FACULTAD DE INGENIERIA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

RAYMUNDO HUGO RANGEL G

GUIA DE ESTUDIO PARA PRESENTAR EXAMEN EXTRAORDINARIO DE INGENIERIA EN PROGRAMACION

FACULTAD DE INGENIERIA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

RAYMUNDO HUGO RANGEL G

GUIA DE ESTUDIO PARA PRESENTAR EXAMEN EXTRORDINARIO DE INGENIERIA EN PROGRAMACION

DIVISION DE INGENIERIA MECANICA Y ELECTRICA
DEPARTAMENTO DE INGENIERIA EN COMPUTACION

La supervisión pedagógica de esta guía estuvo a cargo de la Lic. María Hano Roa del CENTRO DE SERVICIOS EDUCATIVOS de la Facultad de Ingeniería.

Ciudad Universitaria, D.F., abril de 1989.

CONTENIDO

PROPOSITO DE LA GUIA

Esta guía proporciona al alumno una orientación para que por iniciativa propia aborde los objetivos y contenidos de la materia de Ingeniería de Programación. La guía es útil para alumnos que tienen que acreditar la materia en un examen extraordinario.

Esta guía no contiene el desarrollo total de los contenidos de la materia. Sólo propone actividades que ayuden al alumno a asimilar en forma sistemática, crítica y autónoma los contenidos de la materia, y que por otra parte desarrolle habilidades específicas mediante el logro de los objetivos de aprendizaje propuestos.

Esta guía incluye temario y los objetivos tanto generales como específicos de la materia. Incluye además actividades de aprendizaje gradual cuya realización reforzarán los conceptos adquiridos.

1. INTRODUCCION

En la actualidad la materia de Ingeniería de Programación no es una materia obligatoria en la carrera de Ingeniero en -- Computación. Sin embargo, debido a su importancia central para el Ingeniero en Computación, se ofrece como una materia optativa regular. Esta materia se imparte en los últimos semestres (del octavo en adelante) y tiene como materias antecedentes:

Computadoras y Programación
Programación Estructurada y Características de Lenguajes
Estructura de Datos
Estructuras Discretas
Programación de Sistemas
Bases de Datos

y como consecuentes:

Ninguna

Hasta hace relativamente poco tiempo el desarrollo de sistemas de información se realizaba de una manera bastante artesanal, es decir, no existían métodos y herramientas que proporcionaran un marco de trabajo sistemático para un desarrollo ordenado. Esta carencia ocasionó que los sistemas de información nunca estuvieran terminados dentro del tiempo y costo estimados, - ni que cumplieran cabalmente con los requerimientos establecidos. Todo esto hizo sentir, la necesidad de controlar la llamada complejidad de los sistemas. Fué así como comenzó a tomar - forma un cuerpo de conocimientos que actualmente se conoce como Ingeniería de Programación (Software).

La Ingeniería de Programación (o Software) es una rama de la Ingeniería en Computación que se ocupa de proporcionar un --

marco de trabajo sistemático para el desarrollo de sistemas de programación a lo largo del llamado ciclo de vida de estos sistemas.

Los productos de la Ingeniería de Programación tienen los siguientes atributos de calidad:

Portables:	que el producto pueda transportarse con un mínimo de cambios de un computador a otro.
Confiables:	es decir, el producto realiza las funciones requeridas bajo condiciones establecidas en un período de tiempo establecido
Eficientes:	el producto realiza sus funciones con un mínimo de recursos de cómputo
Exactos:	es decir, la medida en que el producto está libre de errores
Tolerables:	la discrepancia entre un valor o condición computado y el valor verdadero, o especificado o teóricamente correcto
Robustos:	la medida en que el producto puede seguir operando correctamente, a pesar de la introducción de entradas inválidas
Correctos:	(1) la medida en que el producto está libre de defectos de diseño y defectos de calificación (2) la medida en que el producto se aviene a los requerimientos especificados (3) la medida en que el producto se aviene a las expectativas del usuario.

La importancia capital de esta materia reside en el hecho de que proporciona una metodología y herramientas ampliamente aplicables en el trabajo profesional del egresado de la carrera. Es por ello que esta materia es de gran importancia en la formación del Ingeniero en Computación. En la gran mayoría de los casos, el egresado, en su trabajo profesional aplicará estos conocimientos en el medio del proceso de datos. Medio en el cual se originaron tanto la metodología como las herramientas que ofrece el curso.

Evidentemente, los conocimientos de la materia tendrán una amplia aplicabilidad en el trabajo profesional del egresado de la carrera de Ingeniero en Computación. Por ejemplo, estos conocimientos podrán aplicarse para el desarrollo de: nóminas, cuentas por cobrar, inventarios, etc.

Por todo lo expuesto anteriormente, es que se ha incluído esta materia en el plan de estudios de la carrera de Ingeniero en Computación.

2. SUGERENCIAS PARA EL USO DE LA GUIA

Esta guía indica la secuencia a seguir en el estudio de los conceptos elementales de la materia y señala las actividades complementarias que el alumno deberá realizar en el desarrollo de cada tema. Para su mayor utilidad se sugiere:

1. Leer cuidadosamente, ubicando el tema y el objetivo correspondiente con la intención de comprender el enfoque y los alcances del tema.
2. Estudiar los contenidos en las referencias bibliográficas señaladas.
3. Analizar los ejemplos, resolver los problemas y ejercicios indicados en las referencias bibliográficas.
4. Acudir a la asesoría de esta asignatura cuando se requiera orientación o surjan dudas.

3. OBJETIVO GENERAL DE LA ASIGNATURA

Al finalizar el curso el alumno será capaz de analizar y diseñar sistemas de Programación, utilizando la metodología y herramientas vistas durante el curso.

4. TEMAS QUE COMPRENDE EL PROGRAMA

I. Evolución de los sistemas de programación.
II. Estudio general del sistema.
III. Planeación del sistema de programación.
IV. Análisis y especificación estructurada.
V. Diseño estructurado.
VI. Codificación y los lenguajes de programación.
VII. Documentación.
VIII. Pruebas y confiabilidad de los sistemas.
IX. Instalación y mantenimiento de los sistemas

5. REFERENCIAS BIBLIOGRAFICAS

BIBLIOGRAFIA BASICA

PRESSMAN S., ROGER: "SOFTWARE ENGINEERING: A practitioner's Approach", McGraw Hill New York, 1982.

DE MARCO, TOM: "SOFTWARE ANALYSIS AND SYSTEM SPECIFICATION", Prentice Hall Englewood Cliffs, N.J., 1979

YOURDON Y CONSTANTINE: "STRUCTURED DESIGN" Prentice Hall, Englewood Cliff N.J., 1979 QA76 6Y937 (*)

P. STEVENS, WAYNE: "USING STRUCTURED DESIGN" Wiley Interscience, New York 1981.

JENSEN & TONIES: "SOFTWARE ENGINEERING" Prentice Hall, Englewood Cliffs, N.J., 1979 QA76 .6 J44 (*)

BIBLIOGRAFIA COMPLEMENTARIA

R. DURELL, WILLIAM: "DATA ADMINISTRATION" Mc Graw Hill, Ney York, 1985.

GEREZ, MIER, NIEVA y RDZ: "DESARROLLO Y ADMINISTRACION DE PROGRAMAS DE COMPUTACION". IIE y CECSA, México, D.F., 1984.

T. WARD, PHUL: "SYSTEMS DEVELOPMENT WITHOUT PAIN", Yourdon Press, New York 1984.

PAGE-JONES, MEILIE: "THE PRACTICAL GUIDE TO STRUCTURED SYSTEMS DESIGN" New York, 1980.

FREEMAN, HERBERT y M. LEWIS II, PHILLIP: "SOFTWARE ENGINEERING" Academic Press, New York, 1980.

FAIRLEY, RICHARD: "SOFTWARE ENGINEERING CONCEPTS" New York, 1985.

DEUSTH S. MICHAEL: "SOFTWARE VERIFICATION AND VALIDATION", Prenticce Hall, Englewood Cliffs, N. J., 1982.

W. BAILEY, ROBERT: "HUMAN PERFORMANCE ENGINEERING: A guide for System Designers", Prentice Hall, Englewood Cliffs, N. J. 1982.

YOURDON, EDWARD: "STRUCTURED WALKTHROUGHS" Prentice Hall, Englewood Cliffs N. J., 1979 QA76 .6Y67 (*)

W. BOHEN, BARRY: "SOFTWARE ENGINEERING ECONOMICS", Prentice Hall, Engle-- wood Cliffs, N. J., 1981.

GANE, CHRIS Y SARSON, TRISH: "STRUCTURED SYSTEM ANALYSIS: Tools and techniques". Prentice Hall, Englewood Cliffs, N. J. 1979. QA76 634 (*).

(*) Ubicación de los textos en la Biblioteca "Antonio Dovalí Jaime" del Edificio Principal de la Facultad de Ingeniería.

6. DESARROLLO

A continuación, se tratará cada uno de los temas que componen el programa, donde se señala: el objetivo, el contenido, las actividades y los problemas propuestos, que deberán ser analizados y resueltos por los alumnos para afirmar y ejercitar los conocimientos adquiridos.

En la última parte de cada tema, se incluye un cuestionario de autoevaluación, cuyas respuestas se ubican al final de la guía.

TEMA I. EVOLUCION DE LOS SISTEMAS DE PROGRAMACION

PRESENTACION

En esta unidad se estudia la problemática que plantea la complejidad de los sistemas de programación, así como la metodología y herramientas que controlan dicha complejidad.

OBJETIVOS

Al término de esta unidad el alumno será capaz de identificar la secuencia lógica de pasos en el desarrollo sistemático de sistemas de programación. Identificar las herramientas pertinentes en cada paso del punto anterior. Identificar los criterios más comunes que cumplen los programas de calidad.

CONTENIDO

I.1 La crisis de los sistemas de Programación.
I.2 El ciclo de vida de los sistemas de Programación.

ACTIVIDADES

1. Estudiar el capítulo I (págs. 1-2-3) y el capítulo 3 (págs. 64-99) del libro SOFTWARE ENGINEERING de Randal N. Jensen y Charles Tonies, Prentice Hall (1979).
2. Estudiar el capítulo 2 (págs. 22-30) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman McGraw Hill (1982).
3. Estudiar los puntos 2.1 y 2.2 del capítulo 2 (págs. 19-25) del libro STRUCTURED ANALYSIS AND SYSTEM SPECIFICATION Tom de Marco, Prentice Hall (1979).

Problemas propuestos

1. Resolver los problemas 1-1, 1-5 y 1-6 del capítulo I (pag. 21) del libro SOFTWARE ENGINEERING: A Practitioner's Approach, de Roger S. -- Pressman McGraw Hill (1982).

CUESTIONARIO DE AUTOEVALUACION

1. ¿Qué originó la crisis de los sistemas de programación?
2. ¿Cuáles son las fases del ciclo de vida de los sistemas de programación?
3. ¿Cuáles son las etapas de la fase de planeación?
4. ¿Cuáles son las etapas de la fase de desarrollo?

TEMA II. ESTUDIO GENERAL DEL SISTEMA

PRESENTACION

En esta unidad se estudian las diversas categorías de sistemas. La derivación de los objetivos del sistema así como -- los documentos: Estudio de factibilidad y especificación del -- sistema.

OBJETIVOS

Al término de esta unidad el alumno será capaz de clasificar los diversos sistemas y de establecer los objetivos de un - Sistema de Programación así como estructurar los documentos de estudio de factibilidad y especificación del sistema.

CONTENIDO

II.1 Definición del Sistema.
II.2 Diagnóstico de la situación actual.
II.3 Análisis de Factibilidad.
II.4 Análisis del Sistema.

ACTIVIDADES

1. Estudiar el capítulo III (págs. 26-42) y el capítulo VII -- (págs. 101-120) del libro SYSTEMATIC SYSTEMS APPROACH de -- Thomas H. Athey Prentice Hall (1982).
2. Estudiar el capítulo III (págs. 31-57) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman Mc Graw Hill (1982).

Problemas Propuestos

1. Resolver los problemas 4 y 6 del capítulo III (pag. 41) del libro

SYSTEMS APPROACH de Thomas H. Athey Prentice Hall. (1982)

2. Resolver los problemas 3-2, 3-3, 3-4, 3-6 (pag. 56) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman Mc Graw Hill (1982).

CUESTIONARIO DE AUTOEVALUACION

1. ¿Qué es un sistema?
2. ¿En qué áreas de interés principal se centra un estudio de factibilidad?
3. ¿Qué consideraciones están normalmente asociadas con la fac tibilidad de recursos?

TEMA III. PLANEACION DEL SISTEMA DE PROGRAMACION

PRESENTACION

En esta unidad se estudia el alcance del trabajo ha hacerse para el desarrollo de un sistema, así como los recursos requeridos, el esfuerzo y costo necesario y finalmente el tiempo de desarrollo.

OBJETIVOS

Al término de esta unidad el alumno será capaz de describir el alcance del Software en términos cuantitativos. Estimar el esfuerzo, costo y tiempo de desarrollo, así como recursos necesarios.

CONTENIDO

III.1 El alcance del Sistema de Programación.
III.2 Recursos
III.3 Estimación de costos
III.4 Herramientas de control de avance.

ACTIVIDADES

1. Estudiar el capítulo 4 (págs. 58-93) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman McGraw Hill. (1982).

Problemas propuestos

1. Resolver los problemas 4-1, 4-2, 4-14 y 4-17 del capítulo IV (págs. 91-92) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Mc Graw Hill (1982).

CUESTIONARIO DE AUTOEVALUACION

1. Durante la planeación del Software ¿Qué recursos se consideran?

2. Mencionar 3 modelos de estimación de costos.

3. ¿Qué reglas debe considerar el planeador cuando Software reusable se especifica como recurso?

TEMA IV. ANALISIS Y ESPECIFICACION ESTRUCTURADA

PRESENTACION

En esta unidad se estudian las herramientas del análisis estructurado, así como el documento: Especificación de los requerimientos del Software.

OBJETIVOS

Al término de esta unidad el alumno será capaz de utilizar el diagrama de flujo de datos (DFD) como una herramienta de modelación para el sistema. Utilizar el diccionario de datos para detallar las interfaces entre los elementos del DFD. Utilizar los árboles y tablas de decisión para esclarecer lógica compleja, utilizar seudocódigos (español estructurado) como una herramienta para especificar la lógica de cada elemento del DFD.

CONTENIDO

IV.1 El diagrama de flujo de datos (DFD)
IV.2 Características del DFD
IV.3 El diccionario de datos
IV.4 Miniespecificaciones
IV.5 Arboles y tablas de decisión
IV.6 Español estructurado

ACTIVIDADES

1. Estudiar el capítulo V (págs. 94-127) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman McGraw Hill (1982).
2. Estudiar los capítulos V, VI t VII (págs. 51-88) y los capítulos XII, XIII y XIV (págs. 129-165) y el capítulo XVI (págs. 179-213) del libro SOFTWARE ANALYSIS AND SYSTEM SPECIFICATION de Tom de Marco Prentice Hall (1979).

3. Estudiar los capítulos II y III (págs. 13-60) del libro AN INTRODUCTION TO DECISION LOGIC TABLES de Herman McDaniel Petrocelli (1978).
4. Estudiar el capítulo V (págs. 77-109) del libro STRUCTURED SYSTEMS ANALYSIS: Tolls and techniques de Chris Gane y Trish Sarson. Prentice Hal (1979).

Problemas propuestos

1. Resolver los problemas 5-4 y 5-5 (pag. 125) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman McGraw Hill (1982).
2. Resolver todos los ejercicios del apéndice A (págs. 101-133) del libro AN INTRODUCTION TO DECISION LOGIC TABLES de Herman McDaniel-Petrocelli (1978).
3. Resolver los problemas 1, 2 y 3 (pag. 108) del libro STRUCTURED SYSTEMS ANALYSIS: Tolls and techniques de Chris Gane y Trish Sarson Prentice Hall (1979).

CUESTIONARIO DE AUTOEVALUACION

1. ¿Cuáles son los elementos de un diagrama de flujo de datos
2. ¿Cuál es la notación utilizada en el diccionario de datos?
3. ¿Qué es una miniespecificación?
4. ¿Cuántos tipos de tablas de decisión hay?
5. ¿Qué es un árbol de decisión?
6. ¿Cuáles son las estructuras básicas del español estructurado (pseudocódigo)?

TEMA V. DISEÑO ESTRUCTURADO

PRESENTACION

En esta unidad se estudian las herramientas del diseño estructurado, los conceptos de cohesión y acoplamiento y el documento de especificación del diseño de Software.

OBJETIVO

Al término de esta unidad el alumno será capaz de utilizar la carta de estructura y los conceptos de cohesión y acoplamiento para evaluar su diseño.

CONTENIDO

V.1 La carta de estructura.
V.2 Características de la carta de estructura.
V.3 Estructuras típicas.
V.4 Modularidad.
V.5 Cohesión.
V.6 Acoplamiento.

ACTIVIDADES

1. Estudiar los capítulos III, IV, V y VI (págs. 11-59) y los capítulos VIII, IX y X (págs. 120-187) del libro USING STRUCTURED DESIGN de Wayne P. Stevens Wiley Interscience (1981).
2. Estudiar los capítulos 6 y 7 (págs. 84-141) y los capítulos 10 y 11 (págs. 165-222) del libro STRUCTURED DESIGN de Yourdon y Constantine Prentice Hall (1979).

Problemas propuestos

1. Derivar la carta de estructura de un sistema conocido por usted y evaluarla usando los conceptos de cohesión y acoplamiento.

CUESTIONARIO DE AUTOEVALUACION

1. ¿Cuáles son los elementos de la carta de estructura?
2. Dar una definición de modularidad.
3. ¿Qué es cohesión?
4. ¿Cuántos tipos de cohesión hay? Explicar.
5. ¿Qué es acoplamiento?
6. ¿Cuántos tipos de acoplamiento hay? Explicar.

TEMA VI. CODIFICACION Y LOS LENGUAJES DE PROGRAMACION

PRESENTACION

En esta unidad se estudian las herramientas de la programación estructurada, así como las características de lenguajes de programación.

OBJETIVOS

Al término de esta unidad el alumno será capaz de utilizar el pseudocódigo, así como los diagramas estructurados en la especificación de un programa.

CONTENIDO

VI.1 La programación sistemática.
VI.2 Las herramientas de programación.
VI.3 Clases y características de los lenguajes de programación.
VI.4 Herramientas de puesta a punto.

ACTIVIDADES

1. Estudiar el capítulo 7 (págs. 306-341) del libro AN INTRODUCTION TO COMPUTER SCIENCE: An algorithmic Approach de Tremblay y Bunt McGraw Hill (1981).
2. Estudiar el capítulo 4 (págs. 221-328) del libro SOFTWARE ENGINEERING de Jensen y Tonies Prentice Hall (1979).

Problemas propuestos

1. Plantear 3 problemas de 'mediana complejidad' y resolverlos mediante el proceso de refinamiento a pasos.

CUESTIONARIO DE AUTOEVALUACION

1. Explicar el proceso de refinamiento a pasos.
2. ¿Cuántas clases de lenguaje de alto nivel existen? Dar ejemplos de cada uno de ellos.
3. ¿Qué lenguajes se considera actualmente que son los mejor diseñados?
4. ¿Cuáles son los diagramas de flujo estructurados correspondientes al pseudocódigo del problema 6 del capítulo IV?

TEMA VII. DOCUMENTACION

En esta unidad se estudian los manuales del usuario y el de operación.

OBJETIVOS

Al término de esta unidad el alumno será capaz de documentar el manual del usuario y el de operación para un sistema de programación.

CONTENIDO

VII.1 Manual del usuario.
VII.2 Manual de operación.

ACTIVIDADES

Hay poca literatura en la actualidad sobre normas para documentar un sistema de programación, se recomienda al alumno consultar los manuales del usuario y de operación de un sistema de programación.

Problemas propuestos

Hacer el manual del usuario y de operación de un sistema pequeño.

CUESTIONARIO DE AUTOEVALUACION

1. ¿Qué es documentación interna?
2. ¿Qué es documentación externa?
3. Mostrar un diagrama jerárquico de un sistema cualquiera.
4. ¿Qué es el manual de operación y usuario?

TEMA VIII. PRUEBAS Y CONFIABILIDAD DE LOS SISTEMAS

PRESENTACION

En esta unidad se estudian 3 tópicos relacionados: pruebas, depuración y confiabilidad.

OBJETIVO

Al término de esta unidad el alumno será capaz de diseñar pruebas de un sistema y así como hacer una estimación de la confiabilidad del mismo.

CONTENIDO

VIII.1 Características de la prueba.
VIII.2 Pasos en las pruebas de los sistemas de programación.
VIII.3 Generadores de datos prueba.
VIII.4 Pruebas de unidades y pruebas de integración.
VIII.5 Prueba de validación.
VIII.6 Prueba de volumen.
VIII.7 Simulación del sistema.

ACTIVIDADES

1. Estudiar el capítulo 12 (págs. 289-321) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman McGraw Hill (1982).

Problemas propuestos:

1. Resolver los problemas 12-6, 12-8, 12-9, 12-11, 12-12, 12-13 y 12-19 (págs. 320-321) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. Pressman Mc Graw Hill (1982).

CUESTIONARIO DE AUTOEVALUACION

1. ¿Cuáles son las reglas de Glen Myres que pueden servir como objetivos de las pruebas?
2. ¿Cuáles son los pasos en una prueba?
3. ¿Qué características de un módulo se evalúan durante - una prueba de unidad?
4. ¿Qué categorías de enfoques de depuración se han pro-- puesto?
5. ¿En qué categorías caen los modelos de confiabilidad?

TEMA IX. INSTALACION Y MANTENIMIENTO DE LOS SISTEMAS

PRESENTACION

En esta unidad se estudia la problemática de la última fase del ciclo de vida de los sistemas de programación a saber el de la instalación y mantenimiento.

OBJETIVOS

Al término de esta unidad el alumno será capaz de identificar los problemas más comunes de la instalación y mantenimiento de los sistemas de programación.

CONTENIDO

IX.1 El plan de instalación.
IX.2 La capacitación.
IX.3 La carga de archivos.
IX.4 Aprobación final
IX.5 Identificación de resultados y desviaciones.

ACTIVIDADES

1. Estudiar el capítulo 13 (págs. 322-341) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de To-- ger S. Pressman McGraw Hill (1982).

Problemas propuestos

1. Resolver los problemas 13-3, 13-4, 13-5 y 13-6 (pag. 380) del libro SOFTWARE ENGINEERING: A Practitioner's Approach de Roger S. - Pressman .McGraw Hill (1982).

CUESTIONARIO DE AUTOEVALUACION

1. Dar una expresión que indique el esfuerzo de mantenimiento.

2. ¿Cuáles son las medidas cualitativas del mantenimiento que relacionan el esfuerzo dedicado durante el mantenimiento?

3. ¿Cuáles son los tipos de mantenimiento?

4. ¿En porcentaje, cómo se distribuyen los tipos de mantenimiento?

7. SOLUCIONES A LOS CUESTIONARIOS DE AUTOEVALUACION

Tema I

1. Está asociada con problemas sobre cómo desarrollar sistemas de programación, cómo dar mantenimiento al volumen creciente de software existente y de cómo mantener el paso con una demanda creciente de más software.

2. a) Fase de planeación
   b) Fase de desarrollo
   c) Fase de mantenimiento

3. a) Definición del sistema
   b) Planeación del software
   c) Análisis y definición de los requerimientos del software

4. a) Diseño preliminar
   b) Diseño detallado

Tema II

1. Es una colección de elementos relacionados de modo tal que permiten el logro de un objetivo tangible.

2. a) Factibilidad económica
   b) Factibilidad técnica
   c) Factibilidad legal
   d) alternativas

3. a) Riesgos de desarrollo
   b) Disponibilidad de recursos
   c) Tecnología

Tema III

1. a) Humanos
   b) Hardware
   c) Software

2. a) El modelo cocomo
   b) El modelo de estimación de putnam
   c) Los modelos de punto de función

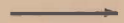
3. a) Adquirir software existente si este cumple con los requerimientos. El costo de adquirirlo por regla general será menor que desarrollarlo.
   b) Adquirir software que va a modificarse siempre y cuando el costo de las modificaciones sea menor que el costo de desarrollarlo.

Tema IV

1. Entidad externa.- Una fuente de entrada al sistema o un receptor de salidas del sistema. Su símbolo es:

Proceso.- Realiza alguna transformación con sus datos de entrada para dar datos de salida. Su símbolo es:

Flujo de datos.- Usado para conectar entre sí procesos y entidades externas o procesos y archivos. La flecha indica la dirección de la transferencia de los datos. Su símbolo es:



Archivos.- Es un contenedor de datos. Las flechas indican entradas y salidas netas al archivo. Su símbolo es:

2.

NOTACION	SIGNIFICADO
=	ESTA COMPUESTO DE
+	Y
[|]	ó – ó
$\{\}^n$	n REPETICIONES DE
( )	DATO OPCIONAL

3. Es un enunciado (descripción de un proceso) de la política que gobierna la transformación del flujo de datos entrante en flujo de datos saliente.

4. a) Tabla de entrada limitada.
   b) Tabla de entrada extendida.
   c) Tabla de entrada mixta.

5. Es una ayuda gráfica que permite aclarar la complejidad de los condicionales.

6. a) Si (condición - A) luego
   instruccion(es) -A
   o si (condición - B) luego
   instruccion(es) - B

   o bien
   instruccion(es) - X
   fin

   b) si(condición) luego
   instruccion(es) - A
   o bien
   instruccion(es) - B
   fin

   c) si(condición) luego
   instruccion(es)
   fin

   d) seleccionar (caso) de
   (Caso - A): instruccion(es) - A
   (Caso - B): instruccion(es) - B

   (o bien): instruccion(es) - X
   fin

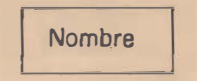
   e) Desde (i expa A hasta exp - B, paso n)
   repetir instruccion(es)
   fin

   f) Repetir
   instruccion(es)
   Hasta (condición)

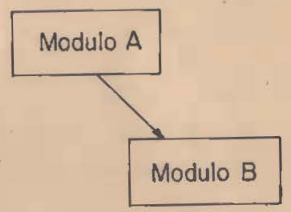
   g) Mientras (condición) repetir
   instrucciones
   fin

Tema V.

1. a) Módulo .- Secuancia de instrucciones con un propósito bien definido (suma de matrices, etc.) a las cuales se hace referencia,- como un todo, mediante un nombre, su símbolo es:

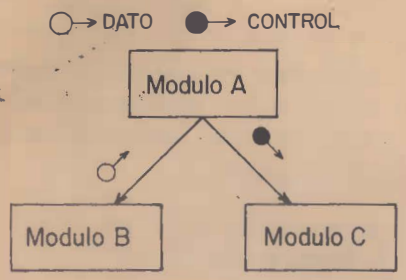


b) Conecciones.- Representan la subordinación o invocación de un módulo por otro. - Por ejemplo el módulo A invoca al módulo B



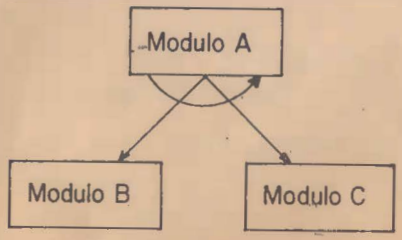
c) Flujo de información.- Notación adjunta a las conecciones entre módulos que indica la dirección del flujo de información (datos y/o control). Su símbolo es:

EJEMPLO:

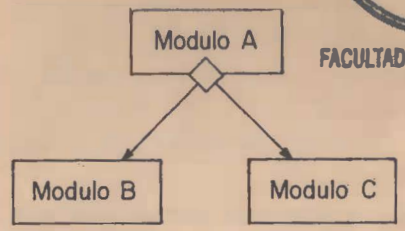


d) Procedimiento.- Notación que indica que un módulo ha sido invocado dentro del alcance de una estructura de control que se encuentra en un módulo superior.

I) Iteración



II) Decisión



2. Significa dividir en pequeñas piezas, bien definidas, un problema, con el objeto de reducir su complejidad y por ende hacerlo manejable intelectualmente.
3. Es el grado de afinidad funcional de los elementos en un módulo.
4. Se listan de acuerdo a un grado de cohesión. Se va de mayor a menor cohesión.
   a) Funcional.- Cuando cada elemento del módulo es una parte integral de, y es esencial a, la realización de una sola función.
   b) Secuencial.- Cuando la salida de un elemento sirve como entrada al siguiente elemento en el módulo.
   c) Comunicación.- Cuando todos los elementos de un módulo operan bajo el mismo conjunto de datos de entrada y/o producen los mismos datos de salida.
   d) Procedimiento.- Cuando la ocurrencia de los elementos sucede durante el mismo período de tiempo limitado durante la ejecución del sistema.
   f) Lógica.- Cuando los elementos se consideran miembros de la misma clase lógica de funciones similares o relacionadas.

g) Coincidencia.- Cuando los elementos de un módulo no tienen una relación constructiva uno al otro.

5. Es una medida de la intensidad de interconexión entre un módulo y otro.

6. Se listan de acuerdo a su grado de acoplamiento. Se va de menor a mayor acoplamiento.

a) Sin acoplamiento - ocurre cuando dos módulos no se comunican.

b) Datos - ocurre cuando se pasan datos simples de - un módulo a otro.

c) Estampilla - ocurre cuando una porción de una estructura de datos se pasa a través de una interfase de módulos.

d) Control - ocurre cuando 2 módulos se comunican -- información de control.

e) Externo - ocurre cuando los módulos están ligados a un medio externo al software.

f) Común - ocurre cuando 2 o más módulos actuan con un ambiente de datos común.

g) Contenido - ocurre cuando parte o todo el contenido de un módulo está incluído en el contenido de otro.

Tema VI

1. Este proceso consiste en subdividir un problema en sus partes elementales, luego cada una de estas partes se considera a su vez para una posible mayor subdivisión. Este proceso se continua hasta que cada parte derivada se codifique directamente en el lenguaje de programación elegido.

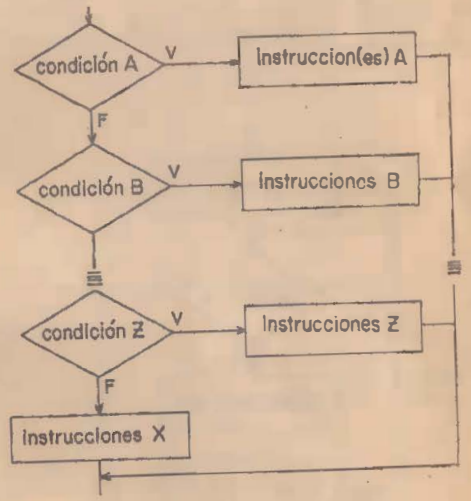
2. Tercera generación
   Cobol
   Fortral
   Pascal

Cuarta generación
   Focus
   Nomad
   Mapper

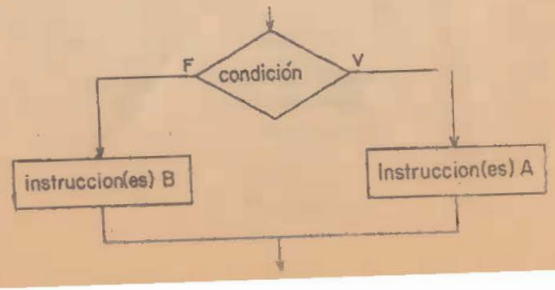
Quinta generación
   Prolog

3. Modula - 2
   Ada
   Smalltalk - 80

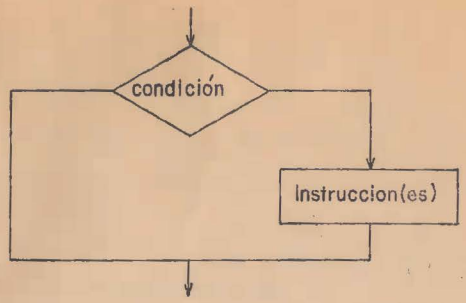
4. a)



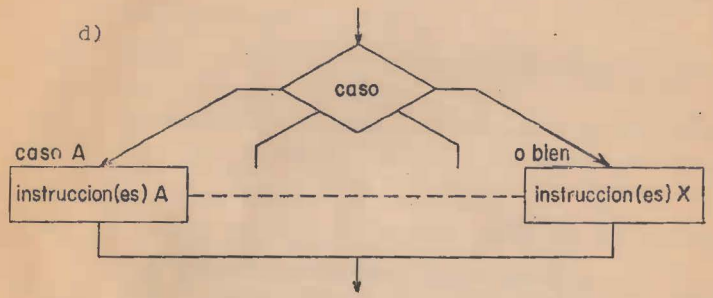
b)



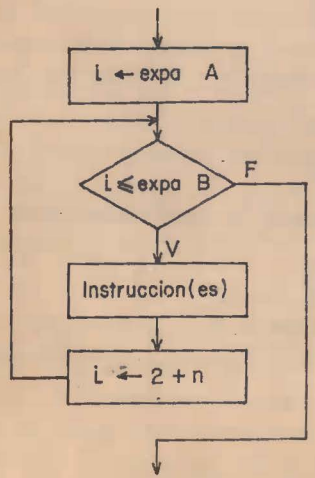
c)



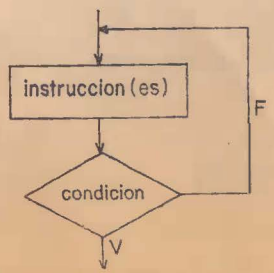
d)



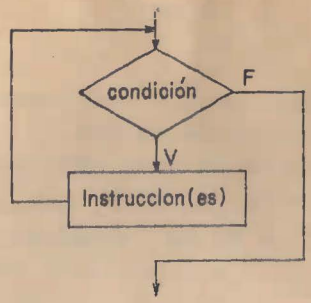
e)



f)

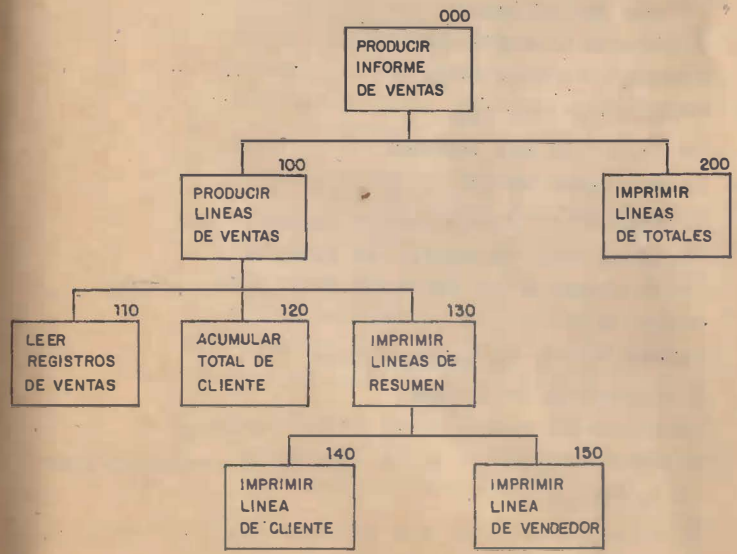


g)



Tema VII.

1. Son los comentarios inmersos en los programas.
2. Puede constar de diagramas jerárquicos, diagramas hipo, etc.
3.



4. El manual de usuario dice cómo hacer uso del sistema (usualmente a través de menús).
El manual de operación dice cómo opera el sistema: funciones, estructuras de datos, interfaces internas y externas y mecánica de las funciones.

Tema VIII.

1. a) La prueba es el proceso de ejecutar un programa con la intención de encontrar un error.
   b) Un buen caso de prueba es aquel que tiene una alta probabilidad de encontrar un error aún no descubierto.
   c) Una prueba exitosa es aquella que descubre un error aún no descubierto.
2. a) Prueba de unidad.
   b) Prueba de integración.
   c) Prueba de validación.
   d) Prueba del sistema.
3. a) La interfaz del módulo.
   b) Estructuras de datos locales
   c) Trayectorias de ejecución importantes.
   d) Trayectorias de manejo de errores.
   e) Condiciones a la frontera que afecta a todo lo anterior.
4. a) Fuerza bruta (a sentimiento)
   b) Eliminación de causas.
   c) Seguimiento hacia atrás (backtracking)
5. a) Modelos derivados de la teoría de confiabilidad de hardware
   b) Modelos basados en las características internas del programa.
   c) Modelos desarrollados al 'sembrar' errores conocidos al software y evaluar el número de errores sembrados detectados contra errores reales detectados.

Tema IX.

1. $M = p + k \exp (c - d)$

   M = esfuerzo total dedicado al mantenimiento.
   p = esfuerzo productivo
   k = una constante empírica
   c = una medida de complejidad que puede atribuirse a la pérdida de diseño estructurado y a la documentación
   d = una medida del grado de familiaridad con el software
2. 1. tiempo debido al reconocimiento del problema
   2. tiempo debido a retraso administrativo
   3. tiempo debido a la colección de herramientas de mantenimiento
   4. tiempo debido al análisis del problema
   5. tiempo debido a la especificación del cambio
   6. tiempo debido a modificación o corrección activo
   7. tiempo debido a pruebas locales
   8. tiempo debido a prueba global
   9. tiempo debido a revisión de mantenimiento
   10. tiempo debido a recuperación total
3. a) mantenimiento adaptivo
   actividad que modifica la interfaz del software debido a cambios en el medioambiente
   b) mantenimiento perfectivo
   actividad que toma en cuenta la mayoría del esfuerzo dedicado al mantenimiento del software
   c) mantenimiento preventivo
   actividad que modifica el software para mejorar mantenimiento o confiabilidad futura o para proporcionar una mejor base para realces futuros.

4.

Impreso por la
Coordinación de Servicios Generales
a través de la Unidad de Difusión,
Departamento de Impresión.
El tiraje consta de 500 ejemplares
y se terminó de imprimir
en el mes de marzo de 1990.