RESOLUCIÓN 25/18/110-00
ACTA 1226/25/08/2025

"POR LA CUAL SE APRUEBA EL PROGRAMA DE ESTUDIO DE LA ASIGNATURA *ESTRUCTURA DE LENGUAJES DE PROGRAMACIÓN,* DE CARRERAS DE GRADO, SEDE SAN LORENZO FILIAL VILLARRICA Y FILIAL CORONEL OVIEDO"

VISTO: El Memorando DA/1707/2025 del Director Académico de la FP-UNA, Prof. MSc. Felipe Santiago Uzabal Escurra, con el cual remite el Memorando CCPTCC/030/2025 de la Comisión Coordinadora del Proyecto de Transformación Curricular de Carreras de Grado de la FP-UNA, en el que presenta la propuesta de Programa de Estudio de Asignatura de las Carreras de Grado.

CONSIDERANDO: La Ley 4995/2013 de Educación Superior, el Estatuto de la Universidad Nacional de Asunción y las deliberaciones sobre el tema.

Que la Comisión Coordinadora del Proyecto de Transformación Curricular de Carreras de Grado, solicita la aprobación del Programa de Estudio de la asignatura **"Estructura de Lenguajes de Programación"**, la cual es común entre Carreras de Grado.

Que los programas fueron elaborados conforme a las disposiciones establecidas por el Consejo Nacional de Educación Superior (CONES) en materia de **créditos académicos**, según lo dispuesto en la Resolución CONES N.° 221/2024, que regula el *Sistema de Créditos Académicos – Paraguay* y los criterios para su publicación en las carreras de grado.

EL CONSEJO DIRECTIVO DE LA FACULTAD POLITÉCNICA
RESUELVE:

25/18/110-01 APROBAR el Programa de Estudio de la Asignatura **"Estructura de Lenguajes de Programación"**, de carreras de Grado, Sede San Lorenzo, Filial Villarrica y Filial Coronel Oviedo**,** detallado en el ANEXO 72 de la presente Acta.

25/18/110-02 COMUNICAR, copiar y archivar.

Prof. Abg. Joel Arsenio Benítez Santacruz
Secretario

Prof. Ing. Silvia Teresa Leiva León, MSc.
Presidenta

TEL/FAX: +595 21 5887000 – C.C. 1130 (Asunción) – 2111 (San Lorenzo)
www.pol.una.py

DEPARTAMENTO DE ENSEÑANZA DE INFORMÁTICA
PROGRAMA DE ESTUDIO

I. IDENTIFICACIÓN

Nivel	Grado				
Asignatura	Estructura de Lenguajes de Programación				
Carrera	**Plan**	**Sede/Filial**	**Carácter**	**Semestre**	**Prerrequisitos**
Ingeniería en Informática	2026	Sede San Lorenzo	Obligatoria	Sexto	Lenguajes de Programación 3.
Licenciatura en Ciencias Informáticas	2026	Sede San Lorenzo/Filial Villarrica/Filial Coronel Oviedo	Electiva	***	Ingeniería de Software I, Base de Datos II, Seguridad Informática, Programación Frontend.

Semanal					**Periodo**				
HT	**HP**	**HTD**	**HTI**	**HS**	**PA**	**THTD**	**THTI**	**THA**	**CA-PY**
2	2	4	4	8	18	72	72	144	5

***HT: Horas Teóricas semanales.**
***HP: Horas Prácticas semanales.**
***HTD: Horas semanales de Trabajo académico con acompañamiento Docente.**
***HTI: Horas semanales de Trabajo académico Independiente del estudiante.**
***HS: Horas Semanales (HTD+HTI).**
***PA: Periodo Académico en semanas.**
*** THTD: Total de Horas de Trabajo académico con acompañamiento Docente (HTD*PA).**
*** THTI: Total de Horas de Trabajo académico Independiente del estudiante (HTI*PA).**
*** THA: Total de Horas de trabajo Académico (THTD+THTI).**
*** CA-PY: Créditos académicos de la asignatura.**

II. FUNDAMENTACIÓN

La inclusión de la asignatura Estructura de Lenguajes de Programación dentro de la malla curricular es esencial debido a que la programación es una habilidad fundamental para los profesionales informáticos y para realizar esta actividad de manera adecuada, es esencial entender los conceptos fundamentales detrás de los lenguajes que se utilizan para codificar. Esta asignatura brinda a los estudiantes una comprensión profunda de cómo funcionan los lenguajes de programación, su evolución, su diseño, y cómo se relacionan con las máquinas que ejecutan ese código. Al comprender estos conceptos, los estudiantes estarán mejor preparados para seleccionar el lenguaje adecuado para una tarea particular, aprender nuevos lenguajes con mayor facilidad y adaptarse a la evolución del campo de la informática.

La asignatura es de naturaleza teórico-práctica. Se enfatiza en la teoría de lenguajes formales y cómo se aplica en la creación y comprensión de lenguajes de programación; a la par, los estudiantes tendrán oportunidades prácticas para experimentar con diferentes lenguajes y paradigmas, consolidando su aprendizaje.

Este curso contribuirá significativamente al desarrollo del perfil profesional del estudiante proporcionando las herramientas teóricas y prácticas necesarias para analizar, diseñar e implementar soluciones informáticas efectivas.

La asignatura está organizada en una serie de unidades que abordan diferentes aspectos de los lenguajes de programación en función a los ejes temáticos, como sigue:

1. Introducción y fundamentos: Establece la base para el estudio de los lenguajes de programación.
2. Descripción de la sintaxis: Introduce a los estudiantes en cómo se define y se interpreta la estructura de un lenguaje.
3. Elementos básicos y estructurales: Examina los componentes clave de cualquier lenguaje.
4. Programación orientada a objetos: Profundiza en uno de los paradigmas de programación más populares y versátiles.
5. Paradigmas de programación funcional y lógico: Expone a los estudiantes a diferentes maneras de pensar sobre la programación.

III. COMPETENCIAS DEL PERFIL DE EGRESO ASOCIADAS

1. Aplicar en la práctica profesional los valores humanos, la ética y los mecanismos de seguridad laboral.
2. Actuar proactivamente frente a los problemas sociales y ambientales.
3. Adaptarse respetuosamente a contextos nuevos o adversos, así como a diversidades personales, disciplinares y culturales.
4. Actualizarse permanentemente mediante la obtención y gestión autónoma de información de calidad, utilizando tecnología de la información y comunicación.
5. Evaluar el comportamiento de diversos fenómenos disciplinares e interdisciplinares relacionados con la informática con una visión de sistema, mediante modelos teóricos validados y actualizados, capaces de abarcarlos integralmente en un contexto de incertidumbre.
6. Planificar, proyectar, diseñar y ejecutar proyectos sostenibles e integrales para la resolución de problemas, la mejora y la innovación en el ámbito de la informática.

IV. ORGANIZACIÓN DE LA ASIGNATURA

Unidades	Contenidos	Resultados de aprendizaje
1. Introducción y fundamentos	1.1. Razones para estudiar conceptos de lenguajes de programación. 1.2. Criterios de evaluación de lenguajes. 1.3. Paradigmas de lenguajes de programación. 1.4. Métodos de implementación. 1.5. Desarrollo y evolución de los lenguajes de alto nivel. 1.6. Breve historia de los lenguajes de programación.	1. Justifica la relevancia del estudio de lenguajes de programación en el contexto tecnológico actual, destacando su importancia en el desarrollo de soluciones innovadoras. 2. Evalúa lenguajes de programación según criterios establecidos, argumentando su adecuación en diferentes contextos y aplicaciones específicas. 3. Analiza escenarios de programación para identificar las ventajas y limitaciones de diferentes paradigmas y lenguajes de programación. 4. Compara métodos de implementación de lenguajes de programación, evaluando su

Unidades	Contenidos	Resultados de aprendizaje
		impacto en la eficiencia, funcionalidad y mantenibilidad del código. 5. Resume la evolución histórica de los lenguajes de programación de alto nivel, identificando tendencias actuales y posibles desarrollos futuros. 6. Evalúa el impacto histórico de lenguajes de programación clave en los avances tecnológicos y su influencia en el desarrollo de nuevos lenguajes.
2. Descripción de la sintaxis de los lenguajes de programación.	2.1. Expresiones regulares y autómatas. 2.1.1. Conceptos básicos. 2.1.2. Introducción a las expresiones regulares: definiciones básicas y operaciones. 2.1.3. Autómatas finitos no deterministas (NFA) y su relación con expresiones regulares. 2.1.4. Autómatas finitos deterministas (DFA) y la conversión desde NFA. 2.1.5. Aplicaciones prácticas: Reconocimiento de patrones y tokenización. 2.2. Gramáticas formales y sintaxis. 2.2.1. Símbolos terminales y no terminales, producciones y derivaciones. 2.2.2. Definición y uso de las gramáticas. 2.2.3. Ambigüedad en las gramáticas y árboles de derivación. 2.3. Análisis Léxico y Parsers 2.3.1. Proceso general del análisis léxico. 2.3.2. Parsers ascendentes y descendentes. 2.3.3. Herramientas populares: Lex, Yacc, ANTLR.	1. Identifica y explica los conceptos básicos de la teoría de lenguajes en el contexto de lenguajes de programación. 2. Diseña y aplica expresiones regulares para reconocer patrones específicos en textos y procesos computacionales. 3. Analiza la relación entre expresiones regulares y autómatas finitos no deterministas (NFA) y convierte NFA a autómatas finitos deterministas (DFA). 4. Aplica técnicas de reconocimiento de patrones y tokenización mediante expresiones regulares y autómatas. 5. Evalúa la importancia de las descripciones formales de lenguajes para el diseño de gramáticas y estructuras sintácticas. 6. Diferencia entre símbolos terminales y no terminales, producciones y derivaciones, y aplica estos conceptos en gramáticas formales. 7. Diseña gramáticas formales para describir estructuras sintácticas y resuelve problemas de ambigüedad mediante árboles de derivación. 8. Explica el rol del análisis léxico en la compilación y describe su proceso general. 9. Compara parsers ascendentes y

Unidades	Contenidos	Resultados de aprendizaje
		descendentes, analizando sus características y aplicaciones. 10. Utiliza herramientas como Lex y Yacc para realizar análisis léxico y sintáctico en proyectos de software.
3. Elementos básicos y estructurales de los lenguajes de programación.	3.1. Nombres, enlaces, tipo y alcance. 3.1.1. Nomenclatura y significado en lenguajes de programación. 3.1.2. Relaciones y enlazamientos entre variables y valores. 3.1.3. Definición y uso de tipos en programación. 3.1.4. Alcance y visibilidad de las variables. 3.1.5. Criterios de diseño para nomenclatura y alcance. 3.2. Tipos de datos. 3.2.1. Tipos de datos primitivos. 3.2.2. Tipos de datos compuestos. 3.2.3. Criterios de diseño para tipos de datos compuestos. 3.2.4. Sistema de tipos. 3.3. Estructuras de control. 3.3.1. Diferenciación y aplicación de expresiones y sentencias. 3.3.2. Mecanismos de control de flujo: iteraciones, decisiones, etc. 3.3.3. Criterios de diseño para estructuras de control. 3.4. Excepciones y manejo de errores. 3.4.1. Definición de excepciones y su importancia. 3.4.2. Métodos y técnicas para el manejo efectivo de errores. 3.4.3. Criterios de diseño para el manejo de excepciones y errores. 3.5. Subprogramas. 3.5.1. Características generales de los subprogramas. 3.5.2. Mecanismos de paso de	1. Evalúa las técnicas y estrategias de alcance, visibilidad y nomenclatura en diferentes lenguajes de programación, considerando su impacto en la claridad y la mantenibilidad del código. 2. Analiza los sistemas de tipos de datos en lenguajes de programación, evaluando sus características, criterios de diseño y su impacto en la robustez del código. 3. Compara y analiza estructuras de control en distintos lenguajes de programación, evaluando sus criterios de diseño y su aplicabilidad en diferentes contextos. 4. Evalúa los mecanismos de manejo de errores y excepciones en lenguajes de programación, analizando sus criterios de diseño y su relevancia en el desarrollo de software confiable. 5. Analiza y compara los mecanismos de paso de parámetros en subprogramas, evaluando su impacto en la flexibilidad y eficiencia del código. 6. Describe los ambientes de tiempo de ejecución de los lenguajes de programación, explicando cómo influyen en la ejecución de subprogramas. 7. Analiza y evalúa los criterios de diseño de subprogramas en distintos lenguajes, considerando aspectos como paso de parámetros, visibilidad y flexibilidad. 8. Describe las corrutinas y analiza su funcionamiento, comparándolas

Unidades	Contenidos	Resultados de aprendizaje
	parámetros. 3.5.3. Ambientes de ejecución de subprogramas. 3.5.4. Corrutinas: definición y usos. 3.5.5. Criterios de diseño para subprogramas.	con funciones tradicionales y evaluando su utilidad en diversos lenguajes de programación.
4. Soporte a la Programación Orientada a Objetos (POO) en lenguajes de programación.	4.1. Soporte a la definición y manipulación de clases y objetos. 4.2. Mecanismos de herencia y cómo los lenguajes los facilitan. 4.3. Implementación de polimorfismo: sobrecarga y ligadura dinámica. 4.4. Soporte para encapsulación: acceso y modificación de datos 4.5. Abstracción y mecanismos de extensión de clases. 4.6. Diseño y criterios de evaluación de soporte a la POO en lenguajes de programación.	1. Identifica y analiza las capacidades de los lenguajes de programación para definir y manipular clases y objetos, evaluando su impacto en el diseño de aplicaciones orientadas a objetos. 2. Examina cómo los lenguajes de programación implementan los principios de la orientación a objetos, incluyendo herencia, encapsulación y polimorfismo, para resolver problemas complejos. 3. Compara y diferencia las diversas formas de polimorfismo, como sobrecarga y ligadura dinámica, evaluando las técnicas de implementación en distintos lenguajes de programación. 4. Evalúa el soporte a la programación orientada a objetos en lenguajes de programación específicos, aplicando criterios establecidos para determinar su idoneidad en diversos contextos de desarrollo.
5. Paradigmas de programación funcional y lógico.	5.1. Funciones como ciudadanos de primera clase. 5.2. Expresiones lambda y clausuras 5.3. Recursión en la programación funcional. 5.4. Constructores de listas y operaciones de alto orden en programación funcional. 5.5. Fundamentos de la programación lógica. 5.6. Clausura y reglas en la programación lógica. 5.7. Mecanismos de inferencia	1. Identifica las características de las funciones como ciudadanos de primera clase y describe el soporte que brindan diferentes lenguajes de programación. 2. Analiza y utiliza expresiones lambda y clausuras para resolver problemas específicos y mejorar la eficiencia del código. 3. Diseña soluciones eficientes mediante el uso de constructores de listas y funciones de alto orden en programación funcional. 4. Explica los fundamentos del paradigma lógico, incluyendo reglas de clausura y mecanismos

Unidades	Contenidos	Resultados de aprendizaje
	y backtracking. 5.8. Comparación entre paradigmas funcional y lógico. 5.9. Diseño y criterios de evaluación de soporte a los paradigmas funcional y lógico en lenguajes de programación.	para modelar problemas en lenguajes lógicos. 5. Aplica técnicas de inferencia y backtracking para resolver problemas prácticos en programación lógica. 6. Compara las ventajas y limitaciones de los paradigmas funcional y lógico, identificando contextos en los que cada uno es más adecuado. 7. Evalúa las características de los lenguajes funcionales y lógicos en relación con su soporte a estos paradigmas y su uso en el desarrollo de aplicaciones. 8. Diseña soluciones utilizando lenguajes funcionales y lógicos, seleccionando el más adecuado según los requerimientos del problema.

V. ESTRATEGIAS DIDÁCTICAS

En el desarrollo de este curso, se aplicarán estrategias didácticas que fomenten tanto la comprensión teórica como la aplicación práctica de los conceptos clave, promoviendo el pensamiento crítico y la participación activa de los estudiantes. Estas estrategias son:

- **Clases magistrales:** El docente expondrá los conceptos fundamentales de cada unidad, utilizando materiales de apoyo como diapositivas, lecturas recomendadas y ejemplos prácticos de código en distintos lenguajes de programación. Durante estas sesiones, se estimulará el debate y la reflexión a través de preguntas relacionadas con los criterios de diseño y evaluación de los lenguajes.

- **Aula invertida:** Se proporcionará a los estudiantes acceso a materiales didácticos, como videos explicativos, lecturas y tutoriales de herramientas como Lex y Yacc, que podrán revisar antes de las clases. Esto permitirá dedicar las sesiones presenciales a discutir aplicaciones prácticas y resolver dudas específicas.

- **Aprendizaje Basado en Problemas (ABP):** Los estudiantes resolverán casos prácticos donde deban aplicar los conceptos de diseño de lenguajes de programación para proponer soluciones, utilizando herramientas y lenguajes apropiados.

- **Trabajos colaborativos:** Los estudiantes trabajarán en equipos para desarrollar proyectos que integren conceptos de diseño y evaluación de lenguajes, fomentando la cooperación, asignación de roles y el cumplimiento de objetivos en grupo.

- **Talleres prácticos:** En sesiones de laboratorio, los estudiantes implementarán soluciones utilizando diferentes lenguajes de programación, poniendo en práctica conceptos de sintaxis, gramáticas y paradigmas. Estas actividades se centrarán en tareas específicas guiadas por el docente.

- **Simulación de evaluación de lenguajes:** Los estudiantes asumirán el rol de evaluadores de lenguajes de programación, aplicando criterios como eficiencia, legibilidad y facilidad de uso para proponer recomendaciones en un contexto dado.

- **Proyecto Final Integrador:** Los estudiantes desarrollarán un proyecto que integre varios conceptos aprendidos durante el curso, eligiendo un lenguaje de programación y justificando su elección en función de un problema específico.

La elección particular de la estrategia didáctica aplicada será explícita en el Planeamiento de la Asignatura, de acuerdo con el perfil de los estudiantes, los recursos disponibles y el contexto educativo.

VI. ESTRATEGIAS EVALUATIVAS

Procesos de producción grupales e individuales, pruebas individuales orales y/o escritas durante el desarrollo de las unidades, desarrollo e implementación de programas que reflejen conceptos teóricos y prácticos de los lenguajes de programación, análisis y evaluación crítica de lenguajes a través de informes escritos, y la participación en simulaciones y talleres prácticos sobre gramáticas, parsers, y paradigmas de programación. Además, se realizará un trabajo práctico que involucre la programación en varios lenguajes de programación y la redacción de un documento donde se comparen las estructuras principales de los lenguajes utilizados forma tal a abordar varios de los resultados de aprendizaje mencionados anteriormente.

Con fines de calificación y promoción se aplicará el Reglamento Académico vigente en la institución que prevé valoraciones de proceso y final.

VII. MEDIOS AUXILIARES

Pizarras, marcadores, computadoras, proyector, parlantes para multimedia, aula virtual.

VIII. BIBLIOGRAFÍA

- Sebesta, R. W. (2022). Concepts of ProgrammingLanguages (12 ed.). Pearson.
- Sebesta, R. W. (2016). Concepts of ProgrammingLanguages (11 ed.). Pearson.
- Louden, K. C. (2004). Construcción de compiladores (2ed.). International Thomson Editores.
- Páginas web de lenguajes utilizados en el semestre de acuerdo con el plan de clases.
- Sipser, M. (2012), Introduction to theTheory of Computation 3rd Edición. CengageLearning.
- Nystrom, R. (2021). CraftingInterpreters. Geneverbenning.
- Franek, F. (2004). Memory as a Programming Concept in C and C++.
- Memory as a Programming Concept in C and C++. Cambridge UniversityPress
- Louden, K. C. (1993). Programminglanguages: Principles and practice. WadsworthPubl. Co.