identificando cada fase en la cual se encontraban y, de paso, para que pudieran ir capitalizando los avances que se fueran dando en cada una de ellas. Es de anotar que esta metodología, en su fase de implementación en el computador, está mediada por el paradigma al cual se acude que, en este caso, corresponde a dos paradigmas: el funcional y el imperativo. La razón por la cual el paradigma influye en la parte puramente instrumental de la metodología radica en que la concepción lógica y el enfoque de implementación en un determinado lenguaje (ligado a dos paradigmas diferentes), involucra criterios, conceptos y elementos de juicio que han de tenerse en cuenta al momento de la codificación.

En la Tabla 2 se presenta el algoritmo, el programa en su versión imperativa y en su versión funcional. Es de anotar que si bien el algoritmo establece un camino lógico para la solución del problema planteado y que éste versa sobre el cálculo de un cociente controlando la cantidad de decimales que aparezcan en el resultado, en su versión imperativa se acude a los elementos que la programación estructurada provee, es decir, ciclo *while*, condicionales y secuencias de instrucciones. Por su parte, en su versión funcional, se hace uso de la recursión (capacidad que tiene una función para llamarse a sí misma), como mecanismo iterativo para construir procesos cíclicos. En esta misma versión, a partir de la recursión, se utiliza el mecanismo de alimentación de los argumentos con nuevos valores facilitar el proceso que sature la condición de finalización del proceso cíclico.

Es de anotar que en ambos casos se hace uso del concepto de función, tanto en la versión funcional como en la versión imperativa, dado que este concepto

simplifica y facilita la construcción de cualquier solución atomizándola a partir de la estrategia "divide and conquer" (divide y vencerás). Debe acotarse que es posible implementar diferentes soluciones para este problema a partir del paradigma imperativo y del paradigma funcional; la que se presenta es tan sólo una de ellas que, a juicio del autor de este artículo, resuelve el problema en unas condiciones favorables tanto para el procesamiento computacional como para el cálculo requerido. No puede desconocerse el hecho de que la recursión, en la versión funcional, implica un costo computacional notoriamente alto lo cual establece diferencias significativas con la versión imperativa en donde los conceptos de estado y variables hacen lo suyo en cuanto a los requerimientos computacionales.

En cuanto a los resultados obtenidos que se presentan en la Tabla 3 debe anotarse que se han seleccionado solo tres ejecuciones, de muchas realizadas, para ejemplificar la efectividad tanto del algoritmo como de los respectivos programas. Como puede verse se ha logrado que la interfaz sea exactamente la misma para efectos de poder hacer comparaciones pertinentes. En ambos casos ha funcionado el programa tal como se esperaba, es decir, obteniendo 4, 9 y 2 decimales en el cálculo de un cociente con decimales con periodo definido. Esto implica que aquellos requerimientos en donde se necesite calcular un cociente que involucre decimales, pero en los cuales el valor requerido sea superior a la cantidad de decimales posibles, no se rellena con ceros, sino que simplemente llega hasta donde sea posible llegar.

En ambas versiones de la solución algorítmica, se solicita el dividendo, el divisor y la cantidad de decimales de forma que, al final, se muestre el resultado obtenido con la cantidad requerida de dígitos decimales. La única diferencia notoria en las dos interfaces de resultados radica en el llamado a los programas que en la versión funcional implica invocar una función llamada *interfaz* con argumentos *1 1 1* (que corresponden a argumentos falsos solo para que la función inicie su funcionamiento –valga la redundancia-). Debe aclararse que, aunque se seleccionaron tres ejecuciones con los mismos valores para mostrar la validez de los resultados, fueron muchas las pruebas que se hicieron con la mira de verificar plenamente la cantidad de decimales requeridos.

En la versión imperativa se tiene la limitante del tipo de datos pues cuando se trabaja con variables de tipo entero (como en la solución que se muestra), las fronteras de almacenamiento limitan las pruebas dada la capacidad límite que éstas tienen. Por su parte, en la versión funcional, se pueden llegar a obtener resultados mucho más amplios puesto que no trabaja con el concepto de tipos de datos y, por lo tanto, su naturaleza funcional posibilita unos resultados mucho más amplios en cantidad y en proceso. Este detalle le da una gran ventaja, en este tipo de soluciones, al paradigma funcional apoyado en un lenguaje como DrScheme por encima del paradigma imperativo apoyado en un lenguaje como C++ arista estructurada. Debe aclararse también que, en un lenguaje basado en el paradigma imperativo, es posible resolver el problema de los límites que imponen los tipos de datos acudiendo a las estructuras de datos, al manejo de cadenas o a la simulación computacional, pero estos son temas que no competen al presente artículo.

La Tabla 4, que presenta los resultados cuantitativos de los estudiantes, invita a realizar una observación detallada de los datos que allí se muestran. Por una parte, los promedios de las notas, teniendo en cuenta la objetividad con que se construyó la evaluación, son notoriamente favorables si se les mira desde el hecho que el valor máximo de calificación es 5.0. Los promedios de cada semestre indican que las notas de esta evaluación han tenido una tendencia a ser más altas que bajas, es decir, si bien se muestra la cantidad de notas con valor menor que 3.0, la proporción de notas mayores o iguales a 3.0 es mucho mayor. En ese grupo de datos, los valores superiores al promedio son mayores que la tendencia hacia los valores menores puesto que solo así se podría lograr que el promedio fuera tan alto.

En la Tabla 5 se presentan algunas opiniones cualitativas seleccionadas de los estudiantes. Para ello se abrió un espacio anónimo, libre y espontáneo en donde cada estudiante pudiera manifestar su opinión y, de esa forma, poder retroalimentar el proceso. Es de anotar que las opiniones de los alumnos han sido notoriamente favorables y en ellas han destacado cinco factores que llaman mucho la atención: a) se sienten muy complacidos con el hecho de que se les haga partícipes, y se les dé a conocer la metodología y el proceso que se va a realizar, b) se involucran plenamente en dichos procesos e intentan sacar el

mayor proceso de cada fase de la metodología, c) participan en la construcción de la solución tanto en lo que corresponde al planteamiento algorítmico como en lo que compete al diseño a partir de funciones (divide and conquer), d) se sienten muy a gusto con la evaluación y la miran como un camino para retroalimentar lo aprendido y e) manifiestan alta complacencia en tener la oportunidad de opinar acerca de todo el proceso. La libertad que se les confiere les permite opinar de una forma como pocas veces lo hacen y son esas opiniones las que hacen que el proceso pueda tener, cada vez mayor éxito, no en vano los resultados cuantitativos, y especialmente el promedio, es cada vez más alto con el paso del tiempo tal como se verifica en la Tabla 4.

De la misma manera, a partir de la retroalimentación que los estudiantes han hecho al proceso, se puede verificar que cada vez son menos las opiniones desfavorables y cada vez son más las opiniones favorables. Nótese que en el I semestre de 2016 no hubo una sola opinión desfavorable y absolutamente todas fueron favorables tal como se presenta en la Tabla 6.

5. Conclusiones

Es posible encontrar, entre enunciados que son muy familiares para los estudiantes, soluciones alcanzables, fáciles de probar y mucho más fáciles de implementar, problemas que puedan ser resueltos aprovechando la capacidad de procesamiento de los computadores modernos. El problema del control de decimales en el cálculo de un cociente real ha sido un problema recurrente a lo largo de los tiempos, es hora de pensar en otras situaciones que, partiendo de soluciones simples, puedan resolverse acudiendo a los fundamentos más elementales de las matemáticas tal como es el caso de la multiplicación concebida como una secuencia finita de sumas sucesivas y la división concebida como una secuencia finita de restas sucesivas.

Socializar una metodología de aprendizaje con los estudiantes antes de empezar a aplicarla permite que ellos conozcan plenamente tanto los objetivos como las fases que se han de recorrer y, de esta manera, puedan articularse tanto con los avances como con el cumplimiento de los propósitos. El estudiante se siente importante y, en rima con eso, asume un rol más proactivo dentro de estos procesos de aprendizaje de manera que es posible que bajo el manto de lo innovador y la fascinación se puedan alcanzar objetivos de apropiación de conocimiento por un camino más expedito.

La implementación de un mismo algoritmo en diferentes paradigmas de programación y usando diferentes lenguajes de programación permite establecer comparaciones entre cada uno de ellos posibilitando capitalizar, en cada uno, sus ventajas en pro tanto del costo de procesamiento computacional como a favor de los objetivos que se quieran lograr. Un buen ejercicio de programación, en la medida en que se avanza en una carrera como Ingeniería de Sistemas y Computación, consiste en codificar una misma solución en diferentes lenguajes de programación y, particularmente, bajo el enfoque de diferentes paradigmas de programación. Eso le permitirá al estudiante adquirir una confianza y seguridad mayor en los procesos de concepción, implementación y puesta a punto de un programa además de posibilitar la comprensión de la programación desde su perspectiva algorítmica y no desde la perspectiva puramente sintáctica asociada a un lenguaje de programación. Vale la pena aprovechar este tipo de ejercicios para capitalizar lo mejor de cada paradigma y, por su conducto, de cada lenguaje de programación.

La interfaz de usuario siempre será el camino a través del cual se le comunican los resultados a la persona que está utilizando el programa y vale la pena que sea tan claro posible no para que conozca el proceso que internamente se está realizando sino para que tenga herramientas suficientes para entender qué es lo que realmente hace y, de esa forma, poder aprovecharlo en su correcta dimensión. En todo proceso investigativo de estas características, la retroalimentación por parte de los estudiantes es vital para incorporar elementos que hagan que cada vez que se aplique sea más exitoso. La realización de pruebas escritas que permitan la validación o, al menos, la aproximación al nivel de apropiación del nuevo conocimiento, así como los espacios de opinión abiertos y libres para los estudiantes, desde una perspectiva anónima, facilitan el camino para que este tipo de investigaciones abran nuevos caminos para el aprendizaje en la formación de los ingenieros que la sociedad necesita.

Referencias

1. Attard, A., Di Ioio, E., & Geven, K. (2010). Student Centered Learning. An insight into theory and practice. Bucarest: Lifelong learning programme - European Community.

2. Ausubel, D. (1963). Psychology of meaningful verbal learning: an introduction to school learning. New York: Grune & Straton.

3. Azad, A., & Smith, D. (2014). Teaching an introductory programming language in a general education course. Journal of Information Technology Education: Innovations in Practice, 13, 57-67.

4. Barriga Arceo, F., & Hernandez Rojas, G. (2002). Estrategias docentes para un aprendizaje significativo: una interpretación constructivista. Ciudad de México: McGraw Hill Interamericana.

5. Blanchard, B. (2000). Ingeniería de Sistemas. Madrid (España): Isdefe.

6. Boyer, C. (2010). Historia de la Matemática. Madrid (España): Alianza Editorial.

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

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

9. Bruner, J. S. (2009). Actos de significado: Mas allá de la revolución cognitiva. Madrid: Alianza Editorial.

10. Chavarría Olarte, M. (2004). Educación en un mundo globalizado: retos y tendencias del proceso educativo. México: Trillas.

11. Coronado Padilla, J. (2013). Sistemas numéricos residuales: fundamentos lógico matemáticos. Bogotá: Universidad de la Salle.

12. De Zubiría Samper, J. (2006). Los modelos pedagógicos: hacia una pedagogía dialogante. Bogotá: Cooperativa Editorial Magisterio.

13. Felleisen, M., Findler, R., Flat, M., & Krishnamurthi, S. (2013). How to design programas (2a Ed. ed.). Boston: MIT Press.

14. Jiménez Murillo, J. (2014). Matemáticas para la computación. Ciudad de México: Alfaomega.

15. Kline, M. (2012). El pensamiento matemático de la antiguedad a nuestros días. Madrid: Alianza Editorial.

16. Schildt, H. (2010). C Programming. México: McGraw Hill.

17. Schildt, H. (2010). The complete reference C (4a. Ed. ed.). Berkeyley, USA: McGraw Hill.

18. Small, G. (2011). El cerebro digital. Madird: Editorial Urano.

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

20. Trejos Buriticá, O. I. (2006). Fundamentos de Programación. Pereira: Papiro.

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