Universidad de Cienfuegos sede “Carlos Rafael Rodríguez”
Facultad de Ingeniería
Carrera de Ingeniería Informática

Trabajo de diploma para optar por el título de Ingeniera Informática

Titulo:

MÓDULO DE PROBLEMAS RESUELTOS PARA DESARROLLAR EL PENSAMIENTO COMPUTACIONAL DE ESTUDIANTES DE PRIMER AÑO DE INGENIERÍA INFORMÁTICA

Autora:
Isabel Nissandra Cawanga Cambinda

Tutor(es):

MSc. Yailem Arencibia Rodríguez del Rey
Dr. Raidell Avello Martínez

Cienfuegos, Cuba
Curso 2017 - 2018

Declaración de autoría

Yo Isabel Nissandra Cawanga Cambinda, declaro que soy la única autora de este trabajo de diploma titulado “El Pensamiento Computacional en la Educación Superior Universitaria” y autorizo al Departamento de Informática de la Facultad de Ingeniería en la Universidad de Cienfuegos sede “Carlos Rafael Rodríguez”, para que hagan el uso que estimen pertinente con el trabajo de diploma.

Para que así conste firmo (firmamos) la presente a los ____ días del mes de ___ del ___.

	(Si procede)
_____	_____
Nombre completo del primer autor	Nombre completo del segundo autor
	(Si procede)
_____	_____
Nombre completo del primer tutor	Nombre completo del segundo tutor

Los abajo firmantes certificamos que el presente trabajo ha sido revisado según acuerdo de la dirección de nuestro centro y el mismo cumple los requisitos que debe tener un trabajo de esta envergadura referente a la temática señalada.

----------	----------
Firma Tutor	Firma Tutor
----------	----------
Firma ICT	Firma Vicedecano

Opinión del usuario

El Trabajo de Diploma, titulado <El Pensamiento Computacional en la Educación Superior Universitaria>, fue realizado en nuestra entidad <Universidad de Cienfuegos sede “Carlos Rafael Rodríguez”>. Se considera que, en correspondencia con los objetivos trazados, el trabajo realizado nos satisface:

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a nuestra entidad los beneficios siguientes (cuantificar):

_____
_____
_____
_____
_____

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a <valor> MN y/o <valor> CUC. (Este valor debe ser REAL, no indica lo que se reportará, sino lo que reporta a la entidad. Puede desglosarse por conceptos, tales como: cuánto cuesta un software análogo en el mercado internacional, valor de los materiales que se ahorran por la existencia del software, valor anual del (de los) salario(s) equivalente al tiempo que se ahorra por la existencia del software).

Y para que así conste, se firma la presente a los __ días del mes de ____ del año ____.

_____ _____

Nombre del representante de la entidad Cargo

_____

Firma Cuño

Agradecimientos

Le doy gracias a Dios en primer lugar y por sobre todas las cosas, por ser mi creador y estar presente en todos los momentos de mi vida.

- *A mi gobierno por ofrecerme esta gran oportunidad. Y a Cuba por recibirme con los brazos abiertos.*
- *A mi madre por darme la vida, por su amor incondicional y por ayudarme en todo lo que ha podido a pesar de la distancia.*
- *A mi tía Anabela por hacer posible mí llegada a Cuba y por todo el apoyo moral que me ha dado.*
- *A mi tío Cawanga por hospedarme en su casa, por dedicarme su tiempo y paciencia.*
- *A mi abuela Isabel, mi tocaya por su perseverancia, por su apoyo y sus sabios consejos. Abuela, te quiero.*
- *A mis hermanas por su cariño y confianza.*
- *A mi novio Vladimir por amarme, cuidarme, por ser mi escudo protector, apoyarme y estar conmigo en las buenas y en las malas, y por haberme ayudado a tomar las mejores decisiones para mi vida y para mi carrera. Por su cariño, por su tiempo y paciencia; por su tolerancia y por estar conmigo incondicionalmente. Mi amor, te amo.*
- *A mis primas Tobalda y Cleusa por su gran apoyo a pesar de la distancia.*
- *A todos mis amigos por su tiempo, y por la confianza que depositaron en mí.*
- *A Deodeth por ser la hermana que me gané en Cuba y por estar conmigo siempre, por todos estos momentos juntas y por confiar siempre en mí sin importar las circunstancias.*
- *A Barros por haberme cuidado durante estos cinco años como un padre.*
- *A Lázara Pérez (mami Lachy) por ser como una segunda madre para mí, por cuidarme, por aconsejarme, por apoyarme siempre y porque siempre estuvo allí para mi sin importar el día, la hora, el tiempo o las circunstancias. Porque siempre tuviste que dejar a un lado otras responsabilidades para atenderme, por todo eso, te estaré eternamente agradecida y te querré siempre mamita.*

- *A todos mis compañeros de aula con quienes compartí cinco inolvidables años de la vida.*
- *A mis tutores Yailem Arencibia y Raidell Avello por dedicarme su tiempo, a pesar de sus compromisos laborales y del hogar, y por haber confiado en mí desde un principio y no dudar en ningún momento.*
- *Quiero agradecer de todo corazón a mi familia, amigos, profesores, compañeros, a todas las personas que de una u otra forma me han ayudado en mi formación profesional, en la realización de esta investigación y en ser una mejor persona cada día.*

Dedicatoria

Quisiera dedicarle este trabajo de tesis:

- *A mi novio Vladimir por ser tan especial, y por brindarme todo su apoyo y amor, sobre todo por ser mi mejor amigo.*
- *En especial a mi mamá, por ser mi inspiración, por ser madre y padre, no solo mío que ya es tarea difícil, sino de tres más; ¿nunca te la pusimos fácil verdad? Te quiero con mi vida madre.*
- *A mi abuela por su duro trabajo, y por sus enseñanzas.*

"El mejor aprendizaje no vendrá de encontrar las mejores formas para que el profesor instruya, sino de darle al estudiante las mejores oportunidades para que construya".

'Seymour Papert'

Resumen

La presente investigación aborda el desarrollo del pensamiento computacional de estudiantes de primer año de ingeniería informática en la Universidad de Cienfuegos, constatando que no se considera la importancia del desarrollo del pensamiento computacional y lo que puede aportar en la formación del ingeniero informático. Se propone como objetivo elaborar un módulo de problemas resueltos para el desarrollo del pensamiento computacional de estudiantes de primer año de la carrera en estudio.

Se propone una definición contextualizada de pensamiento computacional como proceso cognitivo ejecutado por humanos para la resolución de problemas diversos haciendo uso de conceptos computacionales que involucra los componentes siguientes: abstracción, análisis y representación de datos, descomposición del problema, pensamiento algorítmico, reconocimiento y generalización de patrones, simulación / evaluación.

El módulo consta de 22 problemas resueltos que abarcan problemas de búsqueda sin heurística, las matemáticas (I y II) y matemática discreta.

Los métodos utilizados fueron: teóricos (histórico-lógico, inductivo-deductivo, analítico-sintético); empíricos (entrevista, observación, análisis de documentos) y para hacer el análisis estadístico, se utilizó el software SPSS. La investigación realizada es un **diseño experimental** con pre-test y post-test, con un grupo de control (9) y un grupo experimental (9) al cual se le aplicó la intervención. Los 9 estudiantes del grupo experimental se seleccionaron siguiendo un muestreo aleatorio simple.

Las pruebas estadísticas realizadas (Signos y Mann-Whitney) para contrastar los resultados iniciales y finales del grupo experimental y del grupo de control, mostraron que existe superioridad en los resultados finales lo cual evidencia la eficacia de las intervenciones realizadas con el “módulo de problemas resueltos”.

Palabras claves: Pensamiento computacional, módulo de problemas resueltos.

Abstract

This research deals with the development of computational thinking of first-year computer engineering students at the University of Cienfuegos, noting that the importance of the development of computational thinking and what it can contribute in the training of the informatics engineer is not considered. The objective is to develop a module of solved problems for the development of computational thinking of first-year students of the study career.

A contextualized definition of computational thinking is proposed as a cognitive process executed by humans for the resolution of diverse problems making use of computational concepts that involve the following components: abstraction, analysis and representation of data, decomposition of the problem, algorithmic thinking, recognition and generalization of patterns, simulation / evaluation.

The module consists of 22 solved problems that cover search problems without heuristics, mathematics (I and II) and discrete mathematics.

The methods used were: theoretical (historical-logical, inductive-deductive, analytic-synthetic); empirical (interview, observation, document analysis) and to make the statistical analysis, the SPSS software was used. The research carried out is an experimental design with pre-test and post-test, with a control group (9) and an experimental group (9) to which the intervention was applied. The 9 students of the experimental group were selected following a simple random sampling.

The statistical tests performed (Signs and Mann-Whitney) to contrast the initial and final results of the experimental group and the control group showed that there is superiority in the final results, which shows the effectiveness of the interventions carried out with the module of solved problems.

Índice

Índice de tablas

Índice de figuras

Introducción

La transformación que ha generado la tecnología en los últimos años ha tenido un impacto muy significativo. No solo en la vida cotidiana del ser humano sino también en el ámbito académico. Esto ha despertado un gran interés en varias áreas del sector educativo sobre la importancia de desarrollar nuevas habilidades en los alumnos [1].

Las tecnologías de la información y las comunicaciones (TIC) avanzan a pasos agigantados, impulsadas por el desarrollo científico técnico que experimentó el mundo en la segunda mitad del siglo pasado, repercutiendo en casi todas las esferas de la sociedad. En la educación, como base fundamental del desarrollo de cualquier sociedad, esta repercusión casi inmediata se vio evidenciada en la evolución gradual de las Metodologías de Enseñanza para insertarse en dicho proceso de cambio [2].

El impulso tecnológico hizo cada vez más necesario el uso de las computadoras como medio educacional, evidenciando mejoras en el proceso de aprendizaje y un cambio radical en los modelos pedagógicos. Se demostró que cuando se posibilita la intervención del alumno en su proceso de enseñanza aprendizaje se obtiene una notoria mejoría en dicho proceso, expuesto de otro modo, al cambiar el rol pasivo por un rol más activo, transforma al alumno en protagonista de su propio proceso de enseñanza aprendizaje [3] citado en Juan Jesús Granados (tesis, 2016), surge así la denominada Educación Centrada en el estudiante que sienta sus bases en otro de los grandes cambios de la época, aunque esta vez en la esfera de la psicología, denominado psicología constructivista, la cual una vez aplicada a la educación crea un nuevo concepto de aprendizaje, que recibe por nombre aprendizaje significativo. El aprendizaje significativo tiene como premisa mejorar el proceso de enseñanza aprendizaje de los educandos, centrándose en que a la hora de enseñar nuevas informaciones a los alumnos, estos las relacionen con información que ya poseen, o sea, el conocimiento actual va a estar condicionado por el conocimiento previo que posea cada uno [2].

Pensar parece algo común: Todos comparamos, clasificamos, ordenamos, extrapolamos, interpretamos, juzgamos, usamos analogías y elaboramos conclusiones (Lizárraga, Baquedano, 2005). Gracias al desarrollo y a la existencia de nuevas herramientas tecnológicas, esta facultad inherente del ser humano está sujeta a fortalecerse y evolucionar

según las exigencias de un mundo cada vez más sistematizado. Es así como los avances de las nuevas tecnologías, son identificables al facilitar las tareas diarias y al incidir en numerosos procesos (matemáticos, de almacenamiento, comunicativos) que hacen que la forma de razonar cambien [4].

En ésta nueva era de las tecnologías y la comunicación, se exige que habilidades propias de diferentes formas de pensamiento (crítico, matemático, algorítmico, entre otros), se combinen y den lugar a una nueva forma de razonar: El pensamiento computacional.

El origen de esta nueva forma de pensamiento se remonta a los años 80, cuando Seymour Papert interesado por la introducción de la tecnología en la educación, trabaja para dotar de sentido pedagógico la introducción de la tecnología en el aula. En esta nueva formulación pedagógica, Papert propone que para construir conocimiento no basta con trabajar exclusivamente desde formulaciones abstractas sino que la sociedad debe proveer a los alumnos con medios y materiales con los que experimentar y crear conocimiento. Además, plantea que este aprendizaje experiencial debe centrarse en la resolución de problemas y ofrece una de las claves, que más tarde asume Wing, cuando afirma "para resolver un problema busca algo similar que ya comprendas" [5], es decir que las personas acudimos a nuestros conocimientos previos a la hora de enfrentarnos a un nuevo conocimiento o desafío.

Wing, sigue sus investigaciones y plantea la necesidad de formar a los futuros profesionales de múltiples áreas que, necesariamente, tendrán que tener estos conocimientos para desarrollarse en sus carreras profesionales, cada vez más mediadas por la tecnología.

En definitiva, tanto Papert como Wing, ambos acuden al mismo planteamiento: la resolución de problemas utilizando habilidades de pensamiento de los informáticos y de las computadoras, para ser capaz de modelar y de utilizar habilidades de abstracción complejas mediante la practica intensiva, utilizando la creatividad humana como valor fundamental, y, en definitiva, para que los profesionales del mañana adquieran la capacidad de resolver nuevos problemas de manera eficaz y eficiente mediante el uso de tecnología aplicada a cualquier campo.

Fue entonces hasta el año 2006, que Jeannette Wing publicó un artículo intitulado "*Computational Thinking*" en la revista *Communications of the ACM* (Association for Computing Machinery) en el que defendía la necesidad de comenzar a poner en práctica las técnicas que utilizan los informáticos, en la planificación y desarrollo de sus procesos, a

áreas científicas alejadas de la informática. Y lo define: *"el pensamiento computacional implica la resolución de problemas, el diseño de sistemas, y la comprensión de la conducta humana, haciendo uso de los conceptos fundamentales de la informática*. Así, la esencia del PC sería pensar como un científico de la computación cuando uno se enfrenta a un problema [6].

Este breve artículo, que iba dirigido a la comunidad científica del área de la informática, tuvo una gran repercusión y dio el salto a otros ámbitos, como el de la educación. En un planteamiento interesante, Wing defiende que debiéramos preocuparnos seriamente de que nuestros niños y jóvenes aprendan a resolver problemas como lo haría una computadora, poniendo en juego múltiples habilidades de abstracción, que pueden ser entrenadas.

Esta primera definición del pensamiento computacional, viene siendo revisada y especificada en intentos sucesivos a lo largo de los últimos años, sin llegar aún a un acuerdo generalizado sobre la misma.

Al hacer una reflexión sobre las aproximaciones dadas por los diferentes autores sobre pensamiento computacional, se encuentra como constante la necesidad de solucionar problemas; sin importar el escenario donde se encuentre el individuo, siempre hay algo que requiere ser solucionado, mejorado o inventado. No es solo en problemas matemáticos donde la solución directa sea un algoritmo, sino soluciones a problemas del mundo real a los cuales se puede dar respuesta con el desarrollo de una aplicación (software). Es así como en el pensamiento computacional interviene el pensamiento lógico, sistémico y algorítmico [7].

En conclusión, el Pensamiento Computacional es un proceso cognitivo ejecutado por humanos para la resolución de problemas diversos haciendo uso de conceptos computacionales que involucra, de forma relacionada, los componentes siguientes:

- Abstracción
- Análisis y representación de datos
- Descomposición del problema del problema
- Pensamiento algorítmico
- Reconocimiento y generalización de patrones
- Simulación / Evaluación

En la actualidad, existen varios países que ya han integrado el PC en su sistema educativo, como también existen ya algunos modelos de evaluación del PC e instrumentos de medida del mismo direccionados a la educación universitaria específicamente en el primer año de la universidad en carreras de ingeniería informática y computación.

En Cuba, aun no hay documento oficial que mencione el “Pensamiento Computacional”, sin embargo, podemos encontrar algunas conexiones de conceptos relacionados en el currículo de asignaturas determinadas.

Aumentar la calidad de la educación universitaria es un reto internacional, regional y también de Cuba en aras de lograr la formación de un profesional más competente en la sociedad actual. Por esta razón, es necesario desarrollar el pensamiento computacional en los altos estudios, particularmente para despertar el interés de los jóvenes por estudiar carreras de ingeniería informática.

En la carrera de Ingeniería Informática en la Universidad de Cienfuegos, se ha podido constatar que los estudiantes tienen dificultades para organizar y analizar datos de una manera lógica en asignaturas afines y al final de cada ciclo lectivo, se observan dificultades en identificar, analizar e implementar posibles soluciones a problemas diversos, con el objetivo de conseguir la combinación más eficaz de pasos y recursos. Como consecuencia a esto, se evidencian resultados insatisfactorios que conllevan a que la carrera tenga varios estudiantes con arrastres en diferentes asignaturas de la especialidad y bajas académicas.

Actualmente no se considera la importancia del desarrollo del pensamiento computacional y lo mucho que este puede aportar en la formación del ingeniero informático y los componentes del pensamiento computacional antes descritos, se trabajan de forma aislada en las asignaturas de las diferentes disciplinas de la carrera; y tampoco existe una cohesión intencional que defina un lenguaje o un enfoque común interdisciplinario del pensamiento computacional como metodología de resolución de problemas de manera que éstos puedan ser solucionados computacionalmente. Los instrumentos de medida del pensamiento computacional no se adecuan al contexto de la educación superior cubana y además existen carencias en los fundamentos didácticos y teórico-metodológicos para desarrollar el pensamiento computacional a través de las asignaturas de la carrera.

Teniendo en cuenta todo lo planteado anteriormente se identifica como **problema a resolver**: La necesidad de que los estudiantes de primer año de la carrera de ingeniería informática desarrollen el pensamiento computacional a partir de problemas resueltos.

Se presenta como **objeto de estudio**: El proceso de enseñanza – aprendizaje de la carrera de ingeniería informática; y como **campo de acción**: El desarrollo del pensamiento computacional en el proceso de enseñanza – aprendizaje en la carrera de ingeniería informática.

Se plantea como **idea a defender** que: La elaboración y aplicación de un *"módulo de problemas resueltos"* con la utilización de los componentes del pensamiento computacional, favorecerá el aprendizaje de los estudiantes de primer año de ingeniería informática.

La labor científico investigativa está encaminada a dar cumplimiento al siguiente **objetivo general**:

- Elaborar un "módulo de problemas resueltos" para el desarrollo del pensamiento computacional de estudiantes de primer año de ingeniería informática.

Partiendo del objetivo general, se plantean los siguientes objetivos específicos:

- Analizar los fundamentos teóricos metodológicos del uso de los problemas resueltos y el desarrollo del pensamiento computacional de estudiantes de ingeniería informática.
- Diagnosticar el desarrollo del pensamiento computacional de estudiantes de primer año de ingeniería informática mediante una prueba pre test.
- Elaborar un *"módulo de problemas resueltos"* para el desarrollo del pensamiento computacional de estudiantes de primer año de ingeniería informática.
- Aplicar el *"módulo de problemas resueltos"* a los estudiantes de primer año de ingeniería informática.
- Validar los resultados de la aplicación en la práctica a partir de una prueba pre y post-test.

La justificación de esta investigación está dada por la necesidad de tener una metodología que guíe la elaboración del *"módulo de problemas resueltos"* para desarrollar el pensamiento computacional con una representación de estos en el lenguaje de programación Scratch.

Este documento de investigación se estructura de la siguiente forma: Introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos.

Capítulo I: Fundamentación Teórica. En este capítulo se exponen conceptos asociados al dominio del problema y la metodología para su desarrollo.

Capítulo II: Desarrollo de la propuesta. En este capítulo se define y contextualiza la propuesta metodológica a seguir para la elaboración del *"módulo de problemas resueltos"* para desarrollar el PC.

Capítulo III: Validación de la propuesta. En este capítulo se aplica el *"módulo de problemas resueltos"* y se validan los resultados de la aplicación de este, en la práctica a partir de una prueba pre y post-test.

Capítulo I: Fundamentación teórica

1.1 Introducción

En este capítulo se exponen los conceptos asociados al dominio del problema de investigación que consiste en un *"módulo de problemas resueltos"* para desarrollar el pensamiento computacional de estudiantes de primer año de ingeniería informática. Se explican las dificultades que se presentan actualmente en relación con el tema y que dan paso a la presente investigación.

Se describen otras soluciones existentes en el mundo; así como la propuesta realizada, llegando a conclusiones sobre los aspectos en los que esta se diferencia de las otras y las supera.

1.2 Explicación sobre el contexto del dominio del problema

En la actualidad existe una tendencia de las políticas educativas hacia la incorporación de la programación y del pensamiento computacional en los diseños curriculares oficiales. Lo anterior tiene argumentos sólidos basados en demandas y necesidades de los ciudadanos de sociedades digitales que precisan conocer cómo funcionan los dispositivos que los comunican con el mundo y les permite acceder a la información como en ningún otro período de la historia. Para favorecer su desarrollo resulta imperiosa la inclusión de los fundamentos computacionales de la Informática en todos los niveles educativos, incluyendo a la Universidad.

La carrera Ingeniería Informática en Cuba transita hacia el plan de estudios E. El Plan de estudios E plantea entre sus bases los aspectos siguientes:

- Las tendencias a considerar una formación de pregrado con una mayor esencialidad y menor duración y las necesidades de formación de técnicos de nivel universitario.
- La necesidad de reconsiderar un enfoque de formación del profesional hacia la solución de problemas más frecuentes del entorno social y productivo con una necesidad de ajuste al desempeño en el eslabón de base de la profesión.
- Las tendencias internacionales en el desarrollo informático y su relación con otras tecnologías emergentes y en la enseñanza universitaria cubana.

En resumen, el plan de estudios E presenta la importancia y necesidad de que los ingenieros informáticos sean capaces de resolver diferentes problemas profesionales a través de soluciones informáticas viables y eficientes. En contraste con lo anterior, estudiantes de primer año de la carrera Ingeniería Informática de la Universidad de Cienfuegos, muestran los problemas siguientes: dificultad para organizar y analizar datos de una manera lógica; se observan dificultades en identificar, analizar e implementar posibles soluciones a problemas diversos, con el objetivo de conseguir la combinación más eficaz de pasos y recursos. Como consecuencia a esto, se evidencian resultados inadecuados que conllevan a que la carrera tenga varios estudiantes con arrastres en diferentes asignaturas de la especialidad y bajas académicas.

Actualmente no son considerados los aportes que el desarrollo del pensamiento computacional puede traer en la formación integral del ingeniero informático. Los componentes del pensamiento computacional antes descritos no se trabajan de forma cohesionada e intencional en las diferentes disciplinas, por lo que no existe un lenguaje o un enfoque común interdisciplinario del pensamiento computacional como metodología de resolución de problemas.

El pensamiento computacional, plantea una serie de componentes, que en conjunto, ayudan a los estudiantes en el desarrollo de habilidades mentales y prácticas que favorecen su aprendizaje.

1.3 Pensamiento Computacional (PC)

La historia del Pensamiento Computacional, como es el caso de muchos desarrollos fundamentales en la ciencia, refleja la convergencia de múltiples ideas, a menudo de diferentes áreas de estudio (ciencias cognitivas, lingüística, psicología, informática), que después de ser desarrolladas aisladamente, encontraron un efecto sinérgico cuando se aplicaron al área de la educación, y en particular a los procesos que implican sistemas complejos, así como a los lenguajes generativos para la creación de métodos novedosos.

No podemos hablar del PC sin antes mencionar el pensamiento ya que este último, nos ayuda de cierta forma a comprender los procesos mentales involucrados en la resolución de problemas, siendo este el centro de atenciones del PC.

1.3.1 Pensamiento

Del latín “pensare”, se entiende como la facultad o el poder de pensar, también puede definirse como la acción y el efecto de pensar.

Existe tal cantidad de aspectos relacionados con el pensamiento, que dar una definición resulta difícil.

Según la *definición teórica:*

“el ***pensamiento*** es aquello que se trae a la realidad por medio de la actividad intelectual”. El pensamiento puede abarcar un ***conjunto de operaciones de la razón***, como lo son el análisis, la síntesis, la comparación, la generalización y la abstracción.

El pensamiento está relacionado con la inteligencia. La inteligencia es la capacidad de conocer, comprender, entender, crear símbolos abstractos y resolver situaciones nuevas [8].

El pensamiento es también el conjunto de operaciones inteligentes que permiten crear conceptos; siendo estos, creaciones mentales, representaciones simbólicas de objetos reales o ideales, que no siempre son idénticos a los objetos [8].

Tipos de pensamiento

Se pueden encontrar diferentes tipos de pensamiento:

Deductivo: es aquel que va de lo general a lo particular, para encontrar la razón de las cosas.

Inductivo: es aquel que va de lo particular a lo general.

Analítico: parte la realidad en porciones para poder evaluarla a través de mecanismos lógicos

Investigativo: suele utilizar preguntas para lograr llegar al pensamiento o la resolución de problemas.

Creativo: es la base del arte, ya que se basa en la libertad de modificar o crear algo.

Sistémico: propone un sistema orgánico que interrelaciona los conceptos de manera compleja.

Crítico: este examina, evalúa y se pregunta el porqué de las cosas.

Interrogativo: nos permite cuestionar distintos aspectos de nuestro interés, articulando así el proceso de aprendizaje.

Teniendo en cuenta que esta investigación pretende integrar el componente científico de la mente humana y el procesamiento de la información con la solución de problemas para desarrollar el PC, se hace necesaria la inclusión del factor psico-cognitivo para una mejor comprensión del enfoque de esta investigación.

Según Vega (2005) el pensamiento es *"una actividad mental no rutinaria que requiere esfuerzo. Ocurre siempre que nos enfrentamos a una situación o tarea en la que nos sentimos a hallar una meta u objetivo, aunque existe incertidumbre sobre el modo de hacerlo. El pensamiento implica una actividad global del sistema cognitivo, con intervención de los mecanismos de memoria, la atención, las representaciones o los procesos de comprensión; pero no es reducible a estos"* [9].

Aunque la psicología cognitiva ha basado fundamentalmente sus investigaciones en tres aspectos:

- El razonamiento deductivo
- El razonamiento inductivo
- La solución de problemas

*El **razonamiento deductivo**:* parte de categorías generales para hacer afirmaciones sobre casos particulares. Es una forma de razonamiento donde se infiere una conclusión a partir de una o varias premisas.

*El **pensamiento inductivo**:* es aquel proceso en el que se razona partiendo de lo particular para llegar a lo general, justo lo contrario que con la deducción. La base de la inducción es la suposición de que si algo es cierto en algunas ocasiones, también lo será en situaciones similares aunque no se hayan observado.

Otro importante aspecto en el que se han basado las investigaciones de la psicología cognitiva es la solución de problemas.

Podríamos decir que un ***problema*** es un obstáculo que se interpone de una u otra forma ante nosotros, impidiéndonos ver lo que hay detrás. Lo cierto es que no hay consenso entre los psicólogos sobre lo que es exactamente un problema, y por tanto difícilmente puede haberlo en lo que supone una conducta de solución de problemas.

Algunos autores han intentado precisar estos términos. Gagné, por ejemplo, definió la ***solución de problemas*** como “una conducta ejercida en situaciones en las que un sujeto debe conseguir una meta, haciendo uso de un principio o regla conceptual”. En términos restringidos, se entiende por solución de problemas, cualquier tarea que exija procesos de razonamiento relativamente complejos y no una mera actividad asociativa.

Se considera que habitualmente cualquier persona pasa por tres fases a la hora de solucionar un problema y se las denomina: preparación, producción y enjuiciamiento.

En la fase de preparación es cuando se hace un análisis e interpretación de los datos que tenemos. Muchas veces si el problema es muy complejo se subdivide en problemas más elementales para facilitar la tarea.

En la fase de producción intervienen distintos aspectos entre los que hay que destacar la memoria, que se utiliza para recuperar todos los recursos que estén a nuestro alcance y que nos sirvan para llegar a una solución eventual.

En la última fase de enjuiciamiento, lo que se hace es evaluar la solución generada anteriormente, contrastándola con nuestra experiencia, para finalmente darla como buena o no.

1.3.2 Antecedentes del PC

Cada época demanda para las personas diferentes exigencias y el siglo XXI no ha sido la excepción. Basta con los cambios Surgidos a partir de la globalización y la tecnología como para darnos cuenta de la transformación ocurrida en la vida no sólo laboral sino personal y familiar. Lo anterior, a nivel profesional, conlleva el desarrollo de competencias, entre las cuales, estudios a nivel nacional e internacional, destacan como principales a desarrollar: creatividad e innovación, pensamiento crítico, auto aprendizaje, comunicación, colaboración y trabajo en equipo, adaptabilidad, proactividad, orientación a resultados, liderazgo,

responsabilidad y respecto así como habilidades socioculturales y de manejo de información y tecnología. Bien valdría la pena reflexionar sobre la presencia de ellas en nuestro modo de ser, identificando cuáles son las fortalezas que nos distinguen para responder como ciudadanos de esta era y cuáles son las áreas de oportunidad a trabajar [4].

Según Cuny, Snyder, Wing (2010), el PC es un conjunto de habilidades resultado de la combinación de diferentes formas de pensamiento de entre las cuales se encuentran: el Pensamiento Crítico, Matemático, y el pensamiento Algorítmico [10].

Para Lugo (2016), los problemas de este siglo necesitan soluciones innovadoras, creativas y que generen un reto para los estudiantes. Para alcanzar soluciones a estos problemas, el estudiante debe contar con múltiples habilidades para analizar y comprender el problema.

En el intento por identificar estas habilidades mínimas, se encuentra necesario considerar el complejo proceso de pensamiento como un todo. Es así como a partir de la revisión sobre los fundamentos del pensamiento computacional y tal como lo presenta el ISTE (2012), se encuentra que esta forma de pensamiento es sustentada por las habilidades propias de otros modos de razonamiento. Entre ellos está el algorítmico –ya referido en apartado anterior-, el pensamiento crítico y el matemático. La figura 1, busca ilustrar las habilidades propias de otros modos de pensamiento y que sustentan el pensamiento computacional [4].

Figura 1. Relación del pensamiento algorítmico, crítico y matemático con el pensamiento computacional. La sumatoria de las diferentes habilidades propias a cada forma de pensamiento hace posible el pensamiento computacional. (Fuente: [4])

1.3.2.1 Pensamiento matemático

El concepto de pensamiento matemático puede interpretarse de distintas maneras, dependiendo del foco de atención y de los protagonistas implicados [11].

Cantoral y otros (2005) citado en María Saldaña (2012), en su libro sobre “Desarrollo del pensamiento matemático”, refieren varios modos de entender el concepto de pensamiento matemático y, por tanto, de analizar el desarrollo del mismo. Por un lado, atribuyen el término de pensamiento matemático a las formas en que piensan las personas que se dedican profesionalmente a las matemáticas. Por otro lado, entienden el pensamiento matemático como parte de un ambiente científico en el cual los conceptos y las técnicas matemáticas surgen y se desarrollan en la resolución de tareas [12].

Finalmente, Cantoral y otros (2005) concluyen observando que el pensamiento matemático incluye, por un lado, pensamiento sobre tópicos matemáticos, y por otro, procesos avanzados del pensamiento como abstracción, justificación, visualización, estimación o razonamiento bajo hipótesis [12].

1.3.2.2 Pensamiento crítico

En términos generales, el **pensamiento crítico** tiene que ver con la capacidad para razonar eficientemente, hacer juicios y tomar decisiones así como para resolver problemas.

La continua toma de decisiones se enfrenta a una rápida y exagerada cantidad de información la cual hay que saber analizar y seleccionar para elegir lo realmente verdadero y bueno.

Según el Dr. Peter A. Facione (2007), el termino **pensamiento crítico** significa buen juicio. En su artículo “Pensamiento Crítico: ¿Qué es y por qué es importante?” defiende algunas “*habilidades”* del pensamiento crítico, en donde las clasifica como “*habilidades cognitivas” [8].*

Estas habilidades son las siguientes: interpretación, análisis, evaluación, inferencia, explicación y auto regulación [13].

La **interpretación** es “comprender y expresar el significado o la relevancia de una amplia variedad de experiencias, situaciones, datos, eventos, juicios, convenciones, creencias, reglas, procedimientos o criterios”. La interpretación incluye las sub habilidades de categorización, decodificación del significado, y aclaración del sentido.

El **análisis** “consiste en identificar las relaciones de inferencia reales y supuestas entre enunciados, preguntas, conceptos, descripciones u otras formas de representación que tienen el propósito de expresar creencia, juicio, experiencias, razones, información u opiniones”.

La **evaluación** es definida como la “valoración de la credibilidad de los enunciados o de otras representaciones que recuentan o describen la percepción, experiencia, situación, juicio, creencia u opinión de una persona; y la valoración de la fortaleza lógica de las relaciones de inferencia, reales o supuestas, entre enunciados, descripciones, preguntas u otras formas de representación”.

La **inferencia significa** “identificar y asegurar los elementos necesarios para sacar conclusiones razonables; formular conjeturas e hipótesis; considerar la información pertinente y sacar las consecuencias que se desprendan de los datos, enunciados, principios, evidencia, juicios, creencias, opiniones, conceptos, descripciones, preguntas u otras formas de representación”.

La **explicación** es la capacidad de presentar los resultados del razonamiento propio de manera reflexiva y coherente.

La **autorregulación** es definida como “monitoreo auto consciente de las actividades cognitivas propias, de los elementos utilizados en esas actividades, y de los resultados obtenidos, aplicando particularmente habilidades de análisis y de evaluación a los juicios inferenciales propios, con la idea de cuestionar, confirmar, validar, o corregir el razonamiento o los resultados propios”.

1.3.2.3 Pensamiento algorítmico

Es notable como una palabra que nace de los trabajos de un matemático y astrónomo de la Edad Media (muy lejos de las computadoras en el tiempo) se convirtió con los años en uno de los pilares de la Ciencia Informática. En efecto el nombre se relaciona con el matemático árabe Al-Khwarizmi, quien desarrolló gran parte de su carrera en Bagdad, alrededor del año 800 DC. Allí creó un Centro Superior de Investigaciones Científicas y se dedicó especialmente al Algebra y la Astronomía. Sus procedimientos para resolución de ecuaciones y el tratado traducido al latín sobre números “Algoritmi de numero Indorum” lo han dejado como el referente más antiguo de la palabra “Algoritmo” [14].

Nos quedamos entonces con la definición “genérica” de ***Algoritmo:*** “Conjunto ordenado de operaciones que tienen como objetivo resolver un problema”. Esta definición excede la Informática e incluso las Ciencias duras y nos trasmite un par de atributos conceptuales importantes pero que nos queda implícita una competencia básica para elaborar un algoritmo

correcto: la capacidad de abstracción del problema del mundo real, para interpretar y sistematizar su solución.

Cuando acotamos el campo de aplicación de la definición de Algoritmo a la Informática o la Matemática, diremos que tendremos una lista ordenada y finita de pasos que dado un estado inicial nos permite transformarlo en un estado final "solución" en un tiempo también finito.

Teniendo en cuenta lo anteriormente dicho, un ***algoritmo*** es entonces "una secuencia finita, ordenada y lógica de pasos para llegar al objetivo de resolver un problema" y el "***pensamiento algorítmico"*** es la capacidad/ aptitud que tenemos para realizar el proceso de abstracción, modelización del problema, deducciones lógicas y síntesis de la solución que conduzca a escribir el algoritmo correcto.

1.3.3 Inicios del PC

Algunos de los pensadores pioneros en este campo incluyen a Papert, Wing y Wolfram. Una de las primeras referencias al PC se encuentra en (Papert, 1996) donde Papert describe el valor de aplicar primitivas cognitivas humanas a problemas orientados a objetos, poniendo de manifiesto las relaciones entre los componentes de un sistema complejo. Otras referencias similares se pueden encontrar en (Vee, 2013; Wolfram, 2016) donde hay referencias directas a las ideas fundamentales de dividir una tarea compleja en un conjunto de tareas más simples.

En el año 2006, Jeannette Wing publicó un artículo titulado "*Computational Thinking*" en la revista *Communications of the ACM* (Association for Computing Machinery) en el que defendía la necesidad de comenzar a poner en práctica las técnicas que utilizan los informáticos, en la planificación y desarrollo de sus procesos, a áreas científicas alejadas de la informática. En palabras de la propia Wing:

"el pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática" [6].

Este breve artículo, que iba dirigido a la comunidad científica del área de informática, tuvo una gran repercusión y dio el salto a otros ámbitos, como el de la educación. En un planteamiento interesante, Wing defiende que debiéramos preocuparnos seriamente de que

nuestros niños y jóvenes aprendan a resolver problemas como lo haría una computadora, poniendo en juego múltiples habilidades de abstracción, que pueden ser entrenadas.

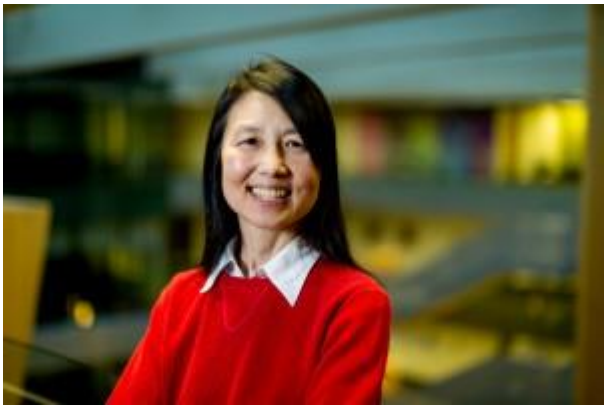


Figura 2. Jeannette Wing propone que las habilidades de abstracción y las técnicas de resolución de problemas utilizados por los científicos e ingenieros de la computación pueden ser enseñadas y aplicadas en otras disciplinas o actividades de la vida cotidiana.

1.3.4 Concepto del PC

Para abordar el concepto de "PC" procederemos a revisar las distintas definiciones que se han ido ofreciendo del mismo. Diferenciaremos en los siguientes epígrafes entre definiciones genéricas; operativas u operacionales, psicológico-cognitivas y educativo-curriculares [15].

1.3.4.1 Definiciones genéricas

Hace una década, Jeannette M. Wing (2006) citada en González (2016), publicó el artículo fundacional de la disciplina, en el cual define que el *"pensamiento computacional implica la resolución de problemas, el diseño de sistemas, y la comprensión de la conducta humana, haciendo uso de los conceptos fundamentales de la informática"* [6]. Así, la esencia del PC sería pensar como un científico de la computación cuando uno se enfrenta a un problema [15].

En dicho artículo, la autora define el PC como una habilidad básica y fundamental para cualquier sujeto inmerso en la actual realidad digital, llegando incluso a afirmar que el PC debería añadirse a la 'lectura, escritura, aritmética' como paquete básico en el desarrollo de las habilidades analítico-instrumentales de cualquier niño.

Sintetizamos a continuación qué es, y qué no es, pensamiento computacional en esta aproximación pionera de Wing (2006) [4]:

- *'Conceptualizing, not programming'* ('conceptualizar, no programar'): pensar como un científico de la computación, es decir, pensar computacionalmente, significa más que ser capaz de programar un ordenador. Requiere pensar en múltiples niveles de abstracción.
- *'Fundamental, not rote skill'* ('habilidad básica, no puramente mecánica'): una habilidad básica es aquella que cualquier persona debe dominar para funcionar en la sociedad actual.
- *'A way that humans, not computers, think'* ('una manera en que los humanos piensan, no las computadoras'): el pensamiento computacional es una manera en que los seres humanos solucionan problemas; no es tratar de que los humanos piensen como las computadoras (…).
- *'Complements and combines mathematical and engineering thinking'* ('se complementa y se combina con el pensamiento matemático e ingenieril'): las ciencias de la computación y, por tanto, el pensamiento computacional, descansan sobre el pensamiento matemático dado que, como todas las ciencias, sus bases formales surgen de las matemáticas. Además, las ciencias de la computación descansan inherentemente en el pensamiento ingenieril dado que trata de construir sistemas informáticos que interactúan con el mundo real-físico.
- *'Ideas, not artifacts'* ('ideas, no artefactos-objetos'): el pensamiento computacional no tiene sólo que ver con los objetos de hardware y software que producimos, y que están presentes físicamente a diario en nuestras vidas; tiene que ver sobre todo con los conceptos computacionales que usamos para aproximarnos a los problemas y solucionarlos, para gestionar nuestra vida, y comunicarnos e interactuar con otras personas de manera efectiva.
- *'For everyone, everywhere'* ('para cualquiera, en cualquier parte'): el pensamiento computacional será una realidad cuando sea tan inherente a las tareas y esfuerzos humanos, que desaparezca como término y filosofía explícitos.

Esta primera definición genérica, casi metafísica, del pensamiento computacional, viene siendo revisada y especificada en intentos sucesivos a lo largo de los últimos años, sin llegar aún a un acuerdo generalizado sobre la misma [15]. Veámoslo.

La propia Wing citada en González (2016), revisita su definición algo más tarde (2008), clarificando que *"el pensamiento computacional incluye los procesos de pensamiento*

implicados en la formulación de problemas y de sus soluciones, de tal modo que éstos estén representados de una manera que pueda ser abordada efectivamente por un agente-procesador de información", como un ordenador [16] .

"El pensamiento computacional es un enfoque o aproximación particular a la solución de problemas, de manera que éstos puedan ser formulados y resueltos con la ayuda de un ordenador. Con su desarrollo, los estudiantes se convierten de meros usuarios de herramientas a constructores de las mismas. A través del desarrollo del pensamiento computacional, los estudiantes adquieren un conjunto de conceptos tales y como la abstracción, la recursividad, y la iteración, que utilizan para procesar y analizar datos, y para crear artefactos tanto físicos como digitales. En suma, el pensamiento computacional es una metodología de solución de problemas que puede ser automatizada, transferida y aplicada a lo largo de las distintas materias y asignaturas" [17].

Siguiendo con este grupo de definiciones, de corte genérico, cuatro años más tarde, la Royal Society (2012) también aporta una definición sucinta que trata de capturar la esencia del PC [18]:

"El pensamiento computacional es el proceso de reconocimiento de los aspectos computables en el mundo que nos rodea, y de aplicar las herramientas y técnicas de las Ciencias de la Computación para comprender y razonar sobre sistemas y procesos, tanto naturales como artificiales".

1.3.4.2 Definiciones operativas

De este grupo de definiciones generales, se va avanzando progresivamente a definiciones más operativas u operacionales, que tratan de enumerar los elementos constitutivos del pensamiento computacional. Así, por ejemplo, la Fundación Nacional para la Ciencia de los Estados Unidos (*'National Science Foundation'*) citado por González (2016), en el marco de su curso "*CS Principles"*, cuyo objetivo es fijar y transmitir las bases de las Ciencias de la Computación al alumnado de Bachillerato y primeros años de universidad, define las siguiente siete ideas como esenciales del pensamiento computacional [15]:

i. "El pensamiento computacional es una actividad humana creativa.

ii. *La abstracción (uno de los elementos constitutivos, sino el central, del pensamiento computacional) reduce y elimina la información y detalles irrelevantes para focalizarse en los conceptos relevantes a la hora de entender y resolver un problema.*

iii. *Los datos y la información facilitan la creación de conocimiento.*

iv. *Los algoritmos son herramientas para desarrollar y expresar soluciones a problemas computacionales. Programar es un proceso creativo que produce artefactos-objetos computacionales.*

v. *Los dispositivos y sistemas digitales, y las redes que los interconectan, posibilitan y potencian una aproximación computacional a la resolución de problemas.*

vi. *El pensamiento computacional permite la innovación en otros campos, incluyendo las ciencias naturales, ciencias sociales, humanidades, artes, medicina, ingeniería, y negocios."*

Esta línea de especificar los elementos que constituyen el pensamiento computacional culmina con la *'Operational Definition of Computational Thinking for K-12 Education'*: una definición operativa del PC que sirve de marco de trabajo y vocabulario compartido para los profesores de informática (*'Computer Science Teachers'*) en las etapas de Educación Secundaria y preuniversitaria estadounidense. Esta definición operativa fue desarrollada inicialmente en 2011 por la *'Computer Science Teachers Association (CSTA)'* y la *'International Society for Technology in Education (ISTE)'* de los Estados Unidos; y sigue plenamente vigente en la actualidad. Así, la (CSTA & ISTE) [19] citadas en González (2016), dicen que *"el pensamiento computacional es un proceso de solución de problemas que incluye (aunque no está limitado a) las siguientes* ***características*** [15]*:*

- *Formular problemas de un modo que se haga posible utilizar un ordenador y otras máquinas en su resolución.*
- *Organizar lógicamente y analizar datos.*
- *Representar datos a través de abstracciones tales como modelos y simulaciones.*
- *Automatizar soluciones a través del pensamiento algorítmico (una serie de pasos discretos y ordenados).*

- *Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos.*
- *Generalizar y transferir este proceso de solución de problemas a una amplia variedad de situaciones".*

La novedad que aporta esta definición operativa (CSTA & ISTE, 2015, en línea) es añadir una dimensión actitudinal al pensamiento computacional, pues continua diciendo: *"estas habilidades son apoyadas y reforzadas por una serie de disposiciones o actitudes que son también dimensiones esenciales del PC* [19]*. Estas disposiciones y actitudes incluyen:*

- *Confianza al manejarse con la complejidad.*
- *Persistencia al trabajar con problemas difíciles.*
- *Tolerancia a la ambigüedad.*
- *Capacidad de hacer frente a problemas abiertos (sin una solución concreta y evidente).*
- *Capacidad de comunicarse y trabajar con otros para llegar a una meta-solución común".*

En cualquier caso, y como síntesis de las distintas definiciones operativas que han ido apareciendo estos últimos años, Grover & Pea (2013ª) citados en González (2016), recogen *"los elementos que han conseguido un cierto consenso, y amplia aceptación, para formar parte del PC; y que deberían estar en la base de cualquier currículum que pretenda su desarrollo* [15]*:*

- *Abstracción y generalización de patrones (incluyendo modelos y simulaciones).*
- *Procesamiento sistemático de la información.*
- *Sistemas de símbolos y representación.*
- *Noción algorítmica de control de flujo.*
- *Descomposición del problema estructurada de problemas (modularización).*
- *Pensamiento iterativo, recursivo y paralelo.*
- *Lógica condicional.*

- *Limitadores de eficiencia y rendimiento.*
- *Depuración y detección sistemática de errores".*

1.3.4.3 Definiciones psicológico-cognitivas

"el pensamiento computacional es aplicable a prácticamente cualquier ámbito (…) e incluye las siguientes 4 fases específicas" [15]*:*

- Descomposición del problema de un problema o tarea en pasos discretos
- Reconocimiento de patrones (regularidades)
- Generalización de dichos patrones y abstracción (descubrir las leyes o principios que causan dichos patrones)
- Diseño algorítmico (desarrollar instrucciones precisas para resolver el problema y sus análogos)".

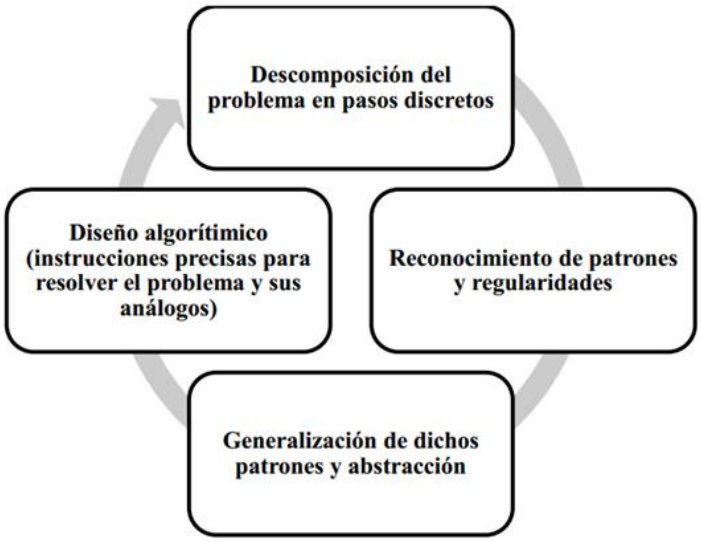


Figura 3. Las 4 faces-pasos cognitivos del pensamiento computacional

- **Descomposición del problema**: es la capacidad para fraccionar una tarea minuciosa y detalladamente en los pasos que la forman, de manera que luego podamos explicar unívocamente el proceso a una tercera persona o a un ordenador, o incluso como notas de procedimiento para uno mismo. Descomponer un problema frecuentemente conduce a

un posterior reconocimiento y generalización de patrones, y así en última instancia a la capacidad para diseñar un algoritmo.

- **Reconocimiento de patrones**: es la capacidad para percibir similaridades o diferencias comunes que nos ayudan a hacer predicciones y nos conducen hacia 'atajos' o 'accesos directos' (*'shortcuts'*) al núcleo de un problema. El reconocimiento de patrones es frecuentemente la base para el diseño algorítmico y la resolución de problemas.

- **Generalización de patrones y abstracción**: es la capacidad para filtrar e ignorar toda la información que no es necesaria para resolver un cierto tipo de problema, y de generalizar la que sí es necesaria. La generalización de patrones y la abstracción nos permiten representar una idea o un proceso en sus términos generales-formales podemos utilizar dicha representación para resolver problemas análogos de similar naturaleza.

- **Diseño algorítmico**: es la capacidad de desarrollar una estrategia 'paso por paso' (secuencia de instrucciones perfectamente definida) para resolver un problema. El diseño algorítmico está basado a menudo en la descomposición del problema previa de un problema y en la identificación de los patrones que ayudan en su resolución. En Ciencias de la Computación, así como en matemáticas, los algoritmos suelen escribirse de manera abstracta, utilizando variables en el lugar de números específicos.

Centrándose únicamente en los procesos mentales, Selby y Woollard (2013) definen el PC como *un proceso cognitivo o mental, de humanos, no de máquinas, de resolución de problemas en el sentido amplio, e involucrando habilidades tales como*:

- Abstracción – Consiste en ocultar la complejidad inherente de la realidad para representar solo sus aspectos esenciales
- Descomposición del problema – Consiste en dividir una tarea o problema en partes más simples para que puedan ser resueltos
- Pensamiento algorítmico – Consiste en definir una tarea como un conjunto de instrucciones simples paso a paso
- Evaluación: consiste en evaluar las ventajas y limitaciones de una solución.
- Generalización: consiste en poder pasar de una situación específica a una más general.

Varias compañías como Microsoft y Google también han desarrollado varios proyectos y programas alrededor de PC y aprendizaje de programación. Google (2017), por ejemplo, ha definido una guía conceptual sobre PC que distingue los procesos mentales de los resultados tangibles. Los primeros implican: abstracción, diseño de algoritmos, descomposición del problema, análisis de datos, reconocimiento de patrones. El último: automatización, recopilación de datos, representación de datos, paralelización, generalización de patrones, simulación [15].

1.3.4.4 Definiciones educativo-curriculares

Más que definiciones en sentido estricto, incluimos en este epígrafe un par de modelos o marcos teóricos de integración del pensamiento computacional (*'computational thinking frameworks'*) en el sistema educativo [15].

a) <u>El modelo CAS ('Computing at School') de Reino Unido</u>

'Computing at School' (CAS) UK Department of Education, 2013), es una alianza estratégica formada por el Ministerio de Educación del Reino Unido, diversas universidades e investigadores del país, y empresas del ámbito informático y tecnológico (p.e. Microsoft); cuyo principal objetivo es promover y apoyar la excelencia en la educación en Ciencias de la Computación dentro del sistema británico. Nace en el contexto del diseño e implantación en Reino Unido del nuevo currículum en Ciencias de la Computación (UK Department of Education, 2013), que introduce, ya desde el curso 2014/2015 y a lo largo de todas las etapas educativas obligatorias (desde los 5 a los 16 años), contenidos relativos a la programación informática, pero siempre tratados desde la perspectiva del pensamiento computacional.

> El objetivo de CAS es *"ayudar a nuestros jóvenes a interiorizar y sentirse confiados con el pensamiento computacional, y a que tomen conciencia de cómo la tecnología y los datos están en la base de nuestra sociedad actual. Para ello, ofrecemos a nuestros profesores esta comunidad con recursos útiles y de alta calidad, unidades didácticas…",*

Pues bien, la comunidad *'CAS Barefoot'* aporta su propia definición:

> *"El pensamiento computacional es abordar un problema de manera que un ordenador pueda ayudarnos luego en su resolución (…) éste es un proceso de dos fases:*

primera, pensamos en los pasos que se necesitan para resolver el problema; segunda, utilizamos nuestras habilidades técnicas para poner a nuestro ordenador a trabajar sobre el problema (…) el pensamiento computacional no es pensar sobre *ordenadores ni pensar* como *ordenadores. Los ordenadores no piensan por sí mismos, ¡al menos no todavía!".*

Así, defienden que el pensamiento computacional incluye **6 conceptos** ('lógica', 'algoritmos', 'descomposición del problema', 'patrones', 'abstracción', y 'evaluación sistemática') y **5 aproximaciones** ('experimentación', 'creación', 'depuración', 'perseverancia', y 'colaboración) **(ver anexo 1)**.

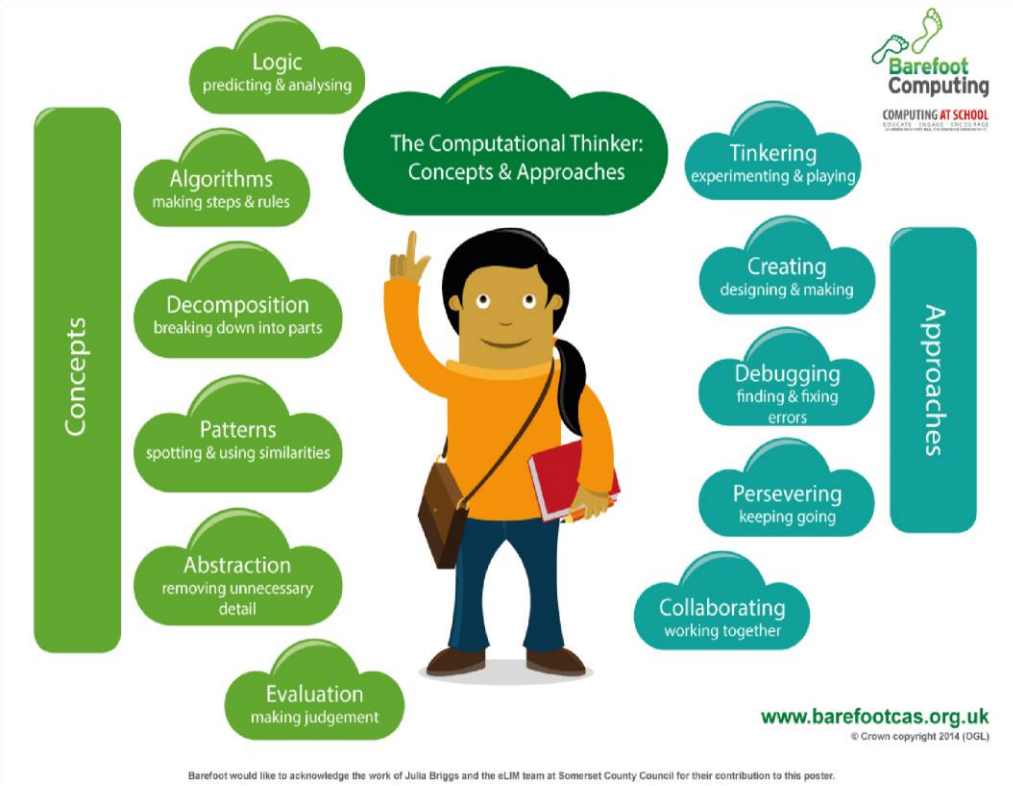


Figura 4. Marco-modelo de pensamiento computacional en el aula propuesto desde 'CAS Barefoot' [1].

b) <u>Un marco desde Estados Unidos: el modelo MIT-Harvard</u>

Desde los Estados Unidos, los prestigiosos investigadores Karen Brennan (Universidad de Harvard) y Mitch Resnick (MIT) han formulado un modelo alternativo del pensamiento computacional [20]. A lo largo de 5 años, han ido desarrollando su *'computational thinking framework'*, a través de la observación directa, entrevistas y análisis de proyectos de niños y jóvenes trabajando como diseñadores digitales interactivos (*'interactive media designers' [3]*).

El contexto para la investigación y formulación de su modelo es Scratch 213: una plataforma-lenguaje visual de programación 'por bloques' que permite a los niños y jóvenes crear sus propias historias interactivas, animaciones, juegos y simulaciones; y posteriormente compartir dichas creaciones a través de una comunidad online con otros jóvenes programadores alrededor del mundo.

En la medida que Scratch se ha convertido en el lenguaje de programación para niños más usado en el mundo, y el que mejores características tiene para desarrollar el pensamiento computacional, el modelo que presentamos a continuación es admitido como una referencia de primer orden a la hora de enmarcar la integración del PC en el aula.

Este modelo, que se llamará MIT-Harvard, se articula alrededor de 3 dimensiones clave (***ver anexo 2)***:

a) 'conceptos computacionales' (*'computational concepts'*), es decir, los conceptos que los niños emplean cuando programan.
b) 'prácticas computacionales' (*'computational practices'*), esto es, las prácticas que los niños emplean mientras programan.
c) 'perspectivas computacionales' (*'computational perspectives'*).

Conceptos computacionales [¿Qué aprenden?]	**Prácticas computacionales [¿Cómo lo aprenden?]**	**Perspectivas computacionales [¿Para qué lo aprenden?]**
Secuencias	Experimentación e iteración	Expresarse
Bucles	Evaluación y depuración	Conectarse
Eventos	Reutilización y remezcla	Interrogarse
Paralelismo	Abstracción y modularización	
Condicionales		
Operadores		
Datos		

Tabla 1.Diagrama resumen del Modelo MIT-Harvard de pensamiento computacional (Brennan & Resnick, 2012) Veámoslo con algo más de detalle (ver anexo 2).

1.3.5 Pensamiento Computacional y la resolución de problemas

La resolución de problemas ha sido objeto de intensa investigación en el proceso de enseñanza – aprendizaje de la carrera de ingeniería informática, atendiendo a las demandas del siglo XXI. Entre los estudios sobre métodos de resolución de problemas destacamos los trabajos de Jeannette Wing (2006), que describe el Pensamiento Computacional como un enfoque analítico general para la resolución de problemas [6].

En realidad el pensamiento computacional es una variante del dominio metodológico que se conoce como "resolución de problemas". Es una restricción de la resolución de problemas a aquellos problemas cuya resolución se puede implementar con ordenadores. En este caso es muy importante distinguir que los aprendices no son sólo los usuarios de la herramienta, sino que sobre todo se convierten en los constructores y en los autores de las herramientas.

Para eso los alumnos utilizan procedimientos, conjuntos de objetos de conocimiento y conceptos que constituyen dominios que tratamos de forma separada en este escrito. Como son la abstracción, la recursividad y la iteración- Los utilizan para procesar y analizar los datos de cara a crear métodos de resolución de problemas, y crear artefactos reales y virtuales para resolverlos. El pensamiento computacional de esta forma se puede considerar también como una metodología de resolución de problemas que se puede automatizar.

La otra vinculación del pensamiento computacional con la resolución de problemas lo constituye la visión que se puede desarrollar en los alumnos y que se manifiesta en el aula para encontrar soluciones a problemas a través del ordenador.

Habilidades asociadas al pensamiento computacional

Aquí se tomarán en cuenta tan solo algunas de las habilidades referidas por los diferentes autores y que se relacionan a las tres formas de pensamiento (algorítmico, matemático y crítico), necesarias en la solución de problemas [4]. Estas habilidades son:

Habilidad para encontrar patrones: Esta habilidad del pensamiento algorítmico es visible cuando el estudiante identifica los pasos para resolver los algoritmos y realiza abstracciones de una secuencia de caracteres [21].

Sentido numérico: Esta habilidad del pensamiento matemático se evidencia cuando se hace una interpretación de los datos que componen un problema y cuando se realizan las operaciones matemáticas básicas asociadas a ese problema. Bajo esta categoría se responde a la pregunta ¿Cuáles operaciones o procesos involucrados han de fortalecerse? Se tiene en cuenta algunos de los rasgos planteados por Berch (2005) como lo son; la capacidad para utilizar las relaciones entre las operaciones aritméticas y de entender el sistema de numeración en base 10, utilizar los números y métodos cuantitativos para comunicarse, procesar e interpretar la información, deseo de dar sentido a situaciones numéricas, mediante la búsqueda de vínculos entre nueva información y el conocimiento previamente adquirido, comprender múltiples relaciones entre números, reconocer los números de referencia y patrones de números, entender los números como referentes para medir las cosas en el mundo real, entre otras [22].

Interpretación/análisis: Bajo esta categoría asociada al pensamiento crítico según Facione (2007), se evidencia la habilidad con la que debe contar un estudiante para identificar el problema planteado, proponer una solución acertada y explicar cómo llegó a la solución de un problema [13].

1.3.6 Componentes del Pensamiento Computacional (CPC)

Al igual que la definición de PC, hablar de sus componentes resulta difícil por la falta de consenso por parte de los autores que han estado hablando sobre el tema; como son Jeannette M. Wing (2006), (Barr & Stephenson, 2011, p. 51), (Royal Society, 2012, p. 29), Grover & Pea (2013ª, p. 39-40), Basogain (2015), (National Science Foundation, 2015, en línea), (CSTA & ISTE, 2015, en línea), (Google for Education, 2015, en línea); y muchos otros autores, aunque la mayoría de ellos coincide en describirlos como fases o elementos del pensamiento computacional.

En resumen, el Pensamiento Computacional es un enfoque para la resolución de problemas que enfatiza la integración del pensamiento crítico, los conceptos de la computación y las tecnologías digitales.

A pesar de que cada uno de estos autores haya planteado a su manera las fases, procesos, elementos o componentes del PC, independientemente de cómo lo hayan nombrado, podemos percatarnos de que la mayoría de ellos coincide en 4 aspectos: la generalización de patrones y abstracción, la descomposición del problema, el

reconocimiento de patrones y el diseño o pensamiento algorítmico. La propia Wing concluye: “el proceso de pensamiento más importante y de alto nivel en el pensamiento computacional es el proceso de abstracción.”

Teniendo en cuenta toda la bibliografía consultada acerca del tema y después de revisar las diferentes definiciones de pensamiento computacional se planteó y contextualizó la siguiente definición que es la que se defiende en esta investigación:

Una definición contextualizada del PC

El Pensamiento Computacional es un proceso cognitivo ejecutado por humanos para la resolución de problemas diversos haciendo uso de conceptos computacionales que involucra, de forma relacionada, los componentes siguientes:

Abstracción: Consiste en ocultar la complejidad inherente de la realidad para representar solo sus aspectos esenciales.

Análisis y representación de datos: Consiste en extraer del problema todos los datos, analizarlos y representarlos adecuadamente para ser usados por métodos de resolución.

Descomposición del problema: Consiste en dividir una tarea o problema en partes más simples para que puedan ser resueltos.

Pensamiento algorítmico: Consiste en definir una tarea como un conjunto de instrucciones simples paso a paso.

Reconocimiento y generalización de patrones: Consiste en reconocer situaciones específicas que se repiten y poder generalizarlas.

Simulación / Evaluación: Consiste en imitar el funcionamiento de un sistema del mundo real cuando evoluciona en el tiempo o extender la solución de un problema en todo su universo de posibles valores.

1.4 Conclusiones del capítulo

- El pensamiento computacional es un tema emergente dentro de las tecnologías actuales para desarrollar el aprendizaje de los individuos y aumentar sus competencias computacionales para hacer frente a las demandas tecnológicas de la sociedad actual.

- Los estudiantes de primer año de ingeniería informática no trabajan los componentes del pensamiento computacional de forma cohesionada desde las diferentes disciplinas, por tanto no se aprovechan los beneficios que estos pueden aportar en su formación.
- Una definición contextualizada de pensamiento computacional es propuesta en esta investigación; definiéndolo como un proceso cognitivo ejecutado por humanos para la resolución de problemas diversos haciendo uso de conceptos computacionales que involucra, de forma relacionada, los componentes siguientes: abstracción, análisis y representación de datos, descomposición del problema, pensamiento algorítmico, reconocimiento y generalización de patrones, simulación / evaluación.

Capítulo II: Propuesta de Solución

2.1 Introducción

En este capítulo, se abordan teóricamente los componentes del pensamiento computacional (CPC) como metodología a emplear en el desarrollo de la solución propuesta. Se presenta una diversidad de problemas resueltos seleccionados de disciplinas que abordan temas referentes a la lógica algorítmica.

2.2 Componentes del Pensamiento Computacional

A pesar de no existir orden entre los componentes ya que éstos se relacionan, en este capítulo se presentarán en el orden siguiente:

- Abstracción.
- Análisis y representación de datos.
- Descomposición del problema.
- Pensamiento algorítmico.
- Reconocimiento y generalización de patrones.
- Simulación/ Evaluación.

2.2.1 Abstracción

Al hacer una reflexión sobre las aproximaciones dadas por los diferentes autores sobre los componentes del pensamiento computacional, se encuentra como constante el proceso de abstracción.

En el mundo, incluso el objeto más simple o el fenómeno más cotidiano se vuelve un auténtico desafío si se pretende analizarlo y comprenderlo. La abstracción es la habilidad que le permite al ser humano combatir la complejidad al considerar sólo lo esencial del objeto o fenómeno que se esté analizando [23].

El concepto de ***abstracción*** está vinculado al verbo abstraer. El verbo abstraer es definido por la real academia española como: “separar por medio de una operación intelectual las

cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción".

En un artículo de marzo de 2006 para las Comunicaciones del ACM, Wing (2006) plantea lo siguiente: (…) la ***abstracción*** se usa para definir patrones, generalizar a partir de instancias específicas y parametrizar. Se usa para dejar que un objeto represente a muchos. Se utiliza para capturar propiedades esenciales comunes a un conjunto de objetos, al tiempo que oculta distinciones irrelevantes entre ellos [6].

La abstracción nos da el poder de escalar y manejar la complejidad. La aplicación recursiva de la abstracción nos permite construir sistemas cada vez más grandes, con el caso base (al menos para ciencias de la computación) como bits (0 y 1) (…).

Para Barr & Stephenson (2011), la ***abstracción*** es simplificar de lo concreto a lo general a medida que las soluciones están siendo desarrolladas [17].

Según López, la abstracción es una habilidad esencial para la construcción de modelos y la descomposición del problema de problemas y nos permite reducir la complejidad [23].

A decir de Olabe y Basogain, la abstracción significa identificar la esencia del proceso que se desea crear eliminando todos los detalles superfluos [24].

Niveles de abstracción

Jeannette Wing confirma la importancia de la abstracción en el pensamiento computacional, haciendo hincapié en la necesidad de pensar en múltiples niveles de abstracción. Considera el ejemplo de un automóvil:

1. Cada pieza de un automóvil, está compuesta por átomos y cada átomo está compuesto por electrones, protones y neutrones. Llamemos a esta manera de abstraer el automóvil, nivel de abstracción **"atómico"**.
2. Los automóviles están compuestos de piezas como lo son: tuercas, varillas, remaches, alambres, envases de plástico, entre otros. Llamemos a esta manera de abstraer el automóvil, nivel de abstracción de **"piezas"**.
3. En un nivel superior de abstracción, el automóvil se encuentra compuesto por diversos mecanismos como lo son: el motor, el alternador, los inyectores de combustible, los frenos, llamemos a esta manera de abstraer el automóvil, nivel de abstracción de **"mecanismos"** [16].

Características de la abstracción

- La eliminación y ocultamiento de los detalles: separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente.
- La generalización: considerar el mismo objeto en su pura esencia o noción.

Desarrollo de habilidades de abstracción

El psicólogo Jean Piaget (1896-1980) mediante extensos estudios aplicados desde niños recién nacidos hasta personas adultas, derivó cuatro periodos de desarrollo cognitivo: sensorio-motor, preoperatorio, operaciones concretas y operaciones formales. La etapa de las operaciones formales, que se encuentra alrededor de los doce años a la edad adulta, es donde las personas son capaces de pensar de manera abstracta y donde es pertinente proporcionar contenidos que desarrollen sus capacidades de abstracción. La abstracción es fundamental para la ciencia e ingeniería en general, juega un papel crítico en la creación de teorías, modelos, análisis y producción de dispositivos de ingeniería. El pensamiento computacional identifica a la abstracción como una de las grandes ideas de las ciencias computacionales y que las habilidades de abstracción son cruciales para el futuro en el desarrollo científico y tecnológico. Sin embargo, incluso en las carreras de ingeniería o ciencias, en sus programas de estudio no contienen cursos sobre abstracción, no obstante, todo depende de la habilidad de abstracción para resolver problemas. Por lo tanto, la abstracción es una habilidad esencial, pero que se desarrolla indirectamente a través de otros tópicos, por ejemplo, mediante cursos de matemáticas, de programación o ingeniería del software.

2.2.2 Análisis y representación de datos

El Análisis de Datos (*Data Analysis, o DA*), según Rouse, es la ciencia que examina datos en bruto con el propósito de sacar conclusiones sobre la información [25].

El análisis de datos es una técnica y por medio de ésta se inspeccionan, purifican y transforman datos, con la finalidad de destacar toda la información que sea de gran utilidad, a fin de poder elaborar conclusiones que sirvan de apoyo en la toma de decisiones.

Técnicas del análisis de datos

Análisis de datos cualitativo: Los datos se presentan de manera verbal (en ocasiones en gráficas). Se basa en la interpretación. Sus tipos más comunes son las entrevistas abiertas, grupos de discusión y grupos de observación.

Análisis de datos cuantitativos: Los datos se presentan en forma numérica. Se basa en resultados tangibles.

Análisis y representación de datos: Consiste en extraer del problema todos los datos, analizarlos y representarlos adecuadamente para ser usados por métodos de resolución.

2.2.3 Descomposición del problema

El concepto de ***descomposición del problema*** está vinculado al verbo descomponer, que es definido por la real academia española como: “Separar las diversas partes que forman un compuesto”.

La descomposición del problema, denominado a veces particionado o elaboración del problema, es una actividad que se asienta en el núcleo del análisis de requisitos del software. Durante la actividad de exposición del ámbito no se intenta descomponer el problema totalmente. Más bien, la descomposición del problema se aplica en dos áreas principales:

- La funcionalidad que debe entregarse.
- El proceso que se empleará para entregarlo.

Barr & Stephenson (2011) plantea a su vez que la ***descomposición del problema*** es fraccionar los problemas en partes más pequeñas, lo que hace más sencilla su resolución [17].

En el documento “El Pensamiento Computacional. Un reflejo del razonamiento y la comprensión humanos”, Luis Puente (2016) hace referencia a Wing (2006) y que para ella, el objetivo de la descomposición del problema, es la de fragmentar la complejidad de un problema en pequeñas series de más fácil manejo [26].

Los autores como Xabier Olabe, Miguel Ángel Basogain y Juan Carlos Basogain (2015), lo definen como, el proceso por el cual se divide el problema en partes, y cada una de ellas en sus correspondientes componentes básicos para transformar un problema grande y complejo en un conjunto pequeño de tareas sencillas e interdependientes [24].

El grupo de investigación educativa de Google o *'Google for Education'* (2016) definen la ***descomposición del problema*** como "la capacidad para fraccionar una tarea minuciosa y detalladamente en los pasos que la forman, de manera que luego podamos explicar unívocamente el proceso a una tercera persona o a un ordenador, o incluso como notas de procedimiento para uno mismo". Descomponer un problema frecuentemente conduce a un posterior reconocimiento y generalización de patrones, y así en última instancia a la capacidad para diseñar un algoritmo *(Google for Education, 2015, en línea),* [15]. Ejemplos:

a. Cuando probamos un nuevo plato de cocina e identificamos los distintos ingredientes que dan forma a su sabor, estamos *descomponiendo* el plato en sus ingredientes particulares.

b. Cuando le damos a alguien indicaciones precisas para llegar a nuestro domicilio (p.e. *"sales de la boca del metro, giras a la derecha, caminas recto y coges la segunda calle a la izquierda…"*), estamos *descomponiendo* el proceso de 'ir de un sitio a otro'

c. En matemáticas, podemos *descomponer* un número, como 256,37, de la siguiente manera: $2*10^{2} + 5*10^{1} + 6*10^{0} + 3*10^{-1} + 7*10^{-2}$

2.2.4 Pensamiento algorítmico

Algoritmo

Para López (2008) (citado en Lugo, 2016), un algoritmo son instrucciones detalladas y precisas de un procedimiento que asegura una solución correcta a un interrogante en particular; estas instrucciones son finitas y con un orden lógico [4].

Pensamiento algorítmico

Cuando se habla de algoritmos también se hace mención a este tipo de pensamiento. En el reporte del Consejo Nacional de Investigación de Estados Unidos (1999), titulado "Being Fluent with Information Technology", se define como una serie de operaciones que incluyen "la descomposición del problema funcional, la repetición (iteración y / o recursividad), organizaciones de datos básicos (registro, matriz, lista), la generalización y la parametrización, el algoritmo de comparación de programas, diseño de arriba hacia abajo, y el refinamiento" [4].

Una de las aproximaciones a este tipo de pensamiento y que se considera en esta investigación, es la proporcionada por Futschek (2006) (citado por Lugo, 2016). Para el autor, el pensamiento algorítmico también puede ser entendido como un conjunto de habilidades que permiten la comprensión de los algoritmos. Algunas de estas habilidades son la capacidad para analizar problemas dados, especificar problemas con precisión, creatividad para crear nuevos algoritmos y/o hacerlos más eficientes [4].

Futscherk (2006) (citado en Lugo, 2016) también propone que para enseñar el pensamiento algorítmico, se deben proponer numerosos problemas, que han de ser escogidos de manera que no sean ni tan simples o tan complejos que frustren al estudiante. Se debe empezar con los más básicos y cuidando no involucrar lenguaje de programación, ya que este lenguaje puede ser complejo para los principiantes.

2.2.5 Reconocimiento y generalización de patrones

Reconocer, será un atributo básico del ser humano así como de otros organismos vivientes. En nuestra vida pasamos todo el tiempo reconociendo, reconocemos que alguien tiene razón, reconocemos una solución a un problema, construcciones mentales equivocadas [27], etc.

Patrón o clase, es la descripción de un objeto, se refiere al conjunto de atributos para definir un objeto. Categoría determinada por atributos comunes.

Proceso de reconocimiento, se refiere a la capacidad del individuo para discriminar los datos de entrada y decidir que un objeto pertenece a una y sólo una de las clases o patrones que se estén estudiando.

Reconocimiento de patrones puede definirse como la “categorización” de los datos de entrada en clases o patrones mediante la extracción de propiedades significativas que permiten discriminar entre las clases en estudio.

Según la psicología cognitiva, el reconocimiento de patrones es un procesamiento de información que consiste en codificar el estímulo de alguna manera y compararlo con un patrón ya existente en la memoria. El mayor problema que se enfrentaba la psicología cognitiva es responder a: ¿cómo se codifica el estímulo para compararlo con el patrón ya

existente en la memoria? ¿Se codifica como una plantilla o como una serie de características?[27].

Proceso de generalización

Radford (citado por Rivera y Becker, 2011) y este a su vez citado por Juan Carmona (2015), considera la generalización como la regularidad que captan los estudiantes, a partir de relaciones establecidas sobre particularidades en la secuencia, extendiendo esa regularidad a la secuencia. Esta regularidad debe permitirles expresar algún término de la secuencia [28].

Radford (2007), argumenta que la generalización de patrones descansa en la capacidad de identificar una regularidad en algunos casos particulares, y extender esa regularidad a otros términos, permitiendo así una expresión vinculada con todos los términos de la secuencia.

Teoría del reconocimiento de patrones de Marry y Nishihara X

Marry y Nishihara han desarrollado la idea de procesamiento en el reconocimiento de patrones. Esta idea consiste en transformar una representación en otra.

En el reconocimiento de patrones es un grave problema almacenar muchas representaciones de un mismo objeto para realizar el reconocimiento. Por eso es mejor no centrarse en los puntos de vista del perceptor para el reconocimiento, porque estos son muchos. Es mejor centrarnos en la cosa misma, porque ésta es una sola. Por ello, el modelo de proceso que adoptan Marr y Nishihara se centra en el objeto, no en el perceptor [27].

Una de las aproximaciones que se considera en esta investigación, es la proporcionada por el grupo de investigación educativa de Google o *'Google for Education' (2015).* Para ellos, el ***reconocimiento de patrones*** no es más que: la capacidad para percibir similaridades o diferencias comunes que nos ayudan a hacer predicciones y nos conducen hacia 'atajos' o 'accesos directos' (*'shortcuts'*) al núcleo de un problema. El reconocimiento de patrones es frecuentemente la base para el diseño algorítmico y la resolución de problemas [29]. Ejemplos:

a. Los niños identifican *patrones* en las reacciones de sus padres y profesores a su comportamiento, en orden a establecer qué está bien y qué está mal. Y basan su comportamiento futuro en función de dichos patrones.

b. Los corredores de bolsa buscan *patrones* en el valor de las acciones para decidir cuándo comprar y cuándo vender.

c. En matemáticas, podemos seguir un *patrón* para explicar la lógica que subyace a que el producto de dos números negativos es un número positivo:

*(-3) * (3) = -9*

*(-3) * (2) = -6*

*(-3) * (1) = -3*

*(-3) * (0) = 0*

*(-3) * (-2) = 6*

d. En geometría, al calcular el mayor área posible para un rectángulo de un perímetro dado, podemos observar *patrones* que involucran el alto, ancho y área del mismo como:

- ✓ *En la medida que el alto y el ancho se aproximan el uno al otro en sus valores, el área se incrementa.*
- ✓ *En la medida que aumenta la diferencia entre los valores del alto y del ancho, el área se reduce.*
- ✓ *Este patrón nos conduce a la conclusión de que el rectángulo con el área mayor es un cuadrado.*

2.2.6 Simulación / Evaluación

Jeannette Wing (citado por López, 2014), sostiene que el pensamiento computacional complementa y combina el pensamiento matemático y la ingeniería porque se basa en las matemáticas como sus fundamentos y recurre a la ingeniería ya que nuestros sistemas interaccionan con el mundo real. En suma, nuestros sistemas están limitados por la física del dispositivo subyacente, pero por medio de la computadora podemos construir mundos virtuales o ***simulaciones*** sin las restricciones de la realidad física [30].

La teoría que hay detrás es antigua ya que toda se basa en la definición de modelos matemáticos y estadísticos, y en el estudio de la evolución de estos modelos a lo largo de un período determinado de tiempo. No obstante, la simulación ha avanzado de forma realmente

importante a partir de la explosión de las herramientas informáticas, por la facilidad que supone generar los resultados y tratar los datos y los cálculos complejos. De hecho, la simulación está estrechamente ligada al inicio de la informática con las simulaciones de balística para aplicaciones militares a mediados de los cuarenta.

El concepto de ***simulación*** es una de las principales ideas del pensamiento computacional en el proceso de solución de problemas. Entre los objetivos operativos del pensamiento computacional que se relacionan estrechamente con la simulación son los siguientes:

- Representar datos mediante abstracciones, como modelos y simulaciones.
- Formular problemas de manera que permitan usar computadores y otras herramientas para solucionarlos.

Un modelo es una representación abstracta (matemática, declarativa, visual, etc.) de fenómenos, sistemas o procesos. El humano ha podido plantear leyes o modelos que representan la esencia de los fenómenos y que tienen como finalidad simplificar el fenómeno real para poder analizarlo, comprenderlo, predecirlo o controlarlo.

Según WINSTON (1994) citado por Galicia (2011), se puede definir la simulación como la técnica que imita el funcionamiento de un sistema del mundo real cuando evoluciona en el tiempo. La simulación no es una técnica de optimización. Más bien es una técnica para estimar las medidas de desempeño del sistema modelado [31].

López considera la ***simulación*** como el proceso de representar o modelar un sistema con la finalidad de comprender su comportamiento. La simulación involucra realizar experimentos usando modelos para probar hipótesis. Las simulaciones por computadora son artefactos de la invención humana, hechas a partir del pensamiento, donde las instrucciones del programador se convierten en la ley del mundo virtual [30].

Según el diccionario de sinónimos y antónimos, ***evaluar*** significa de entre muchas cosas; tasar, valorar, calcular, determinar, estimar, justipreciar [32].

La ***evaluación*** se puede entender de diversas maneras, dependiendo de las necesidades, propósitos u objetivos de la institución educativa, tales como: el control y la medición, el enjuiciamiento de la validez del objetivo, la rendición de cuentas, por citar algunos propósitos.

Desde esta perspectiva se puede determinar en qué situaciones educativas es pertinente realizar una valoración, una medición o la combinación de ambas concepciones [33].

2.3 Desarrollo del PC mediante problemas resueltos

Módulo de Problemas Resueltos (MPR)

Atendiendo a todas las dificultades anteriores, se presenta a continuación varios problemas resueltos desde la perspectiva del pensamiento computacional. El "módulo de problemas resueltos" propuestos en esta investigación está formado por 22 problemas que incluyen temas de las asignaturas: Matemática I, Matemática II y Matemática Discreta; además presenta algunos problemas de búsqueda sin heurística a partir de problemáticas que aparecen en juegos sencillos.

Los problemas mencionados anteriormente están agrupados en cuatro subgrupos en el siguiente orden:

1. **Problemas sin Heurísticas**
   - *Problema Resuelto # 1:* Torres de Hanói
   - *Problema Resuelto # 2:* Misioneros y Caníbales
   - *Problema Resuelto # 3:* El Arriero
   - *Problema Resuelto # 4:* Jarras de agua (8, 5, 3)
   - *Problema Resuelto # 5:* Jarras de agua (4 y 3 litros)
2. **Problemas de matemática I**
   - *Problema Resuelto # 6:* Límite de una función real de una variable real.
   - *Problema Resuelto # 7:* Continuidad de una función real de una variable real.
   - *Problema Resuelto # 8:* Derivada n-ésima de una función real de una variable real.
   - *Problema Resuelto # 9:* Derivada n-ésima de una función real de una variable real.
   - Problema Resuelto # 10: Problema de optimización de una función real de una variable real.
3. **Problemas de Matemática II**
   - *Problema Resuelto # 11:* Interpretación geométrica de la derivada parcial.
   - *Problema Resuelto* # 12: Interpretación física de la derivada parcial.

- *Problema Resuelto # 13:* Derivada direccional de una función real de varias variables.
- *Problema Resuelto # 14:* Extremos condicionados de una función real de varias variables.
- *Problema Resuelto # 15:* Volumen de un sólido.

4. Problemas de Matemática Discreta

- **Cálculo proposicional**

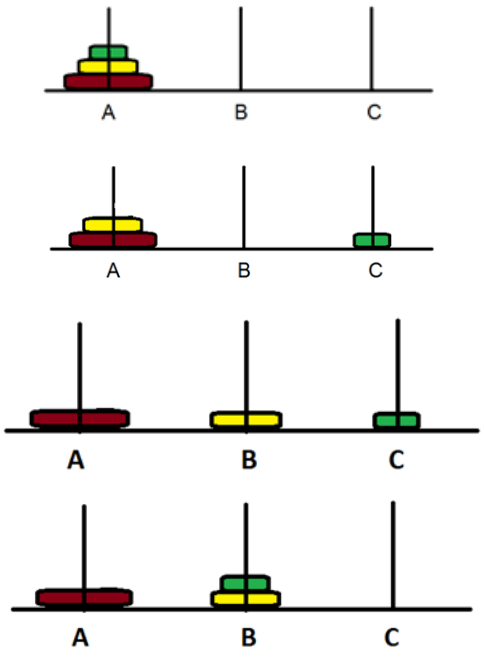
  - ✓ *Problema Resuelto # 16:* Representación de estructuras deductivas.
  - ✓ *Problema Resuelto # 17:* Traducción al lenguaje natural
  - ✓ *Problema Resuelto # 18:* Tabla de verdad.

- ***Algoritmo y Recursividad***

  - ✓ *Problema Resuelto # 19:* Problema de factorial.
  - ✓ *Problema Resuelto # 20:* Sucesión de Fibonacci.
  - ✓ *Problema Resuelto # 21:* Suma recursiva de los N primeros números naturales.
  - ✓ *Problema Resuelto # 22:* Problema del robot.

2.3.1 Problemas de Búsqueda sin Heurística

PROBLEMA RESUELTO # 1: Torres de Hanói	
Se hallan N discos de distinto tamaño apilados sobre una base A de manera que cada disco se encuentra sobre uno de mayor radio. Existen otras dos bases vacías B y C. El objetivo es llevar todos los discos de la base A hasta la base C, para lo cual puede usarse la base B. Considerar que se puede mover sólo un disco a la vez, y cada disco puede descansar solamente en las bases y no en el suelo. Recordar que los discos deben situarse siempre sobre uno de mayor radio. A continuación se presentará una posible solución seguida de una representación en el Scratch.	
Abstracción	Se tiene N discos apilados en una base A (origen). Se tienen dos bases vacías B (auxiliar) y C (destino) Objetivo: Llevar todos los discos de la base A, a la base C,

<table>
<tr><td></td><td colspan="2">usando la base B como intermedia.</td></tr>
<tr><td>Análisis y representación de datos</td><td>Entrada:
Tres bases para los discos, A, B y C:
Torre A = 3 discos
Torre B = vacía
Torre C = vacía</td><td>Salida:
Torre A = vacía
Torre B = vacía
Torre C = 3 discos</td></tr>
<tr><td>Descomposición del problema</td><td colspan="2">Definir restricciones:
Cantidad de discos N = 3.
Hay que partir de la base origen a la base destino usando la base intermedia.
Sólo se puede mover un disco a la vez.
Un disco nunca puede situarse sobre un disco de menor radio.</td></tr>
<tr><td>Pensamiento algorítmico</td><td colspan="2">(Ver anexo 3)</td></tr>
<tr><td>Reconocimiento y generalización de patrones</td><td colspan="2">Patrón # 1: El disco de mayor radio nunca se sitúa sobre el de menor radio.
Patrón # 2: La torre B se utiliza como intermediaria.</td></tr>
<tr><td>Simulación/Evaluación</td><td colspan="2">La corrida del algoritmo llevado a cabo desde el estado inicial hasta el estado final constituye la simulación.
 
</td></tr>
</table>

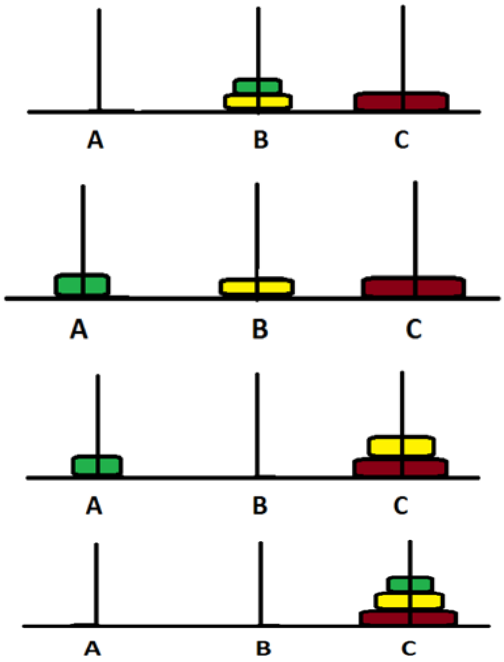
	

Tabla 2.Problema resuelto #1 (Torres de Hanói) [34].

PROBLEMA RESUELTO # 2: Misioneros y caníbales	
Se tienen 3 misioneros y 3 caníbales en la orilla derecha de un río. Existe un bote con capacidad para dos personas como máximo. Se desea que los seis pasen al margen izquierdo del río, pero hay que considerar que no debe haber más caníbales que misioneros en ninguna de las dos orillas del rio porque entonces los caníbales se comen a los misioneros. Además, el bote siempre debe ser conducido por alguien.	
Abstracción	Atravesar los misioneros y caníbales de una orilla del rio a la otra en un bote sin que los caníbales devoren a los misioneros teniendo en cuenta las siguientes restricciones: El bote tiene capacidad para 2 personas, de los cuales 1 se queda en la orilla izquierda del rio a la que se va a atravesar y 1 se queda en el bote para manejarlo. En ninguna orilla debe haber más caníbales que misioneros sino los misioneros serán devorados por los caníbales.
Análisis y representación de datos	Caníbal = C; Misionero = M; Bote = B; Orilla derecha = Od; Orilla izquierda = Oi

	Estado inicial: Cantidad de M Od = 3; Cantidad de C Od = 3 Posición del B = Od; Cantidad de M Oi = 0 Cantidad de C Oi = 0 ***Estado final u objetivo:*** Posición del B = Oi; Cantidad de M Oi = 3; Cantidad de C Oi = 3
Descomposición del problema	Este problema se descompone en la cantidad de viajes que dará el bote para que los misioneros y caníbales lleguen a salvo a la otra orilla del rio.
Pensamiento algorítmico	**(Ver anexo 4)**
Reconocimiento y generalización de patrones	Patrón # 1: cantidad de caníbales <= cantidad de misioneros Patrón # 2: Pasar a los misioneros de una orilla hacia la otra respetando las restricciones y cuando estos estén garantizados entonces pasar a los caníbales.
Simulación/ Evaluación	La corrida del algoritmo realizado desde el estado inicial hasta el estado final constituye la simulación.

Tabla 3. Problema resuelto #2 (Misioneros y caníbales) [35].

PROBLEMA RESUELTO #3: El arriero	
Un arriero se encuentra en el borde de un rio llevando un puma, una cabra y una lechuga. Debe cruzar a la otra orilla por medio de un bote con capacidad para dos (el arriero y alguna de sus pertenencias). La dificultad es que si el puma se queda solo con la cabra la devorará, y lo mismo sucederá si la cabra se queda sola con la lechuga. ¿Cómo cruzar sin perder ninguna pertenencia?	
Abstracción	Atravesar las pertenencias que tiene (puma, lechuga, cabra) de una orilla del rio a la otra en un bote sin que el puma devore la cabra, y sin que la cabra se coma la lechuga. Teniendo en

	cuenta las siguientes restricciones: El bote tiene capacidad para 2 pasajeros, de los cuales se pueden montar el arriero y una de sus pertenencias. En ninguna orilla debe pasar lo siguiente: Si el puma se queda solo con la cabra la devorará. Si la cabra se queda sola con la lechuga, la devorará.
Análisis y representación de datos	Arriero = A; Lechuga = L; Cabra = C; Puma = P; Bote = B ***Estado inicial:*** Cantidad de pertenencias en la Oi = 3 Posición del B = Oi ***Estado final u objetivo deseado:*** Posición del B = Od Cantidad de pertenencias en la Od = 3
Descomposición del problema	Determinar posición del bote, del arriero y de sus pertenencias. Este problema se descompone en la cantidad de viajes que dará el bote para que el arriero y sus pertenencias lleguen a salvo a la otra orilla del rio.
Pensamiento algorítmico	**(Ver anexo 5)**
Reconocimiento y generalización de patrones	El arriero se mantendrá en el bote para trasladar las pertenencias.
Simulación / Evaluación	La corrida del algoritmo realizado desde el estado inicial hasta el estado final constituye la simulación.

Tabla 4. Problema resuelto #3 (El arriero) [36].

PROBLEMA RESUELTO # 4: Jarras de agua (8, 5, 3 litros respectivamente)
Un tonelero quiso repartir entre dos personas, a partes iguales, una jarra con 8 litros de agua, pero al intentar hacer las medidas se vio con el problema de que solamente disponía, aparte de la jarra de 8 litros, de dos jarras con capacidades de 3 y de 5 litros.

Dijo: “No importa. Traspasando adecuadamente el agua, puede hacerse la medición de forma que queden 4 litros en la jarra que ahora contiene 8 y otros cuatro litros en la jarra de capacidad para 5”. ¿Cómo lo va a hacer?	
Abstracción	Se tienen tres jarras, una de 8 litros de capacidad llena de agua, una de 5 litros de capacidad y otra de 3 vacías respectivamente, ninguna de ellas tiene marcas de medición. Averiguar cómo se puede lograr tener exactamente 4 litros de agua en la jarra de 8 litros de capacidad y otros 4 litros de agua en la jarra de 5 litros de capacidad. **Objetivo:** repartir entre dos personas a partes iguales 8 litros de agua.
Análisis y representación de datos	**Estado Inicial:** Jarras de agua $\begin{cases} 8\ litros \\ 5\ litros \\ 3\ litros \end{cases}$ Cantidad de agua disponible = 8 litros **Estado final u objetivo:** Cantidad de agua en la jarra de 8 litros = 4 litros. Cantidad de agua en la jarra de 5 litros = 4 litros. Estado de la jarra de 3 litros = vacía.
Descomposición del problema	Para repartir el agua disponible en la jarra de 8 litros a las dos personas, hay que tener en cuenta que disponemos de dos jarras más (una de 4 y otra de 5 litros), las cuales están vacías. Cada una de las jarras de 5 y 8 litros, debe contener exactamente 4 litros para que la división sea equitativa. Para ello se consideran válidas las siguientes operaciones: Llenar las jarras de agua, Tirar agua de las jarras, Pasar el agua de una jarra a otra.
Pensamiento algorítmico	**(Ver anexo 6)**
Reconocimiento y generalización de	Patrón # 1: Se usa la jarra de 3 litros de intermedio entre las jarras de 5 y 8

patrones	litros en el proceso de llenado y vaciado en las mismas.
Simulación / Evaluación	La corrida del algoritmo realizado desde el estado inicial hasta el estado final constituye la simulación.

Tabla 5. Problema resuelto #4 (Jarras de agua (8, 5, 3 litros respectivamente)).

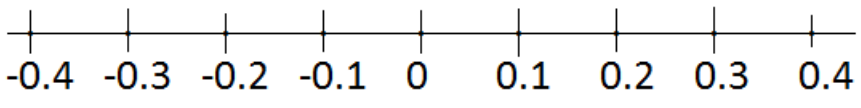
PROBLEMA RESUELTO # 5: Jarras de agua (4 y 3 litros respectivamente)	
Se tienen dos jarras, una de 4 litros de capacidad y otra de 3, sin escala de medición, además, se tiene una bomba que permite llenar las jarras de agua. Se desea tener 2 litros de agua en la jarra de 4 litros de capacidad. Las siguientes operaciones son válidas: llenar las jarras, tirar agua de las jarras, pasar el agua de una jarra a otra.	
Abstracción	Se tienen dos jarras de agua, una para 4 y otra para 3 litros de capacidad sin escala de medición. El objetivo del problema consiste en encontrar una secuencia de movimientos que consiga dejar exactamente 2 litros de agua en la jarra de 4 litros de capacidad, sin importar cuantos litros quedan en la jarra de 3 litros.
Análisis y representación de datos	**Estado inicial:** Cantidad de jarras de agua = 2: $\begin{cases} 4\ litros \\ 3\ litros \end{cases}$ Disponibilidad para llenar las jarras = una bomba **Estado final u objetivo:** Cantidad de agua en la jarra de 4 litros = 2 litros. Estado de la jarra de 3 litros = vacía o con agua, en caso de que no importen los litros de este jarra.
Descomposición del problema	Averiguar cómo se puede lograr tener exactamente 2 litros de agua en la jarra de 4 litros de capacidad sin que esta tenga alguna escala o marca de medición.
Pensamiento algorítmico	***(Anexo 7)***
Reconocimiento y generalización de	Patrón #1: Se usa la jarra de 3 litros para ir llenando y vaciando la de 4

patrones	litros hasta lograr el objetivo que se desea alcanzar.
Simulación / Evaluación	La corrida del algoritmo realizado desde el estado inicial hasta el estado final constituye la simulación.

Tabla 6. Problema resuelto #5 (Jarras de agua (4 y 3 litros respectivamente)) [36].

2.3.2 Problemas de Matemáticas

2.3.2.1 Matemática I

PROBLEMA RESUELTO # 6: Límite de función real de una variable real. Analizar el comportamiento de la siguiente función: $f(x) = 2x * e^{1/x}$ cuando la variable independiente se aproxima a cero.	
Abstracción	Hallar el límite de la función $f(x)$ cuando x tiende a cero: $\lim_{x\to 0}(2x * e^{\frac{1}{x}})$
Descomposición del problema	Teniendo en cuenta que el $\lim_{x\to 0^-}\frac{1}{x} = -\infty$ y el $\lim_{x\to 0^+}\frac{1}{x} = +\infty$, o sea, el comportamiento de $\frac{1}{x}$ es diferente a la derecha y a la izquierda de cero; para resolver el problema en cuestión hay que descomponerlo en los siguientes límites: ¿ $\lim_{x\to 0^-}(2x * e^{\frac{1}{x}})$ ¿ y ¿ $\lim_{x\to 0^+}(2x * e^{\frac{1}{x}})$ ¿
Análisis y representación de datos	Función real de una variable real: $f(x) = 2x * e^{1/x}$ Punto: $x_0 = 0$
Pensamiento algorítmico	***(Ver anexo 8)***
Reconocimiento y generalización de patrones	-
Simulación / Evaluación	

	$\lim_{x\to 0^-} 2x * e^{\frac{1}{x}} = 0$ $\lim_{x\to 0^+} 2x * e^{\frac{1}{x}} = +\infty$ $f(-0.5)$ $f(-0.4)$ $f(-0.3)$ $f(-0.2)$ $f(-0.1)$ ↓ Cuando x tiende a 0^- $f(0.5)$ $f(0.4)$ $f(0.3)$ $f(0.2)$ $f(0.1)$ ↑ Cuando x tiende a 0^+

Tabla 7.Problema Resuelto # 6: Límite de función real de una variable real.

PROBLEMA RESUELTO # 7: Continuidad de una función real de una variable real. Analizar la continuidad de la función: $f(x) = \frac{3x}{5-\lvert x-3\rvert}$	
Abstracción	Para analizar la continuidad de f(x) lo primero es abstraer del problema que esta función es una función real de una variable real definida por tramos o por partes, ya que la expresión modular que aparece en su denominador le da un comportamiento a la izquierda de x=3 y otro a la derecha de x=3.
Descomposición del problema	Para analizar la continuidad de la función dada, se descompone en los siguientes tres análisis: $¿x > 3?$ $¿x < 3?$ $¿x = 3?$
Análisis y representación de datos	$f(x) = \frac{3x}{5-\lvert x-3\rvert} = \begin{cases} \frac{3}{8-x}, & x \geq 3 \\ \frac{3x}{x+2}, & x < 3 \end{cases}$
Pensamiento algorítmico	**Paso 1:**

	$$\left[\begin{array}{l} \textbf{paso 2}: \left[\begin{array}{l} \textbf{paso 3}: \left[\begin{array}{c} if\ x = 3\ \in dom\ f \\ then \\ if\ f(3) = \lim\limits_{x\to 3} f = L \\ then \\ return\ (f(x) es\ continua\ en\ x = 3). \\ else \\ return\ (f(x) es\ descontinua\ finia\ evitable\ en\ x = 3) \end{array}\right] \\ else\ if\ \lim\limits_{x\to 3^-} f = L_1\ \ y\ \ \lim\limits_{x\to 3^+} f = L_2\ \ y\ \ L_1 \neq L_2 \\ then \\ return\ (f(x) es\ discontinua\ finita\ no\ evitable). \\ if\ \lim\limits_{x\to 3^-} f = \pm\infty\ \ o\ \ \lim\limits_{x\to 3^+} f = \pm\infty \\ return\ (f(x) presenta\ una\ discontinuidad\ infinita\ en\ x = 3). \end{array}\right] \\ else \\ return\ (f(x) presenta\ una\ discontinuidad\ infinita\ en\ x = 3) \end{array}\right]$$
Reconocimiento y generalización de patrones	-
Simulación / Evaluación	***paso* 1**: $x = 3 \in dom\ f \qquad f(3) = 9$ ***paso* 2**: $\lim\limits_{x\to 3^-} \dfrac{3x}{x+2} = \dfrac{9}{6} \qquad \lim\limits_{x\to 3^+} \dfrac{3x}{8-x} = \dfrac{9}{5}$ $\therefore \lim\limits_{x\to 3^-} f(x) = \lim\limits_{x\to 3^+} f(x) = \dfrac{9}{5}$ ***paso* 3**:

	$\therefore f(x) es\ continua\ en\ x = 3$ $Respuesta: f(x) es\ continua\ en\ R, excepto\ en\ x = 2\ y\ x = 8\ donde\ presenta\ una\ discontinuidad\ infinita$

Tabla 8. Problema resuelto #7 (Continuidad de una función real de una variable real).

PROBLEMA RESUELTO # 8: Derivada n-ésima de una función real de una variable real. Determinar la derivada n-ésima (orden n) de la siguiente función: $f(x) = \frac{1}{x}$	
Abstracción	Para determinar la derivada de orden n de la función $f(x)$ hay que determinar, necesariamente, la primera derivada de la función ($f'(x)$) la segunda derivada ($f''(x)$), la tercera derivada ($f'''(x)$) y otras derivadas sucesivas hasta encontrar el patrón para la derivada n-ésima.
Descomposición del problema	Determinar las derivadas sucesivas siguientes: 1) De orden 1 (Primera derivada): $f'(x) = (-\frac{1}{x^2})$ 2) De orden 2 (Segunda derivada): $f''(x) = \left(\frac{-1}{x^2}\right)' = \frac{2}{x^3}$ 3) De orden 3 (Tercera derivada): $f'''(x) = \left(\frac{2}{x^3}\right)' = \frac{-6}{x^4}$
Análisis y representación de datos	Función real de una variable real: $f(x) = \frac{1}{x}$
Pensamiento algorítmico	**Paso 1:** $f'(x) = (-\frac{1}{x^2})$ **Paso 2:** $f''(x) = \left(\frac{-1}{x^2}\right)' = \frac{2}{x^3}$

	Paso 3: $f'''(x) = \left(\frac{2}{x^3}\right)' = \frac{-6}{x^4}$ **Paso 4:** $f''''(x) = \frac{24}{x^5}$
Reconocimiento y generalización de patrones	***Patrón # 1:*** (n=1) $f'(x) = -\frac{1}{x^2} \rightarrow x^{n+1}$ (n=2) $f''(x) = \frac{2}{x^3} \rightarrow x^{n+2}$ (n=3) $f'''(x) = \frac{-6}{x^4} \rightarrow x^{n+3}$ (n=4) $f''''(x) = \frac{24}{x^5} \rightarrow x^{n+4}$ $denominador = x^{n+1}$ ***Patrón # 2:*** $(-1)^n \begin{Bmatrix} si\ n\ es\ impar\ (-) \\ si\ n\ es\ par\ (+) \end{Bmatrix}$ ***Patrón # 3:*** (n=1) → numerador = 1 = 1 (n=2) → numerador = 2 = 2*1 (n=3) → numerador = 3 = 3*2*1 (n=4) → numerador = 4 = 4*3*2*1 *numerador = n!* ***Generalización de patrones:*** $f^n(x) = \frac{(-1)^n * n!}{x^{n+1}}$
Simulación / Evaluación	Se calculan derivadas de cualquier orden superior evaluando en la fórmula generalizada: $f^n(x) = \frac{(-1)^n * n!}{x^{n+1}}$

	N = 7 $f^{(7)} = \frac{(-1)^7 * 7!}{x^8}$ N = 8 $f^8(x) = \frac{(-1)^8 * 8!}{x^9}$ N = 9 $f^9(x) = \frac{(-1)^9 * 9!}{x^{10}}$ N = 10 $f^{10}(x) = \frac{(-1)^{10} * 10!}{x^{11}}$

Tabla 9. Problema resuelto #8 (Derivada n-ésima de una función real de una variable real).

PROBLEMA RESUELTO # 9: Derivada n-ésima de una función real de una variable real. Determinar la derivada n-ésima (orden n) de la siguiente función: $f(x) = x * e^{-x}$	
Abstracción	Para determinar la derivada de orden n de la función $f(x)$ hay que determinar, necesariamente, la primera derivada de la función ($f'(x)$) la segunda derivada ($f''(x)$), la tercera derivada ($f'''(x)$) y otras derivadas sucesivas hasta encontrar el patrón para la derivada n-ésima.
Descomposición del problema	Determinar las derivadas sucesivas siguientes: 1) De orden 1 (Primera derivada): $f'(x) = e^{-x} - x * e^{-x}$ 2) De orden 2 (Segunda derivada): $f''(x) = -2e^{-x} + x * e^{-x}$ 3) De orden 3 (Tercera derivada): $f'''(x) = 3e^{-x} - x * e^{-x}$
Análisis y representación de datos	Función real de una variable real: $f(x) = x * e^{-x}$
Pensamiento algorítmico	**Paso 1:** $f'(x) = e^{-x} - x * e^{-x}$ **Paso 2:** $f''(x) = -2e^{-x} + x * e^{-x}$

	Paso 3: $f'''(x) = 3e^{-x} - x * e^{-x}$
Reconocimiento y generalización de patrones	***Patrón # 1: Primer término*** (n=1): e^{-x} (n=2): $-2e^{-x}$ (n=3): $3e^{-x}$ $Por\ tanto: (-1)^{n+1} * n * e^{-x}$ ***Patrón # 2: Segundo término*** (n=1): $-x * e^{-x}$ (n=2): $x * e^{-x}$ (n=3): $-x * e^{-x}$ $Por\ tanto: (-1)^{n} * x * e^{-x}$ ***Generalización de patrones:*** $f^n(x) = (-1)^{n+1} * n * e^{-x} + (-1)^n * x * e^{-x}$
Simulación / Evaluación	Se calculan derivadas de cualquier orden superior evaluando en la fórmula generalizada: $f^n(x) = (-1)^{n+1} * n * e^{-x} + (-1)^n * x * e^{-x}$ N = 7 $f^{(7)} = (-1)^8 * 7 * e^{-x} + (-1)^7 * x * e^{-x}$ N = 8 $f^8(x) = (-1)^9 * 8 * e^{-x} + (-1)^8 * x * e^{-x}$ N = 9 $f^9(x) = (-1)^{10} * 9 * e^{-x} + (-1)^9 * x * e^{-x}$ N = 10 $f^{10}(x) = (-1)^{11} * 10 * e^{-x} + (-1)^{10} * x * e^{-x}$

Tabla 10. Problema resuelto #9 (Derivada n-ésima de una función real de una variable real).

PROBLEMA RESUELTO # 10: Problema de optimización de una función real de una variable real.

Una cerca de 8 piezas de altura corre paralela a un edificio alto a una distancia de 4 pies de éste. ¿Cuál es la longitud de la escalera más corta que llegará desde el suelo, pasando

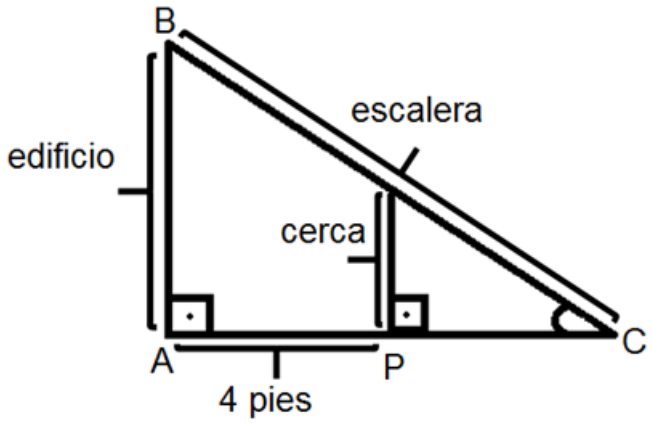
por encima de la cerca hasta la pared del edificio?	
Abstracción	 Se tiene que: $\Delta ABC \sim \Delta PQC$ por el teorema a*a
Descomposición del problema	1) Determinar la función a optimizar. 2) Trabajar con las restricciones del problema hasta encontrar relaciones entre las variables. 3) Plantear la función objetivo como una función real de una variable real. 4) Aplicar la teoría de extremos globales a la función objetivo planteado.
Análisis y representación de datos	$\overline{AP} = 4pies$ $\overline{PQ} = 8\,pies$ $\measuredangle BCA = 90°$ $\measuredangle QPC = 90°$ ¿BC?
Pensamiento algorítmico	**(ver anexo 9)**
Reconocimiento y generalización de patrones	-
Simulación / Evaluación	-

Tabla 11. Problema resuelto # 10 (Optimización de una función real de una variable real).

2.3.2.2 Matemática II

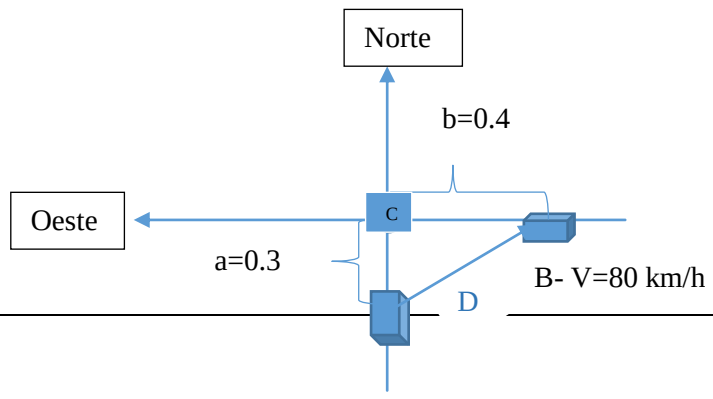
PROBLEMA RESUELTO # 11: Interpretación geométrica de la derivada parcial. Determinar la pendiente de la tangente a la curva de intersección de la superficie $6yz - 2xz + 5xy = 0$ con el plano x=2 en el punto P$(2; 3; -\frac{15}{7})$	
Abstracción	La intersección entre una superficie $z = f(x, y)$ y un plano $x = x_0$ es una curva C y la pendiente (m) de la recta tangente (T) a una curva en un punto (p) se determina por la derivada de C evaluada en el punto, por tanto, $m_T \ a\ C\ en\ el\ punto\ p \Rightarrow \frac{\partial z}{\partial y}(p)$
Descomposición del problema	1) Determinar $z = f(x, y)$ 2) Determinar $\frac{\partial z}{\partial y}$ 3) Evaluar $\frac{\partial z}{\partial y}$ en p
Análisis y representación de datos	$Superficie: 6yz - 2xz + 5xy = 0$ $z = \frac{5xy}{2x-6y}$ $Plano\ x = 2$ $Punto\ P(2; 3; -\frac{15}{7})$
Pensamiento algorítmico	***Paso 1:*** Determinar $z = f(x, y) = \frac{5xy}{2x-6y}$. ***Paso 2:*** Determinar $\frac{\partial z}{\partial y} = \frac{10x^2}{(2x-6y)^2}$. ***Paso 3:***

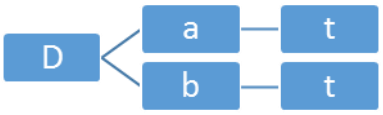
	Evaluar $\frac{\partial z}{\partial y}$ en el punto P$(2; 3; -\frac{15}{7})$. $\frac{10x^2}{(2x-6y)^2}_{\|(2;3;-\frac{15}{7})} = \frac{10}{49}$ ***Paso 4:*** Emitir respuesta literal, la pendiente de la recta tangente a la curva de intersección entre la superficie y el plano dados en el punto P es $\frac{10}{49}$
Reconocimiento y generalización de patrones	**Patrón #1:** La intersección de una superficie $z = f(x, y)$ con un plano $x = x_0$ o $y = y_0$ es una función real de una variable real. **Patrón #2:** Al determinar una derivada parcial se considera como variable con respecto a la cual se está derivando y las otras que puedan aparecer se consideran como constantes en la derivación.
Simulación / Evaluación	-

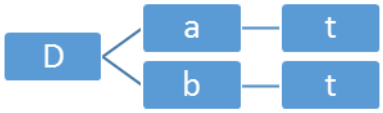
Tabla 12. Problema resuelto #11 (Interpretación geométrica de la derivada parcial).

PROBLEMA RESUELTO # 12: Interpretación física de la derivada parcial

El auto A se desplaza hacia el norte por la carretera 16 y el auto B viaja hacia el oeste por la carretera 83; cada uno se aproxima al cruce de estas carreteras En cierto momento, el auto A está a 0.3 km del cruce y viaja a 90 km/h mientras que el auto B está a 0.4 km del cruce y viaja a 80 km/h ¿Cuál es la razón de cambio de la distancia entre los autos en ese momento?

Abstracción	

	A- V=90 km/h Como en el cruce de las carreteras se forma un ángulo de 90^0 se obtiene un triángulo rectángulo entre el auto A el cruce C y el auto B, entonces la distancia D (hipotenusa del triángulo ACB) se plantea utilizando en Teorema de Pitágoras y depende de las posiciones a y b de los autos en cada momento quedando: $D(a;b) = \sqrt{a^2 + b^2}$ … De esta forma D es una función que depende de las posiciones a y b las cuales a su vez cambian en el tiempo, por tanto el diagrama de dependencia entre estas variables queda como sigue:  Se conoce que la velocidad instantánea de cada auto se puede plantear por las derivadas siguientes $\frac{da}{dt} = 90\ km/h;\ \ \frac{db}{dt} = 80\ km/h$ que son datos, pero como a y b también dependen del tiempo entonces para determinar la razón de cambio de la distancia D entre los autos ($\frac{\partial D}{\partial t} = \frac{\partial D}{\partial a} \cdot \frac{da}{dt} + \frac{\partial D}{\partial b} \cdot \frac{db}{dt}$) es necesario descomponer el problema.
Descomposición del problema	1) Determinar $\frac{\partial D}{\partial a}$ 2) Evaluar $\frac{\partial D}{\partial a}$ en (a=0.3 y b=0.4) 3) Determinar $\frac{\partial D}{\partial b}$ 4) Evaluar $\frac{\partial D}{\partial b}$ en (a=0.3 y b=0.4)
Análisis y representación de	La velocidad de los autos cuando se encuentran en la posición

datos	$a = 0.3\ km\ \ y\ b = 0.4\ km$ del cruce se representa respectivamente por las derivadas siguientes: $\frac{da}{dt} = 90\ km/h;\ \ \frac{db}{dt} = 80\ km/h$
Pensamiento algorítmico	***Paso 1:*** Plantear la función de distancia $D(a;b) = \sqrt{a^2 + b^2}$ ***Paso 2:*** Plantear el diagrama de dependencia entre las variables (diagrama de árbol)  ***Paso 3:*** Reconocer que la razón de cambio de la distancia entre los autos ($\frac{\partial D}{\partial t}$) se determina por $\frac{\partial D}{\partial t} = \frac{\partial D}{\partial a} \cdot \frac{da}{dt} + \frac{\partial D}{\partial b} \cdot \frac{db}{dt}$ (Fórmula de la regla de la cadena para este problema) ***Paso 4:*** Reconocer que son datos las siguientes derivadas $\frac{da}{dt} = 90\ km/h;\ \ \frac{db}{dt} = 80\ km/h$ ***Paso 5:*** Determinar $\frac{\partial D}{\partial a} = \frac{a}{\sqrt{a^2+b^2}}$ ***Paso 6:*** Evaluar $\frac{\partial D}{\partial a}$ en (a=0.3 y b=0.4) $\frac{\partial D}{\partial a} = \frac{0.3}{\sqrt{(0.3)^2 + (0.4)^2}} = 0.91$ ***Paso 7:*** Determinar $\frac{\partial D}{\partial b} = \frac{b}{\sqrt{a^2+b^2}}$ ***Paso 8:***

	Evaluar $\frac{\partial D}{\partial b}$ en (a=0.3 y b=0.4) $$\frac{\partial D}{\partial b} = \frac{0.4}{\sqrt{(0.3)^2 + (0.4)^2}} = 1.21$$ ***Paso 9:*** Calcular $\frac{\partial D}{\partial t}$ sustituyendo todos los resultados anteriores en la fórmula de la regla de la cadena expresada anteriormente. $$\frac{\partial D}{\partial t} = 0.91 \cdot 90 + 1.21 \cdot 80 = 178.7$$ ***Paso 10:*** Emitir respuesta literal, la razón de cambio de la distancia entre los autos es 178.7 km/h.
Reconocimiento y generalización de patrones	***Patrón 1:*** La razón de cambio de “a” en el tiempo “t” se plantea como: $$\frac{da}{dt} = 90\ km/h$$ ***Patrón 2:*** La razón de cambio de “b” en el tiempo “t” se plantea como: $$\frac{db}{dt} = 80\ km/h$$ ***Patrón 3:*** La razón de cambio de “D” respecto a las posiciones “a” y “b” de los autos se plantean: $\frac{\partial D}{\partial a}$ y $\frac{\partial D}{\partial b}$ ***Patrón 4:*** La razón de cambio de “D” en el tiempo “t” se plantea como: $$\frac{\partial D}{\partial t}$$ **Generalización:** La razón de cambio $\frac{\partial D}{\partial t}$ se determina como

	sigue: $\frac{\partial D}{\partial t} = \frac{\partial D}{\partial a} \cdot \frac{da}{dt} + \frac{\partial D}{\partial b} \cdot \frac{db}{dt}$
Simulación / Evaluación	-

Tabla 13. Problema resuelto #12 (Interpretación física de la derivada parcial).

PROBLEMA RESUELTO # 13: Derivada direccional de una función real de varias variables. Determinar la derivada de la función $f(x,y) = 1 + 2x\sqrt{y}$ en el punto P (3; 4) en la dirección del vector $\vec{v} = (4; -3)$.	
Abstracción	Lo que se pide es la derivada direccional de f según el vector $\vec{v}$ en el punto P. Se conoce que puede ser determinado por la fórmula siguiente: $D_u f(3;4) = \overrightarrow{\nabla f}\,(3;4) \cdot \vec{u}$, donde $D_u f(3;4)$ – Derivada direccional de f en el punto (3;4) $\overrightarrow{\nabla f}\,(3;4)$ - Gradiente de f en el punto (3;4) $\vec{u}$ – Vector unitario
Descomposición del problema	1) Determinar el gradiente $\overrightarrow{\nabla f} = < \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} >$ 2) Evaluar el gradiente $\overrightarrow{\nabla f}$ en el punto (3;4) 3) Determinar el vector unitario $\vec{u} = \frac{\vec{v}}{\lvert\vec{v}\rvert}$
Análisis y representación de datos	• Una función de dos variables independientes “x” y “y” $f(x,y) = 1 + 2x\sqrt{y}$ • Un punto P(3;4) • La dirección de un vector $\vec{v} = (4, -3)$
Pensamiento algorítmico	***Paso 1:*** Determinar $\frac{\partial f}{\partial x} = 2\sqrt{y}$ ***Paso 2:***

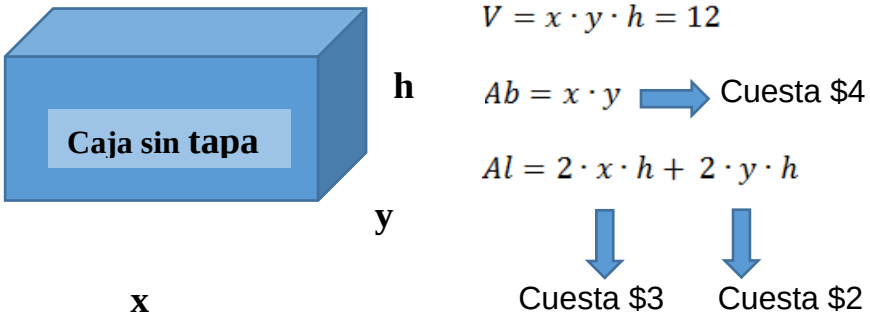
	Determinar $\frac{\partial f}{\partial y} = \frac{x}{\sqrt{y}}$ ***Paso 3:*** Plantear el gradiente $\overrightarrow{\nabla f} = \langle 2\sqrt{y}, x/\sqrt{y} \rangle$ ***Paso 4:*** Evaluar el gradiente $\overrightarrow{\nabla f}$ en el punto (3;4) $\overrightarrow{\nabla f}_{(3;4)} =< 2\sqrt{4}, \frac{3}{\sqrt{4}} >= \langle 4, 3/2 \rangle$ ***Paso 5:*** Determinar $\lvert \vec{v} \rvert = \sqrt{4^2 + (-3)^2} = \sqrt{16+9} = 5$ ***Paso 6:*** Calcular $\vec{u} = \frac{\vec{v}}{\lvert \vec{v} \rvert} = \frac{(4,-3)}{5} = (\frac{4}{5}, \frac{-3}{5})$ ***Paso 7:*** Calcular $D_u f(3;4)$ sustituyendo todos los resultados anteriores en la fórmula $D_u f(3;4) = \overrightarrow{\nabla f}\,(3;4) \cdot \vec{u} = \langle 4, 3/2 \rangle \cdot \langle 4/5\,, -3/5 \rangle = \frac{16}{5} + \frac{-9}{10} = \frac{23}{10}$ ***Paso 8:*** Emitir respuesta literal, la derivada de f en el punto P según el vector $\vec{v}$ es $\frac{23}{10}$
Reconocimiento y generalización de patrones	-
Simulación / Evaluación	-

Tabla 14. Problema resuelto #13 (Derivada direccional de una función real de varias variables).

PROBLEMA RESUELTO # 14: Extremos condicionados de una función real de varias

variables.

Se desea construir una caja de base rectangular (sin tapa) que tenga un volumen de 12 cm^3; el costo por cm^2 del material que se usará para la base es de 4 pesos, el que se usará para dos de los lados opuestos es de 3 pesos y el que se usará para los otros dos lados opuestos es de 2 pesos. Determinar las dimensiones de la caja para que el costo sea mínimo.

Abstracción	 Reconocer que es un problema de optimización con restricción. Minimizar una función de costo sujeta a la restricción $V = x \cdot y \cdot h = 12$. Método de resolución: "Método de los Multiplicadores de Lagrange (M.M.L)"
Descomposición del problema	5) Determinar la función objetivo y la restricción. 6) Aplicar el Método de los Multiplicadores de Lagrange.
Análisis y representación de datos	• V=12 cm^3 • Costo del material de la base: 4 • Costo del material de par de lados opuestos: 3 • Costo del material del otro par de lados opuestos: 2
Pensamiento algorítmico	***Paso 1:*** Determinar a partir de un modelo matemático lo que se quiere optimizar (Función Objetivo). $Al(x, y, h) = Ab + 2 \cdot xh + 2 \cdot yh$ Minimizar: $C(x, y, h) = 4 \cdot Ab + 3 \cdot 2 \cdot xh + 2 \cdot 2 \cdot yh$ FUNCIÓN OBJETIVO: $C(x, y, h) = 4 \cdot Ab + 6 \cdot xh + 4 \cdot yh$

Paso 2:

Plantear la ecuación de enlace (la restricción igualada a cero):

$$V(x,y,h) = x \cdot y \cdot h - 12 = 0$$

Paso 3:

Plantear la función auxiliar de Lagrange (se introduce un multiplicador de Lagange por cada restricción del problema):

$$\varphi(x,y,h,\lambda) = C(x,y,h) - \lambda \cdot V(x,y,h)$$

$$\varphi(x,y,h,\lambda) = 4 \cdot Ab + 6 \cdot xh + 4 \cdot yh - \lambda \cdot (x \cdot y \cdot h - 12)$$

Paso 4:

Plantear el sistema de ecuaciones correspondiente:

$$\begin{cases} \frac{\partial \varphi}{\partial x} = 0 \quad (I) \\ \frac{\partial \varphi}{\partial y} = 0 \quad (II) \\ \frac{\partial \varphi}{\partial h} = 0 \quad (III) \\ \frac{\partial \varphi}{\partial \lambda} = 0 \quad (IV) \end{cases} \qquad \begin{cases} \frac{\partial \varphi}{\partial x} = 4y + 6h - \lambda yh = 0 \quad (I) \\ \frac{\partial \varphi}{\partial y} = 4x + 4h - \lambda xh = 0 \quad (II) \\ \frac{\partial \varphi}{\partial h} = 6x + 4y - \lambda xy = 0 \quad (III) \\ \frac{\partial \varphi}{\partial \lambda} = 12 - xyh = 0 \qquad (IV) \end{cases}$$

Paso 5:

Resolver el sistema de ecuaciones planteado, sus soluciones son los posibles puntos de extremo condicionado.

Paso 5.1:

Despejar λ en cada ecuación del sistema

Despejando $\lambda\ en\ (I)$: $\lambda = \frac{4y+6h}{yh}$

Despejando $\lambda\ en\ (II)$: $\lambda = \frac{4x+4h}{xh}$

Despejando $\lambda\ en\ (III)$: $\lambda = \frac{4y+6x}{xy}$

Paso 5.2:

Igualar estas expresiones hasta obtener relaciones entre las variables hasta poner todas en función de una, de forma que al sustituir estas relaciones en la última ecuación del sistema se

	obtiene una ecuación de una sola variable

$\frac{4y+6h}{yh}=\frac{4x+4h}{xh}$ $xh(4y+6h)=yh(4x+4h)$ $\vdots$ $6x=4y$ $x=\frac{2y}{3}$	$\frac{4x+4h}{xh}=\frac{4y+6x}{xy}$ $xh(4y+6h)=xy(4x+4h)$ $\vdots$ $4y=6h$ $h=\frac{2y}{3}$

Sustituyendo $x=\frac{2y}{3}$ y $h=\frac{2y}{3}$ en (IV):

$$12-\frac{2y}{3}\cdot y\cdot\frac{2y}{3}=0$$

$$12-\frac{2y}{3}\cdot y\cdot\frac{2y}{3}=0$$

$$12-\frac{2y}{3}\cdot y\cdot\frac{2y}{3}=0$$

Paso 5.3:

Resolver la ecuación obtenida

$12=\frac{4}{9}\cdot y^3$ $27=y^3$ $\sqrt[3]{27}=y$ $y=3$

Paso 5.4:

Determinar el valor de cada una de las variables del sistema, excepto de λ

Como $x=\frac{2y}{3}=h$ entonces $x=h=2$

Paso 6:

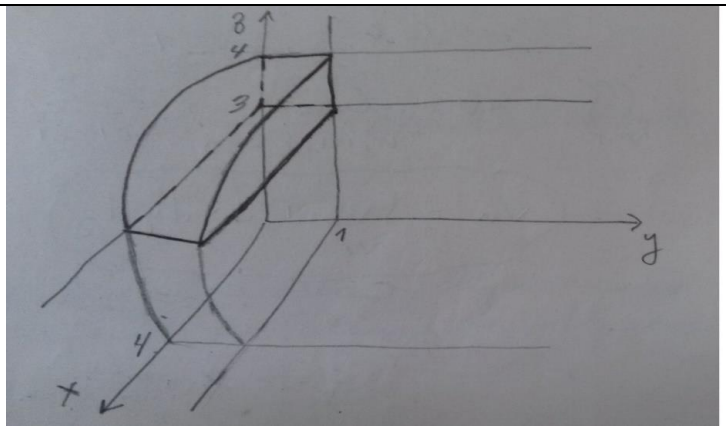
Emitir respuesta literal al problema.

Las dimensiones que minimizan el costo de la caja son las siguientes: Base de la caja 2 x 3 cm y altura 2 cm.

Reconocimiento y generalización de patrones	-
Simulación / Evaluación	-

Tabla 15. Problema resuelto #14 (Extremos condicionados de una función real de varias variables).

<table>
<tr><td colspan="2">PROBLEMA RESUELTO # 15: Volumen de un sólido.
Calcular el volumen del sólido siguiente:
$W = \left\{(x,y,z) \in \mathbb{R}^3 : 0 \leq x \leq \sqrt{16-z^2},\ z \geq 3,\ 0 \leq y \leq 1\right\}$</td></tr>
<tr><td>Abstracción</td><td>Reconocer que para determinar el volumen de un sólido se utiliza un objeto matemático que es la integral triple calculada sobre el sólido en cuestión y tomando 1 como integrando. Por tanto,
$V = \iiint_W 1\, dv$</td></tr>
<tr><td>Descomposición del problema</td><td>1) Representar el sólido W
2) Determinar la variable de la primera integración y la proyección S.
3) Plantear y calcular la primera integración.
4) Representar la proyección S
5) Calcular la integral doble resultante.</td></tr>
<tr><td>Análisis y representación de datos</td><td>• x=0: Plano YZ
• $x = \sqrt{16-z^2}$ $x^2 + z^2 = 16$: Cilindro circular de radio 4 y eje principal sobre el “eje y”
• z=3: Plano paralelo a XY que corta al “eje z” en 3
• y=0: Plano XZ
• y=1: Plano paralelo a XZ que corta al “eje y” en 1</td></tr>
<tr><td>Pensamiento algorítmico</td><td>Paso 1:
Representar el sólido</td></tr>
</table>



Paso 2:

Determinar la variable de la primera integración y la proyección *S*. Tomando a “x” como la variable de la primera integración la cual varía como sigue: $0 \leq x \leq \sqrt{16 - z^2}$ entonces la proyección *S* queda sobre el plano YZ.

Paso 3:

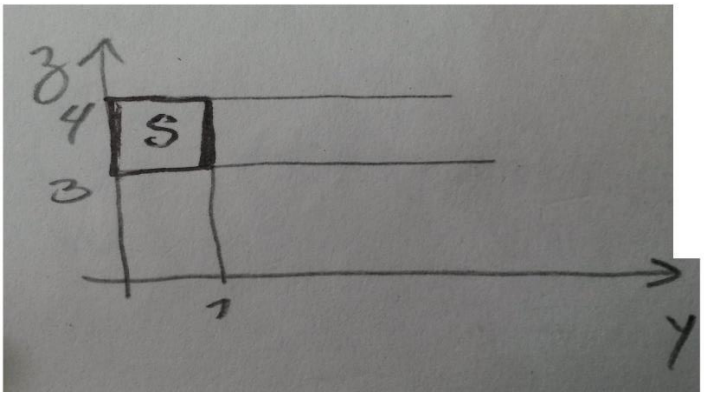
Plantear y calcular la primera integración.

$$V = \iiint_W 1\, dv = \iint_S \left(\int_0^{\sqrt{16-z^2}} 1\, dx \right) dydz$$

$$V = \iiint_W 1\, dv = \iint_S \left(\sqrt{16 - z^2} \right) dydz$$

Paso 4:

Representar la proyección S sobre el plano YZ



Paso 5:

Calcular la integral doble resultante.

	$$\iint_S \left(\sqrt{16-z^2}\right) dydz = \int_3^4 \int_0^1 \sqrt{16-z^2}\, dydz$$ $$\int_3^4 \int_0^1 \sqrt{16-z^2}\, dydz = \int_3^4 \sqrt{16-z^2}\, dz$$ $$= \left[\frac{1}{2}(z \cdot \sqrt{16-z^2} + 16 \cdot \sin^{-1}(\frac{z}{4}))\right]\Big|_3^4$$ $$= \frac{1}{2}\left(4 \cdot \sqrt{16-4^2} + 16 \cdot \sin^{-1}\left(\frac{4}{4}\right)\right) - \frac{1}{2}(3 \cdot \sqrt{16-3^2} + 16 \cdot \sin^{-1}(\frac{3}{4}))=$$ 1.79 ***Paso 6:*** Emitir respuesta literal al problema. El volumen del sólido W es $1.79\ u^3$
Reconocimiento y generalización de patrones	Elección de la primera variable, que es la que varía entre cuerpos o planos. La variable que varía entre constantes queda en la proyección. La generalización sería la integral de $\sqrt{16-z^2} = \left[\frac{1}{2}\left(z(16-z^2) + 16.\sin^{-1}\left(\frac{z}{4}\right)\right)\right]$
Simulación / Evaluación	Variar el tamaño del sólido w y recalcular su volumen (viene de los últimos 3 pasos del algoritmo)

Tabla 16. Problema resuelto #15 (Volumen de un sólido).

2.3.3 Problemas de Matemática Discreta

2.3.3.1 Lógica Proposicional

PROBLEMA RESUELTO # 16. Representación de estructuras deductivas

Represente la estructura deductiva del siguiente razonamiento:

No me preparé correctamente para la CP. No. 1. No me preparé correctamente para la CP. No. 2. Si no me preparo correctamente para las clases prácticas entonces la profesora nota mis dificultades al resolver los ejercicios. Si tengo dificultades al resolver los ejercicios

la profesora me evalúa de 2 en la clase práctica correspondiente. Si me evalúan de mal en varias clases prácticas, tendré un criterio de mal en el corte 1. Con un criterio de mal en el corte 1 tengo probabilidades de fallar la asignatura. La profesora nota mis dificultades al resolver los ejercicios.	
Abstracción	Para representar una estructura deductiva, hay que formar dos sucesiones correctamente formada: ✓ Una sucesión de premisas P_1, P_2,…, P_n. ✓ Una sucesión de conclusiones Q_1, Q_2, …., Q_m Extraer del problema original los elementos que pueden constituir premisas y conclusiones basándose en la definición y significado de premisas y conclusiones.
Descomposición del problema	De dos sucesiones formadas, separar las premisas de las conclusiones. *Premisas:* P1: no me preparé para la clase práctica número 1; P2: no me preparé para la clase práctica número 2; P3: si no me preparo para las clases prácticas entonces tengo dificultades al resolver los ejercicios; P4: si tengo dificultades al resolver ejercicios, entonces me evalúo de 2; P5: si me evalúan de 2 en varias clases prácticas, entonces tendré un criterio de mal en el corte 1. P6: si tengo un criterio de mal en el corte 1, tengo probabilidades de fallar la asignatura.

	Conclusiones: Q1: tengo dificultades en resolver ejercicios.
Análisis y representación de datos	**Entrada:** Un razonamiento en lenguaje natural del que se debe representar su estructura deductiva, teniendo en cuenta las reglas del cálculo proposicional. **Salida:** La estructura deductiva representada según las reglas del cálculo proposicional.
Pensamiento algorítmico	**(Ver anexo 8)**
Reconocimiento y generalización de patrones	***Patrón # 1:*** Para representar un razonamiento o estructura deductiva, hay que formas dos grupos de sucesiones: 1) Premisas 2) Conclusiones ***Patrón # 2:*** las proposiciones atómicas se escriben con letras del abecedario y todas en minúsculas, partiendo de la letra p.
Simulación / Evaluación	-

Tabla 17. Problema resuelto #16 (Representación de estructuras deductivas) [37].

PROBLEMA RESUELTO # 17 Traducción al lenguaje natural	
Formule la siguiente expresión simbólica con palabras utilizando: p = hoy es lunes q = está lloviendo r = hace calor $(p \wedge (q \vee r)) \rightarrow (r \vee (q \wedge p))$	
Abstracción	Traducir la formula proposicional $(p \wedge (q \vee r)) \rightarrow (r \vee (q \wedge p))$, al lenguaje natural utilizando las proposiciones:

	p = hoy es lunes q = está lloviendo r = hace calor
Descomposición del problema	Para traducir la formula dada al lenguaje español, hay que tener en cuenta lo siguiente: 1) Descomponer dicha fórmula aplicando la propiedad distributiva. 2) Traducir la formula resultante al lenguaje español utilizando las proposiciones dadas.
Análisis y representación de datos	**Entrada:** *Proposición:* p = hoy es lunes q = está lloviendo r = hace calor *Fórmula proposicional:* $(p \wedge (q \vee r)) \rightarrow (r \vee (q \wedge p))$ **Salida:** La fórmula proposicional traducida al lenguaje español.
Pensamiento algorítmico	***Paso 1:*** Para formular en palabras la expresión dada, hay que descomponer la fórmula proposicional aplicando la propiedad distributiva $(p \wedge (q \vee r)) \rightarrow (r \vee (q \wedge p))$ $(p \wedge q) \vee (p \wedge r) \rightarrow (r \vee q) \vee (r \vee p)$

	Paso 2: Si la fórmula proposicional ya está disuelta y por lo tanto más cómoda de trabajar, entonces traducirla en palabras, en función de las siguientes proposiciones dadas: p = hoy es lunes q = está lloviendo r = hace calor (p ∧ q) ∨ (p ∧ r)→(r ∨ q) ∨ (r ∨ p) = si hoy es lunes y está lloviendo o hoy es lunes y hace calor, entonces hace calor o está lloviendo, o hace calor u hoy es lunes.
Reconocimiento y generalización de patrones	***Patrón # 1:*** Todas las proposiciones están afirmativas. ***Patrón # 2:*** Las proposiciones atómicas se escriben con letras del abecedario y todas en minúsculas, partiendo de la letra p.
Simulación / Evaluación	-

Tabla 18. Problema resuelto #17 (Traducción al lenguaje natural) [38]

PROBLEMA RESUELTO # 18 Tabla de verdad Confeccione la tabla de verdad correspondiente a la siguiente proposición compuesta: (p Λ q) V (p Λ r)	
Abstracción	Para confeccionar la tabla de verdad de cualquier posición, hay que tener en cuenta: ➢ Si una fórmula tiene **n** variables proposicionales entonces su tabla de verdad tendrá 2^n filas, ➢ Si A es una tautología (cuando el resultado de la tabla de verdad es cero) entonces ~A es una contradicción y viceversa. ➢ Si A es una contingencia (cuando el resultado de la tabla es cero y uno) entonces ~A es otra contingencia. ➢ Como consecuencia de la definición de deducción correcta

<table>
<tr><td></td><td>semánticamente se tiene que:
Una deducción
$P_1,P_2,\ldots, P_n \Rightarrow Q$
Es semánticamente correcta sí y solo sí
$P_1 \wedge P_2 \wedge \ldots \wedge P_n \rightarrow Q$ es una tautología (cuando el resultado de la fórmula es 1).</td></tr>
<tr><td>Descomposición del problema</td><td>1) Atendiendo a que hay tres variables proposicionales, entonces la tabla de verdad tendrá $2^3 = 8$ filas.
2) Teniendo en cuenta que la proposición es compuesta, hay que hacer una tabla de verdad para la conjunción y otra para la disyunción.
3) En la conjunción, la proposición $p \wedge q$ es verdadera si y solo si p y q son verdaderas.
4) En la disyunción, p V q se utiliza en el sentido inclusivo; es decir, p V q es verdadera si p o q o ambas son verdaderas y p V q es falsa solo si las p y q son falsas.</td></tr>
<tr><td>Análisis y representación de datos</td><td>Entrada:
Proposición compuesta = $(p \wedge q) V (p \wedge r)$.
Salida:
Tabla de verdad resultante de proposición compuesta dada.</td></tr>
<tr><td>Pensamiento algorítmico</td><td>Paso 1:
Teniendo en cuenta que la proposición es compuesta, y en ella aparecen una conjunción y una disyunción, hacer una tabla de verdad para cada una de ellas.
<table><tr><td>P</td><td>q</td><td>p v q</td></tr><tr><td>1</td><td>1</td><td>V</td></tr><tr><td>1</td><td>0</td><td>V</td></tr></table></td></tr>
</table>

Conjunción

0	1	V
0	0	F

Disyunción

p	q	p ∧ q
1	1	V
1	0	F
0	1	F
0	0	F

Paso 2:

Una vez construidas las tablas de las proposiciones por separadas, construimos la tabla de verdad de la fórmula proposicional.

P	q	r	P ∧ q	P ∧ r	(p ∧ q) ∨ (p ∧ r)
1	1	1	1	1	1
1	1	0	1	0	1
1	0	1	0	1	1
1	0	0	0	0	0
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0

Reconocimiento de patrones y generalización de patrones	***Patrón # 1*:** El primer valor de la tabla de verdad es verdadero y el último es falso.
Simulación / Evaluación	Construir una nueva tabla de verdad con los mismos valores de entrada pero, cambiando el signo que conecta las dos proposiciones para ver si el comportamiento de la tabla será el

	mismo.

Tabla 19. Problema resuelto #18 (Tabla de verdad) [38].

2.3.3.2 Algoritmo y recursividad

PROBLEMA RESUELTO # 19 Problema del factorial Cálculo del factorial de un número $n > 0$	
Abstracción	La definición recursiva del factorial de un número $n > 0$, considerando que 0! = 1 por definición, se obtiene como consecuencia de aplicar la propiedad asociativa de la multiplicación, es decir: $n! = n * (n-1) * (n-2) * \ldots * 0!$ Asociando los factores de la siguiente forma: $n! = n * (n-1) * [(n-2) * \ldots * 0!]$ Se tiene que: $n! = n * (n-1)!$ Así pues la definición recursiva de factorial es: $n! = \begin{cases} n * (n-1)! & si\ n > 0 \\ 1 & si\ n = 0 \end{cases}$
Descomposición del problema	El problema del cálculo de factorial de un número natural cualquiera, se subdivide en 2 subproblemas: 1) Resolver el problema general recursivamente. 2) Encontrar el caso base y resolverlo.
Análisis y representación de datos	**Entrada:** el número n, del que calcular su factorial y que debe

<table>
<tr><td></td><td>cumplir con la precondición de que $(n > 0)$.

Salida: el factorial del número, es decir, factorial (n) = $n!$</td></tr>
<tr><td>Pensamiento algorítmico</td><td>Paso 1:

Hacer sucesivas llamadas al procedimiento hasta alcanzar el caso base.

$n! = n * (n-1) * (n-2) * ... * 0!$

Un ejemplo para N = 2, sería:

$2! = 2 * (2-1) * (2-2)$

Paso 2:

Resolver el caso base.

$0! = 1$

Paso 3:

Resolver el resto de los subproblemas de abajo hacia arriba.

Si le damos un valor entero a N, o sea N = 3, entonces tenemos como subproblemas asociados a este problema:

<table>
<tr><th>Problema</th><th>Subproblema</th></tr>
<tr><td>$3!$</td><td>$3 * 2!$</td></tr>
<tr><td>$2!$</td><td>$2 * 1!$</td></tr>
<tr><td>$1!$</td><td>$1 * 0!$</td></tr>
<tr><td>$0!$</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Reconocimiento de patrones y</td><td>Patrón # 1</td></tr>
</table>

<table>
<tr><td>generalización de patrones</td><td>$n! = n * (n-1)!$

Este procedimiento se repite n veces para todo $n > 0$, hasta llegar al caso base o condición de parada, donde $\mathrm{n} = 0$

Patrón # 2:

$0! = 1 \rightarrow$ Condición de parada.</td></tr>
<tr><td>Simulación / Evaluación</td><td>Rastrear $n = 4$

<table>
<tr><th>Problema</th><th>Subproblemas</th><th>Resolución</th></tr>
<tr><td>$4!$</td><td>$4 * 3!$</td><td>$4 * 3 = 24$</td></tr>
<tr><td>$3!$</td><td>$3 * 2!$</td><td>$3 * 2 = 6$</td></tr>
<tr><td>$2!$</td><td>$2 * 1!$</td><td>$2 * 1 = 2$</td></tr>
<tr><td>$1!$</td><td>$1 * 0!$</td><td>$1 * 1 = 1$</td></tr>
<tr><td>$0!$</td><td>1</td><td></td></tr>
</table>
</td></tr>
</table>

Tabla 20. Problema resuelto #19 (Problema del factorial) *[39]*.

<table>
<tr><td colspan="2">PROBLEMA RESUELTO # 20 Sucesión de Fibonacci

Escriba una definición recursiva para la sucesión de Fibonacci.</td></tr>
<tr><td>Abstracción</td><td>La sucesión de Fibonacci es una sucesión infinita de números naturales que están determinados por los dos primeros números 0 y 1.

$$f_{(n)} = \begin{cases} 1 & si\ n = 0 \\ 1 & si\ n = 1 \\ f_{n-1} + f_{n-2} & si\ n \geq 2 \end{cases}$$</td></tr>
<tr><td>Descomposición del problema</td><td>Resolver la sucesión de los números de Fibonacci por la</td></tr>
</table>

	ecuación general: $f_n = f_{n-1} + f_{n-2}$ partiendo de los dos primeros valores predeterminados o condición de parada: $f_0 = 1$ $f_1 = 1$
Análisis y representación de datos	**Entrada** n (números naturales). **Salida:** la sucesión de los números de Fibonacci $f_{(n)}$.
Pensamiento algorítmico	**Paso 1:** Encontrar la ecuación para hallar el término enésimo de la sucesión de Fibonacci. $f_n = f_{n-1} + f_{n-2}$ ***Paso 2:*** Resolver el caso base $f_0 = 1$ $f_1 = 1$
Reconocimiento de patrones y generalización de patrones	**Patrón # 1:** $f_n = f_{n-1} + f_{n-2}$
Simulación / Evaluación	Hallar la sucesión de Fibonacci para N = 5 Si hacemos el uso de la ecuación $f_n = f_{n-1} + f_{n-2}$, teniendo en cuenta los dos primeros valores predeterminados, tenemos: $f_5 = f_4 + f_3 + f_2 + f_1 + f_0$ Si bien sabemos, la sucesión de Fibonacci, es un algoritmo recursivo y como tal, se resuelve de atrás hacia adelante.

	Tenemos entonces: $f_0 = 1$ $f_1 = 0 + 1 = 1$ $f_2 = 1 + 1 = 2$ $f_3 = 2 + 1 = 3$ $f_4 = 3 + 2 = 5$ $f_5 = 3 + 5 = 8$

Tabla 21. Problema resuelto #20 (Sucesión de Fibonacci) [40].

PROBLEMA RESUELTO # 21 Suma recursiva de los N primeros números naturales Escriba un algoritmo recursivo para el cálculo de la suma de los N primeros números naturales.	
Abstracción	Plantear una función que se llame a sí misma y un caso base o condición de parada.
Descomposición del problema	*Condición de parada:* $s(n) = 0 \rightarrow si\ n = 0$ *Función recursiva:* $s(n) = n + s(n1) \rightarrow si\ n > 0$
Análisis y representación de datos	**Entrada:** n que pertenece a los números naturales. **Salida:** El sumatorio de los n números naturales $suma(n) = s(n)$
Pensamiento algorítmico	**Paso 1:** Definir caso base o condición de parada.

<table>
<tr><td></td><td>$s(n) = 0 \rightarrow si\ n = 0$

Paso 2: Definir función recursiva para la salida.

$s(n) = n + s(n-1) \rightarrow si\ n > 0$</td></tr>
<tr><td>Reconocimiento de patrones y generalización de patrones</td><td>Patrón # 1:

$s(n) = n + s(n-1)$

Este procedimiento se repite n veces hasta llegar al caso base o condición de parada, donde $n = 0$</td></tr>
<tr><td>Simulación / Evaluación</td><td>Rastrear el algoritmo de la sumatoria para $N = 3$:

<table>
<tr><th>Problema</th><th>Suma</th><th>Subproblema</th><th>Resolución</th></tr>
<tr><td>$s(3)$</td><td>$3 + 2 + 1 = 6$</td><td>$3 + s(2)$</td><td>$3 + 3 = 6$</td></tr>
<tr><td>$s(2)$</td><td>$2 + 1 + 0 = 3$</td><td>$2 + s(1)$</td><td>$2 + 1 = 3$</td></tr>
<tr><td>$s(1)$</td><td>$1 + 0 = 1$</td><td>$1 + s(0)$</td><td>$1 + 0 = 1$</td></tr>
<tr><td>$s(0)$</td><td>0</td><td>0</td><td></td></tr>
</table>
</td></tr>
</table>

Tabla 22. Problema resuelto #21 (Suma recursiva de los N primeros números naturales) [41].

<table>
<tr><td colspan="2">PROBLEMA RESUELTO # 22 Problema del robot (recursividad)

Un robot puede dar pasos de 1 ó 2 metros. Escribir un algoritmo recursivo para calcular el número de formas en que puede recoger n metros.</td></tr>
<tr><td>Abstracción</td><td>Sea $camina(n)$ el número de formas en que el robot puede caminar n metros, hemos observado que:

$camina\ 1 = 1$</td></tr>
</table>

	$camina\ 2 = 2$ Por lo tanto, el algoritmo calcula la función definida como: $$camina(n) = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ camina(n-1) + camina(n-2) & n > 2 \end{cases}$$
Descomposición del problema	Para hallar el número n-ésimo de formas en que puede caminar un robot, hay que plantear: 1) El (los) caso (s) base (s) que nos da el problema: $camina\ 1$ $camina\ 2$ 2) La fórmula general: $camina(n) \rightarrow n > 2$
Análisis y representación de datos	**Entrada:** N: (Número de pasos que puede dar un robot). **Salida:** Número de formas en que el robot puede caminar n metros. $camina(n)$:
Pensamiento algorítmico	***Paso 1:*** Resolver el caso base $camina\ 1 = 1$ $camina\ 2 = 2$ ***Paso 2:*** Para calcular la cantidad de formas en que un robot puede caminar n metros, teniendo en cuenta que ya se han declarado los casos bases, tendremos: $camina(n) = camina(n-1) + camina(n-2) \rightarrow n > 2$

<table>
<tr><td>Reconocimiento de patrones y generalización de patrones</td><td>Patrón # 1: $camina(n) = camina(n-1) + camina(n-2)$

Este procedimiento se repite para todo $n > 2$.</td></tr>
<tr><td>Simulación / Evaluación</td><td>
<table>
<tr><th>Distancia</th><th>Serie de pasos</th><th>Número de formas de recogerlos</th></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td>1,1 o 2</td><td>2</td></tr>
<tr><td>3</td><td>1,1,1 o 1,2 o 2,1</td><td>3</td></tr>
<tr><td>4</td><td>1,1,1,1 o 1,1,2 o 1,2,1 o 2,1,1 o 2,2</td><td>5</td></tr>
</table>
Formas en que el robot puede recorrer 4 metros:

$camina(4) = camina(3) + camina(2) = 3 + 2 = 5$
</td></tr>
</table>

Tabla 23.Problema resuelto #22 (Problema del Robot) [42]

2.4 Conclusiones

- Los 6 componentes del pensamiento computacional (abstracción, análisis y representación de datos, descomposición del problema, pensamiento algorítmico, reconocimiento y generalización de patrones, simulación/evaluación) fueron desarrollados en cada uno de los problemas que conforman el “Módulo de problemas resueltos (MPR)”.
- El MPR quedó conformado de la siguiente forma:
  - ✓ Problemas de búsqueda sin heurística (5)
  - ✓ Problemas de Matemática I (5)
  - ✓ Problemas de Matematica II (5)
  - ✓ Problemas de Matemática Discreta (7)

Capítulo III: Análisis de los resultados

3.1 Introducción

3.2 Población y muestra

El trabajo se desarrolla en la Universidad de Cienfuegos sede: “Carlos Rafael Rodríguez” del municipio de Cienfuegos. La población motivo de estudio está compuesta por 18 estudiantes entre 18 y 21 años de ambos sexos. Para seleccionar estos 18 estudiantes, se realizó una convocatoria por medio de comunicación oral. Estos 18 estudiantes se ubicaron en dos grupos de 9 y 9 estudiantes cada uno siguiendo un muestreo aleatorio simple. Uno de los grupos (el grupo de 9 estudiantes) se tomó como experimental (GE) al que se le aplico el *“módulo de problemas resueltos”* y otro de control (GC) que no se vio afectado por la aplicación del “módulo de problemas resueltos” sino que sigue su aprendizaje a través de los métodos de resolución de problemas tradicionales.

Con los estudiantes del grupo experimental se desarrollaron 4 intervenciones de 90 minutos cada una, en las cuales se explicaron varios problemas resueltos del módulo propuesto como parte de esta investigación. En el epígrafe siguiente se explican las intervenciones realizadas y los problemas trabajados en cada una de estas.

3.3 Diseño de la investigación

La investigación realizada es un ***diseño experimental*** *con pre-test y post-test y, con un grupo de control y un grupo experimental al cual se le aplicó la intervención* [43].

3.4 Métodos y técnicas utilizadas

Del Nivel teórico

- **Histórico lógico:** Permitió la búsqueda de los fundamentos que antecedieron al problema científico tratado, los resultados históricos obtenidos, su desarrollo, significación y su incidencia en los resultados actuales.

  Se utilizó para conocer el desarrollo histórico del tema de investigación y así poder argumentar acerca del estado actual del mismo.

- **Analítico-Sintético:** Se empleó durante el proceso de consulta de la literatura, la documentación especializada y en la aplicación de otros métodos del conocimiento científico.
- **Inductivo-Deductivo:** Este método facilitó la interpretación de los datos empíricos; así como descubrir regularidades importantes y relaciones entre los distintos componentes de la investigación.

Del Nivel empírico

- **Entrevista Informal:** A través de este método se pudo conocer sobre los problemas fundamentales que presentan los estudiantes de ingeniería informática primer año, así como el grado de conocimientos de los profesores acerca de los componentes del pensamiento computacional y su aplicación desde las asignaturas.
- **Observación:** Este método permitió realizar una observación científica del desenvolvimiento de los estudiantes durante la aplicación del "módulo de problemas resueltos".
- **Análisis de Documentos:** Este método fue empleado para profundizar en el estudio y análisis del tema objeto de investigación y los elementos más importante del PC.

Del Nivel Estadístico

Se realizó la prueba de los signos y la prueba de Mann-Whitney, utilizando para el procesamiento estadístico el software SPSS.

3.5 Planificación de Intervenciones

3.5.1 Intervención # 1

El martes 15 de mayo de presente año a las 11:00 am, se llevó a cabo la primera intervención con el grupo experimental seleccionado como muestra para la presente investigación, en donde asistieron los 9 estudiantes seleccionados para ello. Se trabajaron 3 problemas, uno por cada grupo de problemas seleccionados para tal efecto:

- **Problemas sin Heurística:** "*Problema Resuelto # 2: Misioneros y caníbales".*
- **Problema de Matemática I:** "*Problema Resuelto # 8:* Derivada n-ésima de una función real de una variable real".

- **Problemas de Matemática Discreta:** "*Problema Resuelto # 22*: Problema del robot".

3.5.2 Intervención # 2

El jueves 24 de mayo del presente año a las 10:30 am, se llevó a cabo la segunda intervención. En esta, se trabajó un problema de matemática II: "*Problema resuelto # 15: Volumen de un sólido*.

3.5.3 Intervención # 3

En el día 25 de mayo del presente año a las 11:00 am, se llevó a cabo la tercera intervención en donde se trabajaron dos problemas: uno de matemática discreta "*Problema Resuelto #17: Traducción al lenguaje natural"* y uno de problemas de búsqueda sin heurística *"Problema Resuelto # 1: Torres de Hanói"*.

3.5.4 Intervención # 4

El día 5 de mayo del presente año, se llevó a cabo la cuarta intervención, en donde se trabajaron dos problemas: un problema de matemática I "*Problema resuelto # 12: Interpretación física de la derivada parcial"* y un problema de Búsqueda sin Heurística "*Problema resuelto # 4: Jarras de agua (8, 5, 3 litros respectivamente)*".

3.6 Prueba Inicial

Para la aplicación de la prueba inicial se tomaron 18 estudiantes, siendo esta cifra el total de estudiantes de primer año de la carrera de ingeniería informática de la Universidad de Cienfuegos.

La prueba, se hizo constar de tres preguntas / problemas extraídos del "módulo de problemas resueltos" propuesto como solución al problema objeto de investigación ***(ver anexo 9).*** Estos problemas se seleccionaron de la siguiente forma:

- Un problema del grupo de **problemas de búsqueda sin heurística**: "*Problema Resuelto # 2: Misioneros y caníbales".*
- Un problema del grupo de **problemas de matemática I**: "*Problema Resuelto # 8:* Derivada n-ésima de una función real de una variable real".
- Un problema del grupo de **problemas de matemática discreta:** "*Problema Resuelto # 22*: Problema del robot".

Para la evaluación de cada pregunta de la prueba se tuvieron en cuenta los siguientes aspectos:

- Abstracción
- Análisis y representación de datos
- Descomposición del problema del problema
- Pensamiento algorítmico
- Reconocimiento y generalización de patrones
- Simulación / Evaluación
- Nota total

La siguiente tabla refleja la evaluación de cada estudiante desde el punto de vista cualitativo en la prueba inicial:

Grupo Control																					
No	**P # 1**							**P # 2**							**P # 3**						
	Ab	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**
1	2	3	3	3	2	3	**3**	2	2	2	2	2	2	**2**	3	3	4	4	4	2	**4**
2	3	4	4	4	3	4	**4**	2	2	2	2	3	2	**2**	5	4	5	4	4	2	**4**
3	3	3	3	3	2	3	**3**	3	3	5	3	3	2	**3**	5	5	5	4	4	5	**5**
4	4	5	5	4	4	2	**5**	2	2	2	3	2	2	**2**	4	5	5	4	3	3	**4**
5	3	3	3	3	2	2	**3**	2	2	2	2	2	2	**2**	2	3	2	2	3	2	**3**
6	2	2	2	2	2	2	**2**	2	2	2	3	2	2	**2**	2	3	3	3	2	2	**3**
7	2	2	3	3	2	2	**2**	2	2	2	2	2	2	**2**	2	3	3	3	3	3	**3**
8	2	2	3	3	2	2	**2**	2	2	3	2	2	2	**2**	2	2	2	2	2	2	**2**
9	3	2	3	2	2	2	**2**	2	2	2	2	3	2	**2**	2	2	2	2	2	2	**2**
Grupo Experimental																					

No	P # 1							P # 2							P # 3						
	Ab	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**
1	4	5	5	4	3	2	**4**	2	2	2	2	2	2	**2**	3	2	2	2	2	2	**2**
2	3	4	4	3	2	2	**3**	3	2	3	2	2	2	**2**	4	3	4	4	3	2	**4**
3	3	4	4	3	2	2	**3**	2	2	2	2	2	2	**2**	4	4	5	5	3	5	**5**
4	3	2	4	3	2	2	**3**	2	2	2	2	2	2	**2**	4	4	5	5	3	5	**5**
5	3	3	3	2	2	2	**2**	2	2	2	2	2	2	**2**	3	4	4	4	3	2	**4**
6	4	4	4	4	2	3	**4**	3	2	5	3	5	2	**4**	2	2	2	2	2	2	**2**
7	3	3	3	2	2	2	**3**	4	3	5	3	5	2	**4**	3	5	5	4	3	2	**4**
8	3	2	3	4	2	3	**3**	2	2	2	3	2	2	**2**	3	3	3	3	3	2	**3**
9	2	2	2	2	2	2	**2**	2	2	2	2	2	2	**2**	2	2	2	2	2	2	**2**

Tabla 24. Evaluación cualitativa de la prueba inicial por estudiante.

Para comprobar estadísticamente que los dos grupos independientes (GE y GC) proceden de la misma población y así demostrar su aleatoriedad se aplicó la prueba Mann-Whitney (Miller et al. s.d., pág.310) a partir de los datos del diagnóstico. Al aplicar esta prueba no paramétrica para grupos independientes resultó que la significación asintótica (bilateral) es: 0.571 (PI#1_NT), 0.454 (PI#2_NT) y 0.819 (PI#3_NT). Como se puede observar, para las notas totales de las tres preguntas de la prueba inicial los resultados fueron mayores que el nivel de significancia α=0.05, por tanto se acepta la hipótesis de que las dos muestras (GE y GC) proceden de la misma población, o sea, no tienen diferencias significativas (hipótesis nula).

Estadísticos de contraste[a]

	PI#1_NT	PI#2_NT	PI#3_NT
U de Mann-Whitney	34,500	35,000	38,000
W de Wilcoxon	79,500	80,000	83,000
Z	-,567	-,748	-,229
Sig. Asíntota. (bilateral)	,571	,454	,819
Sig. exacta [2*(Sig. unilateral)]	,605[b]	,666[b]	,863[b]

a. Variable de agrupación: Grupo

b. No corregidos para los empates.

Leyenda de variables utilizadas en el SPSS:

PI#1_NT: Nota total de la pregunta No 1 en la Prueba Inicial.

PI#2_NT: Nota total de la pregunta No 2 en la Prueba Inicial.

PI#3_NT: Nota total de la pregunta No 3 en la Prueba Inicial.

3.7 Prueba Final

Para la aplicación de la prueba final se tomaron los 18 estudiantes que hacen el total de estudiantes de primer año.

La prueba, se hizo constar de tres preguntas / problemas extraídos del "módulo de problemas resueltos" propuesto como solución al problema objeto de investigación ***(ver anexo 10)***. Estos problemas se seleccionaron de la siguiente forma:

- Un problema del grupo de **problemas de búsqueda sin heurística**: "*Problema Resuelto #5: Jarras de agua (4 y 3 litros respectivamente)".*
- Un problema del grupo de **problemas de matemática I**: "*Problema Resuelto #9:* Derivada n-ésima de una función real de una variable real".
- Un problema del grupo de **problemas de matemática discreta:** "*Problema Resuelto #* 20 Problema del factorial".

Para la evaluación de esta prueba se tuvieron en cuenta los mismos aspectos que en la prueba inicial descritos anteriormente.

La siguiente tabla refleja la evaluación de cada estudiante desde el punto de vista cualitativo en el diagnóstico inicial:

Grupo Control																					
No	**P # 1**							**P # 2**							**P # 3**						
	Ab	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**
1	3	2	3	2	2	2	**2**	3	2	4	3	3	2	**3**	2	2	2	2	2	2	**2**

2	3	4	4	4	3	4	**4**	3	3	4	3	3	2	**3**	5	4	5	5	4	4	**5**
3	3	3	3	3	2	3	**3**	4	4	5	3	4	3	**4**	5	5	5	4	4	5	**5**
4	4	5	5	4	4	2	**5**	3	3	3	3	4	3	**3**	4	5	5	4	3	3	**4**
5	3	3	3	3	2	2	**3**	2	2	2	2	2	2	**2**	2	3	2	2	3	2	**3**
6	2	2	2	2	2	2	**2**	3	2	3	3	3	2	**3**	3	2	2	3	3	3	**3**
7	3	3	3	3	2	2	**3**	3	2	2	2	3	2	**2**	2	2	2	2	2	2	**2**
8	3	2	3	3	3	3	**3**	2	2	2	2	2	2	**2**	2	2	2	2	2	2	**2**
9	3	2	2	3	2	2	**2**	2	2	2	2	2	2	**2**	3	3	3	2	3	3	**3**
Grupo Experimental																					
No	**P # 1**							**P # 2**							**P # 3**						
	Ab	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**	**Ab**	**AR**	**D**	**Al**	**P**	**S**	**Nt**
1	4	3	5	5	4	5	**5**	4	3	5	4	5	2	**5**	3	4	5	4	4	5	**4**
2	5	4	5	5	4	5	**5**	4	3	4	4	5	3	**4**	5	4	5	5	5	4	**5**
3	5	4	5	5	4	5	**5**	5	3	5	5	5	5	**5**	5	4	5	5	5	5	**5**
4	5	4	5	5	3	5	**5**	4	3	4	4	4	3	**4**	4	3	4	4	4	3	**4**
5	4	3	5	4	3	3	**4**	4	3	4	4	4	3	**4**	4	3	4	5	4	2	**4**
6	5	3	5	5	4	5	**5**	4	4	5	5	5	3	**5**	4	3	5	5	4	2	**4**
7	4	3	4	4	3	4	**4**	4	5	5	3	5	4	**5**	4	5	5	4	3	3	**4**
8	4	4	4	4	3	3	**4**	3	3	4	4	3	3	**3**	4	4	5	4	3	4	**4**
9	4	3	5	5	4	5	**5**	2	2	2	2	2	2	**2**	3	3	4	3	3	2	**3**

Tabla 25. Evaluación cualitativa de la prueba inicial por estudiante.

Se utilizaron los resultados integrales finales de cada grupo para comprobar estadísticamente que el grupo experimental (GE) mostró mejores calificaciones que el grupo de control. Para ello se aplicó la prueba Mann-Whitney, esta es una prueba no paramétrica utilizada para determinar si existen diferencias entre los resultados alcanzados por ambas muestras [44]. Al aplicar la prueba resultó que la significación asintótica (bilateral) de las variables de la nota final en las preguntas no 1 y no 2 es 0.002 y 0.008 respectivamente, siendo menores que el nivel de significancia α=0.05, por tanto se acepta la hipótesis (hipótesis nula) de que existen diferencias significativas entre las muestras (GE y GC). Sin embargo, la significación asintótica (bilateral) de la variable de la nota final en la pregunta no 3 es 0.08, siendo mayor que el nivel de significancia α=0.05, por tanto se acepta la hipótesis (hipótesis alternativa) de que no existen diferencias significativas entre las muestras (GE y GC).

Estadísticos de contraste[a]

	PF#1_NT	PF#2_NT	PF#3_NT
U de Mann-Whitney	7,500	11,500	21,500
W de Wilcoxon	52,500	56,500	66,500
Z	-3,042	-2,644	-1,752
Sig. Asíntota. (bilateral)	,002	,008	,080
Sig. exacta [2*(Sig. unilateral)]	,002[b]	,008[b]	,094[b]

a. Variable de agrupación: Grupo

b. No corregidos para los empates.

Leyenda de variables utilizadas en el SPSS:

PF#1_NT: Nota total de la pregunta No 1 en la Prueba Final.

PF#2_NT: Nota total de la pregunta No 2 en la Prueba Final.

PF#3_NT: Nota total de la pregunta No 3 en la Prueba Final.

3.8 Comparación de los resultados inicial y final en el grupo experimental.

- **Resultado de estadísticos de contraste para la pregunta # 1**

Al aplicar esta prueba al grupo experimental (grupo al que se le aplica el *"módulo de problemas resueltos"* propuesto), resultó que la significación asintótica (bilateral) es 0.004, siendo menor que el nivel de significancia α=0.05, por tanto se acepta la hipótesis de que el "módulo de problemas resueltos" aplicado es eficaz (hipótesis alterna) ***(ver anexo 11)***.

Estadísticos de contraste (b)

	PF#1_Ab_GE - PI#1_Ab_GE	PF#1_AD_GE - PI#1_AD_GE	PF#1_DP_GE - PI#1_DP_GE	PF#1_Al_GE - PI#1_Al_GE	PF#1_P_GE - PI#1_P_GE	PF#1_S_GE - PI#1_S_GE	PF#1_NT_GE - PI#1_NT_GE
Sig. exacta (bilateral)	.008(a)	1.000(a)	.008(a)	.008(a)	.004(a)	.008(a)	.004(a)

a Se ha usado la distribución binomial.

b Prueba de los signos

- **Resultado de estadísticos de contraste para la pregunta # 2**

Al aplicar esta prueba al grupo experimental (grupo al que se le aplica el "módulo de problemas resueltos" propuesto), resultó que la significación asintótica (bilateral) es 0.008, siendo menor que el nivel de significancia α=0.05, por tanto se acepta la hipótesis de que el "módulo de problemas resueltos" aplicado es eficaz (hipótesis alterna) ***(anexo 12).***

Estadísticos de contraste (b)

	PF#2_Ab_GE - PI#2_Ab_GE	PF#2_AD_GE - PI#2_AD_GE	PF#2_DP_GE - PI#2_DP_GE	PF#2_Al_GE - PI#2_Al_GE	PF#2_P_GE - PI#2_P_GE	PF#2_S_GE - PI#2_S_GE	PF#2_NT_GE - PI#2_NT_GE
Sig. exacta (bilateral)	.063(a)	.016(a)	.008(a)	.125(a)	.016(a)	.453(a)	.008(a)

a Se ha usado la distribución binomial.

b Prueba de los signos

- **Resultado de estadísticos de contraste para la pregunta # 3**

Al aplicar esta prueba al grupo experimental (grupo al que se le aplica el "módulo de problemas resueltos" propuesto), resultó que la significación asintótica (bilateral) es 0.219, siendo mayor que el nivel de significancia α=0.05, por tanto se rechaza la hipótesis alterna. Este resultado evidencia que los resultados iniciales y finales de los estudiantes en la pregunta #3 no tuvieron diferencias significativas, los resultados iniciales fueron buenos y se mantuvieron ***(anexo 13)***.

Estadísticos de contraste (b)

	PF#3_Ab_GE PI#3_Ab_GE	PF#3_AD_GE - PI#3_AD_GE	PF#3_DP_GE - PI#3_DP_GE	PF#3_Al_GE - PI#3_Al_GE	PF#3_P_GE - PI#3_P_GE	PF#3_S_GE - PI#3_S_GE	PF#3_NT_GE - PI#3_NT_GE
Sig. exacta (bilateral)	.016(a)	.453(a)	.219(a)	.125(a)	.016(a)	.375(a)	.219(a)

a Se ha usado la distribución binomial.
b Prueba de los signos.

3.9 Conclusiones

La prueba estadística de los Signos realizada para contrastar los resultados iniciales y finales del grupo experimental mostró que existe superioridad en los resultados finales lo cual evidencia la eficacia de las intervenciones realizadas con el “Módulo de problemas resueltos”.

La prueba estadística de Mann- Whitney realizada para contrastar los resultados finales del grupo experimental con los del grupo de control mostró la superioridad de los resultados del grupo experimental.

Conclusiones

- Los estudiantes de primer año de ingeniería informática no trabajan los componentes del pensamiento computacional de forma cohesionada desde las diferentes disciplinas, por tanto no se aprovechan los beneficios que estos pueden aportar en su formación.
- Se propone una definición contextualizada de pensamiento computacional: Proceso cognitivo ejecutado por humanos para la resolución de problemas diversos haciendo uso de conceptos computacionales que involucra, de forma relacionada, los componentes siguientes: abstracción, análisis y representación de datos, descomposición del problema, pensamiento algorítmico, reconocimiento y generalización de patrones, simulación / evaluación.
- El módulo de problemas resueltos quedó conformado de la siguiente forma:
  - ✓ Problemas de búsqueda sin heurística (5)
  - ✓ Problemas de Matemática I (5)
  - ✓ Problemas de Matematica II (5)
  - ✓ Problemas de Matematica Discreta (7)
- Las pruebas estadísticas realizadas (Signos y Mann-Whitney) para contrastar los resultados iniciales y finales del grupo experimental y del grupo de control, mostraron que existe superioridad en los resultados finales lo cual evidencia la eficacia de las intervenciones realizadas con el “módulo de problemas resueltos”.

Recomendaciones

- Continuar desarrollando investigaciones que aborden el desarrollo del pensamiento computacional de estudiantes de ingeniería informática desde asignaturas de otras disciplinas de la carrera y años académicos (segundo a quinto año).
- Completar el componente simulación/evaluación en algunos problemas del módulo propuesto realizando las simulaciones en un lenguaje de alto nivel como Python.

Referencia bibliográfica

[1] L. M. Rivera, «A potenciar el Pensamiento computacional». de diciembre de-2017.

[2] J. J. S. Granados, «Propuesta Metodológica para Desarrollar un Sistema Tutor Inteligente Basado en Web para Estudiantes de Ingeniería», Universidad de Cienfuegos «Carlos Rafael Rodríguez», Cienfuegos, 2016.

[3] A. Ovalle, «Entorno Integrado de Enseñanza Aprendizaje basado en Sistemas Tutoriales Inteligentes & Ambientes Colaborativos.» 2007.

[4] M. Lugo, «Un camino al Pensamiento Computacional», UNIVERSIDAD DE LA SABANA, Chía - Cundinamarca, 2016.

[5] S. Papert, «Children, computers, and powerful ideas». 1980.

[6] J. M. Wing, «Computational Thinking», *Communications of the ACM*, vol. 49, pp. 33-35, mar. 2006.

[7] J. M. Wing, «Cuaderno de investigación: el pensamiento computacional-Qué y por qué?», *El Boletín Enlace*, vol. 6, p. 1.32, 2011.

[8] Malena, «el Pensamiento-Parte II», de diciembre de-2007. .

[9] M. de Vega, «Pensamiento», en *Introducción a la Psicología Cognitiva*, vol. Tomo II, 2 vols., La Habana: Féliz Varela, 2005, p. 439.

[10] J. M. Wing, J. Cuny, y L. Snyder, «Desmitificando el pensamiento computacional para los científicos no informáticos», 2010.

[11] M. A. B. Saldaña, «Apuntes teóricos sobre el pensamiento matemático y multiplicativo en los primeros niveles», *Edma 0-6: Educación Matemática en la Infancia*, p. 17, 2012.

[12] R. y otros Cantoral, *Apuntes teóricos sobre el pensamiento matemático y multiplicativo en los primeros niveles*. México: Universidad Virtual, 2005.

[13] P. A. Facione, «Pensamiento Crítico: ¿Qué es y por qué es importante?», Chicago, 2007.

[14] C. C. de A. UNLP Datos y Programas, «Por qué “pensar algoritmos” es tan importante en Informática?», p. 22, diciembre-2016.

[15] M. González, «Codigoalfabetización y Pensamiento Computacional en Educacion Primaria y Secundaria: Validación de un Instrumento y Evaluacion de Programas», Tesis Doctoral, UNED, España, 2016.

[16] J. M. Wing, «Computational Thinking and Thinking about computing», *Philosophical ransactions of the royal society*, p. 3718, 2008.

[17] V. Barr y C. Stephenson, «Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?», *ACM Inroads*, vol. 2(1), pp. 48-54, 2011.

[18] Royal Society, «Shurt downor restart: the way forward for computing in UK», Reino Unido, 2012.

[19] CSTA & ISTE, «Operational Definition of computational thinking for K-12 Education». 2015.

[20] R. Brennan y C. Resnick, «New frameworks of studying and assessing the devepment of computational thinking», *Proceeding of the 2012 Anual Meeting of the American Educational Research Association*, 2012.

[21] L. Zsakó y P. Szlávi, «ICT Competence», *a Logarithmic Thinking*, vol. 5, 2012.

[22] D. B. Berch, «Making sense of number sense: implications for children with mathematical disabilities», pp. 333-339, 2005.

[23] J. L. Z. López, «Abstracción», 2014. [En línea]. Disponible en: www.pensamientocomputacional.org. [Accedido: 22-mar-2018].
[24] X. B. Olabe, M. Á. O. Basogain, y J. C. O. Basogain, «Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje», *RED-Revista de Educación a Distancia, 46(6).*, vol. 46, p. 6, sep. 2015.
[25] M. Rouse, «Análisis de Datos», 2012.
[26] L. Puente, «El Pensamiento Computacional. Un reflejo del razonamiento y la comprensión humanos», *Netcommerce*, Dic-2016. .
[27] A. VeRa, «Reconocimiento de patrones», de Diciembre de-2015. [En línea]. Disponible en: http://www.aldape.org.mx/sp/rp.htm (Reconocimiento de Patrones). [Accedido: 20-mar-2018].
[28] J. S. C. Carmona, «Maneras de Generalizar Patrones Lineales a partir de Secuencias Pictóricas por niños de quinto grado». 2015.
[29] Google for Education, «Exploring Computational Thinking», 2015. [En línea]. Disponible en: http://www.google.com/edu/resources/programs/exploring-computational-thinking/.
[30] J. L. Z. López, «Simulación», *Pensamiento Computacional*, 2014. .
[31] B. Galicia, «Simulacion definiciones», mar. 2011.
[32] «Evaluar», *Diccionario de sinónimos y antónimos*. .
[33] A. I. M. Vargas, «La Evaluación Educativa: Concepto, Períodos y Modelos», *Revista Electrónica «Actualidades Investigativas en Educación»*, vol. 4, n.º 2, p. 2, jul. 2004.
[34] H. A. Simon, «The funtional equivalence of problem solving skills», en *Cognitive psychology*, vol. 2, 2 vols., 1975, pp. 268-88.
[35] J. V. Q. Moreno, E. L. R. Rodríguez, y L. C. R. Sandoval, «3 Canibales y 3 Misioneros», Universidad Nacional de Trujillo, Peru.
[36] J. Camejo, «Métodos de Solucion de Problemas: Guía de ejercicios para la clase práctica 1: Ejercicios de preparación para la clase práctica 1 sobre formulación de problemas de IA y búsqueda a ciegas». 20-ene-2016.
[37] V. Toledo, «Traducción al lenguaje del cálculo proposicional y determinación de deducción correcta por vía semántica», presentado en Clase Práctica 3, Universidad de Cienfuegos, feb-2013.
[38] R. Johnsonbaugh, «Proposiciones Condicionales y Equivalencia Lógica», en *Matemáticas Discretas*, Cuarta edicion., vol. 1, pp. 8-17.
[39] V. Toledo, «Cálculo del Factorial», Universidad de Cienfuegos, 23-abr-2013.
[40] E. R. Aznar, «Números de Fibonacci, su complejidad y su programación», 2007. .
[41] V. Toledo, «Algorítmo Recursivo: Suma de los N primeros números naturales». 24-abr-2013.
[42] V. Toledo, «Algorítmo Recursivo: cálculo del número de formas en que un robot puede recorrer una distancia n en metros». 24-abr-2013.
[43] R. Hernández Sampieri, C. Fernández Collado, y P. BaptistaLucio, *Metodología de la Investigación*, Segunda Edición. México: Editorial Mexicana, 1998.
[44] I. R. Miller, J. E. Freund, y R. Johnson, *Probabilidad y Estadística para Ingenieros*. La Habana: Félix Varela, 2005.

Glosario de términos

CPC: Componentes del Pensamiento Computacional

GE: Grupo Experimental

GC: Grupo Control

MPR: Módulo de Problema Resueltos

PC: Pensamiento Computacional

UCF: Universidad de Cienfuegos

Anexos

Anexo 1 Modelo CAS (conceptos y aproximaciones)

Conceptos:

- **Lógica (*'Logic'*)**: el razonamiento lógico nos ayuda a explicar por qué algo sucede. Si configuramos dos ordenadores de la misma manera, les damos las mismas instrucciones (el 'programa') y el mismo *'input'*, podemos entonces garantizar que llegaremos al mismo *'output'*. Los ordenadores no hacen las cosas según les apetezca, ni trabajan diferente en función de cómo se sientan ese día. Esto significa que son previsibles. Por esta razón, podemos utilizar el razonamiento lógico para determinar exactamente lo que un programa informático hará.
- **Algoritmos (*'Algorithms'*)**: un algoritmo es una secuencia de instrucciones o conjunto de reglas para llevar algo a cabo. Podemos pensar, por ejemplo, en el camino más rápido para ir del colegio a casa (ejemplo: *"gira a la izquierda, sigue recto 5 kilómetros, gira a la derecha…"*) Podemos concebir esto como un algoritmo (una secuencia de instrucciones que nos lleva al destino seleccionado). Los 'algoritmos' están escritos generalmente para ser entendidos por humanos; en este sentido se diferencian de los 'programas', escritos para ser ejecutados por un ordenador.
- **Descomposición (*'Decomposition'*)**: el proceso de fraccionar un problema en sus partes constitutivas, más pequeñas y manejables, se conoce como descomposición. La descomposición nos ayuda a resolver problemas complejos y a gestionar proyectos grandes. Esta aproximación tiene múltiples ventajas: hace que el proceso sea manejable y abordable (un problema grande es desalentador, pero un conjunto de problemas pequeños y relacionados entre sí es más llevadero). También implica que la tarea puede ser llevada a cabo por un equipo trabajando juntos y coordinados, cada uno aportando su experiencia y habilidades en la parte del problema adecuada.
- **Patrones (*'Patterns'*)**: los patrones están por todas partes, por ejemplo, usamos patrones meteorológicos para predecir qué tiempo hará mañana; los niños pueden percibir patrones en cómo sus profesores reaccionan a sus conductas para decidir cómo comportarse la próxima ocasión. Al identificar patrones podemos hacer predicciones, crear reglas y resolver problemas más generales. En computación, el

método de buscar una aproximación general a toda una clase de problemas análogos se llama ‘generalización’.

- **Abstracción (*‘Abstraction’*)**: la abstracción se sitúa en el corazón del pensamiento computacional. El proceso de abstraer, decidir a qué detalles necesitamos prestar atención y cuáles ignorar, atraviesa (está en el núcleo) del pensamiento computacional. La abstracción tiene que ver con simplificar las cosas; identificando qué es importante sin preocuparnos demasiado por lo anecdótico o irrelevante. La abstracción nos permite manejar la complejidad. Usamos abstracciones para gestionar la complejidad de la vida en las aulas. Por ejemplo, un horario escolar es una abstracción de lo que ocurre en una típica semana de curso: recoge la información clave (qué asignatura es impartida en cada aula, cada día, a cada hora, y por cuál profesor), dejando fuera múltiples capas de complejidad (objetivos de la asignatura, actividades y materiales, etc…)
- **Evaluación sistemática (*‘Evaluation’*):** la evaluación tiene que ver con hacer juicios, de una manera objetiva y sistemática cuando sea posible. Evaluar es algo que hacemos cada día: hacemos juicios sobre qué hacer y qué pensar basados en una variedad de factores y criterios. Por ejemplo, al considerar la compra de un nuevo dispositivo digital para el aula, podría haber una serie de criterios a ser considerados, por ejemplo: el sistema operativo, la portabilidad, memoria, tamaño de pantalla, facilidad de uso, precio… En Ciencias de la Computación, la evaluación es sistemática y rigurosa; tiene que ver con juzgar la calidad, la efectividad y la eficiencia de las soluciones, sistemas, productos y procesos. La evaluación comprueba que las soluciones aportadas hacen el trabajo para el que fueron diseñadas.

Aproximaciones:

- **Experimentación (*‘Tinkering’*):** El *‘tinkering’* significa probar a hacer (activamente) las cosas. Para los niños pequeños es la forma natural de aprender sobre algo: el juego espontáneo, la exploración y la experimentación. Para los estudiantes más mayores y adultos, es una exploración intencionada y un hacer basado en el ‘ensayo, error y mejora’ Habitualmente hacemos *‘tinkering’* cuando nos encontramos con algo nuevo y queremos descubrir cómo funciona; por ejemplo, cuando adquirimos un teléfono móvil nuevo, podríamos probar todas sus características y funcionalidades,

para posteriormente utilizar intencionadamente aquellas que nos sirvan. Tener la libertad de explorar a través de retos lúdicos en un ambiente no punitivo genera confianza y una actitud proactiva. Problemas abiertos animan la creatividad y la diversidad de ideas. Así, el *'tinkering'* (como aproximación al desarrollo del pensamiento computacional en el aula) debería ser divertido, libre, creativo y lleno de preguntas, retos y sorpresas.

- **Creación (*'Creating'*):** crear tiene que ver con planificar, hacer y evaluar cosas (p.e. animaciones, juegos o robots). Programar es un proceso creativo. El trabajo creativo implica tanto originalidad como la generación de un producto valioso: típicamente algo que es útil o que encaja con el propósito previsto. A veces lo que creamos son productos que dan respuesta a necesidades particulares; a veces lo que creamos parte de iniciativas para dar rienda suelta a nuestra expresión. Los productos del pensamiento computacional pueden ser tanto puramente digitales (a través de la creación de software), como artefactos físicos y tangibles (a través de la extensión hacia el hardware).

- **Depuración (*'Debugging'*):** los errores en un algoritmo, programa o código se denominan *'bugs'*, y el proceso de encontrarlos y arreglarlos se denomina *'debugging'* (puede traducirse al español como 'depuración'). El *'debugging'* habitualmente lleva mucho más tiempo que escribir originalmente el código. En la vida cotidiana, depuramos todo el tiempo: sencillamente es darse cuenta de los errores y solucionarlos. Por ejemplo, podemos comprobar una frase que hemos escrito en un correo electrónico para ver si tiene sentido y, en caso contrario, arreglarla-depurarla. El proceso de depurar un programa para que funcione bien puede ocasionar frustración y bullicio en el aula; para afrontarlo, una manera adecuada es proporcionar a los estudiantes un buen conjunto de estrategias de depuración que puedan usar para detectar-arreglar cualquier *'bug'* de programación. Algunos *'bugs'* son errores lógicos, otros son errores sintácticos. Los errores lógicos son como escribir una historia en la cual la trama no tiene sentido; los errores sintácticos son como escribir una historia con fallos gramaticales, ortográficos o de puntuación.

- **Perseverancia (*'Persevering'*):** la programación informática es una tarea dura. Esto es parte de su encanto: escribir código elegante y eficaz es un reto intelectual que

requiere, no sólo una comprensión algorítmica del problema a resolver y codificar (y un conocimiento del lenguaje de programación utilizado para ello), sino también la disposición a perseverar sobre algo que es a menudo difícil y frustrante. Algunos ven en la experiencia de jugar a ciertos videojuegos desafiantes algo similar al *'coding'*: al jugar a un videojuego hay un continuo ciclo retroalimentación entre causas y efectos de la conducta del jugador, similar a cuando un programador codifica y depura. Ambos, *'coders'* y *'gamers'* experimentan una sensación de flujo en la cual permanecen absorbidos y focalizados en esa única tarea; lo que probablemente les ayuda a motivarse y perseverar hasta acabar el juego o resolver el problema.

- **Colaboración (*'Collaborating'*):** colaborar significa trabajar con otros para asegurar un mejor resultado. Es complicado pensar en cualquier trabajo o actividad de ocio en la sociedad digital actual que no involucre la colaboración. El trabajo colaborativo tiene además una larga tradición en nuestro sistema educativo (especialmente en la Educación Primaria), y la computación no debería suponer un cambio al respecto. Al programar, muchos ven el *'pair programming'* ('programación por parejas') con una forma particularmente eficaz de escribir código, con dos programadores compartiendo una pantalla y un teclado, trabajando juntos para crear software. Habitualmente, uno en la pareja actúa como 'conductor' (*'driver'*, el que teclea y presta atención al detalle del código), mientras que el otro toma el papel de 'navegador' (*'navigator'*, el que tiene en mente una visión amplia del problema, el 'paisaje completo'). Los dos miembros de la pareja intercambian regularmente sus roles, de manera que ambos tengan contacto con el código detallado y la visión de conjunto [5].

Anexo 2: Modelo Mit-Harvard

Conceptos computacionales (*'computational thinking concepts'*) **[¿Qué aprenden?]**: según los niños aprenden a programar, diseñando objetos digitales interactivos con Scratch, toman contacto con un conjunto de conceptos computacionales (alineados con los tipos de bloques de programación que ofrece el lenguaje Scratch), que son comunes a la mayoría de los lenguajes de programación. Brennan y Resnick (2012) han identificado 7 conceptos de alto uso y utilidad en un amplio rango de proyectos Scratch, y que pueden ser transferidos a otros entornos, tanto de programación como de resolución de problemas en general [20]:

- **Secuencias** (*'sequences'*): un concepto clave en programación es identificar y expresar una actividad o tarea como una serie de pasos individuales (discretos) y ordenados, que puedan ser ejecutados por un ordenador. Como una receta de cocina, una secuencia de instrucciones de programación específica el comportamiento o acción que debe producirse.
- **Bucles** (*'loops'*): los bucles permiten ejecutar una misma secuencia de instrucciones en múltiples ocasiones; convirtiendo a los programas en expresiones más sucintas y elegantes. Así, en la Figura 4 vemos a la izquierda una secuencia de 9 instrucciones en lenguaje Scratch; a la derecha, la misma secuencia escrita de manera más sucinta al utilizar un bucle dentro del programa.

Figura 5: Una secuencia con instrucciones repetidas (a la izquierda), expresada a través de un bucle (a la derecha)

- **Eventos** (*'events'*): se refiere a 'cuando algo pasa, entonces causa (dispara) que otra cosa ocurra'. Los eventos son componentes esenciales de los objetos programados interactivos. Por ejemplo, un botón de *'start'* que está programado para que, cuando sea pulsado, dispare el inicio de un vídeo musical; o un evento consistente en que, cuando una tecla determinada es presionada, un personaje se mueva en una dirección dada. Ver figura 5.

Figura 6: Eventos disponibles en Scratch.

- **Paralelismos** (*'parallelism'*): son varias secuencias de instrucciones que se ejecutan al mismo tiempo, simultáneamente ('en paralelo'). La mayoría de lenguajes de programación modernos soportan paralelismos.

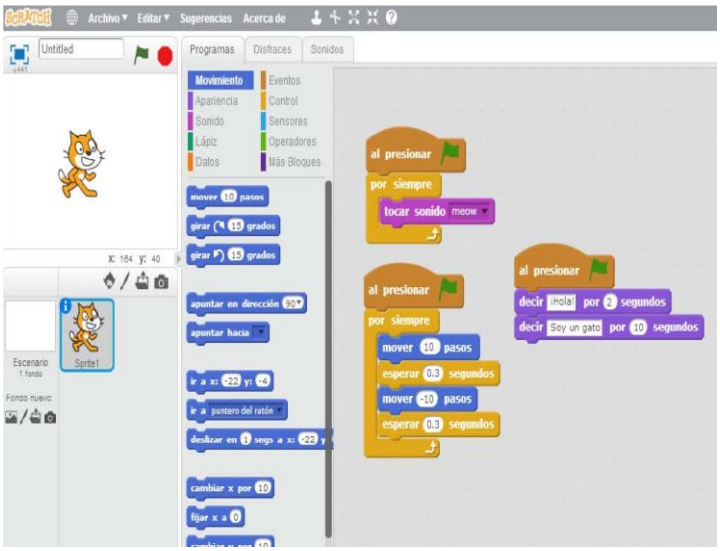


Figura 7: Ejemplo de 3 secuencias ejecutadas en paralelo en respuesta al mismo evento ('al presionar bandera verde').

- **Condicionales** (*'conditionals'*): otro concepto computacional clave en programación son los condicionales, es decir, la capacidad de tomar decisiones basadas en ciertos estados o situaciones. Ello permite a un programa expresar distintos resultados en función de las condiciones dadas, dotándolo de versatilidad. .

- **Operadores** (*'operators'*): los operadores permiten al programador incluir expresiones lógicas, matemáticas y de cadena, en sus programas. Scratch soporta una amplia variedad de operadores.

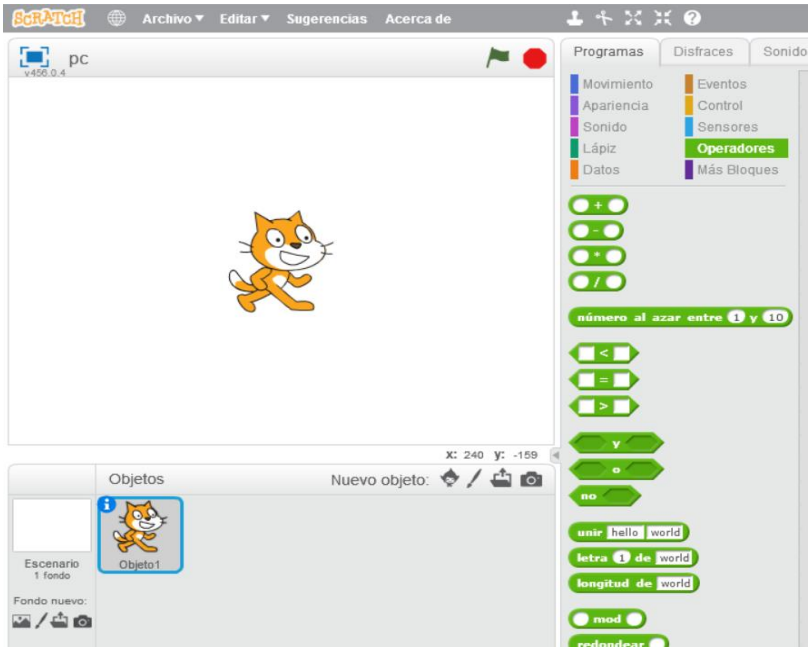


Figura 8: Operadores disponibles en Scratch

- **Datos** (*'data'*): incluye el almacenamiento, recuperación y actualización de valores en un programa. Scratch, por ejemplo, ofrece dos tipos de contenedores de datos: 'variables' (que pueden almacenar, recuperar y actualizar un solo número; algo que se utiliza, por ejemplo, para hacer un marcador de tantos en un videojuego que va actualizando la puntuación del jugador) y 'listas' (que pueden almacenar, recuperar y actualizar conjuntos de números).

Prácticas computacionales (*'computational thinking practices'*) **[¿Cómo lo aprenden?]**: si los conceptos computacionales se enfocan al contenido de aprendizaje (lo que el niño aprende al programar), las prácticas computacionales se focalizan en cómo lo aprenden. Es decir, las prácticas computacionales se refieren a qué tipo de procesos y prácticas ponen en marcha los niños cuando construyen sus programas. Brennan y Resnick (2012) identifican 4 tipos de prácticas:

- **Experimentación e iteración** (*'experimenting and iterating'*): diseñar-programar un proyecto no es un proceso absolutamente 'limpio' y lineal, en el cual primero se concibe una idea, después se desarrolla un plan para su diseño, y finalmente se implementa dicho diseño en forma de código.

- **Evaluación y depuración** (*'testing and debugging'*): los programas raramente funcionan a la primera y en la forma que fueron imaginados. Para los diseñadores-programadores es crítico desarrollar estrategias para anticipar y afrontar los errores que aparecen en sus programas. Según informan los propios 'scratchers', durante el proceso de construcción de sus programas ponen en marcha espontáneamente procedimientos de evaluación y depuración de los mismos a través de sucesivas pruebas de 'ensayo y error', y búsqueda de apoyo en otros pares de la comunidad con mayor nivel de pericia.
- **Reutilización y remezcla** (*'reusing and remixing'*): es decir, escribir un programa a partir de otros programas preexistentes. Construir a partir del trabajo de otros es una práctica habitual con larga tradición en el mundo de la programación; que ha sido amplificada a partir de que las tecnologías digitales se han conectado en red a través de Internet, proporcionando acceso a una enorme cantidad de trabajo de otros programadores listo para ser usado y remezclado. Uno de los objetivos de la comunidad en línea de Scratch es precisamente ayudar a los jóvenes diseñadores programadores a reusar y remezclar, orientándoles para encontrar ideas y código sobre el que puedan construir nuevos programas; capacitándoles para crear objetos digitales mucho más complejos de lo que hubieran podido abordar por sí mismos. La reutilización y la remezcla promueven además la habilidad de 'lectura crítica' del código escrito por otros, y abren preguntas importantes acerca de la propiedad y la autoría de los programas.
- **Abstracción y modularización** (*'abstracting and modularizing'*): la abstracción y la modularización, que podemos caracterizar como el proceso de construir algo de gran tamaño a partir de ir agregando conjuntos de elementos más pequeños, es una práctica muy importante para la solución de problemas en general, y para el diseño-programación en particular. En Scratch es muy habitual desarrollar proyectos muy complejos a partir de agregaciones de conjuntos de código más sencillos.

Perspectivas computacionales (*'computational thinking perspectives'*) **[¿Para qué lo aprenden?]**: a lo largo de sus entrevistas con *'scratchers'*, Brennan y Resnick (2012) recogen frecuentes testimonios de los jóvenes diseñadores-programadores relatando cómo, a partir de su aprendizaje con Scratch, ha evolucionado su comprensión de sí mismos, de la relación con los otros, y del mundo digital-tecnológico que les rodea. Así, los autores añaden

una tercera dimensión a su modelo, las *perspectivas computacionales*, que se centran en los 'cambios de perspectiva' que experimentan los niños a partir de su aprendizaje con Scratch. Distinguen entre:

- **Expresarse** (*'expressing'*): estamos rodeados de objetos digitales interactivos, pero la mayoría de nuestras experiencias con dichos objetos son como 'consumidores'. Un 'pensador computacional' concibe la tecnología como algo más que objetos de consumo; la computación es un medio que puede ser utilizado para la creación, el diseño y la autoexpresión.
- **Conectarse** (*'connecting'*): los jóvenes *'scratchers'* describen el poder que supone tener acceso, a través de la comunidad *'on-line'* de Scratch, a otros jóvenes programadores, a sus proyectos e ideas: *'I can do different things when I have 119roble to others'* ('Puedo hacer cosas distintas cuando tengo acceso a los otros'). Este 'tener acceso a otros' tendría una doble dimensión:
  - El valor de crear *con* otros (reusando y remezclando sus creaciones; o programando colaborativamente con otros; o consultando a otros cuando no se sabe cómo avanzar)
  - Y el valor de crear *para* otros, al tener acceso a audiencias reales que pueden disfrutar de tus diseños-programas (entreteniéndoles con tus creaciones y, a la vez, equipándoles con nuevos conjuntos de código que pueden reutilizar)
- **Interrogarse** (*'questioning'*): se observa que, a través de la programación, los jóvenes *'scratchers'* se sienten empoderados a hacerse preguntas acerca del mundo digital que les rodea. Al aprender a escribir código, aprenden también a no dar por sentado los objetos digitales que consumen a diario, y a preguntarse por su naturaleza y por los posibles cambios que se podrían introducir en los mismos.

Anexo 3: Pensamiento algorítmico problema resuelto #1 (Torres de Hanói)

Todos los discos están en una torre que es la torre origen, y desde aquí empezaremos los procedimientos de trasladar los discos de una torre a otra hasta que estén todos en la torre destino.

Por lo tanto Torre A = 3 discos, Torre B = 0, Torre C = 0

Paso 1: ¿Hay alguna torre vacía?

Si sí, hacer n-1 disco de la torre origen a la torre que este vacía

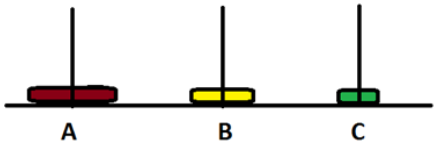
Paso 2: ¿Hay torre vacía?

Si sí y además A = 2

Entonces hacer n-1 disco de la torre A para la Torre que este vacía.

Devolver: A = 1, B = 1, C = 1

Si no repetir procedimiento con B y C,



Paso 3:

Si cantidad de disco en la torre A = cantidad de disco en la torre B = cantidad de disco en la torre C, o sea,

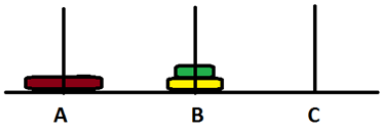
A = B = C = 1

Entontes hacer n-1 disco de la torre C a la torre B

Devolver: A= 1 B= 2 C= 0

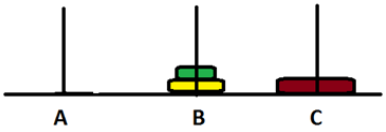
Si no, hacer n-1 de la torre B a la torre C

Devolver: A= 1 B= 0 C= 2



Paso 4: ¿Hay alguna torre vacía?

Si sí, entonces hacer n-1 disco de la torre A para la torre que este vacía



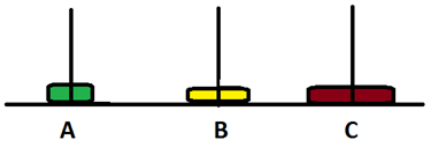
Paso 5: ¿Hay alguna torre vacía?

Si sí y además esta es la torre A y, Hay alguna torre con más de 2 discos y esta es la torre B,

Entonces hacer n-1 disco de la torre B a la torre A.

Si no, si la torre con más de 2 discos es la C,

Entonces hacer n1 disco de la torre c a la torre A.



Paso 6:

Si cantidad de disco en la torre A = cantidad de disco en la torre B = cantidad de disco en la torre C, o sea,

A = B = C = 1

Entontes hacer n-1 disco de la torre B a la torre C

Devolver: A= 1 B= 0 C= 2

Paso 7: ¿Hay disco en la torre A?

Si sí, entonces hacer n-1 de la torre A para la torre C.

Devolver: A= 0 B= 0 C= 3

Todos los discos están en la torre destino C.

Fin del procedimiento.

Anexo 4 Pensamiento algorítmico del problema resuelto #2 (Misioneros y caníbales).

Paso 1: ¿El bote está vacío? ¿Todos los pasajeros están en la orilla derecha?

Si sí, entonces montar un caníbal y un misionero al bote y cruzarlos hacia la otra orilla del rio, dejar el caníbal en la orilla izquierda del rio y regresar con el misionero para manejar el bote.

Paso 2:

Si cantidad de caníbales en la orilla izquierda es ≥ 1,

Entonces cruzar 2 caníbales a la otra orilla del rio.

Dejar uno en la orilla izquierda y regresar con uno en el bote para manejarlo.

Paso 3:

Si cantidad de misioneros en la orilla derecha es mayor que cantidad de caníbales,

Entonces cruzar 2 misioneros a la orilla izquierda del rio.

Dejar un misionero en la orilla izquierda y regresar con un misionero y un caníbal en el bote.

Paso 4:

Si cantidad de misioneros en la orilla derecha es igual a la cantidad de caníbales,

Entonces cruzar 2 misioneros a la orilla izquierda del rio.

Dejar los 2 misioneros en la orilla izquierda y regresar con un caníbal en el bote.

Paso 5:

Si cantidad de misioneros en la orilla izquierda es mayor o igual que cantidad de caníbales en la orilla derecha, entonces, cruzar dos caníbales a la orilla izquierda del rio, dejar uno y regresar con uno en el bote.

Paso 6: ¿Queda pasajeros en la orilla derecha del rio?

¿Si sí, estos pasajeros son caníbales?

Si cantidad de pasajeros en la orilla derecha del rio es = 2, entonces cruzar dos caníbales de la orilla derecha a la orilla izquierda del rio.

Paso 7: ¿Queda algún pasajero en la orilla derecha del rio?

Si no, fin del procedimiento y devolver:

Cantidad de misioneros en la orilla izquierda = 3.

Cantidad de caníbales en la orilla izquierda = 3.

Anexo 5: Pensamiento algorítmico del problema resuelto # 3 (El arriero)

Asumiendo que tanto el bote como el arriero y sus pertenencias están en la orilla derecha, tenemos:

Paso 1: ¿El bote está vacío?

¿Todas las pertenencias están en la orilla derecha?

Si sí, entonces montar el arriero y la cabra al bote y cruzar hacia la orilla izquierda del rio, dejar la cabra en esa misma orilla regresar.

Devolver:

Estado actual en la orilla izquierda

C = 1 P = 0 L = 0

Estado actual en la orilla derecha

C = 0 P = 1 L = 1

Paso 2:

Montar la lechuga al bote y trasladarla de la orilla derecha hacia la orilla izquierda del rio.

Llegando a la orilla izquierda, dejar la lechuga y llevarse la cabra a la otra orilla del rio.

Estado actual en la orilla izquierda

C = 0 P = 0 L = 1

Estado actual en la orilla derecha

C = 0 P = 0 L = 0

Paso 3:

Dejar la cabra en la orilla derecha y montar al puma en el bote para trasladarlo hacia la orilla izquierda.

Llegando a la orilla izquierda, dejar el puma y volver a la orilla derecha.

Estado actual en la orilla izquierda

C = 0 P = 1 L = 1

Estado actual en la orilla derecha

C = 1 P = 0 L = 0

Paso 4:

Llegando a la orilla derecha, montar la lechuga al bote y trasladarla hacia la orilla izquierda del rio.

Llegando a la orilla izquierda del rio, dejar la cabra.

Estado actual en la orilla izquierda

C = 1 P = 1 L = 1

Estado actual en la orilla derecha

C = 0 P = 0 L = 0

Fin del traslado de pertenencias del arriero.

Estado final u objetivo logrado en 4 viajes.

Anexo 6: Pensamiento algorítmico del problema resuelto # 4 (Jarras de agua (8, 5, 3 litros respectivamente))

Paso 1:

Llenar la jarra de 3 litros e inmediatamente pasar el contenido a la de 5 litros.

Las jarras quedan de la siguiente forma:

Jarra de 8 = 5 litros; Jarra de 5 litros = 3 litros; Jarra de 3 litros = vacía

Paso 2:

Llenar la jarra de 3 litros con el contenido de la jarra de 8 litros.

Como resultado tenemos:

Jarra de 8 litros = 2 litros; Jarra de 5 litros = 3 litros; Jarra de 3 = 3 litros

Paso 3:

Coger el agua de la jarra de 3 litros y añadirla a los 3 litros que tenía la jarra de 5 litros hasta llenarla. Quedando así:

Jarra de 8 litros = 2 litros; Jarra de 5 litros = 5 litros; Jarra de 3 litros = 1 litros

Paso 4:

Pasar toda el agua de la jarra de 5 litros a la de 8, quedando:

Jarra de 8 = 7 litros; Jarra de 5 litros = vacía; Jarra de 3 litros = 1 litros

Paso 5:

Pasar el agua de la jarra de 3 litros a la de 5 litros, quedando:

Jarra de 8 = 7 litros; Jarra de 5 litros = 1 litros; Jarra de 3 litros = vacía

Paso 6:

De los 7 litros que hay en la jarra de 8 litros, coger y llenar la de 3 litros, quedando entonces:

Jarra de 8 litros = 4 litros; Jarra de 5 litros = 1 litros; Jarra de 3 litros = 3 litros

Paso 7:

Coger el agua de la jarra de 3 litros y pasarla a la de 5 litros. Quedan entonces las jarras de la siguiente forma:

Jarra de 8 = 4 litros; Jarra de 5 = 4 litros; Jarra de 3 = vacía

Estado final u Objetivo: Las jarras de 4 y 8 litros quedan cada una con 4 litros exactos.

Anexo 7: Pensamiento algorítmico del problema resuelto #5 (Jarras de agua (4 y 3 litros respectivamente))

Paso 1:

Llenar la jarra de 3 litros con la bomba e inmediatamente pasar el contenido a la de 4 litros.

Devolver estado actual de las jarras:

Jarra de 4 litros = 0, Jarra de 3 litros = 3 litros

Paso 2:

Vaciar la jarra de 3 litros y pasar el contenido a la de 4 litros

Devolver estado actual de las jarras:

Jarra de 4 litros = 3 litros, Jarra de 3 litros = vacía

Paso 3:

Llenar la jarra de 3 litros

Devolver estado actual de las jarras:

Jarra de 4 litros = 3 litros, Jarra de 3 litros = 3 litros

Paso 4:

Vaciar la jarra de 3l y pasar el agua a la de 4 litros hasta llenarla

Devolver estado actual de las jarras:

Jarra de 4 litros = 4 litros, Jarra de 3 litros = 2 litros

Paso 5:

Vaciar la jarra de 4 litros y botar toda el agua para el piso.

Devolver estado actual de las jarras:

Jarra de 4 litros = vacía, Jarra de 3 litros = 2 litros

Paso 6:

Vaciar la jarra de 3 litros y pasar toda el agua a la de 4 litros

Devolver estado actual de las jarras:

Jarra de 4 litros = 2 litros, Jarra de 3 litros = vacía
Estado final: La jarra de 4 litros tiene la cantidad de agua deseada que son 2 litros exactos.

Anexo 8: Pensamiento algorítmico del problema resuelto # 6: Límite de función real de una variable real

Paso 1:

Evaluar la función $f(x)$ en un punto $x_0 = 0$ (determinar $f(0)$)

Se obtiene:

$\lim_{x\to 0^-} 2x * e^{1/x} = 0$

$\lim_{x\to 0^+} 2x * e^{1/x} = 0.\infty$

Paso 2:

Si ($f(0)$ = #) o ($f(0)$ = -∞) o ($f(0)$ = +∞)

Terminar

Si no

Si $f(0)$ = $\left\{\frac{0}{0}; \frac{\infty}{\infty};\ \infty - \infty; 0 * \infty; 1^{\infty};\ \infty^0;\ 0^0\right\}$)

Resolver indeterminación

Si $(f(-\infty) = \#)$ o $(f(-\infty) = -\infty)$ o $(f(0) = +\infty)$

Terminar la condición

Devolver $\lim_{x\to 0} 2x * e^{\frac{1}{x}} = f(0)$

Terminar procedimiento

Si no, si $f(0) \in \left\{\frac{0}{0}; \frac{\infty}{\infty};\ \infty - \infty; 0 * \infty; 1^{\infty};\ \infty^0;\ 0^0\right\}$

Devolver “forma indeterminada” y

Resolver indeterminación

Resolviendo indeterminación:

$\lim_{x\to+\infty} 2x * e^{1/x} : [0 * \infty]$

$= 2 * \lim_{x\to+\infty} \frac{e^{\frac{1}{x}}}{1/x} : [\infty/\infty]$

$$= 2 * \lim_{x \to +\infty} \frac{(e^{\frac{1}{x}})'}{(\frac{1}{x})'} = 2 * \lim_{x \to +\infty} \frac{e^{1/x} * (-\frac{1}{x^2})}{(-\frac{1}{x^2})}$$

$$= 2 * \lim_{x \to +\infty} e^{1/x} = +\infty$$

Terminar

$$\lim_{x \to -\infty} f(x) = 0 \neq \lim_{x \to +\infty} f(x) = +\infty$$

Ir al siguiente paso

Paso 3:

Si $\lim_{x \to -\infty} f(x) = \lim_{x \to +\infty} f(x) = L$

Terminar

Devolver L

Si no

Devolver (Ø "no existe el límite").

$$\{ \lim_{x \to -\infty} 2x * e^{1/x} = 0$$

Terminar

Anexo 9: Pensamiento algorítmico del problema resuelto # 10: Problema de optimización de una función real de una variable real

Paso 1: determinar a partir de un modelo matemático lo que se quiere optimizar.

Paso 2: determinar las relaciones existentes entre las variables independientes del modelo anterior a fin de expresarlas una en función de otra.

Paso 3: plantear la función objetivo (modelo matemático $[f(x)]$ a optimizar en términos de una sola variable).

Paso 4: derivar (primera derivada) la función objetivo $[f(x)]$.

Paso 5: determinar posibles extremos soluciones de la ecuación $F'(x) = 0$.

Paso 6: efectuar CNE, determinar $F'(x)$.

Paso 7: evaluar $F''(x)$ y los posibles extremos del paso anterior.

Paso 8: aplicar el criterio de la segunda derivada.

$$\left[\begin{array}{c} repetir\ para\ cada\ x_0\ posible\ extremo: \\ si\ f''(x_0) > 0 entonces\ x_0\ es\ un\ punto\ mínimo\ local \\ f\ (x_0) = min \\ si\ no \\ si\ f''(x_0) < 0\ entonces\ x_0\ es\ un\ punto\ de\ maximo \\ terminar \\ si\ no \\ si\ f''(x_0) = 0\ entonces\ x_0\ no\ es\ un\ máximo\ ni\ mínimo \\ terminar \end{array}\right]$$

En ΔABC rectángulo en A se tiene que:

Paso 1:

$\overline{BC} = \sqrt{(\overline{AB})^2 + (\overline{AB})^2}$ Teorema de Pitágoras.

Paso 2:

$\frac{\overline{PC}}{\overline{AC}} = \frac{\overline{PQ}}{\overline{AB}}$ Lados homólogos de Δ semejantes

$\overline{AC} = 4\overline{PC}$

$$\frac{\overline{PC}}{4\overline{PC}} = \frac{8}{\overline{AB}} \rightarrow \overline{AB} = \frac{8\,(4 + \overline{PC})}{\overline{PC}}$$

Paso 3:

$$\overline{BC} = \sqrt{(\frac{8(4+\overline{PC})}{\overline{PC}})^2 + (4+\overline{PC})^2}$$

$F(x) = \frac{64*(4+x)^2}{x^2} + (4+x)^2$ ***Función objetivo***

Paso 4:

$$F'(x) = \left[64 * \left(\frac{4}{x}+1\right)^2 + (4+x)^2\right]'$$

$$F'(x) = 64 * 2 * \left(\frac{4}{x}+1\right) * \left(\frac{1}{x}+x\right)' + 2 * (4x)$$

$$F'(x) = 64 * 2 * \left(\frac{4}{x}+1\right) * \left(\frac{-1}{x^2}\right) + 2 * (4+x)$$

$$F'(x) = 512 * \left(\frac{4}{x}+1\right) + 8 + 2x$$

$$F'(x) = \frac{512 * 4}{x^3} - \frac{512}{x^2} + 8 + 2x$$

Paso 5:

$F'(x) = 0$

$$\frac{(-512) * 4}{x^3} - \frac{512}{x^2} + 8 + 2x = 0$$

$$-512 * 4 - 512 * x + 8x^3 + 2x^3 = 0$$

⋮

Paso 6:

$$F''(x) = (\frac{512 * 4}{x^3} - \frac{512}{x^2} + 8 + 2x)'$$

Paso 7: evaluar $F''(x_0)$

Paso 8: aplicar criterio de la segunda derivada.

Anexo 10: Pensamiento algorítmico del problema resuelto #16 (Traducción al lenguaje natural)

Si las premisas ya están formadas correctamente, entonces hacer el paso 1

Paso 1: Traducir cada proposición en términos de premisas y conclusiones, escribiendo primero las premisas y luego las conclusiones.

“para traducir una proposición, hay que identificar primero cuales son las proposiciones atómicas”

a)Identificar las proposiciones atómicas:

p: no me preparo;

q: tengo dificultades;

r: me evalúo de 2;

s: tengo criterio de mal

t: fallo la asignatura;

Si las proposiciones atómicas ya están identificadas correctamente entonces ir al siguiente paso.

b)Identificar las conectivas en cada proposición:

Para identificar la conectiva a utilizar en cada una de las proposiciones atómicas formadas, hay que saber el significado de cada conector lógico:

Negación (~): no

Conjunción (∧): y

Disyunción (∨): o

Condicional (→): implicación

Si ya conoces el significado de los conectores y además ya sabes cual usar en cada una de las proposiciones, entonces ir al paso siguiente:

c) Escribir las formulas correspondientes:

Premisas:

P1, P2: $\sim p$

P3: $p \rightarrow q$

P4: $q \rightarrow r$

P5: $r \rightarrow s$

P6: $s \rightarrow t$

Conclusión:

Q1: q

Si las formulas están formadas correctamente, entonces ir al paso 2,

Sino, repetir procedimiento.

Paso 2: Escribir la estructura deductiva correspondiente según las formulas formadas en el paso 1.

$\sim p, p \rightarrow q, q \rightarrow r, r \rightarrow s, s \rightarrow t \Rightarrow q$

Si la estructura deductiva es correcta, mostrarla y terminar procedimiento.

Anexo 11: Prueba Inicial

Al resolver este examen usted contribuye con la investigación “DESARROLLO DEL PENSAMIENTO COMPUTACIONAL EN ESTUDIANTES DE PRIMER AÑO DE INGENEIRÍA INFORMÁTICA”

NOMBRE Y APELLIDOS: **FECHA:**

PRUEBA INICIAL

PROBLEMA # 1

Se tienen 3 misioneros y 3 caníbales en la orilla derecha de un río. Existe una canoa con capacidad para dos personas como máximo. Se desea que los seis pasen al margen izquierdo del río, pero hay que considerar que no debe haber más caníbales que misioneros en ninguna de las dos orillas del rio porque entonces los caníbales se comen a los misioneros. Además, la canoa siempre debe ser conducida por alguien.

PROBLEMA # 2

Determinar la derivada n-ésima (orden n) de la siguiente función: $f(x) = \frac{1}{x}$

PROBLEMA # 3

Un robot puede dar pasos de $1\ o\ 2$ metros. Escribir un algoritmo recursivo para calcular el número de formas en que puede recoger n metros.

SU PARTICIPACIÓN EN ESTA INVESTIGACIÓN SERÁ TOMADA EN CUENTA EN SU EXPEDIENTE (CURRICULUM) DE ESTUDIANTE.

¡GRACIAS POR SU COLABORACIÓN!

Anexo 12: Prueba Final

Al resolver este examen usted contribuye con la investigación “DESARROLLO DEL PENSAMIENTO COMPUTACIONAL EN ESTUDIANTES DE PRIMER AÑO DE INGENEIRÍA INFORMÁTICA”

NOMBRE Y APELLIDOS: **FECHA:**

PRUEBA FINAL

PROBLEMA # 1

Se tienen dos jarras sin escala de medición, una de 4 litros de capacidad y otra de 3 litros. Además, se tiene una bomba que permite llenar las jarras de agua. Se desea tener 2 litros de agua en la jarra de 4 litros de capacidad. Las siguientes operaciones son válidas: llenar las jarras, tirar agua de las jarras, pasar el agua de una jarra a otra.

PROBLEMA # 2

Determinar la derivada n-ésima (orden n) de la siguiente función: $f(x) = x * e^{-x}$

PROBLEMA # 3

Plantear una forma genérica para calcular el factorial de un número $n > 0$.

SU PARTICIPACIÓN EN ESTA INVESTIGACIÓN SERÁ TOMADA EN CUENTA EN SU EXPEDIENTE (CURRICULUM) DE ESTUDIANTE.

¡GRACIAS POR SU COLABORACIÓN!

Anexo 13: Resultados contraste pi y pf # 1 g. exp..spo

Pruebas no paramétricas

Notas

Resultados creados		09-JUN-2018 15:12:11
Comentarios		
Entrada	Datos	C:\Users\Niñas\Desktop\RESULTADOS ISABEL\DATOS #1 PI Y PF_GE.sav
	Conjunto de datos activo	Conjunto_de_datos1
	Filtro	<ninguna>
	Peso	<ninguna>
	Segmentar archivo	<ninguna>
	Núm. de filas del archivo de trabajo	9
Manipulación de los valores perdidos	Definición de los perdidos	Los valores perdidos definidos por el usuario serán tratados como perdidos.
	Casos utilizados	Los estadísticos para cada prueba se basan en todos los casos con datos válidos para las variables usadas en dicha prueba.
Sintaxis		NPAR TEST /SIGN= X1 X2 X3 X4 X5 X6 X7 WITH X8 X9 X10 X11 X12 X13 X14 (PAIRED) /MISSING ANALYSIS.
Recursos	Tiempo de procesador	0:00:00.00
	Tiempo transcurrido	0:00:00.06
	Número de casos permitidos(a)	46290

a Basado en la disponibilidad de memoria en el espacio de trabajo.

[Conjunto_de_datos1] C:\Users\Niñas\Desktop\RESULTADOS ISABEL\DATOS #1 PI Y PF_GE.sav

Prueba de los signos

Frecuencias

			N
PF#1_Ab_GE - PI#1_Ab_GE		Diferencias negativas(a,b,c,d,e,f,g)	0

		Diferencias positivas(h,i,j,k,l,m,n)	8
		Empates(o,p,q,r,s,t,u)	1
		Total	9
PF#1_AD_GE PI#1_AD_GE	-	Diferencias negativas(a,b,c,d,e,f,g)	2
		Diferencias positivas(h,i,j,k,l,m,n)	3
		Empates(o,p,q,r,s,t,u)	4
		Total	9
PF#1_DP_GE PI#1_DP_GE	-	Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	8
		Empates(o,p,q,r,s,t,u)	1
		Total	9
PF#1_Al_GE - PI#1_Al_GE		Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	8
		Empates(o,p,q,r,s,t,u)	1
		Total	9
PF#1_P_GE - PI#1_P_GE		Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	9
		Empates(o,p,q,r,s,t,u)	0
		Total	9
PF#1_S_GE - PI#1_S_GE		Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	8
		Empates(o,p,q,r,s,t,u)	1
		Total	9
PF#1_NT_GE PI#1_NT_GE	-	Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	9
		Empates(o,p,q,r,s,t,u)	0
		Total	9

a PF#1_Ab_GE < PI#1_Ab_GE
b PF#1_AD_GE < PI#1_AD_GE
c PF#1_DP_GE < PI#1_DP_GE
d PF#1_Al_GE < PI#1_Al_GE
e PF#1_P_GE < PI#1_P_GE
f PF#1_S_GE < PI#1_S_GE
g PF#1_NT_GE < PI#1_NT_GE
h PF#1_Ab_GE > PI#1_Ab_GE
i PF#1_AD_GE > PI#1_AD_GE
j PF#1_DP_GE > PI#1_DP_GE
k PF#1_Al_GE > PI#1_Al_GE
l PF#1_P_GE > PI#1_P_GE
m PF#1_S_GE > PI#1_S_GE
n PF#1_NT_GE > PI#1_NT_GE

o PF#1_Ab_GE = PI#1_Ab_GE
p PF#1_AD_GE = PI#1_AD_GE
q PF#1_DP_GE = PI#1_DP_GE
r PF#1_Al_GE = PI#1_Al_GE
s PF#1_P_GE = PI#1_P_GE
t PF#1_S_GE = PI#1_S_GE
u PF#1_NT_GE = PI#1_NT_GE

Estadísticos de contraste (b)

	PF#1_Ab_GE - PI#1_Ab_GE	PF#1_AD_GE - PI#1_AD_GE	PF#1_DP_GE - PI#1_DP_GE	PF#1_Al_GE - PI#1_Al_GE	PF#1_P_GE - PI#1_P_GE	PF#1_S_GE - PI#1_S_GE	PF#1_NT_GE - PI#1_NT_GE
Sig. exacta (bilateral)	.008(a)	1.000(a)	.008(a)	.008(a)	.004(a)	.008(a)	.004(a)

a Se ha usado la distribución binomial.
b Prueba de los signos

```
SAVE OUTFILE='C:\Users\Niñas\Desktop\RESULTADOS ISABEL\Datos #2 PI y PF_ GE.sav'
 /COMPRESSED.
```

PF#1_Ab_GE: Nota del componente Abstracción de la pregunta No 1 en la Prueba Final del GE.

PI#1_Ab_GE: Nota del componente Abstracción de la pregunta No 1 en la Prueba Inicial del GE.

PF#1_AD_GE: Nota del componente Análisis y representación de datos de la pregunta No 1 en la Prueba Final del GE.

PI#1_AD_GE: Nota del componente Análisis y representación de datos de la pregunta No 1 en la Prueba Inicial del GE.

PF#1_DP_GE: Nota del componente Descomposición de la pregunta No 1 en la Prueba Final del GE.

PI#1_DP_GE: Nota del componente Descomposición del problema de la pregunta No 1 en la Prueba Inicial del GE.

PF#1_Al_GE: Nota del componente Pensamiento algorítmico de la pregunta No 1 en la Prueba Final del GE.

PI#1_Al_GE: Nota del componente Pensamiento algorítmico de la pregunta No 1 en la Prueba Inicial del GE.

PF#1_P_GE: Nota del componente Reconocimiento y generalización de patrones de la pregunta No 1 en la Prueba Final del GE.

PI#1_P_GE: Nota del componente Descomposición y generalización de patrones de la pregunta No 1 en la Prueba Inicial del GE.

PF#1_S_GE: Nota del componente Simulación de la pregunta No 1 en la Prueba Final del GE.

PI#1_S_GE: Nota del componente Simulación de la pregunta No 1 en la Prueba Inicial del GE.

PF#1_Ab_GE: Nota total de la pregunta No 1 en la Prueba Final del GE.

PI#1_NT_GE: Nota total de la pregunta No 1 en la Prueba Inicial del GE.

Anexo 14: Resultados contraste pi y pf # 2 g. exp..spo

```
NPAR TEST
  /SIGN= X1 X2 X3 X4 X5 X6 X7  WITH X8 X9 X10 X11 X12 X13 X14 (PAIRED)
  /MISSING ANALYSIS.
```

Pruebas no paramétricas

Notas

Resultados creados		09-JUN-2018 15:28:08
Comentarios		
Entrada	Datos	C:\Users\Niñas\Desktop\RESULTADOS ISABEL\Datos #2 PI y PF_GE.sav
	Conjunto de datos activo	Conjunto_de_datos1
	Filtro	<ninguna>
	Peso	<ninguna>
	Segmentar archivo	<ninguna>
	Núm. de filas del archivo de trabajo	9
Manipulación de los valores perdidos	Definición de los perdidos	Los valores perdidos definidos por el usuario serán tratados como perdidos.
	Casos utilizados	Los estadísticos para cada prueba se basan en todos los casos con datos válidos para las variables usadas en dicha prueba.
Sintaxis		NPAR TEST /SIGN= X1 X2 X3 X4 X5 X6 X7 WITH X8 X9 X10 X11 X12 X13 X14 (PAIRED) /MISSING ANALYSIS.
Recursos	Tiempo de procesador	0:00:00.00
	Tiempo transcurrido	0:00:00.02
	Número de casos permitidos(a)	46290

a Basado en la disponibilidad de memoria en el espacio de trabajo.

```
[Conjunto_de_datos1] C:\Users\Niñas\Desktop\RESULTADOS ISABEL\Datos #2 PI y PF_GE.sav
```

Prueba de los signos

Frecuencias

			N
PF#2_Ab_GE - PI#2_Ab_GE		Diferencias negativas(a,b,c,d,e,f,g)	0

		Diferencias positivas(h,i,j,k,l,m,n)	5
		Empates(o,p,q,r,s,t,u)	4
		Total	9
PF#2_AD_GE PI#2_AD_GE	-	Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	7
		Empates(o,p,q,r,s,t,u)	2
		Total	9
PF#2_DP_GE PI#2_DP_GE	-	Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	8
		Empates(o,p,q,r,s,t,u)	1
		Total	9
PF#2_Al_GE - PI#2_Al_GE		Diferencias negativas(a,b,c,d,e,f,g)	1
		Diferencias positivas(h,i,j,k,l,m,n)	6
		Empates(o,p,q,r,s,t,u)	2
		Total	9
PF#2_P_GE - PI#2_P_GE		Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	7
		Empates(o,p,q,r,s,t,u)	2
		Total	9
PF#2_S_GE - PI#2_S_GE		Diferencias negativas(a,b,c,d,e,f,g)	2
		Diferencias positivas(h,i,j,k,l,m,n)	5
		Empates(o,p,q,r,s,t,u)	2
		Total	9
PF#2_NT_GE PI#2_NT_GE	-	Diferencias negativas(a,b,c,d,e,f,g)	0
		Diferencias positivas(h,i,j,k,l,m,n)	8
		Empates(o,p,q,r,s,t,u)	1
		Total	9

a PF#2_Ab_GE < PI#2_Ab_GE
b PF#2_AD_GE < PI#2_AD_GE
c PF#2_DP_GE < PI#2_DP_GE
d PF#2_Al_GE < PI#2_Al_GE
e PF#2_P_GE < PI#2_P_GE
f PF#2_S_GE < PI#2_S_GE
g PF#2_NT_GE < PI#2_NT_GE
h PF#2_Ab_GE > PI#2_Ab_GE
i PF#2_AD_GE > PI#2_AD_GE
j PF#2_DP_GE > PI#2_DP_GE
k PF#2_Al_GE > PI#2_Al_GE
l PF#2_P_GE > PI#2_P_GE
m PF#2_S_GE > PI#2_S_GE
n PF#2_NT_GE > PI#2_NT_GE

o PF#2_Ab_GE = PI#2_Ab_GE
p PF#2_AD_GE = PI#2_AD_GE
q PF#2_DP_GE = PI#2_DP_GE
r PF#2_Al_GE = PI#2_Al_GE
s PF#2_P_GE = PI#2_P_GE
t PF#2_S_GE = PI#2_S_GE
u PF#2_NT_GE = PI#2_NT_GE

Estadísticos de contraste (b)

	PF#2_Ab_GE - PI#2_Ab_GE	PF#2_AD_GE - PI#2_AD_GE	PF#2_DP_GE - PI#2_DP_GE	PF#2_Al_GE - PI#2_Al_GE	PF#2_P_GE - PI#2_P_GE	PF#2_S_GE - PI#2_S_GE	PF#2_NT_GE - PI#2_NT_GE
Sig. exacta (bilateral)	.063(a)	.016(a)	.008(a)	.125(a)	.016(a)	.453(a)	.008(a)

a Se ha usado la distribución binomial.
b Prueba de los signos

```
SAVE OUTFILE='C:\Users\Niñas\Desktop\RESULTADOS ISABEL\Datos #3 PI y PF GE.sav'
 /COMPRESSED.
```

PF#1_Ab_GE: Nota del componente Abstracción de la pregunta No 2 en la Prueba Final del GE.

PI#1_Ab_GE: Nota del componente Abstracción de la pregunta No 2 en la Prueba Inicial del GE.

PF#1_AD_GE: Nota del componente Análisis y representación de datos de la pregunta No 32en la Prueba Final del GE.

PI#1_AD_GE: Nota del componente Análisis y representación de datos de la pregunta No 2 en la Prueba Inicial del GE.

PF#1_DP_GE: Nota del componente Descomposición de la pregunta No 2 en la Prueba Final del GE.

PI#1_DP_GE: Nota del componente Descomposición del problema de la pregunta No 2 en la Prueba Inicial del GE.

PF#1_Al_GE: Nota del componente Pensamiento algorítmico de la pregunta No 2 en la Prueba Final del GE.

PI#1_Al_GE: Nota del componente Pensamiento algorítmico de la pregunta No 2 en la Prueba Inicial del GE.

PF#1_P_GE: Nota del componente Reconocimiento y generalización de patrones de la pregunta No 2 en la Prueba Final del GE.

PI#1_P_GE: Nota del componente Descomposición y generalización de patrones de la pregunta No 2 en la Prueba Inicial del GE.

PF#1_S_GE: Nota del componente Simulación de la pregunta No 2 en la Prueba Final del GE.

PI#1_S_GE: Nota del componente Simulación de la pregunta No 2 en la Prueba Inicial del GE.

PF#1_Ab_GE: Nota total de la pregunta No 2 en la Prueba Final del GE.

PI#1_NT_GE: Nota total de la pregunta No 2 en la Prueba Inicial del GE.

Anexo 15: Resultados contraste pi y pf # 3 g. exp..spo

```
NPAR TEST

NPAR TEST
  /WILCOXON=X1 X2 X3 X4 X5 X6 X7  WITH X8 X9 X10 X11 X12 X13 X14 (PAIRED)
  /MISSING ANALYSIS.
```

Pruebas no paramétricas

Notas

Resultados creados		09-JUN-2018 15:39:41
Comentarios		
Entrada	Datos	C:\Users\Niñas\Desktop\RESULTADOS ISABEL\Datos #3 PI y PF GE.sav
	Conjunto de datos activo	Conjunto_de_datos1
	Filtro	<ninguna>
	Peso	<ninguna>
	Segmentar archivo	<ninguna>
	Núm. de filas del archivo de trabajo	9
Manipulación de los valores perdidos	Definición de los perdidos	Los valores perdidos definidos por el usuario serán tratados como perdidos.
	Casos utilizados	Los estadísticos para cada prueba se basan en todos los casos con datos válidos para las variables usadas en dicha prueba.
Sintaxis		NPAR TEST /WILCOXON=X1 X2 X3 X4 X5 X6 X7 WITH X8 X9 X10 X11 X12 X13 X14 (PAIRED) /MISSING ANALYSIS.
Recursos	Tiempo de procesador	0:00:00.00
	Tiempo transcurrido	0:00:00.01
	Número de casos permitidos(a)	41418

a Basado en la disponibilidad de memoria en el espacio de trabajo.

```
[Conjunto_de_datos1]  C:\Users\Niñas\Desktop\RESULTADOS  ISABEL\Datos  #3  PI  y  PF
GE.sav
```

Prueba de los rangos con signo de Wilcoxon

Rangos

			N	Rango promedio	Suma de rangos
PF#3_Ab_GE - PI#3_Ab_GE		Rangos negativos	0(a)	.00	.00
		Rangos positivos	7(b)	4.00	28.00
		Empates	2(c)		
		Total	9		
PF#3_AD_GE - PI#3_AD_GE		Rangos negativos	2(d)	3.50	7.00
		Rangos positivos	5(e)	4.20	21.00
		Empates	2(f)		
		Total	9		
PF#3_DP_GE - PI#3_DP_GE		Rangos negativos	1(g)	1.50	1.50
		Rangos positivos	5(h)	3.90	19.50
		Empates	3(i)		
		Total	9		
PF#3_Al_GE - PI#3_Al_GE		Rangos negativos	1(j)	3.00	3.00
		Rangos positivos	6(k)	4.17	25.00
		Empates	2(l)		
		Total	9		
PF#3_P_GE - PI#3_P_GE		Rangos negativos	0(m)	.00	.00
		Rangos positivos	7(n)	4.00	28.00
		Empates	2(o)		
		Total	9		
PF#3_S_GE - PI#3_S_GE		Rangos negativos	1(p)	3.00	3.00
		Rangos positivos	4(q)	3.00	12.00
		Empates	4(r)		
		Total	9		
PF#3_NT_GE - PI#3_NT_GE		Rangos negativos	1(s)	2.50	2.50
		Rangos positivos	5(t)	3.70	18.50
		Empates	3(u)		
		Total	9		

a PF#3_Ab_GE < PI#3_Ab_GE
b PF#3_Ab_GE > PI#3_Ab_GE
c PF#3_Ab_GE = PI#3_Ab_GE
d PF#3_AD_GE < PI#3_AD_GE
e PF#3_AD_GE > PI#3_AD_GE
f PF#3_AD_GE = PI#3_AD_GE
g PF#3_DP_GE < PI#3_DP_GE
h PF#3_DP_GE > PI#3_DP_GE
i PF#3_DP_GE = PI#3_DP_GE
j PF#3_Al_GE < PI#3_Al_GE
k PF#3_Al_GE > PI#3_Al_GE
l PF#3_Al_GE = PI#3_Al_GE
m PF#3_P_GE < PI#3_P_GE
n PF#3_P_GE > PI#3_P_GE
o PF#3_P_GE = PI#3_P_GE
p PF#3_S_GE < PI#3_S_GE
q PF#3_S_GE > PI#3_S_GE
r PF#3_S_GE = PI#3_S_GE
s PF#3_NT_GE < PI#3_NT_GE
t PF#3_NT_GE > PI#3_NT_GE
u PF#3_NT_GE = PI#3_NT_GE

Estadísticos de contraste (b)

	PF#3_Ab_GE PI#3_Ab_GE	PF#3_AD_GE - PI#3_AD_GE	PF#3_DP_GE - PI#3_DP_GE	PF#3_AI_GE - PI#3_AI_GE	PF#3_P_GE - PI#3_P_GE	PF#3_S_GE - PI#3_S_GE	PF#3_NT_GE - PI#3_NT_GE
Sig. exacta (bilateral)	.016(a)	.453(a)	.219(a)	.125(a)	.016(a)	.375(a)	.219(a)

a Basado en los rangos negativos.
b Prueba de los rangos con signo de Wilcoxon

PF#1_Ab_GE: Nota del componente Abstracción de la pregunta No 3 en la Prueba Final del GE.

PI#1_Ab_GE: Nota del componente Abstracción de la pregunta No 3 en la Prueba Inicial del GE.

PF#1_AD_GE: Nota del componente Análisis y representación de datos de la pregunta No 3 en la Prueba Final del GE.

PI#1_AD_GE: Nota del componente Análisis y representación de datos de la pregunta No 3 en la Prueba Inicial del GE.

PF#1_DP_GE: Nota del componente Descomposición de la pregunta No 3 en la Prueba Final del GE.

PI#1_DP_GE: Nota del componente Descomposición del problema de la pregunta No 3 en la Prueba Inicial del GE.

PF#1_AI_GE: Nota del componente Pensamiento algorítmico de la pregunta No 3 en la Prueba Final del GE.

PI#1_AI_GE: Nota del componente Pensamiento algorítmico de la pregunta No 3 en la Prueba Inicial del GE.

PF#1_P_GE: Nota del componente Reconocimiento y generalización de patrones de la pregunta No 3 en la Prueba Final del GE.

PI#1_P_GE: Nota del componente Descomposición y generalización de patrones de la pregunta No 3 en la Prueba Inicial del GE.

PF#1_S_GE: Nota del componente Simulación de la pregunta No 3en la Prueba Final del GE.

PI#1_S_GE: Nota del componente Simulación de la pregunta No 3 en la Prueba Inicial del GE.

PF#1_Ab_GE: Nota total de la pregunta No 3 en la Prueba Final del GE.

PI#1_NT_GE: Nota total de la pregunta No 3 en la Prueba Inicial del GE.