25 aniversari Universitat d'Alacant

Universitat d'Alacant
Universidad de Alicante

Guía Docente

Dpnt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

FUNDAMENTOS DE PROGRAMACIÓN I

Autores:

- J. Balmaseda del Álamo
- P. Compañ Rosique
- D. González del Arco
- M. J. Monllor Doménech
- F. Mora Lizán
- H. Pastor Pina
- R. Satorre Cuerda

GUÍA DOCENTE ECTS

Fundamentos de Programación I

Ingeniería Informática

Departamento Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante

Nombre de la asignatura:	Fundamentos de Programación I
Código:	
Tipo de asignatura:	Troncal
Nivel:	Primer ciclo
Curso	Primero
Semestral/Cuatrimestral:	Cuatrimestral
Número de créditos:	6
Profesores:	José Balmaseda del Álamo Antonio Botía Martínez Patricia Compañ Rosique Pablo Garcés Rovira David González del Arco Mª José Monllor Doménech Francisco Mora Lizán Herminia Pastor Pina Rosana Satorre Cuerda

1. Contextualización

1.1. Perfil de los créditos de la materia. Adecuación al perfil profesional y académico de la titulación

A la hora de elaborar la propuesta docente que nos ocupa, se ha de tener en cuenta cuál ha sido la evolución de las distintas recomendaciones curriculares de informática publicadas por instituciones de prestigio internacional, desde las propuestas iniciales de la ACM (Association for Computing Machinery) [ACM68] y el IEEE (Institute for Electrical and Electronic Engineers) [EC77], de 1968 y 1977, respectivamente -donde por primera vez se intenta dar un carácter autónomo a la informática- hasta las tendencias actuales recogidas en el Computing Curricula 2001 [CC2001], realizado conjuntamente por el IEEE y la ACM (las referencias de dichas recomendaciones quedan recogidas al final de la sección).

En las universidades internacionales, se observa una fuerte presencia de asignaturas relacionadas con la programación, así como una gran diversificación en cuanto a asignaturas implicadas en la programación, abarcando tanto a distintos paradigmas y estilos de programación (programación imperativa, funcional, lógica, orientada a objetos, paralela, distribuida, …) como a distintos lenguajes (Modula, C, C++, Prolog, …) y, por supuesto, Estructuras de Datos y Diseño de Algoritmos, Especificación y Verificación e Ingeniería del Software.

Por otra parte, si nos situamos en el contexto español, debemos observar primero cuáles son las directrices generales propias de la titulación de Ingeniería Informática (Real decreto 1460/1990, de 26 de octubre, BOE 1990). En dicha titulación aparece como materia troncal de obligatoria inclusión en todos los planes de estudio la materia "*Metodología y Tecnología de la Programación",* cuyos contenidos son: Diseño de Algoritmos. Análisis de Algoritmos. Lenguajes de Programación. Diseño de Programas: Descomposición Modular y Documentación. Técnicas de Verificación y Pruebas de Programas, y asociados a ellos, 18 créditos troncales.

1.1.1. Referencias

[ACM68] ACM Curriculum Committee on Computer Science. Curriculum'68: Recommendations for the undergraduate program in computer science. Communications of the ACM, 11(3):151-197, March 1968.

[ACM78] ACM Curriculum Committee on Computer Science. Curriculum'78: Recommendations for the undergraduate program in computer science. Communications of the ACM, 22(3):147-166, March 1979.

[ACM2001] Association for Computing Machinery. ACM code of ethics and professional conduct. New York: The Association for Computing Machinery, May 2001. http://www.acm.org/constitution/code.html

[Bennett86] W. Bennett. A position paper on guidelines for electrical and computer engineering education. IEEE Transactions in Education, E-29(3):175-177, August 1986.

[CC2001] IEEE-CS y ACM. Computing Curricula 2001.
http://www.computer.org/education/cc2001/

[CSTB94] Computing Science and Telecommunications Board. Realizing the information future. Washington DC: National Academy Press, 1994.

[CSTB99] Computing Science and Telecommunications Board. Being fluent with information technology. Washington DC: National Academy Press, 1999.

[Denning89] Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen B. Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. Communications of the ACM, 32(1):9-23, January 1989.
[EAB86] Educational Activities Board. Design education in computer science and engineering. Technical Report 971, Computer Society of the IEEE, October 1986.
[EC77] Education Committee of the IEEE Computer Society. A curriculum in computer science and engineering. Publication EHO119-8, Computer Society of the IEEE, January 1977.
[Gibbs86] Norman E. Gibbs and Allen B. Tucker. Model curriculum for a liberal arts degree in computer science. Communications of the ACM, 29(3):202-210, March 1986.
[Martin96] C. Dianne Martin, Chuck Huff, Donald Gotterbarn, Keith Miller. Implementing a tenth strand in the CS curriculum. Communications of the ACM, 39(12):75-84, December 1996.
[Myers98] J. Paul Myers, Jr. and Henry M. Walker. The state of academic hiring in computer science: An interim review. SIGCSE Bulletin, 30(4):32a-35a, December 1998.
[Ralston80] Anthony Ralston and Mary Shaw. Curriculum '78 -- Is computer science really that unmathematical. Communications of the ACM (23)2:67-70, February 1980.
[Shaw85] Mary Shaw. The Carnegie-Mellon curriculum for undergraduate computer science. New York: Springer-Verlag, 1985.
[Tucker91] Allen B. Tucker, Bruce H. Barnes, Robert M. Aiken, Keith Barker, Kim B. Bruce, J. Thomas Cain, Susan E. Conry, Gerald L. Engel, Richard G. Epstein, Doris K. Lidtke, Michael C. Mulder, Jean B. Rogers, Eugene H. Spafford, and A. Joe Turner. Computing Curricula '91. Association for Computing Machinery and the Computer Society of the Institute of Electrical and Electronics Engineers, 1991.
[Walker96] Henry M. Walker and G. Michael Schneider. A revised model curriculum for a liberal arts degree in computer science. Communications of the ACM, 39(12):85- science as a discipline. Journal of Engineering Education, 58(8):913-916, April 1968.

1.2. Ubicación y relaciones en el plan de estudios

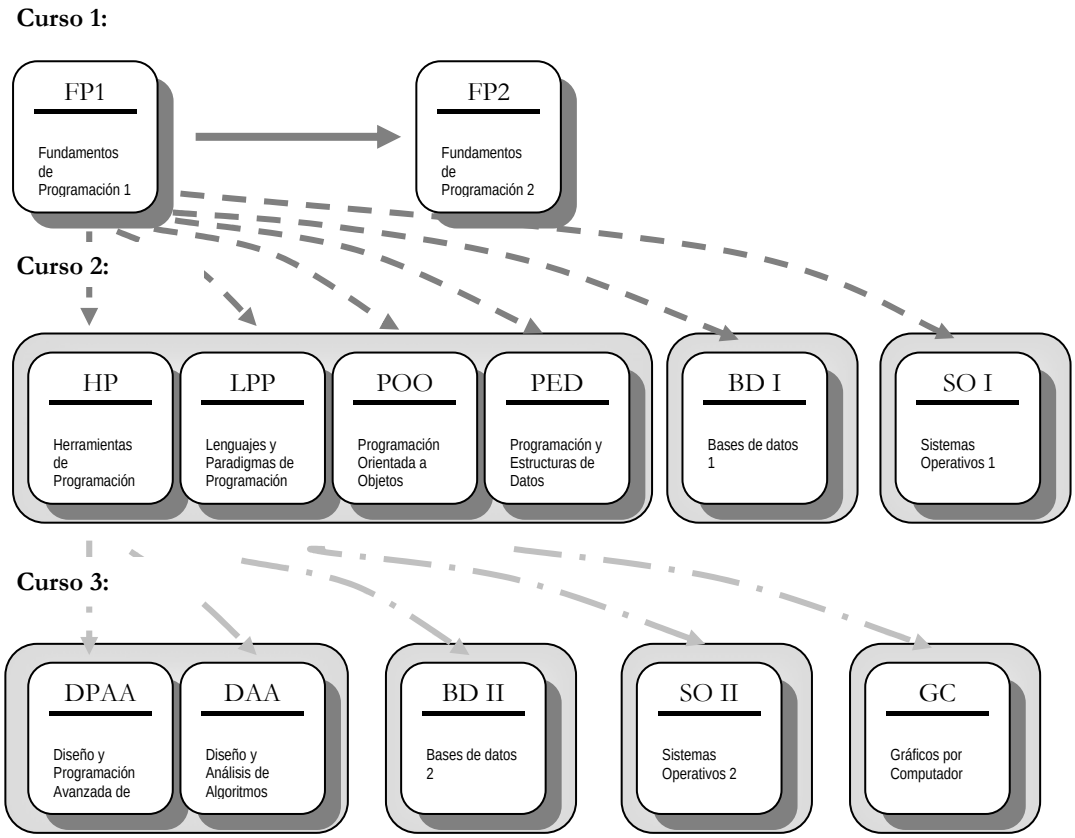
La asignatura Fundamentos de Programación I forma parte del primer curso de Ingeniería Informática como asignatura troncal que se imparte en el primer cuatrimestre. Como se ha mencionado anteriormente los descriptores de dicha asignatura son: Diseño de Algoritmos. Análisis de Algoritmos. Lenguajes de Programación. Diseño de Programas: Descomposición Modular y Documentación. Técnicas de Verificación y Pruebas de Programas. Dicha asignatura tiene una relación clara y estrecha con varias asignaturas de primer curso donde se imparten conceptos relacionados con el diseño de algoritmos, con las expresiones empleadas, así como con el diseño recursivo, y cuyo entendimiento es necesario para abordar con éxito la materia que nos ocupa. Concretamente estas asignaturas son:

Lógica Computacional: forma parte del primer cuatrimestre del primer curso de Ingeniería Informática como asignatura obligatoria. Entre sus descriptores se encuentran, lógica de primer orden (sintaxis y semántica) y sistemas de deducción, que son básicos en cualquier razonamiento formal propio de cualquier asignatura con gran componente matemática.

Fundamentos de Programación II: esta es la continuación de nuestra asignatura. En ella se pretende profundizar los conocimientos en lenguajes imperativos de 3ª generación, con el uso de Pascal y C, afianzando los conceptos vistos en Fundamentos de Programación I. En esta también se hace más hincapié en las estructuras de datos estáticas y ficheros. En ella, el alumno se adentrará en las estructuras de datos dinámicas (esta sería la mayor diferencia).

Además de estas relaciones, en cursos más avanzados, existen otras asignaturas que incluyen entre sus tópicos temas más avanzados relacionados con la programación:

La siguiente figura resume las relaciones anteriormente descritas:



2. Objetivos

2.1. Objetivos generales

2.1.1. Objetivos instrumentales generales

Comprender, interpretar y analizar el concepto de algoritmo y sus distintas interpretaciones.

Adquirir conocimientos para poder Especificar, Diseñar, Analizar y Elaborar (implementar) un algoritmo a partir del planteamiento de un problema.

Adquirir conocimientos para poder Diseñar y Elaborar un programa, independientemente del lenguaje de programación a emplear.

Comprender, conocer, analizar y aplicar los elementos básicos de un lenguaje de programación de 3ª generación: variables, estructuras de programación, funciones, tipos de datos estructurados, ficheros.

Conocer al menos dos lenguajes de programación de alto nivel concretos, el lenguaje C y/o el Pascal, diferenciando claramente aquellos aspectos dependientes de la sintaxis de datos dinámicos y ficheros.

Comprender, conocer, analizar y aplicar adecuadamente el lenguaje de programación de alto nivel adecuado a cada problema.

Saber analizar y resolver, en un plazo de tiempo razonable, algoritmos correctos, eficientes, bien organizados, bien documentados y legibles.

Conocer los distintos paradigmas de programación como alternativa y complemento a los lenguajes de programación procedimentales clásicos.

Conocer y saber aplicar a cada tipo de problema susceptible de resolución por computador las herramientas necesarias para ello.

Aplicar los conocimientos adquiridos mediante la resolución de problemas y prácticas de ordenador.

Utilizar con fluidez las herramientas necesarias en las prácticas relacionadas con la asignatura.

Conocer y utilizar la terminología usual de la asignatura en castellano y/o en valenciano, y conocer dicha terminología en inglés.

2.1.2. Objetivos interpersonales generales:

Destrezas para la participación responsable: capacidad de coordinación, asistencia, contribuciones al grupo, etc.

Capacidad de trabajar en equipo adquiriendo y mejorando las habilidades sociales y la inteligencia emocional.

Comprometerse de forma ética con el trabajo, con el resto de integrantes del grupo y consigo mismo.

2.1.3. Objetivos sistémicos generales:

Capacidad de integrar los conocimientos, métodos, algoritmos y destrezas prácticas de la programación para resolver situaciones reales.

Desarrollar la madurez en programación para abordar problemas o cuestiones planteadas, adquiriendo así destreza en el razonamiento formal y capacidad de abstracción.

Reforzar el hábito de plantearse interrogantes. Ante un problema deben preguntarse por la estructura adecuada para su solución, las posibilidades de error, etc.

Capacidad de aplicar y relacionar, de forma autónoma, la programación de forma interdisciplinar.

2.2. Competencias

2.2.1. Competencias instrumentales:

Las competencias instrumentales se han agrupado por bloques temáticos de la asignatura:

Bloque 1: Computadores y Programas

2.2.1.1. Habilidades cognitivas:

Conocer la historia de los computadores.

Conocer la evolución de la informática.

Entender el modo de representación de la información en los computadores.

Conocer cómo se programa un computador.

Sabe diferenciar entre compilador e intérprete.

Conocer el concepto de Sistema Operativo.

2.2.1.2. Destrezas tecnológicas:

Habilidades básicas para encender y parar un ordenador, abriendo en cerrando las sesiones en el sistema operativo Windows y/o Linux.

Saber iniciar programas en el sistema operativo Windows y/o Linux.

Habilidades básicas para el manejo de archivos (copia, organización, borrado) en el sistema operativo Windows y/o Linux.

2.2.1.3. Destrezas lingüísticas:

Conocer y saber utilizar la jerga informática relativa a construcción de programas.

Emplear adecuadamente los términos relativos a los Sistemas Operativos.

Bloque 2: Lenguaje algorítmico

2.2.1.4. Habilidades cognitivas:

Conocer y utilizar los tipos de datos simples y sus operadores.

Construir de forma adecuada expresiones algorítmicas.

Conocer las prioridades de los operadores.

Conocer los pasos del proceso de construcción de programas.

Analizar los problemas correctamente determinando los datos de entrada y de salida.

Identificar las distintas estructuras básicas (comentarios, asignación, estructura secuencial, estructura alternativa, estructura iterativa) de un algoritmo.

Dominar las distintas técnicas de representación de un algoritmo (lenguaje algorítmico, diagramas de cajas, diagramas de flujo).

2.2.1.5. Capacidades metodológicas:

Declarar variables y constantes basadas en tipos de datos simples.

Utilizar variables mediante sus operadores básicos.

Evaluar expresiones algorítmicas.

Determinar los datos de entrada/salida de un problema.

Diseñar algoritmos utilizando los pasos del proceso de construcción.

Aplicar las distintas estructuras básicas (comentarios, asignación, estructura secuencial, estructura alternativa, estructura iterativa) de un algoritmo para la resolución de un problema.

Estructurar correctamente un algoritmo empleando las distintas técnicas de representación de un algoritmo (lenguaje algorítmico, diagramas de cajas, diagramas de flujo).

2.2.1.6. Destrezas tecnológicas:

Diseñar un algoritmo utilizando un editor gráfico “Flowchart”.

Editar programas utilizando editores especializados y entornos integrados de programación.

Utilizar un entorno integrado para compilar y ejecutar programas.

Corregir programas haciendo uso de un depurador.

Utilizar la herramienta de ayuda de un entorno integrado de programación.

2.2.1.7. Destrezas lingüísticas:

Comprender y utilizar adecuadamente los términos referentes a tipos de datos elementales.

Utilizar adecuadamente la terminología algorítmica.

Comprender y utilizar los términos de entornos integrados de programación.

Conocer la sintaxis para la declaración y uso de tipos de datos elementales en pseudocódigo.

Conocer la sintaxis de las estructuras algorítmicas en en pseudocódigo.

Bloque 3: Evaluación del coste temporal de un algoritmo.

2.2.1.8. Habilidades cognitivas:

Conocer el concepto de coste temporal y de coste espacial de un algoritmo.

Comprender el modo de medir el tiempo de ejecución de un algoritmo.

Reflexionar sobre las capacidades deseables de un algoritmo (corrección, claridad, eficiencia).

2.2.1.9. Capacidades metodológicas:

Aplicar el método de evaluación del coste temporal de un algoritmo.

Aplicar métodos para comparar la eficiencia de varios algoritmos con tamaños de datos diferentes.

2.2.1.10. Destrezas lingüísticas:

Conocer los términos utilizados en la evaluación algorítmica.

Utilizar adecuadamente la nomenclatura de formulación de costes.

Bloque 4: Programación modular.

2.2.1.11 Habilidades cognitivas:

Comprender el concepto de subalgoritmo.

Comprender el concepto de ámbito. Distinguir entre variables locales y globales.

Identificar los distintos tipos de módulos: funciones y procedimientos.

Profundizar sobre el concepto de subalgoritmo: parámetros por valor y referencia.

Reflexionar sobre los posible efectos laterales de la modularidad.

Dominar y profundizar la idea de recursividad.

2.2.1.12.Capacidades metodológicas:

Dividir un problema en soluciones parciales (módulos).

Determinar parámetros de módulos.

Diseñar programas empleando modularización donde se haga uso de funciones.

Diseñar programas empleando modularización donde se haga uso de procedimientos.

Analizar algoritmos para detectar y eliminar efectos laterales.

Aplicar métodos para la construcción de subalgoritmos recursivos.

2.2.1.13.Destrezas tecnológicas:

Utilizar un editor especializado y un entorno integrado para moverse entre los módulos de un programa.

Utilizar un depurador para la realización de trazas sobre módulos, entrando o saltando estos.

Buscar en la ayuda del entorno integrado la definición de funciones predefinidas para el compilador.

2.2.1.14Destrezas lingüísticas:

Aprender las normas sintácticas básicas para la declaración de módulos (funciones y procedimientos) en pseudocódigo

Bloque 5: Tipos de datos estructurados

2.2.1.15.Habilidades cognitivas:

Conocer y aplicar el concepto de la estructura de vectores de datos.

Comprender el uso de vectores para almacenar, ordenar y buscar valores.

Comprender cómo declarar un vector, cómo inicializarlo y cómo referirse a los elementos individuales que lo componen.

Ser capaz de pasar vectores a módulos.

Comprender y aplicar técnicas básicas de clasificación.

Ser capaz de declarar y de manipular vectores de varios subíndices.

Comprender el tipo de datos “cadena”.

Entender los operadores de entrada/salida de para el tipo “cadena” .

Comprender las bibliotecas de funciones como medio de conseguir reutilización de software.

Entender los algoritmos de búsqueda en vectores: secuencial y dicotómica.

Conocer los algoritmos de ordenación de vectores: selección directa, inserción directa y burbuja.

Comprender el concepto de tupla y las operaciones sobre ella.

Ser capaz de pasar estructuras a funciones en llamada por valor y en llamada por referencia.

Conocer los distitos tipos de ficheros y las operaciones que se pueden realizar sobre ellos.

Comprender el proceso de archivos de acceso secuencial.

Entender el proceso de archivos de acceso directo.

2.2.1.16.Capacidades metodológicas:

Definir vectores unidimensionales.

Definir matrices.

Definir vectores n-dimensionales.

Diseñar algoritmos que hagan uso de vectores y sus operadores.

Definir el tipo de dato cadena y variables basadas en ese tipo.

Construir programas que hagan uso de cadenas y sus operadores de entrada / salida.

Aplicar las bibliotecas de funciones como medio de conseguir reutilización de software.

Diseñar e implementar algoritmos de búsqueda en pseudcódigo.

Diseñar e implementar algoritmos de búsqueda en lenguajes imperativos de 3ª generación.

Diseñar e implementar algoritmos de ordenación en pseudcódigo.

Diseñar e implementar algoritmos de ordenación en lenguajes imperativos de 3ª generación.

Dado un problema hipotético crear las estructuras de datos necesarias utilizando sólo tuplas.

Dado un problema hipotético crear las estructuras de datos necesarias utilizando sólo tuplas y vectores.

Construir módulos para leer y escribir datos de tuplas.

Construir módulos para realizar cálculos sobre los datos de tuplas y vectores.

Construir algoritmos para el creación de archivos secuenciales y de acceso directo.

Construir algoritmos para el recorrido de archivos secuenciales y de acceso directo, realizando operaciones con los datos obtenidos.

Construir algoritmos para la actualización de archivos secuenciales y de acceso directo.

2.2.1.17. Destrezas tecnológicas:

Utilizar el depurador para ver los valores de los campos de las tuplas.

2.2.1.18. Destrezas lingüísticas:

Conocer la sintaxis de los vectores y sus operadores en pseudocódigo.

Conocer la sintaxis de las cadenas y sus operadores en pseudocódigo.

Conocer la sintaxis de las tuplas en pseudocódigo.

Adquirir y utilizar la terminología usual de ficheros.

Bloque 6: Lenguajes de programación

2.2.1.19. Habilidades cognitivas:

Diferenciar los distintos lenguajes de programación.

Conocer las ventajas e inconvenientes de los distintos lenguajes de programación.

2.2.2. Competencias interpersonales:

2.2.2.1. Competencias para tareas colaborativas:

Ser capaz de aunar los conocimientos adquiridos en la asignatura y las destrezas obtenidas en la asignatura Fundamentos de Programación I con el fin de realizar opcionalmente un trabajo de calidad en equipo relacionado con los aspectos de diseño, análisis y especificación de algoritmos o con la resolución de una serie de problemas relacionados con la informática. Dicho trabajo requiere a su vez de ciertas habilidades sociales, emocionales y de coordinación.

Ser capaz de realizar opcionalmente un trabajo de calidad en equipo de ampliación de la materia estudiada.

Ser capaz de trabajar en equipo para resolver cuestiones y problemas relacionados con la materia estudiada.

2.2.2.2. Compromiso con el trabajo:

Se ha de definir un plan de trabajo en el que el volumen de trabajo de todos los miembros del equipo sea similar.

Una vez finalizado el trabajo, todos los miembros del grupo deben conocer en profundidad todo el desarrollo realizado.

Se debe cumplir el plazo de entrega de dichos trabajos.

Es importante adquirir un compromiso ético entre todos los componentes del grupo.

2.2.3. Competencias sistémicas:

2.2.3.1. Integración de capacidades cognitivas, destrezas prácticas y disposiciones:

Capacidad de aplicar los conocimientos, métodos y algoritmos vistos en la asignatura a situaciones y problemas concretos del área de informática y de otras disciplinas relacionadas.

Capacidad de aprender y aplicar, de forma autónoma e interdisciplinar, nuevos conceptos y métodos relacionados con la asignatura.

3. Prerrequisitos

3.1. Competencias y contenidos mínimos

La asignatura de Fundamentos de Programación I no tiene otras asignaturas de la carrera como prerrequisitos. Se recomiendan las siguientes habilidades: descomposición de problemas, capacidad de abstracción, así como conocimientos de informática a nivel de usuario.

3.2. Plan de trabajo y actividades para la consecución de los prerrequisitos

Los prerrequisitos necesarios para el estudio y entendimiento de esta asignatura se cubren en la propia asignatura, pues se parte desde el principio, sin presuponer conocimientos al respecto de su contenido.

Además, desde el inicio se les plantea un plan de trabajo que cada uno puede seguir en función de sus conocimientos:

Tanto en Campus Virtual como en la primera clase de la asignatura se asesora al alumnado presentándole una bibliografía no muy extensa que le puede ayudar a seguir la asignatura, así como una lista de problemas relativos a dicha materia.

Además en el Campus Virtual dispondrán de una serie de controles que les permitirá autoevaluar, mediante el examinador, si van cubriendo los conocimientos mínimos.

4. Bloques y temas de contenidos

4.1. Bloques de contenidos de aprendizaje

1. COMPUTADORES Y PROGRAMAS

1. Computadores y Programas

2. LENGUAJE ALGORÍTMICO

2. Tipos de Datos Elementales
3. Lenguaje Algorítmico. Concepto de Programa

3. EVALUACIÓN DEL COSTE TEMPORAL

4. Evaluación del coste temporal de un algoritmo

4. PROGRAMACIÓN MODULAR

5. Programación Modular

5. TIPOS DE DATOS ESTRUCTURADOS

6. Tipos de Datos Estructurados: Vectores y Matrices
7. Tipos de Datos Estructurados: Tuplas
8. Tipos de Datos Estructurados: Ficheros

6. LENGUAJES DE PROGRAMACIÓN

9. Lenguajes de Programación

4.2. Temas o unidades de contenido. Desarrollo

1. COMPUTADORES Y PROGRAMAS

Tema 1: Computadores y Programas

- 1.1. Informática: Computadores y Programas
  - 1.1.1. Computadores
  - 1.1.2. Programas
- 1.2. Representación de la Información
- 1.3. Programación del Computador
  - 1.3.1. Fases del proceso de programación
  - 1.3.2. Compilación e Interpretación
- 1.4. El Sistema Operativo

2. LENGUAJE ALGORÍTMICO

Tema 2: Tipos de Datos Elementales

- 2.1. Elementos Básicos
  - 2.1.1. Constantes
  - 2.1.2. Variables
- 2.2. Tipos de Datos Simples o Básicos
  - 2.2.1. Tipos de Datos Predefinidos
    - 2.2.1.1. Numérico Entero
    - 2.2.1.2. Numérico Real
    - 2.2.1.3. Carácter
    - 2.2.1.4. Booleano o Lógico
  - 2.2.2. Tipos de Datos Definidos por el Programador
    - 2.2.2.1. Subrango o Intervalo
    - 2.2.2.2. Enumerado

2.3. Operadores y Expresiones

Tema 3: Lenguaje Algorítmico. Concepto de Programa

3.1. ¿Qué es un Algoritmo?
- 3.1.1. Procesos
- 3.1.2. Variables
- 3.1.3. Áreas de interés en el trabajo de algoritmos

3.2. Lenguaje Algorítmico

3.3. Especificación de un Algoritmo
- 3.3.1. Predicados
- 3.3.2. Componentes de una especificación

3.4. Estructuras Básicas
- 3.4.1. Los comentarios
- 3.4.2. La asignación
- 3.4.3. La estructura secuencial
- 3.4.4. La estructura alternativa
- 3.4.5. La estructura iterativa

3.5. Técnicas de Representación de Algoritmos
- 3.5.1. Lenguaje Algorítmico
- 3.5.2. Diagramas de Flujo
- 3.5.3. Diagramas de Cajas

3. EVALUACIÓN DEL COSTE TEMPORAL

Tema 4: Evaluación del coste temporal de un algoritmo

4.1. Coste temporal de un algoritmo
- 4.1.1. Medida del tiempo de ejecución de un algoritmo
- 4.1.2. Cálculo del tiempo de ejecución de un algoritmo

4.2. Características deseables de un algoritmo
- 4.2.1. Corrección
- 4.2.2. Claridad
- 4.2.3. Eficiencia

4. PROGRAMACIÓN MODULAR

Tema 5: Programación modular

5.1. Criterios de descomposición modular

5.2. Concepto de subalgoritmo
- 5.2.1. Transferencia de información a/desde subalgoritmos: los parámetros
- 5.2.2. Lista de parámetros actuales y formales
- 5.2.3. Correspondencia de parámetros
- 5.2.4. Paso de parámetros por valor y por referencia

5.3. Variables locales y formales

5.4. Efectos laterales

5.5. Ámbito de un identificador
- 4.5.1. Reglas de ámbito

5.6. Acciones y funciones
- 5.6.1. Diferencia entre acciones y funciones

5.7. Ventajas de los subprogramas

5.8. Recursividad
- 5.8.1. Tipos de recursividad

5. TIPOS DE DATOS ESTRUCTURADOS

Tema 6: Tipos de Datos Estructurados: Vectores y Matrices

6.1. Introducción

6.2. Vectores
- 6.2.1. Vectores unidimensionales

6.2.2. Operaciones básicas
6.2.3. Vectores multidimensionales
6.2.4. Representación de los vectores en memoria
6.3. Vectores de caracteres
6.3.1. Definición de una cadena de caracteres
6.2.2. Otras operaciones con cadenas
6.4. Ordenación y búsqueda de vectores
6.4.1. Búsquedas
6.4.2. Ordenación

6.LENGUAJES DE PROGRAMACIÓN

Tema 7: Tipos de Datos Estructurados: Tuplas
7.1. Registros
7.1.1. Definición de registros
7.1.2. Operaciones con registros

Tema 8: Tipos de Datos Estructurados: Ficheros
8.1. Ficheros
8.1.1. Soportes de almacenamiento
8.1.2. Organización de ficheros
8.1.3. Operaciones básicas sobre archivos

Tema 9: Lenguajes de Programación
9.1. Introducción
9.2. Perspectiva histórica y evolución de los lenguajes de programación.
9.3. Concepto de paradigma de programación :
9.3.1. Paradigma Imperativo
9.3.2. Paradigma Funcional o Aplicativo
9.3.3. Paradigma Lógico
9.3.4. Paradigma Orientado a Objetos (O^2)
9.4. Lenguajes de programación representativos de cada paradigma :
9.4.1. Lenguajes Imperativos : PASCAL, C
9.4.2. Lenguajes Funcionales : LISP
9.4.3. Lenguajes Lógicos : PROLOG
9.4.4. Lenguajes O^2 : SMALTALK, C++
9.5. Sintaxis de los lenguajes de programación :
9.5.1. Elementos sintácticos de un lenguaje
9.5.2. Modelos de representación formal
9.5.1.1. Representación BNF
9.5.1.2. Representación EBNF
9.5.1.3. Árboles Sintácticos
9.6. Semántica de los lenguajes de programación
9.7. Propiedades de un buen lenguaje de programación

5. Metodología y estrategias de aprendizaje

5.1. Metodología docente

Los nuevos paradigmas docentes propugnan los modelos educativos que propician el pensamiento creativo enseñando a aprender por encima de enseñar conocimientos. Nos proponemos diseñar un modelo en el que la clase magistral tiene un papel importante pero no exclusivo en la transmisión de conocimientos.

Este tipo de enseñanza se va a complementar con otros procesos entre los que cabe destacar las prácticas de laboratorio y las actividades en grupos pequeños, que jugarán un papel fundamental. Concretamente, las actividades que se proponen son las siguientes:

Clases de teoría: Según las investigaciones al respecto, la lección magistral, en comparación con otros métodos, es la técnica más eficaz y económica de transmitir y sintetizar información de diversas fuentes. Por ello ocupa un lugar destacado en la docencia universitaria. De todas formas, este método tiene serios defectos como la reducción de las fuentes de información a las palabras del profesor, la suposición de que lo que se enseña es siempre asimilado, la pasividad que promueve, la falta de uniformidad en el ritmo de aprendizaje, o la dificultad de reflexionar mientras se está dedicado a la toma de apuntes. La estructura típica de una clase expositiva de este tipo será la siguiente: en primer lugar se hará una introducción en la que se presentan brevemente los objetivos de la exposición y los contenidos a tratar. Con el fin de proporcionar el contexto adecuado, en la presentación se hará referencia al material expuesto en clases precedentes, de forma que se clarifique la posición de dichos contenidos en el marco general de la asignatura. A continuación se desgranarán los contenidos objeto de estudio incluyendo exposiciones narrativas, desarrollos formales que proporcionen los fundamentos teóricos, e intercalando ejemplos y ejercicios que ilustren la aplicación de los contenidos expuestos. Se resaltarán los elementos importantes de forma que se sea capaz de distinguir lo relevante de los aspectos periféricos. Finalmente, los conceptos introducidos serán resumidos, y se elaborarán las conclusiones incluyendo una valoración de en qué medida se han alcanzado los objetivos propuestos al principio de la lección. Quizás, de los defectos que se le atribuye a la clase magistral, el mayor de ellos sea que refuerza la actitud pasiva. Para favorecer la participación estas clases se complementarán con las siguientes técnicas:

Uso de material de apoyo tales como apuntes o transparencias. No obstante, el uso de transparencias será moderado, a pesar de su atractivo visual y la estructuración que proporciona al curso, ya que empleándolas, existe la posibilidad de acelerar el ritmo de la clase, con la consiguiente disminución de la eficiencia del aprendizaje. La combinación de medios de apoyo a la presentación que se propone es la utilización de transparencias (con retroproyector o cañón proyector) con contenido general para resaltar la estructura que articula los conceptos propuestos y que servirán de guión de las clases teóricas, la pizarra para los desarrollos detallados y ejemplos aclaratorios, y la proyección con cañón para realizar demostraciones prácticas.

Se facilitará la resolución de dudas en clase mediante preguntas directas al profesor. Sucede a menudo que las dudas son compartidas por una proporción alta de la clase, pero sin embargo es difícil conseguir romper el prejuicio de que una pregunta revela las limitaciones de la persona que la hace y no la dificultad intrínseca en la comunicación y asimilación de ciertos conceptos. Con el fin de animar a la participación y crear un clima de confianza, en el que ninguna pregunta sea desdeñada como trivial o irrelevante, a lo largo del curso se programarán varias tutorías en grupos reducidos.

Se comentarán los errores más comunes que se cometen al ir adentrándose en la asignatura. Aprender de los errores tanto propios como ajenos es siempre una buena estrategia.

Se harán preguntas a los estudiantes involucrando a otros estudiantes en las respuestas. Este pequeño ejercicio fuerza a abandonar una actitud de receptor de información a una posición de colaboración en la exposición

Se propondrán, al final de algunas clases, pequeños ejercicios que favorezcan la aplicación de los conceptos introducidos en la clase o a los que se introducirán en la siguiente clase. Estos ejercicios pueden ser resueltos en la siguiente clase por algún estudiante voluntario, o por el profesor, y pueden servir para motivar una incursión en algún tema de interés.

Actividades en grupos pequeños: estas actividades estarán relacionadas con la realización de problemas y cuestiones teórico-prácticas vinculadas con la asignatura. Su objetivo será reforzar y aplicar los conceptos básicos a situaciones reales concretas y fomentar la capacidad de análisis, síntesis y autoevaluación del alumnado. El método empleado en estas clases intentará fomentar el trabajo colaborativo. Estas actividades deben considerarse como una extensión de la teoría, por lo que no deben contemplarse como una unidad aparte. Se compaginarán con temas teóricos con el fin de proporcionar el adecuado dinamismo a la explicación y en ellas,va a resultar fundamental la aplicación de técnicas de dinámica de grupos activos.

Prácticas de laboratorio: la importancia de la práctica en unos estudios de informática es crucial. El trabajo personal en los laboratorios de computación permite fijar los conocimientos que se han

adquirido en las clases expositivas y mediante el material de apoyo. Las prácticas se realizan de forma individual siempre que es posible y, como máximo, en grupos de dos personas. Con las prácticas de laboratorio se intenta impulsar el aprendizaje, experimentación, asimilación y ampliación de algunos de los contenidos de la asignatura de Fundamentos de Programación I con el uso del ordenador. Concretamente, las prácticas de esta asignatura se basarán en el análisis, diseño y desarrollo de algoritmos, no tanto de programar en un lenguaje de programación. Inicialmente, y para cubrir la fase de diseño mediante diagramas de flujo se utilizará el programa de dominio público DFD. Posteriormente, para construir y probar algoritmos se empleará un compilador de lenguaje algorítmico.

Trabajos complementarios: como extensión de los conocimientos teóricos y prácticos adquiridos se propondrán trabajos complementarios de realización voluntaria que incidirán en la nota final de la asignatura. Dichos trabajos pueden ser por tanto de índole teórica, de índole práctica, de índole teórico-práctica o de implementación de algoritmos, y podrán realizarse de forma individual o en grupos reducidos.

Tutorías: el alumnado tiene a su disposición unas horas de tutorías en las cuales puede consultar cualquier duda relacionada con la organización y planificación de la asignatura, así como dudas concretas sobre el contenido de la asignatura. Además de dichas tutorías individualizadas, como ya se ha adelantado, se programarán en función de las necesidades de los estudiantes, algunas tutorías en grupo.

5.2. Estrategias de aprendizaje

Junto a los medios tradicionales como las trasparencias, apuntes y presentaciones por ordenador, las páginas web y el Campus Virtual ofrecen innumerables posibilidades que no hay que dejar pasar. En concreto se ha elaborado, en el Campus Virtual, una página de la asignatura que incluye toda la información que el alumno necesita. El uso de la misma ha sido mayoritario en las experiencias llevadas a cabo hasta el momento. Entre otras cosas, en dicha página podemos encontrar:

Novedades: esto es una especie de tablón de anuncios con la ayuda del cual el alumno puede estar perfectamente informado de cualquier tema relacionado con la asignatura. Además de recordar los plazos de entrega de cualquier trabajo.

Clases teóricas: aquí encontramos todo lo relacionado con el temario, los guiones de teoría, los objetivos, la bibliografía y forma de evaluación.

Clases prácticas: además de contener información sobre los grupos de prácticas y profesorado que lo imparte, aparece el temario de prácticas, la documentación y el software necesario para realizarlas, así como la explicación de lo que se va a hacer en cada sesión de prácticas.

Actividades en grupos pequeños: además de contener información sobre los grupos y profesorado que los imparte, aparece la documentación para realizar dichas actividades y un esquema de lo que se va a hacer en cada sesión.

Enlaces de interés: aquí aparecen una serie de enlaces interesantes que pueden servir para profundizar en algunos contenidos de la materia.

Ejercicios de autoevaluación. en el Campus Virtual se publicará un ejercicio de autoevaluación por cada lección mediante el cual se podrá medir el grado de asimilación obtenido.

En base a todo esto, la estrategia de aprendizaje que se propone se compone de las siguientes fases:

1. Recopilación de toda la documentación de la asignatura.
2. Planificación de las clases teóricas:
   - Lectura previa del guión correspondiente a la sesión de teoría que se trate.
   - Una vez realizada la clase de teoría, se debe estudiar de forma autónoma su contenido y en caso de no entender algo intentar primero contrastarlo con otros compañeros o utilizando la bibliografía recomendada. Si esto no es suficiente, se acudirá a tutorías para tratar de solucionar el problema.
3. Planificación de las actividades en grupos pequeños:

- Una vez entendidas las explicaciones de las clases teóricas se leerá, de forma independiente, la actividad a realizar en grupos pequeños para, al inicio de la actividad, poder preguntar las dudas surgidas en el entendimiento del enunciado.
- En las actividades en grupos pequeños, cada subgrupo tendrá que hacer la actividad propuesta que será corregida en la propia aula entre todos o por el profesor fuera del aula.
- Una vez corregida la actividad propuesta, los grupos deben analizar cuáles han sido los errores cometidos para intentar no volverlos a realizar. Si es necesario, se pedirá ayuda al profesor correspondiente.

4. Planificación de las clases prácticas:
   - Una vez entendidas las explicaciones de las clases teóricas se leerá, de forma independiente, la práctica de laboratorio que se debe realizar en la sesión correspondiente para, al inicio de la sesión, poder preguntar las dudas surgidas en el entendimiento del enunciado.
   - Parte de las prácticas se realizarán en los laboratorios y parte en horas no presenciales de forma individual. Se deberá cumplir el calendario de entrega de prácticas. El profesorado corregirá con bastante celeridad dichas prácticas, indicando una vez corregidas los fallos más comunes. Cada estudiante de forma individual debe analizar cuáles han sido los errores cometidos para intentar no volverlos a realizar. Si es necesario se pedirá ayuda al profesor correspondiente.
5. Autoevaluación: una vez realizadas todas las actividades previas relacionadas con una lección concreta, el estudiante debe discernir si cree que dicha lección ha sido totalmente entendida. En caso de no ser así, debe incidir en el estudio de los contenidos que crea tener más flojos, utilizando si lo cree conveniente las tutorías y realizando algunos problemas de ampliación, bien de los propuestos en las hojas de problemas o bien haciendo uso de la bibliografía. Cuando se crea estar preparado se puede realizar el ejercicio de autoevaluación de la lección correspondiente, publicado en el Campus Virtual. Es conveniente no utilizar los apuntes la primera vez que se haga, ya que luego se podrá rehacer las veces que se quiera.
6. Evaluación final: si el resultado de todos los ejercicios de autoevaluación ha sido satisfactorio, el estudiante estará bastante preparado para la realización del examen final. No obstante, para abordar el examen final con buenas perspectivas, va a ser necesario un repaso exhaustivo del contenido completo de la asignatura incidiendo en las partes en las que se ha tenido más dificultad.
7. De forma opcional y con anterioridad a la realización del examen, se podrá hacer un trabajo complementario, individual o en grupo, para subir la nota siempre y cuando se haya aprobado el examen final.

6. Plan de trabajo de los alumnos. Especificación del tiempo y esfuerzo de aprendizaje

En las siguientes tablas se esquematiza cuál va a ser el plan de trabajo de esta asignatura.

Sesión	Teoría	Prácticas
1	Presentación Cap. 1: Introducción. Computadores y programas	Preparación Laboratorios
2	Cap. 2: Tipos de datos Elementales	Creación Grupos de Prácticas

3	Cap. 3: Lenguaje algorítmico. Concepto de Programa Concepto de Algoritmo Especificación de un Algoritmo Estructuras Básicas : Estructura Secuencial	Prac.1 : Ordenadores y Programas
4	Estructura Alternativa Estructura Iterativa	Prac.2 : Tipos de Datos Simples
5	Técnicas de Representación de Algoritmos	Prac.3 : Algoritmos. 1ª Parte Conceptos Básicos
6	Cap. 4: Evaluación del coste temporal de un algoritmo. Coste temporal de un algoritmo Características deseables de un algoritmo	Prac.4 : Algoritmos. 2ª Parte Estructuras Básicas Técnicas de Representación
7	Cap. 5: Programación Modular Variables Locales y Globales Efectos laterales Ámbito de un identificador	Prac.5 : Algoritmos. 3ª Parte Traza Complejidad
8	Procedimientos y Funciones Ventajas de los subprogramas	Prac.6 : Programación Modular I Descomposición Modular Editor de texto
9	Recursividad	Prac.7 : Programación Modular 2 Descomposición modular. (Práctica a realizar en grupos grandes)
10	Cap. 6: Tipos de Datos Estructurados: Vectores y Matrices Introducción Vectores y Matrices	Prac.8 : Programación Modular 3 Recursividad
11	Vectores de caracteres Ordenación y búsqueda de vectores	Prac.9 : Tipos de Datos Estructurados 1 Vectores y Matrices
12	Cap. 7: Tipos de Datos Estructurados: Tuplas Registros	Prac.10 : Tipos de Datos Estructurados 2 Vectores de caracteres
13	Cap. 8: Tipos de Datos Estructurados: Ficheros Archivos (Ficheros)	Prac.11 : Tipos de Datos Estructurados 3 Tuplas o Registros
14	Cap. 9: Lenguajes de Programación Introducción Perspectiva histórica y evolución Concepto de Paradigma de Program. Lenguajes representativos de cada Para. Propiedades de un buen leng. de Prog.	Prac.12 : Tipos de Datos Estructurados 4 Ficheros
15	Dudas y Ejercicios	(Prac.13 : Lenguajes de Programación Paradigmas de Programación. Programas Ejemplo : Pascal C y C++ Lisp Prolog

7. Bibliografía

7.1. Bibliografía Básica

- *"Fundamentos de Programación. Algoritmos y Estructuras de Datos y Objetos". Luis Joyanes. Ed. McGraw-Hill, 2003*
- *"Fundamentos de Programación. Libro de Problemas". Luis Joyanes. Ed. McGraw-Hill, 2003*
- *"Fundamentos de Programación. Algoritmos y Estructuras de Datos". Luis Joyanes y otros. Ed. McGraw-Hill, 1996*
- *"Fundamentos de Programación. Libro de Problemas". Luis Joyanes. y otros. Ed. McGraw-Hill, 1996*.

7.2. Bibliografía complementaria

- *"Programación en Lenguajes Estructurados". Enrique Quero. Ed. Paraninfo, 2001*
- *"Introducción a la Programación. Lógica y Diseño". Joyce Farrell. Ed. Thomson, 1999*

7.3. Otros recursos

Para la realización de las prácticas:

- Editor e intérprete de algoritmos representados en Diagramas de Flujo - *Dfd*
- Entorno integrado con compilador de leguaje algorítmico.

Además, como ya se ha mencionado, el alumnado dispone de una especie de guión de la asignatura, del que se ayudan en las clases de teoría. En dicho guión aparecen todos los conceptos y resultados teóricos, así como los algoritmos y la bibliografía recomendada en cada tema. Respecto al material de prácticas, el alumnado dispone en la red de un manual en el que se entrelazan las prácticas con las explicaciones, dándoles un carácter autodidáctico, así como de las aplicaciones informáticas Dfd y compilador que son de dominio público. Pero además de estos recursos básicos, las nuevas tecnologías están ayudando en la mejora de la calidad de la enseñanza. Reproducimos aquí una serie de enlaces interesantes:

7.3.1 Asociaciones y grupos de interés:

- ACM: Association for Computing Machinery. http://www.acm.org
- IEEE: Institute of electrical and Electronics Engineers. http://www.ieee.org
- IEEE Computer Society. http://www.computer,org
- ACM SIGCSE: Special Interest Group on computer & Science Education. http://www.acm.org/sigcse

Por otra parte, cabe destacar que actualmente el correo electrónico puede considerarse un recurso docente ya que se hace uso de él para resolver dudas sobre la asignatura. Algunas veces es sencilla la resolución de dudas de esta forma, pero muchas otras, por las características de la materia, no es fácil contestar a través del correo electrónico. No obstante cuando la duda es de compleja resolución es preferible la asistencia a tutorías ya que es la forma de asegurarnos que se ha entendido la explicación.

8. Evaluación de los procesos y resultados de aprendizaje. Sistema de evaluación

8.1. Procedimientos de evaluación

Para evaluar está asignatura debemos ser conscientes que se trata de una asignatura de primero y que por tanto no se está todavía demasiado familiarizado con el entorno universitario. Por tanto, es en el contexto de la realización de las prácticas y de las actividades en grupo donde intentamos fomentar su estudio y aprendizaje. Casi podemos considerar estas prácticas y actividades como clases de repaso y afianzamiento de los conceptos vistos en las clases teóricas. De hecho se exige que se hagan bien y si no es así deben repetirlas con la intención de que con ellas refuercen los conocimientos vistos en teoría. Puesto que las prácticas y actividades en grupos pequeños son primordiales para preparar la asignatura y superarla, se considera importante su nota para el cálculo de la nota final, concretamente, alrededor de un 40 por ciento. El examen propiamente dicho representará el 60 por ciento de la nota total. Dicho examen contendrá preguntas teóricas relativas sobretodo a conceptos, cuestiones teórico-prácticas y problemas propiamente dichos.

Para disponer de una información lo más completa posible en cada examen, el número de preguntas que se deben contestar ha de ser elevado, alrededor de unas cinco, pero teniendo en cuenta que la duración del examen no puede ser excesiva y que realmente se pueda realizar en la mitad de tiempo. En concreto, la duración de un examen no superará las tres horas.

8.2. Criterios de evaluación

La calificación se hará de acuerdo a las siguientes pautas:

Sobresaliente:

- Los problemas y algoritmos relacionados con la asignatura son resueltos con eficiencia y corrección; los procedimientos algorítmicos y de resolución de problemas son ajustados a la naturaleza del problema.
- Las destrezas experimentales son ejemplares y muestran un completo análisis y evaluación de los resultados.
- La actuación en las destrezas transferibles es generalmente muy buena.
- La participación en las clases y distintas actividades ha sido muy correcta y muy satisfactoria.

Notable:

- La comprensión conceptual es notable. Los problemas y algoritmos relacionados con la asignatura son resueltos con eficiencia y corrección; los procedimientos algorítmicos y de resolución de problemas son generalmente ajustados a la naturaleza del problema.
- Las destrezas experimentales son generalmente buenas y muestran un análisis y evaluación de los resultados aceptables.
- La actuación en las destrezas transferibles es generalmente buena.
- La participación en las clases y distintas actividades ha sido correcta y bastante satisfactoria.

Aprobado:

- El conocimiento y la comprensión del contenido cubierto en el curso es básico.
- Los problemas y algoritmos relacionados con la asignatura son generalmente resueltos de forma adecuada.
- Las prácticas de laboratorio estándares son usualmente desarrolladas con éxito razonable aunque el significado y análisis de los resultados pueden no ser entendidos completamente.

- Las destrezas transferibles están a un nivel básico.
- La participación en las clases y distintas actividades ha sido correcta pero no siempre satisfactoria.

Suspenso:

- El conocimiento y la comprensión del contenido cubierto en el curso no ha sido aceptable.
- Los problemas y procesos algorítmicos relacionados con la asignatura no son, generalmente, resueltos de forma adecuada.
- Las prácticas de laboratorio estándares son usualmente desarrolladas no satisfactoriamente y el significado y análisis de los resultados no son entendidos generalmente.
- Las destrezas transferibles están a un nivel deficiente.
- La participación en las clases y distintas actividades ha sido escasa y deficiente.