ESTRUCTURAS DE DATOS Y ALGORITMOS

Juan Hernández Núñez
Fernando Sánchez Figueroa
Luis A. Álvarez Llorente
Jorge Quirós Rosado
e-mail: {juanher, fernando, luisalva, jquiros}@unex.es
Departamento de Informática. Escuela Politécnica. Universidad de Extremadura

RESUMEN

Estructuras de Datos y Algoritmos es una asignatura totalmente teórica complementada con la asignatura de Laboratorio de Programación II; se pretende que el alumno aprenda a manejar con soltura las estructuras de datos y estructuras de programación que se deben utilizar a la hora de desarrollar cualquier tipo de software, de manera eficiente y elegante; teniendo como base siempre la Programación Orientada a Objetos. Exponemos aquí una visión general de la asignatura con el temario impartido.

La asignatura corresponde al segundo curso de las tres titulaciones impartidas por la UEX: Ingeniería Técnica en Informática de Gestión (I.T.I.G), Ingeniería Técnica en Informática de Sistemas (I.T.I.S.) e Ingeniería en Informática (I.I.) y con carácter troncal para cada una de ellas. El número de alumnos matriculados en la asignatura es superior a 400, divididos en tres grupos correspondientes a las tres titulaciones. El número de créditos asignados es de 9, siendo el nº de horas impartidas de tres semanales durante todo el curso escolar.

El principal objetivo es el aprendizaje de las técnicas de desarrollo de software eficiente, utilizando siempre como base la programación orientada a objetos y usando los lenguajes PASCAL orientado a objetos y C++. El aprendizaje se lleva a cabo utilizando transparencias que los alumnos solicitan con anterioridad.

Se aprovechará el conocimiento de programación en PASCAL que los alumnos han adquirido en la asignatura de Elementos de Programación y Laboratorio de Programación.

Se pedirá para aprobar la asignatura que los alumnos superen un examen final, en el que se les plantean desarrollar tres algoritmos en PASCAL/C++ durante tres horas.

OBJETIVOS:

- Dotar al alumno de una sistemática de actuación ante el planteamiento de un determinando problema que se desee resolver.

- Consolidar los conocimientos adquiridos en el curso anterior mediante el diseño, implementación y manipulación de algoritmos y Estructuras de Datos.
- Introducir la abstracción como metodología general de la Ingeniería del Software.
- Capacitar para la elección apropiada de estructuras de almacenamiento de información e implementación de esas estructuras de forma eficiente.
- Introducir la Programación Orientada a Objetos como la técnica de programación de los 90 para el diseño e implementación de programas.
- Conocer distintos entornos de programación.

TEMARIO:

El temario de la asignatura está divido en cinco módulos como a continuación se detalla:

MODULO I.- METODOLOGÍA DE DESARROLLO DE PROGRAMAS.

Se trata de un módulo de introducción de la asignatura en el que se revisan las metodologías de programación más comunes propuestas durante la historia, metodologías ya estudiadas en el primer curso. Se impone la modularización y la programación basada en refinamientos sucesivos como base para el diseño de sistemas que sean legibles y fáciles de mantener y adaptarse a nuevos requerimientos impuestos en las aplicaciones.

Una vez se han recordado todos los aspectos básicos de la programación estudiados en el curso anterior, la asignatura se centra en la Abstracción como metodología de trabajo; se estudian la abstracción de datos y abstracción procedural para confluir en los Tipos Abstractos de Datos (TAD), estudiándose pormenorizadamente la notación de Liskov y la notación de Guttag, exponiendo ejercicios resueltos y proponiendo nuevos ejercicios a los alumnos. A partir de este tema, toda nueva estructura de datos será considerada como un TAD, empleándose la notación algebraica de Guttag para la definición de sus propiedades básicas.

El módulo consta de un tema de Análisis de algoritmos en el que se definen las notaciones asintóticas y sus reglas; se estudian las complejidades de algoritmos más comunes y se aplican dichos conocimientos para el análisis de los distintos algoritmos existentes de ordenación de vectores. Para éstos, se estudian complejidades mínimas, medias y máximas. El tema finaliza proponiendo una serie de algoritmos a los alumnos para su análisis que ellos deben resolver en el encerado con la ayuda del profesor.

Para finalizar, el módulo termina con el tema que será la base para el resto de módulos de la asignatura. Se trata de estudiar profundamente la Programación Orientada a Objeto (POO) al tratarse de la técnica de programación con mayor perspectiva de futuro, y aplicable a diferentes paradigmas de programación. Se estudian los antecedentes y evolución de los lenguajes orientados a objeto, desde sus orígenes con Simula-67 hasta la actualidad. Se estudia la influencia de los tipos abstractos de datos en el concepto de objeto y cómo este se extrapola a diferentes áreas de la informática. Posteriormente, se introducen los principios básicos de la POO, a saber: clases y objetos, herencia, tipos de herencia, polimorfismo, sobrecarga, clases abstractas, y clases genéricas. Estos principios se ilustran con una serie de ejemplos en diferentes lenguajes de programación (Borland Pascal, Borland C++ y Eiffel). El

objetivo no es que conozcan diferentes lenguajes de POO; por el contrario, comprender los principios básicos que aporta la POO y como estos se definen y aplican en diferentes lenguajes. Para finalizar se exponen ejercicios resueltos y se proponen ejercicios a los alumnos para su resolución en clase.

MODULO II.- ESTRUCTURAS BÁSICAS DE DATOS.

Puesto que en el primer módulo se estudiaron las Estructuras de programación y su correspondiente análisis, el objetivo de este segundo módulo es el aprendizaje de las Estructuras de datos usuales en el desarrollo de programas. Las estructuras de datos de las que consta este módulo fueron ya introducidas en el curso anterior. En este módulo se revisarán con un mayor grado de abstracción.

Se estudian en el siguiente orden las estructuras de datos básicas Conjunto, Pilas, Colas y Listas. Todos los temas de este módulo siguen la misma estructura: para cada una de ellas, se define el TAD utilizando la notación de Guttag como notación algebraica. A partir de dicha notación se implementa el TAD como estructura de clases implementando cada una de las operaciones definidas como métodos que operan con los atributos en la manera que el TAD indica. En todas ellas se estudian ejercicios resueltos y se proponen ejercicios a los alumnos para su resolución en clase.

El módulo termina estudiando aplicaciones con listas como Listas circulares, Listas doblemente enlazadas y Listas circulares doblemente enlazadas, proponiéndose las modificaciones necesarias al TAD Lista.

A continuación se expone un ejemplo de la estructura de datos Pila, especificando la notación de Guttag y su posterior representación en estructura de clases:

TAD Pila

TIPO: Pila (elemento)

SINTAXIS:

CrearPila *()*	→	*Pila*
PilaVacia *(Pila)*	→	*Boolean*
CimaPila *(Pila)*	→	*Elemento*
InsPila *(Pila,Elemento)*	→	*Pila*
BorPila *(Pila)*	→	*Pila*

SEMÁNTICA: $\forall P \in Pila; \forall i \in Elemento$

PilaVacia (CrearPila)	→	*Cierto*
PilaVacia (InsPila(P,i))	→	*Falso*
CimaPila (CrearPila)	→	*Error*
PilaVacia (InsPila(P,i))	→	*i*
BorPila (CrearPila)	→	*Error*
BorPila(InsPila(P,i))	→	*P*

Implementación en estructura de clases (Borland Pascal):

```
UNIT PILAS;
INTERFACE
USES Crt;

Type
TipoElemento = << tipo base de los elementos de la pila >>
TipoPila = << Tipo del atributo del objeto pila >>

Pila = OBJECT
        Procedure CrearPila;
        Function PilaVacia:Bolean;
        Procedure CimaPila (Var Elemento: TipoElemento);
        Procedure InsPila (Elemento: TipoElemento);
        Procedure BorPila;

        Private
        E_Pila: TipoPila;
End;

Implementation

{Implementación de los métodos descritos en la clase Pila}

End.
```

MODULO III.- RECURSIVIDAD.

Una vez estudiadas las estructuras de datos básicas, la asignatura se centra en la recursividad como técnica de resolución de problemas de manera natural. Se introduce dicha técnica comparándose con técnicas iterativas y analizándose la complejidad de funciones recursivas. Posteriormente se describen los métodos para emular la recursión. A partir de este momento los alumnos están en disposición de realizar ejercicios propuestos, analizando sus complejidades y comparando los resultados con implementaciones iterativas.

Como segunda parte del módulo se estudian dos métodos de diseño basados en recursión como la técnica "divide y vencerás" y "algoritmos con retroceso (Backtracking)". Una vez estudiada las técnicas mencionadas se proponen ejemplos resueltos como el "problema de las N reinas". Para finalizar se proponen ejercicios a resolver.

MODULO IV.- ARBOLES.

La estructura de datos árbol tiene un comportamiento recursivo, razón por la cual se estudia con posterioridad a las demás estructuras de datos. En una primera toma de contacto se explican las definiciones de árbol y árbol binario, comentando las peculiaridades y los distintos recorridos en un árbol. Se definen los TAD's con la notación de Guttag. Con posterioridad se propone una implementación dinámica como una de las posibles estructuras de almacenamiento de los nodos de un árbol, para pasar con posterioridad a la definición de la clase Arbol Binario. Tras particularizar el árbol binario general en árboles binarios de búsqueda, a continuación se estudia la representación de árboles generales usando árboles

binarios. Finaliza el módulo con problemas resueltos (árbol de expresiones) y problemas a resolver.

MODULO V.- GRAFOS.

Último módulo de la asignatura, dedicado al estudio detallado de grafos. Comienza con las definiciones de vértice, arco, caminos, ciclos, completitud, adyacencia, grafos conexos, grafos fuertemente conexos, grado, etc. Una vez el alumnado tiene estos términos asimilados, se define el TAD grafo utilizando la notación algebraica para definir posteriormente la estructura de clase Grafo.

En este momento, se comparan y evalúan las posibles implementaciones de los atributos de la clase: "matriz de adyacencia", "conjunto de vértices y conjunto de arcos" y "lista de listas"; estudiándose en profundidad la implementación de la "matriz de adyacencia".

Tras analizarse detenidamente los posibles recorridos de un grafo y obtenerse conclusiones derivadas de la forma de procesar los vértices que aportan cada uno de los recorridos, se termina el módulo y el programa de la asignatura con un estudio exhaustivo de distintos algoritmos sobre grafos. En primera instancia, se estudian algoritmos de tratamiento de caminos en grafos no valuados como el "cierre transitivo" y "algoritmo de Warshall", para estudiar con posterioridad el tratamiento de caminos mínimos en grafos valuados mediante los "algoritmos de Floyd" y "algoritmo de Dijkstra". Se proponen finalmente problemas a resolver por parte del alumno, entre los que destacan los algoritmos de Euler, Hamilton y el algoritmo de Kruskal.

BIBLIOGRAFÍA BÁSICA:

[1] M. Collado: *ESTRUCTURAS DE DATOS Y REALIZACION EN PASCAL.* Ed. Díaz de Santos, 1987.

[2] A. Aho; J. Hopcroft; J. Ullman: *ESTRUCTURAS DE DATOS Y ALGORITMOS.* Ed. Addison Wesley Americana, 1988.

[3] E. Horowitz; S. Sahni: *FUNDAMENTALS OF DATA STRUCTURES IN PASCAL.* Ed. Computer Science Press (3ª Edición), 1990.

[4] E. Horowitz et al.: *FUNDAMENTALS OF DATA STRUCTURES IN C++.* Ed. Computer Science Press, 1995.

[5] S. O'Brien. *TURBO PASCAL 7: MANUAL DE REFERENCIA.* Ed. McGraw-Hill, 1993.