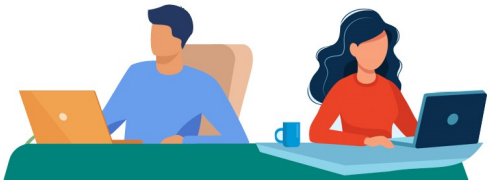


Introducción a la Programación (Pensamiento Computacional)

4 créditos teóricos, 2 créditos prácticos

A. Información del profesor

Nombre del profesor

Teoría: Inga. María Castillo
Laboratorio: Inga. Darsy De León – Inga. Cindy García

Correo electrónico

Teoría: mdcastilloma@correo.url.edu.gt
Laboratorio: cgarciap@correo.url.edu.gt – dbdeleonl@correo.url.edu.gt

Campus o sede

Campus Central

Horario

Teoría: martes y jueves, 8:40 AM – 10:10 AM
Laboratorio: Martes 12:20 – 13:50 Viernes 12:20 - 13:50
Trabajo Supervisado: Jueves 12:20 – 13:50 / Lunes 12:20 - 13:50

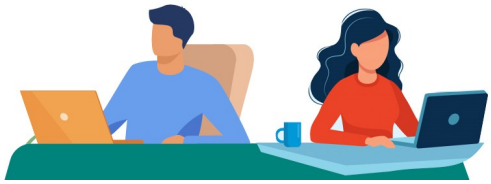
B. Información general

Descripción

Las computadoras se encuentran en todos los ámbitos del mundo contemporáneo e intervienen en cada aspecto de la vida y la sociedad.

Resulta familiar identificar las aplicaciones de las soluciones computacionales, desde finanzas y sistemas de salud hasta participación política. Las computadoras potencias las redes de comunicación global, se encuentran inmersas en la sociedad y generan cada vez más dependencia.

Las computadoras son herramientas que debemos utilizar de manera intencional. En este sentido, el pensamiento computacional provee principios derivados de las ciencias de la computación. Estos principios incluyen el elegir los detalles esenciales de un problema, la formulación de problemas en términos que las computadoras comprendan y cómo resolver problemas en procesos que puedan ser automatizados.



Estos principios son importantes para todos, ya sea que trabajen en ciencias de la computación o no, porque su finalidad es más allá de solo su comprensión, el empoderar a la resolución de problemas, integrando a los sistemas computacionales como parte de las soluciones para que las tareas puedan ser realizadas de forma rápida y eficiente.

Para los que persiguen una carrera en ciencias de la computación, los beneficios resultan evidentes: integra el análisis de problemas con las tecnologías de ciencias de la computación, desarrollando las capacidades de producir soluciones de alta calidad. Es decir, soluciones - principalmente - de software más robustas, poderosas, bien diseñadas, con amplias aplicaciones y libres de errores.

"Es casi imposible investigar alguna ciencia o ingeniería sin la capacidad de pensar computacionalmente". Carnegie Mellon (2016).

El pensamiento computacional es una capacidad para la vida, tan fundamental como la lectoescritura y las matemáticas, que transforma cada disciplina, profesión y sector, dando una ventaja competitiva a quien la desarrolle, Wing (2014).

En este curso se desarrolla el pensamiento computacional y cómo éste apoya al desarrollo de software. Contempla espacios para el desarrollo de conceptos y espacios para el desarrollo de procesos, mediados por actitudes y valores, que permitirán implementar el enfoque que busca la visión del pensamiento computacional como una herramienta creativa y de innovación.

Modalidad
Presencial

C. Malla curricular

COMPETENCIAS GENÉRICAS

El egresado landivariano se identifica por:

Pensamiento lógico, reflexivo y analógico	Pensamiento crítico	Resolución de problemas
Habilidades de investigación	Uso de TIC y gestión de la información	Comunicación efectiva, escrita y oral
Comprensión lectora	Compromiso ético y ciudadanía	Liderazgo constructivo
Aprecio y respeto por la diversidad e interculturalidad	Creatividad	

COMPETENCIAS ESPECÍFICAS

Competencia 1

Aplicar el pensamiento computacional como un enfoque para la resolución de problemas, en distintos escenarios y dominios, fundamentados en los principios de pensamiento lógico y algorítmico, descomposición, abstracción y modelado, anticipación de errores, y evaluación de soluciones.

Competencia 2

Desarrollar soluciones a problemas, de manera eficiente, aplicando el pensamiento computacional y utilizando herramientas de ciencias de la computación.

METODOLOGÍA

Este curso se desarrollará a través de los siguientes métodos de aprendizaje-enseñanza:

Aprendizaje invertido. "Es una técnica didáctica en la que la exposición de contenido se hace por medio de videos que pueden ser consultados en línea de manera libre, mientras el tiempo de aula se dedica a la discusión, resolución de problemas y actividades prácticas bajo la supervisión y asesoría del profesor", Instituto para el Futuro de la Educación.

Aprendizaje basado en proyectos. "Técnica didáctica que se orienta en el diseño y desarrollo de un proyecto de manera colaborativa por un grupo de alumnos, como una forma de lograr los objetivos de aprendizaje de una o más áreas disciplinares y además lograr el desarrollo de las competencias relacionadas con la administración de proyectos reales", Instituto para el Futuro de la Educación.

Aprendizaje basado en retos. "Es una estrategia que proporciona a los estudiantes un contexto general en el que ellos de manera colaborativa deben de determinar el reto a resolver. Los estudiantes trabajan con sus profesores y expertos para resolver este reto en comunidades de todo el mundo y así desarrollar un conocimiento más profundo de los temas que estén estudiando" Instituto para el Futuro de la Educación.

Aprendizaje por indagación. "Consiste en la aplicación de estrategias de enseñanza y aprendizaje que tienen como propósito conectar la investigación con la enseñanza, las cuales permiten la incorporación parcial o total del estudiante en una investigación basada en métodos científicos bajo la supervisión del profesor", Instituto para el Futuro de la Educación.

PROGRAMACIÓN

COMPETENCIA 1

Aplicar el pensamiento computacional como un enfoque para la resolución de problemas, en distintos escenarios y dominios, fundamentados en los principios de pensamiento lógico y algorítmico, descomposición, abstracción y modelado, anticipación de errores, y evaluación de soluciones.

Saber conceptual (contenido temático)

- Conceptuación del pensamiento computacional.
- Pensamiento lógico y algorítmico.
- Resolución de problemas y descomposición. Patrones y generalización.
- Abstracción y modelación. Modelos estáticos y dinámicos, representación de datos.
- Anticipación y gestión de errores.
- Evaluación de soluciones. Funcionalidad, eficiencia, elegancia y utilidad.

Saber procedimental (habilidades y destrezas)

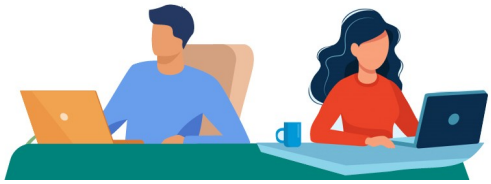
- Experimentación de técnicas y actitudes vinculadas al pensamiento computacional para cada uno de los sabres conceptuales.

Saber actitudinal (conductas observables)

- Perseverancia.
- Experimentación.
- Creatividad.

Indicadores de logro (resultado):

- Definir pensamiento computacional y explicar su aplicación en las ingenierías y la ciencia.
- Comprender la lógica booleana y las propiedades de los algoritmos.
- Definir problemas y aplicar estrategias de generalización y descomposición para su resolución.
- Comprender la abstracción, sus ventajas, limitaciones y la forma en que permite resolver problemas.
- Analizar modelados típicos estáticos y dinámicos para la resolución de problemas.
- Aplicar estrategias para anticipar los errores y defectos de las soluciones planteadas.
- Discutir los atributos de calidad claves para garantizar la efectividad de una solución.



COMPETENCIA 2

Desarrollar soluciones a problemas, de manera eficiente, aplicando el pensamiento computacional y utilizando herramientas de ciencias de la computación.

Saber conceptual (contenido temático)

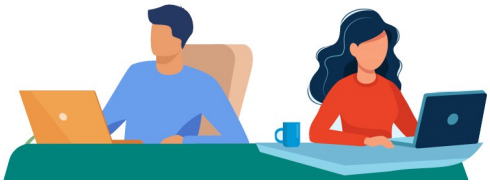
- Introducción al lenguaje de programación.
- Bloques y controles.
- Organización del código.
- Abstracción y patrones.
- Modelado efectivo.
- Testing y evaluación de programas.
- Integración Humano-computadora. Evaluación de interfaces que interactuarán con humanos.

Saber procedimental (habilidades y destrezas)

- Familiarización con el entorno del lenguaje de programación, operaciones básicas y funciones.
- Operadores lógicos, secuencias, condicionales e iteraciones.
- Uso de módulos, interfaces, paquetes.
- Abstracciones en programación.
- Tipos de datos primitivos y compuestos.
- Herencia, polimorfismo, patrones.
- Entidades, clases, paquetes, casos de uso y flujos de secuencias.
- Anticipar errores (sintaxis y semántica). Anticipar defectos.
- Capturar errores potenciales (exception).
- Debugging.
- Verificación: pruebas unitarias.
- Validación: pruebas de aceptación.
- Diseño interactivo (diseño de la interfaz, testing, análisis de resultados y repetir).
- Principios de usabilidad.
- Prototipado.
- Pruebas beta.

Saber actitudinal (conductas observables)

- Perseverancia.
- Experimentación.
- Creatividad.
- Compromiso con la calidad.



Indicadores de logro (resultado):

- Utilizar el entorno de programación, los operadores básicos y las funciones del lenguaje elegido.
- Aplicar el pensamiento lógico y algoritmo en el lenguaje de programación elegido.
- Aplicar la organización, control y separación para la facilidad de uso y comprensión de la solución en el lenguaje de programación elegido.
- Identificar patrones en los programas y discutir cómo se aplica la abstracción.
- Aplicar la modelación visual utilizando UML.
- Comprender el funcionamiento y aplicar excepciones y estrategias defensivas de programación para el control de errores.

EVALUACIÓN

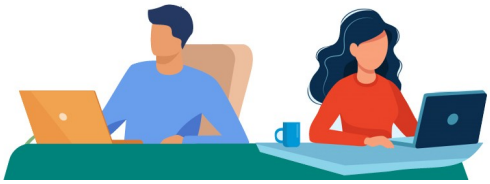
a. Estrategias de evaluación sumativa

Estrategias	Puntaje
Pruebas objetivas	30
Resolución de retos	30
Diario de experiencias de laboratorio	10
Proyectos	20
Portafolio	10
TOTAL	**100**

Es requisito indispensable aprobar el Laboratorio con una ponderación mínima del 65% (experiencias de laboratorio y proyectos), para aprobar el curso. Se debe contar con un mínimo del 75% de asistencia a clases (teoría, laboratorio y trabajo supervisado) para tener derecho a concluir el curso.

b. Estrategias de evaluación formativa

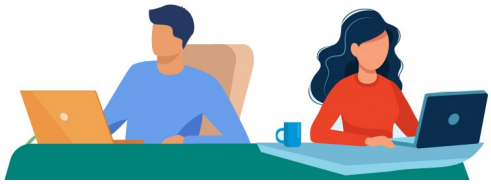
Técnicas	Procedimiento
Lienzos colaborativos	Organizadores gráficos para desarrollar en pequeños equipos.
3-2-1	Los estudiantes escriben en tres minutos lo que aprendieron, lo que encontraron interesante y lo que tienen duda.
3 minute message	Los estudiantes escriben la argumentación a un problema o caso en tres minutos.
Team games	Actividades lúdicas para realizar en pequeños grupos.



CALENDARIO DE REFERENCIA POR TEMAS

Fecha	Contenidos		Actividad de evaluación
	Conceptual + Actitudinal	Procedimental + Actitudinal	
Semana 1	Conceptuación del pensamiento computacional.	Experimentación de técnicas y actitudes vinculadas al pensamiento computacional.	Pruebas objetivas. Resolución de retos. Diario de experiencias de laboratorio.
Semana 2	Pensamiento lógico y algorítmico.	Entrenando a ratones de laboratorio. Escape de laberinto. Base de datos de recetas de cocina.	
Semana 3	Resolución de problemas y descomposición. Patrones y generalización.	Vehículo autónomo de camino a casa. Cadena mágica. Invitados a la fiesta.	
Semana 4	Abstracción y modelación. Modelos estáticos y dinámicos, representación de datos.	Figuras de animales. Editando noticias. Arte con troncos.	
Semana 5	Anticipación y gestión de errores.	Posiciones mágicas. Analizar, encontrar y corregir errores en MinesWeeper.	
Semana 6	Evaluación de soluciones. Funcionalidad, eficiencia, elegancia y utilidad.	Resultados mezclados. Cargando los barcos. Agentes secretos.	Proyecto 1
Semana 7	Introducción al lenguaje de programación.	Familiarización con el entorno del lenguaje de programación, operaciones básicas y funciones. Proporcionar un programa ya realizado, analizarlo, seguirlo paso a paso y ejecutarlo.	Pruebas objetivas. Resolución de retos. Diario de experiencias de laboratorio.
Semana 8	Bloques y controles.	Operadores lógicos, secuencias, condicionales e iteraciones. Codificar las soluciones de la semana 2.	

Fecha	Contenidos		Actividad de evaluación
	Conceptual + Actitudinal	Procedimental + Actitudinal	
Semana 9	Organización del código.	Uso de módulos, interfaces, paquetes. Codificar las soluciones de la semana 3.	Pruebas objetivas. Resolución de retos. Diario de experiencias de laboratorio.
Semana 10	Abstracción y patrones.	Abstracciones en programación. Tipos de datos primitivos y compuestos. Herencia, polimorfismo, patrones. Piedra, papel o tijeras. Codificar las soluciones de la semana 4.	
Semana 11	Modelado efectivo.	Entidades, clases, paquetes, casos de uso y flujos de secuencias. Modelar el funcionamiento de un ATM.	
Semana 12	Testing y evaluación de programas.	Anticipar errores (sintaxis y semántica). Anticipar defectos. Capturar errores potenciales (exception). Debugging. Verificación: pruebas unitarias. Validación: pruebas de aceptación. Aplicar anticipación de errores de la semana 5.	Proyecto 2
Semana 13	Integración Humano-computadora. Evaluación de interfaces que interactuarán con humanos.	Diseño interactivo (diseño de la interfaz, testing, análisis de resultados y repetir). Principios de usabilidad. Prototipado. Pruebas beta.	Pruebas objetivas. Resolución de retos. Diario de experiencias de laboratorio.
Semana 14	Proyecto integrador. Ejemplo guiado.	Análisis y diseño de proyectos.	Prueba objetiva Proyecto final
Semana 15	Proyecto integrador.	Feria de presentación de soluciones.	Proyecto final Portafolio



REFERENCIAS BIBLIOGRÁFICAS

- Bordigon, F. e Iglesias, A. (2020). Introducción al pensamiento computacional. Universidad Pedagógica Nacional, Educar Sociedad del Estado y el Ministerio de Educación Argentina.
- Beecher, K. (2017). Computational Thinking. A beginner's guide to problem-solving and programming.
- ISTE (2018). Computational Thinking. Meets Students Learning. International Society for Technology in Education.
- MIT (2022). Computational Thinking, a live online Julia/Pluto textbook. Julia: A Fresh Approach to Computing. computationalthinking.mit.edu. Massachusetts Institute of Technology

Ing. Cristian Roldán
Coordinador de Área de Programación
Departamento de Ciencias Básicas
Facultad de Ingeniería

Ing. Adolfo Galán
Director
Departamento de Ingeniería en Informática y Sistemas
Facultad de Ingeniería