Departamento de Sistemas

Fecha de aprobación:

PROGRAMA ANALÍTICO

Nivel LICENCIATURA		Unidad de enseñanza-aprendizaje PROGRAMACIÓN ESTRUCTURADA	
Clave 1151038			
Horas de teoría 2.5	Horas de práctica 2.0	Seriación 1112013 y 1112027	Créditos 7

	Licenciatura en Ingeniería...	Ambiental	Civil	En Computación	Eléctrica	Electrónica	Física	Industrial	Mecánica	Metalúrgica	Química
OBLIGATORIA											
Tronco General	X	X	X	X	X	X	X	X	X	X	X
Tronco Básico Profesional											
Área de Concentración											
OPTATIVA											
General											
Área de Concentración											
Otros											
TRIMESTRE											

Observaciones

OBJETIVOS:

Al finalizar el curso el alumno será capaz de:

Describir los conceptos de algoritmo y de programa.
Explicar el paradigma de programación estructurada.
Explicar, elaborar y representar algoritmos.
Implementar programas escritos en lenguaje C, usando el paradigma de programación estructurada.
Desarrollar los programas usando el ambiente UNIX.

CONTENIDO SINTÉTICO:

1. Algoritmos y programas.
2. Diseño de programas estructurados.
3. Ambiente de desarrollo UNIX.
4. Programación modular.
5. Elementos básicos de un programa en lenguaje C.
6. Estructuras de decisión.
7. Estructuras de repetición.
8. Arreglos y estructuras.
9. Cadenas de caracteres.
10. Archivos.

TEMA 1. ALGORITMOS Y PROGRAMAS

OBJETIVOS ESPECÍFICOS:

1. Distinguir los elementos de un problema.
2. Definir el concepto de algoritmo.
3. Conocer diversas formas de representar un algoritmo.
4. Entender los conceptos de programa y lenguaje de programación.

CONTENIDO:

1. Elementos de un problema.
    a. Descripción del problema.
    b. Datos de entrada.
    c. Datos de salida.
2. Definición de algoritmo.
3. Representación de algoritmos.
    a. Diagramas de flujo.
    b. Seudocódigo.
4. Programas y lenguajes de programación.
    a. Lenguajes de alto nivel y el proceso de compilación.
    b. Independencia del algoritmo con el lenguaje de programación.
    c. Capacidades y límites de una computadora.

REFERENCIAS:

G.J. Bronson, C++ para ingeniería y ciencias, México: International Thomson Editores, 2000. Capítulo 1.

C. Gregorio Rodríguez et al., Ejercicios de programación creativos y recreativos en C++, España: Prentice-Hall, 2002. Capítulo 1.

HORAS DE CLASE:

4.5 horas

OBSERVACIONES:

Se recomienda ver al menos dos algoritmos diferentes para problemas similares a los siguientes:

- Cálculo de funciones trigonométricas.
- Resolución de ecuaciones de primer grado.
- Cálculo de valor absoluto.

Los algoritmos pueden ser descritos mediante seudocódigo o alguna otra representación adecuada.

TEMA 2. DISEÑO DE PROGRAMAS ESTRUCTURADOS

OBJETIVOS ESPECÍFICOS:

1. Entender el concepto de variable.
2. Definir el concepto de flujo del programa.
3. Definir programación estructurada.

CONTENIDO:

1. Declaración y uso de variables.
    a. Nombre de una variable.
    b. Asignación de valores.
2. Flujo del programa.
    a. Ejecución secuencial.
    b. Ejecución no secuencial.
3. Definición de programación estructurada.
    a. Bloques de sentencias.
    b. Sentencias de selección.
    c. Sentencias de repetición.
    d. Llamadas a subrutina.

REFERENCIAS:

F. García Carballeira et al., El lenguaje de programación C – Diseño e implementación de programas, España: Prentice-Hall, 2002. Capítulos 1 y 7.

C. Gregorio Rodríguez et al., Ejercicios de programación creativos y recreativos en C++, España: Prentice-Hall, 2002. Capítulo 2.

HORAS DE CLASE:

3.0 horas

OBSERVACIONES:

Se recomienda describir mediante seudocódigo o alguna otra representación adecuada algoritmos que resuelvan problemas similares a los siguientes:

- Conversiones entre escalas de temperaturas.
- Cálculo de las funciones mínimo y máximo.
- Cálculo de las funciones factorial y sumatoria.

TEMA 3. AMBIENTE DE DESARROLLO UNIX

OBJETIVOS ESPECÍFICOS:

1. Entender los conceptos de archivo y directorio del ambiente UNIX.
2. Ser capaz de manipular directorios y archivos desde la consola de comandos.
3. Ser capaz de ejecutar aplicaciones desde la consola de comandos.

CONTENIDO:

1. Archivos y directorios en UNIX.
    a. Concepto de archivo en UNIX.
    b. Tipos de archivos y directorios como contenedores de archivos.
    c. Distinción entre nombre y ruta de un archivo o directorio.
2. La consola de comandos.
    a. Los comandos `ls` y `cd`.
    b. Los comandos `cat` y `mkdir`.
    c. Los comandos `rm` y `rmdir`.
    d. Los comandos `cp` y `mv`.
3. Aplicaciones y programas en UNIX
    a. Búsqueda de programas invocados y la carpeta `bin`.
    b. Ejecución de programas de otras carpetas.
    c. Invocación de programas con argumentos de línea de comandos.

REFERENCIAS:

B.W. Kernighan y R. Pike, El entorno de programación UNIX, México: Prentice-Hall Hispanoamericana, 1987. Capítulos 1, 2 y 3.

Oram y M. Loukides, Programming with GNU software, EUA: O´Really Media, 1996. Capítulos 1, 2, 3 y 4.

HORAS DE CLASE:

3.0 horas

OBSERVACIONES:

Se recomienda realizar prácticas basadas en los ejemplos de los capítulos 2, 3 y 4 de la segunda referencia del tema.

Como práctica adicional se recomienda instalar una consola de comandos que permita el acceso a una cuenta de UNIX hospedada en una computadora remota.

TEMA 4. PROGRAMACIÓN MODULAR

OBJETIVOS ESPECÍFICOS:

1. Definir el concepto de función.
2. Entender el concepto de reutilización de código.
3. Definir programación modular.

CONTENIDO:

1. Funciones.
    a. Funciones como cajas negras.
    b. Dominio y parámetros de una función.
    c. Codominio y valor de retorno de una función.
2. Reutilización de código.
    a. La función como unidad de reutilización.
    b. Importancia de la reutilización en ingeniería.
    c. Bibliotecas de software como ejemplos de reutilización.
3. Definición de programación modular.

REFERENCIAS:

F. García Carballeira et al., El lenguaje de programación C – Diseño e implementación de programas, España: Prentice-Hall, 2002. Capítulo 7.

C. Gregorio Rodríguez et al., Ejercicios de programación creativos y recreativos en C++, España: Prentice-Hall, 2002. Capítulo 3.

HORAS DE CLASE:

1.5 horas

OBSERVACIONES:

Se recomienda describir mediante seudocódigo o alguna otra representación adecuada algoritmos que resuelvan problemas similares a los siguientes:

- Cálculo de las funciones sucesor y predecesor.
- Cálculo de las funciones piso y techo.
- Cálculo de la función redondea.

Se sugiere especificar el dominio y el codominio de las funciones vistas.

TEMA 5. ELEMENTOS BÁSICOS DE UN PROGRAMA EN LENGUAJE C

OBJETIVOS ESPECÍFICOS:

1. Conocer los tipos de dato básicos de C.
2. Ser capaz de definir variables, constantes y funciones en C.
3. Conocer las operaciones aritméticas disponibles en C.
4. Describir la biblioteca estándar de C y su mecanismo de inclusión.
5. Conocer los mecanismos de entrada y salida de la biblioteca estándar de C.

CONTENIDO:

1. Tipos de dato básicos del lenguaje C.
    a. Tipos enteros.
    b. Tipos flotantes.
    c. Tipos de caracteres.

2. Definición de variables y funciones.
    a. Definición de variables, inicialización y asignación.
    b. Definición de funciones.
    c. Definición de constantes.

3. Operaciones aritméticas de C.
    a. Suma y resta.
    b. Multiplicación, división y residuo.
    c. Precedencia de operadores.

4. Biblioteca estándar del lenguaje C.
    a. Organización y división de la biblioteca estándar.
    b. La instrucción `include`.

5. Entrada y salida de la biblioteca estándar.
    a. La función `scanf` y entrada con formato de tipos básicos.
    b. La función `printf` y salida con formato de tipos básicos.

REFERENCIAS:

F. García Carballeira et al., El lenguaje de programación C – Diseño e implementación de programas, España: Prentice-Hall, 2002. Capítulos 3 y 4.

F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011. Capítulo 1.

HORAS DE CLASE:

6.0 horas

OBSERVACIONES:

Se recomienda implementar en lenguaje C algunos de los ejemplos vistos anteriormente.

Además se recomienda resolver en clase tres de los problemas planteados en la segunda referencia de este tema.

TEMA 6. ESTRUCTURAS DE DECISIÓN

OBJETIVOS ESPECÍFICOS:

1. Ser capaz de especificar expresiones condicionales.
2. Expresar ejecución condicional en algoritmos usando la sentencia `if`.
3. Expresar ejecución condicional en algoritmos usando la sentencia `switch`.

CONTENIDO:

1. Expresiones condicionales.
   a. Operadores relacionales.
   b. Conjunciones, disyunciones y negaciones.
   c. Evaluación en corto circuito.
2. La sentencia `if`.
   a. Bifurcación del flujo del programa usando la sentencia `if`.
   b. La sentencia `else` como rama opcional de la bifurcación.
   c. Bifurcaciones encadenadas y construcción `else if`.
3. La sentencia `switch`.
   a. División multicamino del flujo del programa usando la sentencia `switch`.
   b. Las sentencias `case` y `default`.
   c. Comportamiento en cascada de la sentencia `switch`.
   d. Comportamiento de la sentencia `break` dentro de un `switch`.

REFERENCIAS:

G.J. Bronson, C++ para ingeniería y ciencias, México: International Thomson Editores, 2000. Capítulo 4.

F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011. Capítulo 2.

HORAS DE CLASE:

4.5 horas

OBSERVACIONES:

Se recomienda resolver en clase tres de los problemas planteados en la segunda referencia de este tema.

TEMA 7. ESTRUCTURAS DE REPETICIÓN

OBJETIVOS ESPECÍFICOS:

1. Entender el concepto de ciclo e iteración.
2. Expresar algoritmos iterativos utilizando las diferentes sentencias iterativas de C.
3. Controlar la ejecución del ciclo o iteración usando las sentencias `break` y `continue`.

CONTENIDO:

1. Ciclos e iteraciones.
    a. Estado del programa durante un proceso iterativo.
    b. Ciclos como grupos de iteraciones.
2. Sentencias iterativas de C.
    a. El ciclo `while` como sentencia iterativa condicionada en la parte superior del bloque.
    b. El ciclo `do` como sentencia iterativa condicionada en la parte inferior del bloque.
    c. El ciclo `for` como simplificación sintáctica para ciclos contadores.
3. Sentencias `break` y `continue`.
    a. Ejemplos de ciclos condicionados en la parte interna del bloque y la sentencia `break`.
    b. Comportamiento de la sentencia `continue` dentro de un ciclo.

REFERENCIAS:

G.J. Bronson, C++ para ingeniería y ciencias, México: International Thomson Editores, 2000. Capítulo 5.

F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011. Capítulos 3 y 4.

HORAS DE CLASE:

6.0 horas

OBSERVACIONES:

Se recomienda resolver en clase dos problemas de cada capítulo sugerido de la segunda referencia de este tema.

TEMA 8. ARREGLOS Y ESTRUCTURAS

OBJETIVOS ESPECÍFICOS:

1. Entender el concepto de arreglo.
2. Entender el concepto de apuntador y su relación con los arreglos de C.
3. Entender el concepto de estructura de C.

CONTENIDO:

1. Arreglos.
    a. Arreglos como colecciones homogéneas de variables enumeradas.
    b. Acceso a elementos de un arreglo como desplazamientos en memoria.
    c. Declaración, inicialización y uso de arreglos en C.
2. Apuntadores y arreglos.
    a. Direcciones de memoria y apuntadores.
    b. Desreferencia de apuntadores y el apuntador nulo.
    c. Conversión implícita entre arreglo y apuntador.
    d. Paso por referencia usando apuntadores.
3. Estructuras.
    a. Estructuras y tipos compuestos definidos por el usuario.
    b. Declaración de estructuras y las sentencias `struct` y `typedef`.
    c. Inicialización, asignación y acceso a miembros de una estructura.

REFERENCIAS:

B.W. Kernighan y D.M. Ritchie, El lenguaje de programación C, 2da. Edición, México: Prentice-Hall Hispanoamericana, 1995. Capítulos 5 y 6.

F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011. Capítulos 5 y 6.

HORAS DE CLASE:

6.0 horas

OBSERVACIONES:

Se recomienda resolver en clase dos problemas de cada capítulo sugerido de la segunda referencia de este tema.

TEMA 9. CADENAS DE CARACTERES

OBJETIVOS ESPECÍFICOS:

1. Entender el concepto de cadena de caracteres de C.
2. Conocer las funciones de clasificación y conversión de caracteres de la biblioteca estándar de C.
3. Conocer las funciones de manejo de cadenas de la biblioteca estándar de C.

CONTENIDO:

1. Cadenas de caracteres.
    a. Cadenas de caracteres como arreglos y como secuencias delimitadas.
    b. Diferencia entre tamaño y capacidad de una cadena.
    c. Lectura con formato de palabras y líneas.
2. Funciones para manejo de caracteres.
    a. El código ASCII.
    b. Funciones de la cabecera estándar `ctype.h`.
3. Funciones para manejo de cadenas.
    a. La función `strlen`.
    b. La función `strcpy`.
    c. La función `strcmp`.
    d. La función `strcat`.

REFERENCIAS:

F. García Carballeira et al., El lenguaje de programación C – Diseño e implementación de programas, España: Prentice-Hall, 2002. Capítulo 9.

F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011. Capítulo 7.

HORAS DE CLASE:

4.5 horas

OBSERVACIONES:

Se recomienda resolver en clase tres de los problemas planteados en la segunda referencia de este tema.

TEMA 10. ARCHIVOS

OBJETIVOS ESPECÍFICOS:

1. Conocer el concepto de archivo de la biblioteca estándar de C.
2. Ser capaz de leer y manipular el contenido de archivos.

CONTENIDO:

1. El concepto de archivo de la biblioteca estándar de C.
    a. La entrada y salida estándares vistas como archivos.
    b. Redirección de la entrada y salida estándares.
    c. Salida de errores.
2. Manipulación de archivos.
    a. Modos de apertura de un archivo.
    b. Las funciones `fscanf` y `fprintf`.
    c. Fin de archivo y comportamiento de la función `feof`.
    d. Cierre de archivos.

REFERENCIAS:

F. García Carballeira et al., El lenguaje de programación C – Diseño e implementación de programas, España: Prentice-Hall, 2002. Capítulo 12.

F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011. Capítulo 8.

HORAS DE CLASE:

4.5 horas

OBSERVACIONES:

Se recomienda resolver en clase tres de los problemas planteados en la segunda referencia de este tema.

MODALIDADES DE CONDUCCIÓN DEL PROCESO DE ENSEÑANZA-APRENDIZAJE

Clase teórico-práctica a cargo del profesor con participación activa del alumno con al menos seis sesiones prácticas utilizando computadoras o modalidad SAI.

INFORMACIÓN ADICIONAL

MODALIDADES DE EVALUACIÓN

Al menos dos evaluaciones periódicas de resolución de problemas, ejercicios o preguntas conceptuales.
Elaboración y presentación de trabajos y programas.
No hay evaluación terminal.

Las reglas de evaluación serán presentadas en forma escrita por el profesor al inicio del curso.

Admite evaluación de recuperación, consistente en elaboración de programas, resolución de problemas, ejercicios o preguntas conceptuales.
No requiere inscripción previa.

INFORMACIÓN ADICIONAL

BIBLIOGRAFÍA NECESARIA O RECOMENDABLE

1. B.W. Kernighan y D.M. Ritchie, El lenguaje de programación C, 2da. Edición, México: Prentice-Hall Hispanoamericana, 1995.
2. B.W. Kernighan y R. Pike, El entorno de programación UNIX, México: Prentice-Hall Hispanoamericana, 1987.
3. C. Gregorio Rodríguez et al., Ejercicios de programación creativos y recreativos en C++, España: Prentice-Hall, 2002.
4. F.J. Zaragoza Martínez, 64 ejercicios de programación, México: UAM Azcapotzalco, 2011.
5. G.J. Bronson, C++ para ingeniería y ciencias, México: International Thomson Editores, 2000.
6. Oram y M. Loukides, Programming With GNU software, EUA: O´Really Media, 1996.
7. F.J. Cevallos, C/C++ Curso de Programación, 3ra. Edición, México: RA-MA Editorial, 2007.

BIBLIOGRAFÍA ADICIONAL

1. F. García Carballeira et al., El lenguaje de programación C – Diseño e implementación de programas, España: Prentice-Hall, 2002.

Este programa analítico fue elaborado por una comisión académica del Departamento de Sistemas integrada por los profesores

Aprobado

Jefe de Departamento
M. en C. Rafaela Blanca Silva López

Visto bueno

Director de División
Dr. Luis Enrique Noreña Franco