



FACULTAD DE CIENCIAS EXACTAS, FÍSICAS y NATURALES



Universidad  
Nacional  
de Córdoba

Asignatura: **Fundamentos de Programación**

Código: 10-09800

RTF

7

Semestre: Primero

- Ingeniería en Computación

Carga Horaria

96

Bloque: TB (96)

Horas de Práctica

48

Departamento: Computación

Correlativas:

- Matemática

Contenido Sintético:

- Introducción a la programación.
- Elementos de la programación estructurada.
- Estructuras de control.
- Estructuras de datos.
- Funciones y procedimientos.
- Entrada/salida de información.
- Manejo de errores y excepciones.
- Verificación y validación de programas.

Competencias Genéricas:

- CG1: Identificar, formular y resolver problemas de ingeniería.
- CG4: Competencia para utilizar de manera efectiva las técnicas y herramientas de la ingeniería.
- CG7: Comunicarse con efectividad.

Aprobado por HCD: 1042-HCD-2023

RES: Fecha: 27/11/2023

Competencias Específicas para la carrera de Ingeniería en Computación:

- CE7.2.2 Sintetizar, diseñar, desarrollar y analizar programas lenguajes de programación de bajo nivel, como C y C++
- CE7.2.3 Seleccionar y utilizar entornos de desarrollo integrados (IDE) y herramientas de depuración específicas para este tipo de sistemas.

# Presentación

La asignatura “Fundamentos de Programación” pertenece al primer año del plan de estudio de Ingeniería en Computación. Al momento de transitar este espacio curricular, el estudiante ha adquirido conocimientos básicos de matemáticas, que le permitirán comprender los conceptos fundamentales de la programación. En particular, la asignatura se enfoca en la programación estructurada que sienta las bases para comprender conceptos más avanzados en programación y diseño de software en asignaturas posteriores. Los estudiantes aprenderán a desarrollar algoritmos y a escribir código claro y legible, habilidades cruciales en su profesión.

El curso no solo establece las bases conceptuales de la programación, sino que también desarrolla competencias prácticas. La capacidad de identificar, formular y resolver problemas de ingeniería se nutre a medida que los estudiantes enfrentan desafíos de programación y desarrollan soluciones efectivas. Además, se fomenta el uso efectivo de técnicas y herramientas de ingeniería, ya que los estudiantes aprenden a utilizar lenguajes de programación y se familiarizan con entornos de desarrollo integrados (IDE) y herramientas de depuración específicas. Por otro lado, el uso de la comunicación efectiva también se integra en el curso a través de la documentación y el trabajo en equipo, habilidades que son esenciales en la colaboración dentro de equipos de desarrollo de software.

En la actualidad la informática y la programación son componentes esenciales en la mayoría de las industrias y sectores, desde la automatización de procesos industriales hasta la resolución de problemas complejos en campos como la inteligencia artificial y la ciberseguridad. Por lo tanto, esta asignatura representa la puerta de entrada al vasto y dinámico campo de la informática, brindando a los futuros profesionales las competencias fundamentales para abordar los desafíos tecnológicos de hoy y adaptarse a los cambios constantes en el futuro. En última instancia, el conocimiento adquirido en esta asignatura sienta las bases para una carrera profesional que pueda contribuir significativamente al desarrollo tecnológico y la innovación en la sociedad actual.

## Contenidos

### **Unidad 1: Programación estructurada**

Resolución de problemas y algoritmos. Lenguajes de programación. Concepto de programa y sus elementos. Tipos de datos primitivos. Operaciones aritméticas. Variables, constantes y declaraciones. Operaciones de asignación. Entrada y salida estándar de información. Formato de salida. Funciones de biblioteca. Comentarios y convenciones de nomenclatura. Funciones definidas por el usuario: procedimientos y funciones con parámetros por valor. Alcance de variables. Aplicaciones prácticas. Verificación y validación de programas con flujo secuencial.

### **Unidad 2: Estructuras de selección**

Operaciones relacionales. Operaciones lógicas. Precedencia y asociatividad. Estructuras de selección. La estructura de decisión simple. La estructura de decisión doble. Estructuras de decisión anidadas. La estructura de decisión múltiple. Funciones definidas por el usuario: funciones con parámetros por referencia. Aplicaciones. Verificación y validación de programas con flujo selectivo.

### **Unidad 3: Estructuras de repetición**

Estructuras de repetición. Las estructuras de repetición indefinidas. La estructura de repetición definida. Estructuras de repetición anidadas. Alteraciones del flujo normal. Funciones definidas por el usuario: recursividad. Aplicaciones. Verificación y validación de programas con flujo repetitivo.

### **Unidad 4: Estructuras de datos y acceso a archivos**

Arreglos unidimensionales. Inicialización de arreglos. Arreglos bidimensionales. Arreglos como argumentos de función. Estructuras sencillas. Arreglo de estructuras. Estructuras como argumentos de función. Lectura y escritura de archivos. Acceso aleatorio de archivos. Flujo de archivos como argumento de función. Excepciones y comprobación de archivos. Aplicaciones. Verificación y validación de programas con estructuras de datos y archivos.

## **Metodología de enseñanza**

La asignatura se organiza en cuatro unidades didácticas, las cuales se desarrollan en un rango de entre 6 y 8 clases cada una. El dictado del curso se lleva a cabo a través de 2 clases semanales de carácter teórico-práctico, con una duración de 3 horas cada una. La metodología de enseñanza propuesta integra el modelo de aula invertida, el aprendizaje basado en problemas (ABP), el estudio de casos y el uso del aula virtual como complemento esencial a las clases semanales.

Los estudiantes trabajarán con el material de estudio disponible en el aula virtual, que incluye videos, lecturas y casos prácticos, cubriendo conceptos teóricos y ejemplos básicos de programación. Esta plataforma no solo facilita el acceso a los recursos, sino que también permite que los alumnos lleguen a clase con una base sobre la cual construir. Al inicio de cada sesión en clase, se llevará a cabo una revisión y profundización de este material, asegurando que todos los estudiantes hayan comprendido los conceptos clave y estén listos para aplicarlos.

Durante las clases, el docente propondrá y conducirá la actividad de estudio de casos, que se incorporará como una herramienta para analizar situaciones reales o simuladas en el mundo de la programación. Esta actividad permitirá a los estudiantes aplicar y contextualizar los conceptos aprendidos. Estos casos, junto con otros problemas y programas afines propuestos por el docente, servirán como punto de partida para una serie de preguntas conceptuales que fomentarán la discusión y el análisis de las posibles respuestas entre los estudiantes. Esta dinámica tiene como objetivo profundizar en conceptos fundamentales de la programación, permitiendo a los estudiantes no solo entender las soluciones correctas, sino también identificar y discutir errores comunes.

En las clases presenciales, se dedicará tiempo a resolver dudas, realizar ejercicios prácticos y trabajar en problemas específicos. Estos problemas, presentados al inicio de cada unidad, requerirán soluciones a través de la programación. Los estudiantes trabajarán en grupos para discutir y buscar soluciones, con el docente actuando como guía y facilitador. Al final de cada unidad, los grupos presentarán sus soluciones, propiciando un espacio de retroalimentación

y discusión colectiva, enriqueciendo así el proceso de aprendizaje a través de la experiencia compartida.

Además de las clases semanales, y como parte de la fase posterior de la metodología del aula invertida, se complementará la discusión de problemas y ejercicios y la atención de consultas a través de un foro disponible en el aula virtual. Este espacio permitirá la comunicación asincrónica pero continua entre docentes y estudiantes, fomentando un aprendizaje colaborativo y constante, y permitiendo a los estudiantes profundizar y reflexionar sobre lo aprendido en clase.

## Evaluación

La evaluación se estructura en base a un enfoque continuo e integrador, reflejando la naturaleza progresiva y acumulativa del aprendizaje en las áreas de informática y cálculo numérico. Esta metodología de evaluación tiene como objetivo no solo medir el conocimiento adquirido, sino también fomentar una comprensión profunda y aplicada de los conceptos y habilidades aprendidos. En este contexto, la evaluación se realiza a través de una evaluación parcial de cada una de las 4 unidades didácticas a través de las cuales se organiza la asignatura, y una evaluación integradora una vez aprobadas las 4 evaluaciones parciales.

El desarrollo de las competencias se evaluará de forma continua mediante rúbricas, construida en base a los resultados de aprendizaje propuestos.

→ **Evaluación Parcial:** cada una de las cuatro unidades didácticas culmina con una instancia de evaluación la cual es precedida por una actividad práctica.

- ◆ *Trabajo Práctico (TP):* Los estudiantes deberán realizar un trabajo práctico que refleje la aplicación de los conceptos y habilidades aprendidos en la unidad. Este trabajo tiene como objetivo desarrollar la capacidad del estudiante para aplicar de manera práctica y efectiva lo aprendido, y puede incluir ejercicios, problemas, proyectos o simulaciones, según lo que sea más pertinente para la unidad en cuestión. Se dispondrá de al menos 2 semanas para poder realizar esta actividad, debiendo la misma ofrecer realimentación inmediata al estudiante de forma que pueda realizarla de forma asincrónica sin la supervisión del docente dentro del aula virtual.

- ◆ *Evaluación Parcial (EP):* Una vez completado el TP con al menos el 60%, los estudiantes se someterán a una evaluación parcial que medirá su comprensión teórica y práctica de los contenidos de la unidad. Esta evaluación puede incluir preguntas teóricas, problemas prácticos y/o análisis de casos, asegurando una evaluación completa de los conocimientos y habilidades adquiridos en la unidad. La duración de esta actividad no deberá superar los 30 minutos y la misma deberá ser realizada bajo la supervisión del docente durante el horario de clase dentro del aula virtual. Se prevé una instancia de recuperación de esta evaluación al finalizar cada unidad didáctica, y una adicional antes de la evaluación integradora (totalizando 6 instancias de recuperación). En cada instancia será posible recuperar una evaluación parcial de cualquier unidad didáctica ya finalizada.

→ **Evaluación Integradora:** una vez aprobadas las 4 evaluaciones parciales, se propone una instancia de evaluación integradora, de la cual

- ◆ *Práctica Integradora (PI):* Al finalizar cada unidad didáctica, se propondrá al estudiante una actividad integradora, la cual integra todas las unidades ya trabajadas en el curso y consiste en el desarrollo de un programa informático cuya especificación es dada a los estudiantes como enunciado. Esta actividad es temporizada a los fines de poner en práctica la gestión de los tiempos y prioridades en el desarrollo de un programa. Es posible realizarla de forma asincrónica a través del aula virtual, sin necesidad de supervisión docente, y la misma deberá ofrecer realimentación inmediata al estudiante.
- ◆ *Evaluación Integradora (EI):* Una vez aprobadas todas las Evaluaciones Parciales, los estudiantes deberán presentarse a una evaluación integradora de todas las unidades didácticas. Esta evaluación tiene como objetivo medir la capacidad del estudiante para conectar y aplicar de manera integrada los conceptos y habilidades aprendidos a lo largo del curso. Esta evaluación es temporizada y tendrá una duración de 2 horas, debiendo ser realizada en una clase bajo la supervisión de un docente dentro del aula virtual. Se prevé una instancia de recuperación de esta evaluación.

La combinación de evaluaciones parciales y una evaluación integradora garantiza que los estudiantes no solo adquieran y retengan el conocimiento, sino que también desarrollen habilidades críticas y aplicadas esenciales para su futuro profesional en el campo de la ingeniería.

## Condiciones de aprobación

### Condiciones de regularización

Para alcanzar la condición de regular, el estudiante debe cumplir las siguientes condiciones:

1. Asistir al menos al 80% de las clases. La asistencia es registrada en el aula virtual mediante la realización de los TPs y las EPs.
2. Alcanzar un rendimiento no inferior a 60% en cada uno de los Trabajos Prácticos (TP).
3. Alcanzar un rendimiento global no inferior a 60% en cada una de las Evaluaciones Parciales (EP).
4. Alcanzar el criterio de aceptación mínimo para los resultados de aprendizaje propuestos.

### Condiciones de promoción

Para alcanzar la promoción, el alumno debe cumplir las siguientes condiciones:

1. Alcanzar la condición de alumno regular para lo cual se deben cumplir las condiciones indicadas al respecto.
2. Alcanzar un rendimiento no inferior al 60% en la Evaluación Integradora (EI).

Cumplidas estas condiciones, la nota final de promoción se calculará según la siguiente fórmula:

$$\text{Nota Final} = \text{redondear} ( (20\% * \text{TPs} + 20\% * \text{EPs} + 60\% * \text{EI} ) / 10 )$$

Donde TPs y EPs corresponden al resultado de promediar los TP y EP de cada unidad respectivamente.

## Examen final

### Estudiantes Regulares

Los estudiantes regulares deben rendir un examen equivalente a la actividad Evaluación Integradora (EI), el cual será calificado del mismo modo que para los estudiantes que alcanzaron la promoción, considerando para las variables TPs y EPs el rendimiento alcanzado durante la cursada, y para la variable EI el rendimiento alcanzado en el examen final.

### Estudiantes Libres

Los estudiantes libres deben rendir un examen que consta de dos partes:

Una prueba de competencias con la misma metodología y objetivos que la Evaluación Parcial (EP), acotada a ejercicios que serán evaluados automáticamente en el aula virtual. La aprobación de esta primera parte es requisito excluyente para la prosecución del examen, y se deberá obtener un rendimiento no inferior al 60%.

Un examen de Evaluación Integradora (EI), con la misma modalidad y objetivos que el requerido a los estudiantes regulares.

La Nota Final final para los estudiantes libres se obtendrá por la siguiente expresión:

$$\text{Nota Final} = \text{redondear}((40\% \text{ EP} + 60\% \text{ EI})/10)$$

## Actividades prácticas y de laboratorio

Las actividades prácticas se desarrollan a partir de trabajos prácticos disponibles en el aula virtual. Cada una de las 4 unidades sobre la que se organiza el recorrido formativo de la asignatura comprende un Trabajo Práctico (TP) compuesto por ejercicios que proponen el desarrollo de programas como solución a diferentes problemas generales. Cada problema es descrito a través de un enunciado y al menos una solución particular es ejemplificada indicando el resultado esperado. Para cada problema enunciado se espera que el estudiante provea un código en lenguaje de programación que será evaluado de forma inmediata a través de diferentes pruebas predefinidas. Estas pruebas comprenden soluciones a diferentes casos que el programa debe resolver de forma satisfactoria, y ofrece al estudiante una realimentación inmediata sobre las pruebas superadas y las fallidas. De esta forma, el estudiante puede realizar un diagnóstico sobre los casos donde falla y buscar así modificar su solución para que la misma sea de carácter general, responda a la consigna, y pueda entonces superar todas las pruebas propuestas.

## Desagregado de competencias y resultados de aprendizaje

Los resultados de aprendizaje a promover en el desarrollo de la asignatura relacionados con el descriptor "Lenguajes, algoritmos y estructuras de datos" (Tecnología Básica) de la carrera

de Ingeniería en Computación son X. Estos representan una amplia gama de habilidades y conocimientos relacionados con la programación y su aplicación en la resolución de problemas a través del diseño, codificación y verificación de algoritmos en un lenguaje de programación.

- RA1: Comprender y analizar programas codificados en un lenguaje de programación para identificar su funcionalidad y lógica.
- RA2: Analizar y evaluar problemas para determinar los requisitos de programación y diseñar soluciones efectivas antes de iniciar el proceso de codificación.
- RA3: Diseñar soluciones para problemas complejos y traducirlas en algoritmos codificados en un lenguaje de programación.
- RA4: Gestionar eficientemente la memoria de la computadora utilizando variables, constantes y tipos de datos primitivos para almacenar y manipular información.
- RA5: Aplicar operaciones aritméticas y de asignación de manera precisa para resolver problemas matemáticos y científicos.
- RA6: Gestionar la entrada y salida de datos en programas mediante el uso efectivo de flujos de información.
- RA7: Mejorar la legibilidad y la colaboración en el desarrollo de software mediante el uso de comentarios y el seguimiento de convenciones de nomenclatura.
- RA8: Diseñar, crear y aplicar funciones definidas por el usuario para modular y reutilizar el código de manera eficiente.
- RA9: Desarrollar programas con múltiples flujos de ejecución mediante el uso experto de estructuras de control, como condicionales y bucles.
- RA10: Implementar pruebas rigurosas para verificar la precisión y la conformidad con las especificaciones de los programas, y resolver eficazmente los errores identificados.

A continuación, se indican las competencias genéricas y específicas asociadas a los resultados de aprendizaje relacionados con la carrera de Ingeniería en Computación.

<b>Competencias Genéricas</b>	<b>Resultados de Aprendizaje</b>
CG1: Identificar, formular y resolver problemas de ingeniería.	RA1, RA2, RA8, RA9
CG4: Competencia para utilizar de manera efectiva las técnicas y herramientas de la ingeniería.	RA3, RA4, RA5, RA6
CG7: Comunicarse con efectividad.	RA7, RA10

<b>Competencias Específicas (Ingeniería en Computación)</b>	<b>Resultados de Aprendizaje</b>
CE7.2.2 Sintetizar, diseñar, desarrollar y analizar programas lenguajes de programación de bajo nivel, como C y C++.	RA1, RA2, RA3, RA4, RA5, RA6, RA8, RA9
CE7.2.3 Seleccionar y utilizar entornos de desarrollo integrados (IDE) y herramientas de depuración específicas	RA7, RA10



para este tipo de sistemas.	
-----------------------------	--

## Bibliografía

- Bronson, Gary, *C++ para Ingeniería y Ciencias (2da. edición)*, International Thomson Editores, México, 2007
- Deitel, H. M., Deitel, P. J., *Cómo programar en C++*, Pearson Educación, 2015