MODALIDAD ABIERTA :: PLAN DE TRABAJO::

DATOS DE LA ASIGNATURA

Licenciaturas en que se imparte:			Lic. Informática 3er sem.
Nombre:	Estructura de datos		
Clave(s):	2331		
Tipo:	Obligatoria		
Plan de Estudios:	**2024**		

FECHAS DEL SEMESTRE

Inicio semestre:	4 de febrero de 2025
Fin del semestre:	13 de junio 2025
Plataforma educativa:	19 de febrero de 2025 Primer día para entrega de actividades en plataforma
Cierre de plataformas:	25 de mayo de 2025 a las 23:00 hrs. Último día para entrega de actividades en plataforma
Periodo examen global:	6, 7 y del 9 al 12 de junio 2025
Consulta de calificaciones en historia académica:	A partir del 30 de junio 2025

OBJETIVO GENERAL: Al finalizar el curso, el alumnado implementará las estructuras de datos fundamentales y avanzadas por medio de la organización lógica de datos, minimizando los tiempos de acceso y logrando eficiencia en la inserción, la eliminación, la búsqueda y el ordenamiento de datos.

CONTENIDO TEMATICO

Unidad	Tema	Teóricas
1	Fundamentos	4
2	Estructuras de datos fundamentales	20
3	Estructuras de datos avanzadas	16
4	Algoritmos de ordenamiento	10
5	Algoritmos de búsqueda	10
6	Análisis de algoritmos	4
	Total	**64**
	Suma total de horas	64 horas

BIENVENIDA

Estimados estudiantes:

Les doy la bienvenida al curso de Estructura de datos.

Pueden contar conmigo para resolver sus dudas y para obtener retroalimentación de sus actividades, sin embargo, recuerden que el autoaprendizaje es un factor decisivo en la modalidad abierta.

¡Les deseo mucho éxito!

PRESENTACIÓN DE LA ASIGNATURA

La asignatura de Estructura de Datos es clave para tu formación, ya que te brinda las herramientas necesarias para organizar, gestionar y optimizar el almacenamiento y procesamiento de información en sistemas computacionales.

Los conocimientos que adquieras aquí podrás aplicarlos directamente en tus proyectos académicos, como el desarrollo de software y la resolución de problemas algorítmicos, así como en tu futura actividad laboral, al diseñar soluciones eficientes para manejar grandes volúmenes de datos.

Además, esta materia se conecta con otras áreas importantes como la programación, el diseño de algoritmos, bases de datos e inteligencia artificial, permitiéndote construir un puente sólido entre los conceptos teóricos y su implementación práctica.

FORMA EN QUE EL ALUMNADO DEBE PREPARAR LA ASIGNATURA

Todas las actividades, excepto la 4.2 y la 5.2, deberán entregarse en formato PDF y deben incluir una carátula que incluya los escudos de la UNAM y de nuestra Facultad, nombre del alumno, carrera y nombre del asesor.

Cuando se te pida compilar y ejecutar código, además de lo anterior, debes incluir el archivo con extensión .c en tu envío. En las actividades 4.2 y 5.2, este archivo es lo único que debes adjuntar.

Para presentar exámenes parciales, se deben haber entregado previamente las actividades de las unidades correspondientes.

Solo se podrá subir una vez el archivo de cada actividad. Una vez evaluado no se podrá corregir.

El plagio se califica con cero.

Para la realización de tus actividades deberás cuidar tu **ortografía** y usar **fuentes oficiales** como: libros, revistas, artículos, etcétera. Recuerda hacer la referencia en formato APA, ya que, si no lo haces incurrirás en plagio. https://www.revista.unam.mx/wp-content/uploads/3_Normas-APA-7-ed-2019-11-6.pdf .

Si tomas como base algún código disponible en la web o utilizas inteligencia artificial, también debes colocar las referencias correspondientes en formato APA. Un ejemplo de cita de inteligencia artificial en ese formato es:

OpenAI. (2024). *ChatGPT* [Modelo de lenguaje de gran tamaño]. https://chat.openai.com/chat

Para la entrega extemporánea de actividades tendrás una semana más con una calificación máxima de 8.0

ACTIVIDADES POR REALIZAR DURANTE EL SEMESTRE

Unidad	N° Actividad	Fecha de entrega	Descripción	Valor (enteros)
Unidad 1	Actividad 1	19 de febrero	Escribe las definiciones de los siguientes conceptos: - Arreglos estáticos - Arreglos dinámicos - Matrices - Registros - Apuntadores - Estructuras de datos	5 pts

Unidad 2	Actividad 1	26 de febrero	Escribe las definiciones de los siguientes conceptos:
			- Pilas
			- Función push
			- Función pop
			
			Compila y ejecuta el siguiente código en C:
			
			#include <stdlib.h>
			#include <stdio.h>
			typedef struct _nodo {
			int valor;
			struct _nodo *siguiente;
			} tipoNodo;
			typedef tipoNodo *pNodo;
			typedef tipoNodo *Pila;
			/* Funciones con pilas: */
			void Push(Pila *l, int v);
			int Pop(Pila *l);
			int main() {
			Pila pila = NULL;
			Push(&pila, 10);
				5 pts

```
   Push(&pila, 5);
   printf("%d\n", Pop(&pila));
   printf("%d\n", Pop(&pila));
   Push(&pila, 20);
   Push(&pila, 15);
   printf("%d\n", Pop(&pila));
   printf("%d\n", Pop(&pila));
   Push(&pila, 25);
   printf("%d\n", Pop(&pila));
getchar();
return 0; }
void Push(Pila *pila, int v) {
   pNodo nuevo;
   /* Crear un nodo nuevo */
   nuevo = (pNodo)malloc(sizeof(tipoNodo));
   nuevo->valor = v;
   /* Añadimos la pila a continuación del nuevo nodo */
   nuevo->siguiente = *pila;

    /* Ahora, el comienzo de nuestra pila es en nuevo nodo */
   *pila = nuevo;
}
int Pop(Pila *pila) {
   pNodo nodo; /* variable auxiliar para manipular nodo */
   int v;      /* variable auxiliar para retorno */
   /* Nodo apunta al primer elemento de la pila */
   nodo = *pila;
   if(!nodo) return 0; /* Si no hay nodos en la pila
retornamos 0 */
   /* Asignamos a pila toda la pila menos el primer elemento
*/
   *pila = nodo->siguiente;
   /* Guardamos el valor de retorno */
   v = nodo->valor;
   /* Borrar el nodo */
   free(nodo);
```

			return v; } ¿Cuál es la salida del programa? Explica por qué es esa la salida.	

	Actividad 2	5 de marzo	Escribe las definiciones de los siguientes conceptos:
			- Colas
			- Función encolar (enqueue)
			- Función desencolar (dequeue)
			
			Compila y ejecuta el siguiente código en C:
			
			#include <stdlib.h>
			#include <stdio.h>
			
			typedef struct _nodo {
			int valor;
			struct _nodo *siguiente;
			} tipoNodo;
			
			typedef tipoNodo *pNodo;
			
			/* Funciones con colas: */
			void Encolar(pNodo *primero, pNodo *ultimo, int v);
			int Desencolar(pNodo *primero, pNodo *ultimo);
			
			int main() {
			pNodo primero = NULL, ultimo = NULL;
			
			Encolar(&primero, &ultimo, 10);
			Encolar(&primero, &ultimo, 5);
			printf("%d\n", Desencolar(&primero, &ultimo));
			Encolar(&primero, &ultimo, 20);
			Encolar(&primero, &ultimo, 15);
				5 pts

```
    printf("%d\n", Desencolar(&primero, &ultimo));
    printf("%d\n", Desencolar(&primero, &ultimo));
    Encolar(&primero, &ultimo, 25);
    printf("%d\n", Desencolar(&primero, &ultimo));
    printf("%d\n", Desencolar(&primero, &ultimo));

    return 0;
}

void Encolar(pNodo *primero, pNodo *ultimo, int v) {
    pNodo nuevo;

    /* Crear un nodo nuevo */
    nuevo = (pNodo)malloc(sizeof(tipoNodo));
    nuevo->valor = v;

    /* Este será el último nodo, no debe tener siguiente */
    nuevo->siguiente = NULL;

    /* Si la cola no estaba vacía, añadimos el nuevo a continuación de ultimo */
    if (*ultimo) {
        (*ultimo)->siguiente = nuevo;
    }

    /* Ahora, el último elemento de la cola es el nuevo nodo */
    *ultimo = nuevo;

    /* Si primero es NULL, la cola estaba vacía, ahora primero apuntará también al nuevo nodo
*/
    if (!*primero) {
        *primero = nuevo;
    }
}

int Desencolar(pNodo *primero, pNodo *ultimo) {
    pNodo nodo; /* variable auxiliar para manipular nodo */
```

			int v; /* variable auxiliar para retorno */
			
			/* Nodo apunta al primer elemento de la pila */
			nodo = *primero;
			
			if (!nodo) {
			  return 0; /* Si no hay nodos en la pila retornamos 0 */
			}
			
			/* Asignamos a primero la dirección del segundo nodo */
			*primero = nodo->siguiente;
			
			/* Guardamos el valor de retorno */
			v = nodo->valor;
			
			/* Borrar el nodo */
			free(nodo);
			
			/* Si la cola quedó vacía, ultimo debe ser NULL también */
			if (!*primero) {
			  *ultimo = NULL;
			}
			
			return v;
			}
			
			¿Cuál es la salida del programa? Explica por qué es esa la salida.	

	Actividad 3	12 de marzo	Escribe las definiciones de los siguientes conceptos: - Listas enlazadas - Función insertar (en inglés se usa *insert)* - Función eliminar (en inglés se usa *delete*) Compila y ejecuta el siguiente código en C:	5 pts

```
#include <stdlib.h>
#include <stdio.h>

typedef struct _nodo {
   int valor;
   struct _nodo *siguiente;
} tipoNodo;

typedef tipoNodo *pNodo;
typedef tipoNodo *Lista;

/* Declaración de funciones con listas */
void Insertar(Lista *l, int v);
void Borrar(Lista *l, int v);
int ListaVacia(Lista l);
void BorrarLista(Lista *);
void MostrarLista(Lista l);

int main() {
   Lista lista = NULL;

   Insertar(&lista, 10);
   Insertar(&lista, 5);
   Insertar(&lista, 20);
   Insertar(&lista, 15);

   MostrarLista(lista);

   Borrar(&lista, 10);
   Borrar(&lista, 5);
   Borrar(&lista, 25);
   Borrar(&lista, 20);
   Borrar(&lista, 25);
```

```
    MostrarLista(lista);

    BorrarLista(&lista);

    return 0;
}

void Insertar(Lista *lista, int v) {
    pNodo nuevo, anterior;

    /* Crear un nodo nuevo */
    nuevo = (pNodo)malloc(sizeof(tipoNodo));
    nuevo->valor = v;

    /* Si la lista está vacía o el nuevo valor es menor que el primero */
    if (ListaVacia(*lista) || (*lista)->valor > v) {
        nuevo->siguiente = *lista;
        *lista = nuevo;
    } else {
        /* Buscar el nodo de valor menor a v */
        anterior = *lista;

        while (anterior->siguiente && anterior->siguiente->valor <= v) {
            anterior = anterior->siguiente;
        }

        /* Insertar el nuevo nodo después del nodo anterior */
        nuevo->siguiente = anterior->siguiente;
        anterior->siguiente = nuevo;
    }
}

void Borrar(Lista *lista, int v) {
    pNodo anterior = NULL, nodo = *lista;

    /* Buscar el nodo a eliminar */
```

```
    while (nodo && nodo->valor < v) {
        anterior = nodo;
        nodo = nodo->siguiente;
    }

    /* Si el nodo no existe, salir */
    if (!nodo || nodo->valor != v) {
        return;
    }

    /* Borrar el nodo */
    if (!anterior) { // Si es el primer nodo
        *lista = nodo->siguiente;
    } else { // Nodo intermedio o final
        anterior->siguiente = nodo->siguiente;
    }
    free(nodo);
}

int ListaVacia(Lista lista) {
    return (lista == NULL);
}

void BorrarLista(Lista *lista) {
    pNodo nodo;

    while (*lista) {
        nodo = *lista;
        *lista = nodo->siguiente;
        free(nodo);
    }
}

void MostrarLista(Lista lista) {
    pNodo nodo = lista;
```

			if (ListaVacia(lista)) {
			printf("Lista vacía\n");
			} else {
			while (nodo) {
			printf("%d -> ", nodo->valor);
			nodo = nodo->siguiente;
			}
			printf("NULL\n");
			}
			}
			
			¿Cuál es la salida del programa? Explica por qué es esa la salida.	

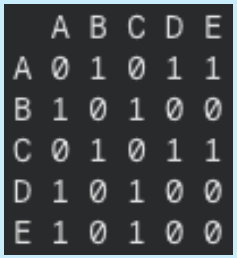
Unidad 3	Actividad 1	19 de marzo	Escribe las definiciones de los siguientes conceptos: - Árboles - Elementos de un árbol (nodo, raíz, hojas, subárbol, nivel y altura) - Tipos de árboles (binarios, binarios de búsqueda, AVL y B) - Recorridos de árboles (inorden, preorden y postorden)	5 pts

	Actividad 2	26 de marzo	Compila y ejecuta el siguiente código en C:
			
			#include <stdio.h>
			#include <stdlib.h>
			
			// Definición de un nodo del árbol
			typedef struct Nodo {
			int valor;
			struct Nodo *izquierdo;
			struct Nodo *derecho;
			} Nodo;
			
			// Función para crear un nuevo nodo
			Nodo* crearNodo(int valor) {
				5 pts

```
    Nodo* nuevoNodo = (Nodo*)malloc(sizeof(Nodo));
    nuevoNodo->valor = valor;
    nuevoNodo->izquierdo = NULL;
    nuevoNodo->derecho = NULL;
    return nuevoNodo;
}

// Función para insertar un valor en el árbol
Nodo* insertar(Nodo* raiz, int valor) {
    if (raiz == NULL) {
        // Si el árbol está vacío, crear el nodo raíz
        return crearNodo(valor);
    }
    if (valor < raiz->valor) {
        // Insertar en el subárbol izquierdo
        raiz->izquierdo = insertar(raiz->izquierdo, valor);
    } else {
        // Insertar en el subárbol derecho
        raiz->derecho = insertar(raiz->derecho, valor);
    }
    return raiz;
}

// Recorrido del árbol
void recorrido(Nodo* raiz) {
    if (raiz != NULL) {
        recorrido(raiz->izquierdo);
        printf("%d ", raiz->valor);
        recorrido(raiz->derecho);
    }
}

// Liberar memoria del árbol
void liberarArbol(Nodo* raiz) {
    if (raiz != NULL) {
        liberarArbol(raiz->izquierdo);
```

			liberarArbol(raiz->derecho);
			free(raiz);
			}
			}
			
			// Programa principal
			int main() {
			Nodo* arbol = NULL;
			
			// Insertar valores en el árbol
			arbol = insertar(arbol, 50);
			arbol = insertar(arbol, 30);
			arbol = insertar(arbol, 70);
			arbol = insertar(arbol, 20);
			arbol = insertar(arbol, 40);
			arbol = insertar(arbol, 60);
			arbol = insertar(arbol, 80);
			
			// Mostrar el recorrido del árbol
			printf("Recorrido: ");
			recorrido(arbol);
			printf("\n");
			
			// Liberar memoria del árbol
			liberarArbol(arbol);
			
			return 0;
			}
			
			¿Cuál es la salida del programa? Explica si el recorrido del árbol se hizo en inorden, preorden o postorden.	

	Actividad 3	2 de abril	Escribe las definiciones de los siguientes conceptos: - Grafos	5 pts

			- Elementos de un árbol (vértices, aristas y bucles) - Tipos de grafos (dirigidos y ponderados) - Representaciones de grafos (matriz de adyacencia, lista de adyacencia, matriz de incidencia) Observa la siguiente matriz de adyacencia:  ¿Cuántos nodos tiene el grafo resultante? ¿Cuántas aristas tiene el grafo resultante? ¿El grafo resultante tiene bucles? ¿El grafo resultante es dirigido? Dibuja el grafo resultante.	
Unidad 4	Actividad 1	9 de abril	Escribe las definiciones de los siguientes conceptos: - Algoritmo de ordenamiento - Tipos de algoritmos de ordenamiento (intercambio directo e indirecto) - Algoritmo burbuja y burbuja bidireccional - Algoritmo de inserción - Algoritmo de selección - Algoritmo *shell* - Algoritmo *quick sort*	5 pts

			- Algoritmo mezcla directa	
	Actividad 2	23 de abril	Implementa el algoritmo burbuja en C, utilizando los siguientes números: 3, 7, 2, 9 y 4. El código debe regresar los números ordenados.	10 pts
Unidad 5	Actividad 1	30 de abril	Escribe las definiciones de los siguientes conceptos: - Algoritmo de búsqueda - Búsqueda secuencial - Búsqueda binaria - Búsqueda *hashing*	5 pts
	Actividad 2	14 de mayo	Implementa el algoritmo de búsqueda secuencial en C, utilizando un arreglo con los números: 3, 7, 2, 9 y 4. Debes pedir al usuario que introduzca el valor a buscar por medio del mensaje "Introduce el valor a buscar", y se debe regresar el mensaje: "Elemento encontrado en la posición", seguido de la posición en la que se encuentra el número ingresado. En caso de que el elemento ingresado sea distinto a 3, 7, 2, 9 y 4, se debe imprimir "Elemento no encontrado en el arreglo".	10 pts
Unidad 6	Actividad 1 (colaborativa)	21 de mayo	Responde las siguientes preguntas en el foro: - ¿Qué es la notación O grande (Big-O)?	5 pts

			- De acuerdo con la notación O grande, ¿cuál es la complejidad de los dos algoritmos implementados en actividades anteriores (burbuja y búsqueda secuencial)?	
			Ponderación total	70 pts

BIBLIOGRAFÍA SUGERIDA

- Cairó. O. y Guardati, S. (2006). *Estructuras de datos.* McGraw-Hill Interamericana.
- Joyanes, L. (2020). *Fundamentos de programación: Algoritmos, estructura de datos y objetos* (2da ed.). McGraw-Hill.

EXÁMENES

- **Exámenes Parciales:**

Deberás haber entregado las actividades correspondientes al parcial que presentaras en las fechas establecidas por el profesor

- **Es importante que te inscribas a los exámenes en la fecha que te corresponde, ya que no podrás presentarlos en un periodo diferente al que se marca en la programación.**

PARCIAL	UNIDADES (que lo integran)	VALOR (núm. enteros)	FECHA DE APLICACIÓN
1ro.	1, 2, 3 y 4	20 pts	**Del 24 al 26 y del 28 al 30 de abril del 2005**
2do.	5 y 6	10 pts	**Del 26 al 31 de mayo de 2025**

- **Global. Examen único**

Valor	Requisitos	Aplicación de global
100%	Ninguno	6,7 y del 9 al 12 de junio de 2025

PORCENTAJES Y ESCALA DE EVALUACIÓN Y ACREDITACIÓN

Concepto	Porcentajes
Actividades de aprendizaje	65 %
Actividades colaborativas	5 %
Exámenes parciales	30 %
Total	**100 %**

- **Escala de evaluación:**

Rango	Calificación
1.00 a 5.99	**5**
6.00 a 6.49	**6**
6.50 a 7.49	**7**
7.50a 8.49	**8**
8.50 a 9.49	**9**
9.50 a 10.00	**10**

FUNCIONES DEL ASESOR

Por apoyar tu proceso de aprendizaje autónomo, el asesor tiene las siguientes funciones:

1. Apoyar y guiar en la resolución de dudas y desarrollo de actividades; a través de los canales de comunicación oficiales.
2. Calificar y retroalimentar las actividades en plataforma educativa en un lapso no mayor a **ocho días hábiles** después de la fecha de entrega establecida en el calendario.

3. Recomendar recursos didácticos para ampliar tu conocimiento. No es su obligación facilitarte: copias, libros, archivos digitales o proporcionarte ligas directas de la BIDI.
4. Enviar las calificaciones al finalizar el semestre de manera personalizada por correo electrónico.

DATOS DEL ASESOR O GRUPO DE ASESORES

Nombre	Correo electrónico
Joaquin Navarro Perales	joaquin_navarro@cuaed.unam.mx

Enseñar no es transferir conocimiento, sino crear las posibilidades para su propia producción o construcción.
Paulo Freire