UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

SYLLABUS

PROYECTO CURRICULAR DE INGENIERÍA INDUSTRIAL

Espacio Académico: Programación básica

Código: 2

Obligatorio	☒	Básico	☒	Complementario	☐
Electivo	☐	Intrínseco	☐	Extrínseco	☐

Número de Créditos	3			**Semestre: II**			
Tipo de Curso:	Teórico	☐	Práctico	☐	Teórico - Práctico	☒	

Alternativas Metodológicas:

Clase Magistral	☒	Seminario	☐	Seminario-Taller	☐	Taller	☒	Prácticas	☒
Proyectos Tutoriados			☐	Otros	Haga clic aquí para escribir texto.				

I. JUSTIFICACIÓN DEL ESPACIO ACADÉMICO

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

Conocimientos Previos: No aplica

II. PROGRAMACIÓN DEL CONTENIDO

OBJETIVO GENERAL

Comprender y aplicar los conceptos de la programación, de tal manera que pueda obtener soluciones a problemas sencillos apoyados en: un computador, un lenguaje de programación y un paradigma algorítmico; en donde el alumno sea capaz de enfrentarse a situaciones o problemas en las que debe identificar los elementos y estados involucrados.

OBJETIVOS ESPECÍFICOS

1. Conocer la evolución de los lenguajes de programación, los paradigmas de desarrollo de software 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. Generar modelos para su representación y manipulación algorítmica.
7. Debe ser capaz de diseñar soluciones para los problemas planteados.
8. Validar su corrección e implementar prototipos para ellas utilizando un lenguaje de programación de tipo estructurado.
9. Resolver problemas sobre el sistema computacional con la ayuda de un lenguaje de programación.
10. Reconocer la sintaxis básica del lenguaje de programación escogido (preferiblemente C).

COMPETENCIAS DE FORMACIÓN

Competencias de Contexto:

Haga clic aquí para escribir texto.

Competencias Básicas:

Haga clic aquí para escribir texto.

Competencias Laborales:

Esta asignatura contribuye al desarrollo de la competencia "Estructura de pensamiento sistémico" a través del uso de la algoritmia en el estudiante con el fin de que analice problemas de cualquier índole y sea capaz de diseñar e implementar su solución. La asignatura se encuentra en el dominio de "programación" del área "básicas de ingeniería" del proyecto curricular de ingeniería Industrial.

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 *(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. 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.

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

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 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 Guías

- Deitel,H & Deitel, P. *C How to program.* New Jersey, United States of America. Editorial: Pearson Prentice Hill.
- Cairó, Oswaldo. Metodología de la Programación. Editorial Alfa Omega.

Textos Complementarios

- Tanenbaum, Andrew. Structured Computer Organization. Editorial: Prentice Hall.
- Levine, Guillermo. Computación y Programación Moderna. Addison Wesley.
- Rodriguez C., Llana L.F, Martinez, R.,Palao P., Pareja, C. Ejercicios de Programación Creativos y Recreativos en C ++. Editorial: Prentice Hall.
- Becerra, Cesar. Lenguaje. Editorial: Por computador

Revistas

Haga clic aquí para escribir texto.

Direcciones de Internet

http://www.mygnet.net/manuales/java//guia_java.1691

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).

TEMA No.	TEMA A DESARROLLAR	SEMANAS ACADÉMICAS															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
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, VonNewman.	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
1.8	El computador hasta hoy: Generaciones.	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
1.9	Evolución de los lenguajes de programación. principal,memoria	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
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 (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	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐
3.4	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	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒

VI. EVALUACIÓN

	TIPO DE EVALUACIÓN	FECHA	PORCENTAJE
PRIMER CORTE	Guías de ejercicios resueltas Pruebas orales/escritas rápidas (Quices) Prueba oral/escrita para el grupo que el docente elabora.	Semana 8 de clases	35%
SEGUNDO CORTE	Guías de ejercicios resueltas Pruebas orales/escritas rápidas (Quices) Prueba oral/escrita para el grupo que el docente elabora.	Semana 16 de clases	35%
EXAMEN FINAL	Proyecto final, Informe de desempeño y sustentación de un prototipo funcional que evalúe las competencias exigidas.	Semana 17 -18 de clases	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.