**Ministerio de Cultura y Educación** **(Programa del año 2022)**
**Universidad Nacional de San Luis**
**Facultad de Ciencias Físico Matemáticas y Naturales**
**Departamento: Informatica**
**Area: Area V: Automatas y Lenguajes**

## I - Oferta Académica

| Materia | Carrera | Plan | Año | Período |
|---|---|---|---|---|
| ANALISIS COMPARATIVO DE LENGUAJES | LIC.CS.COMP. | 18/11 | 2022 | 2° cuatrimestre |
| ANALISIS COMPARATIVO DE LENGUAJES | PROF.CS.COMPUT. | 02/16 | 2022 | 2° cuatrimestre |
| ANALISIS COMPARATIVO DE LENGUAJES | LIC.CS.COMP. | 32/12 | 2022 | 2° cuatrimestre |
| ANALISIS COMPARATIVO DE LENGUAJES | PROF.CS.COMPUT. | 06/09 | 2022 | 2° cuatrimestre |

## II - Equipo Docente

| Docente | Función | Cargo | Dedicación |
|---|---|---|---|
| ROGGERO, PATRICIA BEATRIZ | Prof. Responsable | P.Asoc Exc | 40 Hs |
| ERRECALDE, MARCELO LUIS | Prof. Colaborador | P.Asoc 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 | 4 Hs | 1 Hs | 8 Hs |

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

| Duración | | | |
|---|---|---|---|
| **Desde** | **Hasta** | **Cantidad de Semanas** | **Cantidad de Horas** |
| 08/08/2022 | 18/11/2022 | 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 los 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.

Los contenidos del curso se corresponden con las unidades de conocimiento recomendadas por la ACM/IEEE Computer Society Joint Curriculum Task Force, para el área de lenguajes de programación.

## 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 pretende que alcancen los/las estudiantes a lo largo del dictado del curso.

Propósitos Educativos
- 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.
- Comprender las ideas centrales del proceso completo de diseño de lenguajes de programación.

Objetivos del Aprendizaje:
- Evaluar en forma crítica distintos lenguajes de programación existentes y futuros, desde la perspectiva del diseño de los mismos.
- Responder a cuestiones tales cómo las motivaciones de la existencia de tantos lenguajes de programación, cómo y porqué fueron desarrollados, en qué se asemejan y difieren.
- Reconocer los paradigmas claves usados en el desarrollo de lenguajes de programación modernos, sus bases teóricas, aplicativas y de implementación.
- Entender la implementación de distintos lenguajes, como para reconocer la relación entre un programa fuente y su comportamiento en ejecución.
- 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.
- Identificar, formular y resolver problemas que involucren el uso de lenguajes de programación.

## VI - Contenidos

**A continuación se desarrolla la propuesta de contenidos para la asignatura, que está centrada en la definición de cuatro núcleos temáticos o curriculares:**

**PRIMER NÚCLEO TEMÁTICO: Paradigmas de programación.**

- Características principales de los paradigmas lógico, funcional, imperativo y orientado a objetos. Historia y evolución. Influencia de los paradigmas en el diseño de lenguajes.
- Caso de estudio del paradigma lógico: Lenguaje de programación Prolog, su vinculación con Inteligencia Artificial. Características principales del lenguaje, comparación con otros lenguajes conocidos (C, Java, etc.).

**SEGUNDO NÚCLEO TEMÁTICO: Aspectos centrales del diseño de lenguajes de programación.**
- ¿Cómo surgieron los lenguajes de programación? Historia y evolución de los mismos. ¿A qué llamamos un buen lenguaje?, criterios de selección. Aspectos centrales que influencian el diseño de lenguajes. Modelos de hardware, de firmware y simulados por software. Métodos de implementación (compiladores e intérpretes).
- Aspectos formales del diseño de lenguajes, sintaxis. Métodos de traducción formales: importancia, dispositivos generadores y reconocedores. Autómatas o máquinas de estados. Expresiones Regulares. Gramáticas, árboles de derivación, BNF y BNFE. Las implicancias de estos modelos en el diseño de compiladores. Métodos formales para la descripción semántica.
- Evolución del concepto de tipos de datos, tipos elementales y estructurados, especificación y análisis de las características de implementación de los tipos comúnmente usados. Objetos de datos, significado y usos.
- ¿Cómo almacenar los distintos objetos de datos?, fases de la administración de la memoria. Mecanismos: estático, basado en pila, heap. Métodos de recuperación de la memoria. Definición y activación de subprogramas.
- Control de secuencia implícito y explícito.
- Estructura de llamada-retorno de subprogramas, ¿qué implica?. Subprogramas recursivos, anidamientos de subprogramas, lenguajes representativos. Control en subprogramas, control de datos y datos compartidos entre subprogramas, ambientes de referenciación. Ambientes locales explícitos, a través del uso de parámetros y/o ambientes no comunes implícitos basados en alcance estático o alcance dinámico. Reglas de alcance estático y dinámico, implementación de las mismas, relación con los procesos de interpretación y compilación.

**TERCER NÚCLEO TEMÁTICO: Variantes en el control de subprogramas.**
- ¿Cómo afecta en el diseño de lenguajes de programación la estructura de llamada retorno de subprograma?
- Excepciones y manejadores de excepciones. Corutinas.
Subprogramas planificados. Programación paralela: comandos en guardia, tareas. Conceptos principales de sus implementaciones e implicancias.

**CUARTO NÚCLEO TEMÁTICO: Tópicos avanzados de la Programación Orientada a Objetos**
- Tipos de datos abstractos. Lenguajes orientados a objetos: herencia, polimorfismo y ligadura dinámica, subclases y subtipos, herencia simple y múltiple.
- Análisis comparativo de lenguajes representativos del paradigma orientado a objetos, con la intención de identificar sus potencialidades y la facilidad para proveer buenas prácticas de programación.

## 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, autómatas finitos, 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, corutinas, 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 tipo de datos estructurado.
Objetivos: seleccionar la estructura adecuada para representar tal tipo de datos, resolver aspectos claves de la administración de la memoria.
Metodología: resolución en computadora de escritorio, desarrollado en grupos de 2 estudiantes y entrega de informe escrito individual, donde se explique: el proceso llevado a cabo para la implementación y además se respondan preguntas afines al trabajo.

Práctico de Investigación Grupal:
Desarrolo de un trabajo de investigación sobre el tema Paradigmas de Programación, el cual tiene que incluir en el grupo al menos un/una estudiante extranjero/a. El objetivo principal de la inclusión de un/una estudiante extranjero/a es que haga aportes sobre la temática a investigar. La elección del/la estudiante se realiza a criterio del grupo, pero la cátedra entrega una lista sugerida de carreras afines del exterior.
La consigna es que cada grupo contacte con al menos un/una estudiante de una Universidad del exterior, que esté cursando una carrera de similares característica a la Lic. en Cs. de la Computación, y que haya o esté estudiando el tema Paradigmas de Programación. Lá cátedra proveerá un listado de posibles carreras afines, del exterior, pero con la finalidad de que sirva para orientar a los/las estudiantes. El grupo tendrá que elaborar un informe escrito, que se presente en etapas a lo largo del cuatrimestre, que incluya las respuestas a los interrogantes que se planteen, apoyados en la bibliografía sugerida, pero con la libertad de consultar todo el material que consideren oportuno. También el informe tendrá que exponer la experiencia observada al trabajar con estudiantes del exterior. Una vez finalizado el trabajo cada grupo lo socializará en clase.

## 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.

Con la finalidad de realizar un trabajo colaborativo de comprensión, se efectuarán devoluciones de las instancias evaluativas como parte del proceso de aprendizaje. También se podrán incluir actividades de autoevaluación y co evaluación.

En este sentido, el/la estudiante puede regularizar (para luego rendir el examen final) o promocionar:

A. Para regularizar la asignatura el/la estudiante deberá:

A1.Entregar, en tiempo y forma, y aprobar el 100% de las evaluaciones periódicas obligatorias solicitadas por la cátedra. Estas evaluaciones tienen por objetivo abordar por parte cada núcleo temático (o práctico), de manera que los/las estudiantes puedan reconocer que se han apropiado de los conocimientos.
Estas evaluaciones consistirán de la entrega de ejercicios de prácticos de aula resueltos, y/o entrega de ejercicios extras propuestos por los docentes, y/o entrega de trabajos de desarrollo tipo informes. Los enunciados de las evaluaciones serán provistos en la plataforma Classroom y/o aulas virtuales de la materia.
A2. Aprobar el práctico de laboratorio propuesto, tanto la instancia grupal, es decir se corroborará que el programa fuente obtenga los resultados esperados, y respecto a la actividad individual, se analizará la coherencia entre la implementación y la explicación brindada.
A3. Aprobar el práctico de investigación propuesto, en base al desarrollo del informe y de la socialización en clase.
A4. Aprobar un examen parcial de temas prácticos vistos en la materia o en su defecto, alguna de sus respectivas recuperaciones con al menos el 70% correcto del total y no menos del 50% de cada ejercicio. Tal como lo establece la reglamentación vigente, Ord. 32/14 CS, cada examen parcial tendrá dos (2) recuperaciones.
A5. Tener un mínimo de 70% de asistencia a las clases prácticas y teóricas.

Nota: las evaluaciones correspondientes a los ítems A1, A2, A3 tendrán una nota ponderada, en función a observar que: el/la estudiante ha adquirido los conocimientos desarrollados, ha cumplimentado con las entregas en las fechas establecidas, a su nivel de participación en clase, como así también al desempeño en los grupos de trabajo.

B. Para promocionar la asignatura el/la estudiante deberá:

B1. Cumplir con los requerimientos A1, A2, A3, A4 y A5 enunciados anteriormente.
B2. Aprobar con un mínimo de 70% un examen integrador oral y/o escrito al final del cuatrimestre bajo la modalidad que se comunique oportunamente (Arts. 10 y 11 de la Resol. CD 018-20). Para acceder a este examen integrador y con la finalidad de que haya un seguimiento continuo, el/la estudiante tendrá que aprobar (con al menos el 70%) los test teóricos que se solicitarán junto con las evaluaciones prácticas indicadas en A1 y A4.
B3. La nota final de la materia se computará promediando las notas obtenidas en cada uno de los puntos mencionados previamente.

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

D. El examen final puede ser oral y/o escrito.

## IX - Bibliografía Básica

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

## X - Bibliografia Complementaria

**[1]** "Lenguajes de Programación - Diseño e Implementación". Pratt, Terrence y Zelkowitz, Marvin. Tercera edición. Prentice Hall, 1999.
**[2]** "Smalltalk-80. The Language and its implementation". Goldberg, Adele y Robson, David. Addison-Wesley, 1985.
**[3]** The Java class libraries. Chan, Patrick - Lee, Rosanna y Kramer, Douglas. Addsion Wesley. Segunda Edición, 1998.
**[4]** "Piensa en Java". Eckel Bruce. Pearson Alhambra. Cuarta Edición, 2007.
**[5]** "Introduction to Java". Carlos Kavka. ICTP, 2004.
**[6]** "Introduction to Java". Hume, J.N. Patterson y Stephenson, Christine. Canada Holt Software Associates. Primera Edición, 2000.

## 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. 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

El presente programa puede presentar ajustes y/o cambios provocados por la situación epidemiológica por COVID19 u otra situación o evento no previsto. Toda modificación será acordada y comunicada con el estudiantado e informada a Secretaría Académica.
Vías de comunicación: http://analisiscomparativo.dirinfo.unsl.edu.ar/index.html
Prof. Responsable: Patricia Roggero proggero@email.unsl.edu.ar
Jefe Trabajos Prácticos: Darío Funez dgfunez@email.unsl.edu.ar

## XIV - Otros