Encuentro Internacional de Educación en Ingeniería ACOFI
GESTIÓN, CALIDAD Y DESARROLLO EN LAS FACULTADES DE INGENIERÍA
Cartagena de Indias, Colombia
18 al 21 de septiembre de 2018

ESTRATEGIAS PARA LA ENSEÑANZA DE LÓGICA DE PROGRAMACIÓN EN INGENIERÍA

Gabriel Elías Chanchí Golondrino,
Pedro Harvey Álvarez Sánchez

Institución Universitaria Colegio Mayor del Cauca
Popayán, Colombia

Wilmar Yesid Campo Muñoz

Universidad del Quindío
Armenia, Quindío

Resumen

Una de las problemáticas asociadas al aprendizaje dentro de los cursos de programación, es la apropiación de los diferentes elementos de la lógica de programación debido a la dificultad asociada a la comprensión de conceptos tales como: condicionales, operadores de comparación y relación, ciclos, ciclos anidados, arreglos, matrices, entre otros. Otro de los factores asociados a la anterior problemática, es el uso de términos en inglés dentro de la sintaxis de los lenguajes de programación, lo cual dificulta la asimilación de los conceptos de programación. De igual modo, en lo que respecta a la aplicación práctica de estas temáticas, existen entornos de desarrollo que si bien agilizan la escritura de los programas, hacen transparente al estudiante la comprensión completa del código y la retención de ciertos conceptos que son fundamentales en la enseñanza de la lógica de programación. El anterior problema, se hace más complejo si el lenguaje de programación escogido para la enseñanza, no tiene una curva de aprendizaje de rápido crecimiento, como es el caso de Java. En el presente artículo, se describen un conjunto de estrategias que se han venido utilizando en los cursos introductorios de programación dentro de la Facultad de Ingeniería de la Institución Universitaria Colegio Mayor del Cauca, con el fin de mejorar la apropiación de los conceptos de lógica de programación. Dentro de las estrategias presentadas se encuentran: el uso de herramientas para el diseño y ejecución de diagramas de flujo, la inclusión del lenguaje LPP como medio para apropiar la comprensión de algoritmos en el lenguaje español y como preparación para el uso de un lenguaje de programación convencional, el uso de entornos de desarrollo libres y no convencionales para la enseñanza del lenguaje Java (Dr Java y JGrasp), los cuales al tener funcionalidades limitadas posibilitan una mejor retención de la sintaxis y la estructura de los programas en el lenguaje.

Palabras clave: enseñanza; lenguaje de programación; lógica de programación; programación

Abstract

One of the problems associated with learning in the programming courses is the appropriation of the different elements of the programming logic due to the difficulty associated with the understanding of concepts such as: conditionals, comparison and relationship operators, cycles, nested cycles, arrays, matrices, among others. Another factor associated with the previous problem is the use of terms in english in the syntax of programming languages, which makes it difficult to assimilate the programming concepts. Similarly, with regard to the practical application of these issues, there are development environments that, while streamlining the writing of the programs, make transparent the complete understanding of the code and the retention of certain fundamental concepts in the teaching of logic programming. The previous problem becomes more complex if the programming language chosen for teaching does not have a fast-growing learning curve, such as Java. In this article, we describe a set of strategies that have been used in the introductory courses of programming of the Faculty of Engineering of the University Institution Colegio Mayor de Cauca, in order to improve the appropriation of logical concepts of programming. Among the strategies presented are: the use of tools for the design and execution of flowcharts, the inclusion of the LPP language as a means to appropriate the understanding of algorithms in spanish language and as preparation for the use of a programming language conventional, the use of free and unconventional development environments for the teaching of the Java language (Dr Java and JGrasp), which have limited functionalities and allow a better retention of the syntax and the structure of the programs in the language.

Keywords*: programming; programming language; programming logic; teaching*

1. Introducción

Entre las problemáticas asociadas al aprendizaje dentro de los cursos de programación, se destaca la apropiación de los diferentes elementos de la lógica de programación debido a la dificultad para comprender conceptos computacionales como: condicionales, operadores de comparación y relación, ciclos, ciclos anidados, arreglos, matrices, entre otros. De igual modo, otro de los elementos asociados a la problemática anterior, es el uso de términos en inglés dentro de la sintaxis de los lenguajes de programación, lo cual dificulta la asimilación de los conceptos de programación. Por otra parte, en lo que respecta a la aplicación práctica de estas temáticas, existen entornos de desarrollo que, si bien agilizan la escritura de los programas, hacen transparente al estudiante la comprensión completa del código y la retención de ciertos conceptos que son fundamentales en la enseñanza de la lógica de programación (Fuentes, et al., 2017). El anterior problema, se hace más complejo si el lenguaje de programación escogido para la enseñanza, no tiene una curva de aprendizaje de rápido crecimiento, como es el caso del lenguaje Java.

En este artículo se presenta como aporte el diseño y aplicación de un conjunto de estrategias para la enseñanza de lógica de programación en ingeniería, las cuales se basan en la metodología de aprender haciendo (Rodriguez, et. al., 2014). Así, estas estrategias pretenden mejorar la apropiación de los conceptos de lógica de programación y han surgido a partir de la labor docente desarrollada dentro de los cursos de Fundamentos de Programación y Laboratorio de Programación de los programas de Ingeniería Informática y Tecnología en Desarrollo de Software de la Institución Universitaria Colegio Mayor del Cauca. Dentro de las estrategias planteadas en este artículo se encuentran: el uso de herramientas para el diseño y ejecución de diagramas de flujo; la vinculación del lenguaje LPP como medio para apropiar la comprensión de algoritmos en el lenguaje español y como preparación para el uso del lenguaje de programación Java; el uso de entornos de desarrollo libres y no convencionales para la enseñanza del lenguaje Java (Dr Java y JGrasp), los cuales al tener funcionalidades limitadas posibilitan una mejor retención de la sintaxis y la estructura de los programas en el lenguaje; la construcción de juegos sencillos dentro del curso como medio para apropiar de manera divertida temáticas complejas dentro del curso. De este modo estas estrategias buscan hacer más prácticas ciertas temáticas que se desarrollan dentro de los cursos de programación, de acuerdo a la metodología de aprender haciendo (Rodríguez, et. al., 2014).

Las estrategias planteadas fueron validadas mediante la ejecución de un instrumento dentro de los cursos de Fundamentos de Programación y Laboratorio de Programación del segundo semestre de 2017 y el primer semestre de 2018. Finalmente, las estrategias planteadas en este artículo pretenden servir de referencia para los docentes del área, en cuanto al uso de métodos alternativos para la apropiación de la lógica de programación. El resto del artículo está organizado de la siguiente manera: en la sección 2 se presentan un conjunto de tecnologías que se tuvieron en cuenta para el desarrollo de las estrategias planteadas en este artículo. En la sección 3 se describen las estrategias planteadas en el presente artículo. En la sección 4 se presenta la validación de algunas de las estrategias propuestas. Finalmente, en la sección 5 se presentan las conclusiones derivadas de la presente investigación.

2. Marco conceptual

A continuación, se presentan un conjunto de tecnologías que se tuvieron en cuenta para el desarrollo de las estrategias planteadas en este artículo. Dentro de estas tecnologías se destacan: Dr Java, JGrasp y LPP.

2.1. Dr Java

Dr Java es un entorno liviano de desarrollo para la escritura de aplicaciones en el lenguaje de programación Java. Este entorno ha sido diseñado para estudiantes, proporcionando una interfaz intuitiva que permite evaluar de manera interactiva código escrito en el lenguaje Java. Así mismo Dr Java incluye funcionalidades más complejas para programadores avanzados. Dr Java es una herramienta libre con licencia tipo BSD y ha sido construido por parte del grupo JavaPTL de la Universidad de Rice (Allen et al., 2001).

2.2. JGrasp

JGrasp es un entorno liviano de desarrollo, creado específicamente para posibilitar la creación de programas enfocándose en la sencillez y la visualización sencilla del código, para lo cual permite organizar el flujo del programa. JGrasp ha sido implementado en el lenguaje Java y puede ser ejecutado en todas las plataformas que cuenten con una máquina virtual de Java (con versión superior a la 1.5). JGrasp provee adicionalmente la capacidad de generar diagramas tipo UML y cuenta con un mecanismo para identificar estructuras de datos tradicionales. JGrasp es compatible con diferentes lenguajes de programación tales como: Java, C, C++, Objetive C, Ada, Python, entre otros (Cross et al. 2004).

2.3. LPP

LPP es un lenguaje de programación diseñado para principiantes, el cual fue creado con la idea de facilitar el proceso de enseñanza-aprendizaje de un lenguaje de programación en el idioma español, por lo cual contiene la mayoría de instrucciones que tienen los lenguajes de programación. LPP es ideal para abordar la temática de pseudcódigo de los cursos de programación, dado que no solo permite evaluar la sintaxis, sino ejecutar los programas escritos en pseudocódigo (Castillo, s.f.).

2.4. FreeDFD

FreeDFD es un editor e intérprete de diagramas de flujo para el sistema operativo Windows y Linux (usando Wine), el cual posibilita el diseño y ejecución de algoritmos sencillos y complejos de programación. Los diagramas de flujo creados mediante esta herramienta pueden ser ejecutados y depurados desde la propia interfaz de la aplicación. De este modo es posible mediante esta herramienta agregar gráficamente: asignaciones, decisiones, bucles o salidas (Bueno et al. 2008).

3. Estrategias para la enseñanza de lógica de programación en ingeniería

A continuación, se presentan un conjunto de estrategias que se diseñaron para la enseñanza de la lógica de la programación en ingeniería. Estas estrategias se presentan en la tabla 1.

ESTRATEGIAS PARA LA ENSEÑANZA DE LÓGICA DE PROGRAMACIÓN EN INGENIERÍA		
1	Diagrama de flujo:	Aplicación
	Como para un ingeniero civil o arquitecto es necesario tener el plano en la mano y poder ver la estructura en forma gráfica de cómo puede quedar su obra, también para un desarrollador es representativo ver en un diagrama la secuencia y funcionamiento de una solución informática.	A través de los cursos fundamentos de programación y laboratorio de programación se hizo uso del diagrama de flujo en ejercicios básicos, con el objetivo de presentar al estudiante una forma más representativa de una solución algorítmica, ayudada bajo la aplicación FreeDFD. Esta herramienta permite la generación de diagramas de flujo y la ejecución algorítmica del mismo. Esta estrategia facilita la comprensión de los algoritmos trabajados en el curso por medio de diagramas o dibujos

		significativos que representan un orden o secuencia para solucionar un problema.
2	Pseudocódigo	Aplicación
	El principal objetivo del uso del pseudocódigo es el de dar la solución a un algoritmo de la forma más apropiada posible para un principiante, aplicándolo en el lenguaje propio en este caso el lenguaje español para una mejor comprensión del código.	El uso del pseudocódigo permite la comprensión y escritura de algoritmos de mejor forma, dado que dichos algoritmos son escritos en español. En este orden de ideas se hizo uso del lenguaje LPP o lenguaje para principiantes el cual tiene asociado un compilador que permite la compilación y ejecución de los algoritmos escritos en pseudocódigo. Dentro de los cursos de fundamentos de programación y laboratorio de programación se abordó la sintaxis del lenguaje LPP durante el primer corte y parte del segundo, considerando que mediante la aplicación de esta estrategia que sería más fácil abordar la enseñanza del lenguaje Java en el segundo y tercer corte. Lo anterior está fundamentado en que una vez aprendida la estructura de los algoritmos básicos en LPP, va a resultar más fácil adaptar dichos algoritmos en el lenguaje Java.
3	Prueba de escritorio	Aplicación
	Es un instrumento muy útil para entender línea a línea el comportamiento de un determinado algoritmo, además para verificar si el código funciona correctamente.	El modo de aplicación de la prueba de escritorio en las asignaturas del primer semestre, no es más que efectuar un proceso de simulación sobre el algoritmo desarrollado, consistente en evaluar línea a línea el funcionamiento del mismo. En una prueba de escritorio se suele usar una tabla donde los encabezados son las variables usadas en el algoritmo, después en las siguientes filas se van colocando los valores de cada variable, paso a paso intercambiando valores según el flujo indicado por el algoritmo, hasta llegar al final. Esta prueba se realiza con una ejecución 'a mano' del algoritmo. Esta estrategia se usó en los cursos de fundamentos de programación y laboratorio de programación con el fin de posibilitar en los estudiantes la comprensión del funcionamiento de los algoritmos en el lenguaje LPP y el lenguaje JAVA.
4	Entornos no convencionales de programación	Aplicación
	El objetivo estos entornos es agilizar la escritura, compilación y ejecución de un programa en un lenguaje de programación. En este caso se hizo uso de entornos diferentes a Eclipse y Netbeans.	Si bien el uso de entornos convencionales de programación Java como Eclipse y Netbeans permiten agilizar el desarrollo de aplicaciones, en la práctica docente se ha identificado que el hecho de automatizar la escritura impide la apropiación de ciertos elementos del lenguaje de programación. En este sentido esta estrategia persigue el uso de herramientas que tienen pocas ayudas en la codificación de un algoritmo, posibilitando de este modo la identificación de elementos fundamentales del lenguaje, pero permitiendo las facilidades de compilación y ejecución de un programa. Esta estrategia se usó en los cursos de fundamentos de programación y laboratorio de programación mediante el uso de los entornos libres de programación Dr Java y

		JGrasp, los cuales tienen pocas ayudas en la escritura del código, pero permiten la compilación y la ejecución de un programa. De este modo se busca que los estudiantes puedan ir mecanizando la estructura de los programas y la sintaxis del lenguaje, el cual al no estar en el idioma nativo se ha evidenciado que genera ciertas dificultades.
5	Desarrollo de juegos	Aplicación
	Es una técnica de aprendizaje que permite apropiar conceptos de programación por medio del desarrollo de juegos, la cual es una temática que genera mayor aceptación en los estudiantes. De este modo mediante el desarrollo de aplicaciones entretenidas se buscan afianzar las destrezas en la apropiación de la sintaxis y lógica.	Una de las temáticas que más llama la atención en los estudiantes de programación es el desarrollo de videojuegos. Así esta estrategia consiste en aplicar los diferentes elementos de la sintaxis del lenguaje Java, en el desarrollo de videojuego sencillos. Lo anterior con el fin de aprovechar la motivación en la creación de los videojuegos para apropiar los conceptos de los cursos de programación. Esta estrategia se usó en los cursos de fundamentos de programación y laboratorio de programación mediante la implementación de videojuegos sencillos en los cuales se involucró el uso de temáticas complejas para los estudiantes tales como condicionales, ciclos, vectores unidimensionales, bidimensionales y funciones. El uso de esta estrategia permite abordar de manera más divertida las diferentes temáticas del curso.

Tabla 1 – Estrategias propuestas.

4. Validación de la estrategia propuesta

En esta sección se presentan los resultados de aplicar un instrumento para verificar el cumplimiento de algunas de las estrategias planteadas en la sección 3 dentro de los cursos de Fundamentos de Programación y Laboratorio de Programación de la Institución Universitaria Colegio Mayor del Cauca. Así en primera instancia se presenta el diseño de los instrumentos utilizados y posteriormente se presenta el análisis de la aplicación de dichos instrumentos.

4.1 Instrumento de evaluación

En la tabla 2, se presenta el instrumento aplicado a los estudiantes de Fundamentos de Programación y Laboratorio de Programación, con respecto a la percepción sobre las herramientas de desarrollo usadas durante el curso.

Encuesta de percepción acerca de las herramientas básicas de programación	
Pregunta 1	¿Consideras que las herramientas Dr Java/JGrasp han contribuido a que entiendas de mejor forma la lógica de los programas escritos?
Pregunta 2	¿Crees que las herramientas Dr Java/JGrasp te han permitido mejorar la retención de la sintaxis del lenguaje Java?

Pregunta 3	¿Consideras que las herramientas Dr Java/JGrasp te han ayudado a afianzar tus habilidades para la detección de errores en el código?
Pregunta 4	¿Qué fue lo que más te agradó al usar las herramientas de programación Dr. Java/JGrasp?
Pregunta 5	¿Qué es lo que menos te agradó al usar las herramientas de programación Dr Java/JGrasp?
Pregunta 6	¿Cuál de las herramientas de programación usadas en el curso le pareció más adecuada en el proceso de aprendizaje?

Tabla 2 – Instrumento de evaluación.

4.2 Análisis de los resultados

En la figura 1 se muestran los resultados de aplicar la pregunta 1 de la tabla 2, relacionada con la contribución de las herramientas Dr Java y JGrasp a la mejora de lógica de programación. En esta figura se aprecia como el 28.3% de los estudiantes encuestados consideran muy satisfactoria la contribución de estas herramientas a la lógica de programación. Por su parte, el 40.6 % de los estudiantes encuestados indican que el aporte de las herramientas a la lógica de programación es satisfactorio, el 25% de los encuestados creen que es neutral, el 3% de ellos escogieron en desacuerdo, del mismo modo el 3% selecciono muy desacuerdo. Como puede verse, cerca del 70% de los estudiantes indican que las herramientas Dr Java y JGrasp son pertinentes en cuanto a mejorar la apropiación de la lógica de programación, esto teniendo en cuenta que estas herramientas ayudan a mecanizar la sintaxis del lenguaje y desarrollar las habilidades en cuanto al análisis de los errores.

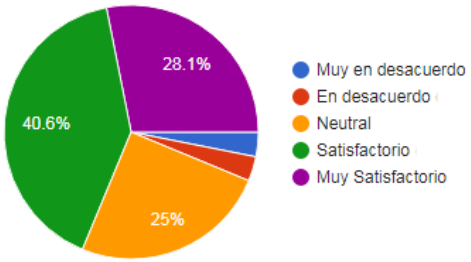

Figura 1 – Resultados obtenidos de la pregunta 1

Al analizar los resultados de la pregunta 2 referente a si las herramientas Dr Java/JGrasp permiten mejorar la retención de la sintaxis del lenguaje Java. Se puede apreciar como el 37.5% en que los estudiantes encuestados consideran muy satisfactoria por que han permitido mejorar la retención de la sintaxis del lenguaje Java. Por su parte, el 34.4 % de los estudiantes encuestados indican que el aporte de las herramientas a mejorar la retención de la sintaxis de java es satisfactorio, el 18.8% de los encuestados creen que es neutral, el 7% de ellos indicaron que están en desacuerdo y 2.3% de los estudiantes indicaron estar en desacuerdo (ver figura xxx). Como puede verse, cerca del 72% de los estudiantes indican que las herramientas Dr

Java/JGrasp han permitido mejorar la retención de la sintaxis del lenguaje Java y además la apropiación de la lógica de programación y desarrollar las habilidades al análisis de los errores del código.

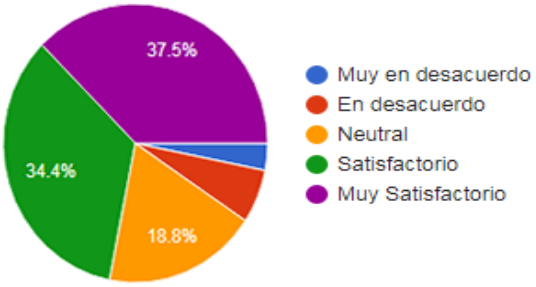

Figura 2 – Resultados obtenidos de la pregunta 2.

En la figura 3 se muestran los resultados obtenidos con la pregunta 3, la cual hace referencia a si las herramientas Dr Java/JGrasp han ayudado a afianzar habilidades para la detección de errores en el código. Al examinar esta figura se puede observar que el 34.4% de los estudiantes encuestados consideran muy satisfactoria la contribución de estas herramientas que permite afianzar habilidades para la detección de errores en el código. Por su parte, el 40.6 % de los estudiantes encuestados indican que el aporte de las herramientas ayudaron a localizar los errores en el código java es satisfactoria, el 18.8% de los encuestados creen que es neutral, el 3.1% de ellos escogieron en desacuerdo, del mismo modo el 3.1% selecciono muy desacuerdo. Al analizar este resultado se puede apreciar que el 75% de los estudiantes indican que las herramientas Dr Java y JGrasp son pertinentes y de gran ayuda en cuanto a la detectar errores de programación y posible solución.

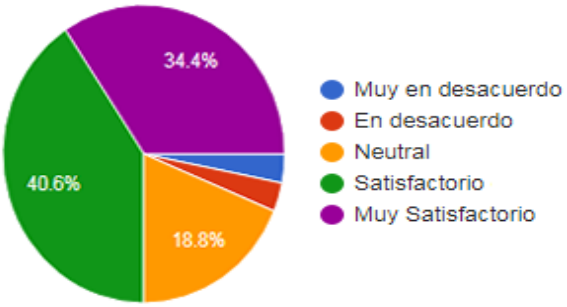

Figura 3 – Resultados obtenidos de la pregunta 3.

Con respecto a la pregunta 4 relacionada con los aspectos que más agradaron a los estudiantes con respecto a los entornos de programación Dr Java y JGrasp, estos destacaron entre otros aspectos que el hecho de no contar ayudas en la escritura del código fue lo que les permitió

apropiar de mejor forma la sintaxis del lenguaje Java. En ese mismo orden de ideas otros de los aspectos que resaltaron los estudiantes de estas herramientas fueron:

- La posibilidad de aprender a partir de los errores cometidos.
- La simplicidad de la interfaz, lo cual permite fácilmente entender cómo desarrollar un programa.
- La practicidad de las herramientas usadas, las cuales se concentran en las operaciones básicas de compilación y ejecución.
- La forma en la que muestra la línea en la que se presentó algún error de codificación.
- El manejo de atajos en el teclado en cuanto a la compilación y ejecución de un programa.

En concordancia con lo anterior, es importante resaltar que las herramientas en mención poseen una interfaz sencilla, centrada en posibilitar tres opciones básicas: codificar, compilar y ejecutar. En este sentido las herramientas Dr Java y JGrasp son mucho menos complejas que entornos de desarrollo más sofisticados como Eclipse y Netbeans, cuya principal ventaja es la posibilidad de autocompletar el código.

Con respecto a la pregunta 5 relacionada con los elementos que menos agradaron a los estudiantes respecto a las herramientas Dr Java y JGrasp, estos manifestaron que en ocasiones las herramientas se quedaban bloqueadas, así mismo no permiten reutilizar código de otras fuentes externas como páginas web o archivos de texto plano. Así mismo algunos de los estudiantes manifestaron que los errores que mostraban estas herramientas estaban en inglés lo cual dificultaba su comprensión.

Finalmente, en lo que respecta a la pregunta 6 relacionada con cuál de las dos herramientas usadas fue de mayor agrado por parte de los estudiantes, en la figura 4 se puede apreciar cómo el 62.5% de los estudiantes encuestados eligió como herramienta más pertinente en el proceso de aprendizaje JGrasp, mientras que el 37.5% de los estudiantes encuestados indicaron que la herramienta programación usada en el curso le pareció más adecuada fue Dr Java (ver figura 4). En este sentido dentro de los cursos de Fundamentos de Programación y Laboratorio de Programación se dio libertad de escoger una de las dos herramientas una vez se presentaron las dos.

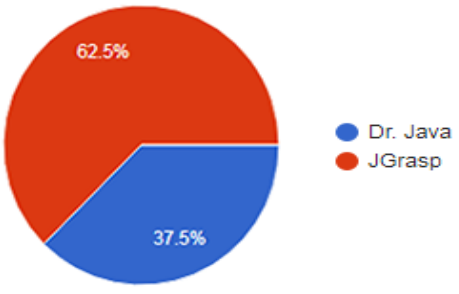

Figura 4 – Resultados obtenidos de la pregunta 6.

5. Conclusiones

En esta sección se presentan un conjunto de conclusiones y trabajos futuros derivados de la presente investigación.

- Las estrategias planteadas en este artículo recogen un conjunto de buenas prácticas adquiridas a partir de la experiencia docente en los cursos de Fundamentos de Programación y Laboratorio de Programación. Así, estas estrategias pueden ser articuladas en las diferentes temáticas de estos cursos.
- El uso de entornos de programación no convencionales como Dr Java y JGrasp permiten a los estudiantes de programación mecanizar de mejor forma la sintaxis del lenguaje de programación Java, en comparación con otros entornos más sofisticados como Eclipse y Netbeans, los cuales permiten autocompletar la escritura de sentencias.
- Herramientas como LPP y FreeDFD permiten apoyar temáticas de interés dentro de los cursos de programación como son: diagramas de flujo y pseudcódigo, las cuales por lo general son abordadas de manera teórica. En este sentido mediante estas herramientas es posible verificar la funcionalidad de los algoritmos escritos por los estudiantes de programación.
- Las estrategias planteadas en este artículo pretenden servir de referencia para los docentes del área, en cuanto al uso de métodos alternativos para la apropiación de la lógica de programación en sus cursos. En ese orden de ideas, estas estrategias demostraron tener un nivel de aceptación alto por parte de los estudiantes del curso de Fundamentos de Programación y Laboratorio de Programación de los programas de Ingeniería Informática y Tecnología en Desarrollo de Software de la Institución Universitaria Colegio Mayor del Cauca.

6. Referencias

- Fuentes, J.I.; Moo, M. (2017). Dificultades para aprender a programar. Revista Educación en Ingeniería. Vol. 12. No 24.
- Rodríguez, A.; Ramírez, L. (2014). Aprender haciendo – Investigar reflexionando: Caso de estudio en paralelo en Colombia y Chile. Revista Academia y Virtualidad. Vol. 7. No 2.
- Allen, E.; Cartwright, R.; Stoler, B. (2012). DrJava: A lightweight pedagogic environment for Java. Presentado en el SIGSCE 2012.
- Cross, J.H.; Hendrix, D.; Umphress, D. (2004). JGRASP: an integrated development environment with visualizations for teaching java in CS1, CS2, and beyond. Memorias de la Conferencia Frontiers in Education.
- Castillo, R. Programación en LPP. Obtenido de: https://mediatecnica.weebly.com/lpp.html
- Bueno, C.; Bueno, J. (2008). Uso de los objetos del programa DFD. Obtenido de: http://jab687.angelfire.com/diagramas_de_flujo.pdf.

Sobre los autores

- **Gabriel Elías Chanchí Golondrino**: Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca. Magister en Ingeniería Telemática de la Universidad del Cauca. Doctor en Ingeniería Telemática de la Universidad del Cauca. Profesor Asistente de la Facultad de Ingeniería de la Institución Universitaria Colegio Mayor del Cauca. gchanchi@unimayor.edu.co
- **Pedro Harvey Álvarez Sánchez**: Ingeniero de Sistemas de la Universidad Incca de Colombia. Magister en Gestión de la Tecnología Educativa. Profesor de la Facultad de Ingeniería de la Institución Universitaria Colegio Mayor del Cauca. alvarezsan@gmail.com
- **Wilmar Yesid Campo Muñoz**: Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca. Magister en Ingeniería Telemática de la Universidad del Cauca. Doctor en Ingeniería Telemática de la Universidad del Cauca. Profesor de la Facultad de Ingeniería de la Universidad del Quindío. wycampo@uniquindio.edu.co

---

Los puntos de vista expresados en este artículo no reflejan necesariamente la opinión de la Asociación Colombiana de Facultades de Ingeniería.

Copyright © 2018 Asociación Colombiana de Facultades de Ingeniería (ACOFI)