



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: 2013	Semestre: 2	Horas Semestre: 60	Horas Semana: 4

OBJETIVOS

- ◆ **Objetivos conceptuales:**
 - Reconocer los conceptos fundamentales del paradigma de programación orientado a objetos.
 - Discernir los tipos de aplicación y las situaciones en las que es posible y necesario aplicar el paradigma orientado a objetos.
 - Comprender globalmente 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.
 - Interpretar una representación de sistema mediante UML para proceder a la codificación en lenguaje C++ a partir de aquella.
 - Realizar POO aplicando buenas prácticas 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: PROGRAMACIÓN ORIENTADA A OBJETOS

1.A. Aspectos Generales

Paradigmas de programación. Conceptos generales. Lenguajes de programación. Ejemplos.

Análisis de inconvenientes de los enfoques solamente procedurales en la resolución de problemas complejos, la reutilización de código y el mantenimiento.

Programación Orientada a 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: Modelos Orientados a Objetos

2.A. UML

El lenguaje de Modelado Unificado UML. Generalidades. Aplicabilidad.

Elementos. Versiones.

Relaciones con el Manual del Usuario y/o del Sistema.



2.B. Diagramas estructurales

Diagrama estructural. Concepto. Diagrama de clases. Relaciones. Asociación. Agregación. Composición. Ejemplos. Ejercitación.
Otros diagramas (de paquetes, de componentes, de despliegue): generalidades y ejemplos.

2.C. Diagramas de comportamiento

Diagrama de comportamiento. Concepto. Tipos. Utilidad.
Diagrama de Secuencia. Diagrama de Actividad. Ejemplos. Ejercitación.
Otros diagramas (de Colaboración, de Estados, de Tiempo): generalidades y ejemplos.
Diagrama de Casos de Uso. Actor. Caso de Uso. Relaciones. Ejemplos. Descripción de un caso de uso.

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. Set. Map.
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.

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 unidades 1 y 2 se plantean conceptos generales, que son trabajados de manera más profunda durante el desarrollo de las unidades 3 y 4.

En caso de disponibilidad de una plataforma de Educación a Distancia se hace uso de la misma 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 edición)	Prentice-Hall	2009	0
Stroustrup, B.	Programming. Principles and Practice Using C++	Pearson Education	2009	0
Ceballos Sierra, F.J.	C/C++ Curso de Programación, 3ª edición	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	2004	0
Ellis, M.A. y Stroustrup, B.	C++ Manual De Referencia Con Anotaciones	Ed. 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
Schildt, H.	C/C++ Manual De Referencia	Anaya Multimedia	1996	0

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 los contenidos conceptuales y prácticos que corresponden a un período indicado en clase oportunamente), con una única instancia de recuperación, también global y acumulativa.

Se proponen además una serie de actividades de desarrollo obligatorio para seguimiento y control. Entre ellas un Trabajo Práctico Especial de desarrollo individual.

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 tanto en las evaluaciones indicadas antes como en las obtenidas del seguimiento continuo.

Para la aprobación de la asignatura (obtenida mediante promoción o examen final si no adquiriera la promoción) se propone la presentación y defensa individual de un producto software elaborado individual o grupalmente.

Este producto incluye una implementación del alumno aplicando la totalidad de los conceptos fundamentales aprendidos en clase en una aplicación que debe ejecutar libre de fallas (jamás terminar anormalmente) y un manual de usuario conteniendo anexa la documentación de la aplicación.

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.

El "tema/objetivo" del producto software es propuesto durante el periodo de dictado de clases y controlado en distintas instancias por docente donde el alumno muestra su nivel de avance.

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

Bolilla 1:	Temas: 1A – 2B – 3D – 4C
Bolilla 2:	Temas: 1B – 2C – 3C – 4B
Bolilla 3:	Temas: 1A – 2A – 3B – 4A
Bolilla 4:	Temas: 1B – 2A – 3A – 4A
Bolilla 5:	Temas: 1B – 2B – 3D – 4B
Bolilla 6:	Temas: 1A – 2C – 3C – 4C
Bolilla 7:	Temas: 1B – 2A – 3B – 4A
Bolilla 8:	Temas: 1A – 2B – 3A – 4C
Bolilla 9:	Temas: 1B – 2C – 3D – 4B

FECHA, FIRMA Y ACLARACIÓN TITULAR DE CÁTEDRA