UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERIA

SYLLABUS

Ingeniería Catastral y Geodesia

NOMBRE DEL DOCENTE:

ESPACIO ACADÉMICO (Asignatura):

Programación Básica .

Obligatorio (X) : Básico (X) Complementario ( )
Electivo ( ) : Intrínsecas ( ) Extrínsecas ( )

CÓDIGO: 2

NÚMERO DE ESTUDIANTES:

GRUPO:

NÚMERO DE CREDITOS: 3

TIPO DE CURSO: **TEÓRICO** [ ] **PRACTICO** [ ] **TEO-PRAC:** [X]

Alternativas metodológicas:
Clase Magistral (x), Seminario ( ), Seminario – Taller ( ), Taller (x), Prácticas (x), Proyectos tutoriados ( ), Otro: _____

HORARIO:

DÍA	HORAS	SALÓN

I. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

Competencias del perfil a las que contribuye la asignatura:	Esta asignatura contribuye al desarrollo de la competencia "Resuelve problemas computacionales algorítmicamente" que se encuentra en el dominio de "programación" del área "básicas de ingeniería" de los proyectos curriculares de la facultad de ingeniería
Contribución a la formación:	En este espacio académico se establecen las bases del pensamiento algorítmico formal que constituye uno de los pilares de la disciplina y contribuye a los dominios de desempeño profesional. Los ingenieros, cualquiera que sea su área específica, necesitan estructurar el pensamiento de forma lógica y racional que le permita resolver problemas utilizando herramientas computacionales.
Puntos de apoyo para otras asignaturas:	• Herramienta fundamental para Programación Orientada por Objetos • Herramienta fundamental para Bases de Datos • Herramienta fundamental para Programación de Interfaces SIG.

Requisitos previos:	Ninguno

II. PROGRAMACIÓN DEL CONTENIDO

OBJETIVO GENERAL

Construir pensamiento organizado para resolver problemas utilizando un lenguaje de programación estructurado sentando las bases que permitan la futura asimilación de nuevos paradigmas y lenguajes de programación.

OBJETIVOS ESPECÍFICOS

1. Presentar la estructura y evolución de los computadores y paradigmas de programación
2. Desarrollar el concepto de algoritmo y aplicarlo en la solución de programas sencillos
3. Solucionar problemas elementales utilizando la lógica computacional
4. Hacer uso de algún lenguaje de programación para la implementación de los algoritmos

COMPETENCIAS DE FORMACIÓN:	
Competencias que compromete la asignatura:	1. Facilitar la futura asimilación de diferentes lenguajes y paradigmas de programación y del uso de herramientas de software en general
Subcompetencias de la asignatura:	Cognitivas - Comprende los elementos básicos del algoritmo y su uso - Utiliza la "decisión" en el contexto adecuado del algoritmo con el concepto lógico correcto - Emplea correcta y eficientemente las sentencias cíclicas en el algoritmo como elemento fundamental en la resolución de muchos problemas computacionales - Resuelve problemas utilizando algoritmos estructurados y sus elementos fundamentales - Utiliza funciones como fundamento de paradigma estructurado realizando algoritmos de forma tal que facilite la eficiencia en programación Procedimentales - Maneja en forma básica una herramienta de software que permita la programación estructurada, empleando la sintaxis adecuada del lenguaje - Utiliza adecuadamente (en el contexto) los operadores del algoritmo en el lenguaje de programación de forma tal que construye expresiones correctas
Programa sintético:	1. Reconocer la estructura y funcionamiento del computador. 1.1. Sistemas numéricos: Sistema binario, hexagecimal y octal. 1.2. Conversiones entre sistemas. Números de precisión finita. 1.3. Operaciones en binario, hexagecimal y octal. 1.4. El computador hasta hoy: Generaciones. 1.5. Evolución de los lenguajes de programación. 1.6. Estructura del computador: Procesador, memoria principal, memoria secundaria, E/S, buses 2. Conceptualizar y abstraer problemas. Desarrollo de algoritmos. 2.1. Concepto de algoritmo 2.2. Los diagramas de flujo como herramienta de modelación de algoritmos. 2.3. Pseudocódigo: Una herramienta de palabras útil. 2.4. Modelar un problema de solución secuencial 2.5. Diseñar una solución algorítmica secuencial 2.6. Analizar una solución algorítmica secuencial 2.7. Modelar un problema cuya solución involucra condiciones 2.8. Diseñar una solución algorítmica que involucra condiciones 2.9. Analizar una solución algorítmica que involucra condiciones 2.10. Modelar problema cuya solución involucra iteraciones 2.11. Diseñar solución algorítmica que involucra iteraciones 2.12. Analizar una solución algorítmica que involucra iteraciones 2.13. Modelar problema complejo cuya solución amerita el uso de descomposición 2.14. Diseñar una solución algorítmica basada en descomposición 2.15. Analizar una solución algorítmica basada en descomposición 3. Diseñar soluciones algorítmicas para problemas computacionales *(Basado en el lenguaje de programación escogido. En este caso se hace referencia al lenguaje de programación C.)*

	3.1. Estructura de un programa en C, restricciones, comentarios 3.2. Tipos de datos, variables y constantes: Caracteres, Boleanos, Reales, Enteros. 3.3. Operadores 3.3.1. Aritméticos: asignación suma, resta, multiplicación, división, módulo, incremento, decremento, y todos asociados con una variable en una cantidad determinada. 3.3.2. Bitwise: And, Or, Or exclusivo, complemento, desplazamiento a izquierda y derecha, combinaciones con el operador de asignación. 3.3.3. Relacionales: menor que, mayor que, menor o igual que mayor o igual que, igual, diferente. 3.3.4. Booleanos: para la estructuración de expresiones: Not, And, Or. Jerarquias de los operadores. 3.3.5. Proposiciones. And, or, xor, tablas de verdad. 3.4. Implementar prototipo de solución algorítmica secuencial 3.5. Conversión entre tipos de datos 3.6. Funciones de lectura y escritura. 3.7. Arreglos y matrices. Definición, inicialización. 3.8. Implementar prototipo de solución algorítmica que involucra condiciones: if, if else, switch. 3.9. Implementar prototipo de solución algorítmica que involucra iteraciones: for, while, do while. 3.10. Estructuras de salto: break, continue. 3.11. Implementar prototipo de solución algorítmica basada en descomposición 3.12. Funciones: Parámetros por valor, retorno de valores, variables locales, globales y estáticas. 3.13. Librerías de funciones. 3.14. Apuntadores. Definición, asignación tipos y niveles de apuntadores, apuntadores a funciones 3.15. Registros o estructuras. Acceso a los elementos de una estructura, estructuras dentro de otras, arreglos de estructuras, estructuras con apuntadores a otras.

III. ESTRATEGIAS

Metodología Pedagógica y Didáctica:

- Asistencia a clases expositivas y de discusión
- Implementación y prueba de prototipos (programas) en laboratorio de computación
- Todos los temas serán tratados en clase por el profesor, recreados con ejemplos totalmente desarrollados, la apropiaron de los conceptos se realizara por medio de ejercicios y talleres que se realizaran todas las clases y que serán enviados al aula virtual.

	Horas			Horas profesor/semana	Horas Estudiante/semana	Total Horas Estudiante/semestre	Créditos
Tipo de Curso	TD	TC	TA	(TD + TC)	(TD + TC +TA)	X 16 semanas	
Asignatura	2	4	3	6	9	144	3

Trabajo Presencial Directo (TD): trabajo de aula con plenaria de todos los estudiantes.
Trabajo Mediado_Cooperativo (TC): Trabajo de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.
Trabajo Autónomo (TA): Trabajo del estudiante sin presencia del docente, que se puede realizar en distintas instancias: en grupos de trabajo o en forma individual, en casa o en biblioteca, laboratorio, etc.)

IV. RECURSOS

Medios y Ayudas:

- 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.

BIBLIOGRAFÍA

TEXTOS GUÍA

- Cairó, Oswaldo. Metodología de la Programación. Editorial Alfa Omega.
- Deitel & Deitel. C How To Program. Prentice Hall.

TEXTOS 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.

REVISTAS

DIRECCIONES DE INTERNET

V. ORGANIZACIÓN / TIEMPOS

Espacios, Tiempos, Agrupamientos:

Se recomienda trabajar una unidad cada cuatro semanas, trabajar en pequeños grupos de estudiantes, utilizar Internet para comunicarse con los estudiantes para revisiones de avances y solución de preguntas (esto considerarlo entre las horas de trabajo cooperativo).

PROGRAMA SINTÉTICO	SEMANAS ACADÉMICAS															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Reconocer la estructura y funcionamiento del computador.																
Sistemas numéricos																
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																
Primeros computadores																
El computador hasta hoy: Generaciones.																
Evolución de los lenguajes de programación.																
Estructura del computador: Procesador, memoria principal, memoria secundaria, E/S, buses																
Conceptualizar y abstraer problemas. Desarrollo de algoritmos.																
Concepto de algoritmo																
Los diagramas de flujo como herramienta de modelación de algoritmos.																
Pseudocódigo: Una herramienta de palabras útil.																
Modelar problema de solución secuencial																
Diseñar una solución algorítmica secuencial																
Analizar una solución algorítmica secuencial																
Modelar problema cuya solución involucra condiciones																
Diseñar solución algorítmica que involucra condiciones																
Analizar una solución algorítmica que involucra condiciones																
Modelar problema cuya solución involucra iteraciones																
Diseñar solución algorítmica que involucra iteraciones																
Analizar una solución algorítmica que involucra iteraciones																
Modelar problema complejo cuya solución amerita el uso de descomposición																
Diseñar solución algorítmica basada en descomposición																
Analizar una solución algorítmica basada en descomposición																
Diseñar soluciones algorítmicas para problemas computacionales																
Estructura de un programa en C, restricciones, comentarios																

Tipos de datos, variables y constantes: Caracteres, Boleanos, Reales, Enteros.																
Operadores																
Implementar prototipo de solución algorítmica secuencial																
Conversión entre tipos de datos																
Funciones de lectura y escritura.																
Arreglos y matrices. Definición, inicialización.																
Implementar prototipo de solución algorítmica que involucra condiciones: if, if else, switch.																
Implementar prototipo de solución algorítmica que involucra iteraciones: for, while, do while.																
Estructuras de salto: break, continue.																
Implementar prototipo de solución algorítmica basada en descomposición																
Funciones: Parámetros por valor, retorno de valores, variables locales, globales y estáticas.																
Librerías de funciones.																
Funciones recursivas.																
Apuntadores. Definición, asignación tipos y niveles de apuntadores, apuntadores a funciones, a arreglos, a matrices, arreglos de apuntadores a enteros, reales y a cadenas de caracteres.																
Registros o estructuras. Acceso a los elementos de una estructura, estructuras dentro de otras, arreglos de estructuras, estructuras con apuntadores a otras.																
Referencias: Parámetros de funciones por apuntador y por referencia.																
Manejo de archivos: persistencia de datos y flujo de datos.																

VI. EVALUACIÓN			
	TIPO DE EVALUACIÓN	**FECHA**	**PORCENTAJE**
PRIMERA NOTA	Prueba teórico – practica elaborada por el docente	Semana 5 ó 6	
SEGUNDA NOTA	Prueba escrita conjunta (para todos los grupos de la asignatura), elaborada por todos los docentes que imparten la asignatura.	Semana 14 ó 15	
TERCERA NOTA	Guías de ejercicios resueltas Informes de conceptos basado en análisis (Paper's) Pruebas orales/escritas rápidas (Quizes)	Varias fechas	
CUARTA NOTA	Informe de desempeño en laboratorio (Para 3 o 4)	Varias fechas	
PROYECTO FINAL	Informe de desempeño y sustentación de un prototipo funcional que evalúe las competencias exigidas.	Semana 17 - 18	30 %

ASPECTOS A EVALUAR DEL CURSO

- Claridad y entendimiento de los conceptos.
- Que se haya identificado correctamente el problema y que el modelo lo represente adecuadamente.
- Que la solución diseñada resuelva el problema.
- Apego a la formalidad y estándares requeridos.
- Que el análisis de corrección sea exhaustivo.
- Que el prototipo corresponda al algoritmo diseñado y no presente errores de sintaxis.
- La asistencia a las clases magistrales y a los laboratorios.
- El esfuerzo y dedicación en la resolución de problemas.
- Que la documentación permita reconocer la forma en que se ha abordado el problema y la estructura del programa implementado.
- En las pruebas escritas se consideran en forma parcial los aspectos considerados en proyectos de programación bajo problemas que requieren un menor tiempo de desarrollo y en una modalidad que no requiere uso del computador, así como la comprensión conceptual.

DATOS DEL DOCENTE
NOMBRE : **PREGRADO :** **POSTGRADO :**

ASESORIAS: FIRMA DE ESTUDIANTES

NOMBRE	**FIRMA**	**CÓDIGO**	**FECHA**
1. **2.** **3.**			

FIRMA DEL DOCENTE

_____

FECHA DE ENTREGA: _____