• PROGRAMA DE ASIGNATURA

<table>
<tr><td>Nombre</td><td colspan="2">Computación I</td></tr>
<tr><td>Carrera</td><td colspan="2">Ingeniería Matemática</td></tr>
<tr><td>Código</td><td colspan="2"></td></tr>
<tr><td>Créditos SCT-Chile</td><td>Nº Sct 5</td><td>Tbjo. Directo: 6 hrs. pedag. – Tbjo. Autónomo: 9 hrs. cronolog.</td></tr>
<tr><td>Nivel</td><td colspan="2">Segundo semestre</td></tr>
<tr><td>Requisitos</td><td colspan="2">Álgebra 1</td></tr>
<tr><td>Categoría</td><td colspan="2">Obligatorio</td></tr>
<tr><td>Área de conocimiento según OCDE</td><td colspan="2">Ciencias Naturales</td></tr>
<tr><td rowspan="4">Descripción</td><td colspan="2">Contribución al Perfil de Egreso
Explicitar el o los desempeños integrales del perfil de egreso al que tributa la asignatura. Indique según numeración de perfil de egreso.</td></tr>
<tr><td colspan="2">Resultado de aprendizaje general

Conocer los problemas y los algoritmos clásicos de la computación.
Conocer los conceptos y las técnicas para analizar la correctitud y el desempeño de algoritmos. Diseñar algoritmos para resolver problemas computacionales básicos.</td></tr>
<tr><td>Resultados de aprendizaje específicos</td><td>Unidades temáticas</td></tr>
<tr><td>Computación y algoritmos:

• Conocer las nociones básicas de la teoría de algoritmos y complejidad.
• Conocer algunos de los algoritmos básicos y sus desempeños.
• Reconocer algunos elementos básicos de algoritmos como expresiones condicionales, repeticiones, etc.</td><td>1. Computación y algoritmos

1.1. Definiciones generales
1.2. Relación con el álgebra y el cálculo
1.3. Conceptos básicos
1.4. Cultura general en relación con computación y algoritmos</td></tr>
</table>

	Diseño y control de programas: • Entender las expresiones y conectores lógicos (“y”, “o”, “no”, “si, entonces”, “si y sólo si”). • Conocer las principales expresiones de control en algoritmos: expresiones condicionales (“if”, “if else”) y bucles (“for”, “while”) y sus operaciones de control (“break”, iteradores). • Reconocer el concepto de funciones/procesos en programación. • Identificar la diferencia entre variables locales y globales y entre variables de entrada y salida. • Conocer la diferencia entre algoritmos directos, iterativos y algoritmos recursivos y como se refleja en la estructura de un algoritmo.	**2. Diseño y control de programas** 2.1. Lógica computacional 2.2. Sentencias de control: if/else, for, while, break, iteradores 2.3. Uso de funciones 2.4. Variables: globales, locales, entradas y salidas, utilización de archivos 2.5. Estructuras de algoritmos: algoritmos directos, algoritmos iterativos y recursividad general
	Estructuras de datos: • Conocer las estructuras básicas encontradas en programación y cómo se pueden utilizar. • Distinguir los ventajas y desventajas de las estructuras de datos básicas en algunas situaciones comunes.	**3. Estructuras de datos** 3.1. Arreglos 3.2. Matrices 3.3. Listas
	Eficiencia y correctitud de algoritmos: • Entender la necesidad de demostrar la correctitud de un algoritmo. • Saber demostrar la correctitud de algoritmos básicos. • Tener conocimientos sobre la medición de la eficacia y eficiencia de los algoritmos para ofrecer soluciones de buen rendimiento.	4. **Eficiencia y correctitud de algoritmos.** 4.1. Correctitud. 4.2. Eficiencia temporal. 4.3. Eficiencia espacial. 4.4. Medición de error y eficiencia computacional.

	Laboratorios (aprendizaje) • Saber implementar algunos algoritmos básicos en el lenguaje de programación Python. • Conocer las etapas claves en la programación de algoritmos (diseño, verificación de correctitud, medición de eficacia y eficiencia, testeo) y aplicarlas en casos prácticos. • Saber utilizar funciones para la programación de un algoritmo. • Entender y aplicar herramientas básicas de recursividad en programación.	**Laboratorios (guía de actividades)** • Entornos de programación: herramientas de programación (Python), ambientes de desarrollo, buenas prácticas de programación. • Diseño, trazabilidad y medición: correctitud, eficacia y eficiencia • Operaciones y uso de funciones en operaciones: operaciones vectoriales y matriciales • Ordenamiento, búsqueda y uso de funciones: retorno, void y recursión simple.
	Metodologías de enseñanza y de aprendizaje La metodología contempla clases expositivas, resolución plenaria de problemas, y aplicación guías de aprendizaje, lo cual se trabajará de forma individual o colaborativa. El trabajo autónomo se desarrollará en base a tareas e implementaciones de algoritmos, las cuales pueden ser resueltas de forma individual y grupal.	
	Procedimientos de evaluación Los procedimientos de evaluación del curso contemplan evaluaciones diagnósticas, formativas y sumativas, alineadas con el contenido de las unidades y los resultados de aprendizaje definidos. Las actividades evaluativas concretas para realizar durante el curso serán claramente definidas al inicio de este.	
	Bibliografía básica ● Parker, J.R. (2016) *Python. An Introduction to Programming*. Mercury Learning & Information, Berlin ● Üçoluk, G; Kalkan, S. (2012) *Introduction to Programming Concepts with Case Studies in Python*. Springer, Vienna. ● Cormen, T.H.; Leiserson, C.E.; Rivest, R.L. (1994) Introduction to Algorithms. MIT Press / McGraw-Hill, Cambridge. ● Yang, X.-S. (2014) Introduction to Algorithms. Elsevier, EE.-UU.	