FCEFyN
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS y NATURALES

UNC Universidad Nacional de Córdoba

Asignatura: **FUNDAMENTOS DE PROGRAMACIÓN**

Código:10-09800	RTF	7
Semestre: Primero	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: RES: Fecha:

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: Arreglos
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.

Unidad 4: Estructuras de datos: Registros
Definición y uso de registros. Declaración, inicialización y acceso a campos. Registros anidados. Arreglos de registros: carga, modificación y baja lógica. Búsqueda secuencial y ordenamiento de arreglos de registros. Archivos de registros. Uso combinado de estructuras, arreglos y archivos para resolver problemas aplicados. Verificación y validación de programas que gestionan conjuntos de registros persistentes.

Metodología de enseñanza

La asignatura se organiza en cinco unidades didácticas, las cuales se desarrollan en un rango de entre 4 y 6 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.

Se ofrecerán comisiones de dictado presencial físico en ambas clases semanales, pudiendo ofrecerse en algunas comisiones una de las clases semanales bajo la modalidad presencial remota, que son acordes a la metodología de enseñanza-aprendizaje de los contenidos teóricos y prácticos de la materia y brindan alternativas flexibles para los estudiantes de primer año. En el caso de las clases presenciales, las mismas se desarrollarán en los Laboratorios de Informática de la Facultad.

En el caso de que se desarrollen instancias de presencialidad remota seguirán las pautas institucionales y se desarrollarán vía Google Meet (o la herramienta que la reemplace) que permite las instancias de interacción y trabajo colaborativo a través de las diversas herramientas disponibles. El acceso será restringido y requerirá identificación del estudiante a través de su cuenta institucional (IdUNC) para quienes estén matriculados en la comisión. Los estudiantes podrán participar de las clases mediadas por tecnología, desde ubicaciones externas o utilizando las instalaciones y la conectividad disponible en la Facultad.

Todos los estudiantes trabajarán con el material de estudio disponible en el aula virtual, a través de la plataforma institucional (Moodle o la que la reemplace), 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. El aula virtual servirá como soporte para complementar las actividades sincrónicas de los estudiantes. Este entorno brinda nuevos canales de comunicación y herramientas de trabajo colaborativo (foros) y acceso a información de la materia.

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 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. Las actividades se enmarcan en la utilización de software de código abierto y gratuito para la resolución de ejercicios y problemas contextualizados, en forma individual o grupal.

Evaluación

La evaluación de la asignatura se organiza bajo un enfoque continuo, integrador y orientado a competencias, acorde a la naturaleza progresiva del aprendizaje en programación. Este enfoque busca no solo medir conocimientos, sino también evidenciar la capacidad del estudiante para comprender, analizar y construir programas, combinando teoría y práctica de manera coherente.

En este marco, la propuesta de evaluación contempla 2 evaluaciones parciales, cada una compuesta por una parte conceptual y una parte práctica, complementadas por actividades de seguimiento dentro de cada unidad.

- Primera evaluación parcial: abarca las primeras 3 unidades, correspondientes a los fundamentos de la programación, estructuras de control y arreglos. Esta evaluación está compuesta por la Evaluación Conceptual 1 (EC1) y la Evaluación Práctica 1 (EP1).
- Segundo evaluación parcial: evalúa las 3 unidades finales, enfocadas en estructuras y archivos, integrando los conocimientos previos de manera acumulativa. Esta evaluación está compuesta por la Evaluación Conceptual 1 (EC1) y la Evaluación Práctica 1 (EP1).

Ambas evaluaciones parciales combinan evaluación conceptual y práctica, lo que garantiza que el estudiante no solo pueda interpretar programas, sino también construir soluciones propias, promoviendo un aprendizaje equilibrado y significativo.

- Enfoque conceptual

  La parte conceptual de cada parcial tiene como objetivo evaluar la comprensión

razonada de los contenidos abordados. Esto incluye la interpretación de algoritmos, el análisis de fragmentos de código, la identificación de errores y la explicación de comportamientos del programa.

Este tipo de evaluación permite constatar que el estudiante comprende las estructuras, tipos de datos y mecanismos de control, más allá de su mera aplicación mecánica.

- Enfoque práctico

  La parte práctica se centra en la elaboración de un programa que resuelva un problema utilizando los conceptos estudiados. Aquí se pone en juego la capacidad para diseñar, implementar y validar soluciones algorítmicas.

  Este componente refleja la naturaleza aplicada de la programación y permite valorar habilidades esenciales: descomposición del problema, uso adecuado de estructuras, manejo de archivos o arreglos, y calidad general del código.

Todas las evaluaciones serán en modalidad presencial físico, no siendo posible realizarlas en la modalidad remota. Se prevé un Recuperatorio tanto para una de las 2 Evaluaciones Conceptuales como para una de las 2 Evaluaciones Prácticas.

Actividades prácticas y de laboratorio

Cada una de las 5 unidades sobre la que se organiza el recorrido formativo de la asignatura comprende una Actividad Conceptual (AC) y una Actividad Práctica (AP). Las actividades conceptuales comprenden ejercicios conceptuales sobre los contenidos abordados. Esto incluye la interpretación de algoritmos, el análisis de fragmentos de código, la identificación de errores y la explicación de comportamientos del programa. Por otro lado, las actividades prácticas están compuestas 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. La actividad práctica se desarrolla sobre un entorno de programación provisto por la asignatura, pudiendo el estudiante optar por otros similares, que le permite desarrollar la solución a cada problema a través de un programa y realizar las pruebas necesarias para alcanzar el resultado esperado. Las actividades prácticas planificadas son:

- Actividad Práctica 1 (AP1): Comprende un conjunto de 30 actividades, de complejidad creciente, que requieren diseñar una solución a través de un programa que se ejecuta de forma completamente secuencial, utilizando operadores aritméticos, relacionales y lógicos en conjunto con variables donde almacenar temporalmente la información.
- Actividad Práctica 2 (AP2): Comprende un conjunto de 30 actividades, de complejidad

creciente, que requieren diseñar una solución a través de un programa que se ejecuta con bifurcaciones en su flujo por medio de condicionales, utilizando estructuras de control e integrando expresiones que combinan diferentes operadores relacionales y lógicos.

- Actividad Práctica 3 (AP3): Comprende un conjunto de 30 actividades, de complejidad creciente, que requieren diseñar una solución a través de un programa que se ejecuta con ciclos en su flujo, utilizando estructuras de control que permitan realizar repeticiones definidas e indefinidas.
- Actividad Práctica 4 (AP4): Comprende un conjunto de 30 actividades, de complejidad creciente, que requieren diseñar una solución a través de un programa que integra estructuras de datos homogéneas basadas en arreglos, utilizando arreglos unidimensionales y bidimensionales.
- Actividad Práctica 5 (AP5): Comprende un conjunto de 30 actividades, de complejidad creciente, que requieren diseñar una solución a través de un programa que integra estructuras de datos heterogéneas basadas en registros, utilizando registros simples y anidados como también arreglos de registros.

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 actividades que debe realizar el estudiante durante la clase.
2. Alcanzar un rendimiento no inferior a 60% en cada una de las Actividades Conceptuales (AC) y Actividades Prácticas (AP).
3. Alcanzar un rendimiento global no inferior a 60% en cada una de las Evaluaciones Conceptuales (ECs).

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 las Evaluaciones Prácticas (EPs).

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

Nota Final = redondear((20%*APs + 20%*ACs + 30%*ECs + 30%*EPs)/10)

Donde APs, ACs, ECs y EPs son el resultado de promediar el total de instancias correspondientes a cada actividad y/o evaluación.

Examen final

Estudiantes Regulares

Los estudiantes regulares deben rendir un examen equivalente a la actividad Evaluación Práctica (EP), el cual será calificado del mismo modo que para los estudiantes que alcanzaron la promoción, considerando para las variables ACs y APs 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 Conceptual (EC). La aprobación de esta primera parte es requisito excluyente para la prosecución del examen, y se deberá obtener un rendimiento no inferior al 60%. Una Evaluación Práctica (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:

Nota Final = redondear((40% EC + 60% EP)/10)

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 10. 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.

Competencias Genéricas	Resultados de Aprendizaje
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

Competencias Específicas (Ingeniería en Computación)	Resultados de Aprendizaje
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 para este tipo de sistemas	RA7, RA10

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
- Garrido, Antonio, Fundamentos de programación en C++, Delta Publicaciones, 2005
- Peñaloza Romero, E., Fundamentos de programación C/C++ (4ª ed.), Alfaomega Grupo Editor, 2004