Universidad
Católica del Norte

Programa de Asignatura

I. IDENTIFICACIÓN				
Carrera o programa: Ingeniería Civil en Computación e Informática				
Unidad responsable: Escuela de Ingeniería				
Nombre de la asignatura: Lenguajes de Programación				
Código: ECIN-00609				
Semestre en la malla[1]**:** 7				
Créditos SCT - Chile: 5				
Ciclo de Formación	Básico		Profesional	X
Tipo de Asignatura	Obligatoria	X	Electiva	
Clasificación de área de conocimiento[2]				
Área: Ingeniería y Tecnología		**Sub área:** Ingeniería Informática		
Requisitos:				
Pre-requisitos:		**Requisito para:**		
• ECIN-00407 Estructuras de Datos				

II. ORGANIZACIÓN SEMESTRAL							
Horas Dedicación Semanal (Cronológicas)	Docencia Directa	4.5	Trabajo Autónomo	3	Total	7.5	
Detalle Horas	Cátedra	Ayudantía	Laboratorio	Taller	Terreno	Exp. Clínica	Supervision
Directas	3			1.5			

[1] Este campo
[2] Clasificación del curso de acuerdo a la OCDE

III. APORTE AL PERFIL DE EGRESO

La asignatura contribuye al dominio 1 del perfil de egreso, “Conocimiento científico y disciplinario”. Además, contribuye al dominio 2 “Habilidades y Actitudes Personales y Profesionales”. También contribuye al dominio 3 “Habilidades Interpersonales”. También contribuye al dominio 4 “Habilidades para la Práctica de la Ingeniería”. Al finalizar la asignatura las y los estudiantes serán capaces de comprender y aplicar los principales elementos involucrados en el diseño, selección y utilización de los lenguajes de programación.

IV. HABILIDADES PERFIL DE EGRESO (RELACIÓN)

1.3 Aplicar conocimientos, métodos y herramientas de la especialidad para resolver problemas complejos de Ingeniería de Software, Plataformas y Gestión de Tecnologías.

2.1 Identificación, formulación, modelación y resolución de problemas complejos de ingeniería considerando las interacciones y la dinámica de las variables.

3.2 Comunicar comprensivamente información técnica en español, en forma oral, escrita, y gráfica, a nivel avanzado

4.5 Implementar las soluciones TIC. Estas soluciones consideran las arquitecturas TI junto a sus modelos de servicios y modelos operativos; los sistemas de software; y las plataformas de cómputo y comunicaciones junto a sus servicios asociados.

V. RESULTADOS DE APRENDIZAJE

1. Explicar la evolución y genealogía de los principales lenguajes de programación.
2. Emplear los diversos criterios técnicos de evaluación aplicables a un lenguaje de programación en un lenguaje de uso masivo actual.
3. Explicar los principales elementos que conforman un lenguaje de programación y cómo ellos se implementan en diversos lenguajes de amplio uso actual.
4. Crear aplicaciones simples haciendo uso de los elementos de programación vistos en clases de un lenguaje de alto nivel.
5. Aplicar los fundamentos de distintos paradigmas de programación al crear pequeñas aplicaciones.
6. Construir un traductor para un nuevo lenguaje de programación, aplicando la teoría, técnicas y herramientas actualmente disponibles.
7. Analizar las relaciones causa efecto de los procesos en estudio.
8. Preparar presentaciones orales y el apoyo audiovisual con un lenguaje apropiado, estilo, tiempo y fluidez.
9. Desarrollar la solución tecnológica más adecuada en base a las características del problema y los recursos disponibles.

VI. ÁREAS TEMÁTICAS

1. Conceptos preliminares de lenguajes de programación (2 semana)
   - 1.1 Razones para estudiar conceptos de lenguajes de programación: Criterios de evaluación, Categorías de lenguajes
   - 1.2 Evolución de los principales lenguajes de programación
   - 1.3 Métodos para describir Sintaxis y Semántica: BNF, EBNF, Grafos de Sintaxis, Gramáticas Atribuidas, Semántica Denotacional
2. Elementos de un lenguaje de programación (2 semana)
   - 2.1 Nombres, Ligamiento, Chequeo de Tipo y Alcance
   - 2.2 Tipos de Datos: Tipos primitivos, Tipos estructurados
   - 2.3 Expresiones y Sentencias de Asignación
   - 2.4 Expresiones Aritméticas, Relacionales y Booleanas
   - 2.5 Sobrecarga de Operadores, Conversiones de tipos

2.6 Evaluación corto-circuito
2.7 Sentencias de asignación
2.8 Estructuras de Control
2.9 Sentencias de selección, de iteración, saltos incondicionales
2.10 Subprogramas
2.11 Aspectos de diseño de subprogramas
2.12 Métodos de traspaso de parámetros
2.13 Subprogramas sobrecargados, genéricos
2.14 Tipos de Datos Abstractos y Constructos para Encapsulación

3. Manejo de Excepciones y Eventos (2 semana)
   3.1 Excepciones
   3.2 Excepciones y seguridad
   3.3 Manejo de Excepciones en C++, Java, C#, Python
   3.4 Concepto de Evento
   3.5 Manejo de Eventos en Java y C#
4. Paradigmas de Programación (4 semana)
   4.1 Programación Estructurada
   4.2 Programación Orientada a Objetos
   4.3 Programación Funcional
   4.4 Programación en Lógica
5. Introducción a los compiladores (4 semana)
   5.1 Lenguajes compilados y lenguajes interpretados
   5.2 Estructura de un compilador
   5.3 Generación de analizadores léxicos
   5.4 Generación de analizadores sintácticos
   5.5 Definición de tablas de símbolos
   5.6 Mecanismos de paso de parámetros
   5.7 Generación de código intermedio

VII. ORIENTACIONES METODOLÓGICAS

1. La metodología a desarrollar en esta asignatura debe favorecer la interacción entre las y los estudiantes a través de trabajos prácticos colaborativos que permitan la solución a problemas específicos contextualizados a la asignatura.
    - Se sugiere el uso de clases expositivas y participativas con método combinado, es decir, clases expositivas con alternancia de trabajos en grupo de corta duración para responder preguntas.
    - Se sugiere la utilización de la metodología activa de análisis de casos para desarrollar experiencias que permitan incorporar los elementos teórico-prácticos asociados a los resultados de aprendizaje de la asignatura.
2. Las experiencias de cátedra/laboratorio/taller deben ser realizadas por medio de la utilización de software moderno aplicable a la asignatura.
3. Se recomienda que las y los estudiantes realicen presentaciones periódicas sobre el trabajo realizado que incluya: contextualización, desarrollo y conclusiones.
4. Actividades prácticas recomendadas: cápsulas teóricas, reuniones de trabajo, taller de trabajo en equipo y liderazgo, presentaciones e informes escritos de avance en español, revisión del estado del arte asociado al problema, lluvia de ideas, análisis de alternativas y descripción detallada de la solución.

VII. ORIENTACIONES Y CRITERIOS PARA LA EVALUACIÓN

1. Se recomienda la aplicación de una evaluación diagnóstica al inicio de la asignatura.
2. La asignatura podría contemplar dos instancias de evaluación de los resultados de aprendizaje: cátedra y taller/laboratorio.
    - En el caso de existir, ambas debieran ser aprobadas por separado: el porcentaje de cada una de ellas deberá ser de 60% para cátedra y 40% para taller/laboratorio.
    - En el caso que la asignatura tenga actividades de taller/laboratorio, éstas deben ser realizadas en grupos de estudiantes y se recomienda la elaboración por parte de los estudiantes de un informe sobre la actividad desarrollada.
3. Se evaluará el conocimiento conceptual y procedimental mediante el desarrollo de al menos dos pruebas sumativas de carácter presencial.
    - Se recomienda además la aplicación de una evaluación mediante la entrega de un trabajo desarrollado en las horas indirectas asociadas a la asignatura.
    - Se recomienda que las y los estudiantes realicen una o más presentaciones de los trabajos realizados, la evaluación de la misma debe ser por medio de la aplicación de una rúbrica.
4. Se recomienda realizar evaluaciones de carácter formativo. Esto permite al docente introducir correcciones, añadir alternativas y reforzar los aspectos para ayudar al estudiantado en el logro de sus habilidades.
5. La asistencia y condiciones de aprobación de la asignatura debe ser acorde a la aplicación del Reglamento de Docencia de Pregrado.

IX. RECURSOS BIBLIOGRÁFICOS

Bibliografía Mínima

– Sebesta, Robert ”Concepts of Programming languages” Novena Edición, Addison Wesley, 2009

Bibliografía Complementaria

– Louden, K. ”Lenguajes de programación. Principios y práctica”, segunda edición. Editorial Thomson, 2004
– Louden, K. “Construcción de Compiladores. Principios y Práctica”, Cengage Learning Editores S.A. de C.V., 2004.