Universidad Nacional de San Agustín
VICE RECTORADO ACADÉMICO
SILABO

CODIGO DEL CURSO: CS102O

1 Datos Generales

FACULTAD : Ingeniería de Producción y Servicios						
DEPARTAMENTO : Ingeniería de Sistemas e Informática			**ESCUELA :** Ciencia de la Computación			
PROFESOR : **TÍTULO :**						
ASIGNATURA : Objetos y Abstracción de Datos						
PREREQUISITO: CS101O	**CREDITOS:** 4		**Año**: 2010-1 **Sem:** 3^{er} Semestre.		**Total Horas:** 2 HT; 2 HT	2 HP 2 HL
Horario	Lun	Mar	Mie	Jue	Vie	Sáb
Total Semanal						
Aula						

2 Exposición de Motivos

Este es el tercer curso en la secuencia de los cursos introductorios a la informática. En este curso se
conceptos señalados por la *Computing Curricula* IEEE(c)-ACM 2001, bajo el enfoque *functional-f*
El paradigma orientado a objetos nos permite combatir la complejidad haciendo modelos a par
de los elementos del problema y utilizando técnicas como encapsulamiento, modularidad, polimor
dominio de estos temas permitirá que los participantes puedan dar soluciones computacionales a p
sencillos del mundo real.

2 Objetivo

- Introducir al alumno a los fundamentos del paradigma de orientación a objetos, permitiendo asimilar los conce
necesarios para desarrollar un sistema de información.

3 Contenido Temático

3 DS/Gráfos y Árboles.(7 horas)

Objetivos Específicos	Contenidos
▪ Ilustrar con ejemplos la terminología básica de teoría de grafos y algunas de las propiedades y casos especiales de cada una. ▪ Mostrar diferentes métodos de recorrido en árboles y grafos. ▪ Modelar problemas en Ciencias de la Computación usando grafos y árboles. ▪ Relacionar grafos y árboles con estructura de datos, algoritmos y conteo.	▪ Árboles. ▪ Grafos no di ▪ Grafos dirigi ▪ Árboles de e ▪ Estrategias d [1]

3 PF/Construcciones fundamentales.(5 horas)

Objetivos Específicos	Contenidos
▪ Analizar y explicar el comportamiento de programas simples involucrando las estructuras de programación fundamental cubiertas por esta unidad. ▪ Modificar y extender programas cortos que usan condicionales estándar, estructuras de control iterativas y funciones. ▪ Diseñar, implementar, probar y depurar un programa que use cada una de las siguientes estructuras fundamentales de programación: cálculos básicos, entrada y salida simple, estructuras estándar condicionales e iterativas y definición de funciones. ▪ Escoger la estructura apropiada condicional e iterativa para una estructura de programación dada. ▪ Aplicar técnicas de descomposición estructurada o funcional para dividir un programa en pequeñas partes. ▪ Describir los mecanismos de paso de parámetros.	▪ Sintaxis básica y semá lenguaje de más alto ni ▪ Variables, tipos, expresi naciones. ▪ Entrada y salida simple ▪ Estructuras de control c e iterativas. ▪ Funciones y paso de pa ▪ Descomposición estruct [2]

3 PF/Algoritmos y Resolución de Problemas.(5 horas)

Objetivos Específicos	Contenidos
▪ Discutir la importancia de los algoritmos en el proceso de solución de problemas. ▪ Identificar las propiedades necesarias de un buen algoritmo. ▪ Crear algoritmos para resolver problemas simples. ▪ Usar pseudocódigo o un lenguaje de programación para implementar, probar y depurar algoritmos para resolver problemas simples. ▪ Describir estrategias útiles para depuración.	▪ Estrategias pa blemas. ▪ El rol de los alg de solución de ▪ Estrategias de algoritmos. ▪ Estrategias de ▪ El Concepto y ritmos. [2]

3 PF/Programación Orientada a Eventos.(2 horas)

Objetivos Específicos	Contenidos
▪ Explicar la diferencia entre programación orientada a eventos y programación por línea de comandos. ▪ Diseñar, codificar, probar y depurar programas de manejo de eventos simples que respondan a eventos del usuario. ▪ Desarrollar código que responda a las condiciones de excepción lanzadas durante la ejecución.	▪ Métodos para la eventos. ▪ Propagación de ev ▪ Manejo de excepci [2]

3 AL/Análisis Básico de Algoritmos.(3 horas)

Objetivos Específicos	Contenidos
▪ Determinar la complejidad de tiempo y espacio de algoritmos simples.	▪ Análisis asintótico de lín casos promedio y superio ▪ Identificar la diferencia comportamiento entre el diano y peor caso. [2]

	Objetivos Específicos	Contenidos
3 AL/Algoritmos Fundamentales.(3 horas)	▪ Implementar los algoritmos cuadráticos más comunes y los algoritmos de ordenamiento $O(NlogN)$. ▪ Diseñar e implementar una función de (*hash*) apropiada para una aplicación. ▪ Diseñar e implementar un algoritmo de resolución de colisiones para tablas de *hash*. ▪ Discutir la eficiencia computacional de los principales algoritmos de ordenamiento, búsqueda y (*hashing*). ▪ Discutir otros factores, además de la eficiencia computacional, que influyen en la elección de los algoritmos, tales como tiempo de programación, mantenimiento y el uso de patrones específicos de aplicación en los datos de entrada. ▪ Resolver problemas usando los algoritmos de grafos fundamentales, incluyendo búsqueda por amplitud y profundidad; caminos más cortos con uno y múltiples orígenes, cerradura transitiva, ordenamiento topológico y al menos un algoritmo de árbol de expansión mínima. ▪ Demostrar las siguientes capacidades: evaluar algoritmos, seleccionar una opción de un rango posible, proveer una justificación para tal elección e implementar el algoritmo..	▪ Algoritmos numéricos simpl ▪ Búsqueda secuencial y bina ▪ Algoritmos cuadráticos de miento (selección, inserción ▪ Algoritmos de tipo $O($ (Quicksort, heapsort, merge ▪ Tablas de (*hash*) incluyendo gias de solución para las co ▪ Árboles de búsqueda binari ▪ Representación de grafos (Matrices de adyacencia). ▪ Recorridos por amplitud y didad. ▪ El algoritmo del camino m (algoritmos de Dijkstra y F ▪ Cerradura transitiva (algor Floyd). ▪ Árbol de expansión mínima mos de Kruskal y Prim). ▪ Ordenamiento Topológico. [2]

3 PL/Declaración y Tipos.(2 horas)

Objetivos Específicos	Contenidos
▪ Explicar el valor de los modelos de declaración, especialmente con respecto a la programación en mayor escala. ▪ Identificar y describir las propiedades de una variable, tales como su: dirección asociada, valor, ámbito, persistencia y tamaño. ▪ Discutir la incompatibilidad de tipos. ▪ Demostrar las diferentes formas de enlace, visibilidad, ámbito y manejo del tiempo de vida. ▪ Defender la importancia de los tipos y el chequeo de tipos para brindar abstracción y seguridad. ▪ Evaluar las ventajas y desventajas en el manejo del tiempo de vida (conteo por referencia vs. recolección de basura).	▪ La concepción de tipos como un conjunto de valores unidos a un conjunto de operaciones. ▪ Declaración de modelos (enlace, visibilidad, alcance y tiempo de vida). ▪ Vista general del chequeo de tipos. ▪ Recolección de basura. [2]

3 PL/Mecanismos de Abstracción.(5 horas)

Objetivos Específicos	Contenidos
	▪ Procedimientos, funciones e res como mecanismos de ción. ▪ Mecanismos de parametriza ferencia vs. valor). ▪ Registros de activación y tración de almacenamiento ▪ Tipos de parámetros y tip metrizados. ▪ Módulos en lenguajes de p ción. [2]

3 PL/Programación Orientada a Objetos.(7 horas)

Objetivos Específicos	Contenidos
▪ Justificar la filosofía de diseño orientado a objetos y los conceptos de encapsulación, abstracción, herencia y polimorfismo. ▪ Diseñar, implementar, probar y depurar programas simples en un lenguaje de programación orientado a objetos. ▪ Describir como los mecanismos de clases soportan encapsulación y ocultamiento de la información. ▪ Diseñar, implementar y probar la implementacion de la relación es-un *IsKindOf* entre objetos usando jerarquía de clases y herencia. ▪ Comparar y contrastar las nociones de sobrecarga y sobreescritura de métodos en un lenguaje de programación. ▪ Explicar la relación entre la estructura estática de una clase y la estructura dinámica de las instancias de dicha clases. ▪ Describir como los iteradores acceden a los elementos de un contenedor.	▪ Diseño orientado a ▪ Encapsulación y oc información. ▪ Separación de co implementación. ▪ Clases y subclases. ▪ Herencia (sobreesc dinámico). ▪ Polimorfismo (poli tipo vs. herencia). ▪ Jerarquías de clase ▪ Clases de tipo colec de iteración. ▪ Representaciones i tos y tablas de mét [2]

3 SE/Diseño de Software.(5 horas)

Objetivos Específicos	Contenidos
▪ Discutir las propiedades del buen diseño de softwarem incluyendo la naturaleza y el rol de la documentación asociada. ▪ Conducir una revisión de diseño de software con material de código abierto utilizando lineamientos apropiados.	▪ Conceptos y principios fundamentales de diseño. ▪ El rol y uso de contratos. ▪ Patrones de diseño. [2]

3 SE/Usando APIs.(1 horas)

Objetivos Específicos	Contenidos	Hora
▪ Explicar el valor de las interfaces para programación de aplicaciones (APIs) en el desarrollo de software. ▪ Usar navegadores de clases y herramientas relacionadas durante el desarrollo de aplicaciones usando APIs. ▪ Diseñar, implementar, probar y depurar programas que usan paquetes API de larga escala.	▪ Programación usando API. ▪ Diseño de API. ▪ Navegadores de clases (*Class browsers*) y herramientas relacionadas. ▪ Depuración en el entorno API. ▪ Introducción a la computación basada en componentes. [2]	

3 SE/Especificación de Requerimientos.(1 horas)

Objetivos Específicos	Contenidos
▪ Discutir los retos de mantener software heredado. ▪ Usar un método común, no formal para modelar y especificar (en la forma de un documento de especificación de requerimientos) los requerimientos para un sistema de software de tamaño medio. ▪ Traducir en lenguaje natural una especificación de requerimientos de software escrita en un lenguaje de especificación formal comunmente usado.	▪ Técnicas de modelar lisis de requerimiento ▪ Prototipeo. ▪ Conceptos básicos de pecificación formal. [2]

4 Actividades

- Asignaciones
- Controles de Lectura
- Exposiciones

5 Recursos Materiales

- Apuntes del curso
- Libro(s) de la bibliografía

6 Metodología

- Clase Magistral.
- Taller didáctico.
- Social Constructivismo.
- Prácticas personales y en grupo.

7 Evaluación

La nota final (NF) se obtiene de la siguiente manera:

NE Nota de Exámenes 60 %, esta nota se divide en
- Exámen Parcial 40 %

- Examen Final 60 %

NT Nota de Trabajos e Intervención en clase 40 %

$$NF = 0{,}6 * NE + 0{,}4 * NT$$

Referencias

[1] Sergei Nakariakov. *The Boost C++ Libraries: Generic Programming.* CreateSpace Independent Publishing Platform, April 2013.

[2] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley, 4th edition, 2013.

Docente del curso