INSTITUTO/S: Tecnología e Ingeniería

CARRERA/S: Tecnicatura Universitaria en Programación / Tecnicatura Universitaria en Programación de Videosjuegos.

MATERIA: Programación estructurada

NOMBRE DEL RESPONSABLE DE LA ASIGNATURA: Churruca, Emiliano

EQUIPO DOCENTE: Churruca, Emiliano; Cornejo, Eliana; Del Castillo, Inés; Gauna, Alan; Lera, Gustavo; Merlo, Ailín; Quiroga, Hernán; Rios, Jesús; Salina, Mauro; Spigariol, Lucas

CUATRIMESTRE: 2do

AÑO: 1ro

PROGRAMA N°: 6 **(Aprob. Por Cons.Directivo fecha / / 2023)**

Instituto/s: Tecnología e Ingeniería
Carrera/s: Tecnicatura Universitaria en Programación/ Tecnicatura Universitaria en Programación de Videosjuegos.
Nombre de la materia: Programación estructurada
Responsable de la asignatura y equipo docente: Churruca, Emiliano; Cornejo, Eliana; Del Castillo, Inés; Gauna, Alan; Lera, Gustavo; Merlo, Ailín; Quiroga, Hernán; Rios, Jesús; Salina, Mauro; Spigariol, Lucas-
Cuatrimestre y año: 2do cuatrimestre del 1er año
Carga horaria semanal: 6 hs
Programa N°: 6
Código de la materia en SIU: 792

Programación estructurada

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/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.

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.

2. Propósitos y/u objetivos

Propósitos

- 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.

Objetivos

- Conozca 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).
- Pueda 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.
- Comprenda el concepto de tipo de datos, y pueda aprovecharlo para organizar el estado y las transformaciones de información de los programas que construye.
- Logre 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.
- Sea 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.
- Maneje 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. 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. 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 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.

4.2 Bibliografía y recursos obligatorios:

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

4.3 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.
- **Wirth, Nikolas**. (1997). Systematic Programming: An introduction, Prentice Hall, Englewood Cliffs (NJ).

5. Metodología 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 los/las estudiantes 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 tercera y última parte se presentará una serie de ejercicios prácticos en donde los/las estudiantes deberán 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 sintaxis de lenguajes o conceptos teóricos complejos, y permitiendo que sea el/la estudiante quien arribe 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 el/la estudiante 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. 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 los/las estudiantes, permitiendo el trabajo colaborativo y el debate entre los miembros del grupo, con el acompañamiento de los/las 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 estudiante (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 los/las estudiantes podrán realizar consultas sobre las diversas actividades que han realizado y cotejar su progreso con el docente. Adicionalmente el/la docente podrá determinar la necesidad de resolver en forma conjunta ciertos ejercicios, realizar puestas en común sobre alguna actividad u otros. Además, el/la docente 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 lo/las estudiantes.
El enfoque será siempre el de oriental al/la estudiante a solucionar las actividades por su cuenta, 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/la 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 solución del mismo. En ese sentido se fomentará el acompañamiento de aquellas personas que no hayan logrado objetivos mínimos en los momentos de acreditación, realizando charlas individuales con esas personas, y manteniendo un seguimiento personalizado a las mismas.

Plan de trabajo en el campus:

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).
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.
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 el/la estudiante deberá realizar en tiempo adicional al de clase, en su casa, y como parte de la propuesta de clase invertida. Las finalizaciones de estas actividades se considerarán obligatorias y se tendrán en consideración tanto para nota conceptual como para computar la asistencia final a clases y recorrido del curso.

6. Actividades de investigación y extensión (si hubiera)

No Aplica

7. Evaluación y régimen de aprobación

7.1 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.
El/La alumno/a deberá poseer una asistencia no inferior al 75% en las clases presenciales.
En cuanto a las cursadas de materias virtuales se requerirá que el estudiante ingrese al aula virtual como mínimo una vez por semana.

7.2 Aprobación 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/as estudiantes que hayan aprobado la 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 administrado, 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.

7.3 Criterios de calificación

La calificación de cada evaluación se determinará en la escala 0 a10, con los siguientes valores: 0, 1, 2 y 3: insuficientes; 4 y 5 regular; 6 y 7 bueno; 8 y 9 distinguido; 10 sobresaliente.

8. Cronograma

Unidad	Semana	Recuperación	Modalidad
Unidad 0: Bienvenida a la materia. **Unidad 1:** Programas y comandos básicos	Semana 1		Presencial / Virtual
Unidad 2: Procedimientos **Unidad 3:** Repetición Simple	Semana 2		Presencial / Virtual

Unidad 4: Parámetros	Semana 3 y Semana 4		Presencial / Virtual
Unidad 5: Expresiones y Tipos	Semana 5		Presencial / Virtual
Unidad 6: Alternativa Condicional	Semana 6		Presencial / Virtual
Unidad 7: Funciones Simples	Semana 7		Presencial / Virtual
Unidad 8: Repetición Condicional	Semana 8		Presencial / Virtual
Unidad 9: Variables y Funciones con Procesamiento	Semana 9 y Semana 10		Presencial / Virtual
1° parcial: Contenidos de Unidad 1a 9	Semana 11	Semana 13	Presencial
Unidad 10: Tipos de Datos Personalizados	Semana 12 y Semana 13		Presencial / Virtual
2° parcial: Contenidos de Unidad 10	Semana 14	Semana 16	Presencial