ESTRUCTURA DE DATOS CON POO
SÍLABO

I. DATOS GENERALES

CARRERA PROFESIONAL	**:**	INGENIERÍA INDUSTRIAL SISTEMAS E INFORMÁTICA
NOMBRE DE LA ASIGNATURA	**:**	ESTRUCTURA DE DATOS CON POO
Nro. DE HORAS TOTALES	:	6 HORAS SEMANALES
Nro. DE HORAS TEORÍA	:	2 HORAS SEMANALES
Nro. DE HORAS PRÁCTICA	:	4 HORAS SEMANALES
CICLO	**:**	VI CICLO
TIPO DE CURSO	**:**	OBLIGATORIO
DURACIÓN DEL CURSO	**:**	16 SEMANAS EN TOTAL
CURSO REGULAR	**:**	17 SEMANAS
EXAMEN SUSTITUTORIO	**:**	1 SEMANA

II. DESCRIPCIÓN DE LA ASIGNATURA

Esta asignatura de carácter obligatorio y de naturaleza teórico práctica ha sido diseñada para proporcionar al estudiante de ingeniería de sistemas, informático, computación y carreras afines, el aprendizaje de algoritmos (desarrollo lógico para resolver problemas que requieren de un computador, independiente del lenguaje), desde una óptica de la programación orientada a objetos. Para el afianzamiento del aprendizaje de dichos algoritmos, se hace uso de un lenguaje de programación vigente (en C#) que utilice las técnicas de la programación orientada a objetos: las técnicas de encapsulación y abstracción de datos, la noción de herencia y polimorfismo para el desarrollo

de un verdadero código reusable y extensible.
Para el desarrollo de la asignatura se ha dividido en 5 unidades:

III. OBJETIVOS

La asignatura pretende ofrecer un escenario adecuado para que los estudiantes consoliden los conocimientos de algoritmos en la programación

Unidad 1.	**Clases y Objetos** Concepto de Objeto y Clase. Propiedades y métodos. Miembros de la clase. Sobrecarga: de constructores, de métodos y de operadores.
Unidad 2	**Arreglos** Arrays Unidimensionales (Vectores) y las operaciones más importantes con un vector. Vectores paralelos, operaciones. Utilizar vectores contadores y acumuladores. Operaciones con varios vectores. Vectores en clases definidas por el usuario. Arreglos bidimensionales (matrices). Las operaciones más comunes con una matriz y con varias matrices. Matrices en clases definidas por el usuario.
Unidad 3.	**Cadenas y Fechas** La clase System.String: El Struct System.char La clase Datetime.
Unidad 4.	**Estructuras** Estructuras - tipos de datos definidos por el usuario
Unidad 5.	**Tipos Abstractos de Datos TAD** TAD Listas Enlazadas TAD Pilas TAD Colas

de computadores, adquiridos en la asignatura previa de Fundamentos de Computación e Informática, del plan de estudios, de la escuela de Ingeniería de Sistemas, para que aprendan nuevas tecnologías informáticas, y para que desarrollen la habilidad de resolver problemas de programación de tamaño medio en equipo.

OBJETIVOS GENERALES

Al concluir el desarrollo de la asignatura se espera que los alumnos logren como objetivos generales adquirir la capacidad de:

ESTRUCTURA DE DATOS CON POO

- Elaborar algoritmos básicos para la programación de computadoras, incidiendo en la secuencia lógica para la resolución de problemas.
- Codificar pequeños programas usando la sintaxis básica del lenguaje C# y las construcciones típicas, accediendo a estructuras de datos básicas (vectores, matrices, struct, pilas, colas y listas enlazadas).
- Definir una clase, en la que se incluyen propiedades y métodos con paso de parámetros (incluso de objetos)
- Planificar un proyecto de programación en grupo (identificar fases, repartir las tareas de las distintas fases entre los miembros del grupo, e integrar los resultados de las diferentes tareas)

Los objetivos de grupo, es decir los alumnos como integrantes de un grupo deben ser capaces de:

- Decidir el tipo de estructura de control necesaria para resolver problemas que requieran de una computadora.
- Decidir sobre la implementación de una aplicación (estructuras de datos, organización de la aplicación, algoritmos a usar, etc.). -
- Programar aplicaciones con tecnología de orientación a objetos y con interfaces visuales.
- Buscar la información necesaria para realizar una tarea y aprender de forma autónoma.
- Depurar, poner a punto y documentar correctamente el código desarrollado.

OBJETIVOS ESPECÍFICOS

Cada alumno al finalizar esta asignatura debe:

Unidad 1:

- Diferenciar las clases de los objetos.
- Conocer y utilizar los métodos y propiedades static de una clase.
- Saber declarar una clase.
- Saber cómo declarar y crear una instancia de una clase.
- Saber Declarar y utilizar métodos declarados en otras clases.
- Utilizar constructores y métodos sobrecargados.
- Saber declarar y utilizar métodos con pasos por valor, por referencia y con parámetros de salida.
- Conocer las propiedades y métodos de la clase Math.
- Conocer y utilizar operadores sobrecargados.

ALGORITMOS Y ESTRUCTURAS DE DATOS

Unidad 2:

- Diferenciar los dos tipos de arrays mas usados (unidimensional o vector y bidimensional o matriz).
- Utilizar las principales operaciones: Lectura, recorrido, calculo de máximos y mínimos, ordenamiento, búsqueda secuencial y binaria.
- Utilizar los Vectores paralelos.
- Utilizar Vectores como contadores y como acumuladores
- Hacer operaciones con varios vectores: suma, resta, producto, etc.
- Conocer las principales operaciones con matrices: lectura, asignación, recorrido,
- Saber sumar filas y sumar columnas
- Hacer operaciones con varias matrices: suma, resta, producto de vector por matriz, por escalar, etc.
- Usar matrices como contadores y acumuladores.
- Saber utilizar los arrays de objetos
- Saber utilizar los arrays como campos de datos.
- Saber utilizar arrays como parámetros

Unidad 3:

- Tratar una cadena de caracteres como un array de caracteres.
- Conocer como invocar a los métodos y propiedades más comunes de la clase string; concatenar, subcadena, eliminar cadenas, convertir a mayúsculas y minúsculas, etc.
- Conocer y utilizar los principales métodos de la clase DateTime.

Unidad 4:

- Declarar un struct y declarar y utilizar variables de tipo struct
- Conocer la diferencia de usar struct y class.
- Conocer y utilizar datos estructurados combinados: class y struct, struct –struct (struct anidados).

Unidad 5:

- Valorar la importancia de la programación con Tipos Abstractos de Datos (TAD) como base del diseño modular, diferenciando los conceptos de especificación e implementación de un TAD.
- Conocer y realizar las implementaciones de los TAD a partir de su especificación, utilizando el paradigma de la POO y el lenguaje de programación C#
- Conocer las estructuras de datos lineales fundamentales y los algoritmos principales que se utilizan para su manipulación.
- Tener la capacidad de elección de la estructura de datos adecuada para cada tipo de problema, y los algoritmos que la gestionan.

ESTRUCTURA DE DATOS CON POO

- Aprender a combinar diferentes estructuras de datos para resolver problemas complejos

Los grupos de alumnos deben ser capaces de:

- Tomar decisiones sobre la implementación de una aplicación (estructuras de datos, organización de la aplicación, algoritmos, etc.), y justificar con claridad y convicción las decisiones tomadas.
- Planificar adecuadamente un proyecto de programación en grupo (identificar fases, repartir las tareas de las distintas fases entre los miembros del grupo, e integrar los resultados de las diferentes tareas)
- Analizar de forma crítica el trabajo realizado (el propio y el de otros), e identificar puntos fuertes y puntos débiles.
- Programar aplicaciones con tecnología de orientación a objetos.
- Buscar la información necesaria para realizar una tarea y aprender de forma autónoma.
- Depurar, poner a punto y documentar correctamente el código desarrollado, que debe ser correcto, robusto y amigable.

IV. METODOLOGÍA

- **MODALIDAD PRESENCIAL**

El docente y los alumnos como actores principales del proceso de enseñanza aprendizaje participan de la siguiente manera:

El docente: El primer día de clases hará la presentación introductoria del curso y explicara la metodología de trabajo a lo largo del ciclo. Se promoverá la participación constante de los alumnos.

El alumno: Deberá ingresar a clases habiendo leído sobre el tema a desarrollar en cada clase, con la separata que el profesor le entregará con anticipación. La separata tiene parte teórica y parte práctica (laboratorio) en la que se indicará al alumno lo que debe desarrollar en cada laboratorio.

CLASE TEÓRICA: Las clases teóricas se desarrollarán en el aula (2 hrs/semana) usando una computadora y data display en la que se desarrollara la teoría y los algoritmos que resuelven problemas reales. Asimismo se mostrará la codificación en el lenguaje de programación C# y se correrá el programa para que el alumno verifique la teoría.

ESTRUCTURA DE DATOS CON POO

CLASE DE LABORATORIO: En las clases de laboratorio (2 hrs/semana) cada alumno usará su separata para correr por lo menos 5 problemas resueltos en pseudocódigo y codificado en C# y resolverá otros 5 problemas propuestos (cuyo avance será revisado por el profesor).

TRABAJOS DE MEDIO CICLO Y FIN DE CICLO: El día del examen parcial y una semana antes del examen final respectivamente, cada alumno entregará resueltos cinco problemas propuestos por el docente y cinco problemas libres que el alumno proponga acerca de los temas desarrollados en clase, anteriores al parcial y final respectivamente resueltos en pseudocódigo y en C#. Se deberá incluir en la presentación del trabajo una prueba de la corrida del programa. La calificación dependerá del grado de complejidad que muestren los problemas libres presentados. Los trabajos se presentarán en grupos de máximo 3 alumnos.

Por la naturaleza del curso, tanto en las clases teóricas como en las de laboratorio, se incidirá en el método de resolución de problemas.

En cuanto a las técnicas: Se emplearan las exposiciones grupales e individuales para la sustentación de sus trabajos, y la participación activa para la resolución de problemas tanto en el aula como en el laboratorio.

Los alumnos tendrán como fuente de consulta las separatas y los manuales recomendadas en la sección Bibliografía de este silabo.

El profesor motivará al alumno para que participe espontáneamente durante el desarrollo del curso y resaltara la importancia de la investigación en los diferentes temas tratados.

Los ejercicios desarrollados en clase serán sobre casos de la vida real y se citarán analogías para permitirles fijar mejor los conceptos. Se solicitará a los alumnos una monografía (o práctica calificada PC), como mínimo, sobre un tema previamente acordado con el profesor para que se familiaricen con la forma de trabajar de los profesionales de su carrera.

V. EVALUACIÓN

MODALIDAD PRESENCIAL

El reglamento vigente de la universidad exige la asistencia obligatoria a clases; el 30% de inasistencias inhabilita al alumno a continuar en el curso, colocando como promedio final: NSP.

El docente deberá tomar lista en cada clase que dicta registrando las asistencias en el sistema que le proporciona la Universidad.

Dada la naturaleza del curso respecto a que imparte conocimientos pero además es de suma importancia la transmisión directa de la experiencia de profesor y que los alumnos participen en clase, se reitera que es de vital importancia la asistencia a clases.

La justificación de las inasistencias sólo serán aceptadas con el informe que pueda elevar la Oficina de Coordinación Académica EAPISI al profesor del curso.

Finalmente, debe quedar perfectamente entendido que sólo cuando el alumno asiste a clases, gana el derecho a ser evaluado y que en todo momento estará presente la normatividad expresada en el reglamento de la Universidad.

La modalidad de Evaluación será la siguiente:

La nota final se establecerá del promedio ponderado de:

NF = 30%EP + 30%EF + 40%PPT

N.F. = Nota final
E.P. = Nota Examen Parcial (30%)
E.F. = Nota Examen Final (30%)
P.P.T. = Promedio de Prácticas y Trabajos (40%)

En el Promedio de Prácticas y Trabajos (P.P.T.), estarán incluídas la Práctica 1, Práctica 2 (prácticas obligatorias programadas por la universidad), además de las prácticas y trabajos adicionales que el docente considere pertinente.

Solamente se considerará el redondeo de decimales para la Nota Final (N.F.).
El examen Sustitutorio (ES), será tomado en la semana 18 del ciclo y consiste

en la evaluación teórico - práctico de conocimiento de todo el curso y donde el alumno dará sus respuestas por escrito.

La nota obtenida en el examen Sustitutorio, podrá reemplazar la nota más baja que el alumno haya obtenido en el examen Parcial o Examen Final y de proceder el reemplazo, se recalculará la nueva nota final (N.F.).

En caso la nota del Examen Sustitutorio sea más baja que el Examen Parcial o Examen final, no se reemplazará ninguna de ellas, quedando el alumno con la nota obtenida hasta antes del examen Sustitutorio.

En todas las evaluaciones se calificará con una escala de 0 a 20 siendo la nota mínima aprobatoria 11 (once).

Es de total aplicación el Reglamento de Estudios de la Universidad entregado al alumno.

- **MODALIDAD A DISTANCIA**

Estimado alumno, dada la naturaleza del curso, es muy importante su participación activa en el proceso de aprendizaje. Por ello, se define en este acápite los criterios de evaluación:

➢ **Exámenes**

Examen es la evaluación escrita del presente curso, se evalúa bajo una escala vigesimal y se dará según como se señala en el siguiente cuadro.

Exámenes	Semana de estudios
Examen Parcial	Cuarta
Examen Final	Octava
Examen Sustitutorio	Decimooctava

La nota mínima aprobatoria de los exámenes tanto parcial como final es de once (11). La máxima calificación a obtenerse en el examen sustitutorio es veinte (20) y la nota mínima aprobatoria del mismo es once (11). Es importante resaltar que la calificación obtenida en el examen sustitutorio reemplazará a la nota del Examen Parcial o al Examen Final. Usted solo podrá acceder al examen sustitutorio si no ha sido evaluado en el examen parcial o en el examen final o haya desaprobado alguno de ellos. Solamente el alumno podrá decidir si rinde el Examen Sustitutorio ya sea para aprobar el curso o para subir su promedio.

ALGORITMOS Y ESTRUCTURAS DE DATOS

- **Actividades Obligatorias**

Son los trabajos que usted entregará obligatoriamente y que es requisito indispensable para aprobar el curso. Existirán actividades obligatorias cuyo desarrollo requiere trabajo en grupo, en otros casos el desarrollo será de forma personal. Las actividades obligatorias serán colocadas en el campus virtual y las aplicaciones de las mismas serán detalladas oportunamente en el foro y en la sala de conversación, así como también el asesoramiento en su desarrollo.
Cada una de las actividades obligatorias se evaluará sobre la escala de 0 a 20 siendo la nota mínima aprobatoria 11 (once). Toda copia de trabajos de Internet detectada en las actividades tendrá la nota 00 (cero)

Pesos :

- Examen Parcial. (35%)
- Examen Final. (35%)
- Actividad Obligatoria (30%)

- **NF = 35%EP + 35%EF + 30%AO**

El trabajo académico está constituido por la actividad obligatoria, cuyas especificaciones han sido dadas a conocer oportunamente.

VI. CONTENIDO DEL CURSO

UNIDAD 1: CLASES Y OBJETOS

SEMANA 01 Modalidad Presencial

1. Concepto de Objeto y Clase. Propiedades y métodos.
1.1 Definición de objeto
1.2 Definición de clase
1.3 Estructura de un programa aplicación en C#
1.4 Declaración de clases – sintaxis.
1.5 Declaración y creación de objetos - sintaxis.

2. Miembros de la clase.
2.1 Estructura de una clase creada por el usuario

a) Los modificadores de acceso.
2.2 Campos de Datos.
2.3 Concepto de Constructor de la clase.
a) Constructor por defecto.
2.4 Método (función o procedimiento).
a) Definición de Método.
b) La estructura de la declaración de un método -Sintaxis
c) Clasificación de los métodos:
- Según el valor del retorno
- Métodos que retornan valor (función)
- Métodos que no retornan valor-void.(procedimiento)
- Según la forma de acceso al método:
- Métodos static.
- Métodos Normales o No static
d) Acceso a los métodos static
- Acceso a los Métodos STATIC definidos en la propia clase:
- Acceso a los Métodos STATIC definidos en otra clase.
e) Acceso a Datos y Métodos NORMALES (No static) de una clase definida por el usuario

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA :

¿Por qué es importante para el Ingeniero de Sistemas el conocimiento y uso de clases y objetos en el desarrollo de software?

SEMANA 02 Modalidad Presencial

f) Los Métodos – Paso de parámetros.
- Paso de parámetros por valor,
- Paso de parámetros por referencia y
- Paso de parámetro de salida.

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA :

¿Por qué es importante para el Ingeniero de Sistemas poder definir los métodos dentro de las clases en un lenguaje de programación particular?

SEMANA 03 Modalidad Presencial

3. Sobrecarga
3.1 sobrecarga de constructores

-Referencia al objeto actual con this
3.2 Sobrecarga de métodos
3.3 sobrecarga de operadores

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas poder definir los métodos dentro de las clases en un lenguaje de programación particular?

UNIDAD 2: ARRAYS.
ARRAYS UNIDIMENSIONALES (ESTRUCTURAS DE VARIAS VARIABLES)

SEMANA 04 Modalidad Presencial

1. Introducción
2. Definición
3. Clasificación.
4. Array unidimensional

4.1 Operaciones en un vector

a) Declaración de un array Unidimensional,
b) Creación (o Instanciación) de un arreglo unidimensional
c) Inicialización del array
d) Utilización de los elementos del array.
- Asignación directa.
- Recorrido: leer, escribir, acumular, contar, máximos y mínimos.
- Actualización:
  Inserción
  Eliminación
- Búsqueda secuencial y binaria
- Ordenamiento – Método de Burbuja

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas conocer y manejar los arreglos unidimensionales?
¿Necesita el Ingeniero de Sistemas conocer métodos de ordenamiento y búsqueda con arreglos unidimensionales?

SEMANA 05 Modalidad Presencial

4.2 Vectores paralelos aplicaciones
4.3 Vectores contadores y acumuladores
4.4 Operaciones básicas con arreglos y escalares
a) Multiplicación de arreglo por escalar
b) Suma o resta de arreglos
c) Producto escalar de dos vectores

Practica Calificada 1 (tema vectores)

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas conocer y manejar los arreglos unidimensionales paralelos y vectores contadores y acumuladores?
¿Por qué es importante para el Ingeniero de Sistemas conocer las operaciones con arreglos y escalares?

SEMANA 06 Modalidad Presencial

4.5 Arreglo como Campos datos en una clase.
4.6 Arreglo de Objetos (de una clase)
4.7 Objeto de una clase como campo de dato de otra clase
4.8 Arreglo de objetos de una clase como campos de datos de otra clase.
4.9 Arreglos unidimensionales - paso de parámetros que son arreglos

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas conocer las diversas formas como se presentan los arreglos unidimensionales dentro de las clases, como arreglo de objetos y como parámetros?

SEMANA 07 Modalidad Presencial

5. ARRAYS BIDIMENSIONALES (MATRIZ O TABLA)
5.1 Introducción.
5.2 Arreglos de dos dimensiones – Bidimensional (matriz)
5.3 Operaciones en una matriz
a) Declaración de un array Bidimensional.
b) Creación (o Instanciación) de un arreglo Bidimensional.
c) Inicialización del array bidimensional
d) Utilización de los elementos del array bidimensional
- Asignación

ALGORITMOS Y ESTRUCTURAS DE DATOS

- Recorrido (barrido o visitado): lectura, mostrar, mínimo, máximo, etc.
- Suma de filas y suma de columnas

e) Operaciones con matrices

- Suma y Resta de Matrices
- Multiplicación de la matriz A por un escalar k

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas conocer y manejar los arreglos bidimensionales?

UNIDAD 3: CADENA Y FECHAS. STRINGS COMO OBJETOS: LITERALES Y OPERADORES

SEMANA 08 Modalidad Presencial

1 Introducción.
2 Definición de cadenas de texto en C#. string como objetos.
Concatenación de cadenas. Propiedad Length.
3.La clase System.String:
3.1 Constructores sobrecargados.
3.2 Propiedades.
a) length
b) Acceso a los datos almacenados en la cadena.
c) Métodos de la clase string:

- Método Copy()
- Método sobrecargado Substring()
- Método sobrecargado ToCharArray()
- Método EndsWith() y StartsWith()
- Método sobrecargado int IndexOf ( )
- Método sobrecargado Replace()
- Metodo public string ToLower () , ToUpper()
- Metodo Join()
- Metodo Insert ()
- Método Remove()
- Metodo string.Split
- Metodo compare()

4. Tipo char. Struct system.char.
Métodos:

I git()
s IsLetter()
D IsUpper() - IsLower
i ToString.

5. La estructura datetime
5.1 Constructores
5.2 Propiedades
5.3 Metodos
a) Parse
b) Tostring()
c) DaysInMonth ()

Practica Calificada 2 (matrices y cadenas - fechas)

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA :

¿Por qué es importante para el Ingeniero de Sistemas el manejo de cadenas, fechas y horas en el desarrollo de software?

SEMANA 09 Modalidad Presencial - SEMANA 04 Modalidad a Distancia

Examen Parcial (toda la primera parte)

UNIDAD 4: TIPOS DE DATOS DEFINIDOS POR EL USUARIO - ESTRUCTURAS

SEMANA 10 Modalidad Presencial

1. Introducción
2. DEFINICIÓN DE STRUCT
3. Declaración de un Tipo de dato struct
4. Declaración de una variable de tipo struct
5. Acceso a los campos de datos y métodos public
6. Combinaciones de struct – class y struct –struct
   Arreglos de datos de tipo struct (array de struct).
   La clase con campos de datos de tipo struct
   Struct con campos de datos arrays.
   Struct anidados (Campo de dato struct declarado dentro de otro struct)

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA :

¿Por qué es importante para el Ingeniero de Sistemas la integración de arreglos, estructuras y objetos en el desarrollo de aplicaciones?

SEMANA 11 Modalidad Presencial

Exposición de trabajos

UNIDAD 5. TIPOS ABSTRACTOS DE DATOS

SEMANA 12 Modalidad Presencial

INTRODUCCIÓN
Tipo Abstracto de Dato TAD – Estructuras Lineales
1. LISTAS ENLAZADAS.
1.1 Definición.
1.2 Terminología
1.3 Composición de los elementos.
1.4 Orden cronológico
1.5 Capacidad
1.6 Operaciones.
1.7 Tipos de Listas Enlazadas.

a) Lista de Enlace Simple – representación en memoria
- Símbolo
- Nodo
- Notación
- Implementación
- Operaciones:
  - Operación de creación
  - Operación ListaVacía()
  - Operación Recorrido ()
  - Operación de búsqueda ()
  - Operación de inserción: Inserción al inicio de la Lista, Inserción al Final de la Lista, Inserción anterior a otro cuya información es Ref
  - Operación de eliminación: Eliminación al Inicio, Eliminación al Final, Eliminación de un nodo específico llamado Dato

  Implementación usando el lenguaje C#

b) Lista doblemente enlazada

c) Lista circular.

Aplicaciones

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA :

¿Por qué es importante para el Ingeniero de Sistemas el conocimiento de las listas enlazadas?

SEMANA 13 Modalidad Presencial

2. PILAS.
2.1 Definición.
2.2 Orden cronológico
2.3 Capacidad.
2.4 Aplicaciones.
2.5 Operaciones.

a) CrearPila,
b) PilaVacia,
c) PilaLlena,
d) Apilar,
e) Desapilar.

Implementación usando el lenguaje C#

Practica calificada 3 (struct y listas enlazadas)

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas el conocimiento de las estructuras lineales pilas?

SEMANA 14 Modalidad Presencial

2.6 Aplicaciones de la estructura de datos PILAS:
Funciones recursivas (foro)
Aplicaciones: notaciones prefijo y sufijo para operaciones matemáticas.(foro)

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA:

¿Por qué es importante para el Ingeniero de Sistemas el conocimiento de las funciones recursivas?

SEMANA 15 Modalidad Presencial

3. COLAS
3.1 Definición TAD COLA

3.2 Orden Cronológico.

3.3 Capacidad

3.4 Operaciones.

3.5 Representación gráfica.

3.6 Tipos de colas

3.7 Operaciones de cola circular usando arreglos- codificación en C#

Operaciones en colas:

a. CrearCola()
b. ColaVacia()
c. ColaLlena()
d. Encolar()
e. Desencolar.

Aplicaciones

PERTINENCIA A LA INGENIERÍA DE SISTEMAS E INFORMÁTICA :

¿Por qué es importante para el Ingeniero de Sistemas el conocimiento de las estructuras lineales colas?

SEMANA 16 Modalidad Presencial

Entrega de informes finales y exposiciones.

Practica calificada 4 (pilas y colas)

SEMANA 17 Modalidad Presencial

EXAMEN FINAL
(Todo – incidiendo en temas struct, listas, pilas y colas)

SEMANA 18 Modalidad Presencial – Modalidad a distancia

EXAMEN SUSTITUTORIO

VII. BIBLIOGRAFÍA

Además de la bibliografía básica, la complementaria y la electrónica, el alumno tendrá acceso al uso del Internet para ampliar los temas de investigación y consulta que requiera.

A. BIBLIOGRAFÍA BÁSICA:

1. Joyanes. Fundamentos de Programación. Algoritmos de datos y Objetos. 4ta edición, España, 2008, 766 pág.
2. Gladys García Vilcapoma
Algoritmos Avanzados y Estructuras de datos
Dirección Universitaria de Educación a Distancia (DUED)
Impreso en los Talleres gráficos de la UAP
Editorial. UAP-FISI. Lima, 290 pág.

B. BIBLIOGRAFÍA COMPLEMENTARIA:

1. Ramírez. Introducción a la programación, Algoritmos y su implementación en VB. NET, C#, JAVA Y C++. 4ta edición, México, 2010, 488 pág.
2. Charte. Guía práctica de Visual C# 2005. 1ra edición, España, Anaya multimedia, 2007, 351 pág.
3. Deitel. Como programar en C#. 2da edición, México, Pearson. 2007, 1166 pág.
4. Sharp. Visual C# 2009 paso a paso.1ra edición, España, Anaya Multimedia, 2009, 832 pág.

C. BIBLIOGRAFÍA ELECTRÓNICA:

1. https://dued.uap.edu.pe/biblioteca_virtual.htm
2. http://www.devjoker.com/contenidos/Tutorial-C/125/Introduccion-a-C.aspx
3. http://msdn.microsoft.com/es-es/library/ms173109(VS.80).aspx