UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:INTRODUCCIÓN A LA PROGRAMACIÓN
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 02, LABORATORIO: 04)

SUMILLA

El curso es de naturaleza teórico-práctica y enseña a escribir programas que no caben en la cabeza: a descomponer un problema difícil, a esconder detalles detrás de una buena interfaz y a comprobar que el resultado es correcto. Se organiza en cuatro partes: **repaso y entornos** (lo esencial de C, lectura de código y modelo de entornos); **memoria y abstracción** (punteros, memoria dinámica, módulos y tipos); **recursión** (recursión, recursión en árbol, backtracking y funciones como valores); y **un intérprete** (datos recursivos, análisis sintáctico y evaluación). El laboratorio se desarrolla en C, en un entorno común.

COMPETENCIAS

- Explica qué pasa en la memoria al ejecutar un programa, con marcos de llamada, alcance y tiempo de vida.
- Usa punteros y memoria dinámica sin fugas ni accesos inválidos, y lo comprueba con herramientas de diagnóstico.
- Divide un programa en módulos con una interfaz clara y define tipos adecuados al problema.
- Resuelve problemas con recursión, recursión en árbol y backtracking, y razona por qué sus programas son correctos y terminan.
- Pasa funciones como valores con punteros a función.
- Prueba y depura sus programas con método, y lee y extiende código ajeno.
- Construye un intérprete pequeño: análisis léxico, análisis sintáctico y evaluación recursiva.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas
I	Repaso y entornos	1–3
II	Memoria y abstracción	4–8
III	Recursión	9–12
IV	Un intérprete	13–16

PROGRAMACIÓN SEMANAL

Cada semana de clases comprende una sesión de teoría de 2 horas y dos sesiones de laboratorio de 2 horas cada una. El capítulo de los apuntes de cada semana es lectura previa. Los temas marcados con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Teoría	Laboratorio
1	Presentación. Repaso: tipos, control, funciones y `assert`	Entorno y programas de repaso
2	Repaso: arreglos, cadenas, `struct` y archivos. Cómo leer código	Leer un programa ajeno y extenderlo
3	El modelo de entornos: marcos, alcance y tiempo de vida	Imágenes: filtros y marcos en `gdb`
4	Punteros: paso por puntero, arreglos y punteros, `const`	Transformar imágenes con punteros. *Práctica 1*
5	Memoria dinámica: `malloc`, `free`, ownership, ASan y valgrind	Imágenes de cualquier tamaño y cacería de fugas
6	Varios archivos: encabezados, `make`, tipos opacos y pruebas por la interfaz	Dividir un programa en módulos con sus pruebas
7	Tipos: `typedef`, `enum`, uniones etiquetadas. Conversiones (*). Comportamiento indefinido	Un valor con varios tipos. *Práctica 2*
8	**Examen parcial**	
9	Recursión: caso base, razonar por inducción, recursión e iteración	Recursión en arreglos y cadenas
10	Recursión en árbol: flood fill, cuadrículas. Memoización (*)	Buscaminas
11	Backtracking: permutaciones, subconjuntos, N reinas y poda	Resolver sudokus. *Práctica 3*
12	Funciones como valores: punteros a función, callbacks, `qsort` y contexto `void *`	Un solucionador genérico
13	Datos recursivos: árboles de expresión, evaluar. Simplificar (*)	Álgebra de expresiones
14	Análisis léxico y análisis sintáctico descendente recursivo	Parser del lenguaje Mini. *Práctica 4*
15	El intérprete: variables, funciones y marcos. Repaso	Terminar y probar el intérprete
16	**Examen final**	

METODOLOGÍA

- **Teoría:** clases expositivas con ejercicios resueltos en clase y programas seguidos paso a paso. El estudiante lee el capítulo de los apuntes antes de cada semana.
- **Laboratorio:** dos sesiones semanales en un entorno común para todos los estudiantes, donde leen código dado, lo extienden, lo prueban y lo depuran.
- **Prácticas:** cuatro prácticas individuales, en las semanas 4, 7, 11 y 14, con pruebas automáticas. Esa semana, una de las dos sesiones de laboratorio es para la práctica y la otra sigue con el laboratorio de la guía.
- Los términos técnicos se presentan en español cuando así se dicen (variable, arreglo, función) y en inglés cuando la traducción suena forzada (bit, commit, router).

NORMAS DEL CURSO

- **Trabajo:** individual. Se puede discutir ideas, pero cada estudiante escribe su propio código y debe poder explicarlo. Copiar soluciones de otras personas o de internet cuenta como plagio.

Herramientas e inteligencia artificial

Las herramientas del entorno del curso se usan todo el tiempo. La IA generativa no se usa en ningún trabajo del curso: laboratorios, prácticas ni exámenes.

Herramienta	¿Se permite?
Diagnósticos del compilador (`gcc -Wall -Wextra`), sanitizers (ASan, UBSan), valgrind y `gdb`	Sí. Son parte del curso
Documentación oficial (páginas `man`, cppreference, manuales de `gcc` y `make`) y búsqueda web	Sí, para consultar cómo funciona algo. No para buscar la solución de un ejercicio
Autocompletado del editor que completa nombres, firmas o paréntesis, sin generar funciones enteras	Sí
Correctores de ortografía, traductores y herramientas de accesibilidad	Sí
Formateadores y linters de código, como `clang-format` o `cppcheck`	Sí
IA generativa para pedir explicaciones, sin que escriba código	No
IA generativa que escribe código, aunque sea una parte	No

- **Cómo declararlo:** si se usa una herramienta permitida que no viene en el entorno del curso, como un traductor o un linter, se dice en un comentario al inicio del archivo entregado: qué herramienta y para qué.
- **Explicarlo de palabra:** en cualquier laboratorio o práctica se puede pedir al estudiante que explique su código de palabra.

BIBLIOGRAFÍA

- Apuntes del curso, un capítulo por semana, en el sitio del curso. No se requiere comprar ningún libro.
- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.