000027

Programación estructurada

INSTITUTOS/S: **INSTITUTO DE TECNOLOGÍA E INGENIERÍA**

CARRERA/S: **Licenciatura en Informática.**

RESPONSABLE DE LA ASIGNATURA Y EQUIPO DOCENTE:
Fabiana Conde

Hernán Quiroga
Nicolás Ríos

AÑO: 2025

CRÉDITOS: 7

CARGA HORARIA DE INTERACCIÓN PEDAGÓGICA: 96

CARGA HORARIA TOTAL: 175

CÓDIGO DE LA MATERIA EN SIU: 792

000027

UNAHUR
UNIVERSIDAD NACIONAL DE HURLINGHAM

1. Fundamentación

Esta materia se considera la base de las siguientes materias del área de Algoritmos y Lenguajes, se busca fomentar el concepto principal de resolución de problemas como clave para entender el proceso de implementación de un programa informático. El estudio de los fundamentos de programación permitirá a los/as estudiantes poder plasmar por ellos mismos soluciones a problemas de distinto grado de complejidad. Además, el conocimiento adquirido en la materia les posibilitará comprender en futuros cursos el funcionamiento de herramientas profesionales de la industria del software.

2. Propósitos y/u objetivos

- *Acercar al/la estudiante al concepto de problema computacional y la forma de resolver estos problemas mediante programación.*
- *Instalar los conocimientos sobre herramientas de programación que sean transversales a diversos paradigmas y lenguajes, y aplicando dichos conocimientos en un lenguaje imperativo puntual.*
- *Promover la correcta resolución de problemas computacionales mediante la técnica de división en subtareas, fomentando el reuso y la generalización.*
- *Invitar a pensar sobre el código realizado y sus implicancias, analizando y escribiendo propósitos y precondiciones para los mismos, y usando estas últimas para comprender mejor cuando los programas pueden fallar.*

3. Contenidos mínimos

Qué es un programa. Entornos de desarrollo y ejecución. Principios de la programación imperativa: Comandos (acciones) y Expresiones (valores), estructuras de control de flujo de programas (secuencia, repetición simple, repetición condicional, alternativa condicional en comandos y expresiones), estado, tipos de datos (números, booleanos, cadenas, enumerativos simples, con estructura mediante campos). División en subtareas como metodología para la resolución de problemas complejos, y necesidad de dar estructura a un programa no trivial. Generalización de programas mediante parametrización. Resolución de problemas y algoritmos mediante programas. Precondiciones como metodología para el

desarrollo de software robusto. Buenas prácticas de programación (indentación, documentación, elección de nombres).

4. Carga horaria

Créditos	*Interacción pedagógica*		*Trabajo Autónomo*	*Total*
7	*Teoría: 48*	*Práctica: 48*	*79*	*175*

4.1Trabajo autónomo del/la estudiante

Actividad	*Carga horaria*
Lectura de contenidos teóricos, apuntes o bibliografía.	*20*
Desarrollo de los ejercicios prácticos propuestos.	*29*
Práctica sobre los mismos ejercicios planteados en clase	*10*
Preparar temas para parcial.	*20*

5. Programa analítico

UNIDAD 1: *Programas y conceptos básicos.*

Conceptos básicos sobre programación:
Concepto de programa (Cómo descripción de solución a problemas computacionales); Concepto de lenguajes de programación; Concepto de problemas computacionales; Conceptos y ejemplos de código fuente; Ambiente de desarrollo (IDEs); Entorno de programación vs. Entornos de ejecución.
Elementos básicos de los lenguajes de programación imperativos (y generales):
Comandos (Como descripción de acciones); Expresiones (Como descripción de información/datos); Argumentos como forma de brindar información a un comando o expresión; Programas simples y cuerpos (bloques) de código; Secuenciación como forma de organización del código; Comentarios.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados:
Concepto de sintaxis estricta; Errores de sintaxis; Indentación; Documentación mediante el uso de comentarios; Propósitos (Como descripción de la transformación realizada); Precondiciones (Como requerimientos sobre el estado/datos iniciales); Concepto de error por fallo de precondición; Comandos totales y parciales; Concepto de estado (inicial, final e intermedios); Concepto de transformación de estados; Equivalencia de programas; Comunicación como sinónimo de programación, y como eje central de buenos programas.

UNIDAD 2: *Procedimientos.*

Concepto de procedimiento enfocados como:
Herramienta para definir nuevos comandos; Forma de comunicar claramente las ideas, tanto en estrategia como en abstracción; Forma de separar el problema en partes más pequeñas que permitan plantear de forma clara la estrategia de solución elegida; Abstracción de los elementos de representación subyacentes; Forma de reutilización de código.
Metodologías de solución a problemas más complejos: División en subtareas como forma de solucionar problemas (divide and conquer); Metodología top-down; Metodología bottom-up; Análisis sobre ventajas y desventajas de cada metodología y situaciones de aplicación.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados:
Elección de buenos nombres como forma de comunicar mejor las ideas elaboradas; Conceptualización de procedimientos como programas independientes; Conceptualización de procedimientos; como múltiples programas que colaboran en la solución de un problema; Procedimentación como forma de encapsulamiento de transformaciones; Documentación de procedimientos.

UNIDAD 3: *Repetición simple***.**

Elementos básicos de los lenguajes de programación imperativos (y generales):
Repetición simple (cantidad fija y finita de veces); Comandos compuestos (Comandos que esperan un cuerpo o bloque); Expresiones numéricas literales; Repetición como nueva forma de organización de código; - Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Casos de borde.

UNIDAD 4: *Parámetros.*

Elementos básicos de los lenguajes de programación imperativos (y generales):
Definición de parámetros (Concepto de parámetro formal, únicamente por valor); Argumentos como forma de dar valor a un parámetro en tiempo de ejecución (Concepto de parámetro real); Uso de parámetros; Invocación de comandos/procedimientos con parámetros (cantidad y orden); Errores por cantidades incorrectas de argumentos.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados:
Parámetros y Alcance; Programas generalizados; Comparativas de programas no generalizados vs. Generalizados; Documentación de parámetros.

UNIDAD 5: *Expresiones y tipos.*

Elementos básicos de los lenguajes de programación imperativos (y generales): Expresiones primitivas (Sensores de información de la máquina); Operadores aritméticos; Operadores de enumeración (Sobre tipos enumerativos).
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Concepto de tipo de datos (para tipos simples); Categorización de expresiones en tipos; Expresiones que esperan argumentos; Errores de tipos; Utilización de expresiones para solucionar problemas que requieren lectura de información de sensores.

UNIDAD 6: *Alternativas condicionales.*

Elementos básicos de los lenguajes de programación imperativos (y generales): Alternativa condicional; Tipos booleanos; Expresiones booleanas; Operadores de comparación; Operadores lógicos (negación, conjunción y disyunción); Alternativa como nueva forma de organizar el código.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Alternativas simples (una rama, if-then) vs. alternativas completas (dos ramas, if-then-else) vs. multialternativas (if-then-elseif-else); Alternativa como comando compuesto; Programación defensiva vs. precondiciones.

UNIDAD 7: *Funciones simples.*

Concepto de función enfocados como: Herramienta para definir nuevas expresiones; Forma de comunicar claramente las ideas, tanto en estrategia como en abstracción; Forma de separar el problema en partes más pequeñas que permitan plantear de forma clara la estrategia de solución elegida; Abstracción de los elementos de representación subyacentes; Forma de reutilización de código.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Funciones como expresiones; Denotación vs. Efecto; Circuito corto como estrategia para garantizar precondiciones.

UNIDAD 8: *Repetición condicional y funciones con procesamiento.*

Elementos básicos de los lenguajes de programación imperativos (y generales): Repetición condicional.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Parcialidad por no terminación; Invariante de bucles; Postcondición de un bucle; Casos de borde; Recorridos secuenciales, como estrategia para garantizar la terminación; Esquemas de recorridos secuenciales comunes para procesamiento y búsqueda;

Recorridos sobre distintos tipos de elementos (elementos de una fila, columna, tablero, camino, objetos de un mapa, etc.)

UNIDAD 9: *Variables y funciones con procesamiento.*

Elementos básicos de los lenguajes de programación imperativos (y generales): Variables;Funciones con procesamiento (Funciones con cuerpo que realizan cálculos para determinar el valor final); Alternativa condicional como expresión.
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Tipos de variables; Inicialización; Asignación (Concepto del value); Propósito de variables (Invariante de ciclo); Construcciones usuales con variables (Contador, acumulador, etc.); Diferencias entre alternativas que son comandos y aquellas que son expresiones.

UNIDAD 10: *Tipos de datos personalizados.*

Elementos básicos de los lenguajes de programación imperativos (y generales): Construcciones para definir nuevos tipos de datos; Tipos de datos variantes (enumerativos); Tipos de datos registro (con estructura); Constructores; Campos de un registro; Funciones de acceso a campos de un registro; Strings (Texto).
Buenas prácticas de desarrollo de software y elementos conceptuales avanzados: Modelado de problemas con registros y variantes; Inmutabilidad vs. Mutabilidad;Problemas como transformaciones de información.

6. Ejes y enunciados Multidimensionales y transversales

Eje	*Nivel de logro de Aprendizaje*	*Acciones de enseñanza*
C1. Identificación, formulación y resolución de problemas de informática	*Alto*	*Resolución de ejercicios y problemas.*
C2. Concepción, diseño y desarrollo de proyectos de informática	*No aporta*	
C3. Gestión, planificación, ejecución y control de proyectos de informática	*No aporta*	

C4. Utilización de técnicas y herramientas de aplicación en la informática	*No aporta*	
C5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas	*No aporta*	
C6. Fundamentos para el desempeño en equipos de trabajo	*medio*	*Aprendizaje cooperativo.*
C7. Fundamentos para la comunicación efectiva	*No aporta*	
C8. Fundamentos para la acción ética y responsable	*Bajo*	*Clases dialogadas.*
C9. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad en el contexto global y local	*No aporta*	
C10.Fundamentos para el aprendizaje continuo	*No aporta*	
C11.Fundamentos para la acción emprendedora	*No aporta*	

7. Bibliografía y recursos.

7.1. Bibliografía obligatoria

Martínez López, Pablo E. (2013). "Las bases conceptuales de la programación: Una nueva forma de aprender a programar" - 1ra Ed. Ebook, La Plata.

Joyanes Aguilar, Luis. (2008). "Fundamentos de programación. algoritmos y estructuras de datos" - 4ta Edición, Mc Graw-Hill.

Zelle, J. M. (2017)." Python programming: An introduction to computer science" (3rd ed.). Franklin, Beedle & Associates.
Disponible en:
https://www.krishnagudi.com/wp-content/uploads/2023/05/Python-Programming-An-Introduction-to-Computer-Science-John-M.-Zelle.pdf

7.2. Bibliografía optativa

Dahl, O. J., Dijkstra E. W. y Hoare, A. R. (1972)." Structured Programming", Academic Press, London and New York.

Knuth, Donald E. (1997). "The Art of Computer Programming." 3rd ed., Addison Wesley.

Cormen, Thomas H. (2001). "Introduction to Algorithms". Cambridge, Mass: MIT Press.

Aho, Alfred V, Ullman, Jeffrey D, Hopcroft, John E. (1983). "Data Structures And Algorithms", Addison-Wesley.

De Giusti, Armando E. (2001). "Algoritmos, Datos y Programas", Prentice Hall.

Wirth, Nikolas. (1997)." Systematic Programming: An introduction", Prentice Hall, Englewood Cliffs (NJ).

7.3. Recursos

El Campus Virtual es un espacio fundamental para el desarrollo de la asignatura. El mismo se utilizará como espacio de seguimiento de los/las estudiantes fuera de los espacios presenciales/sincrónicos, repositorios de materiales y actividades auto-asistidas.
El espacio aúlico virtual contará al menos con:
Un canal de comunicación unidireccional con los/las estudiantes (Foro de avisos).
Enlaces a las herramientas digitales utilizadas (Gobstones / Discord/ Python).
Enlace a los materiales de bibliografía obligatorios.
Enlaces a las actividades de indagación.
Enlaces a los videos teóricos.
Enlaces a las actividades prácticas.
Un espacio de consultas de notas y progresos personales.

8. Metodología de enseñanza

8.1 Modalidades u opciones pedagógicas

Atendiendo a la evolución en las metodologías de enseñanza de la programación, se planificará la materia sobre una secuencia ordenada de actividades basadas en tres ejes,

la indagación, la teoría y la práctica. Estos ejes se aplicarán a cada concepto y herramienta a desarrollar.
Las actividades de indagación consisten en ejercicios que se le presentan imposibles al estudiante con los conocimientos al momento conseguidos, pero que podrá solucionar mediante el descubrimiento de una nueva herramienta (que se logrará mediante el adecuado uso de entornos didácticos de programación).
Las actividades teóricas consisten en explicar las actividades de indagación mediante puestas en común, expandir sobre los conceptos teóricos, relacionándolo con otros conceptos previos y casos de la industria, y mostrar ejemplos adicionales que se solucionarán y pondrán en práctica.
Finalmente se pasará a trabajar los nuevos conceptos en ejercicios prácticos que pondrán diversos desafíos a los/as estudiantes, y que deberán ir solucionando aplicando todos los conceptos obtenidos al momento.

8.2 Formación Práctica

En la práctica se presentará una guía de ejercicios, escrita, en donde se presentarán distintos problemas a resolver. Estos problemas deben poder ser resueltos en una herramienta de programación mediante texto, acercándose un poco más a la forma industrial de programación. También se presentarán una serie de actividades integradoras que permitan trabajar los temas en un nivel más profundo.

9. Evaluación y régimen de aprobación

9.1 Modalidad de evaluación

El sistema normal de evaluación consistirá en 2 (dos) exámenes parciales con recuperatorios, según el cronograma previsto, de la totalidad de la materia descripta en el programa. Los mismos se realizarán en las fechas que, a tal efecto, se establezcan en el cronograma.
Además, se considera como parte de la evaluación de la cursada el desarrollo de las actividades prácticas realizadas durante la cursada de la materia.
El alumno deberá poseer una asistencia no inferior al 75% en las clases presenciales.

9.2 Aprobación de la cursada

Para aprobar la cursada y obtener la condición de regular, el régimen académico establece que debe obtenerse una nota no inferior a cuatro (4) puntos. Todas las

instancias evaluativas deberán tener una instancia de recuperatorio. Podrán acceder a la administración de esta modalidad solo aquellos y aquellas estudiantes que hayan obtenido una nota inferior o igual a 6 (seis) puntos en el examen parcial.

Siempre que se realice una evaluación de carácter recuperatorio, la calificación que los/as estudiantes obtengan reemplazará la calificación obtenida en el examen que se ha recuperado y será la considerada definitiva a los efectos de la aprobación.

9.3 Acreditación de la materia

La materia puede aprobarse por promoción, evaluación integradora, examen final o libre.

Promoción directa: *tal como lo establece el art°17 del Régimen Académico, para acceder a esta modalidad, el/la estudiante deberá aprobar la cursada de la materia con una nota no inferior a siete (7) puntos, no obteniendo en ninguna de las instancias de evaluación parcial menos de seis (6) puntos, sean evaluaciones parciales o recuperatorios. El promedio estricto resultante deberá ser una nota igual o superior a siete (7) sin mediar ningún redondeo.*

Evaluación integradora: *tal como lo establece el art°18 del Régimen Académico, podrán acceder a esta evaluación aquellos estudiantes que hayan aprobado lo cursado con una nota de entre cuatro (4) y seis (6) puntos.*
La evaluación integradora tendrá lugar por única vez en el primer llamado a exámenes finales posterior al término de la cursada. Deberá tener lugar en el mismo día y horario de la cursada y será administrada, preferentemente, por el/la docente a cargo de la comisión. Se aprobará tal instancia con una nota igual o superior a cuatro (4) puntos, significando la aprobación de la materia.
La nota obtenida se promediará con la nota de la cursada.

Examen final*: Instancia destinada a quienes opten por no rendir la evaluación integradora o hayan regularizado la materia en cuatrimestres anteriores. Se evalúa la totalidad de los contenidos del programa de la materia y se aprueba con una calificación igual o superior a cuatro (4) puntos. Esta nota no se promedia con la cursada.*