Entre Ciencia e Ingeniería, vol. 14, no.28, julio-diciembre de 2020, páginas 34-39. DOI: https://doi.org/ 10.31908/19098367.2013.
ISSN 1909-8367 (Impreso), ISSN 2539-4169 (En línea)

Aprendizaje de la programación con estrategia "divide and conquer" vs. Sin estrategia "divide and conquer"[1]

Learning computer programming using "divide and conquer" strategy vs. without "divide and conquer strategy"

O.I. Trejos y L.E. Muñoz

Recibido: septiembre 10 de 2019 – Aceptado: septiembre 29 de 2020.

[1]***Resumen*** **--El aprendizaje de la programación de computadores es una invitación permanente a docentes ingenieros para buscar mecanismos, teorías y modelos que lo faciliten y, de paso, que simplifiquen la resolución de problemas a partir del aprovechamiento del pensamiento computacional. El concepto de función y la estrategia "divide and conquer" se han ido convirtiendo en un camino que facilita la asimilación y aplicación de la programación dentro del paradigma funcional y, al tiempo, simplifica el aprendizaje de otros paradigmas de programación. El presente artículo está basado en una investigación realizada en paralelo con grupos de Programación I Paradigma Funcional a lo largo de los últimos 6 semestres en el programa Ingeniería de Sistemas y Computación. Los resultados son significativamente diferentes si se comparan los grupos en los cuales se adoptó la estrategia "divide and conquer" con los resultados de aquellos con los cuales se trabajó una sola función que incluyera todo el proceso lógico de resolución de un problema. Se concluye que es mucho más conveniente atomizar una solución algorítmica computacional basada en funciones que pensar dicha solución en un solo cuerpo lógico, independiente del paradigma de programación.**

Palabras clave**--Aprendizaje, Funciones, Ingeniería de sistemas, Metodología, Programación.**

Abstract **--Learning computer programming is a permanent challenge for teacher engineers to find mechanisms, theories, and models that facilitate it and simplify the resolution of problems from the use of computational thinking. The concept of a function and the "divide and conquer" strategy facilitates the assimilation and application of programming within the functional paradigm and, at the same time, simplifies the learning of other programming paradigms. This article is based on an investigation made in parallel with groups of Programming Functional Paradigm throughout the last 6 semesters in the Systems and Computing Engineering program. The results are significantly different when comparing the groups in which the "divide and conquer" strategy was adopted with the results of those with a single function was worked that included the entire logical process of solving a problem. It is concluded that it is much more convenient to atomize a computational algorithmic solution into independent functions than think that solution in a single logical body independent of the programming paradigm.**

Keywords**--Functions, Learning, Methodology, Programming, Systems Engineering.**

I. INTRODUCCIÓN

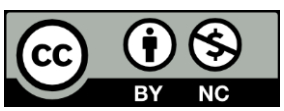
ENSEÑAR programación de computadores pareciera ser una labor tan rutinaria que se ha convertido en un lugar común para cualquier persona, que alguna vez haya ejecutado un programa, considera que está en condiciones de hacerlo. Sin embargo, no hay tal. Enseñar a programar implica abrir caminos para que los estudiantes abandonen la lógica humana deliberativa y apropien los conceptos, modelos y teorías de la lógica computacional (y con ella, el pensamiento computacional) para resolver los problemas que sean computables [1], es decir, aquellos que se pueden solucionar con la participación activa de un computador.

La otra orilla de la enseñanza de la programación está basada en el aprendizaje de la misma, es decir, en poder contar con mecanismos, metodologías y estrategias que permitan verificar que efectivamente el estudiante ha asimilado los conceptos propios de la lógica computacional para resolver problemas [2] al punto de que estén por encima de su propia lógica humana al momento de implementar una solución en un computador.

[1] Artículo derivado de la Tesis Doctoral "Aprendizaje en Ingeniería: un Problema de comunicación", Ciencias de la Educación, Universidad Tecnológica de Pereira.

O.I. Trejos, Universidad Tecnológica de Pereira, Pereira, Colombia, email: omartrejos@utp.edu.co.
L.E. Muñoz, Universidad Tecnológica de Pereira, Pereira, Colombia, email:lemunozg@utp.edu.co.

Como citar este artículo: Trejos, O. I. y Muñoz, L.E. Aprendizaje de la programación con estrategia "divide and conquer" vs. Sin estrategia "divide and conquer", Entre Ciencia e Ingeniería, vol. 16, no. 28, pp. 34-39, julio-diciembre, 2020. DOI: https://doi.org/ 10.31908/19098367.2013.



El proceso tanto de enseñanza y aprendizaje de la programación es, por su misma naturaleza, uno de los retos más grandes que tienen los docentes de esta área. A nivel universitario normalmente estas asignaturas son dictadas por ingenieros que, en su proceso de formación, no tienen la componente pedagógica ni docente [3], razón por la cual el problema no solo se vuelve más complejo de resolver sino que, en muchas oportunidades, termina siendo irresoluto pues los estudiantes quedan en manos de profesores que con dificultad adoptan estrategias efectivas de enseñanza y aprendizaje y, peor aún, profesores que muchas veces no tienen los conceptos de programación suficientemente sólidos como para enseñarlos [4].

El proceso de transformación y apropiación de la lógica humana hacia la lógica computacional es un proceso de gran complejidad invertida debido a que se va de un escenario de pensamiento altamente complejo hacia un escenario que se basa en un contexto, en unas reglas y en un significado [5] tan sencillo que, por momentos, los mismos estudiantes parecieran confundirse sin darse cuenta que están yendo de algo complejo a algo mucho más simple.

En el presente artículo se expone tanto la experiencia como los resultados obtenidos y la discusión asociada con una investigación que propende por comprobar la efectividad del uso de la estrategia "Divide y Vencerás" basado en funciones para implementar un programa frente a la resolución de problemas computables sin esta estrategia y partiendo solamente del uso de un solo cuerpo lógico en donde se condensen todos los elementos que solucionan el problema.

La novedad del artículo se presenta en tres sentidos: a) se realiza un estudio comparativo entre grupos paralelos de la misma asignatura, b) se desarrolla dicha comparación desde la perspectiva de la metodología de la investigación y c) se realizan aportes frente a la discusión permanente sobre la conveniencia de utilizar el concepto de función y la estrategia "Divide y Vencerás" en un curso introductorio de programación de cara a la apropiación y aprendizaje posterior de otros paradigmas de programación.

Este estudio se justifica toda vez que la programación de computadores cada vez se ha ido convirtiendo, por su misma naturaleza de aplicación en el mundo moderno, en un área de conocimiento que podría considerarse entre las ciencias básicas de hoy, es decir, en aquellos saberes que todo profesional del mundo moderno debiera apropiar, aprender, aplicar, retroalimentar y evaluar. La iniciativa Code.Org (Iniciativa de Programación Mundial, s.f.) es un buen ejemplo de ello pues propende porque la programación de computadores, el pensamiento computacional, el pensamiento crítico, el aprovechamiento de las nuevas tecnologías y la algoritmización de las soluciones [6] se conviertan en bastiones temáticos no solo de los primeros semestres de los programas de ingenierías sino de todos los programas universitarios así como de los últimos años de la formación básica secundaria en todo el mundo. Esto represente la relevancia que se le ha ido concediendo a la programación de computadores como área temática.

Para el desarrollo de esta investigación se acudió al estudio detallado del paradigma de programación funcional (lo cual incluye una inmersión en el concepto de función y la estrategia "Divide y Vencerás") [7], el estudio de la programación de computadores y los paradigmas modernos de programación [8], la aplicación de la estrategia mencionada en la programación imperativa [9] y la programación orientada a objetos [10], los conceptos que provee la docencia en lo que compete a la enseñanza y el aprendizaje de un saber tecnológico [11] y la aplicación y presencia de la Ingeniería de Sistemas y Computación en el mundo actual.

Al respecto de este tipo de investigación se han realizado algunos experimentos que propenden por buscar caminos que faciliten el aprendizaje de la programación y, con ello, también se simplifique la enseñanza de la misma haciendo más efectivos los esfuerzos que los ingenieros docentes hacen por llevar dichos conocimientos a sus alumnos y cambiar la base cognitiva de sus conocimientos previos, incluyendo los que provee la lógica humana deliberativa, para que los nuevos conocimientos de la mano de la lógica computacional posibiliten la solución sistemática de problemas computables.

Este artículo es uno de los productos de la Tesis Doctoral "Aprendizaje en Ingeniería: Un problema de comunicación" que ha posibilitado el desarrollo de una segunda Tesis Doctoral titulada "Modelo de socialización del conocimiento a partir del aprovechamiento de nuevas tecnologías, redes sociales y servicios asociados y del desarrollo de competencias blandas de incorporación en grupos interdisciplinarios de los estudiantes, en el proceso de aprendizaje de la programación de computadores. Caso: Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira" [12]. Todo el estudio estadístico y los datos recogidos obedecen a un proceso de levantamiento de información que se realizó desde el I semestre de 2017 hasta el II semestre de 2019 en el curso Programación I de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira.

El contenido de este artículo está organizado según el estándar internacional IMRYD [13] según el cual se expone una introducción y un marco teórico, seguido de una metodología y unos resultados sobre los cuales se realiza una discusión y análisis de los mismos para llegar a unas conclusiones en relación con el objetivo del estudio o investigación. La investigación llega hasta el punto en que compara los resultados cuantitativos y cualitativos de los grupos paralelos que se incluyeron en la investigación y de los cuales con uno de ellos se adoptó la estrategia "Divide y Vencerás" y con el otro no se adoptó dicha estrategia, sino que se condujo el curso de programación funcional a partir de la construcción de una sola unidad lógica de desarrollo tal como se explicará más adelante.

II. MARCO TEÓRICO

La programación de computadores puede definirse como el camino a través del cual se logra alcanzar un objetivo que resuelve un problema computable aprovechando las ventajas y posibilidades de un lenguaje de programación que, a su vez, es un conjunto de instrucciones u órdenes entendibles y realizables por el computador [14]. La programación de computadores es una de las formas como se han capitalizado los dispositivos electrónicos con el ánimo de poder aprovechar su alta velocidad de procesamiento, sus capacidades de

almacenamiento y de respuesta y su posibilidad de organizar, tratar y manejar la información, todo ello, en pro de solucionar situaciones problema que siendo computables aquejan a la sociedad.

Los lenguajes de programación obedecen a un enfoque en su utilización que se conoce como paradigma y que, en palabras sencillas, es una perspectiva desde la cual se mira el problema computable y se resuelve con posibilidades específicas de las tecnologías y dispositivos modernos de manera que se haga más sencilla, manejable y efectiva dicha solución [15]. La apropiación de los paradigmas de programación se hace por varios caminos. Uno de ellos corresponde al estudio detallado del modelo matemático que subyace a la expresión tecnológica que lo encarna. El paradigma de programación funcional, por ejemplo, se basa en el cálculo Lambda según el cual se priorizan las capacidades y velocidades de procesamiento en un sistema computable por encima de cualquiera de sus otras características.

Por su parte, el paradigma de programación imperativa obedece a la aritmética que se hereda de la lógica de proposiciones y que prioriza el concepto de estados como gran bastión del aprovechamiento de los computadores y que le concede a la memoria del computador, y a sus posibilidades, un nivel superior a cualquiera de sus otras características [9]. El paradigma de programación orientada a objetos va mucho más orientado al modelo matemático que intenta interpretar la realidad a partir de la conjunción de atributos y métodos teniendo en cuenta que son los atributos cada una de las características que distinguen un objeto de otro y son los métodos aquellos usos en los cuales dichos objetos son útiles [16].

En tiempos modernos ha tomado gran importancia el pensamiento computacional como camino para capitalizar el pensamiento crítico en las diferentes áreas del conocimiento, el aprovechamiento de las nuevas tecnologías de la información y la comunicación y la algoritmización de la resolución de problemas [17], todas ellas como base para la formación de un pensamiento que simplificando el entorno en su concepción, facilita la resolución de los problemas que dicho entorno encarna.

En el paradigma de programación funcional, el protagonismo lo tiene el concepto de función y la estrategia "Divide y Vencerás" que corresponde a la buena utilización y atomización de soluciones computables a través de la implementación, enlace y articulación entre las funciones como células de trabajo. Por su parte el paradigma de programación imperativo, al priorizar la memoria también le concede vida al concepto de procedimiento lo cual no implica que no se pueda adoptar le estrategia en mención para simplificar las soluciones planteadas e implementadas en este paradigma. En el caso de la programación orientada a objetos, la adopción de la estrategia "Divide y Vencerás" se hace a través del enlace entre los métodos, particularmente dentro de un mismo objeto lo cual facilita, como en todas las demás, la resolución de los errores lógicos y la interpretación dinámica de la solución para posibles cambios posteriores.

Todo esto tiene un sentido cuando, en el proceso de formación de ingenieros, tiene un significado que implica el enlace, desplazamiento o reemplazo de los conocimientos previos por los nuevos conocimientos además de la actitud del estudiante, representada en la motivación para aprender [18]. La programación tiene un especial relevancia en el mundo moderno toda vez que la penetración de los dispositivos electrónicos, de sus servicios, de las redes sociales y de todos los elementos conceptuales y prácticos que les son asociados, ha permitido que la sociedad moderna esté permanentemente interconectada y vea en las nuevas tecnologías de la información y la comunicación el camino posible para acceder a fuentes de información inimaginables, a caminos de comunicación inexplorados y, sobre todo, a soluciones que resuelven problemas de la vida cotidiana.

La enseñanza y aprendizaje de la programación y los caminos investigativos que han de explorarse desde la docencia ejercida por ingenieros, invita a que se piense en que cualquier esfuerzo por simplificar la apropiación, asimilación, utilización, retroalimentación y evaluación de la programación de computadores debe convertirse en un reto permanente de los profesores de esta área toda vez que los estudiantes tienen acceso a mayor información, tienen una mente mejor articulada con los cambios tecnológicos modernos y tiene capacidades de articularse con un mundo digital que se ha ido convirtiendo en su lenguaje natural. Posiblemente sea esta definición la que fortalece gran parte de la razón de ser de un programa como la Ingeniería de Sistemas que, además de formar ingenieros con capacidad de resolver problemas de la sociedad aprovechando los recursos que la ciencia y tecnología moderna proveen, también son capaces de encontrar nuevos y mejores caminos tanto para la utilización y aplicación de los mismos así como para buscar aplicaciones y nuevas soluciones a los problemas ya existentes de la sociedad a la cual se le deben.

III. METODOLOGÍA

La estrategia "Divide y Vencerás" puede resumirse en un ejemplo para efectos de entender lo que se realizó con los grupos que participaron en la presente investigación y debido a que es la espina dorsal de la programación, tal como se propone en este artículo, y de la metodología adoptada. Se presentará la diferencia a partir del siguiente enunciado: Construir un programa que lea un número de 3 dígitos y muestre el resultado de sumar dichos dígitos.

Una versión solución de este enunciado es la que se muestra a continuación en la cual la codificación está escrita en DrRacket y se está utilizando el paradigma de programación funcional con una sola función como base para la solución. Es una solución corta, digerible y entendible.

```
------------------------------------------------------------------------
; VERSIÓN 1.0
; Sin estrategia "Divide y Vencerás" (Una sola función)

; OBJETIVO
; Construir un programa que lea un número de 3 dígitos y muestre el
resultado de sumar dichos dígitos

; ANÁLISIS EPS
; Entrada -> un número que digita el usuario
; Proceso ->
;     a. Leer el número
;     b. Verificar que sea de 3 dígitos
;     c. Si lo es
;         c.1. Separar el último dígito
;         c.2. Separar el penúltimo dígito
;         c.3. Separar el antepenúltimo dígito
```

```
;       c.4. Obtener la suma de estos tres dígitos
;       c.5. Mostrar la suma de dichos dígitos
;     d. Si no lo es, mostrar aviso de "ERROR"

(define (programa num) ; Inicio de la función
  (display "Número: ") ; Aviso de solicitud
  (set! num (read))    ; Lectura de valor digitado por usuario
  (if (and (>= num 100)(<= num 999)) ; Si valor es de 3 dig
      (begin          ; Inicio de acciones, si el condicional es V
        (display "La suma de los 3 dígitos es ") ; título alusivo
        (display (+ (remainder num 10)   ; muestre suma 1er dig
                    (remainder (quotient num 10) 10) ; mas 2o dígito
                    (remainder (quotient num 100) 10))) ; mas 3º dig
      )
   (display "E R R O R") ; Si el número no es de 3 dígitos, mostrar "ERROR"
   )       ; fin del condicional
  )        ; fin de la función

(programa 1)  ; Llamado inicial de la función
```

Seguidamente se presentará la solución del mismo enunciado bajo la estrategia "Divide y Vencerás" codificado también en lenguaje DrRacket paradigma de programación funcional con uso de varias funciones entrelazadas.

```
; VERSIÓN 2.0
; Con estrategia "Divide y Vencerás (varias funciones enlazadas)

; OBJETIVO
; Construir un programa que lea un número de 3 dígitos y muestre el
resultado de sumar dichos dígitos

; ANÁLISIS EPS
; Entrada -> un número que digita el usuario
; Proceso ->
;     a. Leer el número
;     b. Verificar que sea de 3 dígitos
;     c. Si lo es
;        c.1. Separar el último dígito
;        c.2. Separar el penúltimo dígito
;        c.3. Separar el antepenúltimo dígito
;        c.4. Obtener la suma de estos tres dígitos
;        c.5. Mostrar la suma de dichos dígitos
;     d. Si no lo es, mostrar aviso de "ERROR"
; Salida -> Suma de los 3 dígitos o aviso de ERROR

;********** CÓDIGO **********

; Función que verifica si un valor es de 3 dígitos
(define (verif3dig n) ; Definición de función
  ; Si el valor recibido es de 3 dígitos
  (if (and (>= n 100)(<= n 999))
      n   ; retorne el mismo valor
      0)  ; sino, retorne 0
  )       ; fin de la función verif3dig

; Función que obtiene el último dígito de un valor numérico entero
(define (ultdig n) ; Definición de función
  (remainder n 10) ; retorne el último dígito del valor recibido
  )                ; fin de la función ultdig

; Función que obtiene el penúltimo dígito de un valor numérico entero
(define (penultdig n) ; Definición de la función
   ; retorne el penúltimo dígito del valor recibido
  (remainder (quotient n 10) 10)
  )                  ; fin de la función penultdig

; Función que obtiene el antepenúltimo dígito de un valor numérico entero
(define (antepenultdig n) ; Definición de la función
  ; retorne el antepenúltimo dígito del valor recibido
  (remainder (quotient n 100) 10)
  ) ; fin de la función antepenultdig

; Función que recibe 3 valores y retorna su suma
(define (suma3val n) ; Definición de la función
  (+ (ultdig n)      ; suma el último dígito del valor leido
    (penultdig n)    ; mas el penúltimo dígito del valor leido
    (antepenultdig n)) ; mas el antepenúltimo dígito del valor leido
  )                  ; fin de la función suma 3val

; Función de Interfaz
(define (interfaz n) ; Definición de la función
  (display "Número: ") ; Aviso de solicitud
  (set! n (read)) ; Lectura de un valor digitado por el usuario
  (if (= (verif3dig n) n) ; Si el valor es de 3 dígitos
      (begin          ; Inicio de acciones si la condición es V
        "La suma de los 3 dígitos es " ; Aviso alusivo
        (display (suma3val n)) ; Muestre la suma de los 3 dígitos
        )             ; Fin de acciones si la condición es V
      ; Si el valor no es de 3 dígitos, muestre ERROR en pantalla
      (display "E R R O R")
    ) ; fin del condicional
  ) ; fin de la función interfaz

(interfaz 1)  ; Llamado inicial de la función
```

Se puede observar que los dos códigos anteriores son diferentes en cuanto a su enfoque. Al ejecutarse, el resultado de ambos es el mismo y ambos cumplen con sus objetivos. La tabla I presenta las ventajas y desventajas de estos dos enfoques.

TABLA I
COMPARACIÓN DE DOS ENFOQUES DE PROGRAMACIÓN
FUENTE: ELABORACIÓN PROPIA

Ítem	Con estrategia "Divide y Vencerás"	Sin estrategia "Divide y Vencerás"
Ventajas	• Se atomiza la solución • Cada función es más sencilla • Es más fácil encontrar errores lógicos • Es más flexible al momento de cambiar una parte del código • Facilita la reutilización del código • Simplifica el objetivo del programa	• Es más corto el código • Por momentos, podría ser más fácil • Es más directo en cuanto a la solución • Es más ágil su procesamiento • La resolución del objetivo podría ser un poco más compleja
Desventajas	• Requiere mayor conocimiento de programación • Requiere planeación y análisis antes de empezar a codificar • Requiere buen conocimiento técnico para enlazar las funciones	• Es más complejo encontrar errores lógicos a medida que la función tiene más líneas de código • Requiere mayor conocimiento del lenguaje de programación

Este ejercicio se ha tomado tan solo como un ejemplo para presentar las características e implicaciones de la adopción de la estrategia "Divide y Vencerás". Es de anotar que si el enunciado a resolver fuera un problema con mayor complejidad, su solución también lo sería y por lo tanto las desventajas de no utilizarla tomarían más fuerza en relación con las ventajas comparativas frente a la utilización de esta estrategia.

En el proceso investigativo, se seleccionaron dos grupos de Programación I paralelos. Con uno de ellos se adoptó la estrategia "Divide y Vencerás" y con el otro no. Esta metodología se mantuvo durante todo el período académico

desde el I semestre del 2017 hasta el II semestre del 2019. Se realizaron 3 parciales conjuntos en ambos grupos, es decir, se reunieron los dos grupos para que lo resolvieran. De igual manera se procedió con el examen final. Los enunciados, problemas y planteamientos a resolver en las evaluaciones escritas fueron exactamente iguales en ambos casos con el ánimo de garantizar la máxima objetividad posible.

IV. Resultados

La tabla II presenta los estudiantes que participaron en la investigación. Las tablas IIIA y IIIB presentan los resultados de las evaluaciones parciales y examen final, con estrategia "Divide y Vencerás" y sin ella respectivamente.

TABLA II
ESTUDIANTES PARTICIPANTES EN LA INVESTIGACIÓN
FUENTE: ELABORACIÓN PROPIA

Año	Sem	Grp con estrat. "DyV"	Grp sin estrat. "DyV"	Total por Sem
2017	I	22	23	45
	II	20	21	41
2018	I	20	24	44
	II	24	20	44
2019	I	21	21	42
	II	21	18	39
Total x estat		128	127	255

TABLA III A
RESULTADOS CUANTITATIVOS DE PRUEBAS ESCRITAS – GRUPOS CON ESTRATEGIA "DYV"
FUENTE: ELABORACIÓN PROPIA

Año	Sem	I P	II P	III P	E F	Prom x Sem
2017	I	4,5	4,6	4,6	4,5	4,6
	II	4,6	4,5	4,7	4,4	4,6
2018	I	4,3	4,5	4,8	4,6	4,6
	II	4,4	4,2	4,7	4,7	4,5
2019	I	4,4	4,5	4,7	4,6	4,6
	II	4,5	4,3	4,6	4,6	4,5
Prom x Eval		4,5	4,4	4,7	4,6	4,5

"DyV" = Divide y Vencerás

TABLA III B
RESULTADOS CUANTITATIVOS DE PRUEBAS ESCRITAS – GRUPOS SIN ESTRATEGIA "DYV"
FUENTE: ELABORACIÓN PROPIA

Año	Sem	I P	II P	III P	E F	Prom x Sem
2017	I	3,4	3,5	3,4	3,1	3,4
	II	3,4	3,6	3,3	3,2	3,4
2018	I	3,2	3,4	3,4	3,0	3,3
	II	3,1	3,5	3,2	3,1	3,2
2019	I	3,2	3.4	3,3	3.1	3,3
	II	3,3	3,5	3,1	3,2	3,3
Prom x Eval		3,3	3,5	3,3	3,1	3,3

"DyV" = Divide y Vencerás

Para facilitar el análisis se han promediado las notas de los cursos que participaron en la investigación dado que el promedio, como medida de tendencia central, es una medida que facilita mejor la interpretación de resultados dado que se afecta por los resultados extremos que, en este caso, son muy importantes.

V. Discusión y análisis de resultados

En primer lugar, debe tenerse en cuenta la comparación que se hace en la tabla I pues es claro que la adopción de la estrategia "Divide y Vencerás" es un buen camino para simplificar la lógica de programación y, con ello, la resolución de problemas computables. Más allá de las posibles desventajas comparativas que una estrategia como esta puede tener (tal como se especifica en la tabla), sus ventajas son notoriamente favorables para el desarrollo de una solución que se pueda implementar en un computador. Desde la atomización de la solución lo cual implica una mirada más detallada y simple desde la perspectiva del programador, hasta la construcción de funciones sencillas pasando por la localización fácil de los errores lógicos, la flexibilidad en los cambios, la reutilización del código que, en últimas, se traducen en la simplificación del objetivo y, por ende, en la solución del programa.

Por su parte la tabla II presenta una cantidad apreciable de estudiantes pues durante el tiempo en el cual se realizó la investigación, pasaron por las aulas del curso Programación I de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira aproximadamente 2000 alumnos lo cual establece que la participación en el estudio que inspira este artículo fue de un 13% aproximadamente que, en términos estadísticos, es una muestra suficiente para que las inferencias puedan generalizarse, con un margen de error aceptable, a la población estudiada que en este caso correspondería a los estudiantes de I semestre del programa en mención.

Los resultados cuantitativos de las pruebas parciales en los grupos en los cuales se adoptó la estrategia "Divide y Vencerás" son notoriamente favorables hacia ellos si se compara con los grupos en los cuales no se adoptó dicha estrategia. Esto es altamente significativo si se tiene en cuenta que se realizaron las mismas pruebas, el mismo contenido evaluativo y se realizaron en las mismas condiciones conjuntamente para garantizar la mayor objetividad posible. De la misma manera la diferencia entre la Evaluación Final (EF) de los grupos con estrategia "Divide y Vencerás" y de los grupos sin esta estrategia, ha sido completamente favorable a los primeros. En conjunto, la diferencia global es de 1,2 que, en términos cuantitativos, representa una efectividad del 24% que corresponde a la cuarta parte del rendimiento cuantitativo del curso tal como se reflejan estas inferencias en las tablas IIIA y IIIB.

VI. Conclusiones

En términos del objetivo de esta investigación según el cual se cuestionaba si era favorable y positivo adoptar la estrategia "Divide y Vencerás" para el aprendizaje de la programación de computadores en un primer curso de programación en una carrera de Ingeniería de Sistemas y Computación, la respuesta es un absoluto SI frente a la adopción de dicha estrategia. "Divide y Vencerás", como estrategia de aprendizaje y aplicación de los fundamentos de la programación y su posterior uso en otros paradigmas, permite que el estudiante encuentre un camino para atomizar la solución desde una perspectiva técnica, lógica y tecnológica; diseñe funciones

cada vez más sencillas y encuentre fácilmente los errores lógicos; flexibilice el código cuando se trate de realizar cambios; reutilice el código y simplifique la concepción y fragua de la solución a alcanzar en relación con el objetivo a lograr.

La investigación en la enseñanza y el aprendizaje a nivel de los programas de ingeniería es una arista que los docentes ingenieros deben comenzar a pensar con detenimiento puesto que se pueden encontrar, en el mismo conocimiento disciplinar, la solución a lo que se ha querido buscar como es el de encontrar caminos lógicos que modifiquen la base cognitiva del estudiante de forma que puedan aplicar la lógica y el pensamiento computacional para resolver problemas que, sin la tecnología, deberían acudir a la lógica deliberativa humana.

Referencias

[1] Trejos Buriticá, O. Lógica de Programación. Bogotá: Ediciones de la U, 2017.

[2] Wing, J. Computational Thinking. Communications on the ACM, 49(3), p.p. 33-35, 2006.

[3] Brown, N., & Wilson, G. (2018). Ten quick tips for teaching programming. PLos Comput Biol, 14(4), 1-8, 2018.

[4] Diaz Barriga, F., & Hernandez Rojas, G. Estrategias docentes para un aprendizaje significativo. México: McGraw Hill, 2002.

[5] Brassard, G., & Bratley, P. Fundamentos de Algoritmia. New York (USA): Prentice Hall, 1997.

[6] Basogain Olabe, X., Olabe Basogain , M., & Olabe Basogain, J. Pensamiento Computacional a través de la programación: paradigma de aprendizaje. Revista de Educación a Distancia, 46(6), 2015, 2-35.

[7] Trejos Buriticá, O. Metodología para aprender programación funcional en Ingeniería de Sistemas aplicando teoría de aprendizaje por descubrimiento. Revista Educación en Ingeniería - ACOFI, 12(23), 69-75, 2017.

[8] Trejos Buriticá, O. Tesis Doctoral "Aprendizaje en Ingeniería: Un problema de Comunicación". Pereira (Risaralda): Doctorado en Ciencias de la Educación RudeColombia - CADE UTP, 2012.

[9] Trejos Buriticá, O. Programación Imperativa con Lenguaje C. Bogotá: ECOE Ediciones, 2017.

[10] Schildt, H. Java: a beginners guide. New York: Editorial McGraw Hill, 2017.

[11] Gomez Alvarez, M., Sanchez Dams, R., & Barón Salazar, A. A Representation Proposal of Practices for Teaching and Learning Software Engineering Using a Semat Kernel Extension. Revista Ingenierías, 17(32), 129-154, 2018.

[12] Muñoz Guerrero, L. Tesis Doctoral Modelo de socialización del conocimiento profesional aprovechando NTICs, redes sociales y sus servicios asociados y desarrollo de competencias blandas con grupos interdisciplinarios en Ing de Sistemas. Pereira (Risaralda): Doctorado en Ciencias de la Educación RudeColombia, 2019.

[13] Lindsay, D. Scientific Writing Thinking in Words. Sidney, Australia: CSIRO Publishing, 2011.

[14] Burt, B. Beginning programming with Java for Dummies. New York (USA): For Dummies Editorial, 2017.

[15] Gonzalez Hernandez, W. La enseñanza de la informática y de la matemática: ¿semejantes o diferentes? Revista Educación en Ingeniería - ACOFI, 13(26), 20-26, 2018.

[16] Deitel, P., & Deitel, H. Java - Cómo programar. New York (USA): Pearson Education, 2017.

[17] Cárdenas, D., & Martha Idalia Esparza. El pensamiento lógico computacional. México: Editorial Digital del Tecnológico de Monterrey, 2015.

[18] Ausubel, D. The Acquisition and Retention of Knowledge. Washington - USA: Springer, 2012.

Omar Iván Trejos Buritica. Ingeniero de Sistemas. Especialista en Instrumentación Física. MSc en Comunicación Educativa. PhD en Ciencias de la Educación. Docente de planta categoría Titular, Programa Ingeniería de Sistemas y Computación, Facultad de Ingenierías, Universidad Tecnológica de Pereira. Autor de varios libros de programación y de una buena cantidad de artículos de investigación científica educativa en el área de la programación de computadores sobre aproximación a la optimización de procesos de enseñanza y aprendizaje dentro del contexto de la formación de ingenieros con perfil tecnológico.
Orcid: https://orcid.org/0000-0002-3751-6014.

Luis Eduardo Muñoz Guerrero. Ingeniero de Sistemas. MSc en Ingeniería de Sistemas. Docente Titular de Planta Universidad Tecnológica de Pereira, con 15 años de experiencia en el campo de la formación universitaria. Autor de libros académicos y de investigación. Ha publicado artículos en revistas especializadas nacionales e internacionales. Su área de Investigación se centra en los procesos de enseñanza y aprendizaje de la Programación.
Orcid: http://orcid.org/0000-0002-9414-6187.