CARTA DESCRIPTIVA (FORMATO MODELO EDUCATIVO UACJ VISIÓN 2020)

I. Identificadores de la asignatura
Programación de Computadoras I

Instituto	IIT	**Modalidad:**	Presencial
Departamento:	Ingeniería Eléctrica y Computación	**Créditos:**	8
Materia:	Programación I		
Programa:	Ingeniería en Sistemas Computacionales	**Carácter:**	Obligatoria
Clave:	IEC981000	**Tipo:**	Curso
Nivel:	Licenciatura		
Horas:	64 Totales **Teoría:** 90%		**Práctica:** 10%

II. Ubicación

	Clave:
Antecedentes:	
Fundamentos de Programación.	IEC980900
Consecuente:	
Programación II	IEC981100

III. Antecedentes

Conocimientos: Sistemas numéricos, diagramas de flujo, algoritmos, estructura secuencial, estructuras repetitivas, saltos condicionales, arreglos. Esta materia tiene como requisito indispensable la materia de Fundamentos de Programación.

Habilidades: Pensamiento crítico, facilidad para el razonamiento, capacidad de análisis de problemas, razonamiento lógico, razonamiento abstracto, capacidad analítica, capacidad de síntesis, capacidad de observación, capacidad de inferir, capacidad de inducir

Actitudes y valores: Disposición al trabajo en equipo. Iniciativa de aprendizaje. Demostrar honestidad, responsabilidad, respeto, puntualidad. El alumno tendrá disposición a creatividad lógica, tenacidad, dedicación y constancia.

IV. Propósitos Generales

Esta materia tiene como propósito que el estudiante adquiera los conocimientos necesarios de programación estructurada y modular para utilizarla en la resolución de problemas por medio de un lenguaje de programación.

V. Compromisos formativos

Intelectual:

El estudiante se autodirige en la búsqueda de información y aprendizaje de técnicas ó métodos que permitan la solución de problemas relativos a su profesión. Desarrolla o elige soluciones a problemas utilizando la herramienta C. Se comunica efectivamente tanto en forma oral como escrita en el ejercicio de su profesión, siendo capaz de adecuar el nivel y contenido técnico de la comunicación de acuerdo a las necesidades o intereses del destinatario.

Humano:

Aporta esfuerzo, compromiso, integridad y honestidad a cualquier negocio, industria u organización pública o privada en donde ejerza sus servicios profesionales. Participa como un miembro productivo cuando integre equipos de trabajo.

Social:

Respeta las leyes y normas establecidas por la sociedad y de manera particular aquellas relacionadas con el ejercicio de su profesión. Es cuidadoso de actuar bajo los principios éticos de su profesión. Se muestra interesado por contribuir, desde el ejercicio de su profesión, a la conservación del medio ambiente.

Profesional:

El estudiante incorpora a su formación los conocimientos del lenguaje C en todos sus niveles en la resolución de problemas.

VI. Condiciones de operación

Espacio: aula tradicional

Laboratorio: cómputo

Mobiliario: mesa redonda y sillas

Población: 25 - 30

Material de uso frecuente:

A) Rotafolio
B) Proyector
C) Cañon y computadora portatil

Condiciones especiales: No aplica

Temas	Contenidos	Actividades
1. Introducción al curso.	**Encuadre del curso.** Importancia de la programación. Ejemplos donde se utiliza la programación.	El instructor presenta el programa, las políticas del curso y la forma de evaluar. El instructor explica la importancia del curso y da ejemplos. El instructor relaciona los conceptos de la clase de Fundamentos de Programación con el curso. El estudiante lee y responde a las preguntas del profesor, toma nota y subraya los apuntes.
2. Introducción a la programación.	a. Arquitectura general del sistema mínimo de una computadora. b. Bit, nibble byte, palabra y palabra doble. c. Repaso de sistema binario y hexadecimal. d. Unidad Central de Procesamiento (CPU) i. Registros de almacenamiento ii. ALU iii. Unidad de control iv. Memoria caché e. Sistema de buses i. Datos ii. Direcciones iii. Control f. Memoria: ram, rom g. Dispositivos de entrada y salida	El instructor menciona los módulos de un sistema computador mínimo y da ejemplos de manejo de memoria y registros del procesador utilizando debugger (depurador). El instructor hace hincapié en que la memoria es lineal. El instructor realiza ejemplos de direccionamiento utilizando registros apuntadores. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante realiza una práctica de apuntadores utilizando debugger. El estudiante lee y responde a las preguntas del profesor, toma nota y subraya los apuntes. El estudiante realiza una exposición de conceptos acerca de los lenguajes de programación (historia, clasificación, concepto de programación modular y estructurada, paradigmas de programación existentes en los lenguajes de alto nivel. El estudiante realiza la práctica con el debug u otro debugger que permita que permita ver el código máquina y código ensamblado con el fin de visualizar un programa sencillo implantado en la memoria. Dicho programa debe involucrar la transferencia de un dato desde y hacia la memoria para resaltando el concepto de registro apuntador.
3. Representación de datos en memoria y operadores básicos	a. Declaración de datos, apuntadores y manejo de memoria b. Representarción de datos char, unsigned char, int y unsigned int en memoria. c. Direccionamiento (Posicionamiento) de los datos declarados utilizando mapas de memoria. d. Operadores & y * : contextos en que se utilizan y su precedencia. i. &A Dirección del dato. ii. A & B de los bits de A con los bits de B. iii. A*B Producto de dos datos. iv. *P Contenido de la dirección apuntada por el apuntador e. Operadores ++, --, ( ) , +, - .	El instructor repasa la representación de los datos de memoria. El instructor introduce al estudiante al ambiente de programación. El instructor explica los operadores & y * así como los contextos en los que se utilizan y da ejemplos conceptuales. El instructor explica los operadores (++, --, ( ) , +, -) utilizados en operaciones y da ejemplos conceptuales. El estudiante toma nota, subraya sobre los apuntes y realiza ejemplos de la representación de datos en memoria. El estudiante toma nota y subraya sobre los apuntes y realiza ejemplos con apuntadores (direcciones y datos). El estudiante lleva a cabo lleva a cabo su primer proyecto in situ con el instructor. Abrir VC ++ 6.0, abrir proyecto nuevo, abrir archivo nuevo, agregarlo al proyecto, declarar función mandatoria (se explica más adelante) , explicar el uso del punto y coma (;) para indicar fin de línea, también cómo insertar comentarios dentro de un programa (/* */, //). El estudiante realiza ejemplos de operaciones básicas con datos y con apuntadores utilizando los operadores vistos. El estudiante realiza una práctica in situ que integra operadores.

4. Direccionamiento.	a. Declaración de datos, apuntadores y manejo de memoria. b. Capacidad de direccionamiento de los apuntadores. c. Referencia y dereferencia de apuntadores y datos.	El instructor repasa el direccionamiento de datos en memoria vistos al inicio del curso pero ahora relacionándolos con el lenguaje C. Resaltando el hecho de que el tipo de un apuntador es finalmente la cantidad de memoria que cubre un dato. El instructor da ejemplos de los diferentes tipos apuntadores. El instructor ejemplifica la forma de apuntar a los datos y realiza mapas de memoria. El instructor realiza mapas de memoria de los programas realizados. El estudiante toma nota, subraya sobre los apuntes y realiza ejemplos de la declaración de apuntadores y su representación en memoria. El estudiante toma nota y subraya sobre los apuntes y realiza ejemplos con sus respectivos mapas de memoria. El estudiante visualiza la capacidad de los apuntadores. El estudiante resuelve problemas con apuntadores.
5. Datos de punto fijo, punto flotante y doble precisión.	a. Datos de punto fijo (parte entera + parte decimal Qm.n) b. Datos de punto flotante (4 bytes) c. Datos de doble precisión (8 bytes) d. Ventajas y desventajas de datos de punto fijo contra punto flotante. e. Conversión entre tipos de datos (**cast**) y nombres alternativos (**typedef**)	El instructor repasa los datos de punto fijo (formato Qm.n). El instructor da ejemplos de datos de punto fijo. El instructor repasa los datos de punto flotante. El instructor da ejemplos de datos de punto flotante. El instructor repasa los datos doble. El instructor da ejemplos de datos doble. El instructor enseña las instrucciones cast y typedef y da ejemplo de sus usos. El estudiante reflexiona, toma nota, subraya sobre los apuntes y realiza ejemplos con sus respectivos mapas de memoria de datos de punto fijo, de punto flotante y de doble precisión. El estudiante resuelve problemas con apuntadores a datos de punto fijo, de punto flotante y de doble precisión. El estudiante resuelve problemas utilizando **cast** y **typedef**.
6. Elementos de lenguaje.	a. Comentarios, identificadores, palabras reservadas. b. Variables, constantes, datos primitivos. c. Operadores y su precedencia.	El instructor enseña el resto de los elementos del lenguaje y repasa los operadores vistos en la clase de Fundamentos de Programación. El estudiante toma nota, subraya sobre los apuntes y realiza ejemplos con sus respectivos mapas de memoria
7. Expresiones, sentencias y control de flujo.	a. Definición de sentencia b. Tipos de sentencias i. Secuencia 1. Declaraciones 2. Expresiones ii. Decisión 3. if – elseif – else 4. El operador condicional **?** y **:** 5. switch – case iii. Iteraciones 6. for 7. while 8. do – while iv. Sentencias break y continue.	El instructor repasa los conceptos de expresiones y sentencias de control de flujo y explica la sintaxis en C de las mismas. El estudiante repasa los apuntes de la clase pre-requisito, toma nota y subraya los apuntes,. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante realiza diagramas de flujo en la etapa del diseño de la solución de algún problema. El estudiante resuelve problemas que involucran expresiones, sentencias y control de flujo mediante la programación en C involucrando direccionamiento con apuntadores. El estudiante realiza el mapa de memoria de su programa.
8. Funciones.	a. Declaración de funciones. b. Pila de llamada de funciones. c. La función main( ). d. Funciones definidas por el usuario. e. Llamada de funciones f. Pase y retorno a una función por valor y por referencia de un dato.	El instructor explica la plantilla general para la declaración de funciones y El instructor explica cómo se hace un llamado a las funciones. El instructor explica las pilas de llamada de funciones así como la función main(). El instructor explica las funciones definidas por el usuario así

	g. Mapas de memoria. h. Recursión.	como algunas funciones definidas por el lenguaje. El instructor explica el pase por valor y por referencia a una función. El instructor explica el retorno de un dato por medio su valor o por su referencia. El instructor realiza los mapas de memoria correspondientes en cada caso. El instructor explica el concepto de recursión. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante realiza el diseño para la solución de algún problema que involucre control de flujo, funciones y apuntadores, utilizando diagramas de flujo. El estudiante resuelve el problema utilizando la programación C que involucre control de flujo, funciones y apuntadores, utilizando diagramas de flujo.
9. Alcance, cobertura de variables y directivas de preprocesamiento.	a. Prototipos de funciones b. Variables globales y locales c. Constantes d. Directivas del preprocesador	El instructor explica los prototipos de funciones. El instructor explica el concepto de variables globales y locales. El instructor explica el concepto de constantes. El instructor explica cómo funcionan las directivas del proprocesador. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante resuelve problemas que incluyen el uso de control de flujo, funciones, apuntadores, variables locales y globales así como directivas del preprocesador.
10. Bibliotecas.	a. Definición y declaración de bibliotecas propias del lenguaje. ii. stdio.h iii. stdlib.h iv. conio.h b. Funciones printf(), scanf(), getch(), getchar(), putc(), putchar() y especificadores de formato. c. Bibliotecas definidas por el usuario.	El instructor explica el objeto de las bibliotecas y su uso. El instructor explica algunas bibliotecas propias del lenguaje C. El instructor explica funciones del stream así como los especificadores de formato. El instructor explica las bibliotecas definidas por el usuario. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante diseña un diagrama de flujo que resuelva un problema que involucre el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento y bibliotecas. El estudiante resuelve problemas que incluyen el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento y bibliotecas.
11. Entornos de desarrollo.	a. Entornos de desarrollo integrados b. El editor c. El compilador d. El ligador o enlazador e. Ejecución	A estas alturas el estudiante ya ha tenido contacto con algún entorno de desarrollo. Se espera que ahora reflexiones y le sea más fácil entender todo el proceso por el que pasa un programa desde su edición hasta llegar a su ejecución. El estudiante realiza una investigación de los diferentes entornos de desarrollo. El estudiante forma equipos de trabajo para exponer cada parte de un entorno de desarrollo.
12. Arreglos y cadenas.	a. Declaración e inicialización de arreglos. b. Extracción de datos y recorrido de arreglos utilizando apuntadores. v. Utilizando apuntador. vi. Utilizando índices. c. Operaciones (post y pre) incremento y decremento de apuntadores en arreglos. d. Declaración e inicialización de cadenas. e. Extracción caracteres de una cadena. vii. Utilizando apuntadores. viii. Utilizando índices.	El instructor explica el uso y la importancia de los arreglos estáticos y las cadenas. El instructor explica la declaración e inicialización de arreglos y cadenas utilizando inicialización directa o por medio de un apuntador. El instructor explica las operaciones de pos y pre incremento y decremento. El instructor ejemplifica el uso de los arreglos y las cadenas. El instructor explica las operaciones con apuntadores. El instructor explica el pase de arreglos de 1D y 2D a funciones. El instructor ejemplifica las operaciones con arreglos y con cadenas de caracteres específicamente en el manejo de

	f. Pase de arreglos a funciones. g. La biblioteca ctype.h. h. Operaciones con cadenas de caracteres. i. Conversión y pruebas de cadenas de caracteres.	matrices y vectores de datos. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante diseña un diagrama de flujo que resuelva un problema que involucre el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento, bibliotecas y arreglos/cadenas. El estudiante resuelve un problema que incluya el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento, bibliotecas y arreglos/cadenas. El estudiante realiza el mapa de memoria del programa.
13.Estructuras y uniones.	a. Declaración e inicialización de estructuras. b. Declaración de variables tipo estructura y apuntadores a estructuras. c. Arreglos de estructuras. d. Acceso a miembros de una estructura. e. Mapas de memoria de una estructura. f. Declaración e inicialización de uniones. g. Declaración de variables y apuntadores a uniones. h. Arreglos de uniones. i. Acceso a miembros de una unión. j. Acceso a campos de bits de una unión.	El instructor explica las estructuras y la forma de declararlas en un programa en C. El instructor explica la forma de acceder a los miembros de una estructura utilizando el operador de acceso (nombre de la variable estructurada seguida por un punto (**.**) seguido por el nombre del miembro o componente deseado de la estructura). El instructor explica la forma de acceder a los miembros de una estructura utilizando un apuntador a la estructura (APNTDR -> miembro). El instructor explica el diagrama de memoria de las estructuras. El instructor explica la forma declarar arreglos de estructuras y la forma de acceder a los miembros de cada estructura utilizando apuntadores. El instructor explica el diagrama de memoria de los arreglos de estructuras. El instructor explica los campos de bits en una estructura. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante diseña un diagrama de flujo que resuelva un problema que involucre el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento, bibliotecas, arreglos/cadenas y apuntadores a estructuras. El estudiante resuelve un problema utilizando el lenguaje C que incluya el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento, bibliotecas y arreglos/cadenas bibliotecas, arreglos/cadenas y apuntadores a estructuras. El estudiante realiza el mapa de memoria del programa. El instructor explica las es y la forma de declararlas en un programa en C. El instructor explica la forma de acceder a los miembros de una unión utilizando el operador de acceso (nombre de la variable tipo unión seguida por un punto (**.**) seguido por el nombre del miembro o componente deseado de la unión). El instructor explica la forma de acceder a los miembros de una unión utilizando el operador de acceso punto (nombre de la variable tipo unión seguida por un punto (**.**) seguido por el nombre del miembro o componente deseado de la unión). El instructor explica la forma de acceder a los miembros de una unión utilizando apuntadores (APNTDR -> miembro). El instructor explica el diagrama de memoria de las uniones. El instructor explica la forma declarar arreglos de uniones y la forma de acceder a los miembros de cada unión utilizando apuntadores. El instructor explica el diagrama de memoria de los arreglos de uniones. El instructor explica los campos de bits en una unión. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante diseña un diagrama de flujo que resuelva un problema que involucre el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de

		preprocesamiento, bibliotecas, arreglos/cadenas, apuntadores a estructuras y uniones. El estudiante resuelve un problema utilizando el lenguaje C que incluya el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento, bibliotecas y arreglos/cadenas bibliotecas, arreglos/cadenas apuntadores a estructuras y uniones.. El estudiante realiza el mapa de memoria del programa.
14.Memoria dinámica.	a. Definición b. Memoria heap c. Funciones i. calloc() ii. malloc() iii. free() d. Nodos dinámicos utilizando estructuras e. Lista unilateral. f. Lista doblemente enlazada. g. Problema de búsquedas de números y cadenas de caracteres en listas. h. Arreglo bidimensional con arreglo de apuntadores. i. Arreglo bidimensional utilizando apuntador de apuntador.	El instructor explica el concepto de memoria dinámica así como la memoria heap. También expone las ventajas y desventajas. El instructor explica las bibliotecas con las que se pueden llamar a las funciones que reservan la memoria dinámica. El instructor explica las funciones utilizadas para reservar la memoria dinámica. El instructor explica la forma de reservar memoria dinámica. El estudiante crea y destruye (free()) nodos dinámicos realizados en C. El instructor explica el concepto de listas unilaterales y doblemente enlazadas. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante realiza un diagrama de flujo para realizar una lista unilateral. El estudiante realiza un programa en C para realizar una lista unilateral. El estudiante realiza un diagrama de flujo para realizar una lista doblemente enlazada. El estudiante realiza un programa en C para realizar una lista doblemente enlazada EL estudiante resuelve problemas de ordenamiento utilizando listas y conceptos vistos en temas anteriores. El instructor explica los arreglos bidimensionales con memoria dinámica. El instructor explica el concepto de apuntador de apuntador. El instructor explica el concepto de arreglos bidimensionales utilizando apuntador de apuntador. El estudiante resuelve problemas con matrices utilizando arreglos bidimensionales.
15.Archivos.	a. Texto b. Binarios c. Apuntador de archivos tipo FILE y la macro NULL. d. Funciones para trabajar con archivos i. fscanf, fprintf, fread, fwrite, fseek, ftell, fclose, feof, rewind, ferror, fflush(), remove, etc. e. Archivos de acceso secuencial o directo. f. Archivos de acceso aleatorio.	El instructor explica el concepto de archivos de texto y binarios así como el apuntador FILE y la macro NULL. El instructor explica la librería para trabajar con archivos así como las diferentes funciones de la misma. El instructor explica los archivos de acceso secuencial o directo así como los archivos de acceso aleatorio. El estudiante reflexiona, toma nota, subraya los apuntes. El estudiante resuelve un problema utilizando el lenguaje C que incluya el uso de control de flujo, funciones, apuntadores, variables locales y globales, directivas de preprocesamiento, bibliotecas y arreglos/cadenas bibliotecas, arreglos/cadenas apuntadores a estructuras, uniones memoria dinámica y archivos, por ejemplo leer una imagen de 256 tonos de gris de 512 x 512 pixeles y obtener su histograma para ecualizarlo.

VIII. Metodología y estrategias didácticas

Metodología Institucional:

a) Elaboración de ensayos, monografías e investigaciones (según el nivel) consultando fuentes bibliográficas, hemerográficas y en Internet.
b) Elaboración de reportes de lectura de artículos en lengua inglesa, actuales y relevantes.

Estrategias del Modelo UACJ Visión 2020 recomendadas para el curso:

a) aproximación empírica a la realidad
b) búsqueda, organización y recuperación de información
c) comunicación horizontal
d) descubrimiento
e) ejecución-ejercitación
f) elección, decisión
g) evaluación
h) experimentación
i) extrapolación y trasferencia
j) internalización
k) investigación
l) meta cognitivas
m) planeación, previsión y anticipación
n) problematización
o) proceso de pensamiento lógico y crítico
p) procesos de pensamiento creativo divergente y lateral
q) procesamiento, apropiación-construcción
r) significación generalización
s) trabajo colaborativo

IX. Criterios de evaluación y acreditación

a) Institucionales de acreditación:

Acreditación mínima de 80% de clases programadas

Entrega oportuna de trabajos

Pago de derechos

Calificación ordinaria mínima de 7.0

Permite examen único: no

b) Evaluación del curso

Acreditación de los temas mediante los siguientes porcentajes:

Tema 2 = 2 %, Tema 4 = 3%, Tema 6 =5%, Tema 7 = 5%, Tema 8 =5%,
Tema 9 = 5 %, Tema 10 = 5%, Tema 11 = 5%, Tema 12 = 5%, Tema 13 =5%,
Tema 14 = 5%, Tema 15 = 5%, Examen Final: 20 %

Participación: 5%, Tareas: 5%, Prácticas : 5%, Proyectos: 10%

Total: 100 %

X. Bibliografía

Deitel, H. M., & Deitel, P. J. (2009). *Cómo programar en C++.* México: Pearson.

Joyanes Aguilar, L. (2008). *Fundamentos de Programación Algoritmos, estructuras y objetos.* Madrid: McGrawHill.

Venit, Stewart; Drake, Elizabeth;. (2009). *Prelude to Programming Concepts & Design.* Boston: Pearson.

Villalobos S., Jorge A.; Casallas, Ruby G.;. (2006). Problemas, Soluciones y Programas. En *Fundamentos de Programación Aprendizaje Activo Basado en Casos* (págs. 1-61). México: Pearson.

X. Perfil deseable del docente

Maestría, preferentemente doctorado en áreas de ciencias o ingeniería de la computación o tecnologías de información.

XI. Institucionalización
Responsable del Departamento: Jesús Armando Gándara Fernández. Coordinador/a del Programa: Cynthia Vanessa Esquivel Rivera. Fecha de elaboración: Mayo 09, 2011. Elaboró: Patricia Parroquín / Leticia Ortega / Martha Victoria González Demoss. Fecha de rediseño: 13 de febrero del 2013. Rediseño: Humberto de Jesús Ochoa Domínguez.