Universidad Católica del Norte

PROGRAMA DE ASIGNATURA

I. IDENTIFICACIÓN

Carrera o programa: Ingeniería en Tecnologías de Información				
Unidad responsable: Escuela de Ingeniería				
Nombre de la asignatura: Programación				
Código: ECIN-00261				
Semestre en la malla[1]: 2				
Créditos SCT – Chile: 6				
Ciclo de Formación	Básico	X	Profesional	
Tipo de Asignatura	Obligatoria	X	Electiva	
Clasificación de área de conocimiento[2]				
Área: Ingeniería y Tecnología		**Subárea:** Ingeniería Eléctrica, Electrónica e Informática		
Requisitos:				
Prerrequisitos:		**Requisitos para:**		
• Álgebra I • Proyecto Introducción a la Ingeniería		• Programación Orientada a Objetos • Proyecto Programación Avanzada		

II. ORGANIZACIÓN SEMESTRAL

Horas Dedicación Semanal (Cronológicas)	Docencia Directa	4.5	Trabajo Autónomo	5.5	Total	10	
Detalle Horas Directas	Cátedra	Ayudantía	Laboratorio	Taller	Terreno	Exp. Clínica	Supervisión
	1.5	1.5	1.5				

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 4 “Habilidades para la Práctica de la Ingeniería”. Al finalizar la asignatura las y los estudiantes serán capaces de utilizar el computador como una herramienta de apoyo, a través del uso y manejo de un lenguaje de programación que permita resolver problemas que se encuentran en el mundo de la ingeniería.

IV. COMPETENCIAS

La carrera declara las siguientes habilidades:
1.2. Aplicar conocimientos de ciencias de la ingeniería a la solución de problemas complejos de ingeniería.
2.1. Identificar y resolver problemas con un razonamiento analítico.
2.2. Experimentar, investigar y descubrir conocimiento.
2.3. Organizar e integrar componentes de la realidad mediante un pensamiento sistémico.

[1] Este campo sólo se completa en caso de carreras con programas semestrales.

[2] Clasificación del curso de acuerdo a la OCDE

4.3. Concebir soluciones que involucren, por ejemplo, aplicaciones TI, infraestructura TI, toma de decisiones, gestión de datos y gestión de proyectos.
4.4. Diseñar soluciones que involucren, por ejemplo, aplicaciones TI, infraestructura TI, toma de decisiones, gestión de datos y gestión de proyectos.
4.5. Implementar soluciones que involucren, por ejemplo, aplicaciones TI, infraestructura TI, toma de decisiones, gestión de datos y gestión de proyectos.
4.7. Participar en iniciativas de innovación de nuevos productos, procesos o servicios.

V. RESULTADOS DE APRENDIZAJE

1. Identificar las áreas de aplicación de la computación en la ingeniería.
2. Construir programas que satisfagan las especificaciones, verificando el comportamiento esperado de las soluciones implementadas.
3. Analizar las relaciones causa efecto de los algoritmos creados.
4. Identificar los objetivos y requerimientos de las soluciones TIC.
5. Desarrollar la solución tecnológica más adecuada en base a las características del problema y los recursos disponibles.
6. Interactuar con los integrantes del equipo considerando las diversas opiniones de manera que favorezca el logro del objetivo común.

VI. ÁREAS TEMÁTICAS

1. Conceptos básicos de programación.
   1.1. Conceptos básicos: bit, byte, hardware, software, sistema operativo, redes de computadores, sistema computacional.
   1.2. Arquitectura de un computador.
   1.3. Proceso de traducción de un programa (programa fuente, programa objeto, compilador, intérprete).
   1.4. Pruebas (Testing).
   1.5. Depuración de programas.
2. Razonamiento Lógico y Descomposición Modular.
   2.1. Estrategia de refino sucesivo.
   2.2. Descomposición de un problema en subproblemas.
   2.3. Abstracción y encapsulación.
3. Elementos básicos de programación.
   3.1. Algoritmo, programa, lenguaje de programación.
   3.2. Concepto de variable, constante, tipos de datos básicos y operadores. Alcance de las variables, ej. local, global, por bloque.
   3.3. Instrucciones: Creación y asignación de variables; Lectura y despliegue de datos desde y hacia la pantalla (input, print); Condicionales (if, switch); Ciclos (for, while, dowhile).
   3.4. Patrón de programación: contador, acumulador, bandera, entre otros.
   3.5. Programas para resolución de problemas matemáticos, con operaciones básicas y álgebra.
   3.6. Programas para leer y escribir archivos secuenciales.
   3.7. Validación de datos de entrada y casos de prueba (testing).

4. Programación con subprogramas.
   4.1. Estrategia de construcción de programas: dividir para conquistar.
   4.2. Concepto de subprograma: función.
   4.3. Funciones con diferentes tipos de retornos, ej. Void.
   4.4. Criterios para descomponer en subprogramas.
   4.5. Traspaso de parámetros.
   4.6. Alcance de variables: locales y globales.
5. Programas que utilizan arreglos.
   5.1. Concepto de arreglo: vector y matriz.
   5.2. Operaciones básicas sobre un arreglo: declaración y creación del arreglo, ingreso de elementos, obtención de un elemento, despliegue de elementos.
   5.3. Algoritmos en un arreglo: búsqueda, ordenamiento, inserción ordenada, eliminación.

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.

VIII. ORIENTACIONES Y CRITERIOS PARA 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

– Ross, Eric. (2022). Libro guía y compendio de problemas de programación. ISBN: 9789564041438

Bibliografía Complementaria

– González Duque, Raúl. (2011). Python para todos. http://mundogeek.net/tutorialpython/
– Shaw, Zed A. (2013). Learn Python the Hard Way. http://learnpythonthehardway.org/book/
– Downey, Allen B. (2016). Think Python. http://www.greenteapress.com/thinkpython2/index.html
– Idris, Ivan. (2014). Learning NumPy Array. Packt Publishing. ISBN:9781783983902.