Facultad de Ingeniería - Universidad Nacional de Cuyo P1- PROGRAMA DE ASIGNATURA			
Asignatura:	**Programación Orientada a Objetos**		
Profesor Titular:	**César Omar Aranda**		
Carrera:	**Ingeniería en Mecatrónica**		
Año: 2016	**Semestre: 2**	**Horas Semestre: 60**	**Horas Semana: 4**

OBJETIVOS

Objetivos conceptuales:

- Reconocer los conceptos fundamentales del paradigma de programación orientado a objetos.
- Identificar las entidades que participan en un sistema y sus comportamientos asociados.
- Comprender básicamente el proceso de desarrollo de software orientado a objetos.

Objetivos procedimentales:

- Definir las características de la solución de un problema bajo el paradigma de la orientación a objetos.
- Codificar en lenguaje C++ a partir de interpretar una representación UML.
- Construir soluciones mediante POO aplicando buenas prácticas básicas de Ingeniería de Software.

Objetivos actitudinales:

- Adquirir confianza en las propias posibilidades de comprender y resolver problemas.
- Manifestar perseverancia en las tareas a desarrollar.
- Ampliar la capacidad para trabajar de manera autónoma y grupal.
- Mejorar las habilidades de investigación así como su visión crítica y autocrítica de problemas y soluciones.

CONTENIDOS

UNIDAD 1: ORIENTACION A OBJETOS

1.A. Aspectos Generales

Revisión de Paradigmas de programación. Conceptos generales. Lenguajes de programación. Ejemplos.
Análisis de inconvenientes de los enfoques solamente procedurales en la resolución de problemas complejos, en la reutilización de código y en el mantenimiento.
Programación Orientada a Objetos vs Basada en Objetos. Ventajas y desventajas.
Panorama de lenguajes orientados a objetos. Actualidad y tendencias
IDE, CASE y otras herramientas de desarrollo. Construcción de Aplicaciones OO.

1.B. Paradigma Orientado a Objetos

Conceptos fundamentales de la Orientación a Objetos. Objeto y Clase. Atributos. Operaciones.
Principios del paradigma orientado a objetos. Abstracción y Proceso de abstracción. Encapsulamiento. Jerarquías de Clases. Polimorfismo. Modularidad.
Mecanismos de la orientación a objetos. Herencia. Tipos de Herencia. Instanciación. Ocultamiento. Comunicación entre objetos: Mensajes.
Ejemplos y ejercitación.

UNIDAD 2: Proceso y Representación Orientada a Objetos

2.A. Proceso de modelado simplificado

Proceso de Modelado Orientado a Objetos. Conceptos. Fases. Principios de diseño. Documentación de sistema. Relación con el Manual del Usuario.
Generalidades. Aplicabilidad y ejemplos.

2.B. UML

El lenguaje de Modelado Unificado UML. Generalidades. Aplicabilidad.
Elementos. Versiones. Diagramas Estructurales y de Comportamiento.

Diagrama de Casos de Uso. Actor. Caso de Uso. Relaciones. Ejemplos.
Diagrama de clases. Relaciones. Asociación. Agregación. Composición. Ejemplos y ejercitación.
Diagrama de Secuencia. Diagrama de Actividad. Diagrama de Estados. Ejemplos y ejercitación.

UNIDAD 3: Lenguaje de Programación

3.A. Fundamentos del Lenguaje C++

El lenguaje C++. Generalidades. Aplicabilidad. Versiones.
Sintaxis y semántica básica.
Tipos de datos. Operadores. Expresiones. Casting. Punteros.
Declaración de variables. Constantes. Ejemplos. Ejercitación.

3.B. Instrucciones Básicas en C++

Flujos estándares de entrada/salida. Tipos.
Estructuras de Control. Sentencias if, switch, for, while, do, break, exit, [goto].
Ejemplos. Ejercitación.

3.C. Programación de Objetos en C++

Tipos de datos compuestos.
Clases. Atributos y Métodos. Definición. Instanciación.
Métodos constructores y destructores.
Ejemplos y ejercitación.

3.D. Mecanismos Orientados a Objetos en C++

Clase Abstracta. Herencia Simple y múltiple. Visibilidad.
Polimorfismo y reutilización. Sobrecarga. Sobreescritura.
Ejemplos y ejercitación.

UNIDAD 4: Elementos Complementarios

4.A. Tipos de Datos Especiales en C++

Estructuras de datos como Objetos. Tipos Abstractos de Datos.
Objetos contenedores. Concepto. Tipos.
Colecciones de Objetos. Array. Vector. List. Stack. Queue.
Plantilla. Concepto. Utilidad.
Ejemplos. Ejercitación.

4.B. Depuración del software

Gestión de errores y excepciones.
Técnicas de depuración. Breaking. Debugging. Stepping. Tracing.
Ejemplos. Ejercitación.
Depuración vs. Prueba de Software.

4.C. Persistencia en C++

Persistencia: concepto y mecanismos.
Streams asociados. File. Acceso a dispositivos.
Ejemplos.

4.D. Extensiones

Librerías complementarias. Incorporación al proyecto y utilización.
Características básicas e introducción al uso de: gráficos, sonidos, bases de datos.
Gestión de dispositivos de entrada/salida, conectividad en red.
Ejemplos.

METODOLOGÍA DE ENSEÑANZA

Se considera que cada clase es eminentemente teórico-práctica, aplicando en forma inmediata los conceptos expresados teóricamente, por lo que el trabajo, en general, responderá al de un aula-taller.

Las clases teóricas se presentan sobre pizarra o con elementos multimediales, incluyendo ejemplos de

aplicación de distinto nivel de completitud y complejidad.

Se plantean trabajos prácticos, casos de estudio, investigación y/o de extensión.

Algunos de ellos tienen carácter obligatorio (se solicita y califica tanto su desarrollo como su presentación) y otros opcionales (son sólo a efectos de ejercitación de los alumnos, de fijación de conceptos o discusión). Según las características del alumnado y del ciclo lectivo, los trabajos se distribuyen para ser llevados adelante de manera individual y/o grupal.

El seguimiento de alumnos se realiza registrando asistencia, cumplimiento y participación.

En principio el desarrollo de cada clase cuenta con tres momentos sucesivos:

- La introducción: Donde se expone teóricamente un tema nuevo o se realiza el análisis del código de un ejemplo simple ya resuelto.
- La elaboración: Donde se construye la solución a un problema acorde al tema explicado anteriormente y que permite se pongan de manifiesto las principales dudas conceptuales y prácticas.
- El cierre: Donde se analizan críticamente diferentes soluciones obtenidas y/o se incluyen conclusiones pertinentes al tema.

Muchos de los conceptos de UML, uso del IDE y principios de diseño y programación, son desarrollados durante el cursado insertos entre otros contenidos, de manera no secuencial (según el programa de contenidos). En las unidad 1 y 2 se plantean conceptos generales, que son trabajados de manera más profunda durante el desarrollo de las unidades 3 y 4.

Uso de mecanismos compartidos sobre redes y comunicaciones para la organización/distribución de los recursos de estudio así como para la realización de diversas actividades docentes complementarias a las propuestas de manera presencial.

En caso de que quedase algún tema sin dictar, se provee la bibliografía adecuada.

Actividad	Carga horaria por semestre
Teoría y resolución de ejercicios simples	30
Formación práctica	
Formación Experimental – Laboratorio	10
Formación Experimental - Trabajo de campo	0
Resolución de problemas de ingeniería	10
Proyecto y diseño	10
Total	**60**

BIBLIOGRAFÍA

Bibliografía básica

Autor	Título	Editorial	Año	Ejemplares en biblioteca
DEITEL H. M. Y DEITEL P. J.	Cómo Programar en C/C++ 6ª ed.	Prentice-Hall	2009	0
STROUSTRUP, B.	Programming. Principles and Practice Using C++	Pearson Education	2009	0
BRONSON, Gary	C++ para Ingeniería y Ciencias, 2º ed.	Cengage Learning	2007	0
CEBALLOS SIERRA, F.J.	C/C++ Curso de Programación, 3ª	Ra-Ma	2007	0

Bibliografía complementaria

Autor	Título	Editorial	Año	Ejemplares en biblioteca
F.XHAFA, P.PAU VAZQUEZ, A.JORDI y otros	Programación en C++ para ingenieros	Thomson-Paraninfo	2006	0
ECKEL, Bruce	Thinking In C++, Volumen 2	Mindview, Inc	2004	0
ELLIS, M.A. y STROUSTRUP, B.	C++ Manual De Referencia Con Anotaciones	Diaz De Santos	2004	0
CEBALLOS SIERRA, F.J.	Enciclopedia del lenguaje C++	AlfaOmega	2004	2

ACERA GARCIA, M.A.	C/C++ (edición revisada y actualizada 2012)	Anaya Multimedia	2012	0
ECKEL, Bruce	Aplique C++	McGraw-Hill	1991	2

EVALUACIONES

Durante el período de clases, se prevé un sistema de evaluación continua consistente en un registro de los trabajos individuales y grupales realizados en clase.

Esto no implica la entrega de todos los trabajos al profesor por parte del alumno para su revisión. En el caso de la elaboración de programas propuestos, es el alumno quien debe lograr la habilidad de obtener software funcional y verificar la correctitud de su solución a partir del producto final obtenido. Mostrando finalmente sus resultados o conclusiones.

En base a las características del grupo y tiempos disponibles se agregará la elaboración de trabajos monográficos para su entrega y/o exposición.

Se prevé realizar 1 (una) evaluación de carácter global (esto es, abordando contenidos conceptuales y prácticos que corresponden a un período indicado en clase oportunamente), con una única instancia de recuperación, también global y acumulativa en contenidos.

Se proponen además una serie de actividades prácticas de desarrollo obligatorio para seguimiento y control. Entre ellas un Trabajo Práctico Especial de desarrollo y presentación individual o grupal según la población del curso.

Este Trabajo Especial, con características integradoras, tiene el peso de un parcial. Sin embargo, dadas la modalidad de su desarrollo no es recuperable.

Todas las evaluaciones deben dejar una constancia documental, es decir almacenadas en un archivo o escritas en papel. Esta evaluación (o una copia) es devuelta al alumno con las correcciones pertinentes.

El criterio de evaluación a seguir, es adelantado (informado) en clase.

Para obtener la regularidad o la promoción directa en la asignatura se tienen en cuenta las calificaciones obtenidas en la evaluación parcial, en la aprobación del Trabajo Práctico Especial y en el cumplimiento de los trabajos prácticos de seguimiento obligatorios.

Para aprobar la asignatura durante el cursado, esto es alcanzar la Promoción, se deben satisfacer las mismas condiciones de la regularidad y haber obtenido un mínimo del 70% tanto en la evaluación parcial como en el Trabajo Práctico Especial.

Para la aprobación de la asignatura, en condición de Alumno Regular, se propone la presentación y defensa individual de un producto software elaborado individualmente (puede ser la continuación del Trabajo Especial realizado durante el cursado).

Éste producto consiste en la implementación de una aplicación que debe ejecutar libre de fallas (jamás terminar anormalmente), aplicando la totalidad de los conceptos fundamentales aprendidos en clase y en la confección de un informe técnico de la aplicación que incluye elementos conceptuales, elementos de diseño y un manual de usuario.

El “objetivo” del producto software y/o los temas abordados pueden ser propuestos por el alumno y acordados con el profesor con una prudente anticipación a la fecha elegida para el examen final.

La defensa consiste en una exposición coloquial sobre los conceptos fundamentales y estrategias utilizadas en la confección del producto. La finalidad de esta exposición no es otra que la de corroborar autoría.

Para la aprobación de este espacio curricular en condición de Alumno Libre, se debe elaborar, presentar y aprobar un Trabajo Especial similar al realizado durante el cursado, con anterioridad a la fecha elegida para el examen final. Este trabajo cumple con la instancia de evaluación de la práctica al momento del examen.

Dado que la asignatura está orientada a desarrollar habilidades de programación bajo un paradigma estricto se propone, adicionalmente, la resolución individual de un producto software acotado, asignado por el profesor en el momento del examen.

Esta resolución debe hacerse en un tiempo acotado máximo de 1 hora y media. La misma incluye además de las clases propias del lenguaje, clases de usuario y precisa aplicar algunos de los conceptos fundamentales de este programa de estudio (como mecanismos de herencia y polimorfismo, almacenamiento de información usando clases de colección de C++, persistencia, sobrecarga de operadores, plantillas, u otros) según la consigna.

El problema se diseña e implementa en el momento del examen y no requiere de un coloquio de defensa.

Más allá que el programa deba compilar y ejecutar sin errores, es fundamental para su aprobación demostrar que se han aplicado de manera correcta los conceptos y mecanismos tanto de la OO como de C++. Una vez comprobado esto último, la calificación se asigna analizando la implementación realizada.

Con respecto a otros detalles relacionados con adquirir la regularidad y/o la aprobación de la asignatura se siguen los lineamientos generales fijados para la carrera, tanto de tipo académico como administrativos.

Programa de examen

No Aplicable