UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS	**FORMATO DE SYLLABUS**	Código: AA-FR-003	SIGUD Sistema Integrado de Gestión
	Macroproceso: Direccionamiento Estratégico	Versión: 01	
	Proceso: Autoevaluación y Acreditación	Fecha de Aprobación: 27/07/2023	

FACULTAD:	**FACULTAD DE INGENIERÍA**						
PROYECTO CURRICULAR:	**INGENIERÍA**			CÓDIGO PLAN DE ESTUDIOS:			
I. IDENTIFICACIÓN DEL ESPACIO ACADÉMICO							
Nombre del Espacio Académico: **PROGRAMACIÓN BÁSICA**							
Código del espacio académico:		Número de créditos académicos:			**3**		
Distribución horas de trabajo:	HTD	**2**	HTC	**4**	HTA	**3**	
Tipo de espacio académico:	Asignatura	**X**	Cátedra				
NATURALEZA DEL ESPACIO ACADÉMICO:							
Obligatorio Básico	**X**	Obligatorio Complementario		Electivo Intrínseco		Electivo Extrínseco	
CARÁCTER DEL ESPACIO ACADÉMICO:							
Teórico		Práctico		Teórico-Práctico	**X**	Otros:	Cuál:_____
MODALIDAD DE OFERTA DEL ESPACIO ACADÉMICO:							
Presencial	**X**	Presencial con incorporación de TIC		Virtual		Otros:	Cuál:_____

II. SUGERENCIAS DE SABERES Y CONOCIMIENTOS PREVIOS

No tiene.

III. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

La programación es una habilidad fundamental en la formación de un ingeniero. A través de este curso de programación básica, los estudiantes adquieren los fundamentos esenciales para comprender, diseñar y desarrollar software, lo que les proporcionará las bases necesarias para abordar problemas de ingeniería de manera efectiva. Este plan de estudios está diseñado para cumplir con los objetivos académicos y profesionales de los estudiantes en los programas de Ingeniería.

IV. OBJETIVOS DEL ESPACIO ACADÉMICO (GENERAL Y ESPECÍFICOS)

Objetivo General:

Presentar al estudiante, elementos fundamentales que le permitan tener claridad acerca de la evolución de la programación, de tal manera que pueda obtener soluciones a problemas sencillos apoyados en un computador, un lenguaje de programación y un paradigma en donde el alumno sea capaz de enfrentarse a situaciones o problemas más complejos en las que debe identificar los elementos y estados involucrados, generar modelos para su representación y manipulación algorítmica. Debe ser capaz de diseñar soluciones para los problemas, validar su corrección e implementar prototipos para ellas utilizando un lenguaje de programación

Objetivos Específicos:

1.Conocer la evolución de los lenguajes, los paradigmas y la computación.
2.Evidenciar de manera clara y concreta la evolución de la programación con relación a la evolución del computador.
3.Identificar la estructura de un computador.
4.Desarrollar el concepto de algoritmo y aplicarlo en la solución de programas sencillos.
5.Solucionar problemas elementales utilizando la lógica computacional.
6.Resolver problemas sobre el sistema computacional con la ayuda de un lenguaje de programación.
7.Reconocer la sintaxis básica del lenguaje de programación escogido.

V. PROPÓSITOS DE FORMACIÓN Y DE APRENDIZAJE (PFA) DEL ESPACIO ACADÉMICO

Competencias	Dominio-Nivel	**RA**	**Resultados de Aprendizaje**

Competencias que compromete la asignatura:	Piensa ordenadamente para modelar una solución a un problema haciendo uso de la algoritmia, expresando esta solución en un lenguaje computacional.	Cognitivo- Crear	01	Desarrollar algoritmos para solucionar problemas y expresar dichas soluciones utilizando un lenguaje de programación, aplicando un pensamiento ordenado y estructurado.
Competencias específicas de la asignatura:	Analiza, diseña y desarrolla soluciones computacionales utilizando un pensamiento ordenado y estructurado. Aplicando conceptos y abstracciones de sistemas numéricos, localizando históricamente momentos clave en la evolución de los sistemas computacionales. Identifica los componentes de un sistema computacional, representa soluciones utilizando lógica algorítmica, utiliza las diferentes estructuras de control, descompone problemas en subproblemas, aplica recursividad en la resolución de problemas, define e implementr tipos de dato abstracto, y modela mecanismos para el manejo dinámico de memoria y persistencia.	ognitivo-Analiza	02	Utilizar adecuadamente el concepto y la abstracción de los sistemas numéricos en la solución de problemas computacionales.
			03	Localizar históricamente los diferentes momentos en la evolución de los sistemas computacionales para la identificación de posibles soluciones a problemas informáticos.
			04	Identificar los diversos componentes de un sistema computacional para distinguir el ámbito de solución.
			05	Representar soluciones de problemas aplicando el concepto de lógica algorítmica.
			06	Modelar, implementar y evaluar problemas cuya solución algorítmica requiere el uso de las diferentes estructuras de control.
			07	Modelar, implementar y evaluar problemas descomponiéndolos en subproblemas que permitan una solución más simple o la reutilización de soluciones.
			08	Resolver problemas que requieren aplicar el concepto de recursividad.
			09	Definir e implementar tipos de dato abstracto para la optimización en la solución de problemas de información.
			10	Modelar, implementar y evaluar mecanismos para el manejo dinámico de memoria y persistencia.
Competencias Transversales a las que contribuye la asignatura:	implementa prototipos de soluciones algorítmicas utilizando estructuras de programación secuenciales, condicionales e iterativas. Trabaja con distintos tipos de datos, variables y constantes, realizar cálculos aritméticos y proposicionales, y aplicar operadores relacionales y booleanos. Utiliza funciones de lectura y escritura, trabaja con arreglos y matrices, y realizaf conversiones entre tipos de datos. Conoce el uso de estructuras de salto, como break y continue e implementa soluciones algorítmicas basadas en descomposición. De igual manera, conoce el manejo de archivos para la persistencia y flujo de datos, y utiliza funciones recursivas y referencias en sus programas.	Cognitivo-Aplica	11	Discernir que conocimientos y herramientas tecnologías debe apropiar para la resolución de problemas particulares.
			12	Actuar estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos.
			13	Interpretar, argumentar y proponer ideas de forma clara y correcta por medio del lenguaje escrito.
			14	Canalizar clara y comprensiblemente ideas y opiniones hacia los demás por medio del lenguaje hablado.
			15	Interpretar y comunicar con propiedad en una segunda
			16	Actuar éticamente comprometido con el desarrollo de las actividades de la asignatura.
			17	Elaborar y mantener documentación descriptiva de la génesis, producción y operatividad de los proyectos informáticos.
			18	Manejar protocolos de comunicación y etiqueta digital
			19	Comunicarse estratégicamente haciendo uso de la
			20	Actuar en contextos académicos y profesionales con un enfoque culto, ético y humanístico.
			21	Interpretar la realidad y proponer nuevos argumentos para desarrollar soluciones innovadoras en contextos sociales.
			22	Presentar los trabajos de forma estética, ergonómica y conforme al contexto sociocultural al cual se destinan.

VI. CONTENIDOS TEMÁTICOS

1. Reconocer la estructura y funcionamiento del computador. (2 semanas)
- Sistemas numéricos: Sistema binario, hexadecimal y octal.
- Conversiones entre sistemas. Números de precisión finita.
- Representación de números negativos en base 2.
- Representación de número punto flotante en base 2.
- Operaciones.
- Desarrollo histórico del "Hardware": El ábaco, Maquinas de Pascal, Leibniz, Babbage, Turing.
- El computador hasta hoy: Generaciones.
- Evolución de los lenguajes de programación.
- Estructura del computador: Procesador, memoria principal, memoria secundaria, E/S, buses

2. Conceptualizar y abstraer problemas. Desarrollo de algoritmos. (4 semanas)
- Concepto de algoritmo
- Los diagramas de flujo como herramienta de modelación de algoritmos.
- Pseudocódigo: Una herramienta de palabras útil.
- Secuencialidad, selección y repetición: análisis, modelamiento y diseño de algoritmos
- Paradigmas de programación
- Generalidades de los paradigmas: abstracción y encapsulamiento
- Introducción al paradigma orientado a objetos
- Algoritmos orientados a objetos

3. Diseñar soluciones algorítmicas para problemas computacionales (Basado en el lenguaje de programación escogido.) (8 semanas)
- Estructura de un programa, restricciones, comentarios
- Tipos de datos, variables y constantes: Primitivos y TAD.
- Operadores y Cálculo Proposicional: Aritméticos, Bitwise, asignación, relacionales, Booleanos.
- Conversión entre tipos de datos
- Funciones de lectura y escritura.
- Arreglos y matrices. Definición, inicialización.
- Estructuras de salto: break, continue.
- Funciones: Parámetros por valor, retorno de valores, variables locales, globales y estáticas.
- Librerías de funciones.
- Funciones recursivas.
- Registros o estructuras. Acceso a los elementos de una estructura, estructuras dentro de otras, arreglos de estructuras, estructuras con apuntadores a otras.

4. Persistencia de datos y flujo de datos. (2 semanas)

VII. ESTRATEGIAS DE ENSEÑANZA QUE FAVORECEN EL APRENDIZAJE

Tradicional		Basado en Proyectos	X	Basado en Tecnología	X
Basado en Problemas	X	Colaborativo	X	Experimental	X
Aprendizaje Activo	X	Autodirigido	X	Centrado en el estudiante	X

VIII. EVALUACIÓN

Resultados de aprendizaje (RA) a ser evaluados:	Resultados de aprendizaje asociados a las evaluaciones (T: Teórico / P: Práctico)					
	Actividades Entregables (editable)	Talleres (editable)	Parcial Conjunto	Informes de proyecto final (editable)	Proyecto final (editable	Exposiciones (editable)
RA01			**X**			
RA02						
RA03						
Tipo de evaluación**						
Porcentaje de evaluación (%)			20%			
Trabajo Individual (I) o Grupal (G)			I			
Tipo de nota	0-5	0-5	0-5	0-5	0-5	0-5

IX. MEDIOS Y RECURSOS EDUCATIVOS

- Aula normal con pizarrón para sesiones de cátedra y para sesiones de discusión.
- Disponibilidad para acceder a proyector multimedia.
- Laboratorio de computación, con un computador por alumno, para las sesiones de laboratorio; cada computador debe contar con el intérprete para el lenguaje de programación que se va a utilizar para validar los prototipos.
- Página web para publicar material didáctico, guías de ejercicios, soluciones, tareas, etc.
- Acceso fuera de clases a laboratorios de computación que cuenten con el intérprete para el lenguaje de programación que se va a utilizar para validar los prototipos, y con acceso a la página web del módulo.
- Acceso al material bibliográfico recomendado
- Asignación de una persona que tenga las plenas competencias del curso (monitor) para asesorar a los estudiantes en dudas durante las sesiones del laboratorio de computación.

X. PRÁCTICAS ACADÉMICAS - SALIDAS DE CAMPO

- Asistencia a clases expositivas y de discusión.
- Elaboración y lectura de paper (documentación).
- Se debe procurar incentivar el trabajo de grupo más que el trabajo individual. (se recomienda trabajar en grupos de dos o tres estudiantes).
- Implementación y prueba de prototipos (programas) en laboratorio de computación.

XI. BIBLIOGRAFÍA

Referencias Básicas

1. Deitel, P., & Deitel, H. Python: Cómo programar. Pearson, 2020.	Una guía práctica y completa para aprender Python, incluyendo fundamentos de programación orientada a objetos.
2. Sedgewick, R. y Wayne, K. Algoritmos. Addison-Wesley, 2011.	Explica algoritmos esenciales y cómo aplicarlos en problemas básicos de programación.
3. C++ Primer (5th Edition). Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. Editorial: Addison-Wesley.	Este libro es una referencia completa y accesible para los principiantes, cubriendo desde los fundamentos del lenguaje hasta
4. C++ How to Program (10th Edition). Paul Deitel, Harvey Deitel. Editorial: Pearson.	Este libro combina teoría con una gran cantidad de ejemplos prácticos, haciéndolo ideal para estudiantes que buscan aprender
5. "C: Cómo programar" - Deitel & Deitel	Un recurso fundamental para entender los conceptos básicos y avanzados de la programación en C, que sigue siendo un lenguaje
5. "Introducción a los Algoritmos" - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, y Clifford Stein	Este libro es una referencia estándar en algoritmos. Aunque avanzado, es una base sólida para estudiantes de ingeniería.
Parsons, D. y Bowe, H.Fundamentos de programación con Java .Pearson, 2019.	Enfocado en aprender la lógica de programación básica usando Java.

Recursos Complementarios

Tanenbaum, Andrew. Structured Computer Organization. Prentice Hall.
Levine, Guillermo. Computación y Programación Moderna. Addison Wesley.
Bajarme Stroustrup El C ++ Lenguaje de Programación Addison Wesley.
Burton Harvey, Simon Robinson, Julian Templeman, Karli Watson. C ++ Programming . Wrox Press Ltda.
Becerra, Cesar. Lenguaje C. Por Computador.
Rodriguez C., Llana L.F, Martinez, R.,Palao P., Pareja, C. Ejercicios de Programación Creativos y Recreativos en C ++. Prentice Hall.

Recursos en línea

Páginas web:	https://geeksroom.com/2020/04/11-libros-programacion/128041/
Documentación de Python. Oficial de Python con ejemplos y conceptos básicos.	https://docs.python.org /
W3Schools: Excelente para aprender a programar con tutoriales interactivos.	https ://www .w3schools .com /python/
Code Academy: Una plataforma interactiva para aprender los fundamentos de progra	https ://www .codecademy .com /learn /learn -python -3
GeeksforGeeks: Recurso muy utilizado para resolver problemas de programación y alg	https ://www .geeksforgeeks .org/
Sitio oficial de C++. Un recurso esencial con documentación y ejemplos sobre todas las funciones y características del lenguaje.	cppreference .com
Una plataforma con explicaciones claras sobre algoritmos, estructuras de datos y	GeeksforGeeks (https://www.geeksforgeeks.org/)

XII. SEGUIMIENTO Y ACTUALIZACIÓN DEL SYLLABUS

Fecha revisión por Consejo Curricular:			
Fecha aprobación por Consejo Curricular:		Número de acta:	