



PLAN DE ESTUDIOS (PE): Licenciatura en Matemáticas

ÁREA: Interdisciplinaria

ASIGNATURA: Programación II

CÓDIGO:

CRÉDITOS: 6

FECHA: Noviembre 2016





1. DATOS GENERALES

Nivel Educativo:	Licenciatura
Nombre del Plan de Estudios:	Licenciatura en Matemáticas
Modalidad Académica:	Presencial
Nombre de la Asignatura:	Programación II
Ubicación:	Nivel formativo
Correlación:	
Asignaturas Precedentes:	Computación, Programación I
Asignaturas Consecuentes:	Algoritmos, estructuras de datos y objetos, Temas selectos de computación

2. CARGA HORARIA DEL ESTUDIANTE

Concepto	Horas por semana		Total de horas por periodo	Total de créditos por periodo
	Teoría	Práctica		
Horas teoría y práctica (16 horas = 1 crédito)	2	3	100	6





3. REVISIONES Y ACTUALIZACIONES

Autores:	Edgar Santiago Moyotl Hernández, Mónica Macías Pérez Patricia Domínguez Soto, Sergio Adán Juárez
Fecha de diseño:	Diciembre 2009
Fecha de la última actualización:	Noviembre 2015
Fecha de aprobación por parte de la academia de área, departamento u otro:	6 de julio del 2017
Revisores:	Edgar Santiago Moyotl Hernández, Mónica Macías Pérez Sergio Adán Juárez
Sinopsis de la revisión y/o actualización:	Se describieron las competencias profesionales a desarrollar, se reorganizaron las unidades de aprendizaje y se actualizaron tanto el contenido temático como la bibliografía.

4. PERFIL DESEABLE DEL PROFESOR (A) PARA IMPARTIR LA ASIGNATURA:

Disciplina profesional:	Ciencias de la computación, Ingeniería en sistemas computacionales, Matemáticas o carreras afines.
Nivel académico:	Maestría
Experiencia docente:	1 año
Experiencia profesional:	1 año

5. PROPÓSITO: El alumno desarrollará habilidades para aplicar elementos avanzados del lenguaje C en la resolución de problemas computacionales, desarrollando programas que optimicen el aprovechamiento de los recursos computacionales y utilizando la programación estructurada.





6. COMPETENCIAS PROFESIONALES:

- Describir cómo una computadora codifica la información y qué operaciones puede realizar con ella para seleccionar la forma óptima de almacenarla.
- Aplicar los componentes de bajo nivel para manipular datos utilizando los operadores a nivel de bits.
- Definir y aplicar el concepto de apuntador para elaborar programas que manejen de manera eficiente la memoria principal.
- Definir y aplicar el concepto de recursividad para resolver problemas que tengan naturaleza recursiva.
- Describir el concepto de estructura de datos y aplicar el concepto de registro en problemas que requieren procesar elementos relacionados entre sí, pero de distintos tipos.
- Aplicar el concepto de archivo para escribir programas que requieran almacenar y recuperar datos en dispositivos de memoria secundaria.
- Describir las herramientas matemáticas necesarias para la evaluación de la complejidad de un algoritmo.
- Aplicar los conceptos avanzados del lenguaje C para escribir programas que den solución a diversos problemas computacionales.
- Definir y aplicar conceptos básicos de graficación para codificar programas que permitan mostrar la salida de resultados en modo gráfico.
- Evaluar la importancia y el impacto de la programación en el contexto de las matemáticas.





7. CONTENIDOS TEMÁTICOS

Unidad de Aprendizaje	Contenido Temático	Referencias
1. Representación de la información (2 semanas)	- Sistemas de numeración - Sistema binario, octal y hexadecimal - Conversión entre sistemas - Operaciones aritméticas: suma, resta, multiplicación y división - La memoria principal - Organización y representación - Unidades de información: <i>bit</i>, <i>byte</i> y <i>word</i> - Representación de números enteros - Signo y magnitud - Complemento a 1 - Complemento a 2 - Representación de números reales - Notación científica: mantisa, exponente y normalización - Estándar IEEE 754 - Codificación de caracteres - Manejo de bits - Representación de números en C - Operador <code>sizeof</code> - Operadores lógicos de bits - Operadores de desplazamiento de bits	Andries, V. D., Hughes, J., Foley, J. D. y Feiner, S. K. (2007). <i>Computer Graphics Principles and Practice in C</i> (2nd ed.). USA: Pearson. Ceballos, J. (2015). <i>C/C++ Curso de Programación</i> (4ta. ed.). México: Alfaomega Deitel, P. J. y Deitel, H. M. (2010). <i>C how to program</i> (6th. ed.). México: Prentice Hall. Joyanes Aguilar, L. (2014). <i>Programación en C, C++, Java y UML</i> (2da. ed.). México: McGraw-Hill.
2. Apuntadores (3 semanas)	- Direcciones de memoria - Operadores de dirección "&" e indirección "*" - Declaración de variables apuntador - Tipos de apuntadores - Inicialización de apuntadores - Aritmética de apuntadores - Apuntador nulo - Usos de los apuntadores - Apuntadores y arreglos - Apuntadores y cadenas - Arreglos de apuntadores - Parámetros de funciones por referencia - Apuntadores a funciones	Mora Escobar, H. M. (2004). <i>Introducción a C y a métodos numéricos</i>. Colombia: Universidad Nacional de Colombia. Shirley, P., Ashikmin, M. y Marschner, S. (2009). <i>Fundamentals of Computer Graphics</i> (3th ed.). USA: A K Peters/CRC Press.





Unidad de Aprendizaje	Contenido Temático	Referencias
	4. Gestión dinámica de memoria 4.1. Estructura de la memoria: segmento de código, memoria estática, heap, pila. 4.2. Reserva de memoria 4.3. Liberación de memoria 4.4. Variables estáticas y dinámicas	
3. Recursividad (2 semanas)	1. Definición 2. Funciones recursivas 2.1. Caso base 2.2. Caso recursivo 3. Funcionamiento de la recursión 4. Tipos de recursividad 5. Ventajas y desventajas	
4. Estructuras de datos: registros (3 semanas)	1. Datos simples y estructurados 2. Definición y declaración de registros 2.1. Inicialización de registros 2.2. Acceso a los campos de un registro: operador punto "." 2.3. Tipos de datos definidos con <i>typedef</i> 3. Arreglos de registros 4. Registros anidados 5. Apuntadores a registros: operador flecha ">" 6. Registros y funciones 7. Enumeraciones	
5. Graficación (3 semanas)	1. Primitivas de graficación en 2D 2. Trazado de Polígonos 3. Rotación, Traslación y Escala de polígonos 4. Animación básica de polígonos 5. Introducción a los fractales	
6. Archivos (3 semanas)	1. Medios de almacenamiento (memoria secundaria) 2. Tipos de archivos: texto y binario 3. Métodos de acceso: secuencial y directo	





Unidad de Aprendizaje	Contenido Temático	Referencias
	3.1. Indicador de la posición en el archivo 4. Gestión de archivos 4.1. Apertura de archivos 4.2. Modos de apertura 4.3. Procesamiento de archivos: leer, escribir y actualizar 4.4. Cierre de archivos	
7. Complejidad algorítmica (2 semanas)	1. Conceptos iniciales 1.1. Eficiencia y complejidad 1.2. Recursos del sistema 1.3. Coste computacional 2. Cotas de complejidad asintóticas 2.1. Cota superior, notación O 2.2. Cota inferior, notación Ω 2.3. Cota ajustada, notación Θ 3. Resolución de ecuaciones de recurrencia	

8. ESTRATEGIAS, TÉCNICAS Y RECURSOS DIDÁCTICOS

Estrategias y técnicas didácticas	Recursos didácticos
• Resúmenes • Paráfrasis • Mapas conceptuales • Lluvia de ideas • Aprendizaje basado en problemas • Aprendizaje orientado a proyectos • Aprendizaje cooperativo • Aprendizaje colaborativo • Responder a preguntas exploratorias y literales • Prácticas • Elaboración de programas • Investigación bibliográfica extraclase	Materiales: • Impreso: libros y fotocopias. • Digital: libros, artículos y diapositivas. • Pizarrón, plumones y borrador. • Proyector y computadora. • Programas informáticos: se sugiere DevC++, Code::Blocks y/o Zinjai. • Páginas web, correo electrónico, chats y foros.





9. EJES TRANSVERSALES

Eje (s) transversales	Contribución de la asignatura
Formación Humana y Social	Solucionar problemas reales promueve la participación del alumno de manera cooperativa y colaborativa.
Desarrollo de Habilidades en el uso de las Tecnologías de la Información y la Comunicación	El uso de software para programar promueve el uso de medios electrónicos.
Desarrollo de Habilidades del Pensamiento Complejo	Mediante la programación se desarrolla la habilidad de resolver problemas conceptuales y cuantitativos utilizando diferentes formas de razonamiento (lógico, aritmético, algebraico y analógico).
Lengua Extranjera	Dado que la gran mayoría de los lenguajes de programación están en idioma inglés, se desarrolla la habilidad lectora y de comprensión de textos escritos en otro idioma.
Innovación y Talento Universitario	Resolver problemas computacionales ayuda a que el alumno desarrolle la habilidad para crear soluciones innovadoras y generar cambios.
Educación para la Investigación	Mediante la programación estructurada se orienta a una cultura de indagación, descubrimiento y construcción de conocimientos nuevos.

10. CRITERIOS DE EVALUACIÓN

Criterios	Porcentaje
▪ Exámenes	50%
▪ Participación en clase	10%
▪ Tareas	15%
▪ Prácticas de laboratorio	15%
▪ Proyecto final	10%
Total	100%

11. REQUISITOS DE ACREDITACIÓN

Estar inscrito como alumno en la Unidad Académica en la BUAP
Asistir como mínimo al 80% de las sesiones para tener derecho a exentar por evaluación continua y/o presentar el examen final en ordinario o extraordinario
Asistir como mínimo al 70% de las sesiones para tener derecho al examen extraordinario
Cumplir con las actividades académicas y cargas de estudio asignadas que señale el PE

