UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS	**UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS** **FACULTAD DE INGENIERÍA** **SYLLABUS** **PROYECTO CURRICULAR DE INGENIERÍA ELÉCTRICA**	

Nombre del Docente

ESPACIO ACADÉMICO (Asignatura): PROGRAMACIÓN BÁSICA						**Código:**	
Obligatorio	☒	Básico	☒	Complementario		**002**	
Electivo		Intrínseco		Extrínseco			
Número de Estudiantes						**Grupo**	
Número de Créditos		**TRES (3)**					
TIPO DE CURSO:	Teórico		Práctico		Teórico - Práctico	☒	

Alternativas Metodológicas:

Clase Magistral	☒	Seminario		Seminario-Taller		Taller		Teórico - Práctico	☒
Proyectos Tutoriados		Otros							

HORARIO

DÍA	HORAS	SALÓN

I. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

Esta asignatura contribuye al desarrollo de la competencia “Piensa ordenadamente para modelar una solución a un problema haciendo uso de la algoritmia, expresando esta solución en un lenguaje computacional.” que se encuentra en el dominio de “programación” del área “básicas de ingeniería” del proyecto curricular de Ingeniería Eléctrica.

La Sociedad de la Información funciona en un nuevo espacio denominado “Ciberespacio”. Este nuevo espacio facilita la comunicación entre personas, las cuales, debido a su naturaleza de integrarse, ven en este “lugar” una gran posibilidad para actualizarse, compartir, transferir conocimientos, realizar comercio electrónico, comunicarse y aprender. Se considera que el estado evolutivo de esta sociedad concluirá en una sociedad denominada sociedad del conocimiento, en donde el conocimiento será la materia prima para producir riqueza y poder. En donde el Ingeniero Electricista podrá aplicar conceptos sobre como: Analizar, modelar, seleccionar, evaluar, diseñar e implementar soluciones de componentes y sistemas básicos relacionados con su profesión.

En esta nueva sociedad ya no son válidos ciertos modelos ni ciertos conceptos tradicionales, pues el mundo se rige actualmente según un nuevo orden que es la globalización. Los cambios que ha introducido la tecnología en nuestra forma de vida hacen necesario replantear los modelos tradicionales en casi todos los campos siendo uno de ellos la Informática. La informática ha venido a revolucionar la vida del ser humano, pues está presente en todos los campos del conocimiento tales como Artes, Literatura, Derecho, filosofía, medicina, arquitectura, administración y por supuesto en la rama de la Ingeniería.

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".

El área de Informática es la base de la revolución tecnológica, que ha dado incluso origen a una nueva sociedad denominada la Sociedad de la Información y la relación de esta con procesos del Gestión del Conocimiento está dando origen a la transformación en la Sociedad del Conocimiento.

La informática se soporta en la integración de metodologías para construcción de software, lenguajes de modelado, lenguajes de programación, sistemas operativos y en la integración de estos con computadores. El fundamento de la Informática han sido los Lenguajes de programación, en un principio se utilizaron lenguajes de programación estructurada, pero con el pasar del tiempo evolucionaron hacia los lenguajes de programación orientada a objetos (C++, Java) y ahora último con la aparición de nuevos lenguajes de programación orientada a objetos (C#, J#, C++), para operar en plataformas múltiples tenemos una herramienta poderosa para realizar desarrollos de una forma sencilla y rápida.

Los cambios que ha introducido la tecnología en nuestra forma de vida hacen necesario replantear los modelos tradicionales en casi todos los campos siendo uno de ellos la Informática, (Ciencia Básica para la Gestión Tecnológica). La informática no es solo programación de computadores, se soporta en la integración de Metodologías para construcción de software, Lenguajes para modelado, Lenguajes de programación, Sistemas operativos y la aplicación de otras tecnologías en el campo de las comunicaciones, para conformar la base de la Gestión Tecnológica

Puntos de apoyo para otras asignaturas:

- Estructura lógica conceptual basada en programación.
- Herramienta fundamental para Programación orientada a objetos.
- Analizar, modelar, seleccionar, evaluar, diseñar e implementar componentes y sistemas básicos en las diferentes asignaturas de la carrera.
- Puente para comprender como es el modelo multicapa para luego hacer el desarrollo de bases de datos
- Materia transversal para elaborar proyectos en circuitos.

Conocimientos Previos:

Lógica

II. PROGRAMACIÓN DEL CONTENIDO

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 y un lenguaje de programación 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, y no estructurado.

OBJETIVOS ESPECÍFICOS

Al finalizar la materia el alumno estará en capacidad de:

- Conocer la evolución de los lenguajes, los paradigmas y de la computación.
- Evidenciar de manera clara y concreta la evolución de la programación con relación a la evolución del computador.
- Identificar la estructura de un computador.
- Desarrollar el concepto de algoritmo y aplicarlo en la solución de programas sencillos
- Solucionar problemas elementales utilizando la lógica computacional
- Resolver problemas sobre el sistema computacional con la ayuda de un lenguaje de programación.
- Reconocer la sintaxis básica del lenguaje de programación escogido (para Ingeniería Eléctrica C#).

COMPETENCIAS DE FORMACIÓN

Competencias de Contexto:

- 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 Básicas:

- 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 dato abstracto.
- Modela, implementa y evalúa mecanismos para el manejo dinámico de memoria y persistencia.
- Realizar un proceso de análisis del problema antes de intentar emprender cualquier procedimiento de desarrollo, sin seguir una metodología.
- Manejar un lenguaje de modelado visual como UML, con el fin de que cree los modelos, los visualice, especificar su estructura y el comportamiento, generar el código mediante la ayuda de una herramienta CASE y adquiera la disciplina de documentar el proceso.
- Utilizar los principios básicos de la programación orientada a objetos “Alta cohesión, Débil acoplamiento”, para que las clases que construya estén ajustadas a los estándares.
- Diseñar algoritmos a partir del planteamiento de un problema utilizando diagramas ajustados a UML.
- Implementar programas en C#, compilarlos y ejecutarlos.
- Conocer los diferentes entornos y procesos de compilación y ejecución para los diferentes lenguajes de programación y la forma como se administra la memoria.
- Aprender a evaluar el Software resultado del proceso de la Metodología.
- Aprender a manejar los depuradores y las Herramientas de desarrollo, que le dará posibilidades de continuar de forma autónoma

Competencias Laborales:

- Habilidades para resolver problemas en la vida laboral
- Trabajo en equipo y colaboración
- Creatividad e Innovación
- Disciplina de trabajo
- Defensa de argumentos y la resolución de problemas dentro de su área de trabajo.
- Aplicación de procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas.

PROGRAMA (UNIDADES TEMÁTICAS)

I.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, ENAC, 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

II. 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 un problema de solución secuencial
- Diseñar una solución algorítmica secuencial
- Analizar una solución algorítmica secuencial
- Modelar un problema cuya solución involucra condiciones
- Diseñar una 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 una solución algorítmica basada en descomposición
- Analizar una solución algorítmica basada en descomposición

III. Metodología RUP.

- Requerimientos para el RUP, técnicas, formatos empleados, ejercicios prácticos
- Fundamentos de la metodología.
- Objetivos de la metodología.
- Las seis (6) mejores prácticas (Administración de requerimientos, Desarrollo Iterativo, Modelamiento visual, Verificación de la calidad, Arquitectura con componentes, Control de cambios),
- Etapas de la metodología (Análisis, Diseño y Desarrollo, Pruebas, Evaluación),
- Ciclo de vida del Desarrollo de Software (Inicio, Elaboración, Construcción, Transición, Flujo de trabajo de procesos, Flujo de trabajo de soporte)
- Lenguaje de Modelado UML.

√ Objetivos de UML(Visualizar, Especificar, construir, Documentar).

√ Modelo conceptual de UML: Bloques de construcción UML: Elementos UML: estructurales, de comportamiento, de agrupación, de anotación. Relaciones UML:

√ Dependencia, asociación, agregación, Generalización, Realización. Diagramas: Casos de uso, Clases, Componentes, Secuencia, Diagramas de Colaboración, Actividades. (Haciendo énfasis en los diagramas de clases, casos de uso y secuencia). Reglas UML: Nombres, visibilidad, Integridad, Ejecución. Mecanismos Comunes: Especificaciones, adornos, divisiones comunes, Mecanismos de extensibilidad.

√ Manejo de herramientas CASE. Instrucción teórica de algunas herramientas libres y manejo de las mismas para la elaboración de diagramas en UML.

IV. Diseñar soluciones no estructuradas para problemas computacionales (Basado en el lenguaje de programación escogido en este caso C#.)

- Estructura de un programa en C#, restricciones, comentarios
- Tipos de datos, variables y constantes: Caracteres, Boleanos, Reales, Enteros.
- Introducción a Clases y Objetos (notación UML): Definición de Clases y Objetos, atributos y métodos. Comportamiento, Conceptos relacionados con el paradigma de objetos, Clases,
- Introducción a los lenguajes de Programación Orientados a Objetos: Descripción general de ella, principios básicos: “Alta cohesión, Débil acoplamiento”. Conocer en qué consiste la Herencia, el Polimorfismo y la Encapsulación. Principales lenguajes de Programación Orientada a Objetos.
- Operadores: Aritméticos, Relacionales, Booleanos, Proposiciones.
- Conversión entre tipos de datos
- Creación de una solución de dos (2) capas (presentación y lógica de aplicación) uso de herramientas y

sus propiedades, uso de los exploradores, Uso del depurador

- Fundamentos de Programación con C#:
- 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.

III. ESTRATEGIAS

- 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)

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

Trabajo Directo (TD): Trabajo de aula con plenaria de todos los estudiantes.
Trabajo 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 vali3dar 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 Guías

- LARMAN, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Rational Software. Manuales y ayudas de Rational Rose. 2009.
- Microsoft. Manuales de las herramientas de desarrollo Visual Basic, Visual C++, C#, Java y Visual Studio NET.2018.
- BOOCH, Gaddy. CUMMINGS, Benajamin. Object Oriented Analysis and Design with applications. 2 Edición. Ed. Publishing: Colombia.1994.
- WEITZENFELD, Alfredo. Ingeniería de Software orientada a objetos con UML. Ed Thomson: México. 2005.
- SCHACH, Stephen. Ingeniería de Software Clásica y Orientada a Objetos. 6 Edición. Ed McGrawHill:

México. 2006.
- CHAPPELL, David. Aplique. Net. Ed Prentice Hall: México. 2003.
- LARMA, Craig. Uml y Patrones. 2 Edición. Ed Pearson Addison-Wesley: España. 2006.
- BECERRA, Cesar. Una herramienta para la programación orientada a Objetos. 5 Edición. Ed Kimpres Ltda: Bogotá. 2006
- SILVIO, Jose. La Virtualización de la Universidad ¿cómo podemos transformar la educación superior con la tecnología?. 2000.
- TANENBAUM, Andrew. Structured Computer Organization. Prentice Hall.
- LEVINE, Guillermo. Computación y Programación Moderna. Addison Wesley.
- CAIRÓ, Oswaldo. Metodología de la Programación. Editorial Alfa Omega.
- CEBALLOS, Javier. Curso de Programación, Microsoft C#. ED Alfaomega. Ed 2007.
- CEBALLOS, Javier. Visual C#, Enciclopedia de Microsoft. ED Alfaomega. Ed Segunda 2007.
- ARCHER Tom, WHITECHAPEL Andrew. Inside C#. ED Microsoft Press. Ed Segunda 2002Design, and the Unified Process. 2 Ed. Prentice Hall. 2001.
- JACOBSON, Ivar. WESLEY, Addison. Object-Oriented Software Engineering: A Use Case Driven Approach. 1992
- MULLER, Alain. WROX, Pierre. Press Instant UML. 1997 / 1861000871
- BOOCH, Gaddy. RUMBAUGH, James. JACOBSON, Ivar. El lenguaje unificado de modelado. 2 Edición. España: Ed Pearson Addison-Wesley, 2008..

Textos Complementarios

- FERNÁNDEZ, Gregorio. Curso de Ordenadores. Conceptos básicos de arquitectura y sistemas operativos, (4ª Edición). Servicio de Publicaciones de la E.T.S.I. Telecomunicación de Madrid, 2003.
- RUBIO GONZÁLEZ Miguel Ángel. Introducción A La Informática Básica. Editorial UNED. 2017.
- CLYDE hatter. Aprende a programar. Editorial Malpaso ediciones Sl. 2017.
- MAESTRE TORREBLANCA, Jose Mª, RELINQUE PÉREZ, Jesús. A programar se aprende jugando. Editorial: Paraninfo. 2017

Revistas

- Jia-Sheng H. Jyh-Cheng C. Shao-Chun L. Chang, M. (2008). Providing students hints and detecting mistakes made by students in a virtual experiment environment education, IEEE Transactions on. Volume: 51, Digital Object Identifier: 10.1109/TE.2007.901977. Pages: 61 – 68
- Lucanin, D. Fabek, I. (2011). A visual programming language for drawing and executing FLOWCHARTS. Proceedings Of The 34th International Convention Publication Year, Page(s): 1679 – 1684.
- Anfurrutia, Felipe I. Álvarez, Ainhoa. Larrañaga, Mikel. López-Gil, Juan Miguel.(2017). Lecciones aprendidas de experiencias con robots educativos y entornos de programación visuales en asignaturas de programación. Revista Iberoamericana de Informática Educativa págs. 9-22.
- BATTAGLIA, Nicolás et al. Integración de una Herramienta CASE en un Entorno Académico Colaborativo para la Enseñanza de Ingeniería de Software. Revista Abierta de Informática Aplicada (RAIA), [S.l.], v. 3, n. 2, p. 31-42, sep. 2019. ISSN 2591-5320. Disponible en: <http://portalreviscien.uai.edu.ar/ojs/index.php/RAIA/article/view/206>. Fecha de acceso: 27 Nov. 2019.
- Vera Paredes, D. A., Córdova Martínez, L. C., López Bermúdez, R. M., & Pacheco Mendoza, S. R. (2019). Análisis de la metodología RUP en el desarrollo de software académico mediante la herramienta DJANGO. RECIMUNDO, 3(2), 964-979. https://doi.org/10.26820/recimundo/3.(2).abril.2019.964-979.

Direcciones de Internet

- DFD. Editor e intérprete de Diagramas de Flujo. Descarga de programa gratis. <http://dfd.softonic.com/> [Consultado 2019]
- KAREL, el Robot. Descarga de programa gratis. <http://karel-the-robot.softonic.com/palm.> [Consultado 2019].

- ALICE. Programa en 3D. Descarga de programa gratis. <http://www.alice.org>. [Consultado 2019].

V. ORGANIZACIÓN / TIEMPOS

Espacios, Tiempos, Agrupamientos

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.																
Conceptualizar y abstraer problemas. Desarrollo de algoritmos.																
Metodología RUP.																
Diseñar soluciones no estructuradas para problemas computacionales *(Basado en el lenguaje de programación escogido en este caso C#.)*																

VI. EVALUACIÓN

	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
PRIMER CORTE		Semana 8	
SEGUNDO CORTE		Semana 16	
EXAMEN FINAL		Semana 17 -18	

ASPECTOS A EVALUAR DEL CURSO

1. Claridad y entendimiento de los conceptos.
2. Que se haya identificado correctamente el problema y que el modelo lo represente adecuadamente.
3. Que la solución diseñada resuelva el problema.
4. Apego a la formalidad y estándares requeridos.
5. Que el análisis de corrección sea exhaustivo.
6. Que el prototipo corresponda al algoritmo diseñado y no presente errores de sintaxis.
7. La asistencia a las clases magistrales y a los laboratorios.
8. El esfuerzo y dedicación en la resolución de problemas.
9. Que la documentación permita reconocer la forma en que se ha abordado el problema y la estructura del programa implementado.
10. 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 PROFESOR

Nombre:	
Pregrado:	
Postgrado:	
Correo Electrónico:	