Universidad Nacional de Rio Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales

FORMULARIO PARA LA PRESENTACIÓN DE PROGRAMAS DE ASIGNATURAS

Año Lectivo: 2026

UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE CIENCIAS EXACTAS, FÍSICO-QUÍMICAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

CARRERA/S: Analista en Computación (Cód. 12), Licenciatura en Ciencias de la Computación (Cód. 14).

PLAN DE ESTUDIOS: 2024 Versión 0 (para las 2 carreras)

ASIGNATURA: Paradigmas y Lenguajes de Programación.

CÓDIGO: 3386

MODALIDAD DE CURSADO: Presencial

DOCENTE RESPONSABLE: Dra. Valeria Bengolea, Prof. Adjunto Excl. EQUIPO

EQUIPO DOCENTE:

Prof. Sandra Angeli, Ayudante de Primera, semi-exclusiva
An. Agustín Balestra, Ayudante de Primera simple
An. Joel D'Autilio, Ayudante de Primera simple

RÉGIMEN DE LA ASIGNATURA: Cuatrimestral

UBICACIÓN EN EL PLAN DE ESTUDIO: tercer Año

RÉGIMEN DE CORRELATIVIDADES: (para cursado, según plan de estudio vigente)

Para Analista en Computación

Para cursar

Aprobada	*Regular*
3410 - Introducción a la Computación y Programación I	3381 – Organización de Computadoras

Para rendir

Aprobada
3381 – Organización de Computadoras 3410 - Introducción a la Computación y Programación I

Para Licenciatura en Ciencias de la Computación

Para cursar

Aprobada	*Regular*
3375 - Introducción a los Algoritmos	3381 – Organización de Computadoras

Para rendir

Aprobada
3381 – Organización de Computadoras 3375 - Introducción a los Algoritmos

CARÁCTER DE LA ASIGNATURA: Obligatoria

CARGA HORARIA TOTAL: 112 horas (según el plan de estudio vigente)

Teóricas:	**56 hs**	**Prácticas :**	**56hs**	**Teóricas - Prácticas:**	**…. hs**	**Laboratorio:**	**… hs**

CARGA HORARIA SEMANAL: 8 horas (según el plan de estudio vigente)

Teóricas:	**4 hs**	**Prácticas :**	**4 hs**	**Teóricas - Prácticas :**	**…. hs**	**Laboratorio:**	**… hs**

1. CONTEXTUALIZACIÓN DE LA ASIGNATURA

La asignatura **Paradigmas y Lenguajes de Programación** se dicta en el segundo cuatrimestre del tercer año de las carreras de Analista en Computación, Licenciatura en Ciencias de la Computación, y se posiciona como un espacio de integración conceptual dentro del eje de programación.

Para cursar esta asignatura, los estudiantes deben haber regularizado previamente la asignatura **Organización de las Computadoras**, lo que garantiza una formación de base tanto en los aspectos de ejecución de programas y arquitectura subyacente, como en el manejo de abstracciones, verificación de programas y distintos paradigmas de programación. Asimismo, su trayectoria incluye asignaturas como **Introducción a los Algoritmos (Licenciatura)/ Introducción a Computación y Programación I (Analista)**, donde se consolidan los fundamentos de la resolución algorítmica de problemas.

En este contexto, la asignatura propone un cambio de enfoque: pasar del uso de lenguajes como herramientas de implementación a su estudio como objetos de análisis. Se abordan los principios

que subyacen a su diseño, sus mecanismos de funcionamiento y los distintos paradigmas de programación, promoviendo una comprensión abstracta, comparativa y fundamentada.

La materia cumple un rol diferencial según la trayectoria académica del estudiante:

- Para quienes finalizan sus estudios en el título de **Analista en Computación**, constituye una instancia de cierre en la formación en el **estudio de los lenguajes de programación como tales**, consolidando criterios para su análisis, selección y uso en contextos profesionales.

- Para los estudiantes que continúan con la **Licenciatura en Ciencias de la Computación**, establece las bases conceptuales necesarias para asignaturas posteriores como Teoría de la Computación I, Compiladores e Intérpretes, y Teoría de la Computación II, en las que se profundiza el estudio formal de los lenguajes y sus límites computacionales.

De este modo, la asignatura articula saberes previos y futuros, fortaleciendo competencias clave como la abstracción, el análisis crítico, la comprensión de modelos de ejecución y la capacidad de adaptación a nuevas tecnologías, fundamentales para el desempeño profesional y académico en el área de las Ciencias de la Computación.

En cuanto a su **articulación horizontal**, la asignatura se vincula estrechamente con *Sistemas Operativos* e *Ingeniería de Software*, que se cursan en el mismo cuatrimestre. Por un lado, el estudio de los modelos de ejecución, la gestión de memoria y los mecanismos de concurrencia en los lenguajes se relaciona con los conceptos abordados en *Sistemas Operativos*. Por otro, el análisis de paradigmas, estilos de programación y criterios de selección de lenguajes aporta fundamentos para la toma de decisiones en el diseño y desarrollo de software, en consonancia con los contenidos de *Ingeniería de Software*.

2. OBJETIVOS PROPUESTOS

Lograr que el estudiante desarrolle una comprensión profunda de los conceptos fundamentales de los lenguajes de programación, abordándolos como objetos de estudio propios de las Ciencias de la Computación, y adquiera la capacidad de analizarlos críticamente y seleccionar, con fundamento técnico, el lenguaje y paradigma más adecuado en función del problema a resolver.

Asimismo, se espera que el estudiante adquiera conocimientos sobre los principales aspectos involucrados en el diseño e implementación de los lenguajes de programación, comprendiendo

los mecanismos que determinan su comportamiento y sus implicancias en el desarrollo de software.

Con el desarrollo de esta asignatura se espera que el estudiante pueda apropiarse de los conocimientos, y desarrollar habilidades y aptitudes necesarias para:

- Conocer la historia de los lenguajes de programación y su evolución.
- Reconocer diferentes paradigmas de programación y sus características distintivas.
- Identificar y analizar diferentes formas de binding, visibilidad, alcance y tiempo de vida.
- Comprender y aplicar efectivamente los mecanismos de gestión de la memoria y otros recursos.
- Comprender la noción de tipado en lenguajes de programación, sus ventajas y las diferentes formas de chequeo de tipos.
- Seleccionar lenguajes o abstracciones adecuadas a cada tipo de problema.
- Adoptar disciplinas de programación adecuada para la prevención de errores.
- Comprender en profundidad los sistemas en tiempo de ejecución de un lenguaje de programación.
- Aplicar las primitivas y los constructores básicos de la programación concurrente.
- Comprender las dificultades de la concurrencia y sus posibles soluciones.

3. EJES TEMÁTICOS ESTRUCTURANTES DE LA ASIGNATURA Y ESPECIFICACIÓN DE CONTENIDOS

3.1. Contenidos mínimos (según plan de estudio vigente)

Introducción e historia de los lenguajes de programación. Elementos de un lenguaje. Sintaxis y

semántica de lenguajes de programación. El paradigma imperativo. Valores. Expresiones y funciones. Binding. Alcance estático y dinámico. Asignación y otros constructores. Pasaje de parámetros. Constructores de control y manejo de excepciones. Encapsulamiento y tipos abstractos de datos. Gestión de la memoria: manual y automática. Sistemas de tipos. Paradigma de programación orientada a objetos. Introducción a la programación lógica. Modelo lógico y procedural. Programación funcional. Órdenes de evaluación. Uso de funciones de orden superior. Comparación de los paradigmas y lenguajes de programación. Modelos de ejecución. Introducción a la concurrencia, primitivas y constructores para la concurrencia. **3.2. Ejes temáticos o unidades**

Unidad 1: Lenguajes como herramientas de programación

Características generales de los lenguajes de programación. Elementos de un lenguaje. Especificación de lenguajes: sintaxis y semántica. Estudio formal de la sintaxis: jerarquía de Noam Chomsky. Lenguajes regulares y lenguajes libres de contexto. Expresiones regulares, autómatas finitos y gramáticas regulares. Gramáticas libres de contexto y su aplicación en la definición sintáctica de lenguajes de programación. Declaraciones, definiciones, expresiones y comandos. Mecanismos de abstracción funcional y de datos. Herramientas de programación: intérpretes, compiladores y enlazadores. Archivos objeto, bibliotecas y ejecutables.

Unidad 2: Lenguajes y modelos de programación

Paradigmas de programación. Lenguajes declarativos y lenguajes con estado. Tipos de datos y sistemas de tipos. Chequeo de tipos. Tipado fuerte y débil. Polimorfismo y nociones de tipos dependientes. Seguridad de tipos. Declaraciones, ligadura (binding) y ambientes. Manejo de excepciones.

Unidad 3: El modelo declarativo

Asignación única. Valores y tipos de datos primitivos y estructurados. Variables e identificadores. Sintaxis y semántica de un lenguaje núcleo declarativo (Oz). Adornos sintáctico y abstracciones lingüísticas. Tipado estático y dinámico. Manejo de memoria. Unificación y ligadura.

Unidad 4: Lenguajes funcionales

Paradigma Funcional. Lenguajes funcionales. El estilo de la programación funcional. Técnicas de programación funcional. Técnicas de Verificación. Polimorfismo paramétrico. Fundamentos teóricos. Cálculo Lambda. LISP. Lenguajes funcionales modernos: Haskel y ML.

Unidad 5: Programación relacional

Modelo de computación no determinístico. Sentencias *choice* y *fail*. Árboles de búsqueda. Programación lógica y lenguaje Prolog. Forma clausal. Mecanismos de inferencia y resolución. Características extra-lógicas: *cut*, aritmética, entrada/salida. El problema de la negación.

Unidad 6: El modelo con estado

Extensión del lenguaje núcleo con celdas mutables. Semántica de celdas. Punteros y referencias. Aliasing e igualdad. Asignación y efectos colaterales. Razonamiento con estado. Abstracción procedural. Sobrecarga. Efectos colaterales. Lenguajes de programación imperativos.

Unidad 7: Lenguajes imperativos

Declaraciones, expresiones y comandos. Excepciones. Introducción al lenguaje C: tipos de datos, estructuras, declaraciones y definiciones. Proceso de compilación. Funciones. Alcance y tiempo de vida de las entidades. Operadores. Sentencias de control. Tipos estructurados. Punteros. Manejo dinámico de memoria.

Unidad 8: Manejo de la Memoria

Estrategias de administración eficiente de memoria. Organización de la memoria en tiempo de ejecución: manejo del *stack* y del *heap*. Registros de activación: estructura, enlaces estáticos y dinámicos, y su relación con el alcance y la visibilidad. Representación de estructuras de datos en memoria: cálculo de direcciones en arreglos y matrices. Valores creados dinámicamente. Manejo del *heap*: asignación y liberación de memoria. Manejo manual de memoria: problemas asociados, tales como referencias colgadas y generación de memoria no recuperada

(fugas). Manejo automático de memoria: recolección de basura. Estrategias de recolección de basura. Análisis experimental del comportamiento de la memoria en distintos lenguajes. Uso de herramientas de monitoreo y análisis (por ejemplo, entornos de ejecución de Java) para la observación del uso del *heap* y la actividad del recolector de basura.

Unidad 9: Programación orientada a objetos

Objetos y clases. Clases como módulos y tipos. Herencia. Sistemas de tipos en programación orientada a objetos. Control de acceso. Polimorfismo por herencia. Ligadura dinámica. Redefinición de métodos. Clases abstractas e interfaces. Herencia múltiple. Implementación de la ligadura dinámica. Polimorfismo. Tipos parametrizados y genericidad. Plantillas (templates) en C++ y genéricos en Java. Programación genérica y computación estática.

Unidad 10: Concurrencia

Modelos de memoria compartida y mensajes. Threads, eventos, corrutinas, procesos secuenciales y concurrentes. Problemas que plantea la concurrencia: no determinismo, dependencia de velocidad, bloqueos (deadlocks). Progreso finito (starvation). Interacción entre procesos. Procesos independientes, competitivos y comunicantes. Primitivas para la creación y destrucción de procesos y threads. Primitivas de bajo nivel de sincronización para exclusión mutua, semáforos. Constructores para
el manejo de concurrencia de alto nivel: monitores y rendezvous. Ejemplos en Java y C++ . Con currencia en Erlang. Mecanismos de comunicación entre procesos: tuberías (pipes), colas de mensajes, FIFOs.

4. ACTIVIDADES A DESARROLLAR

La asignatura se desarrollará mediante la articulación de clases teóricas y prácticas, con fuerte énfasis en la experimentación y el análisis comparativo de lenguajes de programación desde distintas perspectivas conceptuales y de implementación.

En las clases teóricas se introducirán los conceptos fundamentales de los lenguajes de programación, abordando tanto los aspectos formales (sintaxis, semántica, sistemas de tipos) como los distintos paradigmas de programación y modelos de ejecución. Estos contenidos serán desarrollados mediante ejemplos concretos y experimentación con lenguajes de programación, favoreciendo la integración entre teoría y práctica y promoviendo una comprensión operativa de los conceptos.

Las clases prácticas se desarrollarán en modalidad de laboratorio, donde los estudiantes resolverán ejercicios orientados al análisis, comparación y experimentación con distintos lenguajes de programación. Estas actividades incluirán la implementación de pequeños programas, el estudio de ejemplos representativos de distintos paradigmas (imperativo, funcional, lógico y orientado a objetos), y la realización de experimentos que permitan observar el comportamiento de los lenguajes en aspectos tales como manejo de memoria, sistemas de tipos, alcance y concurrencia.

Se promoverá el uso de múltiples lenguajes de programación (por ejemplo, C/C++, Java, lenguajes funcionales y lógicos), con el objetivo de que los estudiantes puedan contrastar diferentes modelos de programación y comprender las decisiones de diseño subyacentes. En algunos casos, se propondrá la resolución de un mismo problema en distintos paradigmas, con el fin de analizar sus diferencias en términos de expresividad, claridad, eficiencia y mantenibilidad.

Asimismo, se incorporarán actividades orientadas a la comprensión de los mecanismos de implementación de los lenguajes, tales como el análisis de la pila de ejecución, cálculo de direcciones en estructuras de datos, estudio de registros de activación, y experimentación con técnicas de gestión de memoria tanto manual como automática. Se incluirá el uso de herramientas de análisis y monitoreo que permitan observar el comportamiento dinámico de los programas.

Se fomentará la resolución de problemas que involucren concurrencia y comunicación entre procesos, promoviendo la implementación y análisis de distintos modelos (memoria compartida y paso de mensajes), así como la identificación de problemáticas clásicas como condiciones de

carrera, interbloqueos y sincronización.

Se incentivará la lectura de bibliografía complementaria y la exploración autónoma de herramientas y lenguajes, promoviendo el desarrollo de habilidades de autoaprendizaje fundamentales en el área. Asimismo, se fomentará el trabajo colaborativo en la resolución de actividades prácticas.

CLASES TEÓRICAS: 4 horas semanales, organizadas en dos encuentros, con desarrollo integrado de contenidos conceptuales y su aplicación práctica mediante ejemplos y experimentación.

CLASES PRÁCTICAS: 4 horas semanales, desarrolladas en laboratorio y organizadas en dos encuentros.

5. PROGRAMAS Y/O PROYECTOS PEDAGÓGICOS INNOVADORES E INCLUSIVOS

No existen actividades de la asignatura enmarcadas en programas y/o proyectos pedagógicos.

6. CRONOGRAMA TENTATIVO DE CLASES E INSTANCIAS EVALUATIVAS

INCORPORE AQUÍ EL TEXTO

Que muestre coherencia y consistencia con el logro de los objetivos y las competencias definidas. Las fechas de parciales deberán ser consensuadas con los responsables de las demás asignaturas del cuatrimestre correspondiente, en acuerdo con la Res. C.S. 120/17).

Semanas	días	Actividad Teórica	días	Actividad Práctica	Evaluaciones
1	9 al 13 de marzo	Introducción/Características de los lenguajes/Historia	9 al 13 de marzo	No hay clases prácticas	

Universidad Nacional de Rio Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales

		Introducción al estudio formal de Sintaxis. Ejemplos Prácticos			
2	16 al 20 de marzo	Estudio formal de Sintaxis (Continuación) Herramientas de programación: intérpretes, compiladores y enlazadores Ejemplos Prácticos	16 al 20 de marzo	**Práctico 1**: Autómatas finitos, expresiones regulares, uso de grep, gramáticas Regulares y Libres de contexto. EBNF	
3	23 al 27 de marzo	lunes 23 de marzo feriado Paradigmas de programación. Lenguajes declarativos y lenguajes con estado. Tipos de datos y sistemas de tipos. Chequeo de tipos. Tipado fuerte y débil. Polimorfismo y nociones de tipos dependientes. Seguridad de tipos. Declaraciones, ligadura (binding) y ambientes. Manejo de excepciones.	23 al 27 de marzo	lunes 23 y martes 24 de marzo feriado **Práctico 1**: Herramientas de compilación. Creación de bibliotecas está ticas y dinámicas. Linkeo de programas. Archivos objetos.	
4	30 de marzo al 3 de abril	viernes 3 abril feriado Introducción al Lenguaje Oz. Ejemplos prácticos Modelo Declarativo: asignación única, sintaxis y semántica, unificación y ligadura.	30 de marzo al 3 de abri	jueves 2 y viernes 3 abril feriado **Práctico 2**: Ejercicios sobre sistemas de tipos, alcance y ambientes, con análisis comparativo y experimentación en distintos lenguajes.	

5	6 al 10 de abril	Modelo Declarativo (Continuación). Fundamentos de programación funcional: cálculo lambda. Ejemplos prácticos	6 al 10 de abril	**Práctico 3:** Ejercicios en Oz sobre modelo declarativo: unificación, ligadura y ejecución paso a paso en la máquina abstracta. Análisis de alcance y recursión.	
6	13 al 17 de abril	Paradigma Funcional. Lenguajes funcionales. El estilo de la programación funcional. Técnicas de programación funcional. Técnicas de Verificación. Polimorfismo paramétrico. Ejemplos	13 al 17 de abril	**Práctico 3**: continuación **Práctico 4:** Ejercicios sobre cálculo lambda (reducción y estrategias de evaluación) y programación funcional en Haskell y LISP, incluyendo funciones de orden superior y estructuras de datos.	
7	20 al 24 de abril	Modelo de programación lógica: resolución, unificación, no determinismo y árboles de búsqueda. Ejemplos prácticos Lenguaje Prolog: sintaxis, forma clausal y uso de mecanismos de inferencia Ejemplos prácticos	20 al 24 de abril	**Práctico 4**: continuación **Práctico 5:** Ejercicios sobre el modelo Lógico	

8	27 de abril al 1 de mayo	Programación Lógica Continuación Viernes 1 de mayo: Feriado	27 de abril al 1 de mayo	**Práctico 5:** Ejercicios sobre el modelo Lógico, (continuación) PROLOG	
9	4 al 8 de mayo	-El modelo con estado: Extensión del lenguaje núcleo con celdas mutables. Razonamiento con estado. Abstracción procedural. Sobrecarga. -Lenguajes de programación imperativos.	4 al 8 de mayo	**Práctico 6**:Modelo con estado, Oz y la extensión del Lenguaje Núcleo , máquina abstracta y Lenguajes imperativos	
10	11 al 15 de mayo	Introducción al lenguaje C	11 al 15 de mayo	Práctico 6: Continuación Práctico 7: El Lenguaje C	
11	18 al 22 de mayo	Manejo de la Memoria: Manejo de stack: Registros de Activación. Manejo de Heap: Manejo manual y Manejo automático de memoria	18 al 22 de mayo	Práctico 8: -Manejo del Stack, ejercicio registro de activación. Manejo de Heap: ejercicio Manejo manual y automática de memoria.	
12	25 al 29 de mayo	Lunes 25 de mayo: feriado Lenguajes Orientados a Objetos	25 al 29 de mayo	Lunes 25 de mayo: feriado Práctico 9: Lenguaje orientado objetos: Java y C++	

7. BIBLIOGRAFÍA

7.1. Bibliografía obligatoria y de consulta (por lo menos algún material bibliográfico debe ser de edición 2012 o posterior).

Concepts, Techniques and Models of Computer Programming. Van Roy and Haridi. The MIT Press. ISBN: 0-262-22069-5. 2004

◦ Programming Language Design and Implementation (Third Edition). Terrence Pratt, Marvin Zelkowitz. Prentice Hall. 1996. ISBN: 0-13-678012-1

Concepts of programming languages, 12 edition Robert Sebesta, 2024. **ISBN-13 (versión impresa):** 978-0134997186

◦ Programming Language Concepts. Carlo Guezzi, Medhi Jazayeri. 3rd edition. ISBN:0-471-10426-4.

◦ Apunte de la Materia, elaborado por los docentes de la Asignatura Análisis Comparativo de Lenguajes y modificado y actualizado todos los años.

7.2. Otros: materiales audiovisuales, enlaces, otros

8. DÍA Y HORARIOS DE CLASES

Teóricos:

Lunes de 18 a 20hs.

Viernes de 14 a 16hs

Comisión 1:

lunes de 8 a 10 hs y miércoles de 8 a 10 hs.

Comisión 2:

lunes de 14 a 16 hs y martes 16 a 18 hs

Comisión 3:

miércoles 16 a 18 y viernes de 12 a 14 h

9. DÍA Y HORARIO DE CLASES DE CONSULTAS

Martes 15hs

Miercoles 11hs

10. REQUISITOS PARA OBTENER LA REGULARIDAD Y LA PROMOCIÓN

CONDICIONES DE REGULARIDAD: Las condiciones para la regularidad de la asignatura son las siguientes: Aprobación de los dos exámenes parciales o sus respectivos recuperatorios (nota mayor o igual a 5).

CONDICIONES DE PROMOCIÓN: Para acceder a la promoción, se requiere la aprobación de ambos exámenes parciales con una calificación igual o superior a 7. Los estudiantes que cumplan con esta condición deberán rendir un coloquio, que consistirá en una presentación oral centrada en un lenguaje de programación y sus principales características. La calificación final de promoción se determinará mediante un promedio ponderado entre las notas de los exámenes parciales y la del coloquio.

11. CARACTERÍSTICAS, MODALIDAD Y CRITERIOS DE LAS INSTANCIAS EVALUATIVAS

Evaluaciones Parciales:

Se tomarán dos exámenes parciales, cada uno con su correspondiente instancia de recuperación. Estas evaluaciones estarán compuestas por ejercicios prácticos similares a los desarrollados en las clases prácticas y por preguntas teóricas orientadas a la integración y relación de los distintos conceptos abordados.

Evaluación Final:

Alumnos regulares: deberán rendir un examen oral centrado en los conceptos teóricos de la asignatura.

Alumnos libres: en una primera instancia deberán aprobar una evaluación de carácter práctico. Una vez superada esta etapa, accederán a una instancia de evaluación oral equivalente a la de los alumnos regulares.

Universidad Nacional de Rio Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales

Firma Profesor/a Responsable Firma Secretario/a Académico/a