INSTITUTO: Tecnología e Ingeniería

CARRERA/S: Tecnicatura Universitaria en Informática / Licenciatura en Informática

MATERIA: Introducción a la programación

NOMBRE DEL RESPONSABLE DE LA ASIGNATURA:

CUATRIMESTRE: 1° **AÑO:** 1

PROGRAMA N°: Tecnicatura: 2 / Licenciatura: 2

FECHA (de aprobación): 15/02/2022

Instituto/s: Tecnología e Ingeniería

Carrera/s: Tecnicatura Universitaria en Informática **/** Licenciatura en Informática

Nombre de la materia: Introducción a la programación

Responsable de la asignatura y equipo docente: Alan Rodas, Silvia Silvetti, Patricia Lagomarsino, Alan Gauna, Fabiana Conde, Alejandra Pinto

Cuatrimestre y año: 1° del 1er año

Carga horaria semanal: 8 horas

Programa N°: Tecnicatura: 2 / Licenciatura: 2

Código de la materia en SIU: 767

Introducción a la Programación

1. Fundamentación:

Considerada 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 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:

Esta materia tiene los siguientes propósitos:

- Acercar al 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.

Objetivos:

Que los/las estudiantes:

- Conozcan los elementos básicos de un lenguaje de programación del paradigma estructurado y las distintas formas de organizar su código en el mismo (secuencia, repetición y alternativa en distintas formas).
- Puedan estructurar su programa en partes pequeñas y sencillas que interactúan entre sí para solucionar problemas más complejos (procedimientos y funciones), entendiendo los conceptos de parámetro y valor de retorno.
- Comprendan el concepto de tipo de datos, y pueda aprovecharlo para organizar el estado y las transformaciones de información de los programas que construye.
- Logren pensar estrategias de soluciones a problemas computacionales simples, y sea capaz de implementarlas en un lenguaje de programación concreto, comunicando clara y eficazmente las ideas desarrolladas.
- Sean capaz de separar los elementos relevantes de los elementos accesorios en un problema de programación, determinando qué conceptos deben efectivamente ser modelados en sus soluciones y cuáles no.
- Manejen ciertos principios, prácticas y metodologías básicas de la disciplina que resulten en programas más robustos y mantenibles (incluidos comentarios, elección de nombres, precondiciones y poscondiciones, conceptos de terminación y no terminación y documentación).

3. Programa sintético:

Qué es un programa. Entornos de desarrollo y ejecución. Comparativa de paradigmas de programación: Imperativo, Orientado a Objetos, Funcional, Lógico. 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). Principios de la programación estructurada: funciones y procedimientos. 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. Programa Analítico:

4.1 Organización del contenido:

UNIDAD 1: Programas y comandos 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 completa (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 de lvalue).
    - 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.

4.2 Bibliografía obligatoria:

- Martínez López, Pablo E., Las bases conceptuales de la programación: Una nueva forma de aprender a programar - 1ra Ed. Ebook, La Plata, 2013
- de Giusti, Armando E., Algoritmos, Datos y Programas, Prentice Hall, 2001
- Joyanes Aguilar, Luis, Fundamentos de programación. algoritmos y estructuras de datos - 4ta Edición, Mc Graw-Hill, 2008

4.3 Bibliografía de consulta:

- Dahl, O. J., Dijkstra E. W. y Hoare, A. R., Structured Programming, Academic Press, London and New York, 1972.
- Knuth, Donald E. The Art of Computer Programming. 3rd ed., Addison Wesley, 1997.
- Cormen, Thomas H, and Thomas H. Cormen. Introduction to Algorithms. Cambridge, Mass: MIT Press, 2001.
- Aho, Alfred V, Ullman, Jeffrey D, Hopcroft, John E., Data Structures And Algorithms, Addison-Wesley, 1983.
- Wirth, Nikolas. Systematic Programming: An introduction, Prentice Hall, Englewood Cliffs (NJ), 1997.

5. Metodologías de enseñanza:

El curso está estructurado en una serie de unidades que se presentan siempre con una misma dinámica de tres partes. En la primera parte quienes cursan la materia deberán realizar una serie de ejercicios que desafían los conocimientos aprendidos al momento, y que requerirán de la indagación para descubrir por su cuenta un nuevo concepto o herramienta. Acto seguido, en la segunda parte, contarán con material teórico que expandirá sobre ese nuevo concepto, profundizando en su comprensión, conceptualización, análisis de casos, y otros. En la tercer y última parte se presentará una serie de ejercicios prácticos para aplicar el nuevo concepto, sumándose a los ya existentes, para resolver problemas de diversas características. Esta estructura de indagación-teoría-práctica se repite para cada tema en una unidad.

Atentos a las nuevas metodologías pedagógicas en general, y de enseñanza de la programación en particular, las actividades utilizarán formatos modernos y dinámicos.

- Las actividades de indagación se realizarán con una herramienta que permita la indagación y la solución de problemas nuevos, sin necesidad de aprender previamente detalles de sintaxis de lenguajes o conceptos teóricos complejos, y permitiendo que sean las/los estudiante quienes arriben a las soluciones, con nula o escasa asistencia del equipo docente. En ese sentido se utilizarán herramientas de programación visual mediante bloques encastrables.
- En la teoría se optará por una dinámica de aula invertida, en dónde se pretende que se llegue a las clases presenciales/encuentros virtuales sincrónicos con los conceptos teóricos ya vistos y (parcialmente) aprendidos. En ese sentido se dispondrán de una serie de videos de material teórico, ya sea de producción propia o de terceros, que abarquen la totalidad de los temas trabajados en una secuencia

idéntica a la trabajada. Estos videos se complementarán con material bibliográfico, de forma escrita, permitiendo ampliar los temas e incluir a personas invidentes o con otra capacidad diferente en dónde los videos no resultan una solución.
- 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.

Por la dinámica del curso, y por ser parte de sus objetivos, las actividades integradoras tendrán enunciados complejos, planteando situaciones de problemas realistas, que fomentan el análisis y la concepción de soluciones con la participación activa de quienes cursan, permitiendo el trabajo colaborativo y el debate entre integrantes del grupo, con el acompañamiento de los docentes.

De hecho, el trabajo en grupo será un eje central de la materia. Así se desarrollarán múltiples instancias de trabajo grupal, pero se invitará a no compartir de forma general (a todo el curso) materiales que puedan llegar a arruinar la comprensión o el desarrollo propio de cada persona (por ej. compartir soluciones a ejercicios fuera del grupo de trabajo). Este balance será vital, fomentando el intercambio y la discusión, lo cual genera un aprendizaje más significativo, pero nunca perdiendo de vista que debe haber un progreso y aprendizaje individual en las competencias a adquirir.

Los conocimientos serán constantemente cotejados y validados, tanto en las clases prácticas/encuentros virtuales sincrónicos mediante el intercambio entre pares y con los docentes, como también a través de una serie de actividades autoevaluables en el campus virtual.

En cuanto a la dinámica de las clases presenciales/encuentros virtuales sincrónicos se plantea una idea de clase principalmente práctica, en donde las/los estudiantes podrán realizar consultas sobre las diversas actividades que han realizado y cotejar su progreso con el docente. Adicionalmente cada docente podrá determinar la necesidad de resolver en forma conjunta ciertos ejercicios, realizar puestas en común sobre alguna actividad u otros. Además podrá repasar y profundizar sobre conceptos teóricos que considere no han quedado claros. La asistencia a las clases se considera obligatoria, pues se estará evaluando de forma constante el progreso de los estudiantes.

El enfoque será siempre el de orientar a solucionar las actividades en forma personal, en contraposición a presentar soluciones estandarizadas, pues la capacidad de solucionar problemas de forma autónoma responde directamente a uno de los objetivos buscados, y es clave para el desarrollo del estudiante en futuras materias y en su vida profesional. Así, el equipo docente optará siempre por presentar soluciones con distintas estrategias, debatir beneficios y problemáticas, dando lugar al debate constante y dejando en claro que en programación no existen soluciones únicas. Por este motivo se evitará la divulgación de soluciones posibles a ejercicios de forma escrita, optando siempre por la solución en clase. Esto incluye a los exámenes, tanto grupales como individuales.

Con respecto a los exámenes podemos mencionar que los mismos se consideran tanto una instancia de acreditación como de aprendizaje, priorizando esta última en todo momento. Por este motivo se fomentará la devolución puntual del examen, de forma presencial cuando sea posible, y discutiendo en la devolución escrita los distintos errores y aciertos cometidos en la resolución. En ese sentido se fomentará el acompañamiento de aquellas personas que no hayan logrado objetivos mínimos en los momentos de acreditación, realizando charlas individuales, y manteniendo un seguimiento personalizado a las mismas.

Plan de trabajo en el Campus virtual

El Campus Virtual es un espacio fundamental para el desarrollo de la asignatura. El mismo se utilizará como espacio de seguimiento fuera de los espacios presenciales/sincrónicos, repositorios de materiales y actividades auto-asistidas.

El espacio áulico virtual contará al menos con:

- Un canal de comunicación unidireccional con estudiantes (Foro de avisos)
- Enlaces a las herramientas digitales utilizadas (Gobstones / Discord).
- 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.

En todos los casos mencionamos “enlaces” para referirnos a vínculos externos o a material embebido en el aula virtual, cualquiera fuera el caso, según conveniencias técnicas. Se priorizará siempre el contenido embebido en la medida de lo posible.

Las actividades en el campus se encontrarán organizadas en forma de unidades temáticas (según las unidades mencionadas en el programa analítico), y no serán discriminadas por comisiones o grupos de estudiantes. El acceso a las distintas unidades se encontrará limitado por fechas según la planificación calendárica para el cuatrimestre en curso. Las actividades dentro de una misma unidad se limitarán en acceso en base a la previa finalización de actividades anteriores, para garantizar la secuencia indagación-teoría-práctica propuesta como metodología de enseñanza.

Las actividades del campus se considerarán parte del trabajo que cada estudiante deberá realizar en tiempo adicional al de clase, en su casa, y como parte de la propuesta de clase invertida. Completar estas actividades se considerará obligatorio y se tendrán en consideración tanto para nota conceptual como para computar la asistencia final a clases y recorrido del curso.

Otras herramientas físicas y digitales requeridas

La materia requiere el uso de computadoras en los espacios áulicos, a razón de una por estudiante o par de estudiantes. En los casos de actividades virtuales, es necesario contar con computadora de escritorio, notebook o netbook (siendo tablet y teléfonos celulares una alternativa parcial en algunas actividades, pero no en los momentos de

actividades prácticas o evaluaciones). Para quienes no cuenten con un equipamiento adecuado, podrán hacer uso de espacios comunes en las aulas de la universidad.

Adicionalmente al campus virtual la materia utilizará la herramienta de programación Gobstones (que provee tanto un entorno para la propuesta de indagación como para la propuesta práctica). Esta herramienta es software libre y de acceso gratuito, desarrollada en Argentina por la Universidad Nacional de Quilmes, y pensada para la enseñanza inicial de la programación tanto en ámbitos de escuela media como en primer encuentro con la disciplina en nivel universitario.

Fuera de eso, podría ser necesario el uso de otras herramientas y plataformas, como plataformas de almacenamiento y reproducción de vídeos (YouTube o similar), herramientas de teleconferencia (Google Meet, Zoom o similar), herramientas de manejo de comunicaciones grupales de texto y voz (Discord o similar) y eventualmente otras. Se priorizará el uso del campus virtual con materiales embebidos y herramientas de producción local y de software libre cuando sea posible.

6. Evaluación y régimen de aprobación:

El régimen de aprobación de la materia Régimen Académico vigente, adecuando la cantidad de exámenes, instancias de recuperación, y otros elementos de la materia al mismo. Los siguientes regímenes de aprobación de cursada y materia cumplen los requisitos impuestos en el régimen académico al momento de elaboración de este programa, pero podrían cambiar si el régimen académico lo requiriese.

6.1 Aprobación de la cursada

La cursada se organiza en torno a la participación directa de las/los estudiantes en los espacios presenciales/encuentros virtuales sincrónicos. La participación en esos espacios se considera obligatoria. También es obligatoria la realización de las actividades propuestas como autoevaluación y seguimiento en el aula virtual, tanto para cursadas presenciales como virtuales.

En cuanto a los conocimientos, estos se deberán acreditar en el transcurso de la cursada durante las cinco evaluaciones parciales que presenta la materia, de los cuales, los cuatro primeros serán de carácter grupal (3 o 4 estudiantes, sin posibilidad de realización individual) y el último será de carácter individual. En los exámenes grupales se deberá acreditar una serie de saberes o competencias. En el examen individual deberán validar las mismas competencias y obtener una nota igual o superior a 4.

Cada examen contará con su correspondiente instancia de recuperación, tanto en los casos grupales, como en el caso individual. Adicionalmente algunas competencias podrán acreditarse tanto en el examen, en su instancia de recuperación o en otras instancias evaluatorias que puedan actuar adicionalmente como recuperación de instancias previas.

Así, para la aprobación de la cursada se requerirán los siguientes:

- Acreditación del 75% de asistencia a las clases presenciales/encuentros virtuales sincrónicos.

- Acreditación del 70% de las actividades en el campus virtual.
- Acreditación de la totalidad de las competencias en cada instancia de examen grupal, su correspondiente instancia de recuperación o en otras instancias de examen que permitan validar esas competencias (a determinar según indicación del equipo docente).
- Acreditación de la totalidad de las competencias en la instancia de examen individual, o en su correspondiente instancia de recuperación.
- Acreditación de una nota igual o superior a 4 en la instancia de examen individual.

Todas las instancias de evaluación son con materiales a la vista, hacinedo uso de materiales que haya desarrollado durante el transcurso de la actividad para una más rápida y satisfactoria resolución del mismo.

Dependiendo de los resultados en la evaluación final, se podrá acreditar la cursada o aprobar directamente la materia por promoción.

6.2 Aprobación de la materia

La materia se podrá aprobar mediante régimen de promoción directa, examen integrador, examen final o examen libre.

Promoción directa: El estudiante deberá aprobar la cursada de la materia con una nota no inferior igual o superior a siete (7) puntos entre las instancias de evaluación grupal y la individual y hayan validado la totalidad de las competencias en el mismo, o en sus correspondientes recuperatorios.

Evaluación integradora: Podrán acceder a esta evaluación aquellos estudiantes que hayan aprobado la cursada con una nota de entre cuatro (4) y seis (6).

La evaluación integradora tendrá lugar por única vez en el primer llamado a exámenes finales posterior al término de la cursada. Se realizará el mismo día y horario de la cursada y será administrado, preferentemente, por el 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 fallen en aprobar la materia en dicha instancia, pero hayan regularizado la materia en cuatrimestres anteriores. Serán exclusivamente presenciales e individuales, con materiales a la vista. Contarán con una parte escrita y una defensa oral (solo en caso de una parte escrita satisfactoria). La evaluación tendrá en consideración 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.

Examen libre: Instancia destinada a quienes opten por no realizar la cursada. Serán exclusivamente presenciales (sin posibilidad de instancia virtual) e individuales, sin materiales a la vista. Contarán con una parte teórica escrita, una parte práctica escrita (solo en caso de parte teórica satisfactoria) y una defensa oral (solo en caso de una parte práctica escrita satisfactoria). La evaluación tendrá en consideración 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.

6.3 Criterio de calificación.

El estudiante deberá acreditar las siguientes competencias, en dos instancias (una grupal durante el transcurso de la materia) y una individual (en el examen individual de la cursada, la instancia integradora o examen final o libre).

- Uso y escritura de código que usa secuenciación.
- Manejo de documentación de programas.
- Manejo de precondiciones y sus implicancias.
- Capacidad de elección de nombres apropiados.
- Manejo de división en subtareas, creación de procedimientos y funciones.
- Capacidad de comunicación de estrategias mediante subtareas.
- Manejo de repetición simple.
- Uso y definición de parámetros.
- Uso de expresiones de distintos tipos de forma correcta.
- Uso de alternativa condicional en comandos y en expresiones.
- Uso de abstracción sobre representaciones arbitrarias.
- Manejo de repetición condicional y estrategias para garantizar terminación.
- Manejo de variables, inicialización y asignación, en distintas formas.

Estas competencias podrán acreditarse directamente, o deberán acreditarse en etapas según sea el caso. El equipo docente determinará una vía de obtención de las competencias que sea apropiada según el calendario académico en curso.

La calificación del examen individual se determinará en la escala 0 a 10, con los siguientes valores: 0, 1, 2 y 3: insuficientes; 4 y 5 regular; 6 y 7 bueno; 8 y 9 distinguido; 10 sobresaliente.

7 Cronograma

El siguiente es un cronograma propuesto para la materia. El mismo deberá ser revisado y adaptado según el calendario académico en curso y podrá estructurarse en diferentes modalidad distribuyendo las horas semanales de cursadas en modalidades presenciales, virtuales o mixtas.

Unidad	Semana	Recuperación
Unidad 0 Bienvenida a la materia.	Semana 1	
Unidad 1: Programas y comandos básicos	Semana 2	
Unidad 2: Procedimientos	Semanas 2 y 3	
Examen Grupal 1: Contenidos de Unidad 1 y 2	Última clase de Semana 3	Última clase de Semana 7
Unidad 3: Repetición Simple	Semana 4	

Unidad 4: Parámetros	Semana 4 y Semana 5	
Unidad 5: Expresiones y Tipos	Semana 6	
Examen Grupal 2: Contenidos de Unidad 1 y 2	Último día de Semana 6	Última clase de Semana 10
Unidad 6: Alternativa Condicional	Semana 7	
Unidad 7: Funciones Simples	Semana 8	
Examen Grupal 3: Contenidos de Unidad 1 y 2	Última clase de Semana 8	Última clase de Semana 12
Unidad 8: Repetición Condicional	Semana 9	
Unidad 9: Variables y Funciones con Procesamiento	Semana 10 y Semana 11	
Examen Grupal 4: Contenidos de Unidad 1 y 2	Última clase de Semana 11	
Unidad 10: Tipos de Datos Personalizados	Semana 12 y Semana 13	
Examen Individual 1: Contenidos integrados de todas las unidades.	Última clase de Semana 13	Última clase de Semana 15