UNIVERSIDAD SIMÓN BOLÍVAR
Vicerrectorado Académico

1 .Departamento: *COMPUTACIÓN Y TECNOLOGÍA DE LA INFORMACIÓN*

2. Asignatura: Programa de Algoritmos y Estructuras I

3. Código de la asignatura: CI-2611

No. de unidades-crédito: 3

No. de horas semanales: Teoría 3 Práctica 1 Laboratorio 3

4. Fecha de entrada en vigencia de este programa: A partir de Enero 2005

5. OBJETIVO GENERAL:

Introducción a la resolución sistemática de problemas mediante algoritmos y a los principios básicos de la Programación.

6. OBJETIVOS ESPECÍFICOS:

Al finalizar el curso el estudiante deberá conocer nociones básicas sobre:

1) Especificación formal de problemas mediante precondición y postcondición, utilizando el cálculo de predicados de primer orden.
2) Control de Complejidad en la resolución algorítmica de problemas. Utilicización del Diseño Decente y técnicas básicas de Diseño de Algoritmos. Refinamiento de Datos y de Programas.
3) Metodología para la derivación y prueba de la Corrección de programas.
4) Esquemas de Algoritmos Iterativos. Diseños y construcción y Programas Iterativos.
5) Normas básicas de Estilo de Programación.
6) El estudiante deberá ser capaz de leer y entender especificaciones de programas. Entender la metodología que debe seguirse para desarrollar programas correctos.

Durante todo el curso ejercitarán las nociones: Especificación de problemas, Diseños decentes de programas y derivación y corrección de programas.

Se hace hincapié en refinanciamentos de programas, aunque sé da una noción básica de refinamiento de datos (tema principal del curso CI-2616). La noción de tipo abstracto de dato (verlo como clase) y objeto deberá ser dada desde un comienzo, aún cuando solo trabajaremos fundanentalmente los tipos básicos, tipo arreglo y tipo archivo secuencial. Se dará la noción de encapsulamiento y ocultamiento de datos en lenguajes de programación para introducir la noción de clase (implementación de tipos abstractos). Las nociones sobre derivación y corrección de programas serán tratadas más formalmente al final del curso. En las primeras semanas del curso se desarrollarán los programas de forma más intuitiva. Se deberán dar varias soluciones algorítmicas a un mismo problema, mostrando las bondades de unas respecto a las otras y resaltando el buen estilo de la programación.

Es recomendable que los estudiantes ya hayan visto, o sigan en paralelo, un curso de lógica y teoría de conjuntos con el fin de establecer la notación matemática a utilizar.

En la primera parte del curso se determinarán especificaciones de problemas y se desarrollarán los algoritmos de manera intuitiva a partir de estas especificaciones, haciendo énfasis en esquemas de algoritmos y diseño descendente. Dejando en claro cada momento las aserciones (predicados) que representan los cambios de estados de cada acción. La noción de métodos (procedimientos y funciones) se presenta temprano en el curso, de manera de tener lo mas temprano posible este mecanismo de control de la complejidad, y permita al equipo del laboratorio elaborar proyectos donde desde un comienzo se haga uso de este mecanismo de estructuración de programas.

En cuanto a los tipos de datos, estos serán los tipos primitivos de lenguaje JAVA, el tipo archivo secuencial y el tipo arreglo. En cuanto a constructores, se darán en el siguiente orden: la asignación, la secuencia de acciones, Procedimientos y Funciones (como mecanismo de abstracción procedural en la aplicación del diseño decente, paso de parámetros: parámetros de entrada, salida y entrada-salida. Variables locales. Llamadas recursivas, al final del curso), constructores condicionales, constructores interactivos. En la primera parte del curso se enfatizará más el buen estilo de programación, en la segunda parte se puede incluir criterios de eficiencia se trata en el curso CI-2616.

En las últimas semanas del curso se hará énfasis en derivación formal de algoritmos interactivos. Se desarrollaran algoritmos utilizando técnicas de derivación. La razón de dar derivación es para mostrar que existe una forma sistemática para desarrollar e incluso calcular un programa correcto a partir de una especificación.

Al final del curso sé dará una breve introducción al diseño de soluciones recursivas de problemas (utilizando para esto una variedad de ejemplos concretos). Técnica Divide y Conquistarás. Las soluciones recursivas, producto de aplicar la técnica Divide y Conquistaras, deberán ser mostradas de manera natural (el tratamiento formal de corrección y/o derivación de programas recursivos no es materia de este curso). En la medida posible, se dará una solución recursiva de ejemplos mostrados a lo largo del curso.

Para desarrollar los algoritmos en teoría, se utilizará el pseudo lenguaje del libroKaldewaij. En laboratorio se utilizará el lenguaje JAVA.

En la primera parte del curso se determinarán especificaciones de problemas y se desarrollarán los algoritmos de manera intuitiva a partir de estas especificaciones, haciendo énfasis en esquemas de algoritmos y diseño descendente. Dejando en claro cada momento las aserciones (predicados) que representan los cambios de estados de cada acción. La noción de métodos (procedimientos y funciones) se presenta temprano en el curso, de manera de tener lo mas temprano posible este mecanismo de control de la complejidad, y permita al equipo del laboratorio elaborar proyectos donde desde un comienzo se haga uso de este mecanismo de estructuración de programas.

En las primeras semanas del curso se dará gradualmente el pseudo lenguaje que se utilizara para describir los algoritmos: las estructuras básicas de control y los tipos de datos necesarios para resolver problemas cada vez más complejos. También se precisarán la regla de alcance de identificadores y paso de parámetros (distinguiendo parámetros de entrada, salida y entrada-salida).

7. PROGRAMA DESSARROLLADO DE TEORIA:

1) Lógica y Teoría de conjuntos (2 clases. Bibliografía: Guía y Gries):

Breve introducción del cálculo de predicados y de la teoría de conjuntos y relaciones.
Establecer la terminología y notación a utilizar a lo largo del curso.

2) Problemas y su solución mediante algoritmos (2 clases. Bibliografía: Guía):

¿Qué es un problema?. Enunciado de un problema. Especificaciones informales: inconsistencia, ambigüedad, redundancia. Especificaciones Formales. Pasos a seguir par especificar un problema (Precondición y Postcondición: determinación de los objetos de dato y de resultado y representar las relaciones entre estos utilizando las operaciones sobre los objetos). Ejemplo de especificación: Dado un número natural que representa una cantidad de segundos, se desea calcular la representación de estos números en días, horas, minutos y segundos.

Pasos a seguir en la resolución de un problema mediante algoritmos (enunciado, especificación y proceso de abstracción, diseño, construcción del programa, tipos de pruebas, derivación). Concepto de: Proceso/acción (manipulación de objetos), Objetos (identificador, tipo, estado o valor), Tipos Abstractos de Datos o Clases (sólo dar el concepto e ilustrar con ejemplos, como por ejemplo el Tipo Número Complejo). El concepto de tipo abstracto de dato como mecanismo de estructuración de la información que manipula un algoritmo a un determinado nivel de abstracción (los datos compuestos los veremos como objetos, un tipo abstracto se corresponde con la noción de clase). El Concepto de Algoritmo. De la especificación al algoritmo, Diseño decente para construir soluciones algorítmicas de problemas: modelos de maquinas abstractas. Ejemplo para ilustrar en que consiste resolver un problema mediante un algoritmo (la descripción se hace en lenguaje natural). Ejemplo: Se tiene un cesto con papas, se desea pelar un número “suficiente” de papas teniendo en cuenta que el cesto puede vaciarse y las papas pueden reponerse (se hacen varias versiones para ilustrar los conceptos de acción, objetos, diagramas de estados, versiones que resuelven completa o parcialmente el problema). Aserciones para describir estados.

El concepto de máquina, algoritmo versus programas. Lenguajes de Programación, Compiladores, Interpretadores: solo decir que los lenguajes constituyen máquinas abstractas, que es necesario traducir (por compiladores o interpretadores) los programas en lenguaje de alto nivel a programas en lenguaje de máquinas. Ejemplos: describir mediante un algoritmo el lenguaje natural, el proceso que resulta de hacer la división entera de dos números naturales. Noción de máquinas abstractas. Ejemplo: describir el algoritmo de multiplicación de dos enteros en término de las operaciones de suma y resta.

1. Construcción de Programas (12 clases).

3.1. El pseudo lenguaje (2 clases, Bibliografía: guía, Kaldewaij):

3.2. Tipos básicos y sus operaciones, variables. Expresiones. Acciones booleanos, caracteres. Declaración de variables. Expresiones. Acciones elementales: observación, modificación (la asignación). Semántica de la asignación. Semántica de la secuencia de acciones mediante aserciones (Sí {Q}11{P} y {P}12{R} entonces {Q}11; 12{R}). Especificación, desarrollo intuitivo y verificación de algoritmos utilizando el pseudo lenguaje. Ejemplos sencillos sobre secuencia de acciones: suma de dos números, intercambio de valores de dos variables. Otros ejemplos: un número natural que representa una cantidad de segundos, se desea calcularla representación de este número en días, horas, minutos y segundos.

Acciones parametrizadas: Procedimientos y funciones como mecanismo de abstracción de acciones sobre objetos y control de la complejidad. Paso de Parámetros. Utilización del diseño decente. Ejemplo: Máquina de trazados. Dibujar un cuadro con la máquina de trazados (en este ejemplo se ilustran los conceptos de diseño descendentes, acciones parametrizadas). Dibujo de los cuadros encajados. Ejemplo de síntesis: hacer la suma de dos "duraciones" dadas en días, horas, minutos y segundos.

Análisis de caos: Introducción de la acción compuesta (bloque), acción condicional y alternativa. Semántica de las acciones condicionales en términos de pre y post condición. Acciones alternativas válidas, condicionales anidados. Motivación. Ejemplo: derivar un algoritmo que determine el valor absoluto de un número. Ejemplo: Dados tres números ordenarlos de menor a mayor. Ejemplo: hallar el máximo de tres valores (dar solución utilizando condicional, y solución utilizando una función que devuelve el máximo entre dos números). Ejemplo: hallar las raíces reales de un polinomio de segundo grado. Uso de tablas de decisión para el análisis de casos.

3.3. Procesos iterativos. Arreglos. Esquemas de algoritmos (6 clases. Bibliografía: guía, Kaldewaij, Castro): Proceso iterativo. Análisis de procesos iterativos. Elementos de análisis iterativo: estado inicial, estados intermedios, estado final. Noción de variante de un proceso iterativo. Condición de continuación de un proceso iterativo, acción elemental repetida, función de cota. El constructor "Mientras", su semántica en función de pre y post selección. Es muy importante resaltar que el invariante debe surgir de manera natural en el esquema de solución algorítmica del problema, y no ala inversa, es decir, hacer el algoritmo y luego, una vez construido, se determina los invariantes de las acciones iterativas. Heurística básica para determinar invariantes. Ejemplos (aplicar diseño descendente en lo posible): Cálculo de la suma de los n primeros términos de una sucesión, Máximo Común Divisor (solución iterativa, e introducir las soluciones recursivas de manera natural), Raíz entera de un número. Cálculo de potencia n-ésima de un número (ver distintas soluciones y la mas eficiente y sus planteamientos recursivos).

Tipo Arreglo: definición. Ver una instancia de arreglo como un objeto. Motivación de uso (permiten presentar secuencias y acceso directo a los elementos de una secuencia).Ejemplos: suma de los elementos de un arreglo, búsqueda binaria, clasificación respecto a un pivote (los elementos al pivote primero...).

3.4. Diseño descendente (4 clases. Bibliografía: guía, Castro):

Refinamiento de datos y refinamiento de acciones. Refinamiento de acciones: sección 7.2.1 de la guía. Ejemplo con la máquina de trazados: Dibujo de n círculos concéntricos. Refinamiento de datos: Invariante de Acoplamiento. Ejemplo: cálculo de las raíces complejas de un polinomio. Otro ejemplo: contar cuantas veces aparece la palabra "mu" en una secuencia de caracteres. Encapsulamiento y Ocultamiento de datos: el constructor tipo (ó clase). Ventajas. Ejemplo: cálculo de las raíces complejas de un polinomio utilizando el constructor de tipos.

4) Derivación formal de programas y soluciones recursivas de problemas (4 clases. Bibliografía: guía, Kaldewaij):

Derivación de programas iterativos (uso de las heurísticas). Invariante. Tratamiento formal de arreglos.

Soluciones recursivas de problemas. Varios ejemplos. Procedimiento y funciones recursivas.

Evaluación:

Hay que evitar sobrecarga al estudiante con asignaciones que se salgan de los objetivos del curso. En teoría se recomienda elaborar los exámenes parciales, uno a la mitad del trimestre y uno al final. En laboratorio se recomienda la evaluación continua semanal en el aula. Los porcentajes deberían estar entre 60-70% teoría, y 30-40% laboratorio.

7. BIBLIOGRAFIA

- Oscar Meza. Guía de algoritmos I. 2000
- Jorge Castro, Felipe Cucker, Xavier Messeguer, Albert Rubio, Luis Solano, Borja Valles. “Curso de Programación”. McGraw Hill. 1993. ISBN-84-481-1959-2. Capítulos 1, 2 y 3.
- Kaldewaij Anne. “Programing: the derivation of algorithms”. Prentic Hall. 1990. ISBN-0-13-204108-1. Capítulos 1, 2, 3 y 4.
- Gries David, Gries Paul. Pragramlive: Introducing Programming with JAVA”. Software.
- Gries David. “The Science of Programming”. Springer. Verlag. 1981. ISBN 0-387-96480-0. Cota: QA76.6 G747. Páginas 1-85 y 310-319.
- Hostmann Cay, Cornell Gary. “Core JAVA 2: Volumen I-Fundamentals”. Prentice Hall. 1999. Capítulos 1, 2 y 3.