



Universidad Nacional Mayor de San Marcos
Escuela Profesional de
Ciencia de la Computación
Silabo del curso
Periodo Académico 2018-II

1. **Código del curso y nombre:** CS341. Lenguajes de Programación (Obligatorio)
2. **Créditos:** 4
3. **Horas de Teoría y Laboratorio:** 2 HT; 2 HP; 2 HL; (15 semanas)
4. **Docente(s)**

Atención previa coordinación con el profesor

5. Bibliografía

- [RH04] Peter Van Roy and Seif Haridi. *Concepts, Techniques, and Models of Computer Programming*. Cambridge, MA, USA: MIT Press, 2004. ISBN: 0262220695.
- [Seb12] Robert W. Sebesta. *Concepts of Programming Languages*. 10th. USA: Addison-Wesley Publishing Company, 2012. ISBN: 0131395319.
- [Web10] Adam Brooks Webber. *Modern Programming Languages: A Practical Introduction*. 2nd. Franklin, Beedle and Associates, Inc, 2010. ISBN: 978-1-59028-250-2.

6. Información del curso

- (a) **Breve descripción del curso** Los lenguajes de programación son el medio a través del cual los programadores describen con precisión los conceptos, formulan algoritmos y representan sus soluciones. Un científico de la computación trabajará con diferentes lenguajes, por separado o en conjunto. Los científicos de la computación deben entender los modelos de programación de los diferentes lenguajes, tomar decisiones de diseño basados en el lenguaje de programación y sus conceptos. El profesional a menudo necesitará aprender nuevos lenguajes y construcciones de programación y debe entender los fundamentos de como las características del lenguaje de programación están definidas, compuestas e implementadas. El uso eficaz de los lenguajes de programación y la apreciación de sus limitaciones, también requiere un conocimiento básico de traducción de lenguajes de programación y su análisis de ambientes estáticos y dinámicos, así como los componentes de tiempo de ejecución tales como la gestión de memoria, entre otros detalles de relevancia.
- (b) **Prerrequisitos:** CS2101. Teoría de la Computación. (4^{to} Sem)
- (c) **Tipo de Curso:** Obligatorio
- (d) **Modalidad:** Presencial

7. Objetivos específicos del curso.

- Capacitar a los estudiantes para entender los lenguajes de programación desde diferentes tipos de vista, según el modelo subyacente, los componentes fundamentales presentes en todo lenguaje de programación y como objetos formales dotados de una estructura y un significado según diversos enfoques.

8. Contribución a los resultados (Outcomes)

- 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**)
- i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (**Usar**)
- j) Aplicar la base matemática, principios de algoritmos y la teoría de la Ciencia de la Computación en el modelamiento y diseño de sistemas computacionales de tal manera que demuestre comprensión de los puntos de equilibrio involucrados en la opción escogida. (**Usar**)

9. Competencias (IEEE)

- C2.** Capacidad para tener una perspectiva crítica y creativa para identificar y resolver problemas utilizando el pensamiento computacional.⇒ **Outcome a**
- CS1.** Modelar y diseñar sistemas de computadora de una manera que se demuestre comprensión del balance entre las opciones de diseño.⇒ **Outcome b**
- CS2.** Identificar y analizar los criterios y especificaciones apropiadas a los problemas específicos, y planificar estrategias para su solución.⇒ **Outcome i**
- CS3.** Analizar el grado en que un sistema basado en el ordenador cumple con los criterios definidos para su uso actual y futuro desarrollo.⇒ **Outcome j**
- CS6.** Evaluar los sistemas en términos de atributos de calidad en general y las posibles ventajas y desventajas que se presentan en el problema dado.⇒ **Outcome j**

10. Lista de temas a estudiar en el curso

- 1.
2. Pragmática de lenguajes
3. Sistemas de tipos
4. Programación orientada a objetos
5. Programación funcional
6. Programación reactiva y dirigida por eventos
7. Programación lógica

11. Metodología y Evaluación

Metodología:

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.

Sesiones de Laboratorio:

Para verificar que los alumnos hayan alcanzado el logro planteado para cada una de las unidades de aprendizaje, realizarán actividades que les permita aplicar los conocimientos adquiridos durante las sesiones de teoría y se les propondrá retos que permitan evaluar el desempeño de los alumnos.

Exposiciones individuales o grupales:

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.

Lecturas:

A lo largo del curso se proporcionan diferentes lecturas, las cuales son evaluadas. El promedio de las notas de las lecturas es considerado como la nota de una práctica calificada. El uso del campus virtual UTEC Online permite a cada estudiante acceder a la información del curso, e interactuar fuera de aula con el profesor y con los otros estudiantes.

Sistema de Evaluación:

12. Contenido

Unidad 1: (18)	
Competencias esperadas: C2, CS1, CS2	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Reconocer el desarrollo histórico de los lenguajes de programación. [Familiarizarse] • Identificar los paradigmas que agrupan a la mayoría de lenguajes de programación existentes hoy en día. [Familiarizarse] • Explicar como programas que procesan otros programas tratan a los otros programas como su entrada de datos [Familiarizarse] • Describir un árbol de sintaxis abstracto para un lenguaje pequeño [Familiarizarse] • Escribir un programa para procesar alguna representación de código para algún propósito, tales como un interprete, una expresión optimizada, o un generador de documentación [Usar] • Distinguir una definición de un lenguaje de una implementación particular de un lenguaje (compilador vs interprete, tiempo de ejecución de la representación de los objetos de datos, etc) [Familiarizarse] • Reconocer como funciona un programa a nivel de computador. [Familiarizarse] 	<ul style="list-style-type: none"> • Historia de los Lenguajes de Programación • Programas que tienen otros programas como entrada tales como interpretes, compiladores, revisores de tipos y generadores de documentación. • Estructuras de datos que representan código para ejecución, traducción o transmisión. • Estructura de un programa: Léxico, Sintáctico y Semántico • BNF • Interpretación vs. compilación a código nativo vs. compilación de representación portable intermedia. [Familiarizarse]
Lecturas : [Seb12], [Web10]	

Unidad 2: Pragmática de lenguajes (12)	
Competencias esperadas: C2, CS1, CS2	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Discute el rol de conceptos como ortogonalidad y el buen criterio de selección en el diseño de lenguajes [Usar] • Utiliza criterios objetivos y nítidos para evaluar las decisiones en el diseño de un lenguaje [Usar] • Da un ejemplo de un programa cuyo resultado puede diferir dado diversas reglas de orden de evaluación, precedencia, o asociatividad [Usar] • Muestra el uso de evaluación con retraso, como en el caso de abstracciones definidas y controladas por el usuario [Familiarizarse] • Discute la necesidad de permitir llamadas a librerías externas y del sistema y las consecuencias de su implementación en un lenguaje [Familiarizarse] 	<ul style="list-style-type: none"> • Principios de diseño de lenguaje tales como la ortogonalidad. • Orden de evaluación, precedencia y asociatividad. • Evaluación tardía vs. evaluación temprana. • Definiendo controles y constructos de iteración. • Llamadas externas y sistema de librerías.
Lecturas : [Seb12], [Web10], [RH04]	

Unidad 3: Sistemas de tipos (18)	
Competencias esperadas: C2, CS1, CS2	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Definir un sistema de tipo de forma precisa y en su composición [Usar] • Para varias construcciones de tipo fundamental, identificar los valores que describen y las invariantes que hacen que se cumplan [Familiarizarse] • Precisar las invariantes preservadas por un sistema de tipos seguro (<i>sound type system</i>) [Familiarizarse] • Demostrar la seguridad de tipos para un lenguaje simple en términos de conservación y progreso teoremas [Usar] • Implementar un algoritmo de inferencia de tipos basado en la unificación para un lenguaje básico [Usar] • Explicar cómo la sobrecarga estática y algoritmos de resolución asociados influyen el comportamiento dinámico de los programas [Familiarizarse] 	<ul style="list-style-type: none"> • Constructores de tipo composicional, como tipos de producto (para agregados), tipos de suma (para uniones), tipos de función, tipos cuantificados y tipos recursivos. • Comprobación de tipos. • Seguridad de tipos como preservación más progreso. • Inferencia de tipos. • Sobrecarga estática.
Lecturas : [Seb12], [Web10], [RH04]	

Unidad 4: Programación orientada a objetos (12)	
Competencias esperadas: CS2, CS3, CS6	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Diseñar e implementar una clase [Usar] • Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar] • Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar] • Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar] • Explicar la relación entre la herencia orientada a objetos (codigo compartido y <i>overriding</i>) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Usar] • Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar] • Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, seleccionar la forma mas natural por cada lenguaje [Usar] 	<ul style="list-style-type: none"> • Diseño orientado a objetos: <ul style="list-style-type: none"> – Descomposicion en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquia de clases para modelamiento • Definición de las categorías, campos, métodos y constructores. • Las subclases, herencia y método de alteración temporal. • Asignación dinámica: definición de método de llamada. • Subtipificación: <ul style="list-style-type: none"> – Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos. – Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos. – Relación entre subtipos y la herencia. • Lenguajes orientados a objetos para la encapsulación: <ul style="list-style-type: none"> – privacidad y la visibilidad de miembros de la clase – Interfaces revelan único método de firmas – clases base abstractas • Uso de coleccion de clases, iteradores, y otros componentes de la libreria estandar.
Lecturas : [Seb12], [Web10], [RH04]	

Unidad 5: Programación funcional (18)	
Competencias esperadas: CS2, CS3, CS6	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Escribir algoritmos básicos que eviten asignación a un estado mutable o considerar igualdad de referencia [Usar] • Escribir funciones útiles que puedan tomar y retornar otras funciones [Usar] • Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar] • Razonar correctamente sobre variables y el ámbito léxico en un programa usando funciones de cierre (<i>function closures</i>) [Usar] • Usar mecanismos de encapsulamiento funcional, tal como <i>closures</i> e interfaces modulares [Usar] • Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, seleccionar la forma mas natural por cada lenguaje [Usar] 	<ul style="list-style-type: none"> • El efecto de la programación libre: <ul style="list-style-type: none"> – Llamadas a función que no tiene efecto secundarios, para facilitar el razonamiento composicional – Variables inmutables, prevencion de cambios no esperados en los datos del programa por otro código. – Datos que pueden ser subnombrados o copiados libremente sin introducir efectos no deseados del cambio • Procesamiento de estructuras de datos (p.e. arboles) a través de fuciones con casos para cada variación de los datos. <ul style="list-style-type: none"> – Constructores asociados al lenguaje tales como uniones discriminadas y reconocimiento de patrones sobre ellos. – Funciones definidas sobre datos compuestos en términos de funciones aplicadas a las piezas constituidas. • Funciones de primera clase (obtener, retornar y funciones de almacenamiento) • Cierres de función (funciones que usan variables en entornos léxicos cerrados) <ul style="list-style-type: none"> – Significado y definicion básicos - creacion de cierres en tiempo de ejecución mediante la captura del entorno. – Idiomas canónicos: llamadas de retorno, argumentos de iteradores, código reusable mediante argumentos de función – Uso del cierre para encapsular datos en su entorno – Evaluación y aplicación parcial • Definición de las operaciones de orden superior en los agregados, especialmente en mapa, reducir / doblar, y el filtro.
Lecturas : [Seb12], [Web10], [RH04]	

Unidad 6: Programación reactiva y dirigida por eventos (12)	
Competencias esperadas: CS2, CS3, CS6	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Escribir manejadores de eventos para su uso en sistemas reactivos tales como GUIs [Usar] • Explicar porque el estilo de programación manejada por eventos es natural en dominios donde el programa reacciona a eventos externos [Familiarizarse] • Describir un sistema interactivo en términos de un modelo, una vista y un controlador [Familiarizarse] 	<ul style="list-style-type: none"> • Eventos y controladores de eventos. • Usos canónicos como interfaces gráficas de usuario, dispositivos móviles, robots, servidores. • Uso de frameworks reactivos. <ul style="list-style-type: none"> – Definición de controladores/oyentes (handles/listeners) de eventos. – Bucle principal de eventos no controlado por el escritor controlador de eventos (event-handler-writer) • Eventos y eventos del programa generados externamente generada. • La separación de modelo, vista y controlador.
Lecturas : [Seb12]	

Unidad 7: Programación lógica (12)	
Competencias esperadas: CS2, CS3, CS6	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • Usa un lenguaje lógico para implementar un algoritmo convencional [Usar] • Usa un lenguaje lógico para implementar un algoritmo empleando búsqueda implícita usando cláusulas, relaciones, y cortes [Usar] 	<ul style="list-style-type: none"> • Representación causal de estructura de datos y algoritmos. • Unificación. • Backtracking y búsqueda. • Cuts.
Lecturas : [Seb12], [Web10], [RH04]	