	**UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS** **FACULTAD DE INGENIERÍA** ***SYLLABUS*** **PROYECTOS CURRICULAR:** Ingeniería de sistemas

NOMBRE DEL DOCENTE:	
ESPACIO ACADÉMICO (Asignatura) : Programación Básica **Obligatorio (X) : Básico( ) Complementario ( )** **Electivo ( ) : Intrínsecas ( ) Extrínsecas ( )**	**CÓDIGO:** 2
NÚMERO DE ESTUDIANTES:	**GRUPO:**
NÚMERO DE CRÉDITOS: Tres (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” del proyecto curricular de ingeniería de sistemas.
Contribución a la formación	En esta asignatura se establece las bases del pensamiento algorítmico formal que constituye uno de los pilares de la disciplina y contribuye a los dominios de desempeño profesional definidos en el perfil. A través de esta, se pretende mostrar al estudiante de manera práctica, la evolución de los lenguajes y paradigmas que

	han surgido alrededor de la programación, la adquisición de los conceptos básicos acerca de la estructura y funcionamiento del computador, así como el desarrollo del pensamiento algorítmico formal fortaleciendo sus habilidades en el desarrollo de programas computacionales. Estas habilidades se reconocen como claves dentro del dominio del perfil de “Programación”.
Puntos de apoyo para otras asignaturas	En Ingeniería de Sistemas herramienta fundamental para: • Estructura lógica conceptual basada en paradigmas de programación. • Programación orientada a objetos, Programación avanzada y modelos de programación. Ingeniería de software. Bases de datos, Redes, Ciencias de la computación. En Ingeniería Industrial herramienta fundamental para: • Programación lineal, gestión tecnológica, teoría de colas y simulación, programación y control de producción, logística, gestión de operaciones. En Ingeniería Catastral herramienta fundamental para: • SIG, bases de datos, interfaces SIG. En Ingeniería Eléctrica herramienta fundamental para: • Área de circuitos, área de electrónica, probabilidad y estadística, sistemas dinámicos, redes de comunicaciones, digitales, herramientas computacionales, campos, generación hidráulica y generación térmica. En Ingeniería Electrónica herramienta fundamental para: • Digitales.
Requisitos previos	• Lógica

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 de tipo estructurado.

OBJETIVOS ESPECÍFICOS

1. Conocer la evolución de los lenguajes, los paradigmas y de 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.

COMPETENCIAS DE FORMACIÓN

Competencias que compromete la asignatura:	El estudiante está en capacidad de pensar ordenadamente para modelar una solución a un problema haciendo uso de la algoritmia, expresando esta solución en un lenguaje computacional.
Competencias específicas de la asignatura:	• Utiliza adecuadamente el concepto y la abstracción de los sistemas numéricos en la solución de problemas computacionales. • Localiza históricamente los diferentes momentos en la evolución de los sistemas computacionales. • Identifica los diversos componentes de un sistema computacional. • Representa soluciones de problemas aplicando el concepto de algoritmo. • Modela, implementa y evalúa problemas cuya solución algorítmica requiere el uso de las diferentes estructuras de control.

	• Modela, implementa y evalúa problemas descomponiéndolos en subproblemas que permitan una solución más simple o la reutilización de soluciones. • Resuelve problemas que requieren aplicar el concepto de recursividad. • Define e implementa tipos de datos abstractos. • Modela, implementa y evalúa mecanismos para el manejo dinámico de memoria y persistencia. • Identifica y aplica los conceptos básicos sobre metodologías.
Competencias transversales a las que contribuye la asignatura:	• Es capaz de discernir que tecnología debe utilizar para la resolución de problemas particulares. • Comunica ideas de forma clara oralmente o mediante la presentación de documentos escritos. • Actúa estratégicamente dentro de un grupo de trabajo para el desarrollo de proyectos.

PROGRAMA SINTÉTICO

1. Reconocer la estructura y funcionamiento del computador.

1.1. Sistemas numéricos: Sistema binario, hexadecimal y octal.
1.2. Conversiones entre sistemas. Números de precisión finita.
1.3. Representación de números negativos en base 2.
1.4. Representación de número punto flotante en base 2.
1.5. Operaciones.
1.6. Desarrollo histórico del “Hardware”: El ábaco, Maquinas de Pascal, Leibniz, Babbage, Turing.
1.7. Primeros computadores: Mark1, ENAC, EDSAC, UNIVAC 1, Von Newman.
1.8. El computador hasta hoy: Generaciones.
1.9. Evolución de los lenguajes de programación.
1.10. 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

3.1. Estructura básica de un programa, 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. Jerarquías 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. Funciones recursivas.
3.15. 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.
3.16. Registros o estructuras. Acceso a los elementos de una estructura, estructuras dentro de otras, arreglos de estructuras, estructuras con apuntadores a otras.
3.17. Referencias: Parámetros de funciones por apuntador y por referencia.
3.18. Manejo de archivos: persistencia de datos y flujo de datos.

4. Temas complementarios.

4.1. Conceptos básicos sobre metodologías.
4.2. Ejercicios de aplicación.

III. ESTRATEGIAS

Metodología Pedagógica y Didáctica:

- 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.

	Horas			Horas Lectivas/sem	Horas Estud.te/sem	Total Horas Estud.te/sem	Créditos
TIPO DE CURSO	**TD**	**TC**	TA	(TD + TC)	(TD + TC + TA)	X 16 semanas	
	2	4	6	6	12	192	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 de tutoría del docente a pequeños grupos o de forma individual a los estudiantes.

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.

PAGINAS DE INTERNET

- Karel el robot, http://www.olimpiadadeinformatica.org.mx/material/karel
- Guido van robot, http://gvr.sourceforge.net/
- FreeDFD, http://wiki.freaks-unidos.net/freedfd/index
- PseInt, http://pseint.sourceforge.net/
- Dev C++, http://www.bloodshed.net/devcpp.html
- Python programming language, http://www.python.org/
- pygame, http://pygame.org/news.html

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).

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	**Reconocer la estructura y funcionamiento del computador.**																
	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.																
	Primeros computadores: Mark1, ENIAC, EDSAC, UNIVAC 1, Von Newman.																
	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**																
	Concepto de algoritmo																
	Los diagramas de flujo como herramientas de modelación de algoritmos																
	Pseudocódigo: Una herramienta de palabras útil																
	Modelar un problema de solución secuencial																
	Diseñar una solución algorítmica secuencial																
	Analizar una solución algorítmica secuencial																

	secuencial																
	Modelar un problema cuya solución involucra condiciones																
	Diseñar una solución algorítmica que involucra condiciones																
	Analizar una solución algorítmica que involucra condiciones																
	Modelarproblema cuya solución involucra iteraciones																
	Diseñarsoluciónalgorítmica que involucra iteraciones																
	Analizar una solución algorítmica que involucra iteraciones																
	Modelar problema complejocuya soluciónamerita elusode descomposición																
	Diseñarunasolución algorítmica basada en descomposición																
	Analizarunasolución algorítmica basada en descomposición																
3	Diseñar soluciones algorítmicas para problemas computacionales																
	Estructura básica de un programa, restricciones, comentarios																
	Tipos de datos, variables y constantes: Caracteres, Boleanos, Reales, Enteros.																
	Operadores																
	Aritméticos: asignación suma, resta, multiplicación,división,módulo, incremento,decremento,ytodos asociados con una variable en una cantidad determinada.																
	Bitwise:And,Or,Orexclusivo, complemento,desplazamientoa izquierda y derecha, combinaciones con el operador de asignación.																
	Relacionales: menor que, mayor que , menor o igual que mayor o igual que, igual, diferente.																
	Booleanos: para la estructuración de expresiones: Not, And, Or. Jerarquias de los operadores.																
	Proposiciones. And, or, xor, tablas de verdad.																
	Implementar prototipo de solución algorítmica secuencial																
	Conversión entre tipos de datos																
	Funciones de lectura y escritura.																
	Arreglosy 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:Parámetrosporvalor, 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 elementosdeunaestructura, estructuras dentro de otras, arreglos de estructuras,estructurascon apuntadores a otras.																		
	Referencias: Parámetros de funciones por apuntador y por referencia.																		
	Manejo de archivos: persistencia de datos y flujo de datos.																		
4	Temas complementarios.																		
	Conceptos básicos sobre metodologías.																		
	Ejercicios de aplicación.																		

VI. EVALUACIÓN

	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
PRIMER CORTE			
SEGUNDO CORTE			
PROYECTO FINAL			30,00%

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:_____