

**UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS**

PLANIFICACIÓN DE LA ASIGNATURA:

PROGRAMACIÓN II

EQUIPO CÁTEDRA

Ing. Graciela Barchini de Giménez

grael@unse.edu.ar

Ing, Margarita Alvarez de Benítez

malvarez@unse.edu.ar

AÑO 2008

1. IDENTIFICACIÓN

1.1. ASIGNATURA: PROGRAMACIÓN II

1.2. CARRERA: LICENCIATURA EN SISTEMAS DE INFORMACIÓN

1.3. UBICACIÓN DE LA ASIGNATURA:

1.3.1. 7º MÓDULO- 4º AÑO CICLO SUPERIOR

1.3.2. CORRELATIVAS ANTERIORES:

- MÉTODOS NUMÉRICOS (REGULAR)
- LÓGICA II Y PROGRAMACIÓN I (APROBADAS)

1.3.3. CORRELATIVAS POSTERIORES: -----

1.4. CONTENIDOS MÍNIMOS ESTABLECIDOS EN EL PLAN DE ESTUDIOS

Teoría de la computabilidad. Función computable. Complejidad de algoritmos. Clasificación de problemas. Gramáticas de estructuras de frase. Jerarquía de gramáticas y lenguajes. Teoría de autómatas. Autómatas finitos. Autómatas de pila. Máquinas de Turing. Lenguajes de programación. El proceso de traducción. Compiladores. Estructura. Analizadores léxico, sintáctico y semántico. Tabla de símbolos. Generación y optimización de código. Intérprete.

1.5. CARGA HORARIA SEMANAL Y TOTAL: 6 HORAS / 168 HORAS

1.6. AÑO ACADÉMICO: 2008

2. PRESENTACIÓN

2.1. UBICACIÓN de LA ASIGNATURA

Esta asignatura se ubica en el tramo final de la línea curricular iniciada con Fundamentos de la Programación. Se aborda el núcleo teórico fundamental de las Ciencias de la Computación: la teoría de la computabilidad.

Esta teoría estudia los problemas de decisión que pueden ser resueltos por un algoritmo (o equivalentemente por una máquina de Turing) y explora las limitaciones de las computadoras al establecer qué tipos de problemas pueden ser resueltos por una máquina.

Mediante el estudio de la teoría de la complejidad algorítmica el alumno dispone de un marco de referencia para clasificar problemas y saber qué modelo de cálculo requieren y lograr diseñar algoritmos eficientes, según el tipo de problemas.

Se ofrece una introducción a la Teoría de Lenguajes Formales y Autómatas, cuyo núcleo consiste en la jerarquía de lenguajes definida por Chomsky y sus modelos de representación.

Paralelamente a la jerarquía de lenguajes existe otra equivalencia de máquinas abstractas, de tal forma que a cada una de las clases de lenguajes definidas en la jerarquía de Chomsky, a partir de restricciones impuestas a las gramáticas, le corresponde un tipo de máquina abstracta, que no es otra cosa que un método reconocedor de lenguajes.

Esta asignatura proporciona la justificación teórica de los instrumentos necesarios para la construcción de compiladores.

Además, las ideas y técnicas correspondientes al tema Compiladores pueden ser utilizadas en el desarrollo de software de aplicación general.

El valor formativo de todos los conceptos desarrollados en la asignatura es esencial en dos aspectos: para sustentar una posterior profundización de la Teoría de la Computabilidad y el diseño de Compiladores e Intérpretes y además para proveer una base sólida para el cabal ejercicio profesional.

2.2. CONOCIMIENTOS Y HABILIDADES PREVIAS

Los prerequisites para el abordaje de esta asignatura tiene que ver con los conocimientos y habilidades adquiridos con la Teoría de Conjuntos, el cálculo proposicional y la programación.

3. OBJETIVOS

3.1. OBJETIVOS GENERALES

Que el estudiante logre conocer, comprender y manejar conceptos y técnicas vinculados con la Teoría de la Computabilidad, Teoría de Lenguajes y la Teoría de Autómatas.

3.2. OBJETIVOS ESPECÍFICOS

Que los estudiantes logren:

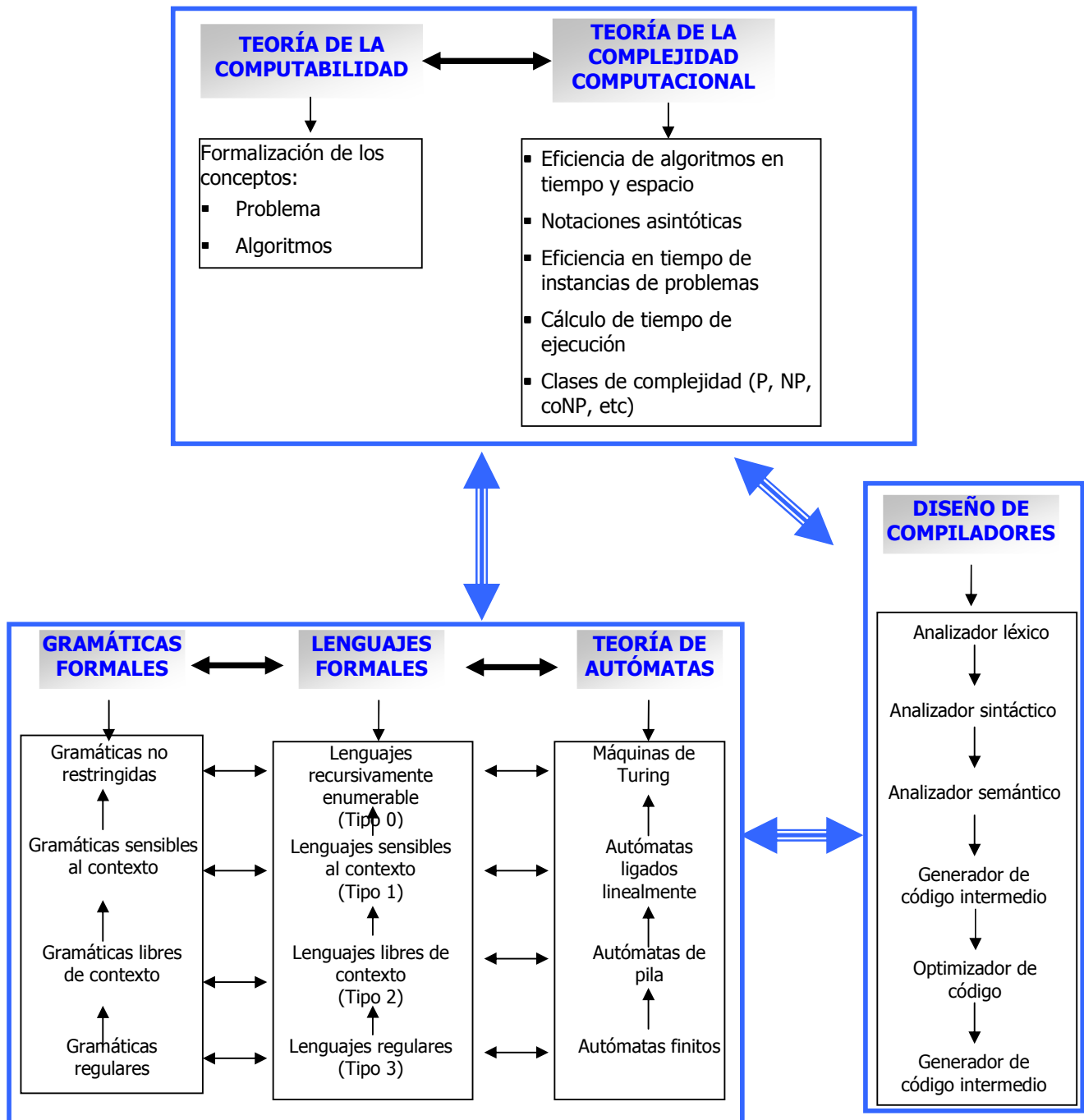
- a) Identificar los conceptos constructivos de la Teoría de Algoritmos.
- b) Habilidad para formalizar problemas.
- c) Capacidad para determinar la eficiencia de los algoritmos.
- d) Habilidad para generar y reconocer gramáticas y lenguajes de distinto tipo.
- e) Determinar las equivalencias entre los lenguajes formales, las gramáticas formales, las expresiones regulares y las máquinas reconocedoras
- f) Capacidad para construir compiladores/intérpretes sencillos.

4. PROGRAMACIÓN DE CONTENIDOS

4.1. PROGRAMA SINTÉTICO

1. Teoría de la Computabilidad. Formalización, representación y solución de problemas. Función computable. Teoría de la Complejidad Computacional. Complejidad y eficiencia de algoritmos. Clasificación de problemas.
2. Teoría de los Lenguajes Formales. Definición y conceptos. Gramáticas de estructura de frase. Jerarquía de los lenguajes y gramáticas. Características de las gramáticas.
3. Teoría de Autómatas. Autómatas finitos. Autómata a pila. Máquina de Turing. Interpretaciones: como reconocedora de lenguajes y como procedimiento.
4. Diseño de Compiladores: Analizadores lexicográfico, sintáctico y semántico. Generación de código intermedio, Optimización y generación de código objeto.

4.2. ARTICULACIÓN TEMÁTICA DE LA ASIGNATURA



4.3. PROGRAMA ANALÍTICO

1. TEORÍA DE LA COMPUTABILIDAD

- (i) **PROBLEMA.** Concepto. Formalización y Solución. Procedimientos y algoritmos.
- (ii) **CONCEPTOS FUNDAMENTALES DE LA TEORÍA:** Función computable. Gödelización.
- (iii) **COMPLEJIDAD Y EFICIENCIA DE ALGORITMOS:** Teoría de la Complejidad Computacional. Análisis de algoritmos. Determinación del orden de un algoritmo. Dominación asintótica. Orden de un algoritmo $O(p(n))$ y $O(c^n)$. Principios y consideraciones para la determinación del orden. Análisis de algoritmos no recursivos y recursivos.
- (iv) **CLASIFICACIÓN DE PROBLEMAS:** Problemas demostrablemente insolubles, demostrablemente difíciles. Clase P, Clase NP, NP completa y CO-NP.

2. TEORÍA DE LENGUAJES FORMALES

- (i) **CONCEPTOS BÁSICOS:** Símbolo, alfabeto, palabra, subhilera. Operaciones con lenguajes: unión, intersección, concatenación. Propiedades de las operaciones.
- (ii) **GRAMÁTICAS FORMALES:** definición, tipos de gramáticas. Jerarquía de Chomsky. Gramáticas no restringidas, contextuales, incontextuales y regulares (tipo 3). Lenguajes generados por cada tipo de gramáticas.
- (iii) **CARACTERÍSTICAS DE LAS GRAMÁTICAS:** Recursividad en las gramáticas contextuales. Gramáticas libres de contexto: árbol de derivación. Derivaciones a izquierda y a derecha. La ambigüedad. Gramáticas propias.
- (iv) **GRAMÁTICAS REGULARES.** Expresiones regulares. Propiedades y equivalencias. Obtención de una gramática regular a partir de una expresión regular.

3. TEORÍA DE AUTÓMATAS

- (i) **AUTÓMATAS FINITOS:** Definición y representación gráfica. El autómata finito como reconocedor de lenguajes. Autómata finito determinista y no determinista. Equivalencia. Minimización de autómatas finitos deterministas.
- (ii) **AUTÓMATAS FINITOS Y EXPRESIONES REGULARES.** Método de ecuaciones para la obtención de un autómata finito. Autómatas finitos con movimientos lambda. Equivalencia entre los autómatas finitos y las expresiones regulares. Método para construir autómata finito a partir de una expresión regular.
- (iii) **AUTÓMATA DE PILA:** Definición formal. Autómata de pila como reconocedor de un lenguaje. Autómata a pila determinístico y no determinístico.
- (iv) **MÁQUINAS DE TURING:** Definición formal. Representación. Interpretaciones de las computaciones. Configuración de una máquina de Turing. Máquina de Turing multicinta. Máquina universal de Turing. Codificación de una máquina de Turing. La insolubilidad del problema de la parada.

4. DISEÑO DE COMPILADORES

- (i) **TIPO DE TRADUCTORES:** Ensambladores, compiladores e intérpretes. Características de cada uno. Comparación. Análisis y síntesis de la compilación. Herramientas para la construcción de compiladores.
- (ii) **ANÁLISIS LÉXICO:** Componentes léxico, patrones y lexemas. Especificación y reconocimiento de tokens. Definiciones regulares. Diagrama de transición.

- (iii) **ANÁLISIS SINTÁCTICO:** Métodos descendentes y ascendentes. Manejo de errores sintácticos. Estrategia. Analizador sintáctico descendente: gramáticas LL(k), por descenso recursivo, analizador sintáctico predictivo y analizador sintáctico predictivo no recursivo. Gramáticas LR(k). Implantación de un analizador sintáctico ascendente mediante poda. Analizador sintáctico por precedencia de operadores. Analizadores Sintácticos LR. Métodos para analizador sintáctico LR.
- (iv) **ANÁLISIS SEMÁNTICO:** Traducción dirigida por la sintaxis. Definición dirigida por la sintaxis. Reglas semánticas. Árbol sintáctico para expresiones. Grafos dirigidos acíclicos para expresiones (GDA). Esquema de traducción. Acciones semánticas. Notación postfija. Comprobación de tipos.
- (v) **GENERACIÓN DE CÓDIGO:** generación de código Intermedio distinto tipo de representaciones. Optimización de Código Intermedio. Generación de Código Objeto.

4.4. PROGRAMA Y CRONOGRAMA DE TALLERES Y PRÁCTICAS

4.4.1. PROGRAMA DE TALLERES Y PRÁCTICAS

	OBJETIVOS	TEMÁTICA	CRITERIOS DE EVALUACIÓN
TALLER 1	<ul style="list-style-type: none"> • Conocer los temas fundamentales de la Teoría de la Computabilidad. • Seleccionar el algoritmo adecuado en el proceso de resolución de problemas • 	Parte A: Ejercicios del tipo “verdadero-falso” y/o “múltiple Choice” con justificación Parte B: <ul style="list-style-type: none"> • Resolución de problemas • Actividades de investigación 	<ul style="list-style-type: none"> • Presentación • Claridad en exposición • Completitud • Manejo conceptual y metodológico • Manejo bibliográfico • Originalidad
TALLER 2	<ul style="list-style-type: none"> • Conocer los temas fundamentales de la Teoría de lenguajes • Caracterizar las gramáticas de estructura de frase • Sistematizar e integrar los conocimientos adquiridos 		
	<ul style="list-style-type: none"> • Conocer los temas fundamentales de la Teoría de Autómatas • Caracterizar los autómatas • Sistematizar e integrar los conocimientos adquiridos. 		
TALLER 3	<ul style="list-style-type: none"> • Introducir al alumno en los problemas concretos que se enfrentan durante el desarrollo de programas traductores. • Reforzar, desde el punto de vista práctico, los conocimientos adquiridos sobre analizadores lexicográficos, sintácticos y estructura de programas traductores. • Capacidad para construir compiladores/intérpretes sencillos. 	Diseño y construcción de un interprete	<ul style="list-style-type: none"> • Presentación • Completitud • Funcionamiento • Programación en lenguaje C • Manejo de errores • Diseño de pantallas • Guía del usuario • Originalidad
Cuadernillo de ejercicios de aplicación	Habilidad para: <ul style="list-style-type: none"> • Formalizar y clasificar problemas • Analizar la eficiencia de algoritmos • Reconocer y utilizar gramáticas de estructura de frase. • Reconocer y utilizar autómatas 	Ejercicios correspondientes a los temas de las 4 (cuatro) unidades	Grado de concordancia con las respuestas requeridas

4.4.2. CRONOGRAMA DE TALLERES Y PRÁCTICAS

MESES	DESARROLLO	DESARROLLO Y PRESENTACIÓN
ABRIL -MAYO	Cuadernillo de ejercicios de aplicación	TALLER 1
JUNIO - AGOSTO		TALLER 2
AGOSTO - NOVIEMBRE		TALLER 3

5. BIBLIOGRAFÍA

5.1. BIBLIOGRAFÍA GENERAL

- Aho A., Hopcroft J. y Ullman J. - *Estructura de Datos y Algoritmos*. Addison-Wesley Iberoamericana, 1988.
 - *The design and analysis of computer algorithms*, Addison-Wesley, Reading, MA., 1974
- Ángel Ernest y Newman J. *El Teorema de Godel*. Consejo Nacional de Ciencia y Tecnología, 1981.
- Bennett, J. P.: Introduction to Compiling Techniques. McGraw-Hill. 1996.
- Hermes, Hans *Introducción a la Teoría de la Computabilidad. Algoritmos y Maquinas*. Editorial Tecnos, 1984.
- Hopcroft, J.E. & Ullman, J. *Introduction to Automata Theory, Languages and Computation*. Addison-.Wesley. 1979.
- Knuth, Donald E. *Algoritmos Fundamentales . Vol I*.
- Levine, J. ; Mason, T. y Brown *Lex & Yacc*.. O'Reilly & Associates, 1.995.
- Lewis y Papadimitriou - *La Eficiencia de los Algoritmos. Maquinas de Turing*. Publicaciones de la revista Ciencia e Investigación
 - *Elements of the theory of computation*, Prentice Hall, 1981
- Presser, Cardenas y Marin *Ciencia de la Computación Vol. II*. Editorial Limusa - Wiley, S.A., 1972.
- Ritchie, Dennis; Kernighan, Brian W; *El Lenguaje de Programación C*. Editorial Prentice-Hall Hispanoamericana, S.A. 1975.
- San Roman, Sancho *Lógica Matemática y Computabilidad*, Díaz De Santos S. A. Madrid, 1990.
- Sanchez, Llorca y Valverde *Compiladores*. Teoría y Práctica.
- Trakhtenbrot, B.A. *Algoritmos y Computadoras*. Editorial Limusa, 1973.

5.2. BIBLIOGRAFÍA ESPECÍFICA

- Aho, Alfred; Ullman, Jeffrey y Sethi, Ravi *Compiladores, Principios, Técnicas y Herramientas*. Editorial Addison Wesley Iberoamericana. 1986.
- Alfonseca Manuel, Sancho, Justo y Martínez Orga *Teoría de Lenguajes, Gramáticas Y Autómatas*. Ediciones Universidad y Cultura, 1990.
- Barchini, Graciela y Alvarez Margarita *Fundamentos Teóricos de la Ciencia de la Computación*, Departamento de Informática. FCEyT 1994 y 1998.
- Baum, Gabriel *Complejidad*. I EBAI. Editorial Kapelusz S.A. 1987.
- Kelly Dean *Teoría de Autómatas y Lenguajes Formales*. Editorial Prentice Hall, 1995.
- Mandrioli Dino y Ghezzi Carlo *Theoretical Foundations of Computer Science*. John Wiley & Sons, 1987.
- Sagastume, Marta y Baum *Problemas, lenguajes y algoritmos"*. I EBAI.
- Schildt, Herbert *Programación en Turbo C*. Editorial Borland-Oshore-Mcgraw-Hill. 1990.

5.3. DOCUMENTOS Y SITIOS WEB

Comon, Hubert; Dauchet Max y otros *Tree Automata Techniques and Applications*. Disponible en URL: <<http://www.grappa.univ-lille3.fr/tata/tata.pdf>>. [Acceso en marzo de 2006].

Crenshaw, Jack *Let's Build a Compiler*. Disponible en URL: <<http://www.maththinking.com/boat/booksIndex.html>>. [Acceso en marzo de 2007].

Grune, Dick and Ceriel J.H. *Parsing Techniques - A Practical Guide*. Originally published by Ellis Horwood, Chichester, England, 1990. Disponible en URL: <<http://www.cs.vu.nl/~dick/PTAPG.html>>. [Acceso en marzo de 2007].

Gurari, Eitan. *An Introduction to the Theory of Computation*. Ohio State University Computer Science Press, 1989, ISBN 0-7167-8182-4 Disponible en URL: <<http://www.maththinking.com/boat/booksIndex.html>>. [Acceso en febrero de 2007].

Papadimitriou, Christos H. *NP-completeness: A Retrospective - ICALP 97*. Springer LNCS. Disponible en URL: <<http://www.cs.berkeley.edu/~christos/>>. [Acceso en marzo de 2007].

Parberry, Ian. *Lecture Notes on Algorithm Analysis and Computational Complexity*. Disponible en URL: <<http://hercule.csci.unt.edu/ian/books/free/lnoa.pdf>>. [Acceso en marzo de 2004].

Wilf, Herbert S. *Algorithms and Complexity*. Internet Edition, Summer, 1994. Disponible en URL: <<http://www.maththinking.com/boat/booksIndex.html>>. [Consultada en marzo de 2006].

6. EVALUACIÓN

6.1. EVALUACIÓN DIAGNÓSTICA

La primera unidad cumple el doble rol de constituir un elemento motivador para incursionar en los fundamentos de la ciencia de la computación y, por otra parte, permite a los docentes disponer de un instrumento para diagnosticar la capacidad de los alumnos para diseñar algoritmos y programas.

6.2. EVALUACIONES PARCIALES

Del resultado de cada una de las **EVALUACIONES PARCIALES** previstas se obtiene:

- **EVALUACIÓN SUMATIVA (PARCIAL)** referida a la puntuación-calificación que se le da a cada alumno sobre la base de los resultados de la prueba.
- **EVALUACIÓN FORMATIVA:** sobre la base de los resultados de todos los alumnos se puede determinar como se encuentra la clase con respecto a aspectos, dimensiones cognitivas y /o conductas previstas en el examen.

6.2.1. PROGRAMA Y CRONOGRAMA DE EVALUACIONES PARCIALES

- **EVALUACIÓN PARCIAL N° 1**

Estructura	Objetivo	Ponderación en la calificación
a) Ejercicio para formalizar problemas y determinar la eficiencia de un algoritmo.	Determinar si los alumnos han adquirido habilidad para formalizar problemas y capacidad para determinar la eficiencia de los algoritmos.	40%
b) Ejercicio para determinar tipo de gramática	Determinar si los alumnos han adquirido habilidad para generar y reconocer gramáticas y lenguajes de distinto tipo.	40%
c) Ejercicio para aplicar algún método desarrollado en la unidad 2.	Determinar si los alumnos son capaces de aplicar los algoritmos estudiados	20%

- **EVALUACIÓN PARCIAL N° 2**

Estructura	Objetivo	Ponderación en la calificación
a) Ejercicio para aplicar algún método desarrollado en las unidad 3.	Determinar si los alumnos son capaces de aplicar los algoritmos estudiados	20%
b) Ejercicio para construir un	Determinar si los alumnos son capaces de	40%

autómata	construir autómatas	
c) Ejercicio para construir un analizador sintáctico	Determinar si los alumnos son capaces de aplicar los algoritmos estudiados	40%

MES	SEMANA	PRÁCTICA
JUNIO	1° semana	Evaluación Parcial N° 1
	2° semana	Recuperatorio Parcial N° 1
OCTUBRE	3° semana	Evaluación Parcial N° 2
	4° semana	Recuperatorio Parcial N° 2

6.2.2. CRITERIOS DE EVALUACIÓN

Se califica cada evaluación con APROBADO o DESAPROBADO. El puntaje mínimo para aprobar los parciales es de **cincuenta (50) puntos (sobre una calificación máxima de 100)**. Se otorga una sola recuperación en caso de desaprobación. Si se obtiene como **mínimo setenta (70) puntos** se puede acceder al **Sistema de Promoción con Coloquio Final**.

6.3. AUTOEVALUACIÓN

La autoevaluación se realiza desde dos perspectivas:

- A partir de los resultados obtenidos de la evaluación formativa y sumativa (Evaluación Parcial N° 1, Evaluación Parcial N° 2 y Talleres)
- A partir del resultado de la evaluación realizada por los alumnos a la cátedra (encuesta diseñada a ese efecto)

6.4. CONDICIONES PARA LOGRAR LA PROMOCIÓN CON COLOQUIO FINAL

- Asistir como mínimo al 75 % del total de sesiones.
- Aprobar los talleres con una calificación ≥ 80 puntos.
- Aprobar las evaluaciones parciales con un puntaje ≥ 70 puntos
- Certificado de aprobación del taller de lenguaje C.

6.5. CONDICIONES PARA LOGRAR LA REGULARIDAD

- Asistir como mínimo al 75 % del total de sesiones.
- Aprobar el taller 3.
- Aprobar las evaluaciones parciales o los recuperatorios.
- Presentar certificado de aprobación del taller de lenguaje C.

6.6. EXAMEN FINAL

La evaluación final será escrita u oral sobre los temas incluidos en la programación de la asignatura.

6.7. EXAMEN LIBRE

Prerrequisitos: a) Presentación, prueba y defensa del taller 3 que deberá ser solicitado con una anticipación de 20 (veinte) días. Duración 2 hs. b) Certificado de aprobación del taller de lenguaje C.

Deberán aprobar las evaluaciones y trabajos correspondientes a las siguientes etapas y subetapas cada una de ellas eliminatorias:

Primera etapa: Evaluación escrita y oral.

- Temas a desarrollar, ejemplos y problemas correspondientes a las Unidades 1 y 2 . Duración 4 hs. Modalidad escrita.
- Temas a desarrollar, ejemplos y problemas correspondientes a las Unidades 3 y 4. Duración 4 hs. Modalidad escrita.

Duración total 8 (ocho) hs. repartidas en 3 días.

Segunda etapa: Evaluación oral.

Se utilizará la misma modalidad que se utiliza para los alumnos regulares.

Ms. Ing. Graciela Barchini de Giménez