DECC | PUJC

Departamento de Electrónica y Ciencias de la Computación

Estructura de Datos

Información Básica

- Créditos: 3
- Horas de trabajo acompañado: 5 / semana (3 horas clase, 2 horas taller)
- Horas de trabajo independiente: 4 / semana
- Pre-requisitos: Introducción a la Programación, Introducción a la Ingeniería de Sistemas y Computación
- Tipo de curso: Núcleo de Formación Fundamental.

Descripción del Curso

Este curso cubre los temas fundamentales de las principales estructuras de datos lineales, así como herramientas importantes para implementar dichas estructuras de datos

Objetivos

Al finalizar el curso los participantes podrán:

1. Identificar la eficiencia de un algoritmo.
    1. Hallar la complejidad de algoritmos iterativos.
    2. Comparar diferentes algoritmos en términos de complejidad.
    3. Encontrar el tiempo de ejecución de un programa
    4. Comparar tiempos de ejecución teóricos y prácticos
2. Detallar las principales características de los lenguajes de programación de alto nivel.
    1. Describir la sintaxis básica de un lenguaje de programación de alto nivel.
    2. Programar usando librerías, sus APIs y ambientes modernos de programación.
    3. Utilizar estándares de codificación, documentación en el código y contratos.
    4. Aplicar estrategias de depuración para encontrar y solucionar errores, y programación defensiva para proteger código inseguro.
    5. Describir las características de las variables en lenguajes de programación de alto nivel

(e.g. tipos de datos, identificadores, alcance, visibilidad).

3. Diseñar e implementar programas en el paradigma orientado a objetos.
    1. Definir e implementar objetos y clases.
    2. Aplicar nociones de programación orientada a objetos.
    3. Usar principios de diseño orientado a objetos.
    4. Describir y utilizar paradigmas de diseño.
    5. Implementar programas en el paradigma orientado a objetos y realizar pruebas unitarias en ellos.
4. Representar información por medio de estructuras de datos lineales.
    1. Diseñar, implementar y evaluar tipos abstractos de datos.
    2. Diseñar, implementar y usar estructuras de datos lineales.
    3. Aplicar conceptos básicos de programación orientada a objetos en la implementación de estructuras de datos.
    4. Describir las características de las estructuras de datos en memoria secundaria.
    5. Escoger la estructura de datos apropiada para resolver problemas de media escala.

Competencias técnicas específicas que se desarrollan

1. Modelado de problemas por medio de estructuras de datos.
2. Desarrollo de programas usando el lenguaje de programación C++.
3. Conocimiento de nuevos paradigmas de programación.
4. Aplicación de técnicas y herramientas de lenguajes de programación de alto nivel.
5. Implementación de software con buenas prácticas de programación.

Contenido

Capítulo 1: Complejidad de algoritmos, características de los lenguajes de programación y detalles de implementación

Sesión	Horas teóricas	Prácticas acompañadas	Temas	Profundidad	Bibliografía
1	3		Diferencias entre el mejor de los casos, el caso esperado y el peor de los casos de los tiempos de ejecución de un algoritmo.	Familiaridad	[3]

			Clases de complejidad: constante, logarítmica, lineal, cuadrática y exponencial. Comparación informal de la eficiencia de algoritmos (e.g. conteo de operaciones).		
2		2	Tiempos de ejecución de un algoritmo.	Evaluación	[3]
3	3		Sintaxis básica de un lenguaje de programación de alto nivel. Programación usando librerías y sus APIs. Ambientes modernos de programación.	Uso	[6,2]
4		2	Abstracciones, interfaces y uso de librerías. Code search.	Evaluación	[6,2]
5	3		Estándares de codificación. Documentación y estilos de programas. El rol y uso de contratos, incluyendo pre- y post-condiciones.	Uso	[1,2]
6		2	Estrategias de Depuración. Programación defensiva (e.g. asegurando el código, manejo de excepciones).	Evaluación	[7,2]
7	3		Variables y tipos básicos primitivos (e.g. números, caracteres, Booleans).	Uso	[1,2]

			Referencias y apuntadores. Arreglos. Registros/Structs. Cadenas y procesamiento de cadenas.		
8		2	Asociación de tipos a variables, argumentos y resultados. Construcción de tipos compuestos a partir de otros tipos (e.g. registros, unions, arreglos, listas, funciones, referencias). Seguridad en tipos y errores causados por el uso de valores inconsistentes dados los tipos de datos.	Evaluación	[1,2]

Total de Horas: 20.

Sesión	**Horas de trabajo independiente**	**Temas**	**Bibliografía**
1-2	4	Tomar diferentes algoritmos iterativos y hallar su complejidad. Luego ejecutar el algoritmo en un computador, calcular el tiempo de ejecución y comparar este tiempo con el tiempo teórico hallado a partir de la complejidad.	[3]
3-4	4	Resolver problemas usando el lenguaje de programación C++ y sus librerías.	[1,2]
5-6	4	Dados unos programas escritos en C++, usando alguna estrategia clara de depuración, encontrar y solucionar los errores que tienen dichos programas, y luego proteger el código usando programación defensiva.	[1,2,7]

7-8	4	Implementar algoritmos en C++ teniendo en cuenta las características de las variables (paso de parámetros por valor y referencia, apuntadores, tipos de datos, etc.).	[1,2]

Total de Horas: 16

Capítulo 2: Programación orientada a objetos

Sesión	Horas teóricas	Prácticas acompañadas	Temas	Profundidad	Bibliografía
9	3		Definición de clases: atributos, métodos y constructores. Privacidad y visibilidad de los miembros de una clase.	Evaluación	[1,2]
10		2	Uso de clases, iteradores y otros componentes comunes.	Uso	[1,2]
11	3		Herencia y polimorfismo. Interfaces y clases abstractas.	Uso	[1,2]
12		2	Descomposición de programas. Descomposición en objetos con estado. Encapsulamiento y ocultación de información. Separación de comportamiento e implementación.	Familiaridad	[2]
13	3		Principios de diseño de sistemas: niveles de	Uso	[2,8,9]

			abstracción (diseño estructural y diseño detallado), separación de tareas, ocultamiento de información, acoplamiento y cohesión, re-uso de estructuras estándar.		
14		2	Relaciones entre requerimientos y diseños: transformación de modelos, diseño de contratos, invariantes.	Familiaridad	[8,9]
15	3		Paradigmas de diseño como diseño estructurado (descomposición funcional top-down), análisis y diseño orientado a objetos, diseño orientado a eventos, diseño a nivel de componentes, orientado a datos estructurados, orientado a aspectos, orientado a funciones, orientado a servicios. Conceptos de verificación y validación.	Uso	[8,9]
16		2	Cualidades internas de diseño y modelos para ellas: eficiencia y rendimiento, redundancia y tolerancia a fallos, trazabilidad de requerimientos. Pruebas unitarias.	Uso	[2,8,9]

Total de Horas: 20.

Sesión	Horas de trabajo independiente	Temas	Bibliografía
9-10	4	Implementar clases que modelen categorías clásicas de la vida real (con sus atributos y operaciones) y crear objetos de dichas clases.	[1,2]
11-12	4	Utilizar herencia y polimorfismo en la solución de problemas dados.	[1,2]
13-14	4	Construir algoritmos que resuelvan problemas dados con un diseño orientado a objetos.	[1,2,8,9]
15-16	4	Tomar programas desarrollados con el paradigma orientado a objetos y crear pruebas para verificarlos y validarlos dado un diseño.	[1,2,8,9]

Total de Horas: 16

Capítulo 3: Tipos abstractos de datos

Sesión	Horas teóricas	Prácticas acompañadas	Temas	Profundidad	Bibliografía
17	3		Tipos Abstractos de datos.	Uso	[3,4,5]
18		2	Implementación de tipos abstractos de datos	Evaluación	[1,3,4,5]
19	3		Listas enlazadas.	Evaluación	[3,4,5]
20		2	Uso de listas enlazadas.	Evaluación	[3,4,5]
21	3		Implementación de listas enlazadas.	Evaluación	[1,3,4,5]
22		2	Implementación de listas enlazadas.	Evaluación	[1,3,4,5]

23	3		Pilas. Colas. Colas de prioridad. Conjuntos. Maps.	Evaluación	[3,4,5]
24		2	Implementación de pilas y colas.	Evaluación	[1,3,4,5]
25	3		Conjuntos, maps y otras estructuras de datos basadas en listas.	Uso	[2,3,4,5]
26		2	Implementación de otras estructuras de datos basadas en listas.	Uso	[3,4,5]
27	3		Tablas Hash. Estrategias para evitar y resolver colisiones.	Uso	[4]
28		2	Implementación de tablas Hash.	Uso	[1,4]
29	3		Representación de datos no-numéricos (códigos caracter, datos gráficos). Representación de registros y arreglos. Almacenamiento y estructura de archivos. Directorios: contenido y estructura. Archivos indexados. Archivos Hashed.	Familiaridad	[3]
30		2	Práctica de estructura de datos en memoria secundaria.	Familiaridad	[1,3]
31	3		Recorridos iterativos y recursivos de estructura	Uso	[3]

			de datos. Estrategias de dividir y conquistar.		
32		2	Estrategias para escoger la estructura de datos apropiada.	Evaluación	[1,2,3,4,5]

Total de Horas: 40.

Sesión	Horas de trabajo independiente	Temas	Bibliografía
17-18	4	Crear formalmente tipos abstractos de datos a partir de diferentes sistemas e implementarlos en C++.	[1,3,4,5]
19-20	4	Implementar y usar listas (enlazadas).	[1,3,4,5]
21-22	4	Implementar listas (vectores y cursores).	[1,3,4,5]
23-24	4	Implementar y usar pilas y colas.	[1,3,4,5]
25-26	4	Implementar otras estructuras de datos basadas en listas utilizando el poder de la programación orientada a objetos.	[1,3,4,5]
27-28	4	Implementar y usar tablas hash.	[1,3,5]
29-30	4	Experimentar diferentes tipos de archivos como secuenciales, relativos e indexados para almacenar y recuperar eficientemente datos.	[1,3,5]
31-32	4	Dados diferentes problemas de media escala, decidir y utilizar la estructura de datos adecuada para resolver dichos problemas.	[1,2,3,4,5]

Total de Horas: 20

Integración Curricular

Resultados de Programa (ABET)

(A) La habilidad para aplicar conocimientos de matemáticas, ciencias e ingeniería.

(B) La habilidad para analizar un problema e identificar los requerimientos necesarios para su definición y solución.

(C) La habilidad para diseñar, implementar y evaluar procesos y sistemas computacionales.

(D) La habilidad para funcionar en equipos de trabajo.

(E) El entendimiento de la responsabilidad profesional y ética.

(F) La habilidad para comunicarse efectivamente.

(G) La habilidad para analizar los impactos de la computación y la ingeniería en las personas, organizaciones y la sociedad.

(H) El reconocimiento de la necesidad de, y la habilidad para, continuar con el desarrollo profesional.

(I) La habilidad para usar las técnicas, destrezas y herramientas modernas para la práctica de la computación.

(J) La habilidad para aplicar los fundamentos y principios de las matemáticas y de la computación en el modelamiento y diseño de sistemas computacionales de manera que se demuestre comprensión de las ventajas y desventajas en las decisiones de diseño.

(K) La habilidad para aplicar los principios de diseño y desarrollo de software en la construcción de sistemas de diferente complejidad.

Relevancia del curso con los resultados de programa

Resultados de Programa											
	A	B	C	D	E	F	G	H	I	J	K

Relevancia	2		5			4			4	1	4

Escala: (1) baja relevancia – (5) alta relevancia.

Integración de objetivos, contenido y metodología del curso

El curso es presencial y con participación y trabajo en clase. Se asignarán investigaciones, ejercicios y lecturas. Durante la sesión se expondrán los conceptos acompañados de ejemplos, se fomentará la participación de los estudiantes. Se realizará un proyecto final en el que se ponga en práctica los conceptos de computación vistos en la clase.

Resultados del Programa	**Indicadores de Desempeño**	**Objetivos/Contenido del Curso**	**Actividades de aprendizaje**	**Instrumentos de medición**
(A) Aplicación de Conocimientos	(A1) Identificar los fundamentos científicos y los principios de ingeniería que rigen un proceso o sistema. (Conocimiento) (A2) Resolver problemas relacionados con la disciplina y otras áreas por medio de la utilización de conocimientos, modelos y formalismos de las ciencias de la computación, las matemáticas y la	Capítulo 1	Exposiciones del profesor, solución de ejercicios y lecturas	Exámenes y tareas

	ingeniería. (Aplicación) (A3) Analizar conjuntos de datos. (Análisis)			
(C) Diseño	(C1) Utilizar estándares de codificación en la implementación de componentes de software. (Aplicación). (C2) Identificar componentes, interacciones, relaciones e interfaces entre componentes. (Análisis).	Capítulos 2 y 3	Solución de ejercicios y lecturas	Exámenes, proyecto y tareas
(F) Comunicación efectiva	(F1) Producir textos de manera efectiva teniendo en cuenta la estructura, coherencia, flujo, ortografía y correcto uso del lenguaje. (Aplicación). (F2) Comunicarse de manera efectiva de acuerdo al público objetivo haciendo uso correcto del	Capítulos 1, 2 y 3	Examenes, proyectos, lecturas y tareas	Reportes escritos

	lenguaje, estilo, tiempo y expresión corporal. (Aplicación). (F3) Utilizar recursos gráficos para comunicar y expresar una idea. (Aplicación). (F4) Defender ideas con precisión y claridad. (Evaluación).			
(I) Uso de herramientas y técnicas	(I1) Utilizar herramientas de desarrollo de software. (Aplicación). (I2) Utilizar herramientas de diseño, modelamiento y simulación. (Aplicación). (I3) Combinar herramientas de software y hardware para resolver un problema. (Síntesis). (I4) Demostrar flexibilidad para adaptarse a	Capítulos 1, 2 y 3	Laboratorios	Examenes, proyecto y tareas

	diferentes paradigmas y lenguajes de programación. (Valuación).			
(J) Modelamiento y diseño de sistemas computacionales	(J1) Reconocer la importancia del modelamiento cuando se resuelve un problema. (Compresión). (J2) Relacionar conceptos y principios teóricos para la resolución efectiva de un problema. (Síntesis).	Capítulos 2 y 3	Solución de ejercicios	Exámenes, proyectos y tareas
(K) Desarrollo de software	(K2) Implementar e integrar componentes de software respetando los criterios de diseño. (Aplicación). (K3) Establecer invariantes y propiedades de componentes de software. (Análisis). (K4) Evaluar y verificar soluciones de	Capítulo 2	Solución de ejercicios	Exámenes, proyectos y tareas

	software con respecto a las restricciones y requerimientos establecidos. (Aplicación – Evaluación).			

Contribución al Desarrollo de Competencias Genéricas

La tabla muestra que aspectos de las competencias de Comunicación Escrita, Lectura Crítica y Razonamiento Cuantitativo son evaluados a través de los factores ABET correspondientes. Por otra parte, las competencias de Ciudadanía e Inglés se favorecen gracias a la metodología del curso y también, gracias a los factores ABET correspondientes.

Resultados de Programa											
	A	B	C	D	E	F	G	H	I	J	K
Ciudadanía					U		U				
Comunicación escrita						E					
Lectura crítica						U					
Inglés						U					
Razonamiento cuantitativo	E		E						E	E	E

E- Se evalúa. U – Se usa

Contribución a los objetivos educacionales

La Carrera de Ingeniería de Sistemas y Computación plantea los siguientes objetivos educacionales, El estudiante graduado de la carrera será capaz de:

1. EO1. Ejercitar la práctica de la Ingeniería de Sistemas y Computación profesionalmente.
2. EO2. Diseñar y operar sistemas de computación que contribuyen a la solución de problemas relacionados a la disciplina, otra área de la ciencia y la ingeniería u otras disciplinas.

3. EO3. Contribuir al bienestar de las comunidades desde posiciones prominentes en la industria, academia, sector público o como un emprendedor.
4. EO4. Ser distinguido por su bases sólidas en computación, su sentido de ciudadanía responsable, su profesionalismo y liderazgo.
5. EO5. Continuar su desarrollo profesional o involucrarse en estudios de posgrado.

Resultados de Programa											
	A	B	C	D	E	F	G	H	I	J	K
EO1	X		X						X	X	
EO2	X		X						X	X	X
EO3						X					X
EO4	X					X				X	
EO5	X										

A través de los factores ABET declarados en la fórmula del curso, éste contribuye a los objetivos educacionales del programa y de esta manera se relaciona con los planes educativos del programa y de la Universidad.

Reglas del curso

Calificación y Balance de Evaluación del Curso

Instrumento	**Porcentaje**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Parcial 1	20%	7%					4%			9%		
Parcial 2	20%	3%		5%			4%					8%
Parcial 3	20%			10%			4%					6%
Tareas	10%						2%			5%		3%
Proyecto	30%			10%			6%			6%	5%	3%

Uso de material en exámenes

No está permitido el uso de computadores personales, teléfonos celulares ó cualquier otro dispositivo electrónico.

Asistencia

Obligatoria.

Bibliografía

1. Bjarne Stroustrup. "The C++ Programming Language (4th Edition)". Addison-Wesley. May, 2013.
2. Bjarne Stroustrup. "Programming: Principles and Practice Using C++ (2nd Edition)". Addison-Wesley. May, 2014.
3. Clifford A. Shaffer. "Data Structures and Algorithm Analysis in C++ (3rd Edition)". Dover Publications. September, 2011.
4. Michael T. Goodrich, Roberto Tamassia, and David M. Mount. "Data Structures and Algorithms in C++ (2nd Edition)". Wiley. February, 2011.
5. William H. Ford and William R. Topp. "Data Structures with C++ Using STL (2nd Edition)". Pearson. July, 2001.
6. Robert W. Sebesta. "Concepts of Programming Languages (11th Edition)". Pearson. February, 2015.
7. Norman Matloff and Peter Jay Salzman. "The Art of Debugging with GDB, DDD, and Eclipse". No Starch Press. September, 2008.
8. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, and Grady Booch. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley Professional. November, 1994.
9. Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, and Kelli A. Houston. "Object-Oriented Analysis and Design with Applications (3rd Edition)". Addison-Wesley Professional. April, 2007.

Instalaciones

Salón de clase con computador y proyector. Laboratorio de Ingeniería de Sistemas y Computación.

Material de este semestre

Curso en Moodle