Ministerio de Cultura y Educación **(Programa del año 2024)**
Universidad Nacional de San Luis **(Programa en trámite de aprobación)**
Facultad de Ciencias Físico Matemáticas y Naturales **(Presentado el 12/09/2024 13:16:02)**
Departamento: Informatica
Area: Area V: Automatas y Lenguajes

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
ANALISIS COMPARATIVO DE LENGUAJES	PROF.CS.COMPUT.	02/16	2024	2° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
ROGGERO, PATRICIA BEATRIZ	Prof. Responsable	P.Asoc Exc	40 Hs
ARAGON, VICTORIA SOLEDAD	Prof. Colaborador	P.Adj Exc	40 Hs
FUNEZ, DARIO GUSTAVO	Responsable de Práctico	JTP Exc	40 Hs
GATICA, CLAUDIA RUTH	Auxiliar de Práctico	A.1ra Semi	20 Hs

III - Características del Curso

Credito Horario Semanal				
Teórico/Práctico	**Teóricas**	**Prácticas de Aula**	**Práct. de lab/ camp/ Resid/ PIP, etc.**	**Total**
Hs	3 Hs	3 Hs	2 Hs	8 Hs

Tipificación	Periodo
B - Teoria con prácticas de aula y laboratorio	2º Cuatrimestre

Duración			
Desde	**Hasta**	**Cantidad de Semanas**	**Cantidad de Horas**
05/08/2024	15/11/2024	15	120

IV - Fundamentación

Los conceptos involucrados en el estudio del diseño de lenguajes de programación y los paradigmas a los que responden, forman parte del conocimiento general de un Licenciado en Ciencias de la Computación, con el perfil requerido por los correspondientes planes de estudios. Esta asignatura complementa los contenidos introducidos en las materias Programación I y Programación II, donde el énfasis está puesto en la resolución de problemas utilizando diferentes lenguajes de programación, esto brinda al/la estudiante una primera aproximación a distintos aspectos y construcciones de los lenguajes de programación utilizados, como así también a los paradigmas a los cuales estos pertenecen.

Circunscribir el conocimiento de un Licenciado en Ciencias de la Computación a un paradigma o lenguaje particular, acota su visión de las herramientas existentes a las particularidades del lenguaje y paradigma utilizados. Esto no responde en general a las necesidades actuales, con un mercado laboral cambiante, que plantea de forma continua problemas de diversa naturaleza, cuya resolución efectiva puede requerir de construcciones y lenguajes diversos.

Es por esto que resulta necesario brindar al futuro/a Licenciado/a una visión global de los lenguajes, donde se exploren los principales conceptos de diseño subyacentes y su efecto sobre la implementación de los lenguajes. Esta visión permite, entre otras cosas, mejorar la habilidad para desarrollar algoritmos eficaces, mejorar el uso del lenguaje de programación disponible, acrecentar el propio vocabulario con construcciones útiles de programación, hacer posible una mejor elección del lenguaje de programación y facilitar el aprendizaje de un nuevo lenguaje. Si a todo esto se le suma la identificación de los principios subyacentes de los principales paradigmas de lenguajes de programación, y una comparación crítica entre los mismos, se puede decir que se le brinda a los/las estudiantes las herramientas necesarias para enfrentar sus necesidades presentes y futuras a la hora de elegir y usar de manera adecuada un lenguaje de programación.

Además, en el convencimiento de que los aspectos básicos del diseño de compiladores deberían formar parte del conocimiento general de cualquier buen programador/a, este curso introduce la estructura básica de un compilador. También se abordan aspectos relacionados a la teoría formal de lenguajes, que son necesarios para entender el funcionamiento de un compilador, pero servirán además como introducción a tópicos más avanzados a desarrollarse en otros cursos de la carrera como Autómatas y Lenguajes, y Diseño y Construcción de Compiladores.
Desde este lugar la propuesta es la de, en un trabajo conjunto de docentes y estudiantes poder dar respuesta a las siguientes preguntas: ¿por qué estudiar lo que subyace al concepto de lenguajes de programación?, ¿es necesario ahondar en los aspectos que refieren a la implementación de lenguajes?, ¿por qué?, ¿es posible pensarnos como futuros/as desarrolladores/as de lenguajes?, ¿cómo se puede visualizar que todo esto ayuda al desarrollo de algoritmos eficaces, a mejorar el uso de lenguajes disponibles, a incrementar el vocabulario en lo que refiere a construcciones de programación útiles? En pos a dar respuesta a estos interrogantes, teniendo como meta trabajar para lograr una buena enseñanza, todo enmarcado en un clima de camaradería entre docentes y estudiantes, propiciando el debate, el trabajo colaborativo, la tarea en grupos, la autonomía, la participación, el contexto particular del aula y la educación para un mundo mejor, y para que los/las estudiantes empiecen a verse/sentirse como futuros profesionales, las clases serán llevadas a cabo de manera coherente para trabajar en estos aspectos descriptos.

V - Objetivos / Resultados de Aprendizaje

Se presentan a continuación una serie de objetivos tanto de carácter general o transversal como de carácter específico, que se pretenden que alcanzar a lo largo del dictado del curso.
Objetivos Generales:
- Analizar críticamente los contenidos abordados, propiciando actividades de análisis de los marcos teóricos con la realidad del campo laboral.
- Vincular la teoría y la práctica con la intención de encontrar alternativas que promuevan la buena enseñanza.
- Integrar el trabajo grupal, colaborativo y la autonomía, en toda la secuencia didáctica propuesta.
- Brindar las ideas centrales del proceso completo de diseño de lenguajes de programación.
Objetivos Específicos:
- Evaluar en forma crítica distintos lenguajes de programación existentes y futuros, desde la perspectiva del diseño de los mismos.
- Responder a cuestiones tales como las motivaciones de la existencia de tantos lenguajes de programación, cómo y porqué fueron desarrollados, en qué se asemejan y difieren.
- Entender la implementación de distintos lenguajes, como para reconocer la relación entre un programa fuente y su comportamiento en ejecución. En particular comprender el funcionamiento de un compilador.
- Extender sus conocimientos sobre los tópicos anteriores con bibliografía adecuada y mínima supervisión.
- Adquirir la capacidad de evaluar lenguajes de programación desde distintos puntos de vista, ya sea como su diseñador/a, implementador/a o como usuario/a del lenguaje.

VI - Contenidos

Contenidos mínimos:
Historia de los lenguajes de programación. Computadora de hardware, de firmware y simulada por software. Traductores. Computadoras virtuales y ligaduras. Tiempos de ligadura. Evolución de paradigmas de programación. Propiedades de tipos y objetos. Tipos de datos elementales y estructurados. Especificación e implementación de tipos de datos. Paradigmas de lenguajes. Sintaxis de los lenguajes de programación. Métodos formales para la descripción de semántica. Modelos de traducción formales. Evolución del concepto de tipo de datos. Tipos de datos abstractos. Lenguajes Orientados a Objetos (00): herencia, polimorfismo y ligadura dinámica, subclases y subtipos, herencia simple y múltiple. Análisis comparativo de lenguajes representativo del paradigma OO. Administración de memoria. Fases de la administración de memoria. Control de subprogramas. Control de datos. Ambientes de referenciación. Alcance estático y dinámico. Datos compartidos en subprogramas. Control de secuencia explícito e implícito. Variantes en control de subprogramas. Excepciones. Co-rutinas. Subprogramas planificados. Comandos en guardia. Tareas. Programación lógica. Análisis de un lenguaje representativo del paradigma lógico.

UNIDAD 1. Características del Diseño de Lenguajes de Programación.
Razones del estudio de lenguajes de programación. Historia de los lenguajes de programación. Características de un buen lenguaje. La estructura y operación de una computadora. Computadora de hardware, de firmware y simuladas por software. Traductores e Intérpretes. Computadoras virtuales y ligaduras. Tiempos de ligadura. Evolución de Paradigmas de

Programación.

UNIDAD 2. Programación Lógica.
El lenguaje Prolog. Elementos básicos: hechos, predicados y consultas. Átomos y variables. Variables anónimas. Matching. Reglas recursivas. Proceso de backtracking. Functores. Listas.

UNIDAD 3. Aspectos Formales de los Lenguajes de Programación
Sintaxis de los lenguajes de programación. Criterios sintácticos generales. Métodos formales para la descripción de semántica. Elementos sintácticos de un lenguaje. Estructura programa subprograma completo. Etapas en la traducción. Modelos de traducción formales. Lenguajes Regulares y Lenguajes Libres del Contexto. Expresiones Regulares. Gramáticas Libres del Contexto. BNF, BNFE.

UNIDAD 4. Tipos de Datos
Propiedades de tipos y objetos de datos. Tipos de datos. Especificación e implementación de tipos de datos elementales. Operaciones. Declaraciones. Chequeo de tipos. Tipos de datos numéricos. Enumeraciones. Booleanos. Caracteres. Tipos de datos estructurados. Especificación e implementación. Vectores y arreglos. Registros. Cadenas de caracteres. Punteros. Control de secuencia explícito e implícito.

UNIDAD 5. Administración de Memoria
Definición y activación de subprogramas. Administración de memoria. Fases de la administración de memoria. Administración de memoria estática. Administración de memoria basada en Pila. Heap con elementos de tamaño fijo y de tamaño variable. Mecanismos de Recuperación.

UNIDAD 6. Control de Datos en Subprogramas
Control de subprogramas. Llamada-retorno simple. La regla de la copia. Subprogramas recursivos. Control de datos. Ambientes de referenciación. Alcance estático y dinámico. Estructura de bloques. Datos compartidos en subprogramas. Parámetros. Pasaje de parámetros. Ambientes comunes explícitos. Alcance dinámico, reglas e implementaciones. Alcance estático, reglas e implementaciones.

UNIDAD 7. Variantes del Control de Subprogramas
Variantes en control de subprogramas. Excepciones y manejadores de excepciones. Co-rutinas. Subprogramas planificados. Programación paralela: comandos en guardia, tareas.

UNIDAD 8. Programación Orientada a Objetos
Tipos de datos abstractos. Evolución del concepto de tipo de datos. Ocultamiento de la información. Encapsulamiento mediante subprogramas. Programación orientada a objetos: herencia, polimorfismo y ligadura dinámica. Subclases y subtipos. Herencia simple y múltiple. Análisis comparativo de lenguajes representativos del paradigma orientado a objetos: Smalltalk, C++ y Java.

VII - Plan de Trabajos Prácticos

Todos los prácticos comienzan con ejercicios sencillos para que los/las estudiantes puedan resolver de manera autónoma y posteriormente en clase consultar la resolución. Además cuentan con ejercicios para que los/las estudiantes resuelvan en clase, con apoyo docente, en algún caso para trabajar de manera individual y en otros buscar las soluciones de manera grupal y con discusión en clase de los resultados obtenidos.

Práctico 1: Prolog (aula y formación experimental).
Objetivos: comprender las características de un lenguaje correspondiente al Paradigma Lógico, aprender los aspectos claves del lenguaje Prolog, reglas de matching, recursión y estructuras.
Metodología: resolución de ejercicios en lápiz y papel, luego ejecutar dichos ejercicios en computadora.

Práctico 2: Aspectos del diseño de lenguajes de programación.
Objetivos: visualizar las características de un buen lenguaje, computadora virtual, ejercitar sobre los distintos tipos de

ligaduras.
Metodología: análisis de ejercicios resueltos y resolución de ejercicios en lápiz y papel.

Práctico 3: Descripción Formal de Lenguajes.
Objetivos: ejercitar expresiones regulares, gramáticas libres del contexto, BNF y BNFE.
Metodología: resolución de ejercicios en lápiz y papel.

Práctico 4: Tipos de Datos Elementales y Estructurados.
Objetivos: ejercitar sobre operaciones, signatura, declaraciones, tipos de datos elementales y estructurados. Chequeo de tipos.
Metodología: resolución de ejercicios en lápiz y papel.

Práctico 5: Administración de Memoria.
Objetivos: ejercitar definición y activación de subprogramas, mecanismos de administración de memoria, problemas asociados al uso de punteros y métodos de recuperación.
Metodología: resolución de ejercicios en lápiz y papel.

Práctico 6: Control de Datos en Subprogramas.
Objetivo: realizar ejercicios referidos a ambientes de referenciación, implementación de reglas de alcance estático y dinámico.
Metodología: resolución de ejercicios en lápiz y papel.

Práctico 7: Variantes en control de subprogramas y POO.
Objetivo: desarrollo de ejercicios con variantes en control de subprogramas: excepciones, co-rutinas, subprogramas planificados y POO.
Metodología: resolución de ejercicios en lápiz y papel.

Práctico de Formación Experimental Grupal/Individual:
Realización de un proyecto de implementación de un problema de la vida real, utilizando conceptos de la teoría formal de lenguajes y compiladores.
Objetivos: analizar el problema planteado y responder una guía de trabajo.
Metodología: resolución en computadora de escritorio, se desarrollará en grupos, la presentación se realizará en al menos 3 etapas. Al finalizar el trabajo completo se requerirá la entrega de un informe escrito (en base a una guía de pautas propuestas), donde se explique el proceso llevado a cabo para la implementación y además se respondan preguntas afines al trabajo, posteriormente el trabajo deberá ser presentado de forma oral para toda la clase.

Práctico de Investigación Grupal:
Dicho trabajo consistirá en la realización de un informe (en base a una guía de pautas propuestas), sobre el tema Análisis Lexicográfico, primera etapa de cualquier compilador típico.
Objetivos: investigar sobre el tema propuesto y escribir un informe que describa los aspectos centrales de esta etapa de análisis, juntamente con el uso de herramientas para la generación de analizadores lexicográficos.
Metodología: recopilación de material, análisis de esta etapa inicial de cualquier compilador típico. Uso de un analizador lexicográfico provisto por la cátedra con la finalidad de comprender su funcionamiento.

VIII - Regimen de Aprobación

La propuesta de trabajo, en lo que respecta al régimen de aprobación, está basada en la evaluación continua o formativa, con el objetivo de relacionar la información sobre la evolución del proceso de aprendizaje de los/las estudiantes con las características de la acción didáctica, a medida que se desarrollan y progresan las actividades de enseñanza y aprendizaje. Al comenzar el dictado de la materia se proveerá a los/las estudiantes el cronograma de las distintas instancias que conforman el proceso de evaluación continua.

El/la estudiante puede regularizar (para luego rendir el examen final) o promocionar, las condiciones que se establecen son las siguientes:

A. Régimen para Estudiantes Regulares

1. Entregar, en tiempo y forma, y aprobar el 100% de las actividades prácticas requeridas por la cátedra, las cuales consistirán en: la resolución de ejercicios del tipo de los prácticos de aula y/o la entrega de ejercicios particulares de los prácticos de aula, el práctico de formación experimental y el trabajo de investigación.
2. Aprobar 1 evaluación parcial o alguna de sus 2 (dos) recuperaciones, según lo establecido en la normativa vigente. Dicha evaluación parcial se aprueba con una nota mínima de 7 (siete).
3. Contar con un porcentaje mínimo igual o superior al 70% de asistencia a clases teóricas y prácticas.
Con la finalidad de realizar un trabajo colaborativo de comprensión, se efectuarán devoluciones de las instancias evaluativas como parte del proceso de aprendizaje. Además, algunas actividades pueden tener una instancia de defensa oral posterior a la entrega donde el/la estudiante exponga, explique y responda dudas o inquietudes de sus compañeros y/o docentes acerca del trabajo realizado en sus entregas.

B. Régimen para Estudiantes Promocionales
1. Ídem a lo requerido para estudiantes regulares, salvo el ítem 3., dado que se requiere un 80 % de asistencia a las clases teóricas y prácticas.
2. Los/las estudiantes deberán responder un test teórico, en la instancia de evaluación de los prácticos 1 y 2, y en la del parcial, en la que se requiere un mínimo del 70% correcto de cada ejercicio, con el objetivo de que el/la estudiante pueda corroborar el nivel de apropiación de los contenidos teóricos, favoreciendo de esta manera la formación continua.
3. Habiendo cumplimentado los ítems 1. y 2., el/la estudiante tendrá que desarrollar y aprobar un coloquio de carácter integrador oral o escrito que incluya el análisis y desarrollo de todos los conceptos teóricos dictados en el curso.
En la nota final de aprobación se contemplarán las distintas instancias propuestas (actividades prácticas y la evaluación parcial). En todas las instancias, la nota obtenida por el/la estudiante debe ser igual a 7 o superior, incluido el coloquio de carácter integrador.

C. El curso no admite rendir el examen final en condición de Libre.

IX - Bibliografía Básica

[1] Programming Languages - Design and Implementation". Pratt, Terrence y Zelkowitz, Marvin. Cuarta edición. Prentice
[2] Hall, 2001.
[3] "Concepts of Programming Languages". Sebesta, Robert. Addison-Wesley. Edición 2019 y Edición 2016.
[4] "Prolog, programming for artificial intelligence". Bratko, Ivan. Addison-Wesley. Tercera Edición, 2001.
[5] "El lenguaje de Programación C". Kernighan, Brian y Ritchie, Dennies. Prentice Hall, 1991.
[6] "Piensa en Java". Eckel Bruce. Pearson Alhambra. Cuarta Edición, 2007.
[7] Apunte de la Cátedra Aspectos formales de la traducción de lenguajes
[8] Apunte de la Cátedra de Evolución del Concepto de Tipos de Datos - Tipo de Dato Abstracto - Programación Orientada a Objetos.

X - Bibliografia Complementaria

[1] "Lenguajes de Programación - Diseño e Implementación". Pratt, Terrence y Zelkowitz, Marvin. Tercera edición. Prentice Hall, 1999.
[2] "Programming Language Pragmatics" por Michael L. Scott. Tercera edición. Morgan Kaufmann Publishers, 2009.
[3] "Smalltalk-80. The Language and its implementation". Goldberg, Adele y Robson, David. Addison-Wesley, 1985.
[4] The Java class libraries. Chan, Patrick - Lee, Rosanna y Kramer, Douglas. Addsion Wesley. Segunda Edición, 1998.

XI - Resumen de Objetivos

El curso tiene como objetivos introducir al/la estudiante a la problemática de las características principales del diseño e implementación de lenguajes de programación, incluyendo fundamentos teóricos y modelos formales. Comprender el funcionamiento de los modelos de implementación de los lenguajes de programación, en particular de compiladores. Adquirir la habilidad de evaluar lenguajes de programación. El estudio se realiza teniendo en cuenta todos los paradigmas actuales de programación, propiciando el análisis crítico de los contenidos, el trabajo grupal, colaborativo y la autonomía.

XII - Resumen del Programa

Historia y evolución de los lenguajes de programación. Características de un buen lenguaje de programación. Métodos de implementación. Características principales de los paradigmas de programación. Programación lógica, lenguaje Prolog. Sistemas de traducción. Sintaxis y semántica. Gramáticas, expresiones regulares, autómatas. Características esenciales de los lenguajes de programación y su implementación: tipos de datos y su representación, control de secuencia y datos. Administración de Memoria. Control de datos a nivel de subprogramas. Variantes en el control de subprogramas. Tópicos avanzados de la programación orientada a objetos.

XIII - Imprevistos

XIV - Otros

Las vías de comunicación con los estudiantes son las siguientes:
- https://dirinfo.unsl.edu.ar/academico/carreras/profesorado-en-ciencias-de-la-computacion
- Classroom
- Correos electrónicos de los docentes:

Patricia Roggero: proggero@email.unsl.edu.ar
Victoria Aragón: vsaragon@email.unsl.edu.ar
Darío Funez: dgfunez@email.unsl.edu.ar
Claudia Gatica: crgatica@email.unsl.edu.ar

ELEVACIÓN y APROBACIÓN DE ESTE PROGRAMA	
	Profesor Responsable
Firma:	
Aclaración:	
Fecha:	