Universidad Nacional de Ucayali (UNU)
Programa Profesional de
Ciencia de la Computación
Sílabo 2023-I

1. CURSO

CS111. Introducción a la Ciencia de la Computación (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Créditos	:	4
2.2 Horas de teoría	:	2 (Semanal)
2.3 Horas de práctica	:	2 (Semanal)
2.4 Duración del periodo	:	16 semanas
2.5 Condición	:	Obligatorio
2.6 Modalidad	:	Híbrido
2.7 Prerrequisitos	:	Ninguno

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Este es el primer curso en la secuencia de los cursos introductorios a la Ciencia de la Computación. En este curso se pretende cubrir los conceptos señalados por la Computing Curricula IEEE-CS/ACM 2013. La programación es uno de los pilares de la Ciencia de la Computación; cualquier profesional del Área, necesitará programar para concretizar sus modelos y propuestas. Este curso introdución a los participantes en los conceptos fundamentales de este arte. Lo tópicos incluyen tipos de datos, estructuras de control, funciones, listas, recursividad y la mecánica de la ejecución, prueba y depuración.

5. OBJETIVOS

- Introducir los conceptos fundamentales de programación.
- Desarrollar su capacidad de abstracción utilizar un lenguaje de programación.

6. COMPETENCIAS

a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (**Usar**)

b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (**Usar**)

c) Diseñar, implementar y evaluar un sistema, proceso, componente o programa computacional para alcanzar las necesidades deseadas. (**Usar**)

i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (**Usar**)

k) Aplicar los principios de desarrollo y diseño en la construcción de sistemas de software de complejidad variable. (**Familiarizarse**)

7. TEMAS

Unidad 1: Historia (5)	
Competencias esperadas: a	
Temas	**Objetivos de Aprendizaje**
• Pre-historia – El mundo antes de 1946. • Historia del hardware, software, redes. • Pioneros de la Computación. • Historia de Internet.	• Identificar importantes tendencias en la historia del campo de la computación [Familiarizarse] • Identificar las contribuciones de varios pioneros en el campo de la computación [Familiarizarse] • Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse] • Comparar la vida diaria antes y después de la llegada de los ordenadores personales y el Internet [Evaluar]
Lecturas : [BB19], [Gut13], [Zel10]	

Unidad 2: Sistemas de tipos básicos (2)	
Competencias esperadas: a	
Temas	**Objetivos de Aprendizaje**
• Tipos como conjunto de valores junto con un conjunto de operaciones. – Tipos primitivos (p.e. números, booleanos) – Composición de tipos construídos de otros tipos (p.e., registros, uniones, arreglos, listas, funciones, referencias) • Asociación de tipos de variables, argumentos, resultados y campos. • Tipo de seguridad y los errores causados por el uso de valores de manera incompatible dadas sus tipos previstos.	• Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse] • Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse] • Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse] • Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar] • Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar] • Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar]
Lecturas : [Gut13], [Zel10]	

Unidad 3: Conceptos Fundamentales de Programación (9)	
Competencias esperadas: a	
Temas	**Objetivos de Aprendizaje**
• Sintaxis y semántica básica de un lenguaje de alto nivel. • Variables y tipos de datos primitivos (ej., numeros, caracteres, booleanos) • Expresiones y asignaciones. • Operaciones básicas I/O incluyendo archivos I/O. • Estructuras de control condicional e iterativas. • Paso de funciones y parámetros. • Concepto de recursividad.	• Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar] • Identifica y describe el uso de tipos de datos primitivos [Familiarizarse] • Escribe programas que usan tipos de datos primitivos [Usar] • Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar] • Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Escribe un programa que usa E/S de archivos para brindar persistencia a través de ejecuciones múltiples [Usar] • Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Familiarizarse] • Describe el concepto de recursividad y da ejemplos de su uso [Evaluar] • Identifica el caso base y el caso general de un problema basado en recursividad [Familiarizarse]
Lecturas : [Gut13], [Zel10]	

Unidad 4: Análisis Básico (2)	
Competencias esperadas: a,b	
Temas	**Objetivos de Aprendizaje**
• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo. • Definición formal de la Notación Big O. • Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial. • Uso de la notación Big O. • Análisis de algoritmos iterativos y recursivos.	• Explique a que se refiere con "mejor", "esperado" y "peor" caso de comportamiento de un algoritmo [Familiarizarse] • En el contexto de a algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Familiarizarse] • Indique la definición formal de Big O [Familiarizarse] • Use la notación formal de la Big O para dar límites superiores asintóticos en la complejidad de tiempo y espacio de los algoritmos [Usar] • Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Usar]
Lecturas : [Gut13], [Zel10]	

Unidad 5: Algoritmos y Estructuras de Datos fundamentales (8)

Competencias esperadas: a,b

Temas	**Objetivos de Aprendizaje**

• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo.
• Algoritmos de búsqueda secuencial y binaria.
• Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción)
• Algoritmos de ordenamiento con peor caso o caso promedio en O(N lg N) (Quicksort, Heapsort, Mergesort)
• Tablas Hash, incluyendo estratégias para evitar y resolver colisiones.
• Árboles de búsqueda binaria:
  – Operaciones comunes en árboles de búsqueda binaria como seleccionar el mínimo, máximo, insertar, eliminar, recorrido en árboles.
• Grafos y algoritmos en grafos:
  – Representación de grafos (ej., lista de adyacencia, matriz de adyacencia)
  – Recorrido en profundidad y amplitud
• Montículos (Heaps)
• Grafos y algoritmos en grafos:
  – Problema de corte máximo y mínimo
  – Busqueda local
• Búsqueda de patrones y algoritmos de cadenas/texto (ej. búsqueda de subcadena, búsqueda de expresiones regulares, algoritmos de subsecuencia común más larga)
	• Implementar algoritmos numéricos básicos [Usar]
	• Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar]
	• Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Usar]
	• Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse]
	• Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Familiarizarse]
	• Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse]
	• Explicar como el balanceamiento del arbol afecta la eficiencia de varias operaciones de un arbol de búsqueda binaria [Familiarizarse]
	• Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Usar]
	• Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar]
	• Describir la propiedad del heap y el uso de heaps como una implementación de colas de prioridad [Familiarizarse]
	• Resolver problemas usando algoritmos de grafos, incluyendo camino más corto de una sola fuente y camino más corto de todos los pares, y como mínimo un algoritmo de arbol de expansion minima [Usar]
	• Trazar y/o implementar un algoritmo de comparación de string [Usar]

Lecturas : [Gut13], [Zel10]

Unidad 6: Algoritmos y Diseño (9)	
Competencias esperadas: a,b	
Temas	**Objetivos de Aprendizaje**
• Conceptos y propiedades de los algoritmos – Comparación informal de la eficiencia de los algoritmos (ej., conteo de operaciones) • Rol de los algoritmos en el proceso de solución de problemas • Estrategias de solución de problemas – Funciones matemáticas iterativas y recursivas – Recorrido iterativo y recursivo en estructura de datos – Estrategias Divide y Conquistar • Conceptos y principios fundamentales de diseño – Abstracción – Descomposición de Program – Encapsulamiento y camuflaje de información – Separación de comportamiento y aplicación	• Discute la importancia de los algoritmos en el proceso de solución de un problema [Familiarizarse] • Discute como un problema puede ser resuelto por múltiples algoritmos, cada uno con propiedades diferentes [Familiarizarse] • Crea algoritmos para resolver problemas simples [Usar] • Usa un lenguaje de programación para implementar, probar, y depurar algoritmos para resolver problemas simples [Usar] • Implementa, prueba, y depura funciones recursivas simples y sus procedimientos [Usar] • Determina si una solución iterativa o recursiva es la más apropiada para un problema [Evaluar] • Implementa un algoritmo de divide y vencerás para resolver un problema [Usar] • Aplica técnicas de descomposición para dividir un programa en partes más pequeñas [Usar] • Identifica los componentes de datos y el comportamiento de mútiples tipos de datos abstractos [Usar] • Implementa un tipo de dato abstracto coherente, con la menor pérdida de acoplamiento entre componentes y comportamientos [Usar] • Identifica las fortalezas y las debilidades relativas entre múltiples diseños e implementaciones de un problema [Evaluar]
Lecturas : [Gut13], [Zel10]	

Unidad 7: Métodos de Desarrollo (1)	
Competencias esperadas: a,b	
Temas	**Objetivos de Aprendizaje**
• Entornos modernos de programación: – Búsqueda de código. – Programación usando libreria de componentes y sus APIs.	• Construir y depurar programas que utilizan las bibliotecas estándar disponibles con un lenguaje de programación elegido [Familiarizarse]
Lecturas : [Gut13], [Zel10]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

********* EVALUATION MISSING ********

10. BIBLIOGRAFÍA BÁSICA

[BB19] J. Glenn Brookshear and Dennis Brylow. *Computer Science: An Overview*. Ed. by PEARSON. Global Edition. Pearson, 2019. ISBN: 1292263423. URL: `http://www.pearsonhighered.com/brookshear`.

[Gut13] John V Guttag. . *Introduction To Computation And Programming Using Python*. MIT Press, 2013.

[Zel10] John Zelle. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc, 2010.