

Carrera/Plan:*Ingeniería en Computación Plan 2024***TALLER DE LENGUAJES I****Año 2026****Año:** 2º año**Régimen de Cursada:** *Semestral***Carácter:** *Optativa***Correlativas:** *Programación II***Profesores:** *Dra. Laura Lanzarini**Esp César Estrebou***Hs semanales:** 2 hs de teoría – 4 hs de práctica**FUNDAMENTACIÓN**

Esta asignatura, como su nombre lo indica, se estructura como un taller orientado a la resolución de problemas concretos mediante el uso de un lenguaje de programación específico. En este contexto, el estudiante enfrenta el desafío de poner en práctica y consolidar los conocimientos adquiridos en asignaturas previas vinculadas con algorítmica y programación. A lo largo del curso se proponen problemas que admiten múltiples enfoques de solución, los cuales son analizados y comparados en forma conjunta. Este enfoque favorece el desarrollo del pensamiento crítico, promueve el intercambio de ideas entre pares y fortalece el trabajo colaborativo.

OBJETIVOS

El objetivo general de esta asignatura es brindar una formación teórico-práctica en un lenguaje de programación procedural (el cual podrá variar en función de la evolución tecnológica), poniendo énfasis en el análisis formal de sus características, su modelo de ejecución y su comparación con otros lenguajes previamente conocidos por el estudiante.

El objetivo específico es estudiar en profundidad el lenguaje de programación **C**, un lenguaje de nivel medio y débilmente tipificado, que combina estructuras propias de los lenguajes de alto nivel con mecanismos que permiten un control detallado de los recursos de hardware. Debido a su amplia utilización en el desarrollo de sistemas, software de base y control de dispositivos, el dominio de este lenguaje resulta especialmente relevante para la formación profesional del ingeniero en computación.

EJES TRANSVERSALES

03. Especificación, proyecto y Desarrollo de Software y Sistemas Conjuntos de Hardware y Software haciendo uso de conceptos, métodos y herramientas de gestión de proyectos, ingeniería de software, base de datos, experiencia del usuario, elicitación, análisis, especificación y validación de requerimientos. (2-Medio)
04. Desarrollo de Redes de Computadoras y de Redes de Computadoras de área amplia, locales, inalámbricas y móviles. (1-Bajo)
06. Proyecto, desarrollo, dirección, control, construcción, operación y mantenimiento de Sistemas de Procesamiento de Señales, Sistemas Embebidos y sus periféricos incluido en software de soporte, Sistemas Computarizados de automatización y control y Sistemas Conjuntos de Hardware y Software. (2-Medio)
09. Identificación, formulación y resolución de problemas de ingeniería en computación. (2-Medio)
12. Utilización de técnicas y herramientas de aplicación en la ingeniería en computación. (2-Medio)
14. Fundamentos para el desempeño en equipos de trabajo. (1-Bajo)
15. Fundamentos para una comunicación efectiva. (1-Bajo)
18. Fundamentos para el aprendizaje continuo. (1-Bajo)

CONTENIDOS MINIMOS

Estudio de un lenguaje de programación (como por ejemplo C, que resulta adecuado para la carrera) en el que se desarrollen aplicaciones concretas.

PROGRAMA ANALÍTICO

1. **Introducción.** La historia de C. ANSI C. Características de C. La estructura de un programa en C. Identificadores. Alcance de un identificador. Cómo trabaja el compilador C. Análisis de un par de programas sencillos en C. Comentarios. Caracteres especiales. La biblioteca estándar de C. Sentencias `printf` y `scanf`.
2. **Tipos de Datos.** Tipos de datos simples. Tipos enteros con una cantidad exacta de bits. Tipos de punto flotante. Conversión de tipos. Conversión de tipos aritméticos. Conversión aritmética usual. Otras conversiones implícitas. Conversión explícita de tipos.
3. **Expresiones, Operadores y Sentencias.** Evaluación de expresiones. Operadores de asignación. Operadores incrementales y decrementales. Sentencia. Bloque de sentencias. Iteraciones. Sentencias `while`, `for` y `do-while`. Loops anidados. Sentencias de selección. Sentencias `if`, `if-else`, operador condicional y `switch`. Saltos incondicionales. Sentencias `break`, `continue`, `return`
4. **Funciones e identificadores.** Definición, declaración e invocación de funciones. Prototipo de una función. Coerción de tipos. Retorno de una función. El tipo `void`; su uso en declaración de funciones. Funciones recursivas. Clases de almacenamiento de un identificador. Persistencia automática: `auto` y `register`. Persistencia estática: `static` y `extern`.
5. **Arreglos.** Arreglos de longitud fija y arreglos de longitud variable. Definición. Acceso a los elementos del arreglo. Inicialización del arreglo. Vector de caracteres. Funciones para cadenas de caracteres. Arreglos y el especificador `static`. Arreglos multidimensionales. Arreglos como argumento de una función.
6. **Punteros.** Declaración de punteros. Inicialización de punteros. Identificador `null`. Operadores de punteros. Visualización del valor de un puntero. Pasaje de parámetros por referencia. El calificador `const` aplicado a arreglos y a punteros. Operaciones aritméticas con punteros. Asignación de punteros. Comparación entre punteros. Arreglos de punteros y punteros a arreglos. Punteros a funciones.
7. **Cadenas de caracteres.** Definición. Declaración e inicialización. Funciones de manejo de caracteres (librería `ctype.h`). Funciones de conversión (librería `stdlib.h`). Funciones de entrada/salida (`stdio.h`). Funciones de comparación, búsqueda y manipulación de cadenas (`string.h`).
8. **Estructuras, uniones y enumeraciones.** Estructuras. Definición. Declaración de variables. Inicialización. Operaciones. Acceso a los campos de una estructura. Punteros como miembros de una estructura. Union. Definición de tipos Union. Operaciones sobre uniones. Operadores a nivel de bits. Operadores de asignación a nivel de bits. Campos de bits. Constantes de enumeración. Ejemplos.
9. **Archivos.** Streams. Dispositivos estándar: `stdin`, `stdout`, `stderr`. Archivos. Definición. Modos de apertura de un archivo. Posición dentro del archivo. Operaciones de alta, baja y modificación de archivos. Acceso secuencial y acceso directo. Manejo de errores de acceso. Archivos binarios.
10. **Manejo de memoria dinámica.** Alocación de memoria de manera dinámica. Redimensionado y liberación de memoria. Arreglos dinámicos. Ejemplificación creando listas. Matrices dinámicas. Alocación y liberación de memoria.
11. **Directivas para preprocesador.**
Directiva `#include`. Acceso a los archivos indicados en la directiva `#include`.

Definición y uso de macros. Macros con y sin parámetros. Macros dentro de macros. Alcance y redefinición de macros. Macros predefinidas. Directiva `#undef`. Compilación condicional. Las directivas `#if` y `#elif`. Directivas `#ifdef` e `#ifndef`.

12. **Librerías.** Headers estándar. Uso de headers. Contenido de los headers estándar. Funciones de la librería estándar según su área de aplicación. Aplicaciones formadas por varios archivos.
13. **Herramientas básicas.** Compilación con GCC. Warnings del compilador. Optimización Debugging. Usando *make* para construir programas en C. Prerrequisitos y comandos. *MakeFile*. Reglas. Comentarios. Variables. Macros. Funciones. Directivas. Ejecutando *make*. Debugging de programas C utilizando GDB. Un ejemplo simple de una sesión de debugging.

BIBLIOGRAFIA

1. C: How to Program. With an Introduction to C++. Deitel, Paul y Deitel, Harvey. Pearson International. 2016
2. C in a Nutshell. Peter Prinz y Torry Crawford. Editorial O'Reilly Media. 2015.
3. C for Programmers with an Introduction to C11. Paul Deitel y Harvey Deitel. Prentice Hall. 2013.
4. Practical C. Programming. Steve Oualline. Editorial O'Reilly Media. 2011.
5. The C Programming Language 2nd Edition. Kernighan y Ritchie. Prentice Hall Software Series. 2016.

METODOLOGÍA DE ENSEÑANZA

La asignatura tiene como objetivo central capacitar al alumno en la resolución de problemas concretos utilizando el lenguaje de programación C. Por tal motivo, el dictado del curso presenta una fuerte coordinación entre la teoría y la práctica. Esta articulación se lleva a cabo mediante la ejemplificación y resolución de situaciones concretas similares a las planteadas en las actividades prácticas. Se trabaja sobre problemas de complejidad moderada cuya solución implica el uso de herramientas de compilación y depuración específicas para C. Se introducen herramientas de desarrollo específicas para C, como GCC y Make. Se enseñan técnicas avanzadas de programación y optimización de código en el contexto del lenguaje C.

Durante la segunda mitad del curso, se presentan conceptos de sistemas embebidos y procesamiento de señales en proyectos con lenguaje C. Su aplicación puede desarrollarse en el contexto de los proyectos de Desarrollo e Innovación con Alumnos que promueve y subsidia la Facultad de Informática. De esta forma, se incentiva el aprendizaje continuo mediante la exploración de nuevas características del lenguaje C y prácticas de desarrollo actualizadas e innovadoras.

El curso posee clases teóricas a cargo del profesor de la asignatura y clases prácticas a cargo de los auxiliares. La carga horaria es de 6 hs semanales repartidas en una clase teórica de 2 hs y dos clases prácticas de 2 hs. cada una.

Se toma asistencia únicamente en los horarios de práctica. Dicha asistencia será registrada con fines administrativos no siendo requerida para rendir o aprobar la materia. Las consultas de los alumnos serán atendidas por los docentes en sus respectivos horarios de clase.

Todo el material del curso estará disponible a través de la plataforma de educación a distancia IDEAS. Tanto alumnos como docentes deberán contar con un usuario y una clave para poder acceder. Se utilizará únicamente la cartelera disponible en IDEAS para dar difusión a las novedades del curso.

Además, la cátedra cuenta con una página web de acceso público en la siguiente dirección

<http://weblidi.info.unlp.edu.ar/catedras/TallerLeng1>

Allí se indica la manera de contacto con la cátedra y la forma de acceder al material publicado en el entorno virtual de enseñanza y aprendizaje **IDEAS**

EVALUACIÓN

El curso está dividido en dos módulos. Al finalizar cada uno de ellos se tomará una evaluación teórico-práctica con un recuperatorio. Al finalizar el curso se tomará una fecha adicional que podrá ser utilizada para recuperar sólo uno de los módulos.

La asignatura ofrece dos modalidades de aprobación

Régimen de promoción

Los alumnos que deseen aprobar la materia por promoción deberán

- Los alumnos que deseen aprobar la materia por promoción deberán obtener en ambos módulos una calificación individual mayor o igual a 5 puntos y un promedio general mayor o igual a 6 puntos.
- La nota final del curso es el promedio de las calificaciones obtenidas en los dos módulos.

Quienes aprueben la promoción y se encuentren inscriptos en el curso bajo esta modalidad, tendrán registrada su nota al final del curso. El resto de los alumnos que promocionen, deberán inscribirse en una mesa de examen final para que se registre oficialmente la calificación obtenida cuando cumplan con las condiciones reglamentarias.

Con evaluación final

- Los alumnos que aprueben ambos módulos con calificación mayor o igual a 4 (cuatro) tendrán aprobada la cursada de la asignatura.

Quienes aprueben sólo la cursada deberán inscribirse en una mesa de examen final donde a través de un coloquio se definirá su calificación final.

Contacto de la cátedra

- e-mail: tallerleng1@gmail.com
- WEB : weblidi.info.unlp.edu.ar/catedras/tallerleng1
- Plataforma virtual de gestión de cursos: IDEAS.



Dra. Lic. Laura Lanzarini