Universidad Nacional de San Agustín
VICE RECTORADO ACADÉMICO
SILABO

CODIGO DEL CURSO: CS101O

1 Datos Generales

FACULTAD : Ingeniería de Producción y Servicios						
DEPARTAMENTO : Ingeniería de Sistemas e Informática			**ESCUELA :** Ciencia de la Computación			
PROFESOR : **TÍTULO :**						
ASIGNATURA : Introducción a la Programación Orientada a Objetos						
PREREQUISITO: CS101F	**CREDITOS:** 5		**Año**: 2010-1 **Sem:** 2^{do} Semestre.		**Total Horas:** 2 HT; 2 HP 4 HL	
Horario	Lun	Mar	Mie	Jue	Vie	Sáb
Total Semanal						
Aula						

2 Exposición de Motivos

Este es el segundo curso en la secuencia de los cursos introductorios a la informática.
El curso servirá como puente entre el paradigma de la imperativo y el orientado al objeto, a
los participantes en los diversos temas del área de computación como: algoritmos, estructuras de
software, etc.

2 Objetivo

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

3 Contenido Temático

3 PL/Visión General de los Lenguajes de Programación.(1 horas)

Objetivos Específicos

- Listar la evolución
de programación ide
es que su historia n
a los paradigmas ac
- Identificar al meno
rística distintiva pa
los paradigmas de p
biertos en esta unid
- Evaluar las ventaja
entre los diferentes p
siderando temas tal
cia de espacio, eficie
po (para ambas part
y programador), seg
der de las expresion

3 PL/Máquinas Virtuales.(2 horas)

Objetivos Específicos	Contenidos
▪ Describir la importancia y poder de la abstracción en el contexto de máquinas virtuales. ▪ Explicar los beneficios de los lenguajes intermedios en el proceso de compilación. ▪ Evaluar las ventajas y desventajas entre desempeño vs. portabilidad.	▪ El concepto de máquina virtual. ▪ Jerarquías de las máquinas virtuales. ▪ Lenguajes intermedios. [3], [1]

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.	▪ 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. [3], [1]

	Objetivos Específicos	Contenidos
3 PF/Construcciones fundamentales.(6 horas)	▪ 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 par ▪ Descomposición estruct [3], [1]

3 PL/Programación Orientada a Objetos.(10 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 ▪ Encapsulación y información. ▪ Separación de implementación. ▪ Clases y subclase ▪ Herencia (sobree dinámico). ▪ Polimorfismo (po tipo vs. herencia) ▪ Jerarquías de cla ▪ Clases de tipo col de iteración. ▪ Representaciones tos y tablas de m [2], [3], [1]

3 PF/Algoritmos y Resolución de Problemas.(3 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. [3], [1]

3 PF/Recursividad.(3 horas)

Objetivos Específicos	Contenidos	Hora
▪ Describir el concepto de recursividad y dar ejemplos de su uso. ▪ Identificar el caso base y el caso general de un problema definido recursivamente. ▪ Comparar soluciones iterativas y recursivas para problemas elementales tal como factorial. ▪ Describir la técnica dividir y conquistar. ▪ Implementar, probar y depurar funciones y procedimientos recursivos simples. ▪ Describir como la recursividad puede ser implementada usando una pila. ▪ Discutir problemas para los cuales el *backtracking* es una solución apropiada. ▪ Determinar cuando una solución recursiva es apropiada para un problema.	▪ El concepto de recursividad. ▪ Funciones matemáticas recursivas. ▪ Funciones recursivas simples. ▪ Estrategias de dividir y conquistar. ▪ *Backtracking* recursivo. [3], [1]	

3 AL/Análisis Básico de Algoritmos.(2 horas)

Objetivos Específicos	Contenidos
▪ Determinar la complejidad de tiempo y espacio de algoritmos simples.	▪ Identificar la diferencia comportamiento entre el diano y peor caso. [3], [1]

	Objetivos Específicos	Contenidos
3 AL/Algoritmos Fundamentales.(6 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. [3], [1]

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] Harvey M. Deitel. *C++ How to Program*. Pearson Educacion, 9th edition, 2013.

[2] Jo Ann Smith. *Desarrollo de Proyectos con Programación Orientada a Objetos con C++*. Thomson Learning, 2001.

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

Docente del curso