METODOLOGIA DE LA PROGRAMACION

A través de Pseudocódigo

M.ANGEL RODRIGUEZ ALMEIDA

METODOLOGIA DE LA PROGRAMACION

A través de Pseudocódigo

MIGUEL ANGEL RODRIGUEZ ALMEIDA
Licenciado en Ciencias Químicas.
Profesor de Informática de Gestión en CENEC. Valladolid.

Revisión técnica:
M.ª DEL ROSARIO JIMENEZ TENORIO
Profesora de Informática en CEDED. Madrid.

McGraw-Hill
MADRID • BOGOTA • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA
MEXICO • NUEVA YORK • PANAMA • SAN JUAN • SANTIAGO • SAO PAULO
AUCKLAND • HAMBURGO • LONDRES • MILAN • MONTREAL • NUEVA DELHI • PARIS
SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY • TOKIO • TORONTO

METODOLOGIA DE LA PROGRAMACION A TRAVES DE PSEUDOCODIGO

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos, sin el permiso previo y por escrito de los titulares del Copyright.

DERECHOS RESERVADOS © 1991, respecto a la primera edición en español por
McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A.
Edificio Oasis-A, 1.ª planta
Basauri, s/n
28023 Aravaca (Madrid)

ISBN: 84-7615-633-2
Depósito legal: M. 12.718-1991

Diseño cubierta: Juan García
Compuesto en: FER, Fotocomposición, S. A.
Impreso en: EDIGRAFOS, C/ Edison B-22, Polígono San Marcos (Getafe)

De esta edición se imprimieron 7.000 ejemplares en abril de 1991

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Contenido

Prólogo ix

1. Generalidades **1**

1.1. Introducción 1
1.2. Memoria del ordenador 1
1.3. Datos. Tipos de datos 3
1.4. Operadores 3
1.5. Orden de evaluación de los operadores 7
1.6. Variables. Tipos 7
1.7. Constantes 10
1.8. Concepto de algoritmo. Programa 11
1.8.1. Distintos tipos de programas 12
1.8.2. Características de los algoritmos 12
1.8.3. Tipos de algoritmos 12
1.9. Estructura de los pseudocódigos 13
1.10. Instrucciones de entrada 13
1.11. Instrucciones de salida 14
Resumen 15
Ejercicios propuestos 16

2. Bucle o ciclo **17**

2.1. Bucle o ciclo 17
2.2. Contador 18
2.3. Sumadores o acumuladores 19
Ejercicios resueltos 19
Resumen 28
Ejercicios propuestos 29

3. Tomas de decisión **30**

3.1. Introducción 30
3.2. Tomas de decisión 30
Ejercicios resueltos 32

3.3. Auxiliar 38
Ejercicios resueltos 39
Resumen 43
Ejercicios propuestos 44

4. Bucles anidados. Subprogramas. Switch **45**

4.1. Bucles anidados 45
Ejercicios resueltos 46
4.2. Subprogramas o subrutinas 54
Ejercicio resuelto 54
4.3. Switch o interruptor 56
Ejercicios resueltos 56
Resumen 58
Ejercicios propuestos 60

5. Presentación en pantalla: Filas y columnas variables. Cabeceras **61**

5.1. Filas y columnas variables 61
Ejercicios resueltos 62
5.2. Cabeceras 69
Ejercicios resueltos 70
Resumen 74
Ejercicios propuestos 75

6. Números aleatorios. Menús **76**

6.1. Números aleatorios 76
Ejercicios resueltos 77
6.2. Menús 81
Ejercicios resueltos 82
Resumen 85
Ejercicios propuestos 86

7. Problemas de propósito general **87**

7.1. Introducción 87
Ejercicios resueltos 87
Resumen 102
Ejercicios propuestos 102

8. Arrays unidimensionales o listas **104**

8.1. Introducción. Estructuras de datos 104
8.2. Estructuras internas de datos. Tipos 104
8.3. Arrays unidimensionales 105
8.4. Estructuras de programación para operar con listas. Ciclo: Repetir... hasta 107
Ejercicios resueltos 107
Resumen 117
Ejercicios propuestos 118

9. Número de elementos variables en las listas **119**

9.1. Número de elementos variables en las listas 119
Ejercicios resueltos 119
9.2. Ejercicios con más de un array unidimensional 123

Ejercicios resueltos 124
9.3. Listas unidimensionales paralelas 128
Ejercicio resuelto 128
Resumen 130
Ejercicios propuestos 131

10. Ordenación de las listas. Búsqueda de un elemento 132

10.1. Introducción 132
10.2. Método de la Burbuja 132
Ejercicio resuelto 132
10.3. Método del Switch 134
Ejercicio resuelto 134
10.4. Método de la Burbuja-Switch 135
Ejercicio resuelto 135
10.5. Ordenación por el Mínimo 136
Ejercicio resuelto 136
10.6. Ordenación por Inserción 138
Ejercicio resuelto 138
10.7. Método Shell 139
Ejercicio resuelto 139
10.8. Ordenación por Mezclas 141
Ejercicio resuelto 141
10.9. Búsqueda de un elemento 143
10.9.1. Búsqueda lineal en un array desordenado 143
Ejercicio resuelto 143
10.9.2. Búsqueda lineal en un array ordenado 145
Ejercicio resuelto 145
10.10. Búsqueda binaria o dicotómica 146
Ejercicio resuelto 146
Resumen 148
Ejercicios propuestos 149

11. Arrays bidimensionales 150

11.1. Arrays bidimensionales 150
Ejercicios resueltos 152
11.2. Número de elementos variables en las tablas 162
Ejercicio resuelto 162
11.3. Ordenación de los arrays bidimensionales 163
Ejercicio resuelto 163
Resumen 164
Ejercicios propuestos 165

12. Arrays tridimensionales 167

12.1. Arrays tridimensionales 167
Ejercicios resueltos 169
Resumen 177
Ejercicios propuestos 178

13. Ficheros secuenciales 180

13.1. Estructuras externas de datos: Ficheros 180
13.2. Características de los ficheros 182
13.3. Clasificación de los ficheros según su uso 182

13.4. Operaciones sobre los ficheros 182
13.5. Organización de los ficheros 183
13.6. Ficheros con organización secuencial 183
Ejercicios resueltos 184
Resumen 198
Ejercicios propuestos 199

14. Técnicas de clasificación de ficheros 201

14.1. Introducción 201
14.2. Clasificación interna 202
Ejercicio resuelto 202
14.3. Clasificación externa 205
14.4. Método de las intercalaciones sucesivas 206
Resumen 215
Cuestiones 215

15. Generación de informes 216

15.1. Introducción 216
Ejercicios resueltos 217
Resumen 233
Ejercicios propuestos 234

16. Rupturas de control 236

16.1. Introducción 236
Ejercicios resueltos 237
Resumen 248
Ejercicios propuestos 248

17. Ficheros con organización directa o aleatoria 250

17.1. Introducción 250
17.2. Métodos de transformación de claves 251
17.3. Estructura de los ficheros aleatorios 251
Ejercicios resueltos 255
Resumen 270
Ejercicios propuestos 271

18. Ficheros con organización secuencial indexada 273

18.1. Introducción 273
18.2. Instrucciones de los ficheros indexados 275
Ejercicios resueltos 276
Resumen 293
Ejercicios propuestos 293

Glosario 295

Bibliografía 298

Indice 299

Prólogo

Prácticamente no hay día en que el ser humano no asombre al resto de la humanidad con nuevos descubrimientos en algunos de los campos de la Ciencia.

Algunos de estos descubrimientos, más tarde o más temprano, entran en nuestras vidas tratando de hacerla más fácil y por ello modificando nuestras costumbres.

La Informática nos asombra cada día con nuevos lenguajes más potentes y más rápidos. Todos estos lenguajes tienen como denominador común la Metodología: el programar los ordenadores para que realicen las tareas que nosotros deseamos, en el menor tiempo posible, utilizando para tal fin unas pautas determinadas.

Este apartado de la Metodología, esencial e imprescindible, para la programación de los ordenadores es el tema de este libro.

La técnica desarrollada para la resolución de los ejercicios es el pseudocódigo, que es la base de la programación estructurada, y en consecuencia de la programación actual.

Se ha procurado que el contenido del libro cubra los temas incluidos en la asignatura de *Metodología de la Programación*, correspondiente a la especialidad de *Informática de Gestión de Formación Profesional de Segundo Grado*, así como la de las Escuelas Universitarias, por lo que la presente obra va dirigida a:

— Alumnos de enseñanzas no regladas de Informática.
— Alumnos de Informática de Gestión.
— Alumnos de enseñanza universitaria con asignaturas de programación de ordenadores.
— A todos los aficionados a la programación.

También se ha tenido muy en cuenta la programación oficial para esta asignatura o bloque temático en el *Bachillerato de Administración y Gestión*, y en el *Módulo Profesional de Programador de Gestión*, en la Reforma de las Enseñanzas Medias.

Los conceptos se han introducido de forma progresiva, de modo que el lector pueda avanzar gradualmente en el conocimiento de la Metodología.

El lenguaje utilizado trata de ser lo más sencillo posible, pero manteniendo el rigor científico que requiere el tema.

Para facilitar su estudio, los conceptos básicos se han conectado con problemas y ejercicios de la vida real.

Cada uno de estos ejercicios consta de:

— Enunciado.
— Estudio previo, donde se dan las pautas a seguir para su resolución.
— Definición de todas y cada una de las variables que se utilizan en el pseudocódigo.
— Pseudocódigo del ejercicio.
— Comprobación final donde se va explicando lo que realiza cada parte del programa y comentando por qué se hace de esta forma y no de otra.

Todo ello encaminado a que el lector entienda los diferentes pasos realizados en el pseudocódigo, con el fin de que por sí mismo sea capaz de resolver cualquier problema que se le pueda presentar. En definitiva, aprenda a pensar desde el punto de vista de la Informática.

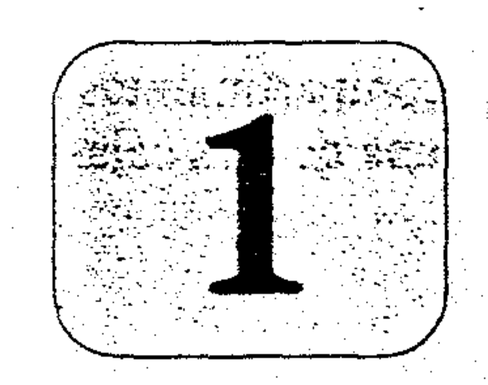
En resumen, este libro intenta sentar las bases de la programación estructurada para todas aquellas personas que quieran aprender a programar, sea cual sea el lenguaje elegido a tal efecto, fundamentalmente si se trata de lenguajes estructurados.

Por último, sólo me resta agradecer públicamente a los profesores que con sus críticas y sugerencias han mejorado notablemente el manuscrito original, a la Editorial McGraw-Hill, y en especial, a Teodoro Bartolomé, que han hecho realidad la edición.

Recibiré con agrado cualquier observación o comentario de profesores y usuarios de esta obra que, sin duda, servirá para mejorar futuras ediciones.

EL AUTOR

CAPITULO



Generalidades

1.1. INTRODUCCION

Podemos definir el ordenador como una máquina con la que conseguir el tratamiento de los datos de una forma automática.

Es importante entender que un ordenador no es más que una herramienta de trabajo, un colaborador rápido, seguro y eficaz.

Los ordenadores, como máquinas creadas por el hombre, sólo pueden realizar las tareas para las que han sido programados, es decir, no tienen ninguna inteligencia y no pueden pensar.

Un ordenador, en realidad, es capaz de hacer muy pocas cosas:

- Las cuatro operaciones básicas: suma, resta, multiplicación y división.
- Operaciones de tipo lógico: comparar dos valores, bien sean numéricos o alfanuméricos, para saber si son iguales o cuál de ellos es mayor.
- Almacenar o recuperar información.

Estas operaciones estarán convenientemente ligadas entre sí; mediante lo que llamamos *programa*, permiten al ordenador hacer trabajos complicados en un período de tiempo muy pequeño.

Se pretende idear, diseñar una solución a nuestro problema, de tal forma, que pueda luego ser utilizada por el ordenador.

1.2. MEMORIA DEL ORDENADOR

Se le suele llamar **memoria central** o **principal**; es el dispositivo de almacenamiento temporal de:

— Los datos.
— Las instrucciones.
— Los resultados intermedios y definitivos de la ejecución de los programas.

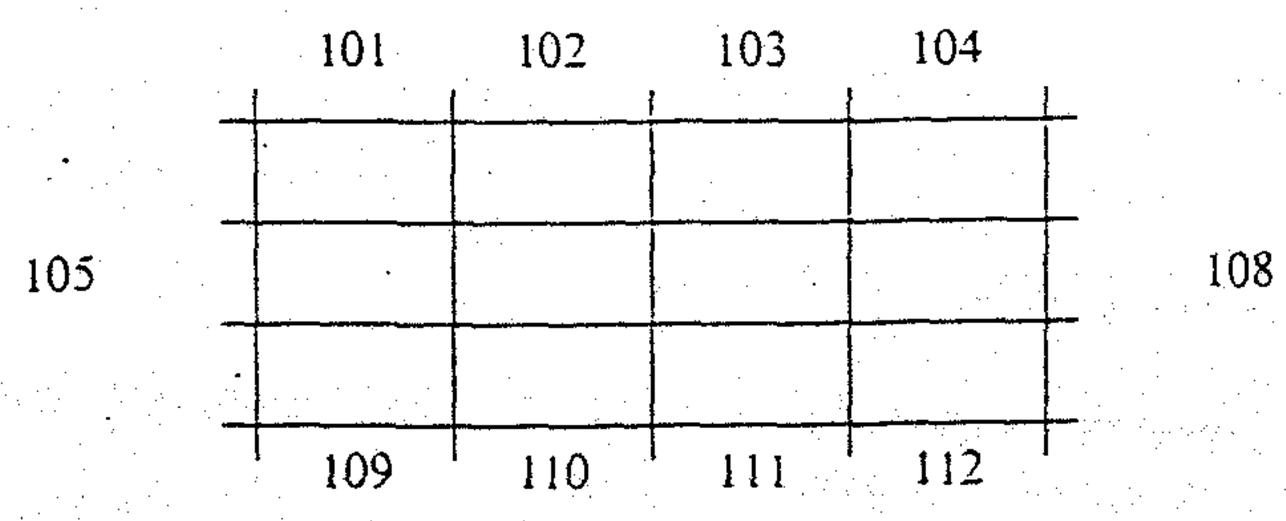
Es decir, toda, absolutamente toda la información debe pasar por la memoria del ordenador.

Podemos imaginar la memoria central como una caja dividida en compartimentos, todos ellos del mismo tamaño, con un número que los identifica. A cada una de esas cajas se le llama *dirección, celdas, posición de memoria* o *byte*.

Bit: unidad mínima de información. Puede ser 0 ó 1.

Byte: conjunto de 8 bits. Ejemplo: 1001 0110.

Supongamos que tenemos la siguiente porción de memoria con las celdas numeradas del 101 al 112.



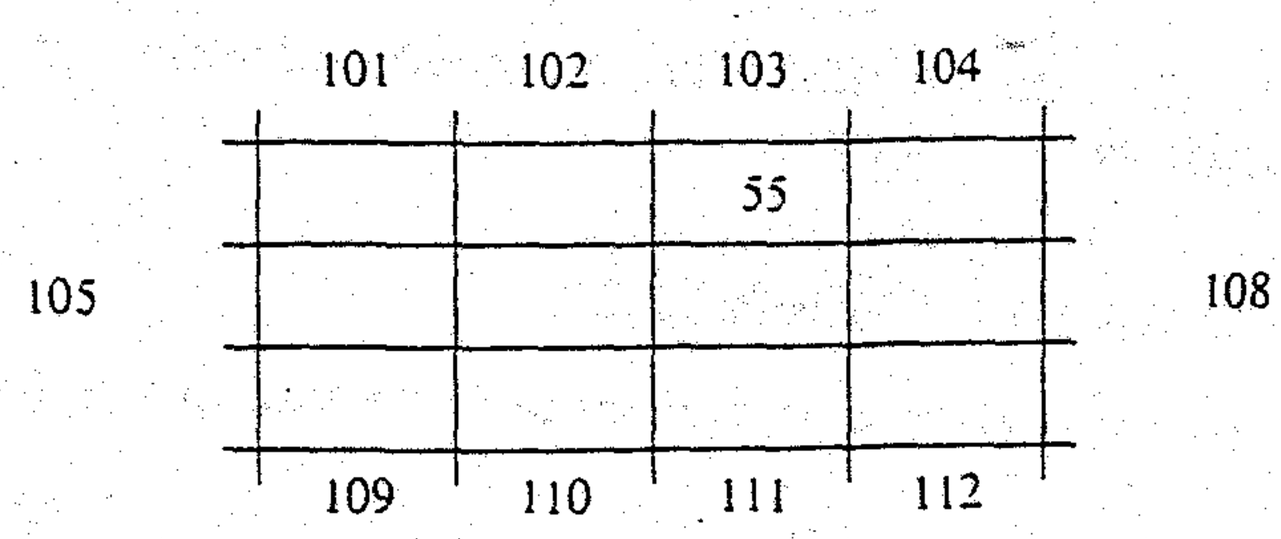
Cada celda posee una dirección distinta mediante la cual se identifica. Por medio de esa dirección se puede:

— leer o recuperar,
— escribir o grabar

información en dichas posiciones.

La lectura de la información es no destructiva, es decir, no borra o modifica el contenido de esta posición de memoria.

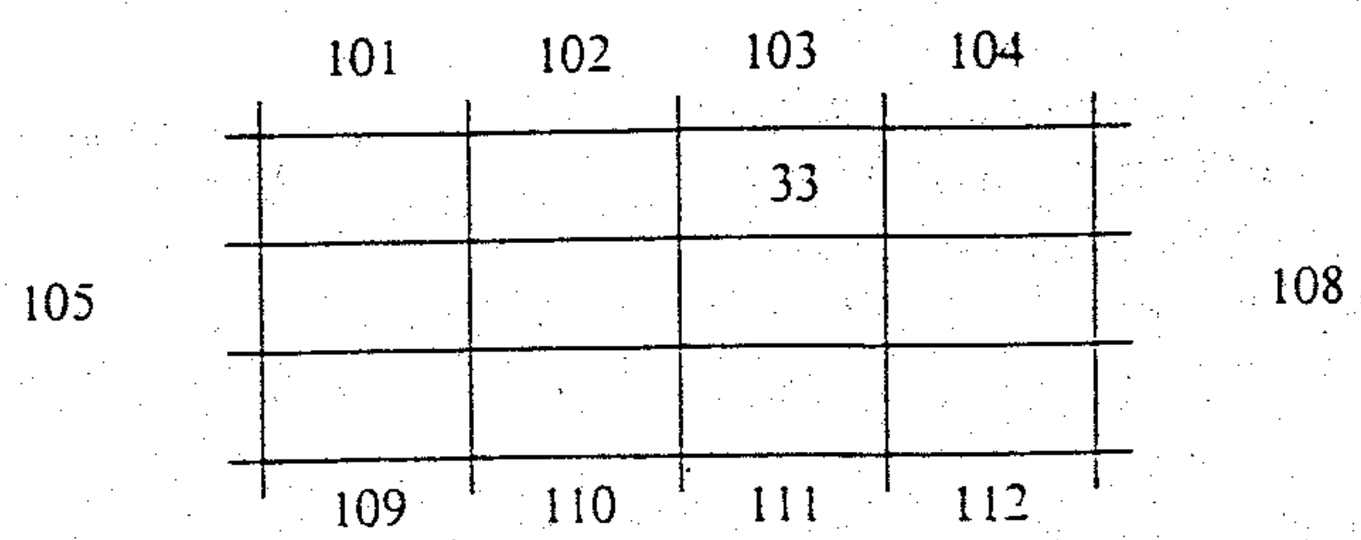
Supongamos que en la posición 103 está almacenado el número 55. Cuando leamos dicha posición siempre obtendremos dicho valor.



Por el contrario, la escritura es destructiva: borra o modifica el contenido de dicha posición por el nuevo valor.

Supongamos que en la casilla referenciada por el número 103 escribimos en ella el valor

33. Al leer de nuevo dicha casilla obtendremos el valor 33. Se ha borrado la información anterior y en su lugar hay una nueva.



Para hacer referencia a las posiciones de memoria, lo haremos por medio de *variables.*

1.3. DATOS. TIPOS DE DATOS

Dato: es toda información que utiliza el ordenador.

Según sea la información que guardemos en los datos, los clasificaremos en los siguientes tipos:

- **Numéricos:** almacenan números y con ellos podremos realizar operaciones aritméticas. A su vez, pueden ser:

  a) Enteros: son todos los números positivos o negativos.
  Ejemplo: 734, −123, 0, −1.

  b) Reales: son los números decimales.
  Ejemplo: −0.123, 67.8, 32.02.

- **Alfanuméricos:** almacena caracteres alfanuméricos. Pueden ser:

  — Letras: A-Z.
  — Caracteres especiales: guiones, paréntesis, signos de puntuación, etc.
  — Números: con ellos no se pueden realizar operaciones aritméticas.
  — Una mezcla de letras, caracteres especiales y números.
  Ejemplo: alta-2, res34.

- **Booleanos:** sólo pueden tener dos valores: *verdad* o *falso.* Estos datos no pueden leerse como datos, pero pueden imprimirse.
  Se forman a partir de los operadores relacionales y lógicos.

1.4. OPERADORES

Todos los símbolos que representan enlaces entre cada uno de los argumentos que intervienen en una operación se llaman **operadores,** y se utilizan para construir expresiones. Los operadores pueden ser:

■ Relacionales o condicionales

Se utilizan para formar expresiones booleanas, es decir, expresiones que al ser evaluadas producen un valor booleano: verdad o falso.

En la Tabla 1.1 se reúnen los distintos operadores relacionales.

Tabla 1.1. Operadores relacionales.

Signo	Significado
<	Menor que
=	Igual
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
< >	Distinto

Ejemplos:

Comparación	Resultado
25 <= 25	Verdad
25 < > 25	Falso
25 < > 4	Verdad

Cuando se comparan caracteres alfanuméricos, se hace uno a uno, de izquierda a derecha. Si las variables son de diferente longitud, pero exactamente iguales hasta el último carácter del más corto, entonces se considera que el más corto es el menor. Sólo son iguales dos datos alfanuméricos si son iguales su longitud y sus componentes.

Las letras minúsculas tienen mayor valor que las mayúsculas.

Comparación	Resultado
"A" < "B"	Verdad
"AAAA" > "AA"	Verdad
"B" > "AAA"	Verdad
"AB" > "AAAA"	Verdad
"C" < "c"	Verdad
"2" < "12"	Falso

■ Aritméticos

Para tratar los números se utilizan los operadores aritméticos, que, junto con las variables numéricas forman expresiones aritméticas.

La Tabla 1.2 resume los operadores aritméticos.

Tabla 1.2. Operadores aritméticos.

Signo	Significado
+	Suma
–	Resta
*	Multiplicación
^	Potenciación
/	División real
DIV	División entera
MOD	Resto de la división

Los operadores DIV y MOD son los de menor prioridad.

Ejemplos:

Expresión	Resultado
12 * 12	144
12 ^ 2	144
123 DIV 4	30
12 MOD 5	2

■ **Alfanumérico**

Se utiliza para unir datos alfanuméricos. En la Tabla 1.3 se describe el operador alfanumérico.

Tabla 1.3. Operador alfanumérico.

Signo	Significado
+	Concatenación

Concatenación: unir expresiones alfanuméricas como si fueran eslabones de una cadena.

Ejemplos:

Expresión	Resultado
"Pseudo" + "código"	"Pseudocódigo"
"3" + "." + "1416"	"3.1416"

De igual forma podemos concatenar expresiones del tipo:

a = "Hola, "
b = " ¿qué tal?"
a+b = "Hola, ¿qué tal?".

■ Lógicos o booleanos

Combinan sus operandos de acuerdo con las reglas del algebra de Boole con el fin de producir un nuevo valor que se convierta en el valor de la expresión.

La Tabla 1.4 define los operadores lógicos.

Tabla 1.4. Operadores lógicos.

Signo	Significado
OR	Suma lógica
AND	Producto lógico
NOT	Negación

- OR u O: es un operador binario, afecta a dos operadores. La expresión que forma es cierta cuando al menos uno de sus operandos es cierto. Es el operador lógico de disyunción.

Ejemplo: estudiamos informática o nos vamos al cine.

- AND o Y: es un operador binario. La expresión formada es cierta cuando ambos operandos son ciertos al mismo tiempo. Es el operador lógico de conjunción.

Ejemplo: es de día y hace sol.

- NOT o NO: es un operador unario. Afecta a la expresión cambiando su estado lógico: si era verdad lo transforma en falso; y al revés.

Ejemplo: no es de día.

El orden de prioridad a la hora de ejecutarse estos operadores lógicos es: NOT, le sigue AND y por último OR.

Ejemplos:

Expresión	Resultado
Verdad AND Falso	Falso
NOT Falso	Verdad
Verdad OR Falso	Verdad

Si ponemos X >= Y lo que realmente se quiere dar a entender es (X > Y) OR (X=Y). Análogamente para X <= Y, se está dando a entender (X < Y) OR (X=Y).

■ Paréntesis

Los paréntesis se utilizan para anidar expresiones, tal como demuestra la Tabla 1.5.

Tabla 1.5. Paréntesis.

Signo	Significado
( )	Anida expresiones

Ejemplos:

Expresión	Resultado
(7>4) OR (5=3)	Verdad
(7>4) AND (5<>3)	Verdad
(16=(4*4)) AND 2<>2	Falso
(2+3) * (4−7)	−15

1.5. ORDEN DE EVALUACION DE LOS OPERADORES

La prioridad a la hora de evaluar los operadores en cualquier expresión es:

— Paréntesis (empezando por los más internos).
— Potencias.
— Productos y divisiones.
— Sumas y restas.
— Concatenación.
— Relacionales.
— Lógicos.

1.6. VARIABLES. TIPOS

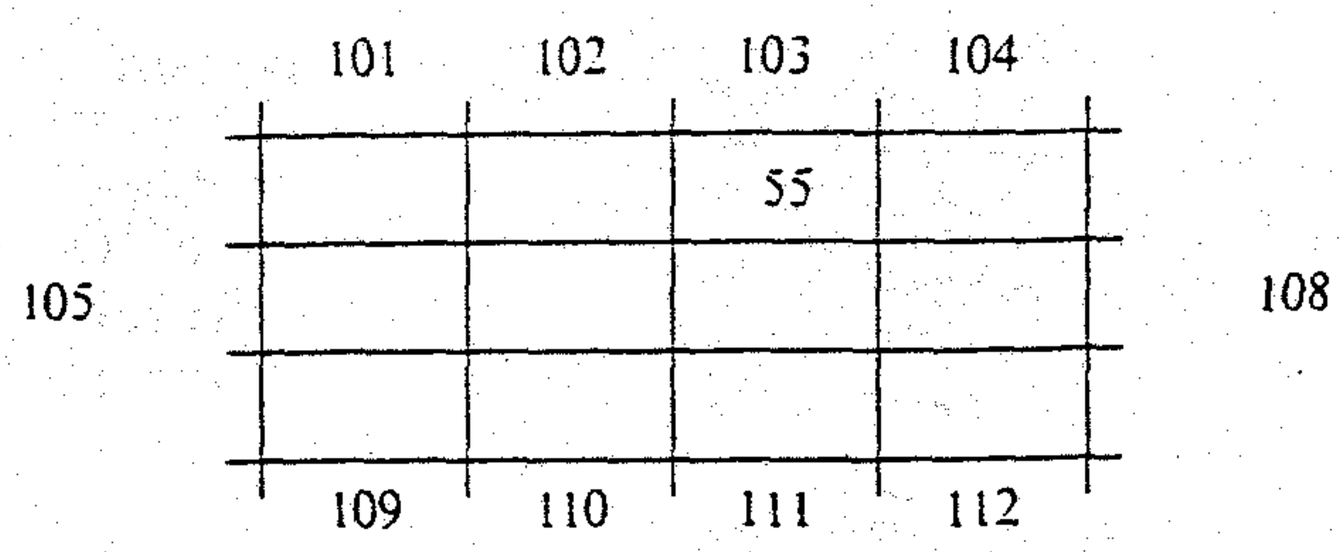
Se considera **variable** a una zona de memoria referenciada por un nombre de variable, donde se puede almacenar el valor de un dato, que puede cambiarse cuando lo deseemos. El nombre de la variable es elegido por el usuario.

Una variable no es un dato. sino un área de memoria que contendrá un dato.

El ordenador asigna a cada variable una dirección de memoria. Cuando hagamos referencia a esa variable, el ordenador siempre irá a esa dirección.

Es como si pudiéramos poner sólo los nombres en nuestras cartas y la oficina de correos mirara la dirección.

Al hablar de la memoria central del ordenador teníamos la siguiente figura:



Para leer el número 55 o cambiarlo, le teníamos que decir al ordenador que se fuera a la dirección de memoria 103.

Con la utilización de las variables, supongamos que designamos una con el nombre *numero*, el ordenador asigna una zona de su memoria libre a dicha variable; supongamos

que le asigna la dirección 103. Para leer el contenido de esa celdilla, simplemente le indicaremos al ordenador *leer numero,* y automáticamente lee el contenido de dicha celda.

Nos referiremos al nombre de la variable como la variable y decimos que su valor cambia. En realidad, la posición de memoria es la variable y su contenido es lo que cambia.

Para que estén perfectamente definidas las variables, hay que especificar:

— Su nombre.
— El tipo de dato: numérico, alfanumérico o booleano.
— El valor que va a tomar en un principio: inicializarlas.

El nombre de las variables sólo puede tener una longitud máxima de ocho caracteres, empezando obligatoriamente por una letra (a-z o A-Z), y no pueden contener espacios en blanco. El resto de los dígitos pueden ser números (0-9) y algunos caracteres especiales como el guión.

Si posee más de ocho caracteres, el resto los ignora.

Las escribimos en minúsculas y las nombramos como deseemos, sin violar la normativa anterior.

Ejemplo:

contador
hola
a3bf4
matriculacoche
matriculaalumno

Las dos últimas variables, por ser los ocho primeros dígitos iguales, están referenciadas por la misma dirección de memoria. El ordenador las reconoce como *matricul.*

Sin embargo, hay *palabras reservadas* que no se pueden utilizar por tener un sentido definido. Ejemplo: IF, ELSE, WHILE, GOTO, SQR (raíz cuadrada), ABS (valor absoluto), OR, AND, NOT, CASE, etc.

Cada lenguaje tiene sus propias palabras reservadas.

■ Tipos de variables

Las variables se utilizan para almacenar cualquier tipo de información. Según los tipos de datos que almacenen, las variables pueden ser:

- **Numéricas:** las escribimos

  nombre-variable = numero

  Ejemplo: a=1, numero=55, contador=675, etc.

- **Alfanuméricas:** las podemos escribir de dos formas:

  nombre-variable = "comentario"

Ejemplo:

b="Hola"
apellido="Fernández"
nombre="José Luis"
res = " " (un espacio en blanco)

El comentario *siempre* va entre comillas. Las comillas no las almacena, sólo su contenido.

nombre-variable = espacios (numero)

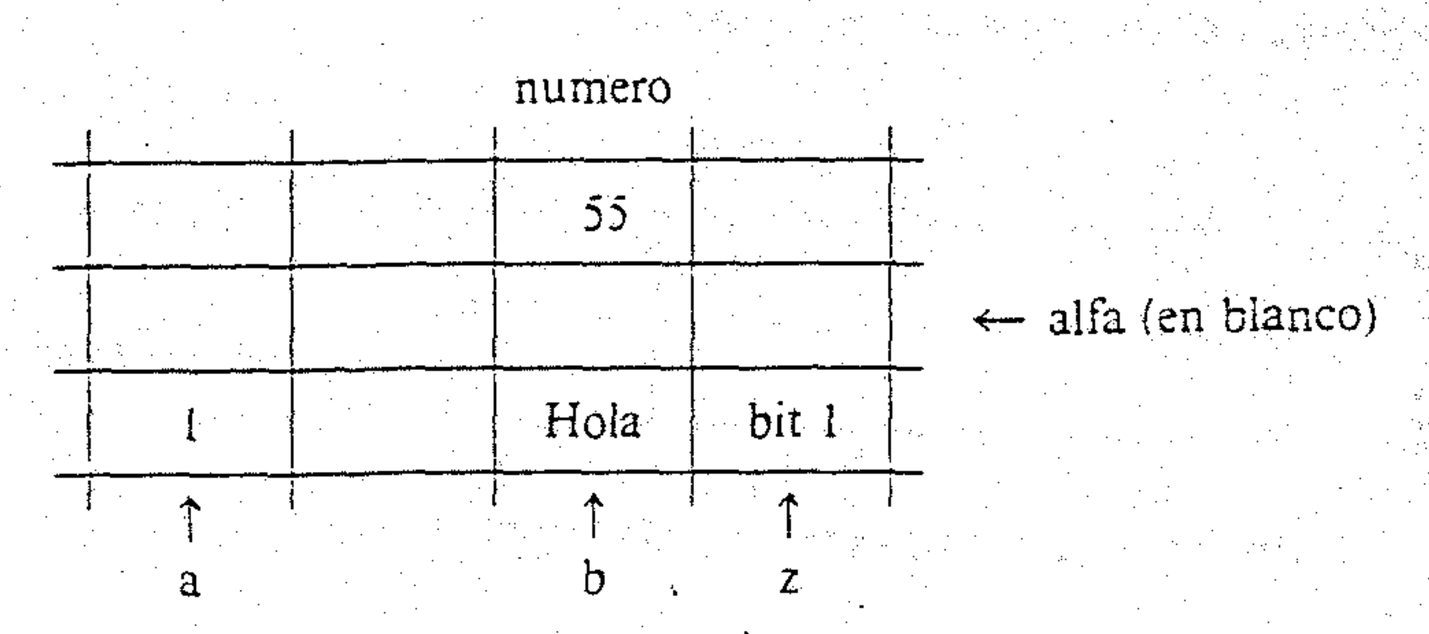
Ejemplo:

alfa=espacios(5)
apellido = espacios(30)

Entre paréntesis la longitud máxima de la variable.

- **Booleanas:** cuando almacena datos booleanos; por ejemplo: z=VERDAD.

La forma de almacenarlas en el ordenador es utilizando un único bit (unidad mínima de información, que pueden ser un 0 o un 1), 1 para verdad y 0 para falso. Algunas veces puede utilizarse 1111 1111 para verdad y 0000 0000 para falso, 1 byte por cada valor (un byte es un conjunto de 8 bits).



El ordenador a una zona de su memoria la llama *a* y guarda en ella el valor *1*. A otra zona de su memoria la llama *numero*, almacena en ella el valor *55*. En otra dirección llamada *b* escribe la palabra *Hola*; en otra denominada *alfa* reserva espacio para guardar en ella cinco caracteres y, por último, en otra posición referenciada por *z* almacena el bit *1*, verdad.

Al hecho de poner a=1 o b="Hola" se le llama *asignación de variables*.

La asignación introduce en una variable un valor o el contenido de otra variable. Tanto estos valores como la variable a la que se asigna deben ser del mismo tipo.

La forma de almacenar el contenido de una variable en otra del mismo tipo es:

nombre-variable-1 = nombre-variable-2

El ordenador guarda el contenido de la variable situada a la derecha del signo igual, *variable-2*, en su izquierda, *variable-1*.

Ejemplo:

numero = 55
contador = numero
la variable *contador* vale 55

ape = "Fernández Ferreras"
apellidos = ape
la variable *apellidos* almacena "Fernández Ferreras"

nom = "Carmen Sanz"
nombre = nom
la variable *nombre* almacena "Carmen Sanz".

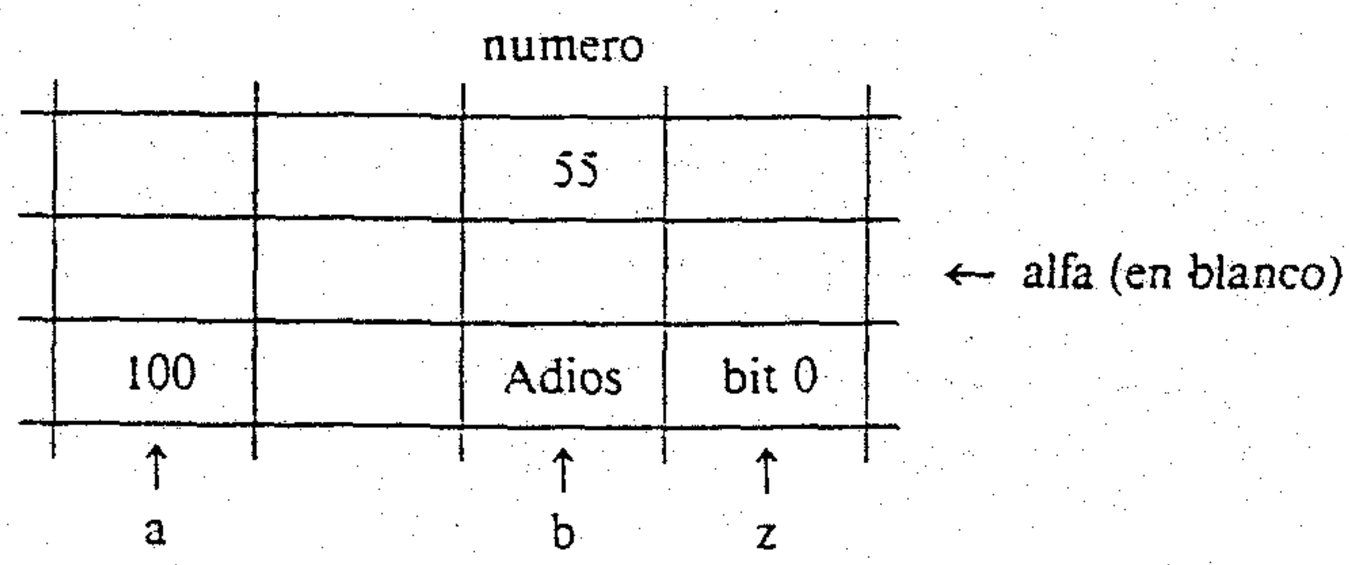
Para *inicializar* las variables se realiza de la forma:

nombre-variable = valor inicial que va a tomar la variable

es decir, mediante una asignación. Ejemplo: a=1, b="Hola", etc.

Cuando deseemos modificar el valor de alguna variable, lo haremos mediante otra asignación; ejemplo: a=100, b="Adios", z=FALSO. Estos valores deben ser del mismo tipo que tenían antes las variables: numéricos, alfanuméricos o booleanos.

En la memoria del ordenador tendríamos:



NOTAS:
Los nombres de las variables en el presente libro se escriben sin acento, aunque ortográficamente lo lleven.

Para una mejor comprensión en una misma casilla se han escrito más de un carácter, en la memoria del ordenador en cada casilla sólo se puede almacenar un carácter.

1.7. CONSTANTES

Se llama **constante** a toda posición de memoria, referenciada por un nombre de constante, donde se almacena un valor que no puede cambiarse o permanece invariable a lo largo del proceso.

Las constantes pueden ser:

- **Enteras:** números con valor entero, ya sea positivo o negativo.
  Ejemplo: an_recto = 90

- **Reales:** números con punto decimal.
  Ejemplo: pi = 3.14159
- **De caracteres o alfanuméricas:** contienen una serie de caracteres tales como letras mayúsculas, minúsculas, dígitos, signos de puntuación y otros símbolos especiales.

  Ejemplo:

  ciudad = "Madrid"
  nacion = "España"
  telefono = "91-123.45.67"

1.8. CONCEPTO DE ALGORITMO. PROGRAMA

La palabra algoritmo significa "método de resolución de un problema en un número finito de pasos".

Algoritmo no significa la resolución de un problema particular para unos datos particulares, sino la resolución de todos los problemas del mismo tipo, sean cuales sean los datos de que se parta, y previendo las alternativas de actuación convenientes según los distintos tipos de datos y situaciones que se nos puedan presentar.

Un algoritmo es la descripción exacta y sin ambigüedades de la secuencia de pasos elementales a aplicar para, a partir de los datos del problema, encontrar la solución buscada. Para que un algoritmo sea completo deberá contemplar todas las alternativas lógicas posibles que las distintas combinaciones de valores de los datos puedan presentar.

Un **algoritmo** es un método general de resolución de todos los problemas del mismo tipo.

Todo lo que se ha de hacer debe estar de forma detallada, paso a paso.

A cada uno de estos pasos de un algoritmo se le llama *sentencia* o *instrucción*.

Una **instrucción** es una combinación de palabras, variables, constantes y símbolos que obedeciendo a la sintaxis propia del lenguaje, son utilizados por el ordenador para realizar una determinada acción.

Cada instrucción se escribe en una línea diferente.

Un **programa** es la expresión de un algoritmo en un lenguaje de programación entendible por el ordenador.

No existe un lenguaje único y universal para la descripción de los algoritmos. Ejemplo: Basic, Cobol, Pascal, C, Clipper, etc.

El algoritmo o programa de ordenador consiste en dos partes fundamentales:

— Descripción de las acciones que deben ejecutarse mediante las instrucciones.
— Descripción de los datos que son manipulados por esas instrucciones mediante declaraciones y definiciones.

1.8.1. Distintos tipos de programas

Según la forma de trabajo, los programas pueden ser:

- **Lineales o secuenciales:** las instrucciones se ejecutan en el mismo orden que se han codificado.
- **Cíclicos:** un grupo de líneas se ejecuta un número determinado de veces.
- **Alternativos:** existen ciertas condiciones que provocan la ejecución de fases diferentes del programa dependiendo de que se cumplan o no dichas condiciones.

1.8.2. Características de los algoritmos

Cualquier problema puede tener diferentes formas de solución, es decir, de construir el algoritmo, cada uno de ellos con sus ventajas e inconvenientes.

Hay que elegir el más adecuado. Su elección debe cumplir una serie de características:

- **Finito:** que finalice.
- **Legibilidad:** debe estar escrito de tal forma que sea fácil de leer y de entender.
- **Modificabilidad:** las modificaciones y actualizaciones necesarias para una nueva situación del programa deben ser fáciles de realizar.
- **Eficiencia:** deben ocupar lo menos posible para aprovechar al máximo la memoria del ordenador y para que el tiempo de ejecución sea el menor posible.
- **Modularidad:** el programa, llamado programa principal, puede estar subdividido en módulos o programas más pequeños, llamados subprogramas, cada uno de los cuales realiza una parte del problema.
- **Estructuración:** comprende todas las características anteriores. Como consecuencia de una mayor estructuración, resulta más fácil:
  - — Leerlo.
  - — Modificarlo.
  - — Eliminar las partes del programa que se puedan repetir.

1.8.3. Tipos de algoritmos

Entre los distintos tipos de algoritmos se encuentran:

- **Diagramas de flujo u ordinogramas:** utiliza símbolos gráficos para su resolución: Presentan los siguientes inconvenientes:
  - — Cualquier mínima modificación en el diagrama nos obliga a reorganizarlo de nuevo.
  - — Utiliza la técnica lineal, en desuso hoy en día.
  - — El proceso de recorrer el diagrama desde el principio al final puede resultar complejo y propicia la omisión de una cierta combinación poco frecuente, pero posible, de condiciones bajo las cuales el algoritmo se comporta mal.

- **Tablas de decisión:** tabulan todas las posibles situaciones que se pueden presentar en el problema y las correspondientes acciones a tomar para cada una de ellas.
- **Pseudocódigo:** describe un algoritmo utilizando una mezcla de frases en lenguaje común, instrucciones de lenguaje de programación y palabras clave que definen las estructuras básicas.

Este método es el que vamos a desarrollar en este libro.
Presenta todas las ventajas de las características de los algoritmos.
El pseudocódigo es la técnica utilizada en todos los lenguajes estructurados: PASCAL, C, ADA, dBase, Cobol, etc.

1.9. ESTRUCTURA DE LOS PSEUDOCODIGOS

Para su realización, los vamos a considerar divididos en tres partes:

- **Inicial:** en ella
  - — inicializaremos las variables que posteriormente vamos a utilizar,
  - — abriremos los archivos o ficheros,
  - — introduciremos por teclado los valores de aquellas variables que deben ir fuera del ciclo,
  - — etcétera.

  En general, todo aquello que el ordenador debe realizar sólo una vez.
- **Repetitiva o Ciclo:** es el cuerpo del programa. Se debe estar repitiendo un número determinado de veces o hasta que se cumpla o deje de cumplirse una determinada condición.
- **Final:** en ella
  - — incluiremos la impresión de los resultados finales que resultan del ciclo,
  - — cerraremos archivos,
  - — etcétera.

No todos los pseudocódigos tienen por qué tener las tres partes perfectamente diferenciadas. Habrá casos en los que la parte inicial o final no están diferenciadas del ciclo. Incluso alguna parte puede contener las tres. Todo depende de nuestro problema.

1.10. INSTRUCCIONES DE ENTRADA

Se utilizan para tomar datos del exterior, guardándolos en variables.

Sintaxis:

introducir nombre-variable

Ejemplo: supongamos que deseamos introducir desde el teclado del ordenador un nú-

mero; para ello es necesario, en primer lugar, definir una variable que recoja dicho valor, *numero*. La forma de realizarlo es la siguiente:

numero = 0
introducir numero

Inicializamos la variable a un valor, el que queramos, mediante la asignación numero = 0.

A continuación, cuando el ordenador se encuentre con la instrucción *introducir numero*, espera a que lo hagamos. Introducimos, por ejemplo, el 3 y pulsamos la tecla <ENTER> o <RETURN>.

El ordenador, en la zona de su memoria conocida con el nombre de *numero*, almacena el valor 3, es decir, numero = 3.

1.11. INSTRUCCIONES DE SALIDA

Sirven para presentar en pantalla o en impresora comentarios, constantes, contenido de las variables y resultado de expresiones.

Sintaxis:

— **Visualización de comentarios:**

imprime "comentario"

entre comillas irá el comentario a imprimir.

Ejemplo: imprime "Hola"

visualiza por pantalla o por impresora la palabra Hola.

— **Visualización de variables, constantes y expresiones:**

imprime nombre-variable

El nombre de la variable irá sin comillas para que visualice su contenido.

Ejemplo: numero = 3
imprime numero

visualiza el contenido de dicha variable, el 3.

Ejemplo: imprime 2+3*4

visualiza el número 14.

Ejemplo: a = "Hola, "
b = "¿qué tal?"
imprime a+b

visualiza el comentario: Hola, ¿qué tal?

RESUMEN

Datos: es toda información que utiliza el ordenador. Tipos:

— Numéricos: almacena números.
— Alfanuméricos: almacena caracteres alfanuméricos.
— Booleanos: sólo pueden tener dos valores *verdad* o *falso*.

Operadores: se utilizan para construir expresiones.

Los operadores pueden ser:

— Relacionales o Condicionales.
— Aritméticos.
— Alfanuméricos.
— Lógicos o Booleanos.
— Paréntesis.

Variable: un área de la memoria del ordenador que contendrá un dato. Quedan definidas mediante:

— Su nombre, a elección del usuario.
— El tipo de dato: numérico, alfanumérico o booleano.
— Su inicialización.

Las **palabras reservadas** no se pueden utilizar como nombres de variables.

Tipos de variables:

— Numéricas: *nombre-variable = numero*
— Alfaméricas: se pueden escribir de dos formas:

nombre-variable = "comentario"
nombre-variable = espacios(numero)

— Booleanas.

Forma de almacenar el contenido de una variable en otra del mismo tipo:

nombre-variable-1 = nombre-variable-2

Constante: un valor que permanece invariable a lo largo del proceso.

Algoritmo: es un método general de resolución de todos los problemas del mismo tipo.

Sentencias o instrucciones: son utilizadas por el ordenador para realizar una determinada acción.

Programa: es la expresión de un algoritmo en un lenguaje de programación entendible por el ordenador.

Según la forma de trabajo, los programas pueden ser:

— Lineales o secuenciales.
— Cíclicos.
— Alternativos.

Pseudocódigo: describe un algoritmo utilizando una mezcla de frases en lenguaje común, instrucciones de lenguaje de programación y palabras claves que definen las estructuras básicas.

Estructura de los pseudocódigos: los consideramos divididos en tres partes:

— Inicial.
— Repetitivo o ciclo.
— Final.

Instrucciones de entrada: se utilizan para tomar datos del exterior.

Instrucciones de salida: sirven para presentar en pantalla o en impresora comentarios, constantes, contenido de las variables y resultado de expresiones.

EJERCICIOS PROPUESTOS

1. Indicar el valor de cada una de las siguientes variables:

   a) x = (2+3)*6 30
   b) x = (12+6) /2*3
   c) x = (2+3)/4
   d) x = (2+3) MOD 4 1
   e) x = (2+3) DIV 4 1
   f) x = (3*4+2) * (15 DIV 2)
   g) x = 2^2+3–2* (5 MOD 2)
   h) x = 6*6*2/8–3* (19 DIV 6)

2. Si a=6, b=2 y c=3, indicar el resultado final de las expresiones:

   a) a–b+c
   b) a*b/c
   c) (a*c) MOD c
   d) (a+b) DIV c
   e) c^b+c*b
   f) MOD (a*b*c/ (b+c))

3. Calcular el valor de cada una de las siguientes asignaciones de variables:

   a)
   a = 3
   b = 0
   c = a+b
   b = a+b
   a = b

   b)
   a = 10
   b = 5
   a = b
   b = a

   c)
   a = 1
   b = 4
   c = a+b
   d = a–b
   a = c+2*b
   b = c+b
   c = a*b
   d = b+d

   d)
   a = 8
   b = 5
   c = 0
   c = c+a
   a = a+c–2*b
   b = b + b
   a = c
   b = v

4. Expresar, utilizando los operadores aritméticos, las siguientes expresiones:

   a) $\dfrac{m + n}{n}$ *b)* $\dfrac{m + n / p}{p - r / s}$ *c)* $\dfrac{m + 4}{p - q}$ *d)* $\dfrac{c\ r\ t}{100}$

5. Decir si son ciertas o falsas las expresiones:

   "12" + "12" = "24"
   "12" + "12" = "1212"
   "hola" = "hola"
   "hola" = "hola "
   12 + 12 = 24
   Verdad AND Verdad = Falso
   NOT Falso = Verdad
   (Verdad AND Verdad) OR Falso = Verdad
   (Falso OR Falso) AND Falso = Verdad
   NOT (Verdad AND Falso) = Falso

CAPITULO

Bucle o ciclo

2.1. BUCLE O CICLO

En la naturaleza y en la vida hay situaciones que se repiten un número de veces en períodos más o menos espaciados.

Todos los días nace el sol por el este y se pone por el oeste. Un año está formado por 365 días, pero cada cuatro años tiene un día más.

Los periódicos salen a la calle todos los días. Las revistas se editan cada semana o cada mes.

En informática, la mayoría de las veces la tarea que debe realizar el ordenador es la misma: lo único que varía son los valores de los datos con los que está operando.

Llamamos **bucle** o **ciclo** a todo proceso que se repite un número de veces dentro de un programa.

Sintaxis:

HACER mientras condición
instrucción1
FIN del HACER
instrucción2

Ejemplo: hacer mientras res = "S"
hacer mientras fi < > 20

donde *res* y *fi* son nombres de variables.

Lo primero que hace el ordenador es examinar la condición. Da como resultado dos posibilidades:

- Se cumple: va a realizar todas las instrucciones que están dentro del ciclo, *instrucción1;* las estará repitiendo hasta que deje de cumplirse la condición. Entonces sale del ciclo y continúa ejecutando las instrucciones que hay fuera de él, *instrucción2.*
- No se cumple: no entrará en el ciclo. Ejecuta las instrucciones que están fuera de él, *instrucción2.*

La condición del bucle no tiene por qué ser única; puede haber más de una, siempre y cuando estén unidas por los operadores lógicos (OR, AND y NOT). Lo expresamos:

HACER mientras condición-1 operador-lógico condición-2 ...

Ejemplo: hacer mientras res = "S" AND fi < > 20.

En las competiciones de coches, los automóviles están dando vueltas al circuito hasta que el comisario baje la bandera a cuadros o hasta que suspenda la carrera porque ha ocurrido un accidente.

Ejemplos de bucles: FOR, WHILE, PERFORM UNTIL, REPEAT.

NOTA:
Las instrucciones que están situadas dentro del bucle HACER, para una mayor claridad a la hora de comprobar el pseudocódigo, las escribiremos desplazadas hacia la derecha.

2.2. CONTADOR

Un **contador** es una variable destinada a contener diferentes valores, que se va incrementando o decrementando cada vez que el ordenador realiza la instrucción que lo contiene.

El incremento, o decremento si es negativo, llamado también paso de contador, es siempre constante.

Ejemplo: en un partido de fútbol cada vez que un equipo marca un gol, su casillero siempre aumenta en una unidad. En las carreras de coches, cada vez que un vehículo pasa por la línea de meta, se incrementa en una unidad el número de vueltas dadas al circuito o se decrementa en una unidad el número de vueltas que le quedan por realizar.

LOCAL	VISITANTE
2	1

a) Marcador en un instante determinado.

LOCAL	VISITANTE
3	1

b) Marcador después de conseguir un gol el equipo local.

Sintaxis:

VARIABLE = VARIABLE ± CONSTANTE

El ordenador primero evalúa la expresión situada a la derecha del signo igual, realiza la suma o la resta y su resultado lo asigna a lo que hay a la izquierda del igual.

Ejemplo:

```
vueltas = vueltas + 1
goles = goles + 1
faltan = faltan - 1
numero = numero + 5
```

El valor de la constante no tiene por qué ser la unidad; puede ser cualquier número, pero en todo el programa se debe conservar siempre dicho valor.

```
i = 5
i = i + 2
imprime i
```

La variable i está inicializada a 5, por la asignación i = 5. En la línea siguiente al valor de i, que es 5, le sumamos 2, dando como resultado 7. Este valor lo guarda de nuevo en i; por tanto, ahora i = 7. Al decirle que imprima el valor de i, dará como resultado 7.

2.3. SUMADORES O ACUMULADORES

Es una variable que nos va a permitir guardar un valor que se incrementa o decrementa de forma *no* constante durante el proceso. En un instante determinado tendrá un valor y al siguiente tendrá otro valor igual o distinto.

Ejemplo: cuando realizamos una imposición en el banco, la cantidad entregada no siempre es la misma; unas veces será una cantidad y otras veces distinta. Análogamente sucede cuando retiramos dinero.

Para realizar la suma de los diez primeros números, primero sumamos el 1, luego el 2, posteriormente el 3, y así sucesivamente. El valor que estamos sumando es diferente cada vez.

Sintaxis:

$$SUMADOR = SUMADOR \pm VARIABLE$$

SUMADOR es la variable que almacena el resultado de la suma.
VARIABLE almacena el número que estamos sumando.

Ejemplo:

```
saldo = saldo + entrega
saldo = saldo - retirar
suma = suma + numero
```

ejecuta en primer lugar lo que hay a la derecha del signo igual. Realiza la operación *suma + numero*, para el último caso; el resultado lo guarda en la variable sumador *suma*.

Ejercicio 1

Hacer un pseudocódigo que imprima los números del 0 al 100.

ESTUDIO PREVIO

Necesitamos una variable contador que vaya tomando los valores de 0 a 100 de una forma consecutiva y de uno en uno.

Al tener que imprimir el 0 debemos inicializarla a 0.

Hay que utilizar un bucle para que imprima el valor de la variable y al mismo tiempo vaya aumentándola en una cantidad hasta que el último valor visualizado sea 100.

VARIABLE

c = contador para generar los números.

PSEUDOCODIGO

```
Borrar pantalla
c=0
Hacer mientras c<101
     imprime c
     c=c+1
fin del hacer
fin del programa.
```

COMPROBACION

Lo primero es borrar pantalla para eliminar todo lo que haya en ella y sólo nos aparezca la ejecución del programa.

Inicializamos el contador. Comprueba la condición *c<101*, es decir, ¿ 0 < 101 ? sí, por tanto entra en el ciclo. Nos imprime su valor: 0, y lo incrementa en uno, *c=c+1*.

Verifica la condición del bucle, ¿ c < 101 ? sí, imprime c, que ahora vale 1, incrementándolo de nuevo. Vuelve a comprobar la condición ...

Cuando c=100, al ser 100 < 101 entra de nuevo, imprime c, que vale 100, y le suma 1, c=101.

Vuelve a comprobar la condición: ¿ 101 < 101 ? no, entonces ya no entra en el ciclo.

La línea siguiente es final de programa y acaba.

Si hubiéramos puesto hacer mientras i<100, sólo habría impreso hasta el 99.

Hay que tener cuidado con la condición de salida.

Es aconsejable comprobar que el *ordenador* realiza aquello que nosotros deseamos hacer en cada instante.

También se podía haber puesto la siguiente condición de salida del bucle:

hacer mientras c<=100.

Ejercicio 2

Hacer un pseudocódigo que imprima los números del 100 al 0. en orden decreciente.

ESTUDIO PREVIO

Necesitamos una variable contador que partiendo del valor 100 se vaya decrementando en una unidad hasta que valga 0.

La inicializamos a 100 por ser el primer número que debe imprimir.

Utilizamos un bucle para que imprima el valor de la variable y al mismo tiempo vaya decrementándola en una unidad.

VARIABLE

c = contador para generar los números.

PSEUDOCODIGO

```
Borrar pantalla
c= 100
Hacer mientras c>=0
     imprime c
     c=c-1
fin del hacer
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos el contador a 100.

Comprueba la condición *c>=0*, es decir, ¿ 100>=0 ? sí, por tanto entra en el ciclo. Nos imprime su valor: 100, y lo decrementa en una unidad, *c=c-1*.

Verifica la condición del bucle, ¿ c>=0 ? sí, imprime c, que ahora vale 99, lo decrementa de nuevo. Vuelve a comprobar la condición ...

Cuando c=0, al ser 0 >= 0 (no cumple la condición de mayor pero sí la de igual) entra de nuevo, imprime c, que vale 0 y le resta 1, c=-1.

Vuelve a comprobar la condición: ¿ -1 >= 0 ? no, entonces ya no entra en el ciclo.

La línea siguiente es final de programa y acaba.

Ejercicio 3

Hacer un pseudocódigo que imprima los números pares entre 0 y 100.

ESTUDIO PREVIO

Los números pares van de 2 en 2. Además empiezan en 2; por tanto, necesitamos un contador que empiece en 2 y aumente en dicha cantidad. Este proceso debe realizarlo hasta que imprima el valor 100.

VARIABLE

c = contador que genera los números pares.

PSEUDOCODIGO

```
Borrar pantalla
c=2
Hacer mientras c<101
     imprime c
     c=c+2
fin del hacer
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos el contador.

Al ser c < 101 entra en el ciclo, imprime c, que vale 2, e incrementa su valor en 2, ahora c=4. Va a comprobar de nuevo la condición ¿ 4 < 101 ? se cumple: entra de nuevo...

Cuando c= 100, comprueba ¿ 100 < 101 ? sí, entra e imprime 100 y le aumenta 2, c=102. Vuelve a comprobar ¿ 102 < 101 ? no, no entra en el ciclo.

Se encuentra con el final del programa y finaliza.

Ejercicio 4

Hacer un programa que imprima la suma de los 100 primeros números.

ESTUDIO PREVIO

La misión a realizar es doble; por un lado, hay que generar los números naturales y, por otro, a medida que se generan se suman.

Para la primera operación necesitamos un contador que aumente de uno en uno, y para la segunda, un sumador que vaya realizando la suma del número generado. Todo dentro de un bucle que vaya de 1 a 100.

La impresión de la suma se realiza fuera del bucle y una vez realizada la misma, porque solamente hay que imprimir el resultado final y no los distintos resultados intermedios.

VARIABLES

c = contador que genera los números.
suma = sumador donde se va realizando la suma de los números.

PSEUDOCODIGO

```
Borrar pantalla
c = 1
suma = 0
Hacer mientras c<=100
     suma = suma + c
     c = c + 1
fin del hacer
imprime "La suma de los 100 primeros números es: "
imprime suma
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos las variables, el contador a 1 por ser el primer número distinto de cero y el sumador a 0 porque antes de empezar a sumar no debe valer nada.

Al ser c<=100 entra en el ciclo. Primero realiza la suma del número, *suma=suma +c*, y posteriormente genera el siguiente número, *c= c+1*.

Si se hubiera realizado la operación al revés, primero generar el número y luego sumar, no sumaría el número 1.

Generado el siguiente número, el 2, va a comprobar de nuevo la condición ¿ 2<=100 ? se cumple: entra de nuevo...

Cuando c = 100, comprueba ¿ 100 <= 100 ? sí, suma el número 100 y genera el siguiente: el 101. No se cumple la condición, sale del ciclo e imprime el comentario de "La suma de los 100 primeros números es: " y a continuación el valor de la suma que está guardada en la variable *suma*.

De esta forma sólo imprime el valor de la suma una única vez, por estar fuera del bucle.

Final del programa.

Ejercicio 5

Hacer un pseudocódigo que imprima los números impares hasta el 100 y que imprima cuantos impares hay.

ESTUDIO PREVIO

Los números impares van de 2 en 2 empezando en 1, hasta el 99.

Hace falta un contador que empiece en 1 y aumente de 2 en 2, para que los vaya generando e imprimiendo. La condición de salida del ciclo será cuando la variable que los genere sea mayor que 100.

Hay que utilizar otro contador para saber el número de impares que hay; aumentará de 1 en 1. Sólo debe visualizar su valor cuando termine de imprimir todos los impares; irá fuera del ciclo.

VARIABLES

c = contador para generar los números impares.
son = contador del número de impares.

PSEUDOCODIGO

```
Borrar pantalla
c = 1
son = 0
Hacer mientras c < 100
     imprime c
     c = c + 2
     son = son + 1
```

```
fin del hacer
imprime "El número de impares:"
imprime son
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos los contadores.

Comprueba la condición ¿ 1 < 100 ? se cumple, entra en el ciclo. Imprime el valor de *c* e incrementa ambos contadores, *c* para generar un nuevo número y *son*, ya que acaba de imprimir un impar.

Comprueba su condición de entrada ¿ 3 < 100 ? sí, vuelve a entrar en el ciclo...

Cuando c=99 verifica la condición y entra de nuevo: imprime 99, aumenta los contadores. Vuelve a comprobar una vez más la condición de entrada ¿ 101 < 100 ? no, sale fuera del ciclo e imprime el comentario "El número de impares:" y el valor de la variable *son*. Termina el programa.

Ejercicio 6

Hacer un programa que imprima los números impares desde el 100 hasta la unidad y que calcule su suma.

ESTUDIO PREVIO

Dentro de un bucle generamos los números empezando en el 99, por ser el primer número impar a imprimir. Nos apoyamos en un contador decreciente, de 2 en 2, para generar el anterior número impar. A medida que los generamos realizamos la suma mediante un sumador.

Fuera del bucle imprimiremos el valor de la suma.

VARIABLES

c = contador para generar los números impares.
suma = acumulador que almacena la suma de los números impares.

PSEUDOCODIGO

```
Borrar pantalla
c = 99
suma = 0
```

```
Hacer mientras c >= 1
     imprime c
     suma = suma + c
     c = c - 2
fin del hacer
imprime "La suma de los números impares es: "
imprime suma
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos los variables.

Comprueba la condición ¿ 99 > = 1 ? se cumple, entra en el ciclo. Imprime el valor de *c*. Guarda el valor del número en *suma* y genera un nuevo número impar decrementando en dos unidades la variable generadora. $c = c - 2$.

Comprueba su condición de entrada ¿ 97 > = 1 ? sí, vuelve a entrar en el ciclo...

Cuando c = 1 verifica la condición y entra de nuevo: imprime el 1. Realiza la suma y genera el número −1. La condición deja de cumplirse, sale del bucle e imprime el comentario "La suma de los números impares es:" y su valor que está almacenado en *suma*.

Termina el programa.

Si hubiéramos puesto:

```
c = c - 2
suma = suma + c
```

no realizaría la suma del número 99.

Lo que sí se puede hacer es intercambiar las órdenes de sumar e imprimir:

```
suma = suma + c
imprime c
```

porque el valor de la variable *c* es el mismo en ambas líneas del programa.

Ejercicio 7

Hacer un pseudocódigo que imprima todos los números naturales que hay desde la unidad hasta un número que introducimos por teclado.

ESTUDIO PREVIO

Hay que introducir un número por teclado, para lo cual necesitamos una variable.
Debemos empezar a imprimir los números desde el 1 hasta dicho número.
El proceso de la introducción del número debe ser:

— Antes del ciclo, para saber hasta qué número debemos imprimir.
— Fuera del ciclo, para que sólo lo pida una vez. Si estuviera dentro de él, cada vez que lo realizara lo estaría pidiendo.

VARIABLES

n = variable para introducir el número.
i = contador para generar e imprimir los números.

PSEUDOCODIGO

```
Borrar pantalla
i=0
n=0
imprime "Introduce un número:"
introduce n
Hacer mientras i < n
     i = i + 1
     imprime i
fin del hacer
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos las variables.

Aparece en pantalla "Introduce un número:", espera a que lo hagamos y pulsemos la tecla <Enter> o <Return>.

Cuando introduzcamos el número puede suceder que sea:

— Cero o negativo: no entra en el ciclo. No hará nada.
— Mayor que cero: entra en él. Aumenta el contador e imprime su valor, pues deseamos imprimir desde el uno y está inicializada a cero.

Vuelve a comprobar la condición...

Ejercicio 8

Introducir tantas frases por teclado como deseemos y contarlas.

ESTUDIO PREVIO

Las frases están formadas por letras; por tanto, se necesita una variable alfanumérica. Para inicializar este tipo de variables se puede hacer de dos formas:

— Asignando a la variable el valor deseado entre comillas; ejemplo: a="hola".
— Reservando para la variable el mayor número de espacios que va a utilizar; ejemplo: b=espacios(10) o c = " ".

El proceso de introducir las frases es repetitivo, tantas como queramos; por tanto, deberá ir dentro del bucle.

Para indicar al ordenador que se desea seguir o no introduciendo más frases, la variable alfanumérica tendrá los valores S de sí o N de no.

A la hora de introducir algo por teclado, siempre que se pueda, es aconsejable hacerlo en el menor número posible de caracteres; por eso le indicamos al ordenador S o N en lugar de SI o NO.

VARIABLES

res = para preguntar si queremos introducir más frases.
frase = almacena la frase.
c = contador del número de frases.

PSEUDOCODIGO

```
Borrar pantalla
res = "S"
frase = espacios(30)
c=0
Hacer mientras res = "S"
     imprime "Frase:"
     introduce frase
     c = c + 1
     imprime "Deseas introducir más frases (S/N)"
     introduce res
fin del hacer
imprime "El número de frases introducidas son:"
imprime c
fin del programa.
```

COMPROBACION

Inicializamos *res="S"* para que nos deje entrar en el ciclo y *frase=espacios(30)*.

Nos pide una frase, aumenta el contador.

Espera a que le digamos si deseamos introducir más frases. Si ponemos S, vuelve a pedirnos otra frase; con otro carácter, aunque sea *s*, sale del ciclo por estar controlando la S, *hacer mientras res="S"*. Para que tenga en cuenta la *s* hay que utilizar el operador lógico OR de la siguiente manera:

hacer mientras res="S" OR res="s".

Imprime el comentario y el valor de la variable *c*.

Ejercicio 9

Hacer un pseudocódigo que sólo nos permita introducir S o N.

ESTUDIO PREVIO

El teclado puede estar en minúsculas; en ese caso, debemos pasarlo a mayúsculas. La expresión:

Convertir_mayusculas(variable)

el contenido de la variable, que está entre paréntesis, lo convierte a mayúsculas.

VARIABLE

res = introducir la S o la N.

PSEUDOCODIGO

```
Borrar pantalla
res = " "
Hacer mientras res <> "S" AND res <> "N"
     imprime "Introduce S o N"
     introduce res
     res = convertir_mayusculas(res)
fin del hacer
fin del programa.
```

COMPROBACION

Inicializamos *res* a blanco para que nos permita entrar en el ciclo, pues lo tiene que estar haciendo mientras sea distinto de *S* y de *N*, para lo cual utilizamos el operador lógico *AND* El contenido de *res* lo transforma en mayúsculas y lo vuelve a almacenar de nuevo en *res*.

RESUMEN

Bucle o ciclo: todo proceso que se repite un número de veces dentro de un programa.

HACER mientras condición
instrucción1
FIN del HACER
instrucción2

Constante: toda variable que no se modifica durante la ejecución del programa:

Nombre-variable = valor constante

Contador: variable que sirve para contar.

Variable = variable ± constante

Sumador o acumulador: variable que guarda el valor de una suma donde el sumador no es constante:

Sumador = sumador ± variable

donde *variable* es la cantidad a sumar.

Inicializar variables:

Variable = valor inicial de la variable

Instrucciones de entrada: para tomar datos del exterior.

Introducir nombre-variable

Instrucciones de salida: visualizar comentarios o textos, variables y expresiones.

— Visualizar comentarios: *imprime "comentario"*; el comentario deberá ir entre comillas.
— Visualizar variables: *imprime variable*, el nombre de la variable sin comillas.
— Visualizar expresiones: *imprime expresión*, la expresión también sin comillas.

Igualación de variables: almacenar el contenido de una variable en otra variable del mismo tipo:

Nombre-variable-1 = nombre-variable-2

Borrar pantalla: *borrar pantalla.*

Convertir letras minúsculas a mayúsculas:

Variable = convertir_mayusculas(variable)

Potencias de números:

Numero = numero exponente

EJERCICIOS PROPUESTOS

1. Hacer un programa que imprima y cuente los múltiplos de 3 que hay entre 1 y 100.
2. Elaborar un programa que calcule la suma de los números impares comprendidos entre el 0 y el 100.
3. Imprimir un programa que imprima y cuente los múltiplos de 5 que hay entre 1 y 500.
4. Imprimir, contar y sumar los múltiplos de 2 hasta un número que introducimos por teclado.
5. Imprimir, contar y sumar los múltiplos de 10 hasta un cierto número que introduciremos por teclado.
6. Introducir un número por teclado menor de 500. Imprimir, sumar y contar los números que están separados entre sí 8 posiciones, desde ese número al 500.
7. Calcular la suma de los cuadrados de los números pares entre el 0 y el 100.
8. Introducir un número por teclado menor de 100. Imprimir la suma de los cuadrados de los números que están separados entre sí 4 posiciones.

CAPITULO

Tomas de decisión

3.1. INTRODUCCION

Los programas, para un mejor funcionamiento y poder realizar un número mayor de tareas, deben permitir:

a) Ejecutar una serie de sentencias o instrucciones.
b) Poder repetir una serie de sentencias hasta que se cumpla o deje de cumplirse una serie de condiciones.
c) Emplear acciones alternativas para poder elegir una de ellas cuando la situación lo requiera.

Las dos primeras se han tratado en todos los ejercicios desarrollados hasta ahora.

La tercera hace los programas más inteligentes y aumenta enormemente la utilidad del ordenador.

Las **instrucciones condicionales** o **tomas de decisión** permiten realizar acciones alternativas; por tanto, la ejecución de una línea o grupos de líneas del programa depende de si cumplen o no una o varias condiciones.

3.2. TOMAS DE DECISION

Para preguntar se utiliza la instrucción o sentencia IF, si.

La contestación sólo puede ser verdadero o falso, es decir, sí o no.

Ejemplo: Si pasas por el kiosco, cómprame el periódico.
Si llueve, coge el paraguas.

La realización de la acción está supeditada a que se cumpla la condición.

Formato de las tomas de decisión:

a) **Instrucción IF**

Sintaxis:

IF condición
    instrucciones
FIN del IF

Ejemplo:

```
if res = "S"
if fi <> 25
if contador = suma
```

donde res, fi, contador y suma son nombres de variables.

El ordenador primero examina la condición. Pueden suceder dos cosas:

- La cumple: realiza todas las instrucciones que hay dentro del IF, luego continúa ejecutando las que están fuera del IF.
- No la cumple: no entra en el IF. Sólo realiza las instrucciones siguientes al IF.

Es decir, las instrucciones del IF sólo las realiza cuando cumple la condición. Las instrucciones que están fuera las realiza siempre, se cumpla o no la condición.

Se puede poner más de una condición, siempre y cuando estén unidas por los operadores lógicos (OR, AND y NOT), que funcionan igual que en el ciclo HACER.

IF condición-1 operador-lógico condición-2 ...
    instrucciones
FIN del IF

Ejemplo:

```
if res = "S" OR res = "s"
if res = "S" AND fi <> 25
if contador NOT = suma
    es igual a if contador <> suma
```

b) **Instrucción IF – ELSE**

A menudo necesitamos realizar dos procesos completamente distintos, dependiendo de si cumple o no la/s condición/es de entrada del IF.

Ejemplo: Si hace frío, ponte el abrigo; en caso contrario, ven en camisa.
Si te tocan las quinielas, vete de fiesta; si no, vuelve a jugar otra vez.

Sintaxis:

IF condición-1 operador-lógico condición-2 ...
    instrucciones-1
ELSE
    instrucciones-2
FIN del IF

Es decir:

- Cumple la/s condición/es: realiza las instrucciones que hay entre el IF y el ELSE, instrucciones-1.

- No las cumple: ejecuta las instrucciones que hay entre el ELSE y el FIN del IF, instrucciones-2.

Todo lo que se encuentre fuera del IF siempre lo va a realizar.
ELSE significa: si no, en caso contrario.

c) **Instrucciones IF – ELSE anidadas**

En el formato general para la sentencia IF, las instrucciones 1 y 2 no están limitadas a ser instrucciones imperativas; pueden ser expresiones condicionales y surge la posibilidad de usar instrucciones IF anidadas.

Sintaxis:

```
IF condición-1
        instrucciones-1
        IF condición-2
                instrucciones-2
        ELSE
                instrucciones-3
        Fin del IF
ELSE
        instrucciones-4
        IF condición-3
                instrucciones-5
        ELSE
                instrucciones-6
        Fin del IF
Fin del IF
```

NOTA:
Las instrucciones situadas dentro del condicional IF, con todas sus variantes, al igual que en el ciclo HACER, se escriben desplazadas hacia la derecha para una mayor claridad del pseudocódigo.

Ejercicio 1

Introducir un número por teclado. Que nos diga si es positivo o negativo.

ESTUDIO PREVIO

Para saber si el número es positivo o negativo, es necesaria la utilización del IF-ELSE: un número es positivo o es negativo, pero nunca puede ser las dos cosas al mismo tiempo. Al número 0 lo consideramos como positivo.

VARIABLE

num = almacena el número introducido por teclado.

PSEUDOCODIGO

```
Borrar pantalla
num = 0
imprime "Introduce un número:"
introduce num
if num >= 0
     imprime "es positivo"
else
     imprime "es negativo"
fin del if
fin del programa.
```

COMPROBACION

Inicializamos *num* y se introduce el número. Preguntamos cómo es; si es mayor o igual a cero, *if num >= 0* es positivo: en caso contrario, es negativo.

El proceso de inicializar las variables es anterior al de introducirlas por teclado. Si se realizara al revés (*introduce num*, *num = 0*), la variable *num* siempre tendría el valor cero, sin tener en cuenta el valor que hayamos introducido por teclado.

Ejercicio 2

Introducir un número por teclado. Que os diga si es par o impar.

ESTUDIO PREVIO

Para saber si es par o no hacemos lo siguiente:

*NUMERO = INT (NUMERO/2) * 2*

Al número lo dividimos por 2 (NUMERO/2) y calculamos su parte entera, INT(NUMERO/2). Al número resultante lo volvemos a multiplicar por 2, INT(NUMERO/2)*2; puede suceder:

— Resulta el mismo número del cual partimos: es par.
— No es el mismo número: no es par.

Ejemplo: sea *numero=8*. Al dividirlo por 2 (NUMERO/2), su cociente es 4. Su parte entera es 4, INT(NUMERO/2); al multiplicarla por 2 da 8, INT(NUMERO/2)*2; es el número del que partimos: obviamente el 8 es par.

Si hubiéramos partido del 9, al dividirlo por 2 da de cociente 4.5. Su parte entera es 4, al volverlo a multiplicar por 2 nos da 8, no es el número del cual habíamos partido. El 9 no es par.

También se puede hacer preguntando si el resto de la división es 0, *if(NUMERO MOD 2) = 0*, en cuyo caso es par.

En este libro se utilizará la primera forma.

VARIABLE

num = almacena el número introducido por teclado.

PSEUDOCODIGO

```
Borrar pantalla
num = 0
imprime "Introduce un número:"
introduce num
if num = int(num/2) * 2
    imprime "es par"
else
    imprime "es impar"
fin del if
fin del programa.
```

COMPROBACION

Se introduce el número. Se comprueba si es par o no mediante la condición: *if num = int(num/2) * 2*. Si se cumple, es par; en caso contrario, es impar.

Ejercicio 3

Imprimir y contar los múltiplos de 3 desde la unidad hasta un número que introduciremos por teclado.

ESTUDIO PREVIO

Deseamos imprimir los múltiplos de 3 y éstos empiezan en 3. Para saber si es múltiplo de 3, procedemos de una forma análoga al ejercicio anterior, pero esta vez dividiendo por 3.

VARIABLES

n = guarda el número introducido por teclado.
i = contador para generar los números desde el 3 hasta n.
c = contador de los múltiplos de 3.

PSEUDOCODIGO

```
Borrar pantalla
i = 3
n = 0
c = 0
imprime "Número:"
introduce n
```

```
Hacer mientras i <= n
     if i = int(i/3) * 3
          imprime i
          c = c + 1
     fin del if
     i = i + 1
fin del hacer
imprime "El número de múltiplos de 3 son:"
imprime c
fin del programa.
```

COMPROBACION

Inicializamos el contador a 3 por ser el mismo el primer múltiplo de 3.

Introducimos un número. Examina la condición de entrada del ciclo, *hacer mientras* $i<=n$. Puede suceder:

— Si el número es más pequeño que 3, no entra en el ciclo, imprimirá cero.
— Si es mayor, entra en el bucle. Comprueba si el número es múltiplo de tres, *if* $i = = int(i/3)*3$:

- Lo es: lo imprime y aumenta el contador de los múltiplos.
- No es múltiplo: no lo hace.

En ambos casos, sea múltiplo o no, tiene que pasar al siguiente número, para lo cual aumenta el contador. Comprueba la condición de entrada en el ciclo...

Cuando llegue al número que hemos introducido, sigue cumpliéndose la condición de entrada del bucle, $i<=n$, entra, comprueba si es múltiplo y sale de él.

Visualiza el comentario "El número de múltiplos de 3 son:" e imprime el valor de la variable *c* y finaliza.

Ejercicio 4

Hacer un pseudocódigo que imprima los números del 1 al 100. Que calcule la suma de todos los números pares por un lado, y por otro, la de todos los impares.

ESTUDIO PREVIO

Necesitamos una variable contador que vaya tomando los valores de 1 a 100. Al mismo tiempo comprobamos si ese número es par o no.

Para saber si el número es par, procederemos de una forma análoga a los ejercicios anteriores.

Si el número es par, se acumula su valor en el sumador que almacene la suma de los pares; en caso contrario, es impar y guarda su valor en el sumador de los impares.

VARIABLES

i = contador de 1 a 100.
sumapar = sumador de los números pares.
sumaimp = sumador de los números impares.

PSEUDOCODIGO

```
Borrar pantalla
i = 1
sumapar = 0
sumaimp = 0
Hacer mientras i < 101
     if i = int(i/2) * 2
          sumapar = sumapar + i
     else
          sumaimp = sumaimp + i
     fin del if
     i = i + 1
fin del hacer
imprime "La suma de los pares es:"
imprime sumapar
imprime "La suma de los impares es:"
imprime sumaimp
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos las variables.
Comprueba la condición del ciclo: la cumple.
Pregunta si el número es par o impar, *if i=int(i/2)*2:*

— Impar: su valor lo acumula en *sumaimp*. Sale del IF y aumenta el contador. Vuelve a verificar la condición de entrada del bucle.
— Par: el valor del número lo suma a *sumapar*. Aumenta el contador y comprueba la condición de entrada en el ciclo.

Cuando salga del bucle imprime el comentario "La suma de los pares es:" y su valor, que está almacenado en la variable *sumapar*. De igual forma con los impares y la variable *sumaimp*.

Ejercicio 5

Imprimir y contar los números que son múltiplos de 2 o de 3 que hay entre 1 y 100.

ESTUDIO PREVIO

Los números deben ser múltiplos de 2 o de 3. En la pregunta tiene que haber dos condiciones unidas por el operador lógico OR, pues sólo es necesario que cumpla una de ellas.

VARIABLES

i = contador generador de los números de 1 a 100.
c = contador de los números que cumplen una o las dos condiciones.

PSEUDOCODIGO

```
i = 1
c = 0
Hacer mientras i<101
     if i = int(i/2)*2 OR i = int(i/3)*3
          c = c+1
          imprime i
     fin del if
     i = i+1
fin del hacer
borrar pantalla
imprime "El número de múltiplos es de:",c
fin del programa.
```

COMPROBACION

Inicializamos las variables.

Al ser *i<101* entra en el ciclo y comprueba si el valor guardado en la variable *i* es múltiplo de 2 o de 3. Si cumple una de las dos condiciones, aumenta el contador de los múltiplos e imprime el número.

Se cumpla o no la condición, genera un nuevo número.

Verifica de nuevo la condición de entrada del bucle.

Cuando salga borra pantalla e imprime el comentario y el valor de la variable *c*.

Ejercicio 6

Imprimir y contar los números que son múltiplos de 2 y de 3 que hay del 1 al 100.

ESTUDIO PREVIO

Los números deben ser al mismo tiempo múltiplos de 2 y de 3. Hay que utilizar dos condiciones unidas por el operador lógico AND: es necesario que se cumplan las dos condiciones al unísono.

VARIABLES

i = contador generador de los números de 1 a 100.
c = contador de los números que cumplen las dos condiciones.

PSEUDOCODIGO

```
i = 1
c = 0
Hacer mientras i<101
     if i = int(i/2)*2 AND i = int(i/3)*3
          c = c+1
          imprime i
     fin del if
     i = i+1
fin del hacer
borrar pantalla
imprime "Los múltiplos son:",c
fin del programa.
```

COMPROBACION

Inicializamos las variables.

Al ser *i<101* entra en el ciclo y comprueba si el valor guardado en la variable *i* es múltiplo de 2 y de 3.

Si cumple una de las dos condiciones, no realiza las instrucciones del IF.

Para entrar en el IF se deben cumplir ambas condiciones, por estar relacionadas con el operador lógico AND.

Cuando salga del bucle imprime el valor de la variable *c*.

3.3. AUXILIAR

Un **auxiliar** es una variable que se utiliza para almacenar el contenido de otra variable.

Sintaxis:

aux = variable

Mediante la asignación anterior el contenido de *variable* se almacena en *aux*.
El auxiliar debe ser del mismo tipo que la variable: numérico o alfanumérico.
Los auxiliares se utilizan para:

— Conservar un valor determinado de la variable, porque ésta cambia de valor con la ejecución del programa.
— Intercambiar el contenido de dos variables.

Ejercicio 7

Hacer un pseudocódigo que imprima el mayor y el menor de una serie de cinco números que vamos introduciendo por teclado.

ESTUDIO PREVIO

Se necesita una variable para saber cuál es el mayor de los números introducidos y otra para el más pequeño, además de otra para contar hasta cinco.

Cuando introduzcamos el primer número, éste debe ser el mayor. La variable que almacene el máximo debe contener el valor de dicho número, para lo cual debe estar inicializada al valor más pequeño de todos los números que podamos introducir.

El primer número, además de ser el máximo, debe ser también el mínimo; por tanto, la variable que guarde el valor del mínimo estará inicializada al valor mayor de todos los números que podamos introducir (supongamos que es 99.999).

VARIABLES

con = contador para introducir cinco números.
n = introducir los números.
maximo = guardar el mayor de los números introducidos.
minimo = guardar el menor de los números introducidos.

PSEUDOCODIGO

```
Borrar pantalla
con = 0
n = 0
maximo = 0
minimo = 99999
Hacer mientras con <= 5
     imprime "Número:"
     introduce n
     if n > maximo
          maximo = n
```

```
    fin del if
    if n < minimo
        minimo = n
    fin del if
    con = con + 1
fin del hacer
imprime "El mayor de los números es: "
imprime maximo
imprime "El menor de los números es: "
imprime minimo
fin del programa.
```

COMPROBACION

Inicializamos las variables para entrar en el ciclo y poder comenzar a introducir los números.

Compara el número que acabamos de introducir con el *maximo;* al ser mayor dicho número, el *maximo* pasará a ser *n,* mediante la asignación *maximo=n,* para el primer número. El resto de los números pueden ser o no mayores que *maximo;* si alguno es mayor, su valor se almacena en *maximo.*

De igual forma procederemos con el *minimo.*

Aumentamos el contador de los números introducidos.

Cuando se hayan introducido los cinco números salimos del ciclo e imprimimos los valores del mayor y del más pequeño, que están almacenados en las variables respectivas *maximo* y *minimo.*

Ejercicio 8

Introducir dos números por teclado. Imprimir los números naturales que hay entre ambos números empezando por el más pequeño, contar cuántos hay y cuántos de ellos son pares. Calcular la suma de los impares.

ESTUDIO PREVIO

Supongamos que introducimos por teclado los números 4 y 8, no importa el orden; los debemos imprimir siempre empezando por el número 4 hasta el 8.

La impresión de los números la realizamos mediante un ciclo con dos variables que contengan al menor y al mayor de los números, y un contador que vaya desde el número inferior al superior.

El problema es saber cuál es el menor.

Si el primero es el más pequeño, no hay dificultad, pero si el mayor es el número introducido en primer lugar, debemos intercambiarlos mediante una variable auxiliar.

Supongamos que tenemos dos vasos, uno con agua y el otro con un refresco, y queremos intercambiar el contenido de los vasos. Para ello debemos coger un vaso auxiliar vacío. En el vaso auxiliar vertemos el agua, por ejemplo. En el vaso vacío, donde antes estaba el

agua, se vierte el refresco. El vaso que antes contenía el refresco está vacío y echamos en él el agua que estaba en el vaso auxiliar. El vaso auxiliar queda vacío; hemos intercambiado el contenido de ambos vasos.

Supongamos que tenemos las variables *menor* y *mayor* con los siguientes valores: *menor=8* y *mayor=4*, y deseamos intercambiar el contenido de las variables.

Nos ayudamos de una variable auxiliar, *aux*, *aux=0*. La situación inicial es:

menor = 8 *mayor = 4* *aux =0.*

Volcamos el contenido de una de las variables en el auxiliar, *aux=menor*. Tenemos la siguiente situación:

menor = 8 *mayor = 4* *aux = 8*

El contenido de *mayor* lo volcamos en *menor*. *menor=mayor*. La situación es:

menor = 4 *mayor = 4* *aux = 8*

Por último, asignamos a *mayor* el contenido del auxiliar. *mayor=aux*. Las variables contienen los valores:

menor = 4 *mayor = 8* *aux = 8*

Es decir, el proceso de asignación de variables es el siguiente:

aux=menor *menor=mayor* *mayor=aux*

Una vez que hemos colocado el menor en la primera variable, operación que debemos realizar antes de entrar en el ciclo, generamos los números y efectuamos las operaciones pedidas con ayuda de la instrucción IF-ELSE para saber si los números son pares o impares.

VARIABLES

num1 = recoge el primer número introducido por teclado.
num2 = recoge el segundo número introducido por teclado.
aux = auxiliar para realizar el intercambio de los números si num2 es más pequeño que num1.
son = contador de los números que hay entre num1 y num2.
pares = contador de los números pares.
sumaimpa = sumador de los números impares.

PSEUDOCODIGO

```
num1 = 0
num2 = 0
aux = 0
son = 0
pares = 0
sumaimpa = 0
borrar pantalla
imprime "Número: "
introduce num1
```

```
imprime "Número: "
introduce num2
if num1 > num2
     aux = num1
     num1 = num2
     num2 = aux
fin del if
Hacer mientras num1 >= num2
     imprime num1
     son = son + 1
     if num1 = int(num1/2)*2
          pares = pares + 1
     else
          sumaimpa = sumaimpa + num1
     fin del if
     num1 = num1 + 1
fin del hacer
imprime "Números visualizados: "
imprime son
imprime "Pares hay:
imprime pares
imprime "La suma de los impares es: "
imprime sumaimpa
fin del programa.
```

COMPROBACION

Inicializadas todas las variables que vamos a utilizar en el programa, borramos pantalla e introducimos los números por teclado.

Para saber cuál es el menor preguntamos, *if num1 > num2:*

— Es verdad: los intercambiamos con la ayuda del auxiliar. El más pequeño está almacenado en num1 y el mayor en num2.
— No es cierto: no debe realizar nada, ya que el menor está en num1 y el mayor en num2, porque se han introducido de esta forma por teclado.

Si los dos números fueran iguales, no cumple la condición del IF ni tampoco la del ciclo, no hace nada e indica que no ha impreso ningún número, que no hay ningún par y que la suma es cero.

Si los números son diferentes, sí entra en el ciclo. Imprime el número menor, incrementa en una unidad el contador de los números impresos.

Comprueba si el número es par o impar, mediante la sentencia: *if num1 = int(num1/2)*2.*

— Par: incrementa el contador de los pares.
— [illegible] el valor de num1 en el sumador sumaimpa.

Tanto sea par como impar, genera el número siguiente, *num1= num1 + 1*. Esta variable, que contiene el número menor, es la que tiene que incrementarse hasta valer el contenido de num2.

Cuando salga del ciclo imprime los comentarios y los valores de las variables.

RESUMEN

Para tomar decisiones, preguntar. con la instrucción IF. La respuesta sólo puede ser sí o no.

Formatos:

Instrucción IF

IF condición
    instrucciones
FIN del IF

Instrucción IF con más de una condición:

IF condición-1 operador-lógico condición-2 ...
    instrucciones
FIN del IF

Instrucción IF – ELSE

IF condición-1 operador-lógico condición-2 ...
    instrucciones-1
ELSE
    instrucciones-2
FIN del IF

Instrucciones IF – ELSE anidadas

```
IF condición-1
        instrucciones-1
    IF condición-2
        instrucciones-2
    ELSE
        instrucciones-3
    Fin del IF
ELSE
    instrucciones-4
    IF condición-3
        instrucciones-5
    ELSE
        instrucciones-6
    Fin del IF
Fin del IF
```

Sabemos averiguar si **un número es divisible por otro número.**

Por ejemplo, para saber si un número es par lo será si cumple la igualdad:

*NUMERO = INT (NUMERO/2) * 2*

Un **auxiliar** es una variable que se utiliza para almacenar el contenido de otra variable.

Sintaxis:

aux = variable

el contenido de la variable, *variable*, se almacena en la variable auxiliar, *aux*.

El auxiliar debe ser del mismo tipo que la variable: numérico o alfanumérico.

Los auxiliares se utilizan para:

— Conservar un valor determinado de alguna variable (la variable está cambiando de valor con la ejecución del programa).
— Intercambiar el contenido de dos variables.

EJERCICIOS PROPUESTOS

1. Introducir una serie de números. Decir si esos números son mayores o menores de 10.
2. Introducir tantos números como queramos. Decir si son mayores o menores que un número que previamente introduciremos por teclado.
3. Imprimir y contar los números que hay del 1 al 100, excepto los múltiplos de 2.
4. Imprimir y sumar los números desde el cero hasta un número determinado, excepto los múltiplos de 5. Imprimir el valor de la suma de los múltiplos de 5.
5. Imprimir, sumar y contar los números, que son a la vez múltiplos de 2 y de 3, que hay entre la unidad y un determinado número.
6. Introducir una serie de números. Sumar los múltiplos de 5, cuál es el mayor y cuántos números se han introducido.
7. Introducir dos números por teclado de tal forma que el segundo sea mayor que el primero. A partir del primero, imprimir los números separados entre sí 7 unidades, contarlos. De éstos, cuántos son pares y cuánto vale la suma de los impares.
8. Introducir una cantidad por teclado. Si es menor de 500, sumarle el 50 por 100; si es mayor o igual a 500 pero menor de 1 000, sumarle el 7 por 100; si es mayor o igual a 1 000 y menor o igual a 5 000, sumarle el 15 por 100, y si es mayor de 5 000, restarle el 5 por 100.

CAPITULO

Bucles anidados. Subprogramas. Switch

4.1. BUCLES ANIDADOS

Una determinada situación se puede repetir si previamente se han repetido otras más pequeñas comprendidas en aquella.

El nacimiento de un nuevo día está supeditado al paso previo de 24 horas. Una hora tiene 60 minutos. Un minuto, a su vez, está formado por 60 segundos.

El paso de una unidad de tiempo a otra superior implica que se ha completado el ciclo de la unidad de tiempo inmediatamente inferior a ella.

Estas situaciones no sólo son propias del tiempo; hay otras muchas que presentan características semejantes.

Cuando leemos una página de un libro no pasamos a la página siguiente hasta no haber leído la página actual. Las páginas están formadas por un conjunto de líneas y éstas por un conjunto variable de caracteres o símbolos.

Un *ciclo* puede estar formado por otro u otros ciclos.

Al igual que sucedía con la instrucción IF, que dentro de un IF podíamos poner todos los IF que fueran necesarios, análogamente, dentro de un bucle HACER pueden ir otro u otros bucles HACER, de tal forma que el último de todos, el situado más interiormente, es el primero en cerrarlo, en acabar. El primero de todos, situado más fuera, es el último en terminar.

Sintaxis:

Hacer mientras condición-1
.....
.....
Hacer mientras condición-2
.....
.....
Hacer mientras condición-3
.....
.....
Fin del hacer
.....
.....
Fin del hacer
.....
.....
Fin del hacer

donde los puntos suspensivos serán las distintas instrucciones a realizar.

Es semejante a lo que hacemos cuando dentro de una caja introducimos otras cajas de menor tamaño. Al cerrarlas, la primera en ponerle la tapa será la situada más internamente; a continuación, la situada más dentro. Por último, cerramos la caja que contiene a las demás, la más grande.

Ejercicio 1

Imprimir diez veces la serie de números del 1 al 10.

ESTUDIO PREVIO

La secuencia de números del 1 al 10 se realiza mediante un ciclo que vaya de 1 a 10 y un contador para generarlos.

Esta secuencia debe realizarse diez veces. Necesitamos otro ciclo que cuente las veces que se han impreso. Este ciclo aumentará en una unidad cuando se hayan visualizado los números del 1 al 10.

El ciclo exterior controla que se imprima 10 veces la secuencia de los números. Dentro irá otro para visualizar los números: 1, 2, 3, ... 10.

VARIABLES

numero = contador para generar los números del 1 al 10.
serie = contador de las series, también irá de 1 a 10.

PSEUDOCODIGO

```
Borrar pantalla
serie = 0
hacer mientras serie <= 10
     numero = 1
     hacer mientras numero <= 10
          imprime numero
          numero = numero + 1
     fin del hacer
     serie = serie + 1
fin del hacer
fin del programa.
```

COMPROBACION

Fuera del bucle, *hacer mientras serie <= 10,* se inicializa solamente la variable serie, y dentro de éste, pero fuera del otro, se inicializa *numero,* porque cada vez que empezamos a imprimir una nueva serie los números deben empezar desde el 1; de esta forma, sólo hace falta inicializarla una sola vez.

Si se hubiera inicializado *numero* antes del ciclo de las series, habría que reinicializarla de nuevo cuando comience la siguiente serie, bien antes o después del otro ciclo, pero siempre fuera del ciclo interior.

Dentro del ciclo, *hacer mientras numero <= 10,* se imprimen los números y se genera uno nuevo.

Completada la serie se incrementa el contador *serie;* vuelve a reinicializarse el contador *numero* para que la nueva serie comience desde el 1.

Ejercicio 2

Imprimir, contar y sumar los múltiplos de 2 que hay entre una serie de números, tal que el segundo sea mayor o igual que el primero.

ESTUDIO PREVIO

Hay que introducir dos números por teclado y obligatoriamente el segundo debe ser mayor o igual que el primero. Por tanto, hará falta un ciclo para introducir el segundo número mientras no se cumpla la condición.

Una vez introducidos los números, a partir del primero se irán generando los números y al mismo tiempo se comprueba si son múltiplos de 2.

Todo esto hay que repetirlo una serie de veces; mientras lo deseemos, se utiliza otro ciclo que comprenda todo lo anterior.

VARIABLES

res = introducir la serie de números.
num1 = introducir el primer número e irá aumentando hasta num2.
num2 = introducir el segundo número.
c = contador de los múltiplos de 2.
sum = sumador de los múltiplos de 2.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     c = 0
     sum = 0
     num1 = 0
     num2 = -999
     imprime "Número"
     introduce num1
     imprime "Número mayor que el anterior"
     hacer mientras num1 >= num2
          introduce num2
     fin del hacer
     num1 = num1 + 1
     hacer mientras num1 <= num2-1
          if num1 = int(num1/2)*2
               imprime num1
               c = c + 1
               sum = sum + num1
          fin del if
          num1 = num1 + 1
     fin del hacer
     imprime "Número de múltiplos de 2:",c
     imprime "Su suma es:",sum
     res = espacios(1)
     hacer mientras res <> "S" AND res <> "N"
          imprime "Otra serie de números (S/N):"
          introduce res
          res = convertir_mayusculas(res)
     fin del hacer
fin del hacer
fin del programa.
```

COMPROBACION

Dentro del bucle, *hacer mientras res="S"*, se inicializan las variables para que si se repite el proceso, empiecen con los valores iniciales. Para *num2=-999* en el caso de que se introduzcan valores negativos en num1 y nos permita entrar en el bucle de introducir por teclado el segundo número.

Se introduce el primer número. El segundo debe ser mayor; se controla por el ciclo *hacer mientras num1>=num2.*

Cuando sea mayor debe empezar a imprimir los números pares a partir del primer número introducido, para lo cual se incrementa *num1*. El proceso de generar los números hasta el número anterior al segundo, *num2*, es repetitivo: *hacer mientras num1 <=num2-1.*

Dentro del bucle se comprueba si el número es par; en caso se serlo, se imprime y se incrementan los contadores.

Al terminar de generar todos los números imprime los valores de las variables y pregunta si se desea introducir otra serie de números.

Ejercicio 3

Hacer un pseudocódigo que cuente las veces que aparece una determinada letra en una frase que introduciremos por teclado.

Repetir el proceso tantas veces como queramos.

ESTUDIO PREVIO

El cálculo del número de caracteres de una frase lo expresamos de la forma:

hallar-longitud(variable)

variable almacena la frase cuyo número de caracteres queremos calcular.

Cada lenguaje tendrá una instrucción característica que lo realiza.

Para conocer el número de veces que aparece una letra determinada es necesario comparar dicho carácter con todos y cada uno de la frase.

El carácter que hay en cada posición lo calculamos de la forma:

caracter(variable,posición,incremento)

donde:

— **Variable:** almacena la frase.
— **Posición:** en la cual estamos dentro de la frase: irá desde 1 hasta el valor de *hallar-longitud(variable).*
— **Incremento:** el número de caracteres que deseamos leer cada vez en la frase. Para nuestro caso valdrá 1.

VARIABLES

frase = almacena la frase.
longitud = guardar la longitud de la frase.
letra = recoge la letra a buscar.
i = contador para recorrer frase desde el carácter 1 hasta longitud.
a = contador del número de veces que aparece la letra buscada.
res = para continuar o no introduciendo frases.

PSEUDOCODIGO

```
frase = espacios(30)
letra = espacios(1)
longitud = 0
a = 0
res = "S"
Hacer mientras res = "S"
     Borrar pantalla
     imprime "Introduce una frase:"
     introduce frase
     longitud = hallar-longitud(frase)
     i = 1
     imprime "Letra a buscar:"
     introduce letra
     Hacer mientras i <= longitud
          if letra = caracter(frase,i,1)
               a = a + 1
          endif
          i = i + 1
     fin del hacer
     borrar pantalla
     imprime "El número de veces que aparece la letra "
     imprime letra
     imprime "en la frase "
     imprime frase
     imprime " es de "
     imprime a
     res = espacios(1)
     Hacer mientras res <> "S" AND res <>"N"
          imprime "Deseas introducir más frases (S/N):"
          introduce res
          res = convertir_mayusculas(res)
     fin del hacer
fin del hacer
fin del programa.
```

COMPROBACION

Inicializamos las variables.

En el primer ciclo, *hacer mientras res="S"*, se introduce la frase, se calcula su longitud y se le indica la letra a buscar.

En el segundo ciclo, *hacer mientras i <= longitud*, recorre toda la frase comprobando si el carácter en el que estamos es igual al carácter buscado, *if letra=caracter(frase,i,1)*.

Al salir nos imprime la frase que introdujimos, así como el número de veces que aparece el carácter buscado.

Pregunta si deseamos introducir más frases o acabar el programa.

Ejercicio 4

Hacer un pseudocódigo que simule el funcionamiento de un reloj digital y que permita ponerlo en hora.

ESTUDIO PREVIO

Necesitamos tres ciclos: para las horas, los minutos y los segundos; uno dentro del otro.

El ciclo más pequeño será el que tiene que ir más dentro; el de los segundos también debe ser el primero en acabar. Cuando termine aumentarán los minutos; los segundos se inicializan a cero.

h	m	s
08	24	59

un segundo después

h	m	s
08	25	00

Los minutos al llegar a 60 tendrán que pasar a valer cero. Habrá una hora más.

h	m	s
08	59	59

un segundo después

h	m	s
09	00	00

VARIABLES

horas = contador de las horas.
minutos = contador para los minutos.
segundos = contador de los segundos.
res = cuando llegue a las 24 horas comience un nuevo día, el reloj no se detenga.

PSEUDOCODIGO

```
horas = 0
minutos = 0
segundos = 0
res = "S"
imprime "Horas: "
introduce horas
imprime "Minutos: "
introduce minutos
imprime "Segundos: "
introduce segundos
Hacer mientras res = "S"
     Hacer mientras horas < 24
          Hacer mientras minutos < 60
               Hacer mientras segundos < 60
```

```
                imprime horas
                imprime minutos
                imprime segundos
                segundos = segundos + 1
            fin del hacer
            minutos = minutos + 1
            segundos = 0
        fin del hacer
        horas = horas + 1
        minutos = 0
    fin del hacer
    horas = 0
fin del hacer
```

COMPROBACION

El ciclo, *hacer mientras res="S"*, se utiliza para empezar un nuevo día. Es uno de los pocos casos, o el único, en que el programa no termina. El programa está realizando un bucle sin fin.

Ejercicio 5

Hallar el factorial de un número.

ESTUDIO PREVIO

El factorial de un número es el producto de los números naturales desde la unidad hasta dicho número inclusive.

Los números negativos no tienen factorial.

0! = 1
1! = 1
2! = 2 * 1
3! = 3 * 2 * 1

7! = 7 * 6 * 5 * 4 * 3 * 2 * 1

n! = n * (n-1) * (n-2) * ... * 3 * 2 * 1

El número debe ir disminuyendo en una unidad hasta llegar a 2 (cualquier número multiplicado por 1 es dicho número) e ir multiplicándolo por la variable que recoja el valor de la multiplicación.

VARIABLES

numero = introducir el número del cual queremos hallar su factorial.
factorial = guarda el resultado de la multiplicación.

PSEUDOCODIGO

```
res = "S"
Hacer mientras res = "S"
     Borrar pantalla
     factorial = 1
     imprime "Número:"
     introduce numero
     if numero < 0
          imprime "No tiene factorial"
     else
          Hacer mientras numero > 1
               factorial = factorial * numero
               numero = numero - 1
          fin del hacer
          imprime "Su factorial es:"
          imprime factorial
     fin del if
     res = " "
     Hacer mientras res<>"S" AND res<>"N"
          imprime "Más factoriales(S/N):"
          introduce res
          res = convertir_mayusculas(res)
     fin del hacer
fin del hacer
fin del programa.
```

COMPROBACION

Inicializamos factorial a la unidad, pues esta variable va a recoger el producto de la multiplicación. Si se inicializara a cero, la multiplicación siempre sería cero.

Está dentro del ciclo, *hacer mientras res="S"*, pues si deseamos calcular más factoriales debemos volver a inicializarla.

En el ciclo, se introduce el número cuyo factorial queremos calcular. Si el número es:

- Negativo: entra por el primer IF, *if numero<0*, e indica que no tiene factorial.
- Cero: no entra en el ciclo, *hacer mientras numero>1*, pero imprime su factorial que es la unidad.
- Positivo: lo calcula. Multiplica dicho número por factorial y decrece el número en una unidad hasta que valga la unidad.

Sea como sea el número, pregunta si deseamos calcular más.

También se podía hacer recorriendo los números desde la unidad hasta el número introducido, pero se necesita una variable más.

4.2. SUBPROGRAMAS O SUBRUTINAS

Los pseudocódigos realizados hasta ahora están todos ellos descritos en un único programa, llamado *programa principal.*

En el Capítulo 1, cuando hablamos de las características de los algoritmos mencionábamos la *modularidad.* que consiste en estructurar el programa principal en módulos más pequeños llamados *subprogramas* o *subrutinas.*

Un **subprograma** es un conjunto de sentencias de un programa que realizan una determinada tarea y que pueden ser ejecutadas desde más de un punto del programa principal.

Cuando termine de ejecutarse el subprograma continúan procesándose las instrucciones siguientes del programa.

Una **subrutina** es como un boomerang: va, realiza lo que tenga que hacer y regresa al punto de partida.

Su estructura básicamente es la de cualquier programa, con las diferencias lógicas en la parte inicial y final.

A su vez, un subprograma puede estar compuesto por varios subprogramas.

Están descritos fuera del programa principal.

Sintaxis:

HACER nombre-subprograma

Ejemplos de llamadas a subprogramas en distintos lenguajes: DO, GOSUB, PERFORM, CALL.

Los nombres de los subprogramas los escribiremos en mayúsculas, para una mayor legibilidad del programa. A la hora de nombrarlos, deben cumplir la normativa de las variables, es decir, deben empezar con una letra y el resto de los caracteres, hasta una longitud de 8, puede ser una combinación de números, letras y el guión, pero no puede contener espacios en blanco.

El número de subprogramas que habrá dentro de cada programa será elección nuestra en función de la complejidad del ejercicio.

La función que deben cumplir los subprogramas es la de conseguir, aún más, la estructuración del programa y, por ello, facilitar la tarea en su construcción y simplificar al máximo las posibles modificaciones posteriores en el programa.

Otras de las misiones de los subprogramas es la de evitar la repetición de instrucciones dentro de un programa; estas instrucciones se escriben en un subprograma. Este subprograma es llamado, ejecutado, las veces que haga falta.

Así, las instrucciones que lo componen sólo están escritas una vez, necesitando menos cantidad de memoria para almacenar el programa.

Ejercicio 6

Calcular el factorial de un número, mediante subprogramas.

PSEUDOCODIGO

```
res = "S"
Hacer mientras res = "S"
     Borrar pantalla
     factorial = 1
     imprime "Número:"
     introduce numero
     if numero < 0
          imprime "No tiene factorial"
     else
          hacer CALCULOS
     fin del if
     hacer MAS
fin del hacer
fin del programa.
```

Estas líneas constituyen el **programa principal.**

Los subprogramas CALCULOS y MAS, descritos fuera del programa principal, son:

```
CALCULOS.
Hacer mientras numero > 1
     factorial = factorial * numero
     numero = numero - 1
fin del hacer
hacer IMPRIMIR.

MAS.
res = " "
Hacer mientras res <> "S" AND res <> "N"
     imprime "Deseas calcular más factoriales(S/N):"
     introduce res
     res = convertir_mayusculas(res)
fin del hacer.
```

El subprograma CALCULOS llama, a su vez, a otro subprograma IMPRIMIR.

```
IMPRIMIR.
imprime "Su factorial es:"
imprime factorial.
```

COMPROBACION

Hemos hecho dos subprogramas, que van a realizar una tarea concreta:

— CALCULOS:
  - calcula el factorial;
  - contiene otro subprograma IMPRIMIR: visualiza el valor del factorial.

— MAS: pregunta si deseamos calcular más factoriales.

Con los subprogramas el programa principal queda más fácil de leer y más corto.

NOTA:
De aquí en adelante, a no ser que el subprograma MAS esté desarrollado, cada vez que aparezca dicho subprograma será semejante al de este ejercicio y, por ello, hará referencia al mismo.

4.3. SWITCH O INTERRUPTOR

El **switch** es una variable que sólo puede tomar dos valores exclusivos: 0 ó 1. Por dicho motivo también se le llama *interruptor:* está encendido o apagado.

A veces, también se le llama *bandera* o *flag.*

Los switch los designamos por *sw;* generalmente se inicializan a cero, *sw = 0.*

Su valor se modifica mediante otra asignación: *sw = 1.*

Se utilizan para saber:

- Si el programa ha pasado por un determinado punto, preguntando por su estado: *if sw = 0,* entonces se imprimirá en pantalla algún tipo de comentario; en caso contrario, se visualiza otra serie de cosas o no se visualiza nada.
- Salir de un ciclo cuando el *sw* tenga un determinado valor, *hacer mientras sw=0,* cuando el *sw=1* abandona el ciclo.
- Dependiendo de su valor realizar una u otra acción: *if sw=0,* entonces el programa ejecuta una serie de acciones y si es *1* otras o no hará nada.

En definitiva, los switch permiten variar la secuencia de ejecución de un programa dependiendo del valor que posean en cada instante.

Ejercicio 7

Imprimir 10 veces, de una forma alternativa, Hola y Adiós.

ESTUDIO PREVIO

Para imprimir un comentario o una serie de números un número determinado de veces por pantalla, como sabemos, necesitamos un ciclo con un contador: para nuestro caso el contador irá de 1 a 10.

La visualización de una forma alternativa de dos comentarios diferentes la hacemos mediante un switch.

El *sw* permite saber, según sea su valor 0 ó 1, si la última palabra visualizada fue Hola o Adiós. Para ello, preguntamos por su estado, y entonces imprimirá una cosa u otra.

Necesitamos un IF-ELSE.

VARIABLES

sw = para saber si acabamos de imprimir Hola o Adiós.
i = contador de 1 a 10.

PSEUDOCODIGO

```
Borrar pantalla
sw = 0
i = 0
Hacer mientras i <= 10
     if sw=0
          imprime "Hola"
          sw = 1
     else
          imprime "Adiós"
          sw = 0
     fin del if
     i = i + 1
fin del hacer
fin del programa.
```

COMPROBACION

Borramos pantalla e inicializamos las variables; el sw, por ejemplo, a 0.
Dentro del ciclo preguntamos por su valor:

— Cero: imprime Hola. Para que la siguiente vez imprima Adiós hay que cambiar su estado de 0 a 1, *sw = 1*.
— Uno: imprime Adiós. La siguiente vez debe imprimir Hola; volvemos a cambiar su estado, ahora de 1 a 0, *sw = 0*.

Fuera del IF-ELSE, incrementamos el contador *i*, para que sólo aparezca una vez dentro del programa; de lo contrario, deberíamos ponerlo una vez en el IF y otra en el ELSE.

Ejercicio 8

Hacer un programa que calcule independientemente la suma de los pares y los impares de los números entre 1 y 1 000, utilizando un switch.

ESTUDIO PREVIO

Los impares y los pares van alternándose.
El *sw* permite saber, según el valor que tenga, si el número en que nos encontramos es impar, *sw=0*, o par, *sw=1*.

VARIABLES

par = sumador de los números pares.
impar = sumador de los números impares.
sw = para saber si acabamos de pasar por un número par o por un impar.
i = contador de los números de 1 a 1 000.

PSEUDOCODIGO

```
Borrar pantalla
par = 0
impar = 0
sw = 0
i = 1
Hacer mientras i <= 1000
     if sw=0
          impar = impar+i
          sw = 1
     else
          par = par+i
          sw = 0
     fin del if
     i = i+1
fin del hacer
imprime "La suma de los pares es:"
imprime par
imprime "La suma de los impares es:"
imprime impar
fin del programa
```

COMPROBACION

Inicializamos las variables y contador.
Dentro del ciclo pregunta por el valor del *sw*:

- Vale 0: es un número impar, por estar inicializado *i* a *1*. Acumula el valor de *i* al sumador *impar*. Cambia el estado del *sw*, por ser el siguiente número par.
- Vale 1: el valor del contador es par. Ejecuta el ELSE: añade el valor de *i* al sumador *par*. Cambia de nuevo el valor del *sw*; el siguiente número es impar.

Valga 0 ó 1 el sw, hay que aumentar el contador.
Cuando acabe el bucle imprime los valores de los sumadores.

RESUMEN

Bucles anidados: un ciclo puede estar formado por otro u otros ciclos.

Sintaxis:

Hacer mientras condición-1

Hacer mientras condición-2
.....
.....
Hacer mientras condición-3
.....
.....
Fin del hacer
.....
.....
Fin del hacer
.....
.....
Fin del hacer

Los puntos suspensivos son las distintas instrucciones a realizar.

El cálculo del número de caracteres de una variable alfanumérica lo expresamos:

hallar-longitud(variable)

Para recorrer los distintos caracteres de una variable alfanumérica o extraer una parte especificada de la misma utilizamos la expresión:

caracter(variable,posición,incremento)

donde:

— **Variable:** contiene la frase alfanumérica.
— **Posición:** es la posición en la cual nos encontramos dentro de la variable.
— **Incremento:** número de caracteres a leer de *variable*.

Programa principal: el programa escrito en primer lugar y, por ello, no es llamado desde otro programa.

Subprograma: conjunto de sentencias o instrucciones de un programa que realizan una determinada tarea; pueden ser ejecutados desde más de un punto del programa principal.
Se escriben con mayúsculas.
Están descritos fuera del programa principal.

Sintaxis:

HACER nombre-subprograma

Switch es una variable que sólo puede tomar dos valores exclusivos: 0 ó 1.
También se le llama *interruptor, bandera* o *flag*.
Los switch los designamos por *sw*.
Se utilizan para saber:

— Si el programa ha pasado por un determinado punto, preguntando por su estado: *if sw = 0*, entonces se imprimirán en pantalla algún tipo de comentario.
— Salir de un ciclo cuando el sw tenga un determinado valor, *hacer mientras sw = 0*.
— Dependiendo de su valor, realizar una u otra acción.

EJERCICIOS PROPUESTOS

1. Hallar el factorial de un número (recorriendo los números desde la unidad).
2. Imprimir los múltiplos de tres que hay entre dos números determinados de una forma alternativa, es decir, uno sí y otro no.
3. Hacer un programa que imprima, sume y cuente los números pares entre dos números determinados de una forma alternativa.
4. Hacer un programa que imprima de una serie de frases la de mayor longitud y el lugar que ocupa dentro de la serie.
5. Imprimir las letras de una frase de una forma alternativa, una de la parte final y la siguiente de la inicial, hasta recorrer todos los caracteres de la misma.
6. Introducir dos números inferiores a 50 por teclado. Al más pequeño aumentarlo de 5 en 5 y al mayor disminuirlo de 2 en 2. Imprimir ambas series de números, de una forma alternativa, hasta que el menor supere al mayor.
7. Introducir dos números por teclado de tal forma que uno sea menor que 50 y el otro comprendido entre 100 y 200. Al mayor se le restan 3 unidades y al menor se le aumenta en 7 unidades. Imprimir la secuencia de los números, hasta que se crucen, de forma alternativa.
8. Dados dos números, imprimir sus tablas de multiplicar de forma alternativa.
9. Dado un número menor que 10, hallar su tabla de multiplicar por duplicado, es decir, empezando por el cero y por el 10 de forma alternativa.

CAPITULO

Presentación en pantalla: Filas y columnas variables. Cabeceras

5.1. FILAS Y COLUMNAS VARIABLES

Al indicarle al ordenador que imprima un comentario o el valor de una variable por pantalla, lo realiza en la primera fila y en la primera columna. La siguiente orden de impresión la realiza en la fila 2, columna 1; la siguiente en la tercera fila, columna 1, etc.

El ordenador realiza cada orden de impresión en una fila nueva y siempre en la columna primera.

Nunca vamos a desear imprimir de esa forma. Lo haremos según las necesidades del problema.

La visualización de cualquier tipo de información, por pantalla o por impresora, debe aparecer con un determinado formato: en unas filas y unas columnas determinadas.

Es necesario que tanto las filas como las columnas se comporten como variables enteras.

Sintaxis:

en FILA,COLUMNA ...

donde FILA y COLUMNA son los nombres de las variables para designar las filas y las columnas, respectivamente.

Los puntos sucesivos indican lo que deseamos hacer en esa fila y columna determinada:

— Introducir un dato por teclado.
  Ejemplo: en 15,30 introduce numero
— Imprimir algún comentario o el valor de alguna variable.
  Ejemplo: en 10,20 imprime "Hola"
  en fi,35 imprime suma
  en fi,co+4 imprime apellidos

fi es la variable que contiene el valor de la fila y *co* la columna donde se desea imprimir el valor de las variables suma y apellidos.

Las filas y las columnas funcionan de manera análoga a las coordenadas *x* e *y* de Matemáticas.

Hay que tener en cuenta que:

— La pantalla tiene 24 líneas y 80 columnas.
— Las hojas de impresora suelen variar de formato. Las más comunes:
  a) Filas 62, aunque a las 55 se suele hacer el salto de página.
  b) Columnas de 80 o de 132.

Ejercicio 1

Introducir una frase por teclado. Imprimirla cinco veces en filas consecutivas, pero cada impresión irá desplazada cuatro columnas hacia la derecha.

ESTUDIO PREVIO

La introducción de la frase la realizamos en unas coordenadas determinadas. La primera visualización estará situada, por ejemplo, tres filas más abajo. Las posteriores impresiones de la frase irán en filas consecutivas. pero cuatro columnas hacia la derecha.

Las filas y las columnas se comportarán como contadores: las filas aumentan de una en una y las columnas de cuatro en cuatro.

Para la frase reservamos una longitud de 30 caracteres.

VARIABLES

frase = almacena la frase que vamos a introducir por teclado.
veces = contador de las veces que se ha impreso la frase.
fi = contador de filas.
co = contador de columnas.

PSEUDOCODIGO

```
Borrar pantalla
frase = espacios(30)
en 5,15 imprime "Frase:"
en 5,22 introduce frase
fi = 8
co = 15
veces = 0
Hacer mientras veces <= 5
     en fi,co imprime frase
     veces = veces + 1
     co = co + 4
     fi = fi + 1
fin del hacer
fin del programa.
```

COMPROBACION

En las coordenadas 5,15, es decir, en la fila 5 y en la columna 15, imprime el comentario de "Frase:", que ocupa 6 caracteres. En la misma fila, la 5, pero siete columnas más hacia la derecha (seis de "Frase:" y uno más para dejar un espacio vacío), le indicamos al ordenador que deseamos introducir una frase por teclado, *en 5,22 introduce frase.*

Antes de entrar en el ciclo inicializamos las variables *fi* y *co*, las filas a 8 y las columnas al mismo valor en que se introdujo la frase.

Dentro del ciclo, en las coordenadas indicadas por las variables *fi* y *co*, visualiza el valor de la frase.

Las filas, para la siguiente impresión, se incrementan en una unidad, *fi = fi + 1*, y las columnas en cuatro, *co = co + 4*.

Cada vez que se imprime la frase se incrementa el contador de *veces*.

Ejercicio 2

Hacer un pseudocódigo que imprima los números del 0 al 100, controlando las filas y las columnas.

Con el formato:

Los números del 0 al 100 son:

-- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --

ESTUDIO PREVIO

Se empezarán a imprimir los números en una fila determinada, por ejemplo la 7, continuando en dicha fila hasta que lleguemos a una columna determinada, por ejemplo la 75. Cuando las filas lleguen a dicho valor, la impresión de los siguientes números debe realizarse en dos filas posteriores, pero en la misma columna donde se empezaron a imprimir los números.

Los números deben ir separados unos de otros, en la misma fila pero en columnas diferentes.

Al tener que controlar tanto las filas como las columnas, hay que utilizar dos variables que realicen dicha función.

VARIABLES

c = contador para generar los números.
fi = contador de filas.
col = contador de columnas.

PSEUDOCODIGO

```
Borrar pantalla
c=0
en 5,20 imprime "Los números del 0 al 100 son:"
fi = 7
col= 5
Hacer mientras c < 101
    en fi,col imprime c
    c = c+1
    col = col+4
    if col >.75
         fi = fi+2
         col = 5
    fin del if
fin del hacer
fin del programa.
```

COMPROBACION

En las coordenadas 5,20, es decir, en la fila 5 y en la columna 20, nos imprime el comentario de "Los números del...".

Antes de entrar en el ciclo inicializamos las variables *fi* y *col*.

En la fila 7, en la columna 5, empieza a escribir los números. Va a continuar en la misma línea mientras el contador de las columnas sea menor o igual a 75. Cuando sea mayor tiene que aumentar la fila, *fi=fi+2*. y empezar a escribir en la columna inicial. *col=5*.

Ejercicio 3

Comprobar si un número mayor o igual que la unidad es primo.

ESTUDIO PREVIO

Un número es primo cuando sólo es divisible por sí mismo y por la unidad.

El número tiene que ser mayor que cero. Hay que controlarlo. Se necesitará un ciclo para introducir el número por teclado hasta que cumpla la condición de ser mayor que cero.

A este número lo iremos dividiendo por todos los números comprendidos entre el inmediatamente inferior a él y el dos.

Al hacer la división puede suceder que sea:

- Exacta: no es primo y debemos terminar. para lo cual hay que utilizar un *sw*.
- No es exacta: es primo, por ahora. Se continúa realizando la división.

En el ciclo habrá dos salidas: cuando lleguemos a dividir al número por la unidad o cuando no sea primo. Necesitamos el operador lógico AND.

VARIABLES

numero = introducir el número.
i = contador decreciente para dividir desde *numero-1* hasta 2.
res = para calcular o no más números primos.
sw = salir del bucle y comprobar si es primo.

PSEUDOCODIGO

```
res = "S"
Hacer mientras res = "S"
     borrar pantalla
     numero = 0
     sw = 0
     Hacer mientras numero < 1
          en 8,10 imprime "Número:"
          en 8,18 introduce numero
     fin del hacer
     i = numero-1
     Hacer mientras i > 1 AND sw <> 1
          if numero = int(numero/i)*i
               sw = 1
          else
               i = i-1
          fin del if
     fin del hacer
     if sw = 1
          en 10,10 imprimir "no es primo"
     else
          en 10,10 imprimir "es primo"
     fin del if
     hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

Inicializamos *res* para entrar en el ciclo.

Hacer mientras numero < 1, controla que el número introducido sea mayor que cero.

Cuando sea mayor que cero se asigna *i=numero–1*; debemos comprobar desde el número anterior al introducido.

Para comprobar si es primo realiza el ciclo *hacer mientras i> 1 AND sw< >1*.

Si el número que introducimos es:

- La unidad o el dos: no entra en el ciclo, por valer i 0 ó 1; el sw no cambia de valor, es primo.
- Mayor que dos: entra, se cumplen las dos condiciones, *i* como mínimo valdrá 2 y el *sw=0*. Empieza a hacer las divisiones. Comprueba si, *if numero=int(numero/i)*i:*

a) Son enteras: no es primo y debe salir del ciclo. Para ello asignamos *sw=1*, con lo cual deja de cumplirse una de las dos condiciones y sale.
b) No son enteras: por ahora va siendo primo. Debe seguir comprobando; por tanto, disminuimos en una unidad el contador.

Cuando salimos del ciclo puede que sea por los siguientes motivos:

— No es primo: el valor del *sw* ha cambiado.
— Se ha realizado la división por todos los números: el valor de *sw* no ha cambiado.

Preguntamos por el valor del *sw*. Según sea su valor, el número será o no primo.
A continuación ejecuta el subprograma MAS.

Ejercicio 4

Introducir un número menor de 5 000 y pasarlo a número romano. Repetir la operación cuantas veces queramos.

ESTUDIO PREVIO

El número a convertir debe estar entre cero y 5 000.
Si introducimos el 1 237, debe visualizarlo de la forma MCCXXXVII.
El número debe imprimirlo en la misma línea. Hay que dejar variable la columna. Cada vez que imprima un carácter, la columna debe aumentar de valor. La fila, por el contrario, debe permanecer constante.
Bucles que hay: mientras sea mayor de 1 000, mayor de 100 y mayor de 1. Debemos particularizar los demás casos.

VARIABLES

res = para convertir más números.
numero = almacena el número.
col = controla la columna en la que imprime.

PSEUDOCODIGO

```
res = "S"
Hacer mientras res = "S"
     borrar pantalla
     num = 0
     Hacer mientras num < 1 OR num > 5000
          en 8,10 imprime "Número:"
          en 8,18 introduce num
     fin del hacer
     col=15
     hacer mientras num >= 1000
          en 15,col imprimir "M"
```

```
     num = num-1000
     col = col+1
fin del hacer
if num >= 900
     en 15,col imprimir "CM"
     num = num-900
     col = col+2
fin del if
if num >= 500
     en 15,col imprimir "D"
     num = num-500
     col = col+1
fin del if
hacer mientras num >= 100
     en 15,col imprimir "C"
     num = num-100
     col = col+1
fin del hacer
if num >= 90
     en 15,col imprimir "XC"
     num = num-90
     col = col+2
fin del if
if num >= 50
     en 15,col imprimir "L"
     num = num-50
     col = col+1
fin del if
if num >= 40
     en 15,col imprimir "XL"
     num = num-40
     col = col+2
fin del if
hacer mientras num >= 10
     en 15,col imprimir "X"
     num = num-10
     col = col+1
fin del hacer
if num = 9
     en 15,col imprimir "IX"
     num = num-9
     col = col+2
fin del if
if num >= 5
     en 15,col imprimir "V"
     num = num-5
```

```
        col = col+1
    fin del if
    if num >= 4
        en 15,col imprimir "IV"
        num = num-4
        col = col+2
    fin del if
    hacer mientras num > 0
        en 15,col imprimir "I"
        num = num-1
        col = col+1
    fin del hacer
    hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

El primer ciclo, *hacer mientras num < 1 OR num > 5000*, controla que el número esté entre 1 y 5 000.

La fila es constante y, por tanto, no hace falta definirla como variable. Va a escribir el número en la fila 15.

La columna, por el contrario, va a ser variable. Le especificamos que empiece en la 15.

Si es mayor de 1 000, debe estar restando dicha cantidad hasta que sea menor. Cada vez que sea mayor debe escribir su equivalencia *M*: aumenta la columna en una unidad por tener de longitud un dígito.

Si es mayor de 900, debe imprimir *CM*, restar dicha cantidad y aumentar en dos la columna.

Análogamente para el resto, hasta convertir la última unidad del número especificado.

Ejercicio 5

Introducir una frase por teclado. Imprimirla en el centro de la pantalla. Repetir el proceso.

ESTUDIO PREVIO

La longitud de la frase es variable, pero siempre tiene que imprimirse centrada en la pantalla.

La pantalla tiene 24 filas (su mitad es 12) y 80 columnas (su mitad es 40).

La fila es constante, 12, pero la columna debe ser variable en función de la longitud de la frase introducida por teclado.

La longitud de la frase la podemos calcular con la instrucción *longitud(variable)*. Una

vez conocida su longitud, le indicamos al ordenador que imprima la mitad antes de la columna 40, mediante la operación:

40-int(longitud(variable)/2).

Para la frase reservamos una longitud de 40 caracteres.

VARIABLES

frase = almacena la frase que vamos a introducir por teclado.
res = para introducir más frases.

PSEUDOCODIGO

```
res = "S"
frase = esepacios(40)
hacer mientras res = "S"
     borrar pantalla
     en 5,15 imprime "Frase:"
     en 5,22 introduce frase
     en 12,40-int(longitud(frase)/2) imprime frase
     hacer MAS
fin del hacer
fin del programa
```

COMPROBACION

Una vez introducida la frase por teclado calculamos su longitud. *longitud(frase)*. Puede ser impar y al calcular su mitad, *longitud(frase)/2*, dar un número fraccionario. El contador de las columnas debe ser entero; por este motivo hay que calcular su parte entera.

A la columna mitad de la pantalla le restamos dicho valor y, a partir de dicha columna, empezamos a imprimir la frase.

5.2. CABECERAS

Toda visualización de datos, además de realizarse en un formato preestablecido, suele ir acompañado de unas cabeceras.

Las **cabeceras** son los comentarios que aparecen al principio de cualquier listado y se repite de forma idéntica, excepto el número de la página, en todas ellas.

Tanto la pantalla como las hojas de la impresora poseen un número determinado de filas donde imprimir los datos; éstos pueden ocupar más de una pantalla o una hoja de impresora.

Si dejamos las filas como variables y no controlamos su valor, cuando sobrepase un determinado valor, 24 para la pantalla, da error la ejecución del programa, porque le estamos indicando que imprima en la línea 25 y el ordenador sabe que sólo posee 24.

La generación de informes con saltos de página y cabeceras por impresora se verá con más detalle en capítulos posteriores. Ahora nos vamos a limitar a la visualización de datos por pantalla.

Es preciso controlar que cuando llegue el contador de las filas a un valor determinado visualicemos por pantalla un comentario, conocido con el nombre de *pie de página*, por ejemplo: "Pulse RETURN o ENTER para continuar", junto con una interrupción, parada, en la ejecución del programa, lo cual permite ver detenidamente lo impreso en pantalla.

La interrupción del programa se consigue con la instrucción:

detener listado

En caso de que nuestro lenguaje no posea una instrucción análoga, podemos conseguir los mismos resultados con la utilización de una variable alfanumérica, *continuar*, de la siguiente forma:

```
continuar = espacios(1)
en 23,20 imprime "Pulse RETURN o ENTER para continuar"
en 23,70 introduce continuar
```

El programa espera hasta que no se pulse RETURN.

Ejercicio 6

Hacer un pseudocódigo que imprima los números del 0 al 1 000, controlando las filas y las columnas.

Con el formato:

Los números del 0 al 1 000 son:

```
--   --   --   --   --   --   --   --
--   --   --   --   --   --   --   --
```

ESTUDIO PREVIO

En la fila 3, por ejemplo, visualizamos el comentario de la cabecera.

Los números se empiezan a imprimir en una fila determinada, la 5, y en la columna 5.

Todos los números no caben en una pantalla. Cuando el contador de las filas llegue a un valor determinado, el 21, en la línea 22 se imprime el mensaje "Pulse RETURN o ENTER para continuar", deteniendo la ejecución del programa.

Borramos pantalla y en la fila 3 volvemos a visualizar la cabecera y continuamos la impresión de los números en la fila 5, columna 5.

Tanto las filas como las columnas deben ser variables.

VARIABLES

numero = contador para generar los números.
fi = contador de filas.
co = contador de columnas.

PSEUDOCODIGO

```
numero = 0
fi = 25
Hacer mientras numero <= 1000
     if fi = 25
          borrar pantalla
          en 3,20 imprime "Los números del 0 al 1000 son:"
          fi = 5
          co = 5
     fin del if
     en fi,co imprime numero
     numero = numero+1
     co = co+4
     if co > 75
          fi = fi+1
          co = 5
     fin del if
     if fi = 21
          en 22,20 imprime "Pulse RETURN o ENTER para continuar"
          detener listado
          fi = 25
     fin del if
fin del hacer
fin del programa.
```

COMPROBACION

En las coordenadas 3,20, es decir, en la fila 3 y en la columna 20, imprime la cabecera: "Los números del...".

Antes de entrar en el ciclo inicializamos la variable *fi* a 25, un valor que nunca puede alcanzar dentro del programa.

Entra en el ciclo, *if fi = 25*, se cumple la condición: borra pantalla y visualiza la cabecera.

En la fila 5 y en la columna 5 empieza a escribir los números. Va a continuar en la misma línea mientras el contador de las columnas sea menor o igual a 75. Cuando sea mayor tiene que aumentar la fila, *fi=fi+1*, y empezar a escribir en la columna inicial, *co=5*.

Llegará un momento en que el contador de las filas llegue a 21; cumple la condición *if fi=21*: en la línea 22 imprime el pie de página, detiene el listado y reinicializa las filas a 25.

Comprueba que el número es menor de 1000, continúa ejecutándose el programa, *if fi = 25*, borra pantalla e imprime de nuevo la cabecera.

Ejercicio 7

Realizar la tabla de multiplicar de un número entre 0 y 10, según el formato:

Tabla de multiplicar del número: --

-- * 0 = -
-- * 1 = --
. . . . . .
. . . . . .
-- * 10 = --

ESTUDIO PREVIO

Al número introducido por teclado se le multiplica por una variable, la cual irá tomando los valores consecutivos de 0 a 10. A medida que está realizando la multiplicación las filas deben aumentar de valor; serán variables.

VARIABLES

num = número del cual realizamos su tabla de multiplicar.
fi = controlar las filas.
i = contador que vaya de 0 a 10.

PSEUDOCODIGO

```
num = -1
hacer NUMERO
borrar pantalla
en 5,10 imprime "Tabla de multiplicar del número:",num
i = 0
fi = 8
hacer mientras i <= 10
     en fi,15 imprime num
     en fi,18 imprime "*"
     en fi,20 imprime i
     en fi,22 imprime "="
     en fi,24 imprime num*i
     fi = fi + 1
     i = i + 1
fin del hacer
fin del programa.

NUMERO.
hacer mientras num < 0
     borrar pantalla
     en 10,25 imprime "Número:"
     en 10,33 introduce num
fin del hacer.
```

COMPROBACION

Se introduce el número controlando que sea cero o mayor que cero.

Realiza la cabecera.

El ciclo lo debe realizar *mientras i <=10.* empezando en cero. Cada vez que hace la multiplicación por un número aumenta la fila y el contador *i.*

Ejercicio 8

Realizar la tabla de multiplicar del 0 al 10. Cada número en una pantalla. Con el mismo formato del ejercicio anterior.

ESTUDIO PREVIO

Los números van del 0 al 10.

A su vez, a cada número hay que multiplicarlo por una variable que vaya del 0 al 10. empezando para cada uno de ellos desde el cero.

Son necesarios dos ciclos. uno dentro del otro; ambos van del 0 al 10.

La tabla de multiplicar de cada número irá en una pantalla y la visualizaremos todo el tiempo que deseemos. Hay que detener el listado.

VARIABLES

num = número del cual realizamos su tabla de multiplicar.
fi = controlar las filas.
i = contador que vaya de 0 a 10. para multiplicar a *num.*

PSEUDOCODIGO

```
num = 0
hacer mientras num <= 10
      i = 0
      borrar pantalla
      en 5,10 imprime "Tabla de multiplicar del número:",num
      fi = 8
      hacer mientras i <= 10
            hacer IMPRIMIR
      fin del hacer
      en 20,20 imprime "Pulse ENTER para continuar"
      detener listado
fin del hacer
fin del programa.
```

```
IMPRIMIR.
en fi,15 imprime num
en fi,18 imprime "*"
en fi,20 imprime i
en fi,22 imprime "="
en fi,24 imprime num*i
fi = fi + 1
i = i + 1.
```

COMPROBACION

El ciclo, *hacer mientras num <= 10*, realiza la tabla de multiplicar de los números.

Cada uno de estos números se multiplica por *i*, que va desde 0 hasta 10; por tanto, antes de entrar en el segundo bucle, *hacer mientras i <= 10*, es necesario reinicializar *i* a cero y las filas a 8.

RESUMEN

Sabemos imprimir en pantalla dejando a las *Filas* y *columnas variables* de la forma:

en FILA,COLUMNA ...

donde FILA y COLUMNA son los nombres de las variables para designar las filas y las columnas, respectivamente.

En los puntos sucesivos indicamos al ordenador que deseamos introducir una información por teclado o bien visualizarla.

— Introducir un dato por teclado.
Ejemplo: en 15,30 introduce numero

— Imprimir algún comentario o el valor de alguna variable.
Ejemplo: en 10,20 imprime "Hola".
en fi,35 imprime suma
en fi,co+4 imprime apellidos

La pantalla posee 24 líneas y 80 columnas.

Las hojas de impresora suelen variar de formato. Las más comunes:

— Filas 62, aunque a la 55 se suele hacer el salto de página.
— Columnas de 80 o de 132.

Cabeceras: son los comentarios que aparecen al principio de cualquier listado y se repiten de forma idéntica, excepto el número de la página, en todas ellas.

Pie de página: los comentarios o visualización de resultados que aparecen al final de cada página.

Detener la ejecución de un programa: *detener listado.*

EJERCICIOS PROPUESTOS

1. Imprimir los números que pertenecen a la serie 1, 3, 6, 10... número determinado. En cada línea se imprimirán cinco números. Calcular su suma.

2. Imprimir la tabla de multiplicar del 2 y del 3 con el formato:

Tabla del 2	Tabla del 3
2 × 0 = 0	3 × 0 = 0
2 × 1 = 2	3 × 1 = 3

de tal forma que aparezca en pantalla primero la del 2 y luego la del 3.

3. Igual que el Ejercicio 2, pero de forma que la tabla del 3 se imprima al revés (empezando por el número 10, 9, 8 ...)

4. Igual que el Ejercicio 2, pero visualizando por líneas (multiplicando ambos números por 0, luego por 1, por 2 ...)

5. Igual que el Ejercicio 2, pero imprimiéndolos en pantallas diferentes.

6. Introducir un número y desglosarlo de forma ideal en billetes y monedas de curso legal.

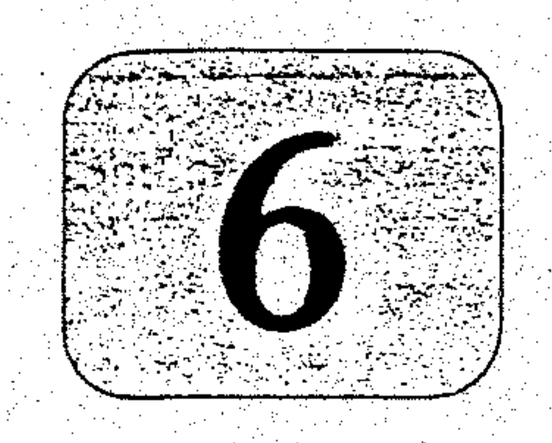
7. Imprimir los múltiplos comunes de 2 y de 3 que hay entre la unidad y un determinado número. Los números se imprimirán en una fila diferente y centrados en ella. Cada 15 números cambiar de pantalla.

8. Calcular los impuestos que debe pagar una casa en función del número de metros cuadrados de que disponga si por metro cuadrado paga 1 000 ptas., por cada balcón 200 ptas., por cada ventana exterior 100 ptas., y por cada interior 25 ptas. Si el tejado es de uralita, paga 1 000 ptas., si es de teja, 800 ptas. y si es de otra clase, 1 100 ptas. Si la casa tiene más de 10 años, se le descuenta un 10 por 100 de los impuestos totales y si es mayor de 20 años, un 15 por 100. Si en el edificio hay menos de 10 vecinos, experimenta un recargo adicional del 2 por 100, pero si hay más de 10 y menos de 20, se le descuenta un 2 por 100, y si hay más de 20, un 4 por 100.

9. Una empresa de autobuses realiza viajes desde A a 3 ciudades diferentes: B, C y D, dos veces al día (mañana y tarde).
Imprimir al final del día el número de viajeros que han ido a cada ciudad y el dinero recaudado según el formato:

	Pasajeros			Recaudación
Ciudad	Mañana	Tarde	Total	
B	---	---	----	------
C	---	---	----	------
D	---	---	----	------
TOTAL:	---	---	----	------

CAPITULO



Números aleatorios. Menús

6.1. NUMEROS ALEATORIOS

En determinadas ocasiones no es necesario introducir números desde el exterior; podemos indicarle al ordenador que los genere él interiormente. Son lo que se llama números aleatorios.

Los **números aleatorios** son números creados internamente por el ordenador.
El valor de los números está comprendido entre cero y la unidad.
La forma de generarlos depende de cada lenguaje.

Sintaxis:

RND

el número generado por RND es mayor que 0 y menor que 1.

Para generar números enteros, mayores que la unidad y comprendidos entre unos límites, lo expresaremos de la forma:

*INT(RND*número)+1*

donde:

— INT: parte entera.
— RND: genera los números de una forma aleatoria.
— Número: valor máximo de los números a generar.
— Se le suma la unidad para generar el 1 y el número máximo deseado.

Hay que tener presente que el ordenador puede generar varios números con igual valor.

Ejemplo: generar números aleatorios entre 1 y 50 sería:

*int(rnd*50)+1.*

Ejercicio 1

Simular el lanzamiento de una moneda al aire e imprimir si ha salido cara o cruz. Repetir el proceso tantas veces como deseemos.

ESTUDIO PREVIO

Una moneda sólo tiene dos posibilidades: cara o cruz, ambas con igual probabilidad de salir.

Generamos un número: si es menor o igual a 0.5, le asignamos el valor cara: si es mayor, cruz.

VARIABLE

res = repetir el proceso.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     borrar pantalla
     if rnd <= 0.5
          en 10,35 imprime "Cara"
     else
          en 10,35 imprime "Cruz"
     fin del if
     hacer MAS
fin del hacer
fin del programa.

MAS.
res = espacios(1)
hacer mientras res <> "S" AND res <> "N"
     en 20,30 imprime "Más lanzamientos (S/N):"
     en 20,57 introduce res
     res = convertir_mayusculas(res)
fin del hacer.
```

COMPROBACION

El proceso es repetitivo, *hacer mientras res="S"*.

Al mismo tiempo que se genera el número preguntamos por su valor, *if rnd <=0.5;* si es menor o igual a 0.5, entonces que imprima Cara; en caso contrario, será Cruz.

Realiza el subprograma MAS, para preguntarnos si deseamos hacer más lanzamientos.

Ejercicio 2

Simular cien tiradas de un dado y contar las veces que aparece el número 6.

ESTUDIO PREVIO

La numeración de los dados es del 1 al 6; por tanto. serán los números a generar.
Al generar un número se comprueba si es el 6.

VARIABLES

i = contador de las tiradas.
c = contador de las veces que aparece el 6.

PSEUDOCODIGO

```
Borrar pantalla
c = 0
i = 0
hacer mientras i<101
     if 6 = int(rnd*6)+1
          c=c+1
     fin del if
     i=i+1
fin del hacer
en 10,20 imprime "Las veces que aparece el 6 son:", c
fin del programa.
```

COMPROBACION

Inicializamos los contadores. El proceso es repetitivo, *hacer mientras i<101.*
Se comprueba si el número generado es 6, en cuyo caso aumenta el contador *c.*
Tanto si el número es 6 como si no, aumenta *i.*
Cuando salga del bucle en la fila 10, columna 20, nos imprime el comentario y a continuación el valor de la variable *c.*

Ejercicio 3

Simular cien tiradas de dos dados y contar las veces que entre los dos suman 10.

ESTUDIO PREVIO

Generamos dos números aleatorios, comprendidos cada uno de ellos entre 1 y 6, uno por cada dado.
Comprobamos que su suma sea 10.

VARIABLES

i = contador de las tiradas.
c = contador de las veces que entre los dos dados suman 10.

PSEUDOCODIGO

```
Borrar pantalla
c = 0
i = 0
hacer mientras i<101
     if int(rnd*6)+int(rnd*6)+2 = 10
          c=c+1
     fin del if
     i=i+1
fin del hacer
en 10,20 imprime "Las veces que suman 10 son:", c
fin del programa.
```

COMPROBACION

Inicializamos los contadores.

Comprueba si la suma de los dos números generados es 10, en cuyo caso aumenta el contador *c*.

Tanto si la suma de los dos números es 10 como si no, aumenta el contador de tiradas.

Cuando salga del bucle imprime el valor de *c*.

Ejercicio 4

Simular una carrera de dos caballos si cada uno tiene igual probabilidad de ganar.

ESTUDIO PREVIO

Un caballo determinado se moverá si el número generado por el ordenador le corresponde.

Los dos caballos tienen igual probabilidad de ganar; quiere decir que de los números generados la mitad corresponden a un caballo y la otra mitad al otro.

Cada uno se moverá por una fila distinta y ganará el que primero llegue a una columna determinada, por ejemplo la 75, avanzando cada vez un número fijo de columnas, por ejemplo 4.

Ambos caballos salen de la misma columna.

A medida que avanzan hay que borrar de la pantalla la estela que dejan.

Los caballos los representamos en pantalla mediante un dibujo.

VARIABLES

col1 = contador de columnas del primer caballo.
col2 = contador de columnas del segundo caballo.
dibujo = representación de los caballos.

PSEUDOCODIGO

```
Borrar pantalla
dibujo = "****"
coll = 4
col2 = 4
en 10,coll imprime dibujo
en 12,col2 imprime dibujo
hacer mientras coll <= 75 AND col2 <=75
     if rnd <= 0.5
          en 10,coll imprime "    "
          coll = coll + 4
          en 10,coll imprime dibujo
     else
          en 12,coll imprime "    "
          col2 = col2 + 4
          en 12,col2 imprime dibujo
     fin del if
fin del hacer
en 16,20 imprime "El ganador es el caballo número:"
if coll >=75
     en 16,54 imprime "1"
else
     en 16,54 imprime "2"
fin del if
fin del programa.
```

COMPROBACION

Se inicializan las columnas desde donde se dará la salida y se imprimen los caballos. cada uno en filas distintas.

La carrera durará mientras que algún caballo no llegue o sobrepase la columna 75. *hacer mientras coll <= 75 AND col2 <=75*. Dentro del bucle se genera un número aleatorio. mediante la instrucción *rnd*. Si dicho número es *<=0.5*. por ejemplo. debe avanzar el caballo 1. pero antes borra su estela imprimiendo tantos espacios en blanco como longitud tenga el dibujo, se aumenta la columna y se imprime el caballo en su nueva posición. Si el número aleatorio es mayor de 0.5. se repite el proceso para el otro caballo.

Cuando alguno llegue o sobrepase la línea de meta para imprimir el número de caballo ganador se debe preguntar cuál de las columnas ha alcanzado dicho valor. *if coll >=75*.

6.2. MENUS

Las operaciones a realizar se están presentando por separado sin ningún nexo de unión entre ellas. Generalmente esto no sucede así; es preciso realizar una serie de operaciones que pueden estar más o menos relacionadas entre sí, indicando cuál es la que deseamos ejecutar en ese instante.

Un **menú** consiste en presentar en pantalla una ventana con una serie de operaciones u opciones a realizar, cada una de las cuales realiza una función determinada.

Cuando termine de ejecutar cada una de ellas, mediante subprogramas, el programa vuelve de nuevo al menú del que había partido.

A veces los menús se presentan anidados, es decir, alguna de las opciones del menú, al ser seleccionada, hace que aparezca otro menú, dando lugar a nuevas posibilidades de elección.

Los menús permiten ejecutar más de un programa, sin necesidad de tener que escribir su nombre, cada vez que se desea ejecutarlo. Simplemente, le indicaremos mediante una variable la opción deseada.

La selección del programa a realizar se puede hacer mediante la instrucción IF. Pero hay una más rápida y más fácil de manejar: es la instrucción condicional múltiple CASE.

Sintaxis:

```
hacer case
        case condición1
            hacer subprograma1
      case condición2
            hacer subprograma2
      . . . . . .
      . . . . . .
      [en caso contrario]
      [. . . . . .]
fin del case
```

Los corchetes indican que esta parte de la sentencia *case* es opcional y se utilizará cuando el valor de la variable no coincida con ninguno de los valores controlados en las distintas condiciones de la instrucción *case*.

Funciona bifurcando la ejecución del programa a las instrucciones que siguen a la evaluación verdadera de una condición CASE. La ejecución del programa continúa hasta encontrarse la próxima orden *case*, *en caso contrario* o *fin del case*. A continuación se ejecuta la primera línea que sigue a la sentencia CASE.

Si ninguna de las condiciones del CASE es verdadero, el siguiente conjunto de instrucciones que sigue a la sentencia *en caso contrario*, si existe, se ejecuta hasta la próxima orden *fin del case*.

En las condiciones puede haber más de una condición, siempre y cuando estén relacionadas mediante los operadores lógicos.

Esta instrucción también se puede utilizar cuando sabemos que los valores de las variables sólo pueden ser unos fijos y determinados.

Ejercicio 5

Introducir dos números por teclado y mediante un menú que calcule su suma, su resta, su multiplicación y su división.

ESTUDIO PREVIO

Una vez introducidos los dos números por teclado, en pantalla nos debe aparecer:

```
Menú de opciones
1.- Suma
2.- Resta
3.- Multiplicación
4.- División
5.- Salir del programa
Elige opción: -
```

Siempre hay que considerar una posibilidad más de las indicadas, la salida; de lo contrario, estaríamos en un ciclo sin fin.

Necesitamos una variable para introducir por teclado la opción que deseamos realizar en ese instante.

VARIABLES

n1 = introducir el primer número.
n2 = introducir el segundo número.
op = introducir la opción deseada.

PSEUDOCODIGO

```
op = 0
borrar pantalla
en 10,20 imprime "Número:"
en 10,29 introduce n1
en 12,20 imprime "Número:"
en 12,29 introduce n2
hacer mientras op <> 5
     op = 0
     borrar pantalla
     en 6,20 imprime "Menú de opciones"
     en 10,25 imprime "1.- Suma"
     en 12,25 imprime "2.- Resta"
     en 14,25 imprime "3.- Multiplicación"
     en 16,25 imprime "4.- División"
```

```
en 22,25 imprime "Elige opción:"
en 22,39 introduce op
borrar pantalla
hacer case
     case op = 1
          en 10,20 imprime "Su suma es:"
          en 10,33 imprime n1+n2
          detener-listado
     case op = 2
          en 10,20 imprime "Su resta es:"
          en 10,33 imprime n1-n2
          detener-listado
     case op = 3
          en 10,20 imprime "Su multiplicación es:"
          en 10,45 imprime n1*n2
          detener-listado
     case op = 4
          en 10,20 imprime "Su división es:"
          en 10,45 imprime n1/n2
          detener-listado
     fin del case
fin del hacer
fin del programa.
```

COMPROBACION

Antes del ciclo, *hacer mientras op < > 5*, introducimos ambos números sobre los cuales realizamos las opciones.

Nos visualiza en pantalla el menú con todas las opciones y elegimos una de ellas. En caso de no corresponder el valor de *op* con los controlados en la instrucción *case*, nos pedirá de nuevo la opción. Si el valor de *op* es:

— Uno: realiza la suma de los números.
— Dos: realiza la resta.
— Tres: realiza la multiplicación.
— Cuatro: realiza la división.
— Cinco: abandona el ciclo *hacer mientras* y termina el programa.

La orden de *detener-listado* también puede ir después de la instrucción *fin del case*; de esta forma sólo hace falta escribirla una vez.

Ejercicio 6

Hacer un programa que nos permita introducir un número por teclado y sobre él se realicen las siguientes operaciones:

1. Comprobar si es primo.

2. Hallar su factorial.
3. Imprimir su tabla de multiplicar.

ESTUDIO PREVIO

Una vez introducido el número por teclado, en pantalla nos debe aparecer:

```
Menú de opciones
1.- Comprobar si es primo
2.- Factorial
3.- Tabla de multiplicar
4.- Salir del programa
Elige opción:-
```

Siempre hay que considerar una posibilidad más de las indicadas, la salida; de lo contrario, estaríamos en un ciclo sin fin.

Necesitamos una variable para introducir por teclado la opción que deseamos realizar en ese instante.

VARIABLES

n = introducir el número.
op = introducir la opción deseada.

PSEUDOCODIGO

```
op = 0
borrar pantalla
en 10,20 imprime "Número:"
en 10,29 introduce n
hacer mientras op <> 4
      op = 0
      borrar pantalla
      en 6,30 imprime "Menú de opciones"
      en 10,25 imprime "1.- Comprobar si es Primo"
      en 12,25 imprime "2.- Factorial"
      en 14,25 imprime "3.- Tabla de multiplicar"
      en 16,25 imprime "4.- Salir del programa"
      en 22,25 imprime "Elige opción:"
      en 22,39 introduce op
      hacer case
            case op = 1
                  hacer PRIMO
            case op = 2
                  hacer FACTORIAL
```

```
        case op = 3
              hacer TABLA
    fin del case
fin del hacer
fin del programa.
```

Donde los subprogramas PRIMO, FACTORIAL y TABLA son iguales a los descritos anteriormente.

COMPROBACION

Antes del ciclo, *hacer mientras op < > 4*, introducimos el número sobre el que vamos a realizar todas las opciones.

Nos visualiza en pantalla el menú con todas las opciones y elegimos una de ellas. En caso de no corresponder el valor de *op* con los controlados en la instrucción *case*, nos pedirá de nuevo la opción. Si el valor de *op* es:

— Uno: ejecuta el subprograma PRIMO.
— Dos: ejecuta FACTORIAL.
— Tres: ejecuta TABLA.
— Cuatro: abandona el ciclo *hacer mientras* y termina el programa.

RESUMEN

Números aleatorios son los números creados internamente por el ordenador. Están comprendidos entre cero y la unidad.
Lo expresamos

RND

el número generado por RND es mayor que 0 y menor que 1.

Para generar números enteros, mayores que la unidad y comprendidos entre unos límites, lo expresaremos de la forma:

*INT(RND*número)+1*

donde:

— INT: parte entera.
— RND: genera los números de una forma aleatoria.
— Número: valor máximo de los números a generar.
— Se le suma la unidad para generar el 1 y el número máximo deseado.

El ordenador puede generar varios números con igual valor.

Los **Menús** presentan en pantalla, mediante una ventana, una serie de operaciones u opciones a realizar.
Cuando termine de ejecutar cada una de esas opciones, mediante subprogramas, el programa vuelve de nuevo al menú del cual había partido.

Los menús pueden presentarse anidados, es decir, alguna de las opciones del menú, al ser seleccionada, hace que aparezca otro menú dando lugar a nuevas posibilidades de elección.

La selección de cada una de las distintas opciones la realizamos con la instrucción condicional *CASE*, cuyo formato es:

hacer case
    case condición1
        hacer subprograma1
    case condición2
        hacer subprograma2
    . . . . . .
    . . . . . .
    [en caso contrario]
    [. . . . . .]
fin del case

Los corchetes indican que esta parte de la sentencia *case* es opcional.

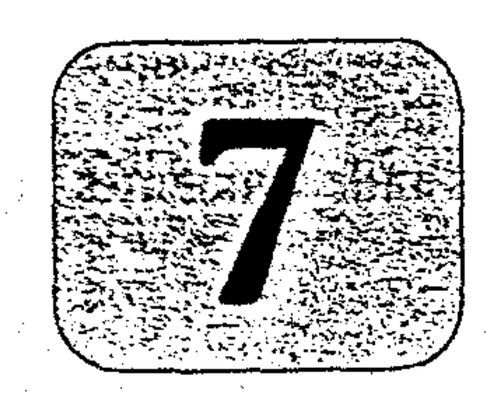
Si el valor de la variable no coincide con ninguno de los valores controlados en las distintas condiciones de la sentencia *case*, vuelve a pedir otra vez la opción deseada o ejecuta las instrucciones reseñadas, en *caso contrario*.

En las condiciones se pueden utilizar los operadores lógicos.

EJERCICIOS PROPUESTOS

1. Simular el lanzamiento de una moneda al aire y que pare cuando salgan 3 caras seguidas. Imprimir el número de tiradas realizadas.
2. Simular el lanzamiento de un número determinado de tiradas de dos monedas al aire. ¿Cuántas caras y cruces han salido? Imprimir el número mayor de caras consecutivas.
3. Generar aleatoriamente una quiniela de una columna. Si la probabilidad de que salga uno es del 50 por 100, la de x es del 30 por 100 y la del 2 es del 20 por 100.
4. Generar aleatoriamente una quiniela de 8 columnas.
5. Simular una carrera de cuatro caballos si cada uno tiene igual posibilidad de ganar.
6. Simular una carrera de caballos donde correrán entre dos y cinco. El número de caballos será a elección nuestra.
7. Introducir el número del mes e imprimirlo en letra utilizando la instrucción CASE.
8. Introducir la nota de una asignatura por teclado que esté comprendida entre 0 y 10, y la escriba en letra: Muy Deficiente si es menor de tres, Insuficiente si es mayor de tres y menor de cinco, Suficiente entre cinco y seis, Bien ...
9. Introducir por teclado V o H e imprimir Varón o Hembra.

CAPITULO



Problemas de propósito general

7.1. INTRODUCCION

En este capítulo vamos a realizar una serie de ejercicios que abarcan todo los conceptos vistos hasta ahora.

Alguno de ellos, cuando se codifiquen en algún lenguaje, sólo pretenden hacer pasar unos momentos divertidos jugando con el ordenador, como pueden ser el intentar adivinar un número, o la simulación de una serie de partidas de pelota a mano entre dos pelotaris.

Otros tienen aplicaciones matemáticas al tratarse de verdaderos ejercicios de Matemáticas, pero fáciles de realizar: resolver una ecuación de segundo grado, el cálculo de la suma de una serie de términos de una progresión aritmética, o la resolución de un sistema de dos ecuaciones con dos incógnitas.

También los hay con vistas a futuras aplicaciones como puede ser el caso: ver cuál es la persona que más sueldo tiene en una determinada empresa o introducir por teclado una fecha lógica.

Todos ellos tienen un denominador común: aprender a razonar desde el punto de vista informático, utilizando las herramientas descritas en los capítulos anteriores.

Al final del capítulo, y a modo de resumen, se vuelve a recordar las tres grandes instrucciones para la realización de cualquier programa estructurado: el ciclo *HACER*, la instrucción condicional *IF* y la instrucción condicional múltiple *CASE*.

Ejercicio 1

Hacer un programa que obtenga el producto de dos números enteros positivos mediante sumas sucesivas.

ESTUDIO PREVIO

Un número, el que sea, se suma tantas veces como indique el otro número y su resultado se va almacenando en una variable. Se necesita un bucle para realizar la suma.

Los números deben ser positivos, para lo cual, necesitamos dos bucles independientes, uno para cada número.

Si algún número es cero, el producto automáticamente también lo es.

VARIABLES

num1 = almacena el primer factor.
num2 = almacena el segundo factor.
sum = guarda el resultado de la suma.

PSEUDOCODIGO

```
num1 = -1
hacer PRIMER-NUMERO
num2 = -1
hacer SEGUNDO-NUMERO
sum = 0
hacer mientras num1 <> 0 AND num2 <> 0
     sum = sum + num2
     num1 = num1 - 1
fin del hacer
borrar pantalla
en 10,20 imprime "El producto de: ",num1
en 12,20 imprime "por ",num2
en 14,20 imprime "es ",sum
fin del programa.

PRIMER-NUMERO.
hacer mientras num1 < 0
     borrar pantalla
     en 10,25 imprime "Número:"
     en 10,33 introduce num1
fin del hacer.

SEGUNDO-NUMERO.
hacer mientras num2 < 0
     borrar pantalla
     en 10,25 imprime "Número:"
     en 10,33 introduce num2
fin del hacer.
```

COMPROBACION

Para controlar que ambos números sean positivos se inicializan a un número negativo y

el proceso de la entrada de los números será repetitivo hasta que sean mayores o iguales a cero.

Se utilizan dos ciclos en lugar de uno para esta operación, pues si se utilizara sólo uno podría suceder que un número fuera positivo y el otro no, teniendo que volver a introducir otra vez ambos números. De esta forma, hasta que el primer número no sea mayor o igual a cero, no se pasa al proceso de introducir el otro número.

Si uno de los números es cero, o ambos lo son, no entra en el ciclo, *hacer mientras num1 < > 0 AND num2 < > 0*, e imprime el valor del producto que es cero.

Si ambos son distintos de cero, suma la variable *num2* tantas veces como valga *num1* y visualiza su resultado.

Ejercicio 2

Obtener el cociente y el resto de dos números enteros positivos mediante restas sucesivas.

ESTUDIO PREVIO

El proceso es análogo al anterior, sólo que, en lugar de sumar, hay que restar hasta que el resto sea menor que el divisor.

De nuevo los números deben ser positivos.

Podemos tener los casos:

- El divisor es cero: no se puede realizar la división.
- El dividendo es cero: el cociente y el resto son cero. No hay que realizar ninguna resta.
- Tanto el dividendo como el divisor son distintos de cero: al dividendo, mientras sea mayor o igual al divisor, se le disminuye en el valor del divisor. Cuando sea más pequeño que el divisor no se puede restar y su valor será el resto.

VARIABLES

num1 = almacena el dividendo.
num2 = almacena el divisor.
cociente = guarda el cociente de la división.

PSEUDOCODIGO

```
num1 = -1
hacer PRIMER-NUMERO
num2 = -1
hacer SEGUNDO-NUMERO
cociente = 0
borrar pantalla
if num2 = 0
     en 10,20 imprime "No se puede dividir por cero"
```

```
else
     hacer mientras num1 >= num2
          num1 = num1 - num2
          cociente = cociente + 1
     fin del hacer
     en 10,20 imprime "La división de: ",num1
     en 12,20 imprime "por ",num2
     en 14,20 imprime "cociente  ",cociente
     en 16,20 imprime "resto ",num1
fin del if
fin del programa.

PRIMER-NUMERO.
hacer mientras num1 < 0
     borrar pantalla
     en 10,25 imprime "Dividendo:"
     en 10,33 introduce num1
fin del hacer.

SEGUNDO-NUMERO.
hacer mientras num2 < 0
     borrar pantalla
     en 10,25 imprime "Divisor:"
     en 10,33 introduce num2
fin del hacer.
```

COMPROBACION

El proceso de introducción de ambos números es igual que antes.
El divisor, *num2*, puede ser:

— Igual a cero: no se puede hacer la división.
— Distinto de cero: pero el dividendo, *num1*, puede ser:
  a) Cero: tanto el cociente como el resto son cero.
  b) Distinto de cero: realiza la división mediante restas, al dividendo lo disminuye en una cantidad igual al divisor y aumenta en una unidad el cociente.

Ejercicio 3

Introducir una serie de nombres de personas con su sueldo. Imprimir el nombre y el sueldo de la persona que más gana y de la que menos. Si hay varias con igual sueldo, imprimir la primera de todas. Contar el número de personas que introducimos.

ESTUDIO PREVIO

Es necesario utilizar variables auxiliares para guardar el nombre y el sueldo de las personas con mayor y menor sueldo.

Cada vez que se introduzca un dato nuevo se comparará con las variables auxiliares del sueldo para saber si éstos son mayores o menores de los que hasta ese momento había.

Para imprimir los datos de la primera persona que introdujimos, en el caso que hubiera dos o más con el mismo sueldo, hay que preguntar si el sueldo es mayor o menor que las variables auxiliares.

Si preguntamos si es mayor o igual, o menor o igual, tomará los datos de la última persona.

Las variables del mayor y menor sueldo tendrán que estar inicializadas a los valores menor y mayor, respectivamente.

Formato de impresión de los resultados:

```
La persona que más gana es: ----------
su sueldo asciende a: -------

La de menor sueldo es: ----------
que sólo gana: -------

Datos introducidos: --
```

VARIABLES

res = para seguir introduciendo datos.
nombre = introducir el nombre de la persona.
sueldo = introducir el sueldo de la persona.
maxnom = almacena el nombre de la persona con mayor sueldo.
maxsuel = almacena el sueldo mayor.
minnom = almacena el nombre de la persona con menor sueldo.
minsuel = almacena el sueldo menor.
con = contador del número de personas introducidas.

PSEUDOCODIGO

```
res = "S"
nombre = espacios(30)
maxnom = espacios(30)
minnom = espacios(30)
sueldo = 0
maxsuel = 0
minsuel = 999999
con = 0
hacer mientras res = "S"
     borrar pantalla
     en 10,25 imprime "Nombre:"
     en 10,33 introduce nombre
     en 12,25 imprime "S
```

```
        con = con + 1
        if sueldo > maxsuel
             maxnom = nombre
             maxsuel = sueldo
        fin del if
        if sueldo < minsuel
             minnom = nombre
             minsuel = sueldo
        fin del if
        res = espacios(1)
        hacer MAS
    fin del hacer
    borrar pantalla
    en 10,25 imprime "La persona que más gana es:",maxnom
    en 11,25 imprime "su sueldo asciende a ",maxsuel
    en 15,25 imprime "La de menor sueldo es:",minnom
    en 16,25 imprime "que sólo gana ",minsuel
    en 20,25 imprime "Datos introducidos:",con
    fin del programa.
```

COMPROBACION

Se estarán introduciendo datos mientras deseemos, *hacer mientras res="S"*.

El ordenador espera a que le introduzcamos los datos del nombre y del sueldo.

El sueldo se compara con el sueldo máximo, *if sueldo > maxsuel;* en el caso de cumplir la condición el sueldo máximo será el que acabamos de introducir, *maxsuel=sueldo,* y al mismo tiempo guardamos el nombre de la persona, *maxnom=nombre.*

De forma idéntica procederemos con el mínimo.

Cuando no deseemos introducir más datos imprime los datos y sueldos de las personas con mayor y menor sueldo, así como el número de datos introducidos.

Ejercicio 4

Intentar adivinar un número entre 1 y 100, generado aleatoriamente por el ordenador, indicando en cada momento el intervalo en el que se encuentra el número. Imprimir el número de intentos que hemos necesitado para adivinarlo.

Repetir el proceso tantas veces como queramos.

ESTUDIO PREVIO

Para indicar el intervalo en el cual está comprendido el número a adivinar se necesitan dos variables que nos lo indiquen: una el extremo inferior y otra el superior. Al principio valdrán 1 y 100.

Si no acertamos con el número, hay que modificar los extremos, bien el inferior o el superior, que pasarán a valer el número introducido. Este proceso se debe estar repitiendo

mientras no acertemos el número. Cuando demos con él, hay que salir del ciclo, para lo cual necesitaremos una variable, un *sw*.

Presentación en pantalla:

El número a adivinar está entre -- y ---
Número: ---

VARIABLES

numero = introducir número.
num = número generado aleatoriamente por el ordenador.
mayor = número mayor del intervalo.
menor = número menor del intervalo.
i = número de intentos.
sw = salir del bucle cuando acertemos el número.
res = para seguir intentando adivinar más números.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     num = int(rnd*100)+1
     sw = 0
     i = 0
     menor = 1
     mayor = 100
     hacer mientras sw <> 1
          borrar pantalla
          en 10,20 imprime "El número a adivinar esta entre",menor
          en 10,55 imprime "y ",mayor
          en 12,20 imprime "Número"
          en 12,28 introduce numero
          i = i + 1
          if num = numero
               sw = 1
          else
               if num > numero
                    menor = numero
               else
                    mayor = numero
               fin del if
          fin del if
     fin del hacer
     en 14,20 imprime "Número de intentos",i
```

```
    borrar pantalla
    hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

Antes del ciclo, *hacer mientras sw < > 1*, se deben inicializar las variables, asignar los valores extremos y hacer que el ordenador genere un número por si se desea intentar adivinar otro número.

El proceso de adivinar el número tendrá que realizarse hasta acertarlo; por tanto, irá dentro del bucle, *hacer mientras sw< >1*. Al introducir un número pueden suceder los siguientes casos:

- Se acierta el número: hay que abandonar el bucle inmediatamente, *sw=1*.
- El número es más pequeño: el extremo superior del intervalo pasará a ser dicho número, *mayor=numero*.
- El número es mayor: se reasigna el extremo inferior, *menor=numero*.

Ejercicio 5

Hacer un programa que calcule las raíces de una ecuación de segundo grado. Repetir el proceso tantas veces como se desee.

ESTUDIO PREVIO

Las ecuaciones de segundo grado son de la forma:

$$a x^2 + b x + c = 0$$

donde a, b y c son los coeficientes.

El cálculo de las raíces se realiza aplicando la fórmula:

$$x = \frac{-b \pm (b^2 - 4 a c)^{1/2}}{2 a}$$

Según el valor de lo situado dentro de la raíz, el discriminante, tendremos los casos:

— Cero: existe una raíz doble.
— Mayor que cero: posee dos raíces reales distintas.
— Menor que cero: posee dos soluciones imaginarias conjugadas. Para poder realizar la raíz cuadrada se cambia al discriminante de signo.

VARIABLES

a = coeficiente de segundo grado.
b = coeficiente de primer grado.

c = término independiente.
res = repetir el proceso.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     en 8,10 imprime "Coeficiente de segundo grado:"
     en 8,41 introduce a
     en 10,10 imprime "Coeficiente de primer grado:"
     en 10,41 introduce b
     en 12,10 imprime "Término independiente:"
     en 12,34 introduce c
     if (b*b-4*a*c) = 0
          en 14,25 imprime "Raíz doble:"
          en 14,38 imprime -b/(2*a)
     else
          if  (b*b-4*a*c) > 0
               en 14,25 imprime "Raíces distintas de valor:"
               en 16,25 imprime (-b+(b*b-4*a*c)^(1/2))/(2*a)
               en 18,25 imprime (-b-(b*b-4*a*c)^(1/2))/(2*a)
          else
               en 14,25 imprime "Raíces imaginarias:"
               en 16,25 imprime (-b+(-(b*b-4*a*c))^(1/2))/(2*a)
               en 18,25 imprime (-b-(-(b*b-4*a*c))^(1/2))/(2*a)
          fin del if
     fin del if
     hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

Después de introducir los valores de los coeficientes se comprueba el valor del discriminante. Según tenga un valor u otro calcula los valores de las soluciones de la ecuación.

Ejercicio 6

Hacer un programa que determine cuántas cifras posee un número entero positivo introducido por teclado. Repetir el proceso.

ESTUDIO PREVIO

El paso de una unidad a otra inmediatamente superior se realiza multiplicando por 10.

Para saber el número de cifras procederemos en orden inverso, es decir, dividiendo por 10. Este proceso será repetitivo hasta que el número sea menor de 10. Pues el número debe ser positivo; por tanto, como mínimo tendrá una cifra.

Presentación en pantalla:

```
Número: ----

Número de cifras: --
```

VARIABLES

num = introducir el número.
c = almacena el número de cifras de num.
res = repetir el proceso.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
      num = -1
      hacer NUMERO
      c = 1
      hacer mientras num >= 10
            num = num/10
            c = c + 1
      fin del hacer
      en 14,25 imprime "Número de cifras:",c
      hacer MAS
fin del hacer
fin del programa.

NUMERO.
hacer mientras num < 0
      borrar pantalla
      en 10,25 imprime "Número"
      en 10,32 introduce num
fin del hacer.
```

COMPROBACION

El subprograma *NUMERO* debe realizarlo hasta que el número introducido sea positivo.

La realización del ciclo, *hacer mientras num>=10*, depende del valor del número introducido; si es:

— Menor de 10, sólo posee una cifra. No entra en el bucle.

— Mayor o igual a 10, entra en el ciclo. Divide al número por 10, cuenta una cifra más.

Cuando el número que resulta de dividir sea más pequeño de 10 sale del bucle e imprime el número de cifras.

Ejercicio 7

Calcular la suma de los *n* primeros términos de una progresión aritmética, dada por el valor de su primer término y la diferencia entre dos términos consecutivos cualesquiera.

ESTUDIO PREVIO

En una progresión aritmética, cada uno de los términos, excepto el primero, se obtiene del anterior sumándole una cantidad, llamada diferencia.

$$a_n = a_{n-1} + d$$

Se introducirán por teclado el número de términos que deseamos sumar, el valor del primero de ellos y la diferencia entre dos términos consecutivos.

Para generar los términos se suma al primero de ellos el valor de la diferencia. Este proceso será repetitivo hasta que no hayamos sumado tantos términos como le indicamos.

El número de términos debe ser positivo.

VARIABLES

n = número de términos a sumar.
primero = primer número de la progresión en una primera instancia y posteriormente los demás términos de la progresión.
d = diferencia entre dos términos.
suma = suma de los términos de la progresión.

PSEUDOCODIGO

```
Borrar pantalla
suma = 0
primero = 0
d = 0
n = -1
en 10,20 imprime "Primer término:"
en 10,36 introduce primero
en 12,20 imprime "Diferencia entre dos términos:"
en 12,52 introduce d
en 14,20 imprime "Número de términos:"
hacer TERMINOS
```

```
hacer mientras n >= 0
     suma = suma + primero
     primero = primero + d
     n = n - 1
fin del hacer
en 16,20 imprime "La suma es: ",suma
fin del programa.

TERMINOS.
hacer mientras n < 0
     en 14,40 introduce n
fin del hacer.
```

COMPROBACION

Después de introducir los tres datos, teniendo en cuenta que *n* debe ser mayor que cero, para lo cual se inicializa a un número negativo y tiene que realizar el subprograma TERMINOS mientras sea menor que cero.

Se suman los términos, *hacer mientras n >= 0,* si *n* es:

- Cero: no entra en el ciclo e imprime que la suma es cero.
- Mayor que cero: acumula el valor *primero* en *suma,* luego calcula el siguiente término, guardándolo también en *primero,* y decrece en una unidad el número de sumas que quedan por realizar.

Cuando salga imprime el valor de la suma.

Ejercicio 8

Hacer un programa que escriba los *n* primeros números de la sucesión de Fibonacci. El primer número de la serie es 0, el segundo es 1 y cada uno de los siguientes es la suma de los dos anteriores a dicho número.

ESTUDIO PREVIO

La sucesión de Fibonacci es:

0, 1, 1, 2, 3, 5, 8, 13, 21 ...

Los dos primeros términos de la sucesión se introducen en dos variables.

El proceso de generar e imprimir los términos se repetirá tantas veces como se le haya indicado.

En cada iteración, ciclo, se calcula el siguiente término sumando los dos anteriores y reasignando los valores de las variables. Por tanto, se necesita una nueva variable para realizar la suma de los términos e intercambiar los elementos.

VARIABLES

a = almacena un término de la sucesión, al principio el primero.
b = almacena otro término, al principio el segundo.
c = almacena la suma de los dos anteriores. También para poder intercambiar los elementos.
n = número de términos a generar.

PSEUDOCODIGO

```
n = 0
a = 0
b = 1
c = 0
borrar pantalla
en 10,20 imprime "Número de términos"
en 10,39 introduce n
hacer mientras n > 0
     imprime a
     c = a + b
     a = b
     b = c
     n = n - 1
fin del hacer
fin del programa.
```

COMPROBACION

Los dos primeros términos de la serie se almacenan en la variable *a* y *b;* por eso deben tomar los valores de 0 y 1, respectivamente.

Se introduce el número de términos que deseamos imprimir; si es:

- Cero o negativo: no entra en el ciclo. No imprime nada.
- Mayor que cero: entra en el ciclo. Visualiza el valor de *a*. Primero calcula el nuevo término, *c=a+b*, y reasigna los valores de las variables. En *a* se guarda el valor del siguiente término a imprimir, que está en *b*. En esta última variable se almacena el nuevo elemento generado en *c*. Decrementa en una unidad el número de términos que aún quedan por imprimir.

Ejercicio 9

Hacer un programa que acepte una sucesión de fechas lógicas.

ESTUDIO PREVIO

Introduciremos por teclado, en este orden, el año, el mes y el día.

El mes estará comprendido entre los valores 1 y 12. Mientras no esté entre estos valores habrá que seguir introduciendo su valor. Hay que utilizar un *sw*.

Una vez que el mes tiene un valor lógico, se introduce el día, el cual depende del año, si es bisiesto o no, y del mes. Mientras su valor no se corresponda con estos datos habrá que seguir introduciéndolo por teclado. Se controla con un *sw*, el mismo que se utiliza para el mes.

El año es bisiesto si es divisible por cuatro.

Formato de pantalla:

```
Año: ----

Mes: --

Día: --
```

VARIABLES

dia = almacena el día.
mes = introducir el mes.
aa = guarda el año.
res = seguir introduciendo más fechas.
sw = controlar que el *mes* y el *dia* sean correctos.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
      aa = 0
      borrar pantalla
      en 10,20 imprime "Año:"
      en 10,25 introduce aa
      en 12,20 imprime "Mes:"
      sw = 0
      hacer mientras sw = 0
            mes = 0
            en 12,25 introduce mes
            if mes >= 1 AND mes <= 12
                  sw = 1
            fin del if
      fin del hacer
      sw = 0
      en 14,20 imprime "Día:"
      hacer mientras sw = 0
            dia = 0
            en 14,25 introduce dia
            if dia > 0 AND dia <= 31
```

```
            if mes = 2 AND aa = int(aa/4)*4 AND dia < = 29
                sw = 1
            fin del if
            if mes = 2 AND aa < > int (aa/4)*4 AND dia < = 28
                sw = 1
            fin del if
            if (mes = 4 OR mes = 6 OR mes = 9 OR mes =11) AND dia < = 30
                sw = 1
            fin del if
        fin del if
    fin del hacer
    res = espacios(1)
    hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

En primer lugar se introduce el año.

A continuación el mes; mientras no esté comprendido entre 1 y 12, se debe repetir el proceso, *hacer mientras sw=0:*

— Es correcto: hay que salir del bucle, *sw=1.*
— No es correcto: introducirlo de nuevo. el *sw* sigue a 0.

Cuando el mes sea correcto se introduce el día y se reasigna el valor del sw, *sw=0*, pues con la misma variable se controlan los días.

El número máximo de días de un mes será de 31 y, por supuesto, siempre mayor que 0. *if dia>0 AND dia<=31.* Pueden suceder estos dos casos:

— No está entre dichos valores: volver a introducirlo.
— Está: queda por comprobar el mes de Febrero y los que poseen 30 días.

Si el año es bisiesto. el mes 2 puede tener 29 días; en caso contrario, como máximo deberá tener 28. No se puede utilizar un ELSE porque en el IF hay varias condiciones: en caso de no cumplirse alguna. no sabemos a cuál se refiere. Por tanto, hay que hacer dos IF para Febrero.

Cuando el día sea correcto se pone el *sw=1* para salir del ciclo.

Tanto *mes* como *dia* están reinicializadas dentro del ciclo para blanquearlas. para que cuando vuelva a pedir los datos, en el caso de que no fueran lógicas, aparezcan en blanco.

Los comentarios de *imprime "Mes"* e *imprime "Día"* van fuera de los ciclos respectivos para que sólo lo realice una vez.

La introducción del mes también puede realizarse de la siguiente forma:

```
mes= 0
hacer mientras mes < 1 OR mes > 12
    mes= 0
    en 12,25 introduce mes
fin del hacer.
```

RESUMEN

Los **bucles anidados** los expresamos:

Hacer mientras condición-1
. . . . . .
. . . . . .
Hacer mientras condición-2
. . . . . .
. . . . . .
Fin del hacer
. . . . . .
. . . . . .
Fin del hacer

El formato de las instrucciones condicionales **IF–ELSE** anidadas es:

IF condición-1
. . . . . .
. . . . . .
ELSE
IF condición-2
. . . . .
. . . . .
ELSE
. . . . .
. . . . .
Fin del IF
Fin del IF

Para la instrucción **CASE** el formato es:

Hacer CASE
case condición-1
. . . . .
case condición-2
. . . . .
. . . . .
. . . . .
. . . . .
[en caso contrario]
[. . . .]
Fin del CASE

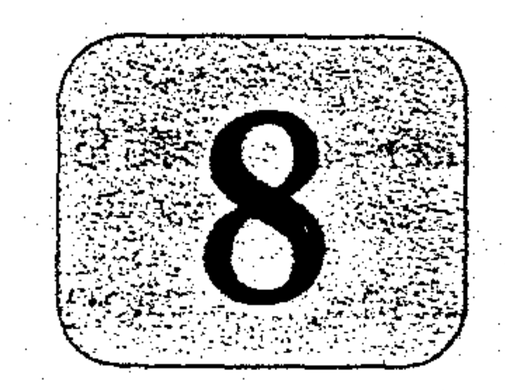
EJERCICIOS PROPUESTOS

1. Diseñar un programa que calcule los múltiplos comunes de 4 y 7 menores que 1000 y mayores de 300.

2. Hacer un programa que compruebe si dos números son amigos.
   Dos números son amigos si la suma de los divisores del primero, excepto él, es igual al segundo número, y viceversa.

3. Calcular e imprimir los números perfectos menores que un número introducido por teclado.
   Un número es perfecto si la suma de sus divisores, excepto él mismo, es igual al propio número.
4. Crear un programa que nos escriba el día completo de la semana si se introducen por teclado los dos primeros caracteres.
5. Se pretende hacer un programa que acepte una fecha con el formato *dd mm aa* (día, mes, año) y la imprima en su formato normal (el mes en letra y el año con todas sus cifras).
6. Calcular el menor número primo mayor que un número introducido por teclado.
7. Calcular la última cifra del cuadrado de un número y el número de cifras.
8. Hacer un programa que escriba tres números en orden creciente.
9. Realizar la tabla de multiplicar del 0 al 10 visualizando en cada pantalla la tabla de dos números.
10. Introducir una serie de números y escribirlos en binario.

CAPITULO



Arrays unidimensionales o listas

8.1. INTRODUCCION. ESTRUCTURAS DE DATOS

Hasta ahora, para hacer referencia a un dato utilizábamos una variable.

El problema se plantea cuando tenemos gran cantidad de datos que guardan entre sí una relación. Para cada uno de estos datos se debería utilizar una variable distinta, lo que acarrea una gran laboriosidad a la hora de construir el programa, unida a la cantidad de variables a usar.

Para resolver estas dificultades se agrupan los datos en un mismo conjunto, bajo un nombre común, que se pueden tratar como una sola unidad.

Estos conjuntos reciben el nombre de **estructuras de datos.**

Tipos de estructuras

Las estructuras de datos, dependiendo del lugar donde residan o se almacenen, se clasifican en:

- **Internas:** residen en la memoria del ordenador, ejemplo: los arrays.
- **Externas:** residen en un soporte externo, ejemplo: los ficheros que se almacenan en dispositivos magnéticos.

8.2. ESTRUCTURAS INTERNAS DE DATOS. TIPOS

Un **array** es una estructura interna de datos con un conjunto de elementos homogéneos del mismo tipo, numérico o alfanumérico, reconocidos por un nombre en común, que residen en la memoria del ordenador.

A cada elemento se accede por la posición que ocupa dentro del conjunto de datos.

Según el tipo de datos almacenados en los arrays, éstos pueden ser:

- **Numéricos.**
- **Alfanuméricos.**

Antes de poder utilizarlos, hay que reservar una zona de la memoria para su uso, así como definir el número de parámetros necesarios para acceder a cada elemento de la estructura, es decir, *dimensionarlos.*

Frente a la *ventaja* de poder acceder directamente a sus datos y el escaso tiempo que emplean para tal fin, presentan el grave *inconveniente* de que al apagar el ordenador sus datos se pierden.

Según el número de parámetros necesarios para dimensionar los arrays, se pueden clasificar en los siguientes tipos:

- **Unidimensionales.**
- **Bidimensionales.**
- **Multidimensionales.**

8.3. ARRAYS UNIDIMENSIONALES

También se llaman *listas, vectores* o simplemente *arrays.*

Un **array** es un conjunto de *n* datos homogéneos, cada uno de los cuales debe referenciarse por un índice, I, a veces llamado *subíndice,* que debe cumplir la propiedad de que:

$$1 \leqslant I \leqslant n.$$

Al proceso de reservar una zona de la memoria para albergar los datos del array se llama *dimensionar* el array.

Para **dimensionar un array** lo expresamos:

DIMENSIONA NOMBRE-LISTA(INDICE)

NOMBRE-LISTA: hace referencia al nombre del array. Debe cumplir la misma normativa que el nombre de las variables.

INDICE: es el número máximo de elementos que va a contener el array. El índice va entre paréntesis.

Ejemplo: dimensiona nombres(20). Indicamos al ordenador que reserve una zona de su memoria, que reconocerá como *nombres,* para almacenar 20 elementos.

Cuando el ordenador ejecute el programa debe conocer exactamente la dimensión del array.

Las listas sólo se dimensionan una vez. Irán en la parte inicial del pseudocódigo.

Los **elementos del array** vendrán referenciados de la siguiente forma:

NOMBRE-LISTA(I)

- I, el índice, es la variable que toma el valor de la posición en la cual está el elemento al que deseamos acceder.

Tiene como límite inferior 1 y como límite superior el valor de *INDICE.*

Sólo puede tomar valores enteros.

Si el índice sobrepasa la dimensión del array, se produce un error.

La posición del array a la que deseamos acceder va entre paréntesis.

Podemos imaginar un array como una fila o columna de cajas vacías idénticas, colocadas una al lado de la otra en la memoria del ordenador. El programa dispone las cajas al

dimensionar el array, pero es trabajo del programador el llenarlas y manipular sus datos, tal como indica la Figura 8.1.

Posición (índice)	NOMBRES
1	Luis Redondo
2	Antonio
3	Felipe
4	Ana Herrera
5	Patricia
6	Angel Ayllón
.	.
.	.
20	Rosario

NOMBRES(1): sería el primer elemento del grupo de 20. Su valor es "Luis Redondo".

NOMBRES(I): sería el I-ésimo elemento del grupo de 20. La variable I debe contener un valor entero entre 1 y 20.

Figura 8.1.

Conviene distinguir entre posición y elemento.

— La **posición** es el lugar que ocupa cada caja en el conjunto del array.
— El **elemento** es el dato que hay almacenado en dicha caja, en esa posición

Ejemplo:

```
i = 5
imprime nombres(i)
```

El ordenador se situará en la posición 5, por valer *i=5*, imprimirá el dato del array *nombres* que está en dicha posición, es decir, *Patricia*.

Esto mismo se puede poner de la forma:

```
imprime nombres(5)
```

El resultado es *Patricia*.

Los arrays, al residir en la memoria del ordenador, son de acceso directo: podemos situarnos directamente en la posición deseada.

Los elementos de una lista se utilizan de una forma análoga a la de cualquier variable; intervendrán en instrucciones de:

— **Asignación:** *variable=nombres(i)* o *lista1(k)=lista2(k)*.
— **Entrada:** *introduce nombres(j)*.
— **Salida:** *imprime lista1(i)*.

8.4. ESTRUCTURAS DE PROGRAMACION PARA OPERAR CON LISTAS. CICLO: REPETIR ... HASTA

En los *arrays* siempre conocemos de antemano los límites entre los que varía el subíndice I para realizar su recorrido.

En estos casos, en lugar de utilizar el ciclo *hacer mientras*, podemos usar el ciclo *repetir ... hasta*.

Sintaxis:

Repetir desde índice=valor–inicial hasta valor–final
instrucciones
Fin del repetir

Las instrucciones dentro del ciclo repetir se ejecutan varias veces, con un valor de *índice* variante, desde el *valor inicial* hasta el *valor final*.

Ejemplos de ciclos repetir: REPEAT, FOR–NEXT.

Para imprimir los 20 primeros números utilizando el ciclo *repetir...hasta* el pseudocódigo sería:

```
i = 1
repetir desde i = 1 hasta 20
   imprime i
   i = i + 1
fin del repetir
```

Ejercicio 1

Crear una lista unidimensional de 20 elementos con nombres de personas.

ESTUDIO PREVIO

Al no estar creada la lista, en primer lugar hay que dimensionarla.

Para introducir los datos, sólo necesitamos una variable, el índice, el cual recorrerá las distintas posiciones del array. Los datos, los nombres, se guardan en dichas posiciones.

El índice es como un ascensor que va recorriendo el array. El array es el edificio.

VARIABLES

i = índice para recorrer la lista desde la posición 1 hasta la 20.
datos = nombre de array, almacena valores alfanuméricos.

PSEUDOCODIGO

```
Dimensiona datos(20)
i = 1
```

```
Hacer mientras i<21
     borrar pantalla
     en 10,10 imprime "Nombre:"
     en 10,18 introduce datos(i)
     i = i + 1
fin del hacer
fin del programa.
```

COMPROBACION

En primer lugar, se dimensiona el array: *dimensiona datos(20)* y se inicializa la variable *i*.

El proceso de introducir los nombres en el array es repetitivo, *hacer mientras i<21*.

Borrar pantalla va dentro del ciclo, para que a la hora de introducir los datos esté limpia.

En la fila 10, columna 10, visualiza el comentario, *Nombre;* en la posición 10,18 se introduce el nombre por teclado que se almacena en la lista *datos* en la posición *i*.

Utilizando el ciclo *repetir* el pseudocódigo sería:

```
Dimensiona datos(20)
i = 1
Repetir desde i = 1 hasta 20
     borrar pantalla.
     en 10,10 imprime "Nombre:"
     en 10,18 introduce datos(i)
     i = i + 1
fin del repetir
fin del programa.
```

Ejercicio 2

Imprimir los elementos situados en las posiciones 5, 18 y 3 (por este orden) de la lista anterior.

ESTUDIO PREVIO

La lista ya está creada; por tanto, no hace falta dimensionarla.

VARIABLE

datos = nombre del array.

PSEUDOCODIGO

```
Borrar pantalla
en 10,30 imprime "El elemento 5 es:"
en 10,50 imprime datos(5)
en 12,30 imprime "El elemento 18 es:"
en 12,50 imprime datos(18)
en 14,30 imprime "El elemento 3 es:"
en 14,50 imprime datos(3)
fin del programa.
```

COMPROBACION

Borramos pantalla e imprimimos directamente los elementos que están en las posiciones deseadas, 5, 18 y 3.

Ejercicio 3

Visualizar los elementos de la lista anterior. Cada elemento debe ir en una fila.

ESTUDIO PREVIO

La lista ya está creada: no hace falta dimensionarla.

Para imprimir los datos necesitamos una variable, que vaya pasando por todas las posiciones del array.

Cada elemento irá en una línea distinta: hay que controlar las filas, para lo cual deben ser variables.

VARIABLES

i = índice para recorrer la lista desde la posición 1 hasta la 20.
fi = contador de filas.

PSEUDOCODIGO

```
Borrar pantalla
i = 1
fi = 3
en 1,20 imprime "Elementos de la lista"
Hacer mientras i<21
     en fi,28 imprime datos(i)
     fi = fi + 1
     i = i + 1
fin del hacer
fin del programa.
```

COMPROBACION

Borrar pantalla debe ir fuera del ciclo para que lo realice una sola vez; si fuera dentro del bucle, sólo se visualizará el último elemento.

Se inicializa las variables *i* y *fi*.

Fuera del ciclo imprimimos la cabecera: "Elementos de ..."

El proceso de visualizar los nombres del array es repetitivo hasta que i=20, *hacer mientras i<21.*

En las coordenadas, *fi,28*, imprimimos el contenido de los elementos de la lista *datos(i)*. Se aumenta la fila para que el nuevo elemento vaya en la siguiente linea e incrementamos la posición del índice.

El ciclo lo podíamos haber expresado también: *repetir desde i=1 hasta 20.*

Ejercicio 4

Visualizar los elementos que ocupan las posiciones pares de la lista anterior. Cada elemento debe ir en una fila distinta.

ESTUDIO PREVIO

Para imprimir los datos que ocupan las posiciones pares, simplemente hay que recorrer la lista comprobando si esa posición es par. En caso de serlo, imprimimos el valor que hay en dicha posición.

Las filas deben de ser variables para que cada elemento vaya en una fila distinta.

VARIABLES

i = índice para recorrer la lista desde la posición 1 hasta la 20.
fi = contador de filas.

PSEUDOCODIGO

```
Borrar pantalla
i = 1
fi = 3
en 1,15 imprime  "Elementos de la lista que ocupan las posiciones pares"
Hacer mientras i<21
     if i=int(i/2)*2
          en fi,28 imprime datos(i)
          fi = fi + 1
     fin del if
     i = i + 1
fin del hacer
fin del programa.
```

COMPROBACION

Se borra pantalla y se inicializan las variables *i* y *fi*.

Fuera del ciclo imprimimos la cabecera: "Elementos de ..." para que lo realice una sola vez.

Comprobamos el valor de la posición, *i=int(i/2)*2:*

— Es par: se imprime el elemento, datos(i), y se aumenta la fila.
— No es par: no lo visualiza.

En ambos casos se incrementa la posición.

También se podía haber hecho de la siguiente forma:

```
Borrar pantalla
i = 2
fi = 3
en 1,15 imprime  "Elementos de la lista que ocupan las posiciones pares"
Repetir desde i=1 hasta 20
     en fi,28 imprime datos(i)
     fi = fi + 1
     i = i + 2
fin del repetir
fin del programa.
```

Se inicializa la variable al primer número par. Dentro del ciclo se aumenta de dos en dos, con lo cual nos evitamos el tener que preguntar si la posición es par o no.

Ejercicio 5

Imprimir y sumar los elementos que ocupan las posiciones pares de la lista numérica DATOS.

ESTUDIO PREVIO

Para imprimir los elementos que ocupan las posiciones pares, se recorre la lista comprobando si esa posición es par. En caso de serlo, imprimimos el valor que hay en dicha posición y se van sumando los elementos.

O también recorriendo las posiciones pares, para lo cual se inicializa el índice a dos y se incrementa de dos en dos.

Las filas deben de ser variables para que cada elemento vaya en una fila distinta.

VARIABLES

sum = acumulador de los elementos que ocupan las posiciones pares.
i = índice para recorrer la lista desde la posición 1 hasta la 20.
fi = contador de filas.

PSEUDOCODIGO

```
Borrar pantalla
sum = 0
i = 1
fi = 3
en 1,15 imprime "Elementos de la lista que ocupan las posiciones pares"
Hacer mientras i<21
     if i=int(i/2)*2
          en fi,28 imprime datos(i)
          fi = fi + 1
          sum = sum + datos(i)
     fin del if
     i = i + 1
fin del hacer
en fi+2,25 imprime "La suma de los elementos es:",sum
fin del programa.
```

COMPROBACION

Se borra pantalla y se inicializan las variables *i* y *fi*.

Fuera del ciclo imprimimos la cabecera: "Elementos de ..." para que lo realice una sola vez.

Comprobamos el valor de la posición, *i=int(i/2)*2:*

— Es par: se imprime el elemento, se aumenta la fila y se suma el elemento.
— No es par: no hace nada.

En ambos casos se incrementa la posición.

Cuando acabe de recorrer todo el array imprime el resultado de la suma, *en fi+2,25*, dos filas más abajo del último elemento y en la columna 25.

También se podía haber hecho de la forma siguiente:

```
Borrar pantalla
sum = 0
i = 2
fi = 3
en 1,15 imprime "Elementos de la lista que ocupan las posiciones pares"
Repetir desde i=2 hasta 20
     en fi,28 imprime datos(i)
     fi = fi + 1
     sum = sum + datos(i)
     i = i + 2
fin del repetir
en fi+2,25 imprime "La suma de los elementos es:",sum
fin del programa.
```

Se inicializa la variable al primer número par. Dentro del ciclo se aumenta de dos en dos, con lo cual nos ahorramos el tener que preguntar si la posición es par o no.

Ejercicio 6

. Visualizar, contar y sumar los elementos pares que ocupan las posiciones impares de la lista anterior. Imprimir las posiciones que ocupan dichos elementos en la lista.

Posición	Elementos
--	---
--	---
La suma es: -----	

ESTUDIO PREVIO

Se recorren las posiciones impares del array, comprobando si el elemento que está en esa posición es par:

*datos(i)=int(datos(i)/2)*2*

En caso de serlo se suma, se imprime y se cuenta.
Las filas deben ser variables.

VARIABLES

sum = acumulador de los elementos pares que ocupan las posiciones impares.
i = índice para recorrer la lista desde la posición 1 hasta la 20.
c = contador de los elementos pares que ocupan las posiciones impares.
fi = contador de filas.

PSEUDOCODIGO

```
Borrar pantalla
sum = 0
i = 1
c = 0
fi = 3
en 1,25 imprime "Posición"
en 1,35 imprime "Elementos"
Hacer mientras i < 21
     if i <> int(i/2)*2   AND   datos(i) = int(datos(i)/2)*2
          en fi,28 imprime i
          en fi,39 imprime datos(i)
          fi = fi + 1
          c = c + 1
          sum = sum + datos(i)
     fin del if
     i = i + 1
```

```
fin del hacer
en fi+2,25 imprime "La suma es:",sum
fin del programa.
```

COMPROBACION

Se borra pantalla y se inicializan las variables *i* y *fi*.

Fuera del ciclo imprimimos la cabecera.

Comprobamos si la posición es impar y además si el elemento es par, *if i < > int(i/2)*2 AND datos(i)=int(datos(i)/2)*2;* en caso de cumplirse ambas condiciones:

— Imprime la posición: *en fi,28 imprime i.*
— Imprime el elemento: *en fi,39 imprime datos(i).*
— Cuenta y suma.

Tanto si cumple o no la condición, se incrementa la posición.

Cuando salga del bucle visualiza el resultado de la suma.

También se podía haber hecho de la siguiente forma:

```
Borrar pantalla
sum = 0
i = 1
c = 0
fi = 3
en 1,25 imprime "Posición"
en 1,35 imprime "Elementos"
Repetir desde i=1 hasta 20
     if datos(i) = int(datos(i)/2)*2
          en fi,28 imprime i
          en fi,39 imprime datos(i)
          fi = fi + 1
          c = c + 1
          sum = sum + datos(i)
     fin del if
     i = i + 2
fin del repetir
en fi+2,25 imprime "La suma es:",sum
fin del programa.
```

La variable para recorrer las posiciones impares de la lista se inicializa al primer número impar y se incrementa de dos en dos dentro del bucle.

Sólo pasamos por las posiciones impares.

Para comprobar si los elementos son pares sólo hace falta una condición. *if datos(i)=int(datos(i)/2)*2.*

Ejercicio 7

Visualizar los elementos de la lista L(200) de la siguiente forma:

Elementos de la lista				
----	----	----	----	----
----	----	----	----	----

es decir, cinco elementos en cada fila.

ESTUDIO PREVIO

La lista ya está creada; no hace falta dimensionarla.

Para visualizar los datos, hay que controlar tanto las filas como las columnas, pues se desea que vayan cinco elementos en una fila. No todos los elementos entran en una pantalla. Cuando llegue a una fila determinada, por ejemplo la 20, hay que detener el listado de los elementos mientras lo deseemos, para poder ver los que están en pantalla.

Cuando continúe ejecutándose el programa borramos pantalla para eliminar los datos anteriormente visualizados para que no se mezclen los datos de ambas pantallas. Imprimimos de nuevo la cabecera. Esta debe ir dentro del bucle, pero sólo debe aparecer una vez por pantalla.

Las filas se inicializarán al valor en el cual se producirá el borrado de pantalla.

VARIABLES

i = variable índice para recorrer la lista desde la posición 1 hasta la 200.
fi = contador de filas.
co = contador de columnas.

PSEUDOCODIGO

```
i = 1
fi = 23
co = 6
Repetir desde i=1 hasta 200
     if fi = 23
          hacer CABECERA
     fin del if
     en fi,co imprime L(i)
     if co = 46
          fi = fi + 1
          co = 6
     else
          co = co + 10
```

```
    fin del if
    if fi = 20
        en 23,20 imprime "Pulse ENTER para continuar"
        detener-listado
        fi = 23
    fin del if
    i = i + 1
fin del repetir
fin del programa.

CABECERA.
Borrar pantalla
en 1,20 imprime "Elementos de la lista"
fi = 3.
```

COMPROBACION

Borrar pantalla debe ir dentro del subprograma CABECERA para que sólo lo realice cuando pasamos a la siguiente pantalla.

Se inicializan las varibles *i, fi* y *co.*

El proceso de visualizar los nombres del *array* es repetitivo. *repetir desde i=1 hasta 200.*

Preguntamos por el valor del contador de las filas *¿fi=23?:*

- Sí: borra pantalla, hace el subprograma CABECERA y asigna a *fi* el valor de 3 para que empiece a visualizar los elementos desde esta fila.
- No: no realiza la cabecera.

Controlamos las columnas. *¿co=46?:*

- No: aumenta 10 a *co* para que el siguiente elemento lo visualice en la misma línea pero 10 columnas más a la derecha.
- Sí: ha impreso los cinco elementos en la misma fila; tiene que empezar a imprimirlos en la línea siguiente, *fi=fi+1*, empezando en la misma columna, *co=6.*

Al llegar a la fila 20, el listado debe detenerse para poder ver los elementos en pantalla. Imprime el comentario «Pulse...» y se detiene el programa hasta que pulsemos ENTER.

Los siguientes elementos deben aparecer en una nueva pantalla e imprimir la cabecera. Se asigna al contador de las filas el valor 23,*fi=23*, para que el ordenador ejecute de nuevo el subprograma CABECERA.

RESUMEN

Una **estructura de datos** es un conjunto de datos, bajo un nombre común, que se pueden tratar como una sola unidad.

Según donde residan se clasifican en:

— Internas: residen en la memoria del ordenador: por ejemplo, las listas.
— Externas: residen en un soporte externo: por ejemplo, los ficheros que se almacenan en disco.

Las **estructuras internas de datos** son un conjunto de elementos reconocidos por un nombre en común que residen en la memoria del ordenador.

A cada elemento se accede por la posición que ocupa dentro del conjunto de datos.

Según el tipo de datos, las estructuras internas de datos pueden ser:

— Numéricas.
— Alfanuméricas.

Antes de poder utilizarlas hay que reservar una zona de la memoria para su uso, es decir, *dimensionarlas.*

Según su dimensión las estructuras internas de datos pueden ser:

— Unidimensionales.
— Bidimensionales.
— Multidimensionales.

Las **listas unidimensionales** o **arrays** o **vectores** son un conjunto de *n* datos homogéneos, cada uno de los cuales debe referenciarse por un índice, *I*, a veces llamado *subíndice*, que debe cumplir la propiedad de que:

$$1 \leqslant I \leqslant n.$$

Al proceso de reservar una zona de la memoria para albergar los datos del array se llama **dimensionar** el array, y se expresa:

DIMENSIONA NOMBRE-LISTA(INDICE)

NOMBRE-LISTA: hace referencia al nombre del array. Debe cumplir la misma normativa que el nombre de las variables.

INDICE: es el número máximo de elementos que va a contener el array. Va entre paréntesis.

Cuando el ordenador ejecute el programa debe conocer exactamente la dimensión del array.

Los arrays sólo se dimensionan una vez. Van en la parte inicial del pseudocódigo.

Los **elementos** del array vendrán referenciados de la siguiente forma:

NOMBRE-LISTA(I)

I, el índice, es la variable que toma el valor de la posición en la cual está el elemento al que deseamos acceder.

Tiene como límite inferior 1 y como límite superior el valor de INDICE.

I sólo puede tomar valores enteros y no puede sobrepasar la dimensión del array.

La posición del array a la que deseamos acceder va entre paréntesis.

La **posición** de un elemento dentro de un array es el lugar que ocupa dicho elemento en el conjunto del array.

Elemento es el dato que hay almacenado en dicha posición.

Los elementos de los arrays intervienen en operaciones de:

— *Asignación:* variable = nombres(i) o lista1(k) = lista2(k).
— *Entrada:* introduce nombres(j).
— *Salida:* imprime lista1(i).

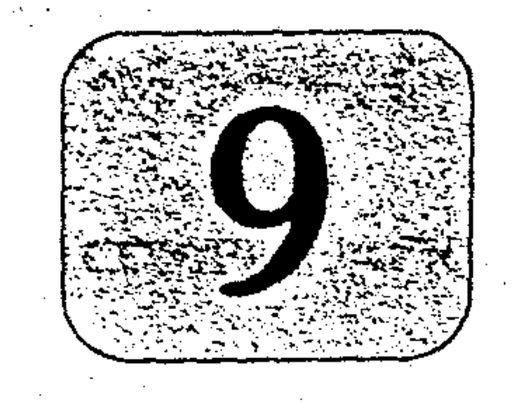
El ciclo **repetir ... hasta** se utiliza cuando conocemos de antemano el número de repeticiones a efectuar. Tiene especial aplicación en los arrays. Lo expresamos:

Repetir desde índice=valor-inicial hasta valor-final
instrucciones
Fin del repetir

EJERCICIOS PROPUESTOS

1. Imprimir los elementos pares de la lista numérica L(50).
2. Imprimir y sumar los elementos que ocupan las posiciones impares de la lista anterior.
3. Imprimir, sumar y contar los elementos impares que ocupan las posiciones pares de la lista anterior.
4. Imprimir los elementos así como la posición que ocupan los elementos de la lista anterior, que sean al mismo tiempo múltiplos de 2 y de 3. Calcular su suma.
5. Generar e imprimir una lista con la tabla de multiplicar de un número determinado que se introducirá por teclado. Imprimir el elemento que ocupa la tercera posición.
6. Imprimir la lista del ejercicio anterior empezando a recorrerla desde la última posición.
7. Crear una lista unidimensional con 5 frases que se introducirán por teclado.
   a) Imprimir la lista.
   b) Imprimir la frase de mayor longitud y el lugar que ocupa.
   c) Imprimir la frase más pequeña y el lugar que ocupa.
8. Crear e imprimir una lista unidimensional de 50 elementos con números aleatorios comprendidos entre 1 y 100, de tal forma que no se repita ninguno. Imprimir los elementos que ocupan las posiciones 3 y 40.
9. Crear aleatoriamente una lista unidimensional de 50 elementos con números aleatorios comprendidos entre 1 y 100, de tal forma que no se repita ningún elemento, y no puede haber ningún elemento que sea múltiplo de la posición que ocupa.

CAPITULO



Número de elementos variables en las listas

9.1. NUMERO DE ELEMENTOS VARIABLES EN LAS LISTAS

Al dimensionar las listas unidimensionales con un número determinado de elementos estamos limitando nuestro problema a dicho número y no lo podemos cambiar; ejemplo: datos(20), 1(200), nombres(50), etc.

Muchas veces, dentro de un mismo programa, podemos necesitar un número determinado de elementos y cuando volvamos a ejecutarlo necesitar otro número distinto de elementos.

Para solucionar este problema se puede dejar la dimensión del array variable e introducir su valor por teclado o igualarla a alguna variable.

Ejercicio 1

Crear una lista con N elementos numéricos. Visualizar cinco elementos en cada fila.

ESTUDIO PREVIO

Dimensionamos la lista desde el teclado por medio de la variable, N.
Todo lo demás es idéntico a lo visto hasta ahora.
Formato de visualización de los elementos del array:

```
Elementos de la lista
---   ---   ---   ---   ---
---   ---   ---   ---   ---
..    ..    ..    ..    ..

Pulse ENTER para continuar
```

VARIABLES

N = dimensión de la lista.
L = nombre de la lista.
i = índice para recorrer la lista.
fi = contador de las filas.
co = contador de las columnas.

PSEUDOCODIGO

```
Borrar pantalla
N = 0
i = 1
en 12,20 imprime "Elementos de la lista"
en 12,42 introduce N
dimensiona L(N)
hacer INTRODUCIR-DATOS
hacer VISUALIZAR-DATOS
fin del programa.

INTRODUCIR-DATOS.
Hacer mientras i < N+1
     borrar pantalla
     en 12,28 imprime "Número:"
     en 12,26 introduce L(i)
     i = i + 1
fin del hacer.

VISUALIZAR-DATOS.
Borrar pantalla
i = 1
fi = 23
co = 6
hacer mientras i < N+1
     if fi = 23
          hacer CABECERA
     fin del if
     en fi,co imprime L(i)
     if co = 46
          fi = fi + 1
          co = 6
     else
          co = co + 10
     fin del if
     if fi = 22
          en 23,20 imprime "Pulse ENTER para continuar"
          detener-listado
          fi = 23
```

```
    fin del if
    i = i + 1
fin del hacer.
```

COMPROBACION

Con los subprogramas INTRODUCIR-DATOS y VISUALIZAR-DATOS el programa principal queda reducido a unas pocas líneas, lo que facilita su lectura.

Aunque los subprogramas tienen más de ocho caracteres, el ordenador sólo considerará los ocho primeros. El problema sería si hubiera más de un subprograma con los ocho primeros caracteres iguales, pues no sabría cuál tendría que realizar.

Primero introducimos la dimensión de la lista.

El subprograma INTRODUCIR-DATOS carga el array con los elementos y el de VISUALIZAR-DATOS los imprime en pantalla, colocando cinco elementos en cada fila; además llama al subprograma CABECERA para hacer las cabeceras al principio de cada pantalla.

Es necesario utilizar dos bucles: uno para introducir los datos y el otro para visualizarlos, por estar introduciendo los elementos desde el teclado.

Utilizando el ciclo *repetir* el subprograma INTRODUCIR-DATOS queda de la forma siguiente:

```
Repetir desde i = 1 hasta N
    borrar pantalla
    en 12,28 imprime "Número:"
    en 12,26 introduce L(i)
    i = i + 1
fin del repetir
```

Ejercicio 2

Pseudocódigo que seleccione el máximo de los números de la lista LISTA(N) y lo lleve a la última posición.

ESTUDIO PREVIO

Recorremos la lista desde la primera posición hasta la última y comparamos los elementos dos a dos:

— Están ordenados: se dejan tal como están.
— No están ordenados: se intercambian de posición mediante un auxiliar, *aux*.

En un caso y en otro se sigue recorriendo la lista.
Sea la lista:

9 7 6 5 3 4 11 8

Comparamos el primer elemento con el segundo, ¿ 9 > 7 ?: sí, se intercambian:

aux = 9
lista(1) = lista(2)
lista(2) = aux.

Los elementos quedan de la forma:

7 9 6 5 3 4 11 8

Comparamos el segundo con el tercero: si el segundo es mayor, se intercambian:

7 6 9 5 3 4 11 8

Análogamente haremos con los demás elementos. La lista quedará de la forma:

7 6 5 3 4 9 8 11

El mayor elemento de todos está en la última posición.

VARIABLES

i = índice para recorrer la lista.
aux = auxiliar para realizar el intercambio de los elementos.

PSEUDOCODIGO

```
i = 1
hacer mientras i <= N-1
     if lista(i) > lista(i+1)
          aux = lista(i)
          lista(i) = lista(i+1)
          lista(i+1) = aux
     fin del if
     i = i + 1
fin del hacer.
```

COMPROBACION

El ciclo se debe estar repitiendo hasta *N–1*, porque comparamos el elemento de la posición en que estamos con el siguiente.

Se comparan los dos primeros elementos. Si el primero es mayor, se intercambian con la ayuda del auxiliar, *aux=lista(i)=9 lista(i)=lista(i+1)=7 y lista(i+1)=9.*

Sean como sean los elementos, incrementamos el índice.

Cuando *i=N–1*, comparamos el elemento que está en dicha posición con el último por la condición: *lista(i) > lista(i+1).*

Este pseudocódigo es la base para algunos métodos de ordenación.

Ejercicio 3

Hallar el mínimo y la posición que ocupa en una lista de N elementos, L(N).

ESTUDIO PREVIO

Consideramos que el primer elemento de la lista es el mínimo.

Se compara el mínimo con los demás elementos. Si algún elemento es más pequeño, éste pasa a ser el mínimo.

El nuevo mínimo se sigue comparando con los elementos posteriores hasta llegar al último de la lista.

Sólo hará falta guardar la posición, pues con ella accedemos al elemento.

VARIABLES

i = índice para recorrer la lista desde la posición 2 hasta la N.
posmin = guarda la posición del mínimo, en principio es la primera.

PSEUDOCODIGO

```
Borrar pantalla
i=2
posmin=1
hacer mientras i <= N
     if L(i) < L(posmin)
          posmin = i
     fin del if
     i = i+1
fin del hacer
en 10,25 imprime "La posición del mínimo es: ", posmin
en 12,25 imprime " Su valor es: ", L(posmin)
fin del programa.
```

COMPROBACION

El bucle lo debe estar haciendo desde la posición 2 hasta la *N*, posición del último elemento.

Compara el elemento de la posición *i* con el de la posición del mínimo, *posmin*. Si es más pequeño, guarda en *posmin* dicha posición.

Al terminar de comparar todos los elementos de la lista imprime la posición del mínimo, *posmin*, y su valor, *L(posmin)*.

9.2. EJERCICIOS CON MAS DE UN ARRAY UNIDIMENSIONAL

Nuestros problemas no tienen por qué limitarse a la utilización de una sola lista unidimensional.

Habrá ocasiones en que sea necesario trabajar al mismo tiempo con más de una lista, porque:

- Los datos no son del mismo tipo; en un array todos deben ser homogéneos: numéricos o alfanuméricos, pero nunca una mezcla de ambos.
- Dividir un array en otros más pequeños.
- Mezclar varios arrays en uno solo.

Ejercicio 4

Dada la lista numérica L(N), generar a partir de ella otra lista, NUEVA(N), que contenga los elementos de la lista anterior multiplicados por un número determinado. Imprimir ambas listas.

ESTUDIO PREVIO

Fuera del ciclo dimensionamos la nueva lista e introducimos el número por el cual multiplicamos los elementos.

Suponemos que la dimensión de la lista L está guardada en N, por ello, no hace falta introducir por teclado la dimensión de NUEVA. Ambos vectores deben poseer igual número de elementos.

VARIABLES

NUEVA = nombre de la lista a generar.
i = índice para recorrer la lista L y la lista NUEVA.
num = almacena el número por el cual vamos a multiplicar.
fi = contador de las filas.

PSEUDOCODIGO

```
Borrar pantalla
dimensiona NUEVA(N)
en 10,35 imprime "Número:"
en 10,45 introduce num
fi = 23
repetir desde i = 1 hasta N
     if fi = 23
          borrar pantalla
          en 1,25 imprime "Lista Uno"
          en 1,45 imprime "Lista Dos"
          fi = 3
     fin del if
     NUEVA(i) = L(i)*num
     en fi,27 imprimir L(i)
```

```
    en fi,47 imprime NUEVA(i)
    fi = fi + 1
    if fi = 20
         en 23,20 imprime "Pulse ENTER para continuar"
         detener listado
         fi = 23
    fin del if
    i = i + 1
fin del repetir
fin del programa.
```

COMPROBACION

Dimensionamos la nueva lista a N elementos, los mismos que posee la otra lista.

La lista *L* se recorrerá desde la posición uno hasta la última: *repetir desde i = 1 hasta N*. Primero realiza la cabecera por estar *fi* inicializado a 23.

A cada elemento de L lo multiplicamos por el número introducido, *NUEVA(i) = L(i)* num*, y lo guardamos en la nueva lista.

El mismo ciclo lo utilizamos para imprimir ambas listas.

Ejercicio 5

Dada la lista numérica L(N), generar a partir de ella otra lista que contenga los elementos que están en las posiciones pares de L(N).

ESTUDIO PREVIO

El número de elementos del vector a generar es la parte entera de la mitad de los elementos de L.

VARIABLES

NUEVA = nombre de la lista a generar.
i = índice para recorrer la lista L.
j = índice para recorrer la lista NUEVA.

PSEUDOCODIGO

```
Borrar pantalla
dimensiona  NUEVA(int(N/2))
i = 2
j = 1
```

```
hacer mientras i <= N
     NUEVA(j) = L(i)
     j = j + 1
     i = i + 2
fin del hacer
fin del programa.
```

COMPROBACION

Dimensionamos la nueva lista a la parte del número de elementos de L, *dimensiona NUEVA(int(N/2))*.

La lista *L* se recorrerá desde la posición dos hasta la última, *hacer mientras i<=N*. El índice se incrementa de dos en dos; sólo nos interesa pasar por las posiciones pares, *i = i + 2*.

Dichos elementos se almacenan en la lista *NUEVA* mediante la asignación *NUEVA(j) = L(i)*. Al estar generándola, su índice aumenta de uno en uno, *j=j+1*.

Hacer de nuevo el ejercicio utilizando sólo *un* índice, por ejemplo la *i*, para recorrer ambas listas.

Ejercicio 6

Dada la lista numérica L(N), generar a partir de ella otra lista que contenga los elementos que están en las posiciones impares de L(N).

Imprimir ambas listas con el formato:

Lista Uno	Lista Dos
--- --- ---	--- ---

La cabecera deberá aparecer en todas las pantallas. Cambia de pantalla en la fila 20.

ESTUDIO PREVIO

Para crear la nueva lista es necesario saber si el número de elementos de L es par o impar. Si es par, el número de elementos es la mitad; en el caso de ser impar, será la mitad más uno.

Al tener que imprimir todos los elementos de la lista L, necesariamente hay que pasar por todas sus posiciones. Los elementos que estén en las posiciones impares se guardan en la nueva lista y además se imprimen.

Las filas serán variables para controlar las cabeceras. Son necesarias dos variables, una para cada lista.

VARIABLES

NUEVA = nombre de la lista a generar.
A = número de elementos de NUEVA.
i = índice para recorrer la lista L.
j = índice para recorrer la lista NUEVA.
fi = contador de las filas para imprimir la lista L.
fi2 = contador de las filas para imprimir la lista NUEVA.

PSEUDOCODIGO

```
A = N
if A = int(A/2)*2
     dimensiona NUEVA(A/2)*2
else
     dimensiona NUEVA((A/2)+1)
fin del if
i = 1
j = 1
fi = 23
fi2 = 0
hacer mientras i <= N
     if fi = 23
          hacer CABECERA
          fi = 3
          fi2 = 3
     fin del if
     en fi,15 imprime L(i)
     fi = fi + 1
     if i <> int(i/2)*2
          NUEVA(j) = L(i)
          en fi2,30 imprime NUEVA(j)
          j = j + 1
          fi2 = fi2 + 1
     fin del if
     if fi = 20
          en 23,20 imprime "Pulse ENTER para continuar"
          detener-listado
          fi = 23
     fin del if
     i = i + 1
fin del hacer
fin del programa.

CABECERA.
borrar pantalla
en 1,10 imprime "Lista Uno"
en 1,27 imprime "Lista Dos".
```

COMPROBACION

Comprobamos si el número de elementos de *L* es par o impar, para dimensionar la nueva lista.

La lista *L* se recorrerá desde la posición uno hasta la última, *hacer mientras i<=N.*

Comprueba si tiene que ejecutar el subprograma CABECERA. Lo realizará cuando el contador de las filas de la lista de la cual tiene que imprimir un mayor número de elementos sea 23, *fi;* además se inicializan ambos contadores de las filas a 3 para que empiece a imprimir los elementos de ambos vectores a partir de esa fila.

Imprime todos los elementos de la lista *L* en su fila correspondiente, *en fi,15 imprime L(i).*

Si la posición es impar, guarda el elemento en la nueva lista, *NUEVA(j)=L(i),* lo imprime en su fila correspondiente e incrementa su valor.

Si el contador de las filas es 20, *if fi=20,* detenemos la ejecución del listado para poder visualizar todos los elementos de esa pantalla. Para realizar la cabecera asignamos *fi=23.*

Aumenta la posición de la lista *L.*

El salto de página se controla con la variable *fi,* que corresponde al array *L,* por imprimir todos sus elementos, mientras que la lista NUEVA sólo contiene la mitad de los elementos de *L.*

9.3. LISTAS UNIDIMENSIONALES PARALELAS

Las listas unidimensionales paralelas son aquellas en las que el contenido de la posición de una lista determinada se corresponde con el contenido en la misma posición de otras listas.

Ejemplo: en la lista NOMBRES podemos almacenar los nombres de los alumnos y en otra paralela, NOTAS, las notas. Al alumno que ocupa la primera posición en NOMBRES le corresponde la nota almacenada en la primera posición en NOTAS. Al segundo alumno le corresponde la nota almacenada en la segunda posición. En general, al alumno i le corresponde la nota almacenada en la posición i.

Ejercicio 7

Un restaurante guarda en la lista PLATOS(20) los platos de que dispone y en otra lista paralela PRECIO(20) el valor de cada plato. Sacar la factura de un cliente de la siguiente forma:

Restaurante: EL BUEN YANTAR		
Concepto	Cantidad	Importe
--- ---	--- ---	--- ---
	Total:	----

ESTUDIO PREVIO

Las listas están creadas. Un cliente pedirá el número del plato y las raciones que desee. Con dicho número accedemos a la lista de PRECIO para saber el valor del plato, que lo multiplicamos por el número de raciones y nos dará el importe de ese plato. El importe total será la suma de todos los platos consumidos.

Para pedir los platos y las raciones de cada uno de ellos se realiza la operación en unas filas determinadas, por ejemplo a partir de la 18, según el formato:

Número de plato: –
Número de raciones –

Una vez pedido el plato se imprimen sus datos en la factura y se pregunta si se desea más platos, también en una fila determinada. Si no se quieren más platos se imprime el importe total de la factura.

Las operaciones de pedir la consumición como la de imprimir la factura deben ir en la misma pantalla.

VARIABLES

numero = número del plato de la carta.
racion = número de raciones del plato pedido en *numero*.
total = acumula el importe total de la factura.
res = para pedir más platos.
fi = contador de las filas.

PSEUDOCODIGO

```
res = "S"
borrar pantalla
en 5,70 imprime "Restaurante: EL BUEN YANTAR"
en 7,35 imprime "Concepto          Cantidad        Importe"
fi = 9
total = 0
hacer mientras res = "S"   OR   res = "s"
     numero = 0
     racion = 0
     en 18,20 imprime "Número de plato:"
     en 18,38 introduce numero
     en 20,20 imprime "Número de raciones:"
     en 20,42 introduce racion
     en 18,20 imprime espacios(30)
     en 20,20 imprime espacios(30)
     en fi,38 imprime PLATOS(numero)
     en fi,50 imprime racion
     en fi.65 imprime PRECIO(numero)*racion
     fi = fi + 1
```

```
     total = total + PRECIO(numero)*racion
     hacer MAS
fin del hacer
en 22,20 imprime espacios(30)
en fi+2,60 imprime "Total:",total
fin del programa.

MAS.
en 22,20 imprime "Desea más platos (S/N)"
en 22,44 introduce res.
```

COMPROBACION

Fuera del ciclo, para no tener que andar controlando las cabeceras, mediante las filas, imprimimos las cabeceras e inicializamos las filas y el total.

Dentro del ciclo se pide el número de plato y las raciones; estas variables están dentro del bucle para que cada vez que se ejecuten aparezcan blanqueadas.

Con el número del plato accedemos a la posición donde aparece la denominación del plato, *PLATOS(numero)*, y calculamos el importe de manera análoga mediante *PRECIO(numero)*racion*. Incrementamos el número de las filas y el importe se acumula en total, *total=total+ PRECIO(numero)*racion*.

Cuando el cliente no desee más platos, fuera del bucle se imprimirá el importe total de los platos consumidos.

Después de elegir la comida deseada se borran las filas donde se piden el número de plato y su ración, *imprime espacios(30)*, para que no aparezcan dichas filas si no se desean más raciones. De igual forma sucede, ya fuera del ciclo, con el comentario si se desea más platos y se imprima el total de la factura.

También se puede hacer introduciendo los números de platos deseados y las raciones correspondientes almacenando sus valores en listas. Por último se imprime la factura.

RESUMEN

Mediante una variable se pueden dimensionar las listas unidimensionales:

```
imprime "Número de elementos del array:"
introduce N
dimensiona L(N)
```

Cuando los datos no son homogéneos es necesario trabajar al mismo tiempo con más de un array.

Cuando la información de dos o más arrays se corresponde elemento a elemento; se dice que son **paralelos.**

EJERCICIOS PROPUESTOS

1. Generar e imprimir una lista de N elementos con números aleatorios entre 1 y 100. Sumar los que ocupan las posiciones pares.

   Los elementos situados en las posiciones impares que sean múltiplos de 4 se elevan al cuadrado. Imprimir su suma, así como el número de elementos que cumplen dicha condición.

2. Generar una lista de N elementos e imprimirla. Calcular el máximo, así como su posición.

3. Crear una lista de N elementos con nombres de personas. Imprimir el nombre de mayor longitud; en caso de haber más de un nombre, imprimir el primero introducido.

4. Dadas dos listas numéricas de 100 elementos, hallar la suma de los elementos que ocupan las posiciones pares de la primera con los elementos impares de la segunda.

5. Introducir en una lista la tabla de multiplicar del 7 y en otra la del 4. Realizar el producto de ambas listas elemento a elemento.

6. Supongamos que tenemos en la memoria del ordenador una lista numérica de N elementos. Los elementos que ocupan las posiciones pares los multiplicamos por ocho y a los que ocupan las posiciones impares les restamos el valor de su posición. Los nuevos valores los guardamos en otra lista.

7. Una empresa almacena en una lista de N elementos los nombres de sus empleados. En otra lista paralela guarda el número de horas trabajadas por cada empleado. Si la hora se paga a una cantidad determinada, imprimir la cantidad a cobrar por cada trabajador.

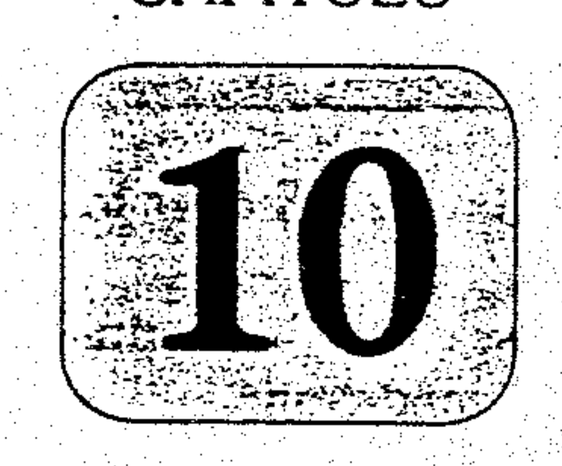
8. Un colegio almacena en una lista de 20 elementos los nombres de los estudiantes de un curso determinado. En otra lista, ASIGNATURAS, se almacenan los nombres de las cuatro asignaturas que se imparten en el curso en cuestión y en otras cuatro listas numéricas paralelas, NOTA1, NOTA2, NOTA3 y NOTA4, las notas correspondientes a las asignaturas. Imprimir:

Nombre	Asig1	Asig2	Asig3	Asig4	Media
----------	--	--	--	--	--
----------	--	--	--	--	--
----------	--	--	--	--	--
				Nota media global:	--

   La media es el resultado de sumar las notas de cada alumno y dividir la suma por el número de asignaturas.

9. Igual que el ejercicio anterior, pero ocupando cada alumno una pantalla.

CAPITULO



Ordenación de las listas. Búsqueda de un elemento

10.1. INTRODUCCION

A la hora de visualizar los elementos de una lista, en la mayoría de las ocasiones no nos interesa ver los datos tal como se introdujeron, sino de una forma ordenada, para lo cual es necesario ordenar la lista.

La ordenación puede ser:

- **Ascendente:** los elementos están situados de la *A-Z* o *a-z* o del *0-9* (el ordenador distingue entre las mayúsculas y las minúsculas).
- **Descendente:** inversa a la anterior.

Los valores repetidos, en caso de existir, quedan en posiciones contiguas (uno al lado del otro).

Para llevar a cabo la ordenación hay muchos métodos, y sirven tanto para listas numéricas como para las alfanuméricas.

10.2. METODO DE LA BURBUJA

Ejercicio 1

ESTUDIO PREVIO

Se basa en llevar el máximo de la lista a la última posición.

Sea la lista de 8 elementos:

9 7 6 5 3 4 11 8

En una *primera etapa* llevamos el máximo al final, aplicando el método visto anteriormente. Los elementos quedan de la forma siguiente:

7 6 5 3 4 9 8 11

Resumen: etapa=1
comparaciones necesarias 7 = número de elementos menos el valor de la etapa.

En una *segunda etapa* se repite el proceso de llevar el segundo máximo a la penúltima posición. No hará falta comparar con el último elemento, porque en la anterior etapa ya hemos llevado el máximo a la última posición:

6 5 3 4 7 8 9 11

Resumen: etapa=2
comparaciones 6 = número de elementos menos la etapa.

De igual forma procederemos con las demás etapas:

Tercera:	6	5	3	4	7	8	9	11
Cuarta:	6	5	3	4	7	8	9	11
Quinta:	5	3	4	6	7	8	9	11
Sexta:	3	4	5	6	7	8	9	11
Séptima:	3	4	5	6	7	8	9	11

En general:

- Número de etapas a realizar = número de elementos − 1.
- Comparaciones en cada etapa = número de elementos − número de la etapa.

VARIABLES

N = número de elementos de la lista.
i = contador de etapas y al mismo tiempo índice para recorrer la lista, desde 1 hasta N−1.
j = contador de comparaciones en cada etapa, desde 1 hasta N−i.
aux = auxiliar para realizar el intercambio de los elementos.

PSEUDOCODIGO

```
i = 1
hacer mientras i <  N
     j = i
     hacer mientras j <= N-i
          if lista(j) > lista(j+1)
               aux = lista(j)
               lista(j) = lista(j+1)
               lista(j+1) = aux
          fin del if
          j = j + 1
     fin del hacer.
     i = i + 1
fin del hacer.
```

COMPROBACION

El primer ciclo, *hacer mientras* $i<N$, es el contador de las etapas.

Antes de entrar en el segundo ciclo, *hacer mientras* $j<=N-i$, hay que reinicializar j a la unidad, $j=1$, para realizar las comparaciones en cada etapa desde la posición 1 hasta la $N-i$. Este bucle está llevando el máximo a la última posición.

10.3. METODO DEL SWITCH

Ejercicio 2

ESTUDIO PREVIO

Es una variante del método de la Burbuja. También está basado en llevar el máximo al final.

El *sw* nos permite conocer si la lista está o no ordenada.

Se recorre el *array*, los elementos se comparan dos a dos, y puede suceder que:

- Se produzca algún intercambio de elementos: la lista no estaba ordenada y puede que aún no lo esté. Al no estar ordenados se cambia el estado del *sw*.

  Cambiamos el estado del interruptor y volvemos a recorrer la lista, desde la primera posición hasta la última, comparando los elementos dos a dos.

  Este proceso debe repetirse mientras se produzca algún intercambio de elementos.
- No se produzca ningún cambio: la lista está ordenada.

VARIABLES

N = número de elementos de la lista.
i = índice para recorrer la lista, desde 1 hasta N−1.
aux = auxiliar para realizar el intercambio de los elementos.
sw = para saber si la lista ya está ordenada o aún no.

PSEUDOCODIGO

```
sw = 1
hacer mientras sw <> 0
     i = 1
     sw = 0
     hacer mientras i <= N-1
          if lista(j) > lista(j+1)
               aux = lista(i)
               lista(i) = lista(i+1)
               lista(i+1) = aux
               sw = 1
          fin del if
          i = i + 1
     fin del hacer.
fin del hacer.
```

COMPROBACION

El primer ciclo, *hacer mientras sw<>0*, permite saber si la lista está o no ordenada. Para que entre en él debe estar inicializado a 1, *sw=1*.

Una vez dentro del ciclo, se asigna al sw el valor 0, *sw=0*, para poder salir del bucle.

Se comparan los elementos, *hacer mientras i <= N-1*, dos a dos:

- Si no están ordenadas, se intercambian entre sí y se modifica el valor del sw, *sw=1*. La lista, al no estar ordenada, debe hacer una comprobación más; para ello volvemos a recorrer la lista desde la primera posición hasta la última. Previamente inicializamos el sw a cero.
- Si están ordenados, se dejan como están; el sw sigue a 0 y saldremos del bucle, *hacer mientras sw<>0*.

10.4. METODO DE LA BURBUJA-SWITCH

Ejercicio 3

ESTUDIO PREVIO

Este método es una mezcla de la Burbuja y del Switch. Mejora los defectos de ambos métodos.

Del de la Burbuja mejora el no tener que seguir haciendo comparaciones si en una etapa intermedia queda ordenada.

Del Switch, no tener que comparar siempre hasta el último elemento.

VARIABLES

N = número de elementos de la lista.
i = contador de etapas, en cada una de ellas desde 1 hasta N-1.
j = contador de comparaciones, desde 1 hasta N-i.
aux = auxiliar para realizar el intercambio de los elementos.
sw = permite conocer si la lista ya está ordenada.

PSEUDOCODIGO

```
i = 1
sw = 1
hacer mientras i <= N-1  AND  sw <> 0
     sw = 0
     j = 1
     hacer mientras j <= N-i
          if lista(j) > lista(j+1)
```

```
                aux = lista(j)
                lista(j) = lista(j+1)
                lista(j+1) = aux
                sw = 1
            fin del if
            j = j + 1
        fin del hacer
        i = i + 1
fin del hacer.
```

COMPROBACION

Mientras no se efectúen todas las etapas posibles y el sw sea distinto de cero la lista no está ordenada.

Dentro del ciclo se asigna:

- *sw=0:* para poder salir del bucle si la lista queda ordenada en alguna etapa intermedia o de por sí está ordenada.
- *j=1:* para ejecutar las comparaciones desde la posición 1.

El segundo ciclo lleva a cabo las comparaciones en cada etapa desde la primera posición hasta la $N-i$, ya que en cada una de ellas se lleva el máximo al final.

Si no está ordenada, se reasigna el *sw=0* y *j=1* para que siga haciendo las comparaciones desde la posición 1.

10.5. ORDENACION POR EL MINIMO

Ejercicio 4

ESTUDIO PREVIO

En una primera etapa se considera que la posición del mínimo es la primera. Debemos recorrer la lista, desde la segunda, para encontrar la posición del mínimo, en el caso de que no sea el elemento situado en la primera posición, e intercambiarlos de tal forma que el mínimo ocupe la primera posición.

En la segunda etapa consideramos que el mínimo ocupa la segunda posición. Recorremos la lista desde la posición tercera. Buscamos el mínimo y lo intercambiamos con el elemento de la segunda posición, en caso de que no ocupe dicho lugar, de tal forma que el segundo elemento más pequeño de toda la lista esté situado en la segunda posición.

Así sucesivamente hasta N−1 etapas, pues si los N−1 primeros elementos están ordenados, el último, el N, también lo estará.

Las etapas van desde 1 hasta N−1.

Dentro de cada etapa las comparaciones se realizan desde el elemento que ocupa la posición etapa+1 hasta el último, N.

VARIABLES

L = nombre de la lista.
N = número de elementos de la lista L.
aux = para realizar el intercambio de los elementos.
i = contador de etapas desde 1 hasta N−1.
j = recorrer la lista desde la posición etapa+1, *i+1*, hasta N para encontrar el mínimo.
posmin = guarda la posición del mínimo, coincide con el número de la etapa.

PSEUDOCODIGO

```
i=1
hacer mientras i <= N-1
     posmin = i
     j = i+1
     hacer mientras j <= N
          if L(j) < L(posmin)
               posmin = j
          fin del if
          j = j+1
     fin del hacer
     aux = L(posmin)
     L(posmin) = L(i)
     L(i) = aux
     i = i+1
fin del hacer.
```

COMPROBACION

Fuera del ciclo se asigna al contador de etapas el valor 1, por considerar que el mínimo ocupa la primera posición.

Dentro del bucle las comparaciones deben empezar desde la posición siguiente, para lo cual hacemos *posmin=i* y *j=i+1*. Comparamos y vamos guardando la posición del mínimo en *posmin*. Al terminar las comparaciones de cada etapa será cuando se realice el intercambio de los elementos.

Se aumenta el contador de las etapas. El mínimo pasa a ser el elemento situado en la posición i-ésima, *posmin=i*. Las comparaciones empiezan en el siguiente elemento, *j=i+1*.

Al finalizar todas las etapas la lista queda ordenada.

10.6. ORDENACION POR INSERCION

Ejercicio 5

ESTUDIO PREVIO

Considera que el primer elemento está ordenado.

Ordena el segundo respecto al primero, es decir, lo coloca delante o lo deja donde está.

Luego, coloca el tercer elemento en su lugar correspondiente respecto a los dos anteriores, es decir, lo sitúa en la primera posición o en la segunda o lo deja donde está.

A continuación, sitúa el cuarto elemento en su posición respecto a los tres anteriores.

Así sucesivamente hasta el último elemento, que lo debe colocar en su lugar respecto a todos los demás.

Se necesitan dos variables: una para recorrer la lista desde la posición 2 hasta la última y otra para ir desde la posición actual hacia las posiciones inferiores con el fin de encontrar el lugar adecuado al elemento.

Cuando se encuentre la posición no debe continuar comparando con los elementos que ocupan las posiciones anteriores, por encontrarse ya ordenados.

Ejemplo: para el array de cuatro elementos

4 3 1 2

la ordenación de los elementos para cada etapa es la siguiente:

Etapa 1:	3	4	1	2
Etapa 2:	1	3	4	2
Etapa 3:	1	2	3	4

VARIABLES

L = nombre del array.
N = número de elementos del array.
i = índice para recorrer la lista desde la posición 2 hasta la N.
j = índice para recorrer la lista desde la posición *i* hasta encontrar la posición del elemento o, en el peor de los casos, hasta la 2.
aux = auxiliar para el intercambio de los elementos.

PSEUDOCODIGO

```
i = 2
hacer mientras i<= N
     j = i
     hacer mientras L(j)<(j-1) AND j>=2
          aux = L(j)
          L(j) = L(j-1)
```

```
            L(j-1) = aux
            j = j-1
        fin del hacer
        i = i + 1
fin del hacer.
```

COMPROBACION

Se recorre la lista desde la posición 2 (se ha supuesto que el primer elemento está ordenado) hasta la N, *hacer mientras i<=N.*

Al elemento que está en la posición *i* hay que encontrarle el lugar que debe ocupar respecto a los elementos anteriores y mientras no lleguemos a la posición 2, *hacer mientras L(j)<L(j-1) AND j>=2,* puede suceder que sea:

- Más pequeño que el anterior: se intercambian entre sí y se sigue comparando, para lo cual disminuimos en una unidad la variable *j, j=j-1.* En el caso más extremo, hasta la posición 2.
- Mayor: está ordenado, sale del ciclo.

Una vez encontrada la posición de un determinado elemento, se pasará a buscar el lugar correspondiente del siguiente, i=i+1.

10.7. METODO SHELL

Ejercicio 6

ESTUDIO PREVIO

Está basado en la ordenación por inserción.

Se ordenan los elementos que están separados por una distancia de comparación que en un principio en la mitad de la longitud del array.

En cada etapa esta distancia se va reduciendo a la mitad hasta que sea menor que la unidad, en cuyo caso la lista ya estará ordenada.

Si el número de elementos es impar, calculamos la parte entera de su mitad.

La comparación de los elementos que se encuentran a una distancia determinada debe estar realizándose mientras haya cambios, es decir, no estén ordenados los elementos respecto a ese salto. Para saber si están o no ordenados con respecto a dicha separación, necesitamos un interruptor.

Ejemplo: consideremos una lista con 9 elementos. La parte entera de su mitad es 4. En la primera ordenación compara los elementos: el 1 con el 5, el 2 con el 6, etc.

Cuando estén ordenados los elementos respecto a esta distancia, se ordenan respecto a la mitad de la distancia, comparando el primero con el tercero, el segundo con el cuarto, etc., y luego respecto a la distancia 1.

Una vez que la distancia sea menor que la unidad, significa que el array está ordenado.

Ejemplo: sean los elementos del array

d f a b g a e b c

la situación de los elementos para cada distancia es la siguiente:

salto 4:	d	a	a	b	c	f	e	b	g
	c	a	a	b	d	f	e	b	g
salto 2:	a	a	c	b	d	b	e	f	g
salto 1:	a	a	b	b	c	d	e	f	g

VARIABLES

L = nombre del array.
N = número de elementos del array.
i = índice para recorrer la lista desde la posición 1.
aux = auxiliar para el intercambio de los elementos.
salto = comparar dos elementos separados entre sí esa distancia.
sw = para saber si con *salto* se ha producido un intercambio de elementos.

PSEUDOCODIGO

```
salto = int(N/2)
hacer mientras salto >= 1
     sw = 1
     hacer mientras sw <> 0
          sw = 0
          i = 1
          hacer mientras i <= (N-salto)
               if L(i) > L(i+salto)
                    aux = L(i+salto)
                    L(i+salto) = L(i)
                    L(i) = aux
                    sw = 1
               fin del if
               i = i + 1
          fin del hacer
     fin del hacer
     salto = int(salto/2)
fin del hacer.
```

COMPROBACION

En un principio el salto es la mitad de los elementos. Si es:

- Cero: quiere decir que la lista sólo posee un elemento y, por tanto, está ordenada.
- Mayor de cero: debe ordenarla, *hacer mientras salto >= 1*. Con este salto debe estar realizando las comparaciones hasta que no haya ningún intercambio de elementos, *hacer mientras sw <> 0;* puede suceder que:

— Se produzca algún intercambio: se asigna *sw=1*. Debe volver a comparar los elementos con este cambio.
— No se produzca ningún intercambio: para este salto la lista está ordenada. Sale de este ciclo y calcula la mitad del *salto*.

Se calcula dos veces el salto, la primera fuera del ciclo, por si sólo hay un elemento en la lista, y la otra dentro del ciclo y al final de todo, por ser la condición de salida.

10.8. ORDENACION POR MEZCLAS

Ejercicio 7

Se dispone de dos listas ordenadas A(N) y B(M). Se desea construir una nueva lista C, de tal forma que quede ordenada, cuyos elementos sean los de las listas A y B. Si hay algún elemento repetido en A y B, en C sólo aparecerá una vez.

ESTUDIO PREVIO

La dimensión de la nueva lista, al no saber si hay o no elementos repetidos (en caso de que los haya tampoco sabemos cuántos), será la suma de las dimensiones de las otras dos listas.

Para mezclar las dos listas en una nueva de forma que ésta quede ordenada, hay que proceder de la siguiente manera: se comparan los elementos situados en la primera posición de ambas listas y el más pequeño se coloca en C.

Se compara el elemento situado en la posición dos del array, del cual se pasó el elemento a C, con el primero del otro array. El más pequeño se coloca en la segunda posición, en C, y así sucesivamente.

Solamente hay que acceder a la siguiente posición en el array cuyo elemento se ha pasado a C: el otro conserva la posición actual.

Cuando los elementos sean iguales se pasará uno de ellos a C y se leerá de ambas listas.

Llegará un momento en que se hayan pasado todos los elementos de un array a C, quedan solamente elementos en el otro array. Dichos elementos habrá que pasarlos directamente, tal como están, a la lista C.

El razonamiento es válido tanto para listas numéricas como para las alfanuméricas.

Ejemplo:

A:	8	10	12	14	16	18	20
B:	4	8	9	10	11	17	

la nueva lista, C, contendrá los elementos ordenados:

C:	4	8	9	10	11	12	14	16	17	18	20

VARIABLES

C = nombre de la nueva lista.
a = posición en la lista A.
b = posición en la lista B.
c = posición en la lista C.

PSEUDOCODIGO

```
dimensiona C(M+N)
a = 1
b = 1
c = 1
hacer mientras a <= N  AND  b <= M
     if A(a) = B(b)
          C(c) = A(a)
          a = a + 1
          b = b + 1
     else
          if A(a) < B(b)
               C(c) = A(a)
               a = a + 1
          else
               C(c) = B(b)
               b = b + 1
          fin del if
     fin del if
     c = c + 1
fin del hacer
if a > N
     hacer mientras b <= M
          C(c) = B(b)
          c = c + 1
          b = b + 1
     fin del hacer
else
     hacer mientras a <= N
          C(c) = A(a)
          c = c + 1
          a = a + 1
     fin del hacer
fin del if
fin del programa.
```

COMPROBACION

Dimensionamos la nueva lista e inicializamos los índices.

Ejecuta el bucle mientras no haya llegado al final de alguna de las listas, la *A* o la *B*, *hacer mientras a<=N AND b<=M*, y comienza a comparar dos a dos los elementos:

- Son iguales: el elemento de A lo graba en C e incrementamos los índices de las listas A y B.
- No son iguales: preguntamos cómo son, *if A(a)<B(b)*, entre sí:
  - — Si es más pequeño el elemento de A, lo grabamos en la lista *C*, *C(c)=A(a)*, e incrementamos el índice de A.
  - — Si es más pequeño el elemento de B, lo grabamos en C, *C(c)=B(b)*, e incrementamos el índice de B.

En todos estos casos siempre incrementa el valor del índice en la lista C; está fuera de todos los IF para que sólo se ejecute la instrucción una vez.

Cuando salga del bucle es debido a que todos los elementos de alguna de las listas se han pasado a C. Para saber cuál, se pregunta por el valor del índice de alguna de ellas, por ejemplo la A, *if a > N*:

- Sí: ha terminado la lista A, quedan por pasar elementos de la B. Se realiza el bucle *hacer mientras b <= M* y se incrementan los índices de B y de C.
- En caso contrario ha terminado la lista B; terminaremos de pasar los elementos de A, *hacer mientras a <= N*. Se aumentan los índices de A y de C.

10.9. BUSQUEDA DE UN ELEMENTO

La búsqueda consiste en saber si un determinado elemento está en la lista y, en caso de hallarse, indicar la posición que ocupa.

Hay diversos métodos para realizar la búsqueda de un elemento en un array:

— **Búsqueda lineal:** válido tanto para arrays ordenados como para desordenados.
— **Búsqueda binaria** o **dicotómica:** exclusivamente para arrays ordenados.

10.9.1. Búsqueda lineal en un array desordenado

Se recorre el array desde la primera posición hasta que encontremos el elemento. En este preciso instante abandonamos la búsqueda. Si recorremos toda la lista y no hallamos el elemento, indicaremos la no presencia del elemento.

Ejercicio 8

Comprobar si existe un determinado nombre en la lista DATOS(N) e indicar la posición que ocupa. Repetir el proceso.

ESTUDIO PREVIO

Mientras no nos digan que está ordenada debemos suponer que está desordenada.

El nombre a buscar se debe introducir por teclado, para lo cual necesitamos una variable.

Para salir del proceso repetitivo de lectura de los elementos, una vez que se haya encontrado, es necesario el uso de un *sw* y del operador lógico AND.

VARIABLES

nombre = dato a buscar.
i = variable para recorrer la lista.
res = para seguir buscando más elementos.
sw = salir del bucle.

PSEUDOCODIGO

```
res = "S"
Hacer = mientras res="S"
     borrar pantalla
     en 10,25 imprime "Nombre:"
     en 10,33 introduce nombre
     sw = 0
     i = 1
     hacer mientras i <= N  AND  sw = 0
          if datos(i) = nombre
               en 14,25 imprime "Está en la posición", i
               sw = 1
          else
               i = i + 1
          fin del if
     fin del hacer
     if sw = 0
          en 14,25 imprime "No está"
     fin del if
     res = espacios(1)
     hacer MAS
fin del hacer
fin del programa.

MAS.
Hacer mientras res<>"S" AND res<>"N"
     res = espacios(1)
     en 20,25 "Buscar más datos(S/N):"
     en 20,48 introduce res
     res = convertir_mayusculas(res)
fin del hacer.
```

COMPROBACION

El ciclo lo debe estar haciendo mientras no sea el final de la lista y no haya encontrado al elemento buscado, *hacer mientras i<=N AND sw = 0.*

Empezamos a recorrer la lista. Se comprueba si el elemento que está en esa posicion es el buscado, ¿ datos(i) = nombre ? :

- Sí: imprime la posición que ocupa, *i*. Para salir del bucle asignamos *sw=1*, con lo cual deja de cumplirse una de las condiciones del ciclo.
- No: pasamos al siguiente elemento.

Cuando abandonemos el ciclo puede ser por dos motivos:

- Está el elemento: *sw = 1*.
- No está: *sw = 0*.

Para saber si está el elemento preguntamos por el *sw*.
Realiza el subprograma *MAS* para saber si deseamos buscar más datos.

10.9.2. Búsqueda lineal en un array ordenado

Se recorre el array desde la primera posición hasta la última posición, siempre y cuando los elementos del array sean menores que el elemento a buscar.

Si es mayor el elemento del array que el dato a buscar, al estar ordenado, quiere decir que no estará en las posiciones posteriores con lo cual se abandonará la búsqueda inmediatamente.

Ejercicio 9

Comprobar si existe un determinado nombre en la lista ordenada DATOS(N) e indicar la posición que ocupa.

ESTUDIO PREVIO

El nombre a buscar se debe introducir por teclado.

Saldremos del proceso repetitivo de lectura de los elementos una vez que se haya encontrado o su valor sea superior al dato buscado.

VARIABLES

nombre = dato a buscar.
i = variable para recorrer la lista.
sw = salir del bucle.

PSEUDOCODIGO

```
Borrar pantalla
en 10,25 imprime "Nombre:"
en 10,33 introduce nombre
sw = 0
i = 1
```

```
Hacer mientras i <= N  AND  datos(i) <= nombre  AND  sw = 0
     if datos(i) = nombre
          en 14,25 imprime "Está en la posición", i
          sw = 1
     else
          i = i + 1
     fin del if
Fin del hacer
if sw = 0
     en 14,25 imprime "No está"
fin del if
fin del programa.
```

COMPROBACION

El ciclo se debe estar ejecutando mientras no sea el final de la lista, los elementos sean menor o igual que el dato y mientras no encuentre al elemento buscado, *hacer mientras i <= N AND datos(i) <=nombre AND sw = 0.*

Recorremos la lista desde la primera posición comprobando si el elemento que hay en esa posición es el buscado:

- Sí: imprime la posición que ocupa, *i*. Para salir asignamos *sw=1*, con lo cual deja de cumplirse una de las condiciones de salida del ciclo y lo abandona.
- No: pasamos al siguiente elemento. Si es mayor que el buscado, sale del bucle por dejar de cumplirse la condición *datos(i) <=nombre.*

Cuando abandone el ciclo, si el estado del *sw* es cero, el dato no se encuentra en la lista.

10.10. BUSQUEDA BINARIA O DICOTOMICA

Ejercicio 10

Dada la lista alfabética ordenada L(N) encontrar la posición que ocupa un determinado dato en la misma.

ESTUDIO PREVIO

Es válida exclusivamente para listas ordenadas.

Consiste en comparar el dato a buscar con el elemento que ocupa la posición mitad del array. En caso de no ser igual al elemento buscado se reduce el intervalo de búsqueda a la inferior o superior, dependiendo del lugar donde se pueda encontrar el elemento que

Para cambiar los extremos de búsqueda del intervalo se necesitan dos variables, una para el extremo superior y la otra para el inferior.

La búsqueda termina cuando se ha encontrado el elemento, para lo cual necesitamos un *sw*, o bien cuando el extremo superior sea menor que el extremo inferior.

Siempre que el elemento a buscar no se encuentre en la lista, el pseudocódigo llega a un estado en que ambos extremos, el inferior y el superior, y la posición mitad coinciden. En el paso siguiente, el extremo superior es menor que el inferior y el programa sale del bucle.

VARIABLES

L = nombre del array.
N = número de elementos del array.
nombre = dato a buscar.
m = variable para encontrar la posición mitad del intervalo de búsqueda.
sup = contiene la posición superior del intervalo donde puede estar el elemento.
inf = contiene la posición inferior del intervalo donde puede estar el elemento.
sw = salir del bucle.

PSEUDOCODIGO

```
Borrar pantalla
en 10,25 imprime "Dato a buscar:"
en 10,40 introduce nombre
inf = 1
sup = N
m = int((sup+inf)/2)
sw = 0
hacer mientras inf <= sup  AND  sw = 0
     if L(m) = nombre
          en 14,20 imprime "Está en la posición",m
          sw = 1
     else
          if L(m) > nombre
               sup = m - 1
          else
               inf = m + 1
          fin del if
     fin del if
     m = int((sup+inf)/2)
fin del hacer
if sw = 0
     if L(m) = nombre
          en 14,20 imprime "Está en la posición",m
     else
          en 14,20 imprime "No está"
     fin del if
fin del if
fin del programa.
```

COMPROBACION

Una vez introducido el valor del elemento a buscar, asignamos los extremos del intervalo. Al principio del programa coinciden con la primera posición de la lista y con la última, *inf=1, sup=N*. Calculamos la posición mitad del array.

El ciclo lo debe realizar mientras el extremo inferior sea menor o igual que el superior y no encuentre el elemento a buscar.

Compara el valor del elemento de la posición mitad con el dato a buscar:

— Son iguales: para salir del ciclo se asigna al sw el valor 1.
— No son iguales; en este caso, el elemento de la lista puede ser:
- Mayor: se cambia el valor del límite superior *sup=m−1* y se resta la unidad, porque se ha comparado ya con el elemento que está en la posición *sup* y no estaba allí.
- Menor: se cambia el valor del límite inferior *inf=m+1*.

Al abandonar el ciclo preguntamos por el valor del sw:

— Uno: encontró al elemento en la posición *m*.
— Cero: ha comprobado todas las posiciones menos la *m*. Compara dicho elemento con el dato, *if L(m)=nombre:*
- Son iguales: está en la posición *m*.
- No son iguales: no está el elemento en la lista.

RESUMEN

Los métodos utilizados para la ordenación de los arrays unidimensionales son:

- Burbuja.
- Switch.
- Burbuja-Switch.
- Ordenación por el mínimo.
- Ordenación por inserción.
- Shell.
- Ordenación por mezclas (para dos arrays ordenados).

Para buscar un elemento en un array unidimensional, hay que tener en cuenta, si el array está ordenado o desordenado.

Los métodos de búsqueda son:

- Array desordenado
  — búsqueda lineal o secuencial.
- Array ordenado
  — búsqueda lineal o secuencial.
  — búsqueda binaria o dicotómica.

EJERCICIOS PROPUESTOS

1. Crear una lista de 20 elementos con los 20 primeros números primos. Imprimirla.
2. Cargar aleatoriamente una lista de 1 000 elementos con las vocales y contar las veces que aparece la a.
3. Generar aleatoriamente una lista de 10 elementos con valores de 1 a 50. Crear su inversa, es decir, que el primero de una esté el décimo en la otra, el segundo el noveno, etc.

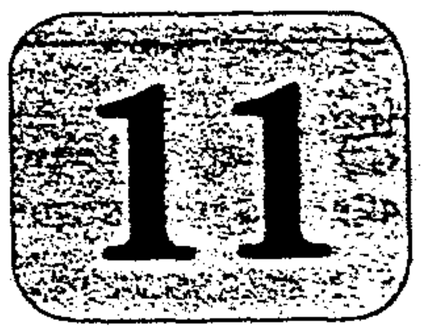
   Hallar la suma de los términos correspondientes de cada lista e imprimirlas según el formato:

Lista A	Inversa	Suma
--	--	---
--	--	---
--	--	---

4. Crear una lista con las notas de 100 alumnos. Hallar la media y la desviación de cada nota respecto a la media e imprimirla.

   Nota: la media se calcula sumando todas las notas y dividiendo la suma por el número de alumnos. La desviación de cada nota respecto a la media es el valor absoluto de la nota menos la media.
5. Crear una lista de N elementos con palabras introducidas por teclado. Imprimirla. Ordenarla por la longitud de cada palabra e imprimirla.
6. Crear una lista de N elementos con palabras introducidas por teclado. Imprimirla. Ordenarla alfabéticamente e imprimirla.
   ¿Cuántas palabras empiezan por C?
7. Introducir un número de N cifras. Comprobar si es capicúa.
8. Introducir una serie de nombres de personas con su dirección y población.
   a) Ordenarla por apellidos.
   b) Imprimirla.
   c) Buscar la dirección de una persona determinada.
9. Crear una lista de N elementos con ceros y unos, alternativamente.
10. Crear una lista de N elementos. Imprimirla. Desplazar los elementos de la lista una posición hacia la derecha de tal forma que el último pase a la primera posición, el primero a la segunda, el segundo a la tercera, y así sucesivamente.
11. Crear una lista de N elementos. Imprimirla. Desplazar los elementos de la lista una posición hacia la izquierda de tal forma que el primero pase a la última posición, el segundo a la primera, el tercero a la segunda, y así sucesivamente.

CAPITULO



Arrays bidimensionales

11.1. ARRAYS BIDIMENSIONALES

Se les denomina también *matrices* o *tablas*.

Un **array bidimensional** o un array m × n es un conjunto de datos homogéneos, cada uno de los cuales debe referenciarse por dos índices, I y J, llamados *subíndices*, que deben cumplir la propiedad de que:

$$1 \leqslant I \leqslant m \quad y \quad 1 \leqslant J \leqslant n$$

Para **dimensionarlos** se necesitan dos índices:

DIMENSIONA NOMBRE–TABLA(INDICE1, INDICE2)

donde:

- *NOMBRE–TABLA* es el nombre mediante el cual el ordenador va a reconocer a ese conjunto de datos homogéneos. Debe cumplir la misma normativa que las variables.
- *INDICE1* es el número máximo de filas de la matriz.
- *INDICE2* es el número máximo de columnas de la matriz.

Los **elementos del array bidimensional** vendrán referenciados de la siguiente forma:

NOMBRE–TABLA(I,J)

donde:

- *I*, el primer índice, es la variable que toma el valor de la fila en la cual está el elemento al que deseamos acceder. Tendrá como límite inferior *1* y como límite superior el valor de INDICE1.

  Una fila es una lista horizontal de elementos.

- *J*, el segundo índice, es la variable que toma el valor de la columna en la cual está el

elemento al que deseamos acceder. Tendrá como límite inferior *1* y como límite superior el valor de INDICE2.

Una columna es una lista vertical de elementos.

Los elementos de una fila tienen todos ellos igual valor para el primer subíndice, I, mientras que los de una columna tienen igual valor para el segundo, J.

Ambos subíndices siempre son números enteros.

El hecho de asignar el primer índice, I, a las filas y el segundo, J, a las columnas sólo es cuestión de nomenclatura. No hay ningún impedimento en asignar I a las columnas y J a las filas siempre y cuando seamos coherentes en todo el programa.

Ejemplo: la representación estándar para un array A de 4×5 sería una matriz de 4 filas y 5 columnas:

		Columnas				
		1	2	3	4	5
Filas	1	A(1,1)	A(1,2)	A(1,3)	A(1,4)	A(1,5)
	2	A(2,1)	A(2,2)	A(2,3)	A(2,4)	A(2,5)
	3	A(3,1)	A(3,2)	A(3,3)	A(3,4)	A(3,5)
	4	A(4,1)	A(4,2)	A(4,3)	A(4,4)	A(4,5)

El número de elementos de la matriz es $4 \times 5 = 20$.

Supongamos que tenemos el array bidimensional NOMBRES en la memoria del ordenador, con los nombres siguientes tal como indica la Figura 11.1.

Filas \ Columnas	1	2	3	4	5
1			Fernando		
2					Rosario
3		Amparo			
4		Pedro	Julio	Nieves	

Figura 11.1.

de tal forma que:

NOMBRES(1,3) = Fernando
NOMBRES(3,2) = Amparo
NOMBRES(4,2) = Pedro
NOMBRES(2,5) = Rosario
NOMBRES(4,3) = Julio
NOMBRES(4,4) = Nieves

Como los arrays bidimensionales están almacenados en la memoria del ordenador, sus elementos son de acceso directo.

Podemos ir directamente a una posición determinada simplemente indicando sus coordenadas.

El tipo de elementos que se almacenan en las tablas es el mismo en todas sus posiciones, bien numérico o bien alfanumérico, pero nunca una mezcla de ambas.

Cuando se desee guardar información de distinto tipo en listas, habrá que utilizar arrays unidimensionales paralelos.

Ejercicio 1

Generar una tabla A(4,5) con números aleatorios entre 1 y 100. Imprimir la tabla.

ESTUDIO PREVIO

A medida que vamos generando los números los vamos imprimiendo. Los elementos pertenecientes a una misma fila del array se visualizarán en pantalla en la misma línea pero en columnas diferentes.

Los de la fila segunda se imprimirán en otra línea diferente pero en las mismas columnas que los elementos de la fila anterior.

Análogamente para las demás filas del array bidimensional, para lo cual las filas y las columnas serán variables.

Al no estar creada la tabla es necesario dimensionarla.

VARIABLES

A = nombre de la matriz.
i = índice de las filas.
j = índice de las columnas.
fi = contador de las filas.
co = contador de las columnas.

PSEUDOCODIGO

```
Borrar pantalla
dimensiona A(4,5)
i = 1
fi = 10
co = 15
en 6,25 imprime "Elementos de la matriz"
hacer mientras i <= 4
     j = 1
     hacer mientras j <= 5
          A(i,j) = int(rnd*100)+1
          en fi,co imprime A(i,j)
          co = co + 5
          j = j + 1
     fin del hacer
     co = 15
     fi = fi + 2
     i = i + 1
fin del hacer
fin del programa.
```

COMPROBACION

Fuera del ciclo se dimensiona la tabla y se inicializan los índices y el contador de las filas.

El primer ciclo, *hacer mientras i<=4*, sirve para recorrer las filas.

Dentro de este ciclo inicializamos el índice de las columnas, *j=1*, porque al cambiar la fila hay que comenzar siempre desde la columna uno.

En cada fila, al mismo tiempo que se generan, se almacenan en la matriz, *A(i,j) = int(rnd*100)+1*, y se imprimen en pantalla. Esta operación se debe realizar mientras no se hayan generado los cinco elementos que hay en cada columna, *hacer mientras j<=5*.

Después de imprimir un elemento, el siguiente, *j = j+1*, se imprimirá en la misma fila, pero cinco columnas hacia la derecha, *co = co+5*.

Cuando se hayan generado e impreso todos los números de la misma fila, comenzamos a generar los elementos de la fila siguiente, *i = i+1*, imprimiéndolos en filas diferentes, *fi=fi+2*, pero en la misma columna, *co = 15*, que la fila anterior, y siempre desde la columna 1 de la matriz, *j = 1*.

Ejercicio 2

Generar una tabla A(5,5) introduciendo los valores por teclado. Imprimirla. Sumar los elementos de las columnas impares. Imprimir los elementos A(2,3) y A(3,2).

Formato de visualización de los elementos:

Elementos de la tabla
-- -- -- -- -- -- -- -- -- --
Suma de los elementos de las columnas impares: ---
Elemento A(2,3): -- Elemento A(3,2): --

ESTUDIO PREVIO

Primero introducimos los valores por teclado y al mismo tiempo sumamos los elementos que estén en las columnas impares. Posteriormente imprimimos los datos pedidos.

Para introducir los datos utilizamos el formato:

```
Fila: -
Columna: -
Número: --
```

donde fila y columna indican la posición que va a ocupar el elemento en la tabla. Esta información la debe dar el ordenador y nosotros sólo debemos introducir los números que deseemos.

VARIABLES

A = nombre de la matriz.
i = índice de las filas.
j = índice de las columnas.
fi = contador de las filas.
co = contador de las columnas.
sum = acumulador de los elementos que están en las columnas impares.

PSEUDOCODIGO

```
Borrar pantalla
dimensiona A(5,5)
fi = 0
co = 0
hacer INTRODUCIR-DATOS
hacer VISUALIZAR
hacer RESULTADOS
fin del programa.

INTRODUCIR-DATOS.
i = 1
hacer mientras i <= 4
     j = 1
     hacer mientras j <= 5
          en 10,20 imprime "Fila:",i
          en 12,20 imprime "Columna:",j
          en 14,20 imprime "Número:"
          en 14,28 introduce A(i,j)
          if j <> int(j/2)*2
               sum = sum + A(i,j)
          fin del if
          j = j + 1
     fin del hacer
     i = i + 1
fin del hacer.

VISUALIZAR.
i = 1
fi = 4
co = 15
```

```
hacer mientras i <= 4
     j = 1
     hacer mientras j <= 5
          en fi,co imprime A(i,j)
          co = co + 5
          j = j + 1
     fin del hacer
     co = 15
     fi = fi + 2
     i = i + 1
fin del hacer.

RESULTADOS. -
fi = fi + 2
en  fi,20  imprime  "La suma de los  elementos  de  las  columnas impares:",sum
en fi+2,20 imprime "Elemento A(2,3): ",A(2,3)
en fi+4,20 imprime "Elemento A(3,2): ",A(3,2).
```

COMPROBACION

Para una mejor comprensión se ha estructurado en subprogramas que realizan las funciones de introducir los datos, visualización de los elementos introducidos en el subprograma anterior e impresión de los resultados finales.

Antes de ellos dimensionamos la tabla e inicializamos los contadores de las filas.

En el primer subprograma, INTRODUCIR-DATOS. se guardan los elementos en la matriz. Para ello recorremos todas las posiciones a partir de la fila 1, *hacer mientras i <= 4*, hasta la 4. Dentro de cada fila pasamos por todas las columnas, *hacer mientras j<=5*.

El índice de las columnas siempre irá dentro del ciclo de las filas, para que cuando accedamos a la siguiente columna vuelva a empezar desde la columna uno.

El subprograma VISUALIZAR imprime los elementos de la tabla, pero antes hay que inicializar los índices de las filas y de las columnas a 1, *i=1, j=1*, y se asigna el número de la fila y de la columna, en las cuales se empezarán a imprimir los datos.

Por último, en RESULTADOS se imprimirá, en filas diferentes, el resultado de la suma y de los dos elementos pedidos.

Los elementos se imprimen posicionándonos directamente en las coordenadas específicas de cada uno de ellos, por ser las tablas de acceso directo.

Ahora no podemos imprimir los elementos del array al mismo tiempo que estamos introduciendo los números. En el ejercicio anterior sí se podía realizar, porque los números los generaba internamente el ordenador; ahora tenemos que introducirlos nosotros por teclado.

El cálculo de la suma también se puede efectuar en el subprograma VISUALIZAR.

Ejercicio 3

Generar una tabla A(5,5) de tal forma que sus filas pares sean múltiplos de 2 y las impares sean múltiplos de 3. Imprimirla.

Hallar la suma de todos sus elementos.

Imprimir los elementos de la diagonal principal.

Formato de visualización de los elementos:

Elementos de la tabla
-- -- -- -- -- -- -- -- -- --
Suma de los elementos: ---
Elementos de la diagonal principal -- -- -- -- -- --

ESTUDIO PREVIO

A medida que vamos generando los elementos de la tabla los iremos imprimiendo y sumando, porque la operación de generar los números la realiza internamente el ordenador.

Para crear los múltiplos de 2 y de 3 necesitamos dos variables, una para cada uno de ellos.

Los elementos de la diagonal principal poseen la particularidad de tener igual valor el índice de las filas que el de las columnas.

VARIABLES

A = nombre de la matriz.
i = índice de las filas.
j = índice de las columnas.
fi = contador de las filas.
co = contador de las columnas.
sum = acumulador de todos los elementos.
dos = contador para generar los múltiplos de 2.
tres = contador para generar los múltiplos de 3.

PSEUDOCODIGO

```
dimensiona A(5,5)
fi = 4
co = 15
i = 1
hacer GENERAR
hacer SUMAS
```

```
hacer DIAGONAL
fin del programa.

GENERAR.
dos = 0
tres = 0
sum = 0
borrar pantalla
en 2,25 imprime "Elementos de la tabla"
hacer mientras i <= 4
      j = 1
      hacer mientras j <= 5
            if i = int(i/2)*2
                  dos = dos + 2
                  A(i,j) = dos
            else
                  tres = tres + 3
                  A(i,j) = tres
            fin del if
            en fi,co imprime A(i,j)
            sum = sum + A(i,j)
            co = co + 5
            j = j + 1
      fin del hacer
      co = 15
      fi = fi + 2
      i = i + 1
fin del hacer.

SUMAS.
fi = fi + 2
en fi,25 imprime "Suma de los elementos: ", sum

DIAGONAL.
i = 1
fi = fi + 2
co = 25
en 2,25 imprime "Elementos de la diagonal principal"
hacer mientras i <= 4
      j = 1
      hacer mientras j <= 5
            if i = j
                  en fi,co imprime A(i,j)
                  co = co + 5
            fin del if
            j = j + 1
```

```
    fin del hacer
    i = i + 1
fin del hacer.
```

COMPROBACION

Antes de los subprogramas dimensionamos la matriz e inicializamos los contadores de las filas y de las columnas, así como el índice de las filas.

En el subprograma GENERAR, en primer lugar imprimimos la cabecera. Recorremos la matriz y controlamos la paridad o imparidad de las filas, *if i=int(i/2)*2:*

- Par: se genera un número múltiplo de 2, *dos=dos+2*, y se almacena en la tabla, *A(i,j)=dos.*
- Impar: se genera un múltiplo de 3, *tres=tres+3*, almacenándose en la matriz, *A(i,j)= tres.*

En ambos casos, imprime el elemento, lo almacena en *sum* y aumenta los contadores.

En SUMAS imprimimos el valor de la suma de los elementos.

Por último, en DIAGONAL, recorremos de nuevo toda la tabla, desde la fila 1 y columna 1, *i=1, j=1*, para buscar aquellos elementos que cumplen la condición cuya fila sea igual a su columna, *if i=j*, para imprimirlos.

Ejercicio 4

Generar una tabla A(4,5) cuyos valores sean aleatorios entre 1 y 100. A partir de ella crear su transpuesta, es decir, B(5,4).

Imprimir ambas tablas según el formato:

Tabla uno					Transpuesta			
--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--

Imprimir los elementos A(3,2) y B(3,3).

La matriz transpuesta se forma poniendo ordenadamente las filas de la matriz A como columnas en la matriz B.

ESTUDIO PREVIO

A medida que vamos generando los elementos de la tabla A, rellenamos la tabla B e imprimimos ambas tablas.

La visualización de los elementos de ambas tablas debe ir en posiciones distintas, para lo cual se necesitan contadores de las filas y de las columnas, independientes para cada matriz.

Los elementos de la matriz *A* se imprimen por filas, mientras que los de *B* se visualizan por columnas.

VARIABLES

A = nombre de la matriz.
B = nombre de la matriz transpuesta.
i = índice de las filas.
j = índice de las columnas.
fi = contador de las filas de la matriz A.
co = contador de las columnas de la matriz A.
fit = contador de las filas de la matriz transpuesta B.
cot = contador de las columnas de la matriz transpuesta B.

PSEUDOCODIGO

```
Borrar pantalla
dimensiona A(4,5)
dimensiona B(5,4)
fi = 8
co = 10
fit = 8
cot = 40
i = 1
en 6,15 imprime "Tabla uno"
en 6,45 imprime "Transpuesta"
hacer mientras i <= 4
     j = 1
     hacer mientras j <= 5
          A(i,j) = int(rnd*100)+1
          B(j,i) = A(i,j)
          en fi,co imprime A(i,j)
          en fit,cot imprime B(j,i)
          co = co + 4
          fit = fit + 2
          j = j + 1
     fin del hacer
     fi = fi + 2
     co = 10
     fit = 8
     cot = cot + 4
     i = i + 1
fin del hacer.
en fit+3,20 imprime "Elementos A(3,2):",A(3,2)
en fit+5,20 imprime "El
```

COMPROBACION

Dimensionamos las tablas e inicializamos las variables.

Para crear la tabla *A*, los índices recorrerán todas las posiciones de la misma. *hacer mientras i<=4 y hacer mientras j<=5.*

Generamos un número y se almacena en A y al mismo tiempo en la matriz transpuesta, *B(j,i)=A(i,j)*, y se imprimen.

Los elementos de A se imprimirán en la misma fila pero en columnas diferentes, *co=co+4*, mientras que los de su transpuesta irán en la misma columna pero en filas diferentes, *fit=fit+2*.

Al terminar de generar los elementos de la misma fila para la matriz *A*, generaremos los de la siguiente fila.

Para imprimirlos hay que reorganizar las filas y las columnas en las cuales se imprimirán los elementos de ambas matrices.

Los elementos de la matriz *A* se visualizarán en filas distintas, *fi=fi+2*, empezando en la misma columna, *co=10*.

Los de la transpuesta se imprimirán en la misma fila de la pantalla que los elementos de la fila anterior, *fit=8*, pero en distintas columnas, *cot=cot+4*.

Antes de terminar, se visualizan los elementos pedidos posicionándonos directamente en ellos.

Ejercicio 5

Una empresa tiene 100 obreros repartidos en 4 categorías profesionales.

La empresa dispone de una matriz de 100×31; cada posición contiene el número de horas extraordinarias trabajadas por cada obrero cada día.

También se dispone de dos listas lineales, una de 100 elementos con la categoría de cada trabajador y otra de 4 con el precio de las horas extraordinarias por categoría.

Calcular:

— Cuánto ha cobrado cada obrero en ese mes por horas extraordinarias.
— El total pagado por dichas horas.

ESTUDIO PREVIO

Recorreremos la matriz de las horas extraordinarias.

Las horas realizadas por cada obrero están en la misma fila; a medida que recorremos la tabla las iremos sumando.

Para calcular lo que cobrará ese obrero por las horas extraordinarias trabajadas durante el mes, multiplicamos las horas realizadas por el precio correspondiente a su categoría laboral.

Para ello, primero tenemos que buscar cuál es su categoría profesional en el array de las categorías.

Una vez encontrada su categoría, buscamos el precio que corresponde a dicha categoría en el array del precio de las horas extras.

VARIABLES

HORAS(100,31) = matriz con las horas extraordinarias.
CATEGORIA(100) = lista paralela a la tabla de las horas con la categoría de cada trabajador.
PRECIO(4) = lista con el precio de las horas extra por categoría.
h = acumulador de las horas para cada obrero.
total = acumulador del total pagado a todos los trabajadores.
i = índice para recorrer las filas de la tabla HORAS.
j = índice para recorrer las columnas de la tabla HORAS.

PSEUDOCODIGO

```
total = 0
i = 1
borrar pantalla
hacer mientras i <= 100
     h = 0
     j = 1
     hacer mientras j <= 31
          h = h + HORAS(i,j)
          j = j + 1
     fin del hacer
     imprime h*PRECIO(CATEGORIA(i))
     total = total + h*PRECIO(CATEGORIA(i))
     i = i + 1
fin del hacer
imprime total
fin del programa.
```

COMPROBACION

La variable *h* almacena las horas de cada obrero y nos va a servir para todos ellos: por tanto, antes de pasar al siguiente trabajador se inicializará a cero.

Recorremos todo el mes para cada trabajador, *hacer mientras j<=31*, acumulando las horas en *h*.

Una vez que el ordenador haya recorrido todos los días del mes, la variable *h* tenemos que multiplicarla por el precio de las horas extra de acuerdo con la categoría del obrero.

Al ser la lista CATEGORIA paralela a las HORAS, la categoría del trabajador *i* estará en CATEGORIA(i) y el precio en la posición que devuelva CATEGORIA(i); todo esto se puede expresar *PRECIO(CATEGORIA(i))*.

El importe de estas horas se acumula en *total* y accedemos al siguiente trabajador.

Cuando terminemos de calcular lo que va a cobrar cada trabajador imprimimos el total de todos ellos.

11.2. NUMERO DE ELEMENTOS VARIABLES EN LAS TABLAS

Al igual que sucedía en las listas unidimensionales, en las bidimensionales también se pueden dejar como variables el número de filas y el de columnas, introduciéndolas por teclado en función de las necesidades del programa.

Estos valores se introducirán antes de dimensionar el array bidimensional, ya que cuando el programa ejecute la instrucción de dimensionar la matriz debe conocer exactamente los valores máximos de los índices para reservar la memoria necesaria en el ordenador.

Ejercicio 6

Cargar en una tabla las notas de los alumnos de un colegio en función de las necesidades del mismo en cuanto al número de cursos y alumnos por curso.

ESTUDIO PREVIO

Dimensionamos el número de filas (los cursos) y las columnas (número de alumnos por curso) de la tabla desde el teclado.

VARIABLES

A = nombre de la matriz.
N = número de filas, de cursos, del colegio.
M = número de columnas, alumnos por curso.
i = índice para recorrer las filas.
j = índice para recorrer las columnas.

PSEUDOCODIGO

```
Borrar pantalla
en 10,20 imprime "Número de cursos:"
en 10,39 introduce N
en 12,20 imprime "Número de alumnos:"
en 12,40 introduce M
dimensiona A(N,M)
i = 1
borrar pantalla
en 2,25 imprime "Introducción de las notas"
hacer mientras i <= N
     en 10,25 imprime "Curso:",i
     j = 1
     hacer mientras j <= M
          en 14,25 imprime "Alumno:",j
          en 16,25 imprime "Nota:"
          en 16,32 introduce A(i,j)
```

```
        j = j + 1
    fin del hacer
    i = i + 1
fin del hacer.
fin del programa.
```

COMPROBACION

Por teclado se introduce el número de cursos, *N*, así como el número de alumnos por curso, *M*. Dimensionamos la matriz, *dimensiona A(N,M)*.

Se recorren los cursos, al ser variables será *hacer mientras i<=N*, y los alumnos que hay en cada curso, *hacer mientras j<=M*.

El resto es idéntico a los demás ejercicios realizados con las tablas.

11.3. ORDENACION DE LOS ARRAYS BIDIMENSIONALES

Se ordenan utilizando cualquiera de los métodos descritos para la ordenación de los arrays unidimensionales.

Al tener más de una columna, debemos decidir por cuál de ellas queremos ordenarla.

Además, hay que tener presente que cada vez que se produzca un intercambio de los elementos de la matriz afecta a todos los elementos de la misma fila.

Para intercambiar los elementos que están situados en la misma fila, pero en columnas distintas, al ser del mismo tipo (todos numéricos o todos alfanuméricos), utilizamos un ciclo que recorra todas las columnas para esa misma fila y los intercambie.

Ejercicio 7

Ordenar la lista L(N,M) por la primera columna utilizando el método SHELL.

VARIABLES

i = índice para recorrer las filas de la tabla.
j = índice para recorrer las columnas de la tabla.
aux = auxiliar para el intercambio de los elementos.
salto = comparar dos elementos separados entre sí esa distancia.
sw = para saber si con salto se ha producido un intercambio de elementos.

PSEUDOCODIGO

```
salto = int(N/2)
hacer mientras salto >= 1
    sw = 1
    hacer mientras sw <> 0
```

```
sw = 0
i = 1
hacer mientras i <= (N-salto)
     if L(i,1) > L(i+salto,1)
          hacer CAMBIOS
     fin del if
     i = i + 1
fin del hacer
salto = int(salto/2)
fin del hacer
fin del programa.

CAMBIOS.
j = 1
hacer mientras j <= M
     aux = L(i+salto,j)
     L(i+salto,j) = L(i,j)
     L(i,j) = aux
     j = j + 1
fin del hacer
sw = 1
```

COMPROBACION

Para recorrer la tabla sólo es necesario un ciclo, *hacer mientras i<= (N–salto)*, comparándose los elementos que están situados en la columna uno, *if L(i,1) > L(i+salto,1)*.

Si no están ordenados, se realiza el subprograma CAMBIOS, el cual, mediante el bucle *hacer mientras j <= M*, realiza el intercambio de los elementos situados en distintas filas pero en la misma columna.

RESUMEN

Un **array bidimensional**, también llamado, **matriz** o **tabla**, o un array $m \times n$ es un conjunto de datos homogéneos, cada uno de los cuales debe referenciarse por dos índices, I y J, llamados *subíndices*, que deben cumplir la propiedad de que:

$$1 \leqslant I \leqslant m \quad y \quad 1 \leqslant J \leqslant n$$

Para dimensionar las matrices se necesitan dos índices:

DIMENSIONA NOMBRE–TABLA(INDICE1,INDICE2)

donde:

- NOMBRE–TABLA es el nombre mediante el cual el ordenador va a reconocer a ese conjunto de datos homogéneos. Debe cumplir la misma normativa que las variables.
- INDICE1 es el número máximo de filas de la matriz.
- INDICE2 es el número máximo de columnas de la matriz.

Los elementos del *array* bidimensional lo representamos por:

NOMBRE-TABLA(I,J)

I, el primer índice, es la variable que toma el valor de la fila en la cual está el elemento al que deseamos acceder.
Tiene como límite inferior 1 y como límite superior el valor de INDICE1.

J, el segundo índice, es la variable que toma el valor de la columna en la cual está el elemento al que deseamos acceder.
Tiene como límite inferior 1 y como límite superior el valor de INDICE2.

Una fila es una lista horizontal de elementos.

Una columna es una lista vertical de elementos.

Los elementos de una fila tienen todos ellos igual valor para el primer subíndice, I.

Los elementos de una columna tienen todos ellos igual valor para el segundo subíndice, J.

Ambos subíndices siempre son números enteros.

Los elementos de los arrays bidimensionales, al estar almacenados en la memoria del ordenador, son de acceso directo.

Para acceder a un elemento en particular sólo hace falta indicarle al ordenador su posición, su fila y su columna, dentro del conjunto de los elementos.

Todos los elementos de un array bidimensional son del mismo tipo: numéricos o alfanuméricos.

El valor de INDICE1 e INDICE2, al igual que sucedía en las listas unidimensionales, también se pueden dejar como variables e introducir su valor por teclado o asignándoles el valor de alguna variable.

Los arrays bidimensionales se ordenan utilizando cualquiera de los métodos descritos para la ordenación de los arrays unidimensionales.

Se ordenan por la columna que deseemos.

Al intercambiar dos elementos que estén en distintas filas hay que intercambiar toda la fila completa.

EJERCICIOS PROPUESTOS

1. Generar una tabla A(N,M) cuyos elementos sean múltiplos de 3. Imprimir la tabla y los elementos de la fila media.
2. Generar una tabla A(N,M) introduciendo los valores por teclado. Imprimirla. Sumar los elementos de las columnas impares.
3. Crear una lista A(N,N) cuyas filas pares sean múltiplos de 2 y las impares sean el producto del número 3 por la posición de la columna. Imprimirla. Imprimir los elementos que cumplan la condición de pertenecer a fila par y a columna impar. Sumar aquellos elementos en los que la fila más la columna que ocupan sea múltiplo de cuatro.
4. Construir la tabla A(N,M) en la que los elementos de las filas pares sean múltiplos de 2 y los de las filas impares sean 5 por la posición de la columna.

A partir de ella crear otra B(M,N) cuyas columnas impares sean las de A elevadas al cuadrado y las pares las de A.

Imprimir ambas tablas y los elementos A(1,1) y B(2,1).

5. Construir la matriz A(N,N) de tal forma que los elementos de la diagonal sean unos y los demás elementos se introducirán por teclado. Imprimirla.

6. Construir la matriz A(N,N) de tal forma que los elementos de la diagonal sean unos y los demás elementos sean ceros. Imprimirla.

7. Hacer un programa que genere una matriz A(20,4), donde en las columnas se almacenen el nombre, el primer apellido, el segundo apellido y el número de teléfono de 20 personas. Imprimirla.

   Ordenar la tabla alfabéticamente por el primer apellido e imprimirla ordenada.

8. Hacer un programa que genere una tabla de 10×5, de tal forma que el valor de los elementos de cada columna sean, respectivamente, las tablas de multiplicar del 1, 3, 5, 7 y 9.

   Imprimir la tabla de la siguiente forma:

Número	Tabla 1	Tabla 3	Tabla 5	Tabla 7	Tabla 9
–	–	–	–	–	–
–	–	––	––	––	––
–	–	––	––	––	––

9. Generar una matriz de 12×31 donde las filas representen los meses del año y las columnas la temperatura media de cada día.

   Calcular:

   a) La temperatura media de cada mes.

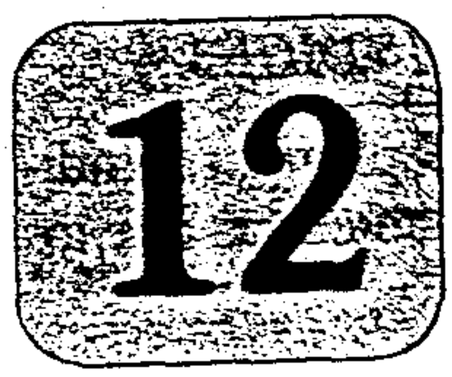
   b) El día más caluroso y el más frío de cada mes.

   c) El día más caluroso del año.

10. Dada la matriz A(10,5), cargar una nueva matriz B(10,5) de tal forma que la primera fila de B sea la última de A y que la fila que ocupa el lugar i de B sea la que ocupa el lugar i−1 de la matriz A.

11. Dada la matriz A(10,5), cargar una nueva matriz B(10,5) de tal forma que la primera columna de B sea la última de A y que la columna que ocupa el lugar i de B sea la que ocupa el lugar i−1 de la matriz A.

CAPITULO



Arrays tridimensionales

12.1. ARRAYS TRIDIMENSIONALES

Un array tridimensional o un array m × n × p es un conjunto de m×n×p datos homogéneos, del mismo tipo: numéricos o alfanuméricos, cada uno de los cuales debe referenciarse por tres índices, por ejemplo I, J y K, llamados *subíndices*, que deben cumplir la propiedad de que:

$$1 \leq I \leq m \quad y \quad 1 \leq J \leq n \quad y \quad 1 \leq K \leq p$$

Para dimensionar estos arrays lo expresamos de la forma:

DIMENSIONA NOMBRE(INDICE1,INDICE2,INDICE3)

donde:

- NOMBRE es el nombre mediante el cual el ordenador va a reconocer a ese conjunto de datos homogéneos. Debe cumplir la misma normativa de las variables.
- INDICE1 indica el número máximo de *niveles* o *páginas* del array.
- INDICE2 señala el número máximo de *filas*.
- INDICE3 hace referencia al número máximo de *columnas*.

Los elementos del array tridimensional vendrán referenciados de la siguiente forma:

NOMBRE(I,J,K)

donde:

- I es la variable que toma el valor del nivel, o página, en el cual se encuentra el elemento al que deseamos acceder.
  Tiene como límite inferior *1* y como límite superior el valor de *INDICE1*.
- J es la variable que toma el valor de la fila del elemento al que queremos acceder.
  Tiene como límite inferior *1* y como límite superior el valor de *INDICE2*.

- K es la variable que toma el valor de la columna en la cual está el elemento requerido.

  Tiene como límite inferior *1* y como límite superior el valor de *INDICE3*.

Los elementos de una misma página tienen todos el mismo valor para el primer subíndice, I. Los de una fila tienen todos igual valor para el segundo subíndice, J, mientras que los de una columna tienen igual valor para el tercero, K.

Todos los subíndices son siempre números enteros.

El hecho de asignar al primer índice la variable I, al segundo J y al tercero K sólo es cuestión de nomenclatura. No hay ningún impedimento en cambiar los nombres de las variables y tampoco en cambiar el orden: página, fila, columna, siempre y cuando seamos consecuentes en todo el programa con la nomenclatura elegida.

Ejemplo: pará un array tridimensional *A* de 3×2×4, contiene 3 · 2 · 4=24 elementos, para cuya representación utilizamos tres planos. cada uno de los cuales tiene un array bidimensional de 2×4, 2 filas y 4 columnas:

Página 3

A(3,1,1)	A(3,1,2)	A(3,1,3)	A(3,1,4)
A(3,2,1)	A(3,2,2)	A(3,2,3)	A(3,2,4)

Página 2

A(2,1,1)	A(2,1,2)	A(2,1,3)	A(2,1,4)
A(2,2,1)	A(2,2,2)	A(2,2,3)	A(2,2,4)

Página 1

A(1,1,1)	A(1,1,2)	A(1,1,3)	A(1,1,4)
A(1,2,1)	A(1,2,2)	A(1,2,3)	A(1,2,4)

Supongamos que tenemos en la memoria del ordenador el array tridimensional NOMBRES(3,2,5), con los nombres siguientes, tal como indica la figura:

Página 1

Filas					
1			Celeste		
2	Evaristo				
Columnas	1	2	3	4	5

Página 2

Filas					
1		Vicenta			
2		Manuel			Tina
Columnas	1	2	3	4	5

Página 3

Filas					
1					Matea
2	José	Luisa			
Columnas	1	2	3	4	5

de tal forma que:

NOMBRES(1,2,1) = Evaristo
NOMBRES(2,2,2) = Manuel
NOMBRES(2,2,5) = Tina
NOMBRES(3,2,1) = José
NOMBRES(3,1,5) = Matea
NOMBRES(1,1,3) = Celeste
NOMBRES(2,1,2) = Vicenta
NOMBRES(3,2,2) = Luisa

Algunos lenguajes tienen limitada la dimensión de una lista tridimensional.

Aunque se pueden diseñar arrays de dimensión superior a tres, en la práctica no se construyen, por las siguientes causas:

— Dificultad en acceder a algunos datos en particular.
— Posible repetición de alguno de los datos.
— Posible pérdida de espacio al no estar ocupadas todas las posiciones.

Ejercicio 1

Crear una tabla de 3×3×3 donde el primer elemento valga 1, el segundo 2, el tercero 3, y así sucesivamente. Imprimirla.

ESTUDIO PREVIO

Para crear los elementos necesitamos un contador que aumente de uno en uno. A medida que vamos generando la lista tridimensional podemos ir visualizándola, utilizando el mismo formato de páginas expuesto en el apartado anterior.

VARIABLES

A = nombre de la lista.
i = índice de las páginas.
j = índice de las filas.
k = índice de las columnas.
b = contador para generar los números.
fi = contador de las filas.
co = contador de las columnas.

PSEUDOCODIGO

```
dimensiona A(3,3,3)
i = 1
b = 0
hacer mientras i <= 3
     borrar pantalla
     fi = 8
     co = 12
     en fi,co imprime "Elementos de la página:",i
     fi = fi + 2
     j = 1
     hacer mientras j <= 3
          k = 1
          hacer mientras k <= 3
               b = b + 1
               A(i,j,k) = b
               en fi,co imprime A(i,j,k)
               co = co + 4
               k = k + 1
          fin del hacer
          fi = fi + 2
          co = 12
          j = j + 1
     fin del hacer
     en fi+2,20 imprime "Pulse ENTER para continuar"
     detener-listado
     i = i + 1
fin del hacer
fin del programa.
```

COMPROBACION

Empezamos a recorrer el array por el nivel uno, *i=1*, hasta el último, *hacer mientras i<=3*.

Borramos pantalla para que sólo aparezcan en ella los elementos de cada página. Inicializamos el contador de las páginas y la variable para generar los números.

Al igual que en los arrays bidimensionales, la inicialización de las filas irá antes de su ciclo, para que cuando accedamos a la siguiente página empiece siempre desde la fila uno. Imprimimos la página actual.

Dentro de este nivel empezamos a recorrer las filas desde la primera, *j=1*, hasta la última, *hacer mientras j<=3*.

La inicialización de las columnas es análoga a la de las filas; irá antes de su ciclo.

Para cada fila recorremos todas las columnas empezando en la primera, *k=1*, hasta la última, *hacer mientras k<=3*.

Empezamos a generar los números en la variable *b*, lo almacenamos en su lugar correspondiente dentro del array, *A(i,j,k) = b*, y lo imprimimos.

Cuando haya finalizado de imprimir los elementos de la página *i*, detenemos el listado y accedemos a la página siguiente.

Ejercicio 2

Se dispone de una lista de 5×10×20 elementos donde está almacenada la nota de un alumno.

El primer índice se refiere al centro, el segundo al curso y el tercero al número de alumnos.

Imprimir:

a) La nota media por curso.
b) La nota media máxima y su centro de pertenencia.

Formato de impresión:

Centro	Nota media
–	––
–	––
–	––

Nota media máxima: ––
pertenece al centro: –

ESTUDIO PREVIO

Leeremos las notas por curso, las sumamos y calculamos la media dividiendo por el número de alumnos del curso, en nuestro caso 20.

Esta nota media la comparamos con la variable que almacena la mayor de las medias; si es mayor, guardamos dicha nota y el centro al que pertenece.

VARIABLES

A = tabla donde están almacenadas las notas.
i = índice de las páginas.
j = índice de las filas.
k = índice de las columnas.
sum = acumulador de las notas por curso.
fi = contador de las filas.
max = almacena la nota media máxima.
centro = almacena el centro al cual pertenece *max*.

PSEUDOCODIGO

```
Borrar pantalla
en 8,18 imprime "Centro"
en 8,38 imprime "Nota media"
max = -1
centro = 0
fi = 10
i = 1
hacer mientras i <= 5
     sum = 0
     j = 1
     hacer mientras j <= 10
          k = 1
          hacer mientras k <= 20
               sum = sum + A(i,j,k)
               k = k + 1
          fin del hacer
          j = j + 1
     fin del hacer
     en fi,20 imprime i
     en fi,40 imprime sum/20
     fi = fi + 2
     if sum/20 > max
          max = sum/20
          centro = i
     fin del if
     i = i + 1
fin del hacer
en fi+2,20 imprime "Nota media máxima:",max
en fi+4,20 imprime "pertenece al centro:",centro
fin del programa.
```

COMPROBACION

Fuera de los ciclos imprimimos la cabecera e inicializamos las variables que sólo hay que hacer una vez.

Sumamos las notas de cada curso, *sum = sum + A(i,j,k)*. Al terminar imprimimos el curso de que se trata, *i*, así como la nota media, *sum/20*.

Preguntamos si es mayor que el máximo, *if sum/20 > max;* en caso de serlo, se guarda en *max* la media del curso actual y el centro al cual pertenece, *centro = i*.

Cuando terminemos de recorrer todo el array se imprime la nota media máxima y el centro al que pertenece.

Ejercicio 3

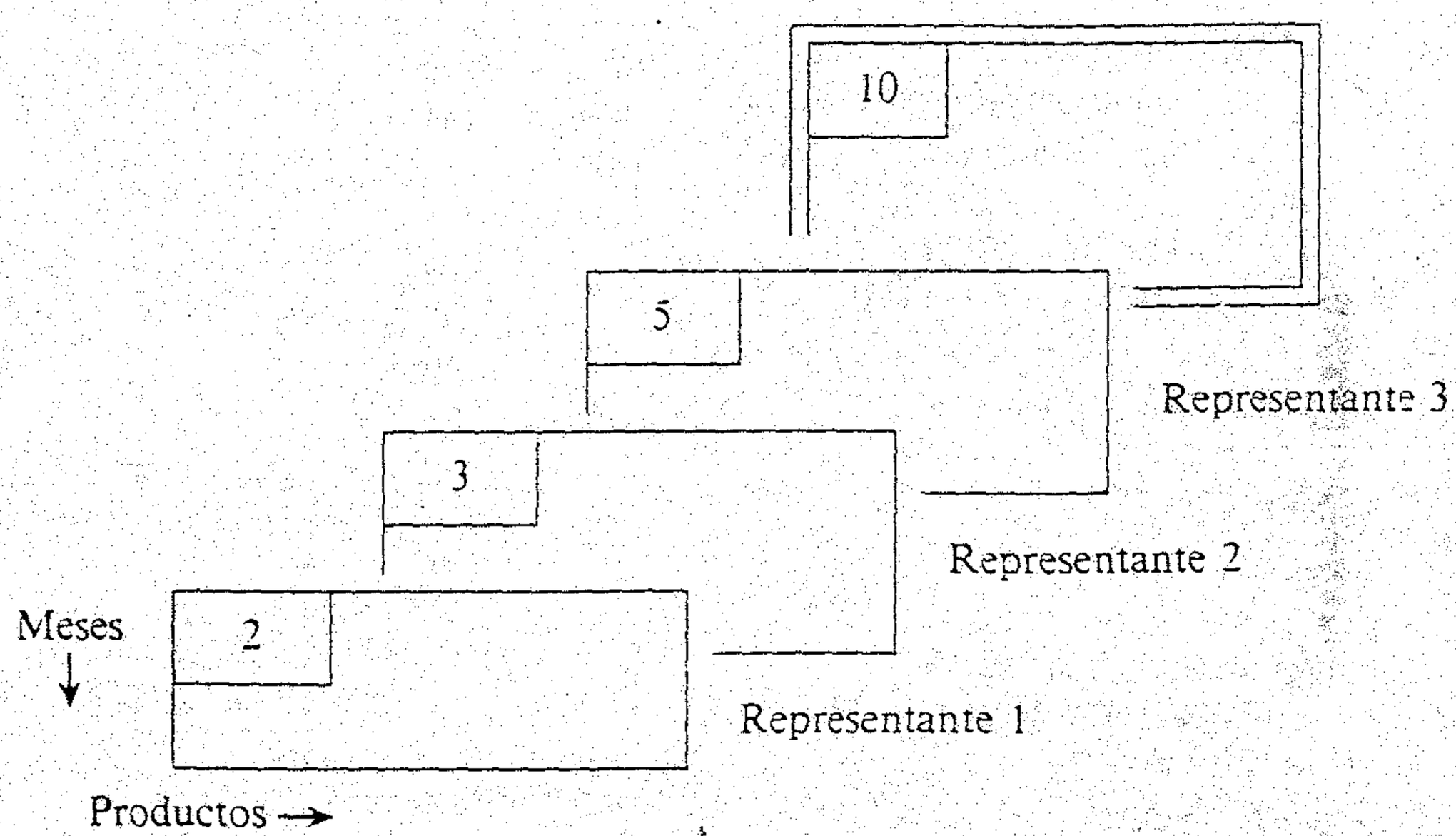
Una empresa guarda en un array de 3×12×4 las ventas realizadas por sus tres representantes, de sus cuatro productos a lo largo de los doce meses. VENTAS(representante,mes, producto).

Queremos proyectar el array tridimensional sobre uno de dos dimensiones que represente el total de ventas TOTAL(mes,producto), para lo cual sumamos las ventas de cada producto de cada mes de todos los representantes.

Imprimir ambos arrays, el de tres dimensiones y el de dos.

ESTUDIO PREVIO

Nos están pidiendo que sumemos las ventas de la siguiente forma:



Recorreremos el array de tres dimensiones por los productos, dentro de ellos por los representantes y los sumamos.

Una vez realizada la suma de las ventas de los tres representantes, su valor lo almacenamos en el array bidimensional, en la misma posición del mes y del producto que en el array tridimensional.

Una vez creada la matriz de dos dimensiones imprimimos ambas listas.

VARIABLES

TOTAL(12,4) = array bidimensional suma por representantes.
i = índice para recorrer los representantes.
j = índice para recorrer los meses en ambos arrays.
k = índice para recorrer los productos en ambas listas.
suma = acumulador de las ventas de los representantes.
fi = contador de las filas.
co = contador de las columnas.

PSEUDOCODIGO

```
hacer VOLCAR
hacer IMP-TRES
hacer IMP-DOS
fin del programa.
VOLCAR.
dimensiona TOTAL(12,4)
j = 1
hacer mientras j <= 12
      k = 1
      hacer mientras k <= 4
            i = 1
            suma = 0
            hacer mientras i <= 3
                  suma = suma + VENTAS(i,j,k)
                  i = i + 1
            fin del hacer
            TOTAL(j,k) = suma
            k = k + 1
      fin del hacer
      j = j + 1
fin del hacer
IMP-TRES
i = 1
hacer mientras i <= 3
      borrar pantalla
      fi = 8
      co = 12
      en fi,co imprime "Ventas del representante:",i
      fi = fi + 2
      j = 1
      hacer mientras j <= 12
            k = 1
            hacer mientras k <= 4
                  en fi,co imprime VENTAS(i,j,k)
                  co = co + 4
                  k = k + 1
            fin del hacer
            fi = fi + 2
            co = 12
            j = j + 1
      fin del hacer
      detener-listado
      i = i + 1
fin del hacer.
```

```
IMP-DOS.
borrar pantalla
j = 1
en 8,20 imprime "Ventas totales"
fi = 10
co = 16
hacer mientras j <= 12
     k = 1
     hacer mientras k <= 4
          en fi,co imprime TOTAL(j,k)
          co = co + 4
          k = k + 1
     fin del hacer
     fi = fi + 2
     co = 12
     j = j + 1
fin del hacer.
```

COMPROBACION

En el subprograma VOLCAR se realiza el volcado del array de tres dimensiones sobre el de dos; posteriormente imprimimos el array tridimensional en IMP-TRES, detenemos el listado y, por último, se visualiza la matriz de dos dimensiones en IMP-DOS.

Una vez dimensionada la tabla TOTAL(12,4), empezamos a recorrer la tabla de VENTAS por meses, *hacer mientras j<=12*. Dentro de cada mes por productos, *hacer mientras k<=4*, y dentro de productos por representantes, *hacer mientras i<=3*. Efectuamos la suma, *suma = suma + VENTAS(i,j,k)*.

Cuando hayamos recorrido todos los representantes se guarda el valor de la suma en la matriz. *TOTAL(j,k) = suma*.

Accedemos al siguiente producto e inicializamos el valor del acumulador.

Visualizamos el array tridimensional VENTAS, recorriéndolo tal como lo hemos realizado hasta ahora.

Detenemos el listado, después de imprimir las ventas del último representante, y comenzamos a visualizar el array bidimensional TOTAL.

Para recorrer este array, utilizamos los índices *j* y *k*, aunque se puede utilizar cualquier par de índices.

Ejercicio 4

Tenemos el array A(4,4,4). Sumar los elementos situados de la forma siguiente:

- I es impar:
  a) Si J es impar, entonces K es impar.
  b) Si J es par, entonces K es par.

- I es par:
  a) Si J es par, entonces K es impar.
  b) Si J es impar, entonces K es par.

Es decir, se suman los elementos de una forma alternativa empezando por el 1,1,1.

ESTUDIO PREVIO

El índice de las columnas, K, siempre va de dos en dos, pero unas veces empieza en uno y otras en dos, dependiendo de que el valor de las filas, J, sea par o impar, que a su vez depende del nivel, I.

VARIABLES

i = índice para recorrer las páginas del array.
j = índice para recorrer las filas.
k = índice para recorrer las columnas.
suma = acumulador de los elementos situados en las posiciones requeridas.

PSEUDOCODIGO

```
i = 1
suma = 0
hacer mientras i <= 4
     j = 1
     if i = int(i/2)*2
          hacer I-PAR
     else
          hacer I-IMPAR
     fin del if
     i = i + 1
fin del hacer
borrar pantalla
en 10,20 imprime "La suma es:",suma
fin del programa.

I-PAR.
hacer mientras j <= 4
     if j = int(j/2)*2
          k = 1
     else
          k = 2
     fin del if
     hacer mientras k <= 4
          suma = suma + A(i,j,k)
          k = k + 2
     fin del hacer
     j = j + 1
fin del hacer
```

```
I-IMPAR.
hacer mientras j <= 4
     if j = int(j/2)*2
          k = 2
     else
          k = 1
     fin del if
     hacer mientras k <= 4
          suma = suma + A(i,j,k)
          k = k + 2
     fin del hacer
     j = j + 1
fin del hacer
```

COMPROBACION

Empezamos a recorrer el array desde la página uno. Comprobamos si la página actual es par o impar, *if i=int(i/2)*2:*

— Par: ejecuta el subprograma I–PAR.
— Impar: ejecuta el subprograma I–IMPAR.

Tanto en uno como en otro, recorremos las filas empezando en la primera; antes de ejecutarlos inicializamos *j = 1*.

En I–PAR recorremos todas las filas, *hacer mientras j<=4*, y comprobamos si es par o impar, *if j=int(j/2)*2:*

— Par: inicializamos k a 1.
— Impar: k vale 2.

Recorremos todas las columnas, *hacer mientras k<=4*, y vamos sumando los valores de los elementos, *suma = suma+A(i,j,k)*.

El valor del índice k lo incrementamos en dos, por realizar la suma de los elementos de una forma alternativa.

Hacemos lo mismo en I–IMPAR, pero en este caso, si *j* es par, entonces *k = 2*; en caso contrario, *k = 1*.

Al terminar de recorrer todo el array imprimimos el valor de la suma.

RESUMEN

Un **array tridimensional** o un array $m \times n \times p$ es un conjunto de m×n×p datos homogéneos, del mismo tipo: numéricos o alfanuméricos, cada uno de los cuales debe referenciarse por tres índices, I, J y K, llamados subíndices.

Los subíndices deben cumplir la propiedad de que:

$$1 \leq I \leq m \quad y \quad 1 \leq J \leq n \quad y \quad 1 \leq K \leq p$$

Para dimensionar un array tridimensional lo expresamos de la forma:

DIMENSIONA NOMBRE(INDICE1,INDICE2,INDICE3)

NOMBRE es el nombre mediante el cual el ordenador va a reconocer a ese conjunto de datos homogéneos. Debe cumplir la misma normativa que las variables.

INDICE1 indica el número máximo de *niveles* o *páginas* del array.

INDICE2 señala el número máximo de *filas*.

INDICE3 hace referencia al número máximo de *columnas*.

Los elementos del array tridimensional vendrán designados de la siguiente forma:

NOMBRE(I,J,K)

I es la variable que toma el valor del nivel, o página, en el cual se encuentra el elemento al que deseamos acceder.
Su límite inferior es *1* y el superior el valor de *INDICE1*.

J es la variable que toma el valor de la fila del elemento al que queremos acceder.
Su límite inferior *1* y el superior el valor de *INDICE2*.

K es la variable que toma el valor de la columna en la cual está el elemento requerido.
Su límite inferior *1* y el superior el valor de *INDICE3*.

Los elementos de una misma página tienen todos ellos el mismo valor para el primer subíndice, I.

Los elementos de una misma fila tienen todos ellos igual valor para el segundo subíndice, J.

Los elementos de una misma columna tienen todos ellos igual valor para el tercer subíndice, K.

Todos los subíndices siempre son números enteros.

La designación del nombre de los subíndices para las páginas, filas y columnas, así como la función de cada uno de ellos, es elección nuestra.
Sólo debemos ser consecuentes en todo el programa con la nomenclatura elegida.

Algunos lenguajes tienen limitada la dimensión de una lista tridimensional.

EJERCICIOS PROPUESTOS

1. En un array A(5,6,30) donde el primer índice señala el número de centros, el segundo el número de cursos por centro y el tercero el número de alumnos por curso, se guardan las notas de los alumnos.
   Se pide:
   — La nota media por curso.
   — La nota media por centro.
   — La nota media global.
   — Los dos mejores centros y sus notas.
   — El mejor alumno de cada curso con su nota correspondiente.

2. En un array se guardan los siguientes datos referidos a una población:

   T(estado civil, sexo, edad)

donde:

— El estado civil puede tomar los valores de 1 a 3 (soltero, casado, viudo).
— El sexo, 1 ó 2 (masculino, femenino).
— La edad, un valor comprendido entre 1 y 65.

Imprimir el número de personas que hay de un estado civil, sexo y edad determinada, que se introducirán por teclado.

3. Supongamos que tenemos un array cuyo contenido es el importe de las ventas diarias de los mercados centrales de toda España, cuyas dimensiones son 31×12×51, donde el primer índice indica los días del mes, el segundo los meses del año y el tercero las provincias de España. Se pide:

   — Calcular la provincia con mayor importe de ventas al año.
   — Listar ordenadamente los meses según el importe de ventas en toda España.
   — La desviación sobre la media mensual en toda España.

4. Una red de almacenes mantiene en un array de 30×10×200 los stocks, de sus productos, donde el primer índice señala el almacén, el segundo el departamento y el tercero el número del producto.

   También dispone de otro array de las mismas características, cuyo contenido es el stock mínimo de cada producto.

   Realizar un programa que realice el mantenimiento, entrada y salida de productos de los almacenes sobre estos arrays y al final del día realice la valoración de los mismos: para ello dispone de una tabla 10×200 con el precio unitario de cada artículo.

5. Un equipo de baloncesto de 10 jugadores guarda en un array de 10×20×6 sus estadísticas, donde el segundo índice indica el número de partidos que hay en el campeonato y el tercero las veces que ese jugador ha tirado a canasta desde la posición de tiro libre, de dos puntos y de tres puntos, y las veces que acertó en cada una de ellas.

   En otro array unidimensional almacena el nombre de cada jugador, que se corresponde con el array de las estadísticas.

   Se desea hacer un programa con las siguientes opciones:

   — Imprimir las estadísticas de todos los jugadores de la plantilla.
   — Estadísticas de un jugador determinado.
   — Nombre de los jugadores con el mejor y el peor porcentaje de tiros.

CAPITULO

Ficheros secuenciales

13.1. ESTRUCTURAS EXTERNAS DE DATOS: FICHEROS

Los datos tratados por los programas, que hemos utilizado hasta ahora, tienen limitaciones importantes:

- La cantidad de datos viene limitada por la capacidad de la memoria principal de nuestro ordenador.
- La duración en el tiempo; es decir, una vez terminado de ejecutar nuestro programa los datos se pierden por estar almacenados en la memoria principal del ordenador.

Para salvar estos inconvenientes se utilizan las estructuras externas de datos: los ficheros o archivos.

Los ficheros tienen la particularidad de estar almacenados en soportes externos (como las cintas magnéticas o los discos flexibles) o en la memoria del ordenador, con lo cual la duración en el tiempo es ilimitada y podremos utilizar esos datos cuantas veces lo deseemos.

Un **fichero** o **archivo** es un conjunto de registros lógicos, todos ellos de las mismas características. Vienen referenciados de la siguiente forma:

Nombre–del–fichero.Extension

- **Nombre–del–fichero:** debe cumplir la misma normativa que el nombre de las variables.
- **Extensión:** es un nombre de tres caracteres, de los cuales el primero debe ser una letra.
  Ejemplos: DATOS.DAT, ALUMNOS.COB, ARTICULO.BAS, PERSONAL.DBF.

Registro lógico o **registro** es cada uno de los componentes del fichero.

Está formado por un conjunto de campos que son tratados por el ordenador de forma unitaria.

Un **campo** es un conjunto de caracteres que expresan una información.

Un campo puede estar compuesto por *subcampos*.

Los campos dentro de un registro pueden ser heterogéneos, de distintos tipos: una mezcla de campos numéricos y alfanuméricos.

Sea el fichero DATOS.DAT, cuya estructura es la siguiente:

DNI	NOMBRE	APELLIDOS	DIRECCION	PROVINCIA
11111	Juan José	Calvo	Rua 12	Valencia
34512	Patricia	Rico San José	Sol 55	Valladolid
21322	David G.	Rodríguez	Centro 99	Valladolid
33333	Esperanza	Rodríguez	Páramo 33	León
12121	José M.	Robles	Carabela 1	Madrid
66777	Marisa	Robles	Carabela 1	Madrid
44444	Marisa	Robles	Carabela 1	Madrid
/*				

donde /* es la marca de final de fichero, que indica que ya no hay ningún registro más.

Los campos son: DNI, NOMBRE, APELLIDOS, DIRECCION y PROVINCIA.

Los registros son toda la información que hay en una línea: por ejemplo, el registro número dos es:

DNI = 34512
NOMBRE = Patricia
APELLIDOS = Rico San José
DIRECCION = Sol 55
PROVINCIA = Valladolid

El campo APELLIDOS podemos desdoblarlo en dos subcampos: primer apellido y segundo apellido. Análogamente la DIRECCION en calle y número.

Es conveniente que el fichero posea un *campo clave* o *identificativo* mediante el cual vamos a acceder a toda la información contenida en el registro.

El valor de este campo clave será único en todo el fichero: no se puede repetir. De tal forma que cada registro tenga al menos un campo identificativo y cada campo identificativo corresponda o identifique a un solo registro del fichero.

En el ejemplo anterior el campo clave es el DNI. Cada persona, cada registro, posee un número del DNI propio e intransferible.

Otro ejemplo de un fichero sería la guía telefónica. Los registros son todos los abonados, y los campos, los datos personales y el número de teléfono de cada uno de ellos. El campo identificativo sería el número telefónico, aunque en este caso un abonado puede tener más de un número, pero no hay dos personas con igual número de teléfono.

Para acceder a la información de un registro, generalmente lo haremos a través de su campo identificativo.

También se puede acceder por cualquier otro campo del registro. En este caso puede haber más de un registro con la misma información, como sucede con los registros:

21322	David G.	Rodríguez	Centro 99	Valladolid
33333	Esperanza	Rodríguez	Páramo 33	León

que tienen igual el campo APELLIDOS, o los registros:

66777	Marisa	Robles	Carabela 1	Madrid
44444	Marisa	Robles	Carabela 1	Madrid

que sólo se diferencian en el DNI, el campo identificativo.

13.2. CARACTERISTICAS DE LOS FICHEROS

Los ficheros presentan las siguientes características:

- Independencia de la información respecto a los programas que acceden a dicha información.
- Se puede acceder a su información por diversos programas en distintos momentos.
- Gran capacidad de almacenamiento.
- Duración en el tiempo.

13.3. CLASIFICACION DE LOS FICHEROS SEGUN SU USO

Se clasifican en tres grupos:

- **Ficheros permanentes:** contienen información que varía poco en el tiempo. En algunos casos es menester actualizarlos periódicamente. Se subdividen en tres grupos:
  - *a*) **Ficheros de constantes:** su información permanece prácticamente inamovible. Se utilizan principalmente como consulta. Por ejemplo: red del metro de una ciudad, tabla de logaritmos, empleados de una empresa.
  - *b*) **Ficheros de situación** o **maestros:** contienen la información actualizada. Por ejemplo: stock de un almacén, estado de las cuentas corrientes.
  - *c*) **Ficheros históricos:** contienen información acerca de situaciones ya pasadas. Se suelen utilizar para estadística.
- **Ficheros de movimientos** o de **transacciones:** contienen la información para actualizar los ficheros maestros. Los registros del fichero de movimientos pueden ser de tres clases: *altas, bajas* y *modificaciones.*

  Una vez realizada la actualización del maestro, el fichero de movimientos pierde su valor y podemos borrarlo.
- **Ficheros de maniobra** o **de trabajo:** son de vida limitada, utilizándose como auxiliares de los anteriores.

13.4. OPERACIONES SOBRE LOS FICHEROS

Las más usuales son:

- **Creación:** consiste en la escritura o grabación de los registros que van a conformar el fichero. Los datos pueden introducirse por teclado, desde otro fichero o como resultado de algún proceso intermedio.

- **Consulta:** lectura de uno o todos los registros.
- **Actualización:** consiste en añadir, modificar o dar de baja a algún registro.
- **Clasificación:** reubicación de los registros de tal forma que queden ordenados por algún campo determinado.
- **Fusión o mezcla:** a partir de dos ficheros de idéntica estructura, obtener un nuevo fichero con los registros de los anteriores.
- **Partición:** descomponer un fichero en dos, atendiendo a alguna característica de algún campo.
- **Borrado:** eliminación física del fichero sobre el soporte, dejando libre el espacio que ocupaba.

13.5. ORGANIZACION DE LOS FICHEROS

Para su almacenamiento y para acceder a sus datos, los ficheros se organizan de las siguientes formas:

- Secuencial.
- Directa o aleatoria.
- Secuencial indexada.

13.6. FICHEROS CON ORGANIZACION SECUENCIAL

En los ficheros con organización secuencial los registros se escriben en orden consecutivo, en secuencia, y se leen en el mismo orden en que están escritos, empezando por el primero.

El ordenador, para leer el registro situado en la posición n, previamente ha tenido que pasar por todos los registros que ocupan posiciones anteriores a dicho registro.

Entre dos registros consecutivos no puede haber ningún hueco.

Para acceder a la información de los ficheros, primero hay que abrirlos, como si de un libro se tratara. Al abrirlos el puntero se sitúa en el primer registro.

El **puntero** es una información interna del ordenador para saber en qué registro está posicionado, activo, en cada instante.

Funciona de manera análoga a cuando nosotros buscamos una determinada información en un volumen grande de la misma.

Por ejemplo: una palabra en el diccionario; nos ayudamos del dedo índice o de un objeto para controlar mejor la palabra que hemos leído.

- **Instrucciones de los ficheros secuenciales.**

Para abrir ficheros:

abrir nombre–del–fichero

Siempre que se abra el fichero, el puntero se sitúa en el primer registro.
Para leer el registro siguiente:

leer siguiente registro

y el puntero se coloca en el siguiente registro.

Si continuamos leyendo, llegará un momento en que hemos leído todos los registros del fichero, y el puntero encuentra la marca del final del fichero.

Una vez que el ordenador lee la marca de final de fichero, y se encuentra con la instrucción *leer siguiente registro,* da error.

Sólo se puede leer del fichero mientras no sea final del mismo; lo indicamos de la forma:

hacer mientras no sea EOF()

EOF: es una palabra reservada usada en casi todos los lenguajes; son las iniciales de End Of File: final de fichero.

Escribir o grabar un nuevo registro en el fichero:

escribir registro

previamente debemos indicarle al ordenador que abra un hueco entre el último registro y la marca de final de fichero para acoger al nuevo registro. Lo expresamos:

posicionarse al final del fichero

No se puede insertar, dar de alta, a un registro entre dos consecutivos, es decir, al principio del fichero o en posiciones intermedias; sólo se pueden dar altas al final del fichero, después del último registro, indicándoselo de la forma que acabamos de ver.

Para modificar información en un registro existente:

regrabar registro

Cerrar ficheros:

cerrar nombre–del–fichero

Para buscar, borrar, grabar o modificar información en un fichero, obligatoriamente, debe estar abierto.

Si el fichero está cerrado, ya no podemos leer ni escribir sobre él.

Veamos a continuación unos ejercicios estándar sobre las operaciones más comunes que puede soportar un fichero, tales como: añadir, borrar, modificar registros. Operaciones que siempre van acompañadas de la lectura de un número indeterminado de registros.

Ejercicio 1

Hacer un programa que nos permita dar altas en el fichero secuencial DATOS.DAT, cuyos campos son:

DNI	NOMBRE	APELLIDOS	DIRECCION	PROVINCIA

controlando las altas duplicadas.

ESTUDIO PREVIO

Introduciremos por teclado el DNI y comprobaremos si ya existe o no.

Para saber si existe, hay que leer el fichero desde el primer registro hasta el último, comparando el campo DNI con el valor introducido por teclado.

En caso de que exista se imprimirá el mensaje de "Alta duplicada" y se deja de leer inmediatamente, para lo cual utilizaremos un interruptor, un *sw*.

Si hemos leído todos los registros, hemos llegado a su final y no se ha encontrado. Entonces se introducirán los demás datos y se graba el registro en el fichero. Por esta razón, en un principio solamente se introducirá el campo indicativo y no hará falta introducir los demás datos, pues en el caso de que ya esté no hay que grabar el registro en el fichero.

Supongamos que tenemos los siguientes valores del DNI en nuestro fichero de DATOS. DAT, tal como indica la Figura 13.1.

Vamos a dar de alta al DNI número:

- 22: está, una vez encontrado dejamos de leer. Imprimimos "Alta duplicada".
- 44: leemos hasta el final. No está, le damos de alta. El ordenador lo graba después del último registro.

DNI
33
77
22
11
55
/*

a) Situación inicial del fichero

DNI
33
77
22
11
55
44
/*

b) Situación después de dar de alta al DNI número 44.

Figura 13.1.

Primero le indicaremos al ordenador que abra un hueco para poder grabar el nuevo registro.

VARIABLES

num = para introducir el DNI del nuevo registro.
sw = controla si el valor *num* está o no en el fichero.
res = para seguir introduciendo más registros.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     abrir DATOS
```

```
sw = 0
num = 0
borrar pantalla
en 5,10 imprime "D.N.I.:"
en 5,18 introduce num
hacer mientras no sea EOF()  AND  sw = 0
     if DNI = num
          en 10,10 imprime "Alta duplicada"
          en 15,10 imprime "Pulse RETURN para continuar"
          detener-listado
          sw = 1
     else
          leer siguiente registro
     fin del if
fin del hacer
if sw = 0
     en 7,5 imprime "Nombre:"
     en 9,5 imprime "Apellidos:"
     en 11,5 imprime "Dirección:"
     en 13,5 imprime "Provincia:"
     en 7,16 introduce nombre
     en 9,16 introduce apellidos
     en 11,16 introduce direccion
     en 13,16 introduce provincia
     dni = num
     posicionarse al final del fichero
     escribir registro
fin del if
cerrar DATOS
res = espacio(1)
     hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

Introducimos el valor del campo indicativo y recorremos todo el fichero, desde el primer registro hasta el último, o bien hasta encontrarlo, *hacer mientras no sea EOF() AND sw=0.*

Comparamos el campo con la variable, *if DNI=num.* Puede suceder:

— Son iguales: está, es alta duplicada. Debemos abandonar la búsqueda; para ello asignamos *sw=1.*
— No son iguales: leemos el siguiente registro.

Cuando salga del ciclo puede ser por dos motivos:

— Sea alta duplicada; el *sw* está a uno.
— Es final del fichero y no hemos encontrado el dni; el *sw* está a cero.

Para saber por dónde hemos salido, preguntamos por el valor del *sw*. Si es cero: introducimos los demás datos en los campos, le indicamos al ordenador que deseamos grabar un registro, para lo cual debe posicionar el puntero al final del archivo, abrir un hueco, y grabamos el registro.

Tanto en un caso como en otro cerramos el fichero y ejecutamos el programa MAS.

La operación de cerrar el fichero dentro del bucle es por si damos una nueva alta: hay que empezar a leer desde el primer registro para comprobar si es un alta duplicada. La forma de realizarlo es cerrando y abriendo de nuevo, con lo cual el puntero se sitúa al principio del fichero.

Ejercicio 2

Hacer un programa que nos permita dar bajas en el fichero secuencial DATOS.DAT, controlando las bajas inexistentes.

ESTUDIO PREVIO

Los registros en la organización secuencial están juntos físicamente, sin huecos entre ellos.

Para dar de baja a una serie de registros nos ayudamos de un fichero secuencial auxiliar, AUX.DAT. Este fichero debe tener la misma estructura, es decir, los mismos campos y del mismo tipo, numéricos o alfanuméricos, que el fichero de DATOS.

Primero buscamos el registro a dar de baja, para lo cual hay que leer. A medida que leemos puede suceder:

- No es el buscado: lo grabamos en un fichero auxiliar.
- Si es el buscado: no se graba en el auxiliar. Pero debemos seguir leyendo hasta el final para pasar el resto de los registros y no perderlos.

Supongamos que tenemos los siguientes valores del campo DNI y deseamos dar de baja al 22. La situación de los ficheros DATOS.DAT y AUX.DAT es:

Campo	DNI	
Ficheros	DATOS.DAT	AUX.DAT
Registros	33 77 22 11 55 /*	33 77 11 55 /*

Una vez pasados todos los registros, menos el que hemos dado de baja, tenemos la información por duplicado. Borramos el fichero de DATOS y renombramos el actualizado

como el de DATOS. Para realizar las operaciones de borrar y renombrar archivos, éstos deben estar cerrados. Lo expresamos de la forma:

borrar nombre–fichero
renombrar nombre–fichero como nuevo–nombre–fichero.

Si el dato no está en el fichero DATOS, se imprimirá el mensaje de "Baja inexistente".

Ejemplo: deseamos dar de baja del fichero DATOS.DAT al DNI de valor 44. La situación de ambos ficheros sería la siguiente:

Campo	DNI	
Ficheros	DATOS.DAT	AUX.DAT
Registros	33 77 22 11 55 /*	33 77 22 11 55 /*

Al no estar el registro a dar de baja en el fichero, ambos ficheros contienen igual información: borramos el fichero AUX.DAT.

VARIABLES

AUX = fichero auxiliar donde se van a grabar los registros que no damos de baja del archivo DATOS.
num = para introducir el dni a dar de baja.
sw = controla si el DNI a dar de baja está o no en el fichero.
res = para dar más bajas.

PSEUDOCODIGO

```
re = "S"
hacer mientras res = "S"
      abrir DATOS
      abrir AUX
      sw = 0
      borrar pantalla
      en 5,10 imprime "D.N.I.:"
      en 5,18 introduce num
      hacer mientras no sea EOF() del fichero DATOS
            if DNI = num
                  sw = 1
            else
                  posicionarse al final de fichero AUX
```

```
            grabar registro en AUX
        fin del if
        leer siguiente registro del fichero de DATOS
    fin del hacer
    cerrar DATOS
    cerrar AUX
    if sw = 0
        en 12,10 imprime "Baja inexistente"
        en 16,10 imprime "Pulse RETURN para continuar"
        borrar AUX
        detener-listado
    else
        borrar DATOS
        renombrar AUX como DATOS
    fin del if
    res = espacio(1)
    hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

Introducimos el valor del campo indicativo y recorremos todo el fichero DATOS, desde el primer registro hasta el último, preguntando si ese DNI es el buscado, *if DNI=num*. Puede suceder:

- Son iguales: está, no se graba en AUX y se pone el *sw* a uno para saber que se le ha dado de baja.
- No son iguales: grabamos el registro de DATOS en AUX.

En ambos casos debe seguir leyendo del de DATOS.

Cuando termine de leer todos los registros de DATOS, cerramos ambos ficheros y preguntamos por el valor de *sw*:

- Es cero: no estaba. Al no haber dado a ningún registro de baja, ambos ficheros contienen igual información: borramos el AUX.
- Es uno: se ha dado de baja. El fichero que contiene la información actualizada es AUX; borramos el archivo de DATOS y renombramos al fichero AUX.DAT como DATOS.DAT.

Tanto en un caso como en otro realiza el programa MAS.

Actualmente hay lenguajes que la operación de borrar un registro la realizan en dos pasos:

- *Marcar* o *señalar* los registros que deseamos borrar. Estos registros continúan físicamente en el fichero, pero delante del primer campo tienen una marca, un asterisco: *.
- *Empaquetar* o *compactar* los registros no marcados. Es decir, borrar físicamente del fichero los registros marcados.

Al desaparecer los registros marcados, los que ocupan posiciones posteriores son desplazados por el ordenador tantos lugares como registros se hayan borrado.

El fichero queda estructurado, sin huecos entre registros.

En estos casos no es necesario la utilización de ningún fichero auxiliar, ya que todas las operaciones se están realizando sobre el propio fichero.

Supongamos que deseamos borrar los registros cuyo DNI tiene los valores 22 y 77 en nuestro fichero secuencial DATOS.DAT. Los pasos a seguir son los expresados en la Figura 13.2.

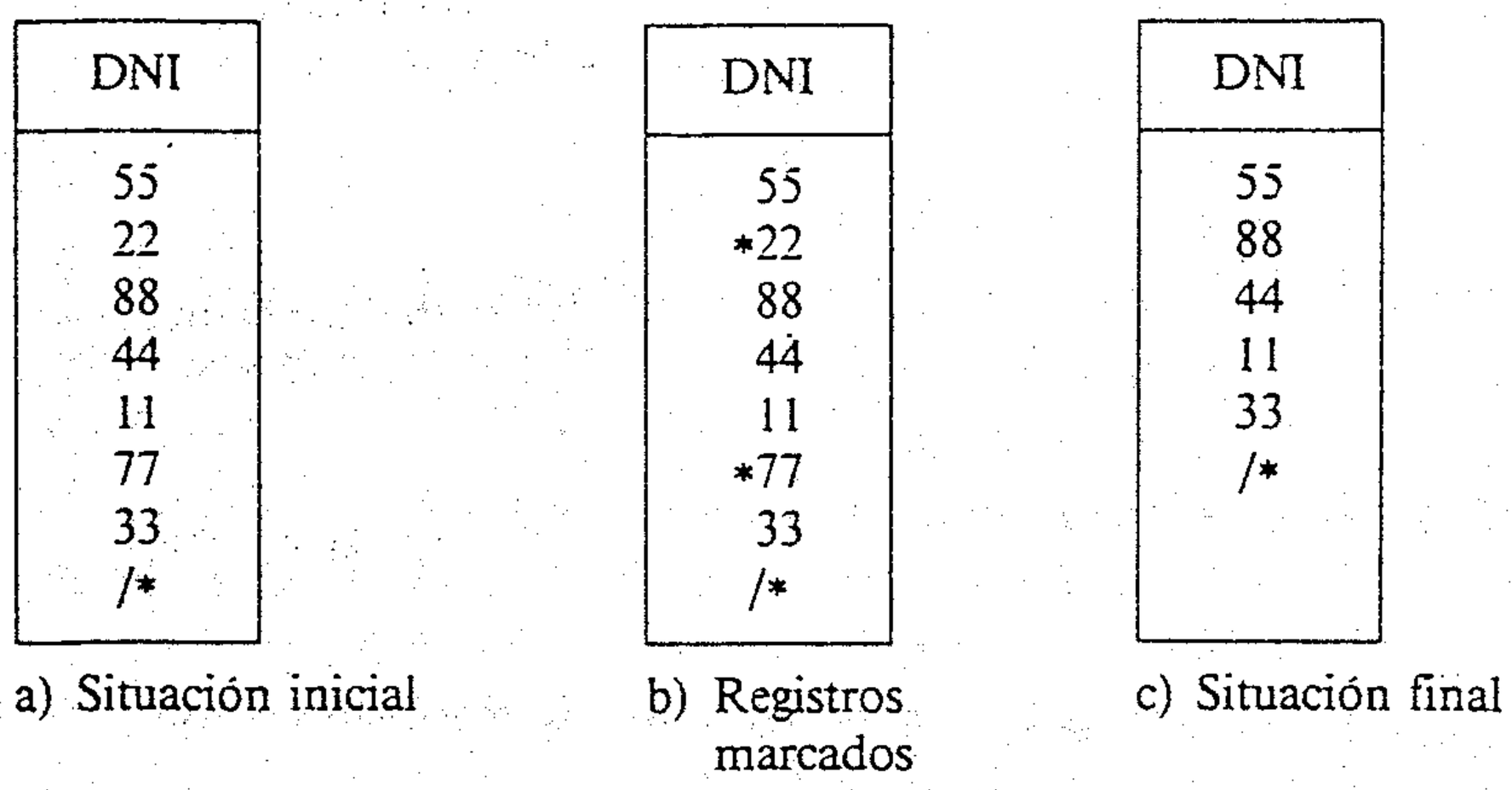


Figura 13.2.

A su vez, estos lenguajes poseen una instrucción para saber si el registro está marcado o no. Lo expresamos:

if está marcado

Esta pregunta devuelve un valor lógico: verdad o falso.

La operación de marcar el registro que deseamos borrar la indicamos:

marcado para ser borrado

Para empaquetar, el fichero debe estar abierto; lo expresamos:

empaquetar

Este proceso de empaquetar se realizará cuando no deseemos dar de baja a ningún registro más, por necesitar un tiempo más o menos largo, en función de la cantidad de registros que contenga el fichero.

También permiten posicionarse en el primer registro del fichero sin tener que cerrar y abrir fichero, lo expresamos:

ir al primer registro

De la misma forma, podemos posicionarnos en el último registro; lo escribimos:

ir al último registro.

VARIABLES

num = introducir el dni a dar de baja.
sw = controla si el DNI a dar de baja está o no en el fichero.
sw1 = empaquetar en el caso de producirse alguna baja.
res = dar más bajas.

PSEUDOCODIGO

```
res = "S"
sw1 = 0
abrir DATOS
hacer mientras res = "S"
     ir al primer registro
     sw = 0
     borrar pantalla
     en 5,10 imprime "D.N.I.:"
     en 5,18 introduce num
     hacer mientras no sea EOF()  AND  sw = 0
          if DNI = num
               if esta marcado
                    en 10,20 imprime "Ya ha sido dado de baja"
                    en 16,10 imprime "Pulse RETURN para continuar"
                    detener-listado
               else
                    en 10,12 imprime "Nombre:",nombre
                    en 12,28 imprime "Apellidos:",apellidos
                    en 14,55 imprime "Dirección:",direccion
                    en 16,69 imprime "Provincia:",provincia
                    en 20,20 imprime "Deseas borrarlo (S/N)"
                    res = espacios(1)
                    hacer mientras res <>"S" AND  res <>"N"
                         res = espacios(1)
                         en 20,43 introduce res
                    fin del hacer
                    if res = "S"
                         marcado para ser borrado
                         sw1 = 1
                    fin del if
               fin del if
               sw = 1
          else
               leer siguiente registro
          fin del if
     fin del hacer
     if sw = 0
          en 12,10 imprime "Baja inexistente"
```

```
            en 16,10 imprime "Pulse RETURN para continuar"
            detener-listado
        fin del if
        res = espacio(1)
        hacer MAS
    fin del hacer
    if sw1 = 1
        empaquetar
    fin del if
    cerrar DATOS
    fin del programa.
```

COMPROBACION -

Al podernos posicionar en el primer registro sin necesidad de tener que cerrar y abrir ficheros, estas dos operaciones irán ahora fuera del bucle.

Introducimos el DNI a dar de alta y buscamos en el fichero hasta el final o lo encontremos, *hacer mientras no sea EOF() AND sw=0.*

La instrucción de *ir al primer registro* es por si deseamos dar más altas y situar el puntero al principio del fichero para inicializar la búsqueda.

Comparamos el valor de la variable con el campo. Puede suceder:

— Son iguales: a su vez, caben las posibilidades:
  a) Está marcado, *if está marcado,* es como si estuviera borrado, aunque físicamente continúe en el fichero.
  b) No está marcado, visualizamos el registro y preguntamos si deseamos darle de baja: Sí: lo marcamos, *sw1=1.*

Tanto si lo deseamos borrar como si no, para que no continúe leyendo asignamos *sw=1.*

Al salir del bucle, si el *sw* tiene el valor cero nos está indicando que el registro no está en el fichero; baja inexistente.

Cuando no deseemos dar más bajas, preguntamos por el valor del *sw1.* Si tiene el valor uno, indica que hay registros marcados y debemos compactar el fichero de datos.

Fuera del ciclo cerramos el fichero.

Ejercicio 3

Dado el fichero secuencial DATOS.DAT, realizar un programa que nos permita realizar modificaciones cuantas veces deseemos.

ESTUDIO PREVIO

Primero buscaremos el registro a modificar introduciendo por teclado el valor del DNI. Si dicho valor no se encuentra en el fichero, imprimiremos el mensaje "Registro inexistente".

Si se encuentra, visualizaremos en pantalla los campos del registro e introduciremos los nuevos valores deseados.

VARIABLES

num = dni del registro a modificar.
sw = controla si el valor *num* está o no en el fichero.
nom = nuevo valor del nombre.
ape = nuevo valor de los apellidos.
dir = nueva dirección.
pro = nueva provincia.
res = modificar más registros.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     abrir DATOS
     sw = 0
     num = 0
     nom = espacios(15)
     ape = espacios(30)
     dir = espacios(20)
     pro = espacios(20)
     borrar pantalla
     en 5,10 imprime "D.N.I.:"
     en 5,18 introduce num
     hacer mientras no sea EOF()  AND  sw = 0
          if DNI = num
               hacer IMPRIMIR
               hacer CAMBIOS
               sw = 1
          else
               leer siguiente registro
          fin del if
     fin del hacer
     if sw = 0
          hacer DETENER
     fin del if
     cerrar DATOS
     res = espacio(1)
     hacer MAS
fin del hacer
fin del programa.

IMPRIMIR.
en 7,5 imprime "Nombre:"
```

```
en 9,5 imprime "Apellidos:"
en 11,5 imprime "Dirección:"
en 13,5 imprime "Provincia:"
en 7,16 imprime nombre
en 9,16 imprime apellidos
en 11,16 imprime direccion
en 13,16 imprime provincia.

CAMBIOS.
nom = nombre
ape = apellidos
dir = direccion
pro = provincia
en 7,16 introduce nom
en 9,16 introduce ape
en 11,16 introduce dir
en 13,16 introduce pro
nombre = nom
apellidos = ape
direccion = dir
provincia = pro
regrabar registro.

DETENER.
en 10,20 imprime "Registro inexistente"
en 20,18 imprime "Pulse RETURN para continuar"
detener-listado.
```

COMPROBACION

Una vez introducido el valor del DNI a modificar, leemos el fichero desde el principio hasta el final o hasta que lo encontremos.

A medida que vamos leyendo comparamos el valor del campo con el DNI introducido por teclado.

Si son iguales, realiza los subprogramas IMPRIMIR y CAMBIOS, y ponemos el *sw* a uno, *sw=1*, para que abandone la búsqueda.

En IMPRIMIR, simplemente visualizamos el registro por pantalla.

En CAMBIOS, realizamos las modificaciones. Primero volcamos los valores de los campos en las variables, en el caso de que no deseemos modificar algún campo y al pulsar RETURN no perdamos el valor anterior.

Introducimos los nuevos valores en los campos deseados y los volcamos sobre los campos para regrabarlos.

Cuando salga del bucle preguntamos por el estado del *sw*. Si tiene el valor cero, quiere decir que el registro buscado no está.

Ejercicio 4

Se desea actualizar el fichero DATOS.DAT mediante el fichero secuencial MODIFICA.DAT, cuyos campos son:

DNI2	NOMBRE2	APELLI2	DIRE2	PROVIN2	TO

donde TO indica el tipo de operación a realizar:

— A: alta.
— B: baja.
— M: modificación.

Suponer que ambos ficheros están ordenados por el campo que contiene el DNI.

ESTUDIO PREVIO

Aparte de los dos ficheros, DATOS y MODIFICA, necesitamos un fichero auxiliar, AUX, cuya estructura sea igual a la del archivo DATOS, donde iremos grabando los registros actualizados. Este proceso se denomina **enfrentamiento de ficheros.**

El fichero de DATOS es el maestro y el de MODIFICA es el de modificaciones.

Supongamos que tenemos una situación como la siguiente:

Ficheros:	DATOS	MODIFICA		AUX
Campos:	DNI	DNI2	TO	DNI
Registros:	2	1	A	1
	4	2	M	2
	8	4	B	5
	10	5	A	8
	/*	8	A	10
		11	A	11
		/*		/*

Empezamos a leer del fichero MODIFICA hasta que sea su final. Comparamos ese registro con el de DATOS:

- Son iguales: leemos el tipo de operación; puede ser:
  — A: alta duplicada; se graba el registro de DATOS en AUX.
  — B: no se graba el registro en AUX.
  — M: se graba el registro de MODIFICA en AUX.
  En los tres casos, leemos de ambos ficheros: MODIFICA y DATOS.
- No son iguales; puede suceder que sea más pequeño el DNI del fichero:
  — MODIFICA y el tipo de operación es A: lo grabamos en AUX, leemos solamente del archivo MODIFICA.

— DATOS: lo grabamos en AUX, leemos del fichero de DATOS.

Si se termina primero el fichero de MODIFICA, pasaremos los registros que quedan de DATOS a AUX.

En caso de terminar primero el fichero de DATOS, seguiremos leyendo del archivo de MODIFICA y grabaremos en AUX solamente aquellos registros cuyo tipo de operación sea A.

PSEUDOCODIGO

```
abrir DATOS
abrir MODIFICA
abrir AUX
hacer mientras no sea EOF() de MODIFICA
     if dni = dni2
          if to = "A"
               posicionarse al final de AUX
               grabar registro de DATOS en AUX
          else
               if to = "M"
                    posicionarse al final de AUX
                    grabar registro de MODIFICA en AUX
               fin del if
          fin del if
          leer siguiente registro de DATOS
          leer siguiente registro de MODIFICA
     else
          if dni < dni2
               posicionarse al final de AUX
               grabar registro de DATOS en AUX
               leer siguiente registro de DATOS
          else
               if to = "A"
                    posicionarse al final de AUX
                    grabar registro de MODIFICA en AUX
                    leer siguiente registro de MODIFICA
               fin del if
          fin del if
     fin del if
fin del hacer
if NOT EOF() de DATOS
     hacer mientras no sea EOF() de DATOS
          posicionarse al final de AUX
          grabar registro de DATOS en AUX
          leer siguiente registro de DATOS
     fin del hacer
fin del if
```

```
cerrar todos los ficheros
borrar fichero DATOS
renombrar fichero AUX como DATOS
borrar fichero MODIFICA
fin del programa.
```

COMPROBACION

Una vez abiertos todos los ficheros, controlamos el final del archivo de movimientos, MODIFICA, *hacer mientras no sea EOF() de MODIFICA.*

Comparamos los campos indicativos del fichero maestro, DATOS, con el de movimientos, MODIFICA, *if dni=dni2.* Si el tipo de operación:

- Es una alta, *if to="A"*, al ser alta duplicada, grabamos en el archivo AUX el contenido del registro de DATOS.
- Es una modificación, *M*, grabamos el registro de MODIFICA en AUX.
- No es ninguna de las dos cosas anteriores, será una baja, *B;* no hace falta preguntarlo, por estar controladas la *A* y la *B*. No se graba el registro en AUX.
  Leemos de ambos ficheros.

En el caso de no ser iguales los campos indicativos preguntamos cuál de ellos es el menor: *if dni < dni2.* Si el más pequeño del DNI correspondiente al fichero:

- Maestro, grabamos el registro en AUX y leemos sólo de él.
- De movimientos y si el tipo de operación es un alta, lo grabamos en AUX y leemos sólo del de movimientos.

Cuando termine el archivo de movimientos, puede ser que no haya terminado aún el maestro. Podemos tener la situación:

Ficheros	DATOS	MODIFICA		AUX
Campos	DNI	DNI2	TO	DNI
Registros	2	1	A	1
	4	2	M	2
	8	4	B	5
	10	5	A	8
	12	/*		10
	14			12
	16			14
	/*			16
				/*

Lo controlamos preguntando si aún no ha terminado de leer el maestro, *if NOT EOF() de DATOS,* en cuyo caso lo leeremos hasta que sea su final y pasamos todos los registros al fichero AUX.

Cuando se haya terminado, cerramos todos los ficheros, borramos el fichero maestro y el de las modificaciones y renombramos al AUX como DATOS.

RESUMEN

Un **fichero** o **archivo** es un conjunto de registros lógicos. Se nombran de la forma:

Nombre–del–fichero.Extensión.

Un **registro lógico** o **registro** es cada uno de los componentes del fichero. Está formado por un conjunto de campos.

Un **campo** es un conjunto de caracteres que expresan una información. Un campo puede estar compuesto por **subcampos.**

Los campos pueden ser heterogéneos: numéricos y alfanuméricos.

La marca de final de fichero la indicamos de la forma /*.

Los ficheros deben poseer un **campo clave** o **identificativo** mediante el cual vamos a acceder a toda la información contenida en el registro.

El valor de este campo clave será único en todo el fichero: no se puede repetir.

Cada registro viene identificado por su campo clave.

Cada campo clave identifica a un único registro.

Para acceder a un registro, generalmente se realiza a través de su campo identificativo.

También se puede acceder a un registro por cualquier otro campo del mismo. Pero puede haber más de un registro con varios campos que contengan igual información.

Las **características** de los ficheros son:

— Independencia de la información respecto a los programas.
— Su información puede ser tratada por diversos programas.
— Gran capacidad de almacenamiento.
— Duración en el tiempo.

Los ficheros, según su **uso**, se clasifican en:

— Ficheros permanentes. Se subdivide en tres grupos:
  Ficheros de constantes.
  Ficheros de situación o maestros.
  Ficheros históricos.
— Ficheros de movimiento o de transacciones.
— Ficheros de maniobra o de trabajo.

Las **operaciones** que puede soportar un fichero son:

— Creación.
— Consulta.
— Actualización.
— Clasificación.
— Fusión o mezcla.
— Partición.
— Borrado.

Los ficheros, según su **organización**, se clasifican en:

— Secuencial.
— Directa o aleatoria.
— Secuencial indexada.

Los ficheros con **organización secuencial** se caracterizan porque:

— Los registros se escriben en orden consecutivo.
— Los registros se leen empezando por el primero.
— Entre dos registros consecutivos no hay huecos y no se puede insertar uno nuevo.
— Sólo se pueden añadir registros después del último.

El ordenador utiliza una información interna: un **puntero** para saber en qué registro está posicionado, activo, en cada instante.

Para abrir ficheros: *abrir nombre–del–fichero*, el puntero se sitúa en el primer registro.

Para posicionar el puntero en el siguiente registro, leer:

leer siguiente registro

Una vez que el ordenador lee la marca del final del fichero, no se le puede indicar que lea un nuevo registro.

Para abrir un hueco entre el último registro y la marca de final de fichero para acoger al nuevo registro:

posicionarse al final del fichero

Escribir o grabar un nuevo registro en el fichero:

escribir registro

Para modificar información en un registro existente:

regrabar registro

Todas estas operaciones deben realizarse con el fichero abierto.

Para cerrar ficheros: *cerrar nombre–del–fichero*

Hay lenguajes que permiten dar bajas a registros de un fichero secuencial sin necesidad de utilizar un fichero auxiliar, de la siguiente forma:

— Marcar un registro: *marcado para ser borrado.*
— Eliminar el registro físicamente del fichero: *empaquetar.*

Para saber si el registro ya ha sido marcado lo indicamos: *if está marcado*, devuelve un valor lógico.

La operación de posicionar el puntero en el primer registro: *ir al primer registro* y llevarlo al último: *ir al último registro.*

EJERCICIOS PROPUESTOS

1. Hacer un programa que nos permita dar de baja por teclado al registro deseado del fichero DATOS.DAT, de igual estructura al de los ejercicios anteriores.

   Los registros que se den de baja se almacenan en un fichero histórico de bajas: BAJAS.DAT, también secuencial, con igual estructura que el fichero de DATOS.

2. Hacer un programa que nos permita modificar el fichero secuencial DATOS.DAT mediante el fichero secuencial MODIFICA.DAT, de la misma estructura que en el Ejercicio 4, resuelto en este capítulo, suponiendo que ambos ficheros están desordenados.

   Nota: cada vez que se lee del fichero DATOS.DAT hay que situar el puntero en el primer registro para realizar la búsqueda.

3. En el fichero secuencial CONCEPTOS.DAT cuyos campos son:

NUMERO	ARTICULO	PROVEEDOR

se guardan los artículos de un almacén. El número que se les asigna son consecutivos y en el mismo orden en que llegan los nuevos productos al almacén.

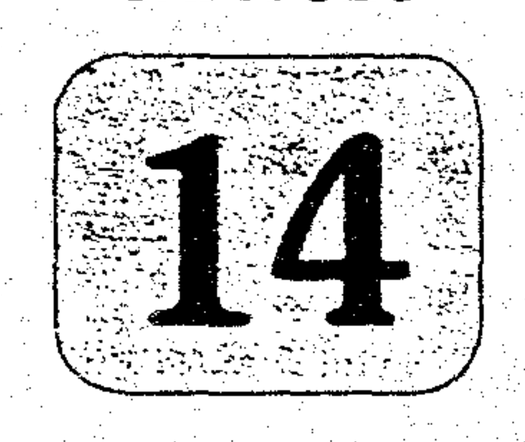
Para ello se dispone de un fichero secuencial NUMERO.DAT que sólo tiene un registro con el número correspondiente al nuevo artículo que se va a dar de alta en CONCEPTOS. El número está almacenado en el campo VAN.

Hacer un programa que nos permita dar altas de nuevos artículos y sólo tengamos que introducir por teclado los nombres del artículo y del proveedor.

4. Actualizar el fichero secuencial DATOS.DAT mediante el fichero de movimientos MODIFICA.DAT, ambos con la misma estructura que en el Ejercicio 4, resuelto en este capítulo.

Las actualizaciones no realizadas: altas duplicadas, bajas y modificaciones inexistentes, se almacenan en un fichero secuencial INCIDENCIAS.DAT, de la misma estructura que MODIFICA, para un posterior listado de las mismas.

CAPITULO



Técnicas de clasificación de ficheros

14.1. INTRODUCCION

Es frecuente que antes de utilizar un fichero se ordenen sus registros.

La operación de ordenación es una de las más frecuentes en cualquier proceso de datos: se le suele llamar clasificación.

La **clasificación** consiste en poner los registros de un fichero en un orden determinado siguiendo una secuencia ascendente o bien descendente.

Cuando ordenábamos arrays la operación se efectuaba internamente; en todo momento el array estaba en la memoria del ordenador y sólo había que mover los datos de unas posiciones a otras: los arrays se ordenan internamente.

Para clasificar ficheros, al estar almacenados en soportes magnéticos y ser la cantidad de información muy voluminosa, se descarta una clasificación interna que implicaría el uso de gran cantidad de memoria principal, la cual no siempre estará disponible.

La técnica a utilizar es la de introducir en memoria el número de registros que, de acuerdo con el tamaño de la memoria disponible en el ordenador y la longitud del registro, resulte más aconsejable.

Estos registros se graban en memorias auxiliares, en subficheros (del fichero a ordenar) que se van ordenando. Posteriormente se fusionan, obteniendo de esta forma el archivo original ordenado.

Dependiendo del lugar donde se realice esta clasificación, la denominamos:

— **Internas:** se lleva a cabo en la memoria central del ordenador.
— **Externas:** se lleva a cabo sobre soportes externos, fuera de la memoria central del ordenador.

Un concepto importante en la clasificación es la subordinación de unos campos a otros. Puede interesar que por un/os campo/s esté ordenado de forma ascendente y por otro/s de forma descendente.

Por ejemplo: que el fichero de facturas esté ordenado de una forma creciente por el número de cliente y dentro de cada cliente de una forma descendente por fecha.

En el fichero de datos personales la clave principal o primer nivel de ordenación podría ser el primer apellido, como segundo nivel el segundo apellido y como tercer nivel el nombre.

Nivel 1	Nivel 2	Nivel 3
Sánchez	Alvarez	Antonio
Sánchez	Alvarez	José Luis
Sánchez	Arranz	Luisa

14.2. CLASIFICACION INTERNA

En la **clasificación interna** la totalidad de la información a ordenar se encuentra en la memoria central del ordenador, moviendo la información de unas posiciones de la memoria a otras.

Esta clasificación presenta la ventaja de su rapidez, ya que no tiene que acceder a memorias auxiliares, si bien presenta el inconveniente de que si el volumen de datos es muy grande, no quepan estos en la memoria.

Hay tantos métodos de clasificación interna como métodos hay para la ordenación de los arrays: Burbuja, Shell, Mínimo...

Ejercicio 1

Hacer un programa que nos permita ordenar el fichero secuencial ARTICULOS.DAT por el campo ARTICULO.

ESTUDIO PREVIO

Los datos de un fichero se hallan situados fuera de la memoria, en soportes externos como los discos flexibles.

Al utilizar la clasificación interna, los datos deben estar presentes en la memoria del ordenador. De alguna forma tenemos que pasar todos los registros a su memoria, volcándolos sobre una matriz.

La matriz debe poseer tantas columnas como campos tenga el archivo. Siempre conocemos cuántos campos hay, bien porque hemos creado nosotros mismos el fichero o bien porque nos lo tienen que decir.

Al ordenar la matriz hay que intercambiar todos los campos de un registro para que el fichero siga conservando su identidad.

Una vez ordenado el array, lo volvemos a volcar sobre el fichero. El fichero queda ordenado.

Se presentan varias particularidades a tener en cuenta:

- El número de registros del fichero es variable, podemos añadir o borrar registros, por tanto la dimensión de la tabla debe ser variable.

¿Cómo podemos saber el número de registros que tiene el fichero en un momento determinado?

Hay lenguajes que poseen una instrucción que devuelve cuantos registros hay en ese instante en el fichero; lo designamos

num = número registros del fichero nombre–fichero

siendo *num* una variable numérica.

Otros lenguajes no poseen esta instrucción; en estos casos, no queda más remedio que leer el fichero desde el principio hasta el final y contar el número de registros que tiene.

- Los campos de los ficheros pueden ser numéricos y de tipo texto o alfanuméricos. Por el contrario, las matrices son homogéneas: todos sus datos son numéricos o alfanuméricos.

Supongamos que nuestro fichero ARTICULOS posee los siguientes campos y los siguientes registros:

NART	ARTICULO	PVP	STOCK	STOCKMIN
5	Melones	200	100	15
8	Melocotones	200	230	11
1	Tomates	100	112	10
11	Fresas	100	75	4
6	Sandías	150	111	25
2	Pimientos	175	54	15
9	Manzanas	100	123	17

Al hacer el volcado del fichero a la matriz hay que convertir los **campos numéricos a alfanuméricos.**

Sintaxis:

str(variable–numerica).

Posteriormente, al volcar la matriz sobre el fichero se volverá a convertir las **variables numéricas en alfanuméricas,** para que el fichero conserve la misma estructura que tenía en un principio. Lo indicamos de la forma:

val(cadena–alfanumerica).

Una vez volcado el fichero sobre la matriz, se ordena por la columna o columnas deseadas, de la misma forma que se vio en la ordenación de los arrays bidimensionales.

Al pasar los elementos de la matriz ordenada sobre el fichero, este queda de la forma:

NART	ARTICULO	PVP	STOCK	STOCKMIN
1	Tomates	100	112	10
2	Pimientos	175	54	15
5	Melones	200	100	15
6	Sandías	150	111	25
8	Melocotones	200	230	11
9	Manzanas	100	123	17
11	Fresas	100	75	4

VARIABLES

aux = auxiliar para guardar el número de registros del fichero.
num = índice para recorrer la tabla desde 1 hasta *aux*.
L = nombre de la tabla.

PSEUDOCODIGO

```
abrir ARTICULOS
num = numero de registros del fichero ARTICULOS
* num = 0
* hacer mientras no sea EOF()
*     num = num + 1
*     leer el siguiente registro
* fin del hacer
aux = num
hacer TABLA
hacer ORDENAR
hacer FICHERO
fin del programa.
```

Nota: las filas señaladas con el * indican que si nuestro lenguaje de programación no nos permite conocer directamente el número de registros que posee el fichero, debemos ejecutar dichas líneas.

Los asteriscos son comentarios.

Los **comentarios** son líneas del programa que no se ejecutan, pero que nos aclaran lo que realiza.

```
TABLA
posicionarse al principio del fichero
dimensionar L(num,5)
num = 1
hacer mientras no sea EOF()
     L(num,1) = str(nart)
     L(num,2) = articulo
     L(num,3) = str(pvp)
```

```
    L(num,4) = str(stock)
    L(num,5) = str(minimo)
    num = num + 1
    leer siguiente registro
fin del hacer

FICHERO.
posicionarse al principio del fichero
hacer mientras num <= aux
    nart = val(L(num,1))
    articulo = L(num,2)
    pvp = val(L(num,3))
    stock = val(L(num,4))
    minimo = val(L(num,5))
    posicionarse al final de ARTICULOS
    grabar registro en ARTICULOS
    num = num + 1
fin del hacer
```

COMPROBACION

En *num* recogemos el número de registros de ARTICULOS directamente si nuestro lenguaje lo permite o bien recorremos el fichero desde el primer registro hasta el final y calculamos el valor de *num*. Salvamos el valor de num en *aux*.

El subprograma CARGAR, a medida que recorre el fichero desde el primer registro, va grabando los registros en la tabla. Primero convierte los campos numéricos a variables alfanuméricas.

En ORDENAR se ordenaría la matriz (véase ordenación de un array bidimensional).

Por último, en FICHERO, recorremos la lista mediante la variable *num* desde la primera posición hasta la de *aux*.

Para conservar el número de registros del fichero se utiliza el auxiliar, ya que para recorrer una matriz no necesariamente los índices deben llamarse *i* o *j*.

14.3. CLASIFICACION EXTERNA

La clasificación externa se lleva a cabo sobre ficheros, fuera de la memoria central del ordenador.

Presenta la desventaja, respecto a la clasificación interna, de ser más lenta, ya que la velocidad de lectura sobre un soporte externo es siempre menor que cuando la lectura se realiza sobre la propia memoria central del ordenador.

Pero tiene la ventaja de no presentar ningún límite en cuanto al número de registros, excepto, claro está, el de la capacidad de nuestro soporte de datos.

14.4. METODO DE LAS INTERCALACIONES SUCESIVAS

Se le conoce también con el nombre de merge o de mezclas.

ESTUDIO PREVIO

Supongamos que tenemos el fichero secuencial *F* con los siguientes registros para el campo por el cual deseamos ordenarlo:

DATOS iniciales
F
8 6 15 4 3 12 7 1 9 5 10

Las fases a seguir para realizar este método son las siguientes:

- **Fase de distribución:** se desdobla la información contenida en el fichero original, F, en otros dos ficheros, F1 y F2, también llamados subficheros.

  Para ello, grabamos alternativamente los registro de F en F1 y en F2. Nos ayudamos de un *sw*. A medida que vamos distribuyendo los registros contamos cuantos hay.

Fase de distribución	
F1	F2
8	6
15	4
3	12
7	1
9	5
10	

- **Primera fase:** una vez obtenidos los subficheros F1 y F2, se procede a leer un registro de cada uno de ellos. Esos dos registros se graban ordenados, según el criterio deseado, en otros dos subficheros de salida, de escritura, F3 y F4. Una vez en uno de ellos, F3, y la siguiente vez en otro, F4.

  Este segundo paso se repite hasta terminar con los registros de los subficheros F1 y F2.
- **Segunda fase:** los subficheros de escritura del paso anterior, F3 y F4, se toman ahora como de lectura y los de lectura de la fase anterior, F1 y F2, se toman como de escritura.

  De cada subfichero F3 y F4 se leen dos registros, ordenándose entre sí. Esos dos registros ordenados se graban una vez en F1 y la siguiente vez en F2.

  El proceso continúa hasta que se hayan leído todos los registros de F3 y F4.

  Tanto en F1 como en F2 tenemos ordenados entre sí cuatro registros de cada uno de los ficheros.

  El número de registros a ordenar es el doble que el de la fase anterior.
- **Tercera fase:** se vuelven a intercambiar los ficheros de lectura y de escritura. F3 y F4 pasan a ser de escritura, F1 y F2 de lectura.

  El número de registros a ordenar en esta nueva secuencia es el doble del apartado anterior, grabándolos de una forma ordenada una vez en un fichero de escritura, F3, y luego en el otro, F4, hasta ordenar todos los registros de los subficheros de lectura.

  El proceso es igual que en la fase segunda; solamente hay que intercambiar la función de los subficheros, lectura por escritura, y viceversa. La secuencia de registros a ordenar es el doble de la fase anterior.

  Este paso se estará repitiendo hasta que los subficheros F1 y F2, o bien F3 y F4, estén ordenados.

FASE 1	
F3	F4
6 8	4 15
3 12	1 7
5 9	10

FASE 2	
F1	F2
4 6 8 15	1 3 7 12
5 9 10	

FASE 3	
F3	F4
1 3 4 6 7 8 12 15	5 9 10

- **Cuarta fase:** con los dos subficheros ordenados procedemos a su fusión, a su mezcla, obteniendo así el fichero F ordenado.

Operación de Mezclar
DATOS iniciales ordenados
F
1 3 4 5 6 7 8 9 10 12 15

En las distintas fases, excepto la de distribución y la última, tenemos que ir intercambiando los ficheros de lectura por los de escritura, y viceversa.

Para controlar en todo momento cuáles son los subficheros de lectura y cuáles son los de escritura, lo realizamos por medio del número de la fase.

Si la fase es:

— **Impar:** leemos de F1 y de F2. Grabamos la información en F3 y en F4.
— **Par:** leemos de F3 y F4. Grabamos en F1 y en F2.

Necesitamos una variable, por ejemplo *fase*, para saber cuáles son los subficheros de lectura y cuáles de escritura.

Al realizar la ordenación externamente vamos a tener la información del fichero original F por triplicado. Una la del propio fichero F, la segunda la de los subficheros F1 y F2, y por último la de F3 y F4.

Si el fichero F es muy grande, podemos tener problemas de capacidad en el disco. Para solucionarlo podemos borrar el fichero original una vez que hayamos distribuido sus registros en F1 y F2.

Antes de volcar los subficheros de nuevo sobre el fichero F, podemos borrar los subficheros que están desordenados.

Cuadro resumen de los ficheros a utilizar con sus campos claves correspondientes:

Ficheros	F	F1	F2	F3	F4
Campo clave	num	num1	num2	num1	num2

VARIABLES

n = número de registros del fichero *F*.
sw = interruptor para grabar una secuencia de registros en un subfichero o en otro.
fase = contador de las fases.
i = contador del número de registros que forman la secuencia.
impar = contador del número de registros a leer de los subficheros F1 o F3, va desde uno hasta *i*.
par = contador del número de registros a leer de los subficheros F2 o F4, va desde uno hasta *i*.

PSEUDOCODIGO

```
n = 0
sw = 0
hacer DISTRIBUIR
hacer ORDENAR
hacer FUSIONAR
fin del programa.

DISTRIBUIR.
abrir F (lectura)
abrir F1 y F2 (escritura)
hacer mientras no sea EOF()  (del fichero F)
     if sw = 0
          posicionarse al final del fichero F1
          grabar el registro de F en F1
          sw = 1
     else
          posicionarse al final del fichero F2
          grabar el registro de F en F2
          sw = 0
     fin del if
     n = n + 1
     leer siguiente registro de F
fin del hacer
cerrar todos los ficheros
borrar el fichero F

ORDENAR.
i = 1
fase = 1
hacer mientras i * 2 <= n
     sw = 0
     if fase <> int(fase/2)*2
          hacer LEER-DE-FI-Y-F2
     else
```

```
            hacer LEER-DE-F3-Y-F4
        fin del if
        i = i * 2
        fase = fase + 1
fin del hacer

LEER-DE-F1-Y-F2.
abrir F1 y F2 (lectura)
abrir F3 y F4 (escritura)
hacer mientras no sea EOF() de F1  AND no sea EOF() de F2
        impar = 0
        par = 0
        hacer mientras impar <> i  AND  par <> i
                if num1 > num2
                        if sw = 0
                                grabar el registro de F2 en F3
                        else
                                grabar el registro de F2 en F4
                        fin del if
                        par = par + 1
                        leer siguiente registro de F2
                else
                        if sw = 0
                                grabar el registro de F1 en F3
                        else
                                grabar el registro de F1 en F4
                        fin del if
                        impar = impar + 1
                        leer siguiente registro de F1
                fin del if
        fin del hacer
        if impar <> i
                hacer mientras impar <> i  AND  no sea EOF() de F1
                        if sw = 0
                                grabar el registro de F1 en F3
                        else
                                grabar el registro de F1 en F4
                        fin del if
                        impar = impar + 1
                        leer siguiente registro de F1
                fin del hacer
        else
                hacer mientras par <> i  AND  no sea EOF() de F2
                        if sw = 0
                                grabar el registro de F2 en F3
                        else
```

```
                grabar el registro de F2 en F4
            fin del if
            par = par + 1
            leer siguiente registro de F2
        fin del hacer
    fin del if
    hacer CAMBIAR-SW
fin del hacer
cerrar ficheros

LEER-DE-F2-Y-F3.
abrir F3 y F4 (lectura)
abrir F1 y F2 (escritura)
hacer mientras no sea EOF() de F3  AND no sea EOF() de F4
    impar = 0
    par = 0
    hacer mientras impar <>i  AND  par <> i
        if num1 > num2
            if sw = 0
                grabar el registro de F4 en F1
            else
                grabar el registro de F4 en F2
            fin del if
            par = par + 1
            leer siguiente registro de F4
        else
            if sw = 0
                grabar el registro de F3 en F1
            else
                grabar el registro de F3 en F2
            fin del if
            impar = impar + 1
            leer siguiente registro de F3
        fin del if
    fin del hacer
    if impar <> i
        hacer mientras impar <> i  AND  no sea EOF() de F3
            if sw = 0
                grabar el registro de F3 en F1
            else
                grabar el registro de F3 en F2
            fin del if
            impar = impar + 1
            leer siguiente registro de F3
        fin del hacer
    else
```

```
        hacer mientras par <> i  AND  no sea EOF() de F4
             if sw = 0
                  grabar el registro de F4 en F1
             else
                  grabar el registro de F4 en F2
             fin del if
             par = par + 1
             leer siguiente registro de F4
        fin del hacer
   fin del if
   hacer CAMBIAR-SW
fin del hacer
cerrar ficheros

FUSIONAR.
abrir F (escritura)
if fase = int(fase/2)*2
     borrar F3 y F4
     abrir F1 y F2 (lectura)
     hacer MEZCLAS
     cerrar ficheros
     borrar F1 y F2
else
     borrar F1 y F2
     abrir F3 y F4 (lectura)
     hacer MEZCLAS
     cerrar ficheros
     borrar F3 y F4
fin del if

CAMBIAR-SW.
if sw = 0
     sw = 1
else
     sw = 0
fin del if.
```

COMPROBACION

En primer lugar, el ordenador realiza el subprograma DISTRIBUIR, el cual va repartiendo los registros entre los ficheros F1 y F2, según sea el estado del *sw:*

- Cero: graba el registro en F1. Para que el siguiente registro lo grabe en F2 cambiamos el estado del sw a uno.
- Uno: graba el registro en F2. El siguiente lo debe grabar en F1. Vuelve a cambiar el estado del sw a cero.

Aumenta el contador de registros y accedemos al siguiente registro.

Una vez que todos los registros de F están distribuidos entre F1 y F2, tenemos la información por duplicado. La que nos interesa es la que se encuentra en los subficheros por lo cual borramos la información contenida en F. Previamente cerramos los ficheros, pues de lo contrario daría error.

También se cierran F1 y F2, pues en el siguiente paso tenemos que leer de ellos desde el primer registro, la forma de conseguirlo es cerrando los ficheros y volverlos a abrir de nuevo. Se podrían haber cerrado en el siguiente subprograma, pero es preferible abrirlos al principio y cerrarlos al final del subprograma.

Empezamos a ordenar los ficheros mediante el subprograma ORDENAR.

Fuera del ciclo se inicializan los contadores a uno, *i=1*, por ser la secuencia inicial de un registro, y *fase=1*, ya que empezamos a leer de F1 y F2.

El ciclo lo debe estar haciendo *mientras i*2<=n*, por ser *i* el número de registros de cada secuencia, y esta va siendo cada vez el doble de la anterior.

Al dividir los registros del fichero original en otros dos, en cada uno de ellos habrá como máximo la mitad (si el número de registros de F es par).

El *sw* va inicializado dentro del ciclo, porque cada vez que empezamos a leer de los ficheros debe empezar a colocar los registros de la secuencia en un subfichero determinado.

Si la frase es:

- Impar: al principio lo es, los registros están en F1 y en F2 y los debemos grabar ordenados en F3 y en F4; realiza el subprograma LEER–DE–F1–Y–F2.
- Par: al contrario de cuando es impar, ejecuta LEER–DE–F3–Y–F4.

Cuando salga de IF, se han terminado de ejecutar los subprogramas LEER, la secuencia pasa a valer el doble, *i=i*2*, y cambia la fase de par a impar o al revés, *fase=fase+1*.

El subprograma LEER–DE–F1–Y–F2, como su nombre indica, lee de F1 y F2, los graba ordenados en F3 y en F4.

Hay que leer de los ficheros hasta su final, pero en secuencias de registros cuyo valor es *i*, *hacer mientras impar<>i AND par<>i*. Los contadores van dentro del ciclo, porque la secuencia se tiene que estar repitiendo hasta el final del fichero.

Compara el valor del campo por el que estamos ordenando los ficheros F1 y F2. Si es mayor el de F1, el menor de ellos, el de F2, lo graba, según el estado del *sw*, en F3 o en F4.

Al ser menor el valor del campo de F2, debemos leer de este fichero y aumentar la variable, *par*, del número de registros leídos hasta completar la secuencia.

En caso de ser menor el del F1 hace la misma operación, pero grabando en el fichero correspondiente, F3 o F4, el registro de F1.

De un fichero, de F1 o de F2, podemos haber leído todos los registros que forman la secuencia, pero no del otro. Hay que leer, solamente, del que no ha llegado hasta su final, *if impar<>i*:

- Sí: hay que leer de F1 pero controlando su final, *hacer mientras impar<>i AND no sea EOF() de F1*; según como sea el estado del *sw* los grabará en F3 o en F4. Los graba de la misma forma que los lee, ya que los registros estarán ordenados por las secuencias anteriores.
- No: igual que antes, pero ahora se lee de F2.

Una vez ordenados los registros de esta secuencia, pasará a ordenar los registros siguien-

tes de los ficheros, siempre y cuando no hayamos llegado a su final, sin variar el valor de la secuencia. Ahora los registros se grabarán en el otro subfichero, F4 o F3; para ello cambiamos el estado del *sw*, ejecuta CAMBIAR–SW.

Cerramos ficheros. Hay que intercambiar el papel de los ficheros y empezar a leer desde el primer registro.

El subprograma LEER–DE–F3–Y–F4 es igual que el de LEER–DE–F1–F2 pero cambiando los ficheros de lectura por escritura, y viceversa.

Una vez que ambos subficheros están ordenados debemos unirlos; ejecuta FUSIONAR.

Para saber cuáles son los subficheros que están ordenados preguntamos por el valor de la fase, *if fase=int(fase/2)*2;* ahora es al contrario, porque antes de abandonar el ciclo se aumenta la *fase:* .

— Fase par: están ordenados F1 y F2.
— Fase impar: están ordenados F3 y F4.

En ambos casos borramos los que no están ordenados.

Abrimos el fichero *F* donde se van a grabar los registros clasificados de los subficheros ordenados; para ello hay que ejecutar el subprograma MEZCLAS; es idéntico al subprograma de igual nombre para unir dos arrays ordenados en uno.

Cerramos ficheros y borramos los subficheros ordenados.

Mejora:

Puede lograrse un ahorro considerable de tiempo si en lugar de empezar intercalando secuencias de un registro, se empieza intercalando un gran número de registros.

Para ello, en la fase de distribución, se leerá un número de registros lo mayor posible en la memoria central del ordenador y se procederá a clasificarlos en él antes de grabarlos sobre los subficheros. El ordenar esta secuencia de registros en memoria es una clasificación interna, que, como se ha indicado, es más rápida.

Fichero original	Distribución		Fase 1		Fase 2		Fichero original ordenado
F	F1	F2	F3	F4	F1	F2	F
45	12	9	9	1	1	5	1
12	33	14	12	10	9	8	5
33	45	25	14	11	10	16	8
9			25	22	11	24	9
14	10	1	33	28	12	38	10
25	28	11	45	44	14		11
10	44	22			22		12
28			5		25		14
44	8	5	8		28		16
22	16	38	16		33		22
11	24		24		44		24
1			38		45		25
8							28
16							33
24							38
38							44
5							45

El número de registros de las secuencias de ordenación siguen siendo el doble de registros de la secuencia anterior.

Para clasificar estas secuencias de registros hay muchas técnicas; entre ellas están:

- **Intercalación:** es igual a la que acabamos de estudiar.
- **Selección:** consiste en ver cuál de los registros que forman la secuencia tiene el indicativo menor o mayor y llevarlo a la primera posición. Recordar ordenación por el mínimo.
- **Cambio:** examina los registros por parejas: el primero con el segundo, el segundo con el tercero, el tercero con el cuarto... y así sucesivamente, colocando los registros comparados en el orden deseado, comprobando si se han efectuado algún cambio o no. Recordar ordenación de Burbuja-Switch.

RESUMEN

La **clasificación** de un fichero consiste en poner los registros en un orden determinado.

La **clasificación interna** se lleva a cabo en la memoria central del ordenador.

La **clasificación externa** se lleva a cabo fuera de la memoria del ordenador, en soportes externos.

Para convertir **variables numéricas a alfanuméricas:**

str(variable–numerica).

Para pasar de **variables alfanuméricas a numéricas:**

val(cadena–alfanumerica).

CUESTIONES

1. Ventajas de la clasificación interna.
2. Inconvenientes de la clasificación externa.
3. ¿Cualquier método de la ordenación de los arrays se puede aplicar en la clasificación interna? ¿Cuáles son esos métodos?
4. ¿Cuántos ficheros son necesarios en la clasificación interna? ¿y en la externa?

CAPITULO

Generación de informes

15.1. INTRODUCCION

La generación de informes es una de las aplicaciones que más se utilizan en la programación de los ordenadores. Es de general conocimiento el gran número de listados que debe manejar el ejecutivo empresarial si quiere disponer en todo momento de una información completa y, además, actualizada.

A la hora de confeccionar los programas generadores de informes, hay que tener en cuenta dos aspectos importantes:

- El formato del informe: se refiere a la forma en que deseamos imprimir la información, bien por pantalla o por impresora.
- La lógica del tratamiento: depende en mayor o menor medida del formato del informe.

Los conceptos fundamentales en los informes son:

- **Encabezamiento del reporte:** un título para el listado. Es aconsejable que vaya en una página separada.
- **Cabecera:** suele incluir el número de la página, el nombre de cada una de las columnas del listado, etc.

  La cabecera, en caso de existir, se escribe sólo una vez por página.
- **Línea de detalle:** contiene la información detallada, es decir, el valor de los registros y/o el resultado de alguna operación efectuada con algún campo del registro. Su número es variable, dependiendo de las condiciones especificadas y del número de registros del fichero.
- **Pie:** se escribe después de las líneas de detalle.

  En caso de existir sólo debe aparecer una vez por página.
- **Final del reporte:** será impreso al final del listado, que puede contener un resumen final de todo el reporte, así como un comentario indicando el final del mismo.

Ejercicio 1

Tenemos el fichero secuencial DATOS.DAT, cuyos campos son:

DNI	NOMBRE	APELLIDOS	DIRECCION	PROVINCIA

Hacer un programa que nos permita listar, por pantalla, todos los registros del fichero, controlando el salto de página cuando llegue a la línea veinte.

ESTUDIO PREVIO

Abriremos el fichero y leeremos desde el primer registro hasta el último.

A medida que vamos pasando por los registros imprimimos su contenido, visualizando el valor de los campos y controlando el salto de página.

Formato de impresión en pantalla:

DNI	NOMBRE	APELLIDOS	DIRECCION	PROVINCIA
-----	--------	----------	----------	----------
-----	--------	----------	----------	----------
-----	--------	----------	----------	----------
Pulse RETURN para continuar				

VARIABLE

fi = contador de las filas.

PSEUDOCODIGO

```
abrir DATOS
fi = 22
hacer mientras no sea EOF()
     if fi = 22
          hacer CABECERA
     fin del if
     en fi,2 imprime dni
     en fi,12 imprime nombre
     en fi,28 imprime apellidos
     en fi,55 imprime direccion
     en fi,69 imprime provincia
     fi = fi + 1
     if fi = 20
```

```
        en 22,20 imprime "Pulse RETURN para continuar"
        detener-listado
        fi = 22
    fin del if
    leer siguiente registro
fin del hacer
cerrar DATOS
fin del programa.

CABECERA.
borrar pantalla
en 3,4 imprime "D.N.I."
en 3,20 imprime "NOMBRE"
en 3,35 imprime "APELLIDOS"
en 3,60 imprime "DIRECCION"
en 3,70 imprime "PROVINCIA"
fi = 5.
```

COMPROBACION

Abrimos el fichero, con lo cual el puntero se sitúa en el primer registro del mismo. El proceso de lectura, así como el de imprimir en pantalla, es repetitivo: *hacer mientras no sea EOF()*.

Si no hay ningún registro en el fichero, no entra en el bucle y no imprime nada.

En el caso que haya registros, lo primero que debe hacer es imprimir la cabecera, para lo cual realiza el subprograma CABECERA, por estar inicializada *fi* a 22, *fi=22*.

La información contenida en los registros, la línea de detalle, empieza a imprimirla en la fila cinco, *fi=5*.

Al estar situado el puntero del fichero en el primer registro, visualizamos la información que contiene e imprimimos los campos del registro como si fueran variables.

Accedemos al siguiente registro, para lo cual le indicaremos al ordenador *leer siguiente registro*. Si hay algún registro más, el puntero se posiciona en el siguiente registro; en caso contrario, encuentra la marca de EOF y sale del bucle.

Cuando el ordenador termine de imprimir todos los registros hay que cerrar el fichero.

En COBOL la codificación sería:

```
    OPEN INPUT DATOS.
    MOVE 0 TO SW.
    READ DATOS AT END MOVE 1 TO SW.
    PERFORM LEER UNTIL SW = 1.

LEER.
    IF FI = 22
    . . . .
    . . . .
    . . . .
    READ DATOS AT END AND MOVE 1 TO SW.
```

donde LEER es el nombre de un procedimiento y el sw una variable que se declara a nivel 77.

Ejercicio 2

Tenemos el fichero secuencial DATOS.DAT, cuyos campos son los del ejercicio anterior. Crear un programa que nos permita consultar un registro.

ESTUDIO PREVIO

Para buscar un determinado registro se introduce por teclado, mediante una variable, el campo indicativo y se busca en el fichero, empezando en el primer registro hasta que lo encontremos. Si hemos llegado al final del archivo y no lo hemos hallado, indicaremos que ese dato no está.

Si lo encontramos, imprimimos su contenido y abandonamos la búsqueda. Es necesario utilizar un sw.

Formato de la pantalla en caso de encontrar el registro buscado:

```
DNI: --------
NOMBRE: ---------------
APELLIDOS: ------------------
DIRECCION: -----------
PROVINCIA: ---------
Más registros (S/N): -
```

VARIABLES

num = introducir el DNI a buscar.
sw = controla si hemos hallado el dato.
res = para buscar más datos.

PSEUDOCODIGO

```
res = "S"
hacer mientras res = "S"
     sw = 0
     num = 0
     borrar pantalla
     en 8,20 imprime "D.N.I. a buscar:"
     en 8,38 introduce num
     abrir DATOS
```

```
    hacer mientras no sea EOF()  AND  sw = 0
        if dni = num
            borrar pantalla
            en 8,20 imprime "DNI:".num
            en 10,20 imprime "NOMBRE:",nombre
            en 12,20 imprime "APELLIDOS:",apellidos
            en 14,20 imprime "DIRECCION:",dirección
            en 16,20 imprime "PROVINCIA:",provincia
            sw = 1
        else
            leer siguiente registro
        fin del if
    fin del hacer
    if sw = 0
        en 12,20 imprime "No está"
        en 16,20 imprime "Pulse RETURN para continuar"
        detener-listado
    fin del if
    cerrar DATOS
    res = espacios(1)
    hacer MAS
fin del hacer
fin del programa.
```

COMPROBACION

Dentro del ciclo, *hacer mientras res="S"*, abrimos el fichero, porque si deseamos buscar más de un registro, debemos situar el puntero en cada búsqueda al principio del archivo y esto solamente se consigue abriendo el fichero. Al final de ciclo hay que cerrarlo, pues si intentamos abrirlo de nuevo, y el archivo ya está abierto, da error.

Introducimos por teclado el valor del DNI a buscar y abrimos el fichero, con lo cual el puntero se coloca en el primer registro. Hay que leer hasta el final del archivo o hasta encontrar el dato, *hacer mientras no sea EOF() AND sw=0.*

Preguntamos si el DNI del registro actual es el buscado, *if dni=num:*

— Sí: imprimimos el resto de los datos. Para abandonar la búsqueda hacemos *sw=1.*
— No: debemos acceder al siguiente registro.

Cuando salgamos del ciclo puede ser por dos motivos: que hayamos encontrado el dato, *sw=1*, o bien que hayamos leído hasta el EOF y no lo hayamos encontrado, *sw=0.* Para saber por cuál de ellos preguntamos por su estado.

En ambos casos cerramos el fichero y hacemos MAS.

Ejercicio 3

Tenemos el fichero secuencial DATOS.DAT cuyos campos son los del ejercicio anterior. Listar por impresora todos los registros cuya provincia sea una determinada que introduciremos por teclado.

El formato de impresión será el siguiente:

Pag: --			
Relación de las personas que viven en la provincia: -------			
D.N.I.	Nombre	Apellidos	Dirección
-----	----------	----------	----------
-----	----------	----------	----------
-----	----------	----------	----------

Cambio de página cuando llegue a la línea 55.
Al final del listado imprimir el número de personas que habitan en esa provincia.

ESTUDIO PREVIO

La impresora funciona en muchos lenguajes como un fichero.

La visualización por impresora se realiza de la misma forma que por pantalla; previamente hay que activarla. Lo indicamos de la forma:

activar impresora

La mayoría de las veces la instrucción de activar pantalla irá antes de entrar en el bucle. para que el ordenador sólo la active una vez.

Una vez activada, todas las órdenes de imprimir se realizarán por la impresora.

Para producir el salto de página lo indicamos:

salto de pagina

el cabezal de la impresora se sitúa en la siguiente página y en la misma fila en la que había comenzado a efectuar el listado.

Una vez terminado el listado hay que desactivar la impresora y activar la pantalla:

activar pantalla.

Introduciremos por teclado la provincia. Leeremos el fichero desde el primer registro hasta el último y compararemos el campo *provincia* con el valor de la variable. Si son iguales, imprimimos el registro y aumentamos el contador para saber cuántas personas hay en la provincia.

Las cabeceras se realizan de la misma forma que hemos visto.

VARIABLES

pro = introducir la provincia a buscar.
c = contador de las personas que viven en esa provincia.
fi = contador de las filas.
pag = contador del número de páginas.

PSEUDOCODIGO

```
borrar pantalla
fi = 55
c = 0
pag = 1
pro = espacios(15)
en 10,20 imprime "Provincia:"
en 10,32 introduce pro
abrir DATOS
activar impresora
hacer mientras no sea EOF()
     if provincia = pro
          if fi = 55
               hacer CABECERA
          fin del if
          en fi,5 imprime dni
          en fi,15 imprime nombre
          en fi,35 imprime apellidos
          en fi,65 imprime direccion
          fi = fi + 1
          c = + 1
     fin del if
     leer siguiente registro
fin del hacer
if pag <> 1
    en fi+2,20 imprime "Total de personas:",c
fin del if
activar pantalla
cerrar DATOS
fin del programa.

CABECERA.
salto de pagina
en 2,65 imprime "Pag:",pag
en 4,10 imprime "Relación de las personas que viven en la provincia:",pro
en 6,7 imprime "D.N.I."
en 6,18 imprime "Nombre"
en 6,40 imprime "Apellidos"
en 6,68 imprime "Dirección"
```

```
en 7,4 imprime "_____________________________________________"
fi = 9
pag = pag + 1.
```

COMPROBACION

Antes del ciclo abrimos el fichero, activamos la impresora, se inicializan variables e introducimos la provincia a buscar.

Leemos desde el primer registro hasta el último.

A medida que leemos preguntamos si es la provincia buscada. En caso de serlo, realiza la CABECERA, imprime la línea de detalle y aumenta los contadores.

Coincida o no la provincia, leemos el siguiente registro.

La CABECERA sólo debe realizarla cuando encontremos el primer registro que cumpla la condición especificada, por lo cual va dentro de la pregunta *if provincia=pro*, y estar el contador de las filas inicializado a 55.

En este subprograma primero avanzamos página, aunque la primera vez deja una página en blanco.

Imprime la cabecera, asigna al contador de las filas, que debe comenzar a imprimir la línea de detalle, y aumenta el contador de las páginas.

Para saber si ha impreso algún registro o no, preguntamos por el número de página, *if pag< >1*. Si tiene el valor:

- Uno: no ha impreso nada, bien porque nos hemos equivocado al introducir el nombre de la provincia o porque realmente no hay ningún registro de esa provincia.
- Otro valor: como mínimo ha impreso una línea de detalle. Nos dice cuántas mediante el contador *c*.

Activamos la pantalla y cerramos el fichero.

Ejercicio 4

En el fichero secuencial, VENTAS.DAT, están almacenadas las ventas de los productos durante el día, cuyos campos son:

NART	VENTAS

Puede haber más de un registro para un mismo artículo, pero todos ellos están uno a continuación del otro.

Se desea hacer un programa que liste por impresora todas las ventas realizadas durante el día, de tal forma que cada producto sólo aparezca una vez, con el formato que se muestra en la página siguiente:

Pag: --	
LISTADO DE LAS VENTAS DE LOS PRODUCTOS AL DIA: ---	
Número	Cantidad
---	----
---	----
---	----

Cuando llegue a la línea 55 se producirá un salto de página.
Al final del listado que imprima el total de las unidades vendidas.

- ESTUDIO PREVIO

Podemos tener una situación como la siguiente en el fichero de VENTAS:

NART	VENTAS
1	10
2	20
2	15
5	50
8	10
8	20
8	30
10	5

y al imprimir deseamos que aparezca:

NART	VENTAS
1	10
2	35
5	50
8	60
10	5

Nos debemos de ayudar de un *sw* y de una variable auxiliar. El sw nos permite guardar en el auxiliar el primer artículo del fichero.

Comparamos el auxiliar con el número del artículo; si son iguales, estamos con el mismo artículo y vamos acumulando las ventas realizadas para ese artículo; en caso contrario, hay que cambiar el valor del auxiliar al nuevo valor del artículo.

Para imprimir el día se puede hacer de dos formas:

- Introduciéndolo por teclado mediante la utilización de una variable. Iría antes del ciclo para realizarlo sólo una vez.
- Tomándolo del sistema. No hace falta ninguna variable. Lo indicamos de la forma:

fecha del sistema.

VARIABLES

aux = guardar el NART.
sw = almacena el valor de NART en *aux* para el primer registro.
uno = acumulador de las ventas de un artículo.
total = acumulador de las ventas de todos los artículos.
pag = contador de las páginas.
fi = contador de las filas.

PSEUDOCODIGO

```
total = 0
uno = 0
fi = 55
sw = 0
aux = 0
pag = 1
activar impresora
abrir VENTAS
hacer mientras no sea EOF()
     if fi = 55
          hacer CABECERA
     fin del if
     if sw = 0
          aux = nart
          sw = 1
     fin del if
     if nart = aux
          uno = uno + ventas
     else
          hacer IMPRIMIR
          uno = 0
          aux = nart
          uno = ventas
     fin del if
     leer siguiente registro
fin del hacer
hacer IMPRIMIR
en fi+2,20 imprime "Unidades vendidas:",total
activar pantalla
```

```
cerrar VENTAS
fin del programa.

IMPRIMIR.
en fi,32 imprime aux
en fi,42 imprime total
fi = fi + 1
total = total + uno

CABECERA.
salto de pagina
en 2,65 imprime "Pag:",pag
en 4,20 imprime "LISTADO DE LAS VENTAS DE LOS PRODUCTOS AL DIA:"
en 4,68 imprime fecha del sistema
en 6,30 imprime "Número"
en 6,40 imprime "Cantidad"
en 7,18 imprime "_____________________________________"
fi = 9
pag = pag + 1.
```

COMPROBACION

Una vez abierto el fichero VENTAS, el puntero se coloca en el primer registro. El registro tenemos que guardarlo en el auxiliar. Solamente debe realizarlo una vez, para lo cual, una vez asignado su valor al auxiliar, *aux=nart*, ponemos el sw a uno, *sw=1*, para que por aquí no vuelva a pasar el programa.

Preguntamos si *nart=aux*:

- Lo son, por lo menos la primera vez. Acumulamos el valor de las ventas en *uno* y leemos el siguiente registro. Si se trata del mismo artículo, para lecturas posteriores de más registros continúa acumulando las ventas en *uno*.
- No son iguales: significa que estamos con un artículo distinto. Se han terminado todos los registros del artículo anterior y debemos imprimir sus datos, que están almacenados en *aux* y en *uno*. Sus ventas las acumulamos en total, *total=total+uno*, y se reinicializa a cero el acumulador de las ventas, *uno=0*.

Al estar en un artículo nuevo, puede haber más de una venta para este artículo. Asignamos al auxiliar el valor del nuevo artículo, *aux=nart*, y guardamos sus ventas en el contador; de lo contrario, no se acumularían y se perderían. Leemos el siguiente registro.

Cuando termine de leer todos los registros del fichero debemos imprimir los datos del último artículo, porque la impresión está dentro del ciclo y nosotros estamos fuera del bucle por haberse terminado el fichero. Ejecuta el subprograma IMPRIMIR e imprime las ventas de todos los artículos del fichero de VENTAS.

Ejercicio 5

Dado el fichero secuencial ARTICULOS.DAT, cuyos campos son:

NART	ARTICULO	PVP	STOCK	MINIMO

donde MINIMO es el stock mínimo de ese artículo.

En otro fichero secuencial, VENTAS.DAT, están almacenadas las modificaciones de los productos durante el día, cuyos campos son:

NART2	VENTAS	TIPO

el campo TIPO puede tomar los valores:

— Cero: es una venta.
— Uno: compra del producto.

Puede haber más de un registro para un mismo artículo.
Se desea hacer un programa que realice:

— Actualización del fichero de ARTICULOS.
— Un listado por impresora de las entradas y salidas de los artículos, de tal forma que cada producto sólo aparezca una vez, con el siguiente formato:

Pag: --		
LISTADO DE LAS ENTRADAS DE ARTICULOS AL DIA: ---		
Número	Artículo	Cantidad
--- ---	---------- ----------	---- ----

Pag: --				
LISTADO DE SALIDAS DE ARTICULOS AL DIA: ---				
Número	Artículo	Cantidad	PVP	Importe
--- * ---	---------- ----------	---- ----	-- --	---- ----
			Total:	-----

Si en algún artículo el stock es inferior al mínimo en el listado de ventas, delante del número de artículo se imprimirá un asterisco.

Cuando llegue a la línea 55 se producirá un salto de página.

Ambos ficheros están ordenados por número de artículo.

ESTUDIO PREVIO

Hay que tener en cuenta que de un mismo artículo puede haber entradas y salidas.

Supongamos que tenemos la siguiente situación:

VENTAS

NART2	VENTAS	TIPO
1	10	1
2	20	0
2	15	1
5	50	1
8	10	0
8	20	1
8	30	1
10	5	0

ARTICULOS

NART	ARTICULO	PVP	STOCK	MINIMO
1	Tomates	100	112	10
2	Pimientos	175	54	15
3	Zanahorias	80	450	20
5	Melones	200	100	15
6	Sandías	150	111	25
8	Melocotones	200	230	11
9	Manzanas	100	123	17
10	Peras	185	12	10
11	Fresas	100	75	4
12	Uvas	155	112	12

El listado de entradas de productos:

Número	Concepto	Cantidad
1	Tomates	10
2	Pimientos	15
5	Melones	50
8	Melocotones	50

El listado de salidas de productos:

Número	Concepto	Cantidad	PVP	Importe
2	Pimientos	20	175	3500
8	Melocotones	10	200	2000
* 10	Peras	5	185	925
			Total:	6425

El fichero de ARTICULOS actualizado queda de la siguiente forma:

NART	ARTICULO	PVP	STOCK	MINIMO
1	Tomates	100	122	10
2	Pimientos	175	49	15
3	Zanahorias	80	450	20
5	Melones	200	150	15
6	Sandías	150	111	25
8	Melocotones	200	270	11
9	Manzanas	100	123	17
10	Peras	185	7	10
11	Fresas	100	75	4
12	Uvas	155	112	12

Para controlar si de un mismo concepto hay más de un registro, utilizamos un *sw* y un auxiliar, de idéntica forma que el ejercicio anterior.

Al mismo tiempo que vamos imprimiendo podemos actualizar el STOCK del fichero de ARTICULOS.

Los listados no se pueden realizar los dos al mismo tiempo: primero haremos el de las entradas de productos y luego el de las salidas.

Para no tener que leer el fichero de VENTAS dos veces, que será muy grande, podemos realizar lo siguiente: a medida que leemos vamos actualizando el stock y sacamos el listado de las entradas de productos.

Las ventas de los artículos las podemos guardar en un nuevo fichero, SALIDAS, donde estarán almacenados el número del artículo y el de las ventas, en los campos NART3 y VENTAS3.

Una vez que se han impreso las entradas de productos en el almacén, leeríamos el archivo de SALIDAS y lo imprimiríamos tal como está. Cuando finalice la impresión se borra.

Con el número del artículo del fichero de VENTAS buscamos su nombre y precio en el archivo de ARTICULOS.

VARIABLES

SALIDAS = fichero con las salidas del almacén. Sus campos son NART3 y VENTAS3.
aux = guardar el NART.
sw = almacena el valor de NART en *aux*.
entra = acumulador de las entradas de un producto.
sale = acumulador de las salidas de un producto.
total = acumulador de las ventas de todos los artículos.
pag = contador de las páginas.
fi = contador de las filas.

PSEUDOCODIGO

```
entra = 0
sale = 0
total = 0
fi = 55
sw = 0
aux = 0
pag = 1
activar impresora
abrir ARTICULOS
abrir VENTAS
abrir SALIDAS
hacer mientras no sea EOF() de VENTAS
      if fi = 55
            hacer CABECOMPRAS
      fin del if
      if sw = 0
            aux = nart2
            hacer BUSCAR
            sw = 1
      fin del if
      if nart2 = aux
            hacer CALCULOS
      else
            hacer GRABAR
            hacer COMPRA        * imprimir los articulos que entran
            entra = 0
            sale = 0
            aux = nart2
            hacer BUSCAR
            hacer CALCULOS
      fin del if
      leer siguiente registro de VENTAS
fin del hacer
hacer GRABAR
```

```
hacer COMPRA
posicionarse en el primer registro de ARTICULOS
posicionarse en el primer registro de SALIDAS
fi = 55
hacer mientras no sea EOF() de SALIDAS
     if fi = 55
          hacer CABESAL
     fin del if
     hacer mientras nart3 <> nart
          leer siguiente registro de ARTICULOS
     fin del hacer
   * aux = nart3
   * hacer BUSCAR
     hacer SALE                    *imprimir los articulos vendidos
     leer siguiente registro del fichero SALIDAS
fin del hacer
en fi+4,55 imprime "Total:",total
activar pantalla
cerrar todos los ficheros
borrar el fichero SALIDAS
fin del programa.

CALCULOS.
if tipo = 0
     entra = entra + ventas
else
     sale = sale + ventas
fin del if

GRABAR.
stock = stock + entra - sale
regrabar registro en ARTICULOS
nart3 = aux
ventas3 = sale
posicionarse al final del fichero SALIDAS
grabar registro en SALIDAS

CABECOMPRAS.
salto de pagina
en 2,65 imprime "Pag:",pag
en 4,20 imprime "LISTADO DE LAS ENTRADAS DE ARTICULOS AL DIA:"
en 4,66 imprime fecha del sistema
en 6,18 imprime "Número"
en 6,35 imprime "Artículo"
en 6,65 imprime "Cantidad"
en 7,15 imprime "_____________________________________________"
```

```
fi = 9
pag = pag + 1

COMPRA.
en fi,16 imprime aux
en fi,30 imprime articulo
en fi,67 imprime entra
fi = fi + 1

CABESAL.
salto de pagina
en 2,65 imprime "Pag:",pag
en 4,20 imprime "LISTADO DE SALIDAS DE ARTICULOS AL DIA:"
en 4,60 imprime fecha del sistema
en 6,8 imprime "Número"
en 6,16 imprime "Artículo"
en 6,40 imprime "Cantidad"
en 6,54 imprime "PVP"
en 6,64 imprime "Importe"
en 7,6 imprime "____________________________________________"
fi = 9
pag = pag + 1

SALE.
salto de pagina
if stock < minimo
     en fi,4 imprime "*"
fin del if
en fi,6 imprime nart3
en fi,14 imprime articulo
en fi,40 imprime ventas
en fi,54 imprime pvp
en fi,65 imprime ventas*pvp
total = total + ventas*pvp
fi = fi + 1

BUSCAR.
hacer mientras nart <> aux
     leer siguiente registro de ARTICULOS
fin del hacer
```

COMPROBACION

Abiertos los ficheros, leemos de VENTAS hasta el final.

Imprime la cabecera de la entrada de artículos. Almacena en el auxiliar el valor del *nart2* del fichero de VENTAS y ejecuta el subprograma BUSCAR, que busca los datos de *nart2* en el archivo de ARTICULOS, *hacer mientras nart <> aux*.

No hace falta controlar el final del fichero ni cerrar y abrir de nuevo para colocar el puntero al principio del mismo, porque estamos suponiendo que los ficheros están ordenados por número de artículo.

Mientras no cambie el valor del *nart2*, y por tanto del auxiliar, no tiene por qué buscar los datos de ese artículo; por eso debe ir dentro del *sw*.

Siempre que el valor del *nart2* sea el mismo, el número de artículo se mantiene igual y ejecuta el subprograma CALCULOS. Dependiendo del valor de *tipo* acumula las ventas en la variable que recoge las entradas o en la de salidas. Continúa leyendo del fichero de VENTAS.

Cuando cambie el valor del *nart2*, significa que hemos cambiado de producto. Ejecuta el subprograma GRABAR: del artículo anterior. Actualizamos el stock, *stock=stock+entra–sale*, del fichero de ARTICULOS y regrabamos ese registro.

El auxiliar y las ventas del producto se graban en el fichero de SALIDAS para imprimirlas posteriormente.

Reinicializamos los contadores de entrada y salida de productos a cero.

Al igual que en el ejercicio anterior, se guarda el valor del nuevo artículo en el auxiliar. Al cambiar el producto hay que buscar sus características en el fichero de ARTICULOS, para lo cual se ejecuta de nuevo el subprograma BUSCAR. Por último, ejecuta CALCULOS para almacenar las entradas o salidas del nuevo producto en su variable respectiva, pues de lo contrario no guardaríamos esa modificación del stock.

Leemos el siguiente registro de VENTAS.

Cuando termine el fichero de VENTAS debe actualizar el stock correspondiente al último artículo del fichero e imprimirlo. Ejecuta de nuevo los subprogramas GRABAR y COMPRA.

Ahora sacamos el listado de las ventas de artículos que se encuentran en el fichero SALIDAS creado para este fin.

Nos posicionamos en el primer registro de los ficheros ARTICULOS y SALIDAS, inicializamos el contador de las filas y leemos del fichero de SALIDAS hasta llegar al final.

Al ser distinto el formato de las cabeceras para ambos listados, ejecuta CABESAL. Para buscar los datos de ese artículo debe leer del archivo de ARTICULOS o bien hacer *aux=nart3* y ejecutar el subprograma BUSCAR.

Para saber si tenemos que imprimir un asterisco o no, preguntamos si el stock es inferior al mínimo, *if stock<minimo*. Lo imprime en la columna cuatro, además del resto de los datos.

Cuando termine imprime las ventas totales, activa la pantalla y cierra ficheros.

Realizar el ejercicio sin utilizar el fichero secuencial SALIDAS.

RESUMEN

Un **reporte** es la visualización en pantalla o en impresora de una determinada información con un formato preestablecido, llamado *formato del informe*.

Nuestro programa, la lógica del tratamiento del reporte, depende del formato del informe.

Los conceptos fundamentales en los informes son:

- **Encabezamiento del reporte:** un título para el listado. Es aconsejable que vaya en una página separada.
- **Cabecera:** suele incluir el número de la página, el nombre de cada una de las columnas del listado, etc. Se escribe sólo una vez por página.
- **Línea de detalle:** contiene la información detallada, es decir, el valor de los registros y/o el resultado de alguna operación efectuada con algún campo del registro.
- **Pie:** se imprime después de las líneas de detalle. Sólo debe aparecer una vez por página.
- **Final de reporte:** se visualiza al final del listado con el resumen del reporte.

Algunos lenguajes tratan a la impresora como si fuera un fichero.

Para activar impresora: *activar impresora.*

El salto de página en la impresora: *salto de página.*

Para activar pantalla: *activar pantalla.*

Tomar la fecha interna del ordenador: *fecha del sistema.*

EJERCICIOS PROPUESTOS

1. Hacer un programa, que se ejecutará sobre el fichero de DATOS, que nos permita realizar las siguientes opciones:
   a) Consultar un registro por el DNI.
   b) Sacar un listado por impresora de todas las personas cuyo primer apellido empiece por una letra determinada.
   c) Listado por pantalla de todos los registros del fichero.
2. Del fichero de ARTICULOS sacar un listado por impresora de todos los artículos cuyo stock sea menor que el stock mínimo.
3. En el fichero secuencial ALUMNOS se guardan los datos personales de los alumnos de un determinado colegio, cuyos campos son:

NALUMNO	NOMBRE	DIRECCION	TELEFONO	CURSO

donde el campo CURSO es el curso en el cual está matriculado.

Por otro lado, en el fichero CURSOS están almacenados los distintos cursos que imparte el colegio, así como las asignaturas de cada curso, cuyos campos son:

CURSO2	ASIG1	ASIG2	ASIG3	ASIG4	ASIG5

En el fichero NOTAS guarda las notas, en valor numérico, de cada alumno con los campos:

NALUM	A1	A2	A3	A4	A5

Se desea hacer un programa con las siguientes opciones:

a) Introducir la nota, en número, a todos los alumnos de un curso.
b) Listado de todos los alumnos de un curso.

c) Listado de notas, que tiene las siguientes opciones:
   1. Notas de un alumno determinado.
   2. Notas de todos los alumnos de un curso determinado.
   3. Notas de todos los alumnos del colegio.

   El boletín de las notas tendrá el siguiente formato:

Curso Académico: ----/--	
Nombre: ----------	
Curso: ---	
Asignaturas	Calificación
----------	------
----------	------
----------	------
Nota media:	------

donde tanto la calificación como la nota media deben aparecer en letra, es decir: Muy Deficiente, Insuficiente, Aprobado, etc.

4. Con los mismos ficheros del ejercicio anterior, sacar un listado de todos los alumnos que hayan aprobado todas las asignaturas y en otra hoja aparte todos los alumnos que tengan alguna asignatura suspensa, así como el nombre de la/s asignatura/s.

5. Tenemos el fichero secuencial, ARTICULOS, con los siguientes campos:

NARTICULO	DESCRIPCION	ALMACEN	STOCK

Sacar un listado con todos los registros del fichero con el siguiente formato:

Relación de existencias al día: --/--/--			Pag.: --
Núm.Art.	Nombre	Almacén	Stock
----	----------	----	-----
----	----------	----	-----
----	----------	----	-----
		Número artículos en el almacén 5:	---
		Número total de piezas.....:	-----

La cabecera ocupará 6 líneas y se desea que el salto de página se efectúe cada 55 líneas.

En la línea 53 debe aparecer el número de artículos distintos que hay en el almacén número 5, y en la línea 54 la cantidad de piezas (suma del stock) que hay en cada página.

Al terminar el fichero debe imprimir:

Número total de artículos en el almacén 5:	------
Suma total de piezas.....:	----------

CAPITULO

Rupturas de control

16.1. INTRODUCCION

Dentro de la generación de informes se puede producir lo que se llama **ruptura de control**, que consiste en la modificación de la línea de detalle del informe debido a que los datos dejan de estar relacionados entre sí, porque varía el valor de alguno de los campos.

Ejemplo: si se listan las ventas realizadas por meses en unos grandes almacenes, se produce una ruptura de control cuando cambia el departamento. Dentro del departamento, al variar el mes y dentro de este al variar el nombre del vendedor. Además, controlando que cuando llegue el listado a una línea determinada se debe producir un salto de página.

Informe total de ventas						
Dpto.	Mes	Vendedor	Producto	Ventas	Vendedor Total-mes	TOTAL AL MES
A	1	Juan	---	---		
			---	---	----	
		Concha	---	---		
			---	---	----	-----
	2	Juan	---	---		
			---	---	----	
		Concha	---	---		
			---	---	----	-----
B	1	Julio	---	---		
			---	---	----	
		Nieves	---	---		
			---	---	----	-----
	2	Julio	---	---		
			---	---	----	
		Nieves	---	---		
			---	---	----	-----
					VENTAS TOTALES	---------

Ejercicio 1

Dado el fichero secuencial ALUMNOS.DAT, que contiene los datos de los alumnos de un colegio, cuyo formato es:

CURSO	NUMERO	NOMBRE	DIRECCION	IMPORTE

que está ordenado por el campo CURSO.

Se desea sacar un listado de cobro de tal forma que los alumnos de distintos cursos vayan en diferentes páginas según el formato:

Página: --	
Curso: ----	
Listado de alumnos	
Nombre	Importe
----------	----
----------	----
----------	----
Total: -----	

El cambio de página se realizará en la línea 55 o bien cuando cambie el curso.

Calcular el importe total por curso.

ESTUDIO PREVIO

Los alumnos de un mismo curso están uno a continuación del otro. Para saber dónde termina un curso y cuándo comienza otro distinto, necesitamos un *sw* y un auxiliar.

El *sw* nos permite guardar en el auxiliar el primer curso del fichero. Al comparar el auxiliar con el campo CURSO puede suceder:

- Son iguales: estamos con el mismo curso, imprimimos la línea de detalle y vamos acumulando los cobros.

  Si llegamos a la línea 55, hay que hacer un salto de página y volver a escribir la cabecera para este mismo curso.
- No son iguales: ha cambiado de curso.

  Escribimos el total del curso que acaba de terminar, realizamos un salto de página, el acumulador de los cobros se reinicializa a cero e imprimimos la cabecera para el nuevo curso.

  El nuevo curso se almacena en el auxiliar.

VARIABLES

aux = guardar el CURSO.
sw = permite almacenar el valor de CURSO en *aux* para el primer registro.
total = acumulador de los cobros de todos los alumnos de un mismo curso.
pag = contador de las páginas.
fi = contador de las filas.

PSEUDOCODIGO

```
total = 0
fi = 55
sw = 0
aux = espacios(12)
pag = 1
activar impresora
abrir ALUMNOS
hacer mientras no sea EOF()
     if sw = 0
          aux = curso
          sw = 1
     fin del if
     if fi = 55
          hacer CABECERA
     fin del if
     if curso = aux
          hacer IMPRIMIR
     else
          hacer TOTAL
          total = 0
          aux = curso
          hacer CABECERA
          hacer IMPRIMIR
     fin del if
     leer siguiente registro
fin del hacer
hacer TOTAL
activar pantalla
cerrar ALUMNOS
fin del programa.

CABECERA.
hacer salto de página
en 1,40 imprime "Página:", pag
en 3,40 imprime "Curso:"
en 3,48 imprime aux
en 5,30 imprime "Listado de alumnos"
```

```
en 6,20 imprime "Nombre"
en 6,45 imprime "Importe"
fi = 8
pag = pag + 1
IMPRIMIR.
en fi,15 imprime nombre
en fi,46 imprime importe
fi = fi + 1
total = total + importe
TOTAL.
en fi+2,37 imprime "Total:"
en fi+2,45 imprime total
```

COMPROBACION

Una vez abierto el fichero ALUMNOS, guardamos el valor del campo *curso*, correspondiente al primer registro, en el auxiliar. Esta operación solamente la realiza una vez por cambiar el estado del *sw* a uno.

El contador de las filas está inicializado a 55 e imprime la cabecera.

Preguntamos si *nart=aux*:

- Lo son, por lo menos la primera vez, realiza el subprograma IMPRIMIR, donde escribe la línea de detalle, y acumula el valor del campo IMPORTE en *total*.

  Leemos el siguiente registro; si el alumno pertenece al mismo curso, continúa ejecutando el subprograma IMPRIMIR y acumulando los cobros en *total*.

  En el caso de llegar a la línea 55 ejecuta de nuevo el subprograma CABECERA.
- No son iguales: estamos con un alumno perteneciente a un curso distinto.

  Hemos impreso todos los alumnos pertenecientes a un mismo curso. Ejecuta el subprograma TOTAL, el pie de página, es decir, el total de los cobros del curso finalizado, e inicializamos el acumulador *total* a cero.

  Al estar en un nuevo curso asignamos al auxiliar el valor de este curso, *aux=curso*. Ejecuta la cabecera e imprime los datos de este alumno; de lo contrario, no se acumularían y se perderían. Leemos el siguiente registro.

Cuando termine de leer todos los registros del fichero debemos imprimir los cobros del último curso, para lo cual ejecuta el subprograma TOTAL, porque nos encontramos fuera del ciclo, *hacer mientras no sea EOF()*.

Ejercicio 2

Dado el fichero secuencial ALUMNOS.DAT, que contiene los datos de los alumnos de un colegio, cuyo formato es:

CURSO	GRUPO	NOMBRE	IMPORTE

el cual está ordenado por los campos CURSO y GRUPO.

Sacar un listado de cobros por curso teniendo en cuenta que por cambio de grupo imprimimos el importe total del grupo y se realiza un salto de página. Por cambio de curso se imprime el importe total de todos los grupos correspondientes a ese curso y también cambiamos de página.

Imprimir la suma del importe de todos los alumnos del fichero.

Formato de impresión:

Página: --	
Listado de alumnos del curso: -----	
Grupo: -	
Nombre	Importe
----------	----
----------	----
----------	----
Importe: ------	

Página: --	
Listado de alumnos del curso: -----	
Grupo: -	
Nombre	Importe
----------	----
----------	----
----------	----
Importe: ------	
Total curso ------- : -------	

El cambio de página se realizará en la línea 55.

ESTUDIO PREVIO

Los alumnos están ordenados dentro del fichero por curso y dentro del curso por grupo.

Para saber cuándo termina un curso y cuándo comienza uno nuevo necesitamos un *sw* y un auxiliar, lo mismo que en el ejercicio anterior; pero ahora, además, hay que controlar [illegible] otro auxiliar.

Sólo es necesaria la utilización de un *sw*. Este nos permite almacenar al mismo tiempo en los auxiliares los valores del curso y del grupo para el primer alumno del fichero.

El grupo siempre cambiará antes o al mismo tiempo, en el caso de que el curso no posea más grupos, que el curso, pero nunca al revés. Es decir, la condición de cambio de curso irá dentro de la de cambio de grupo por haber menor o igual número de cursos que de grupos (será igual cuando cada curso sólo tenga un grupo).

Mientras el grupo no cambie, imprimimos la línea de detalle y sumamos el importe. En el caso de llegar a la línea 55 se hace un salto de página y debe hacer la cabecera para ese mismo curso.

Cuando cambie el grupo se imprimirá el importe correspondiente al grupo finalizado. Se acumula para calcular el importe del curso. Hacemos salto de página, además de reinicializar el valor del importe del curso a cero.

Al terminar los alumnos pertenecientes a un mismo curso se imprime el total del curso, se acumula su valor en el totalizador de todo el fichero y se salta de página.

El cálculo del importe de todos los alumnos del fichero también se puede realizar mediante una variable que vaya recogiendo el valor del importe de todos los alumnos del archivo.

Cuando el ordenador finalice de leer todos los registros del fichero imprimimos el valor de esta variable.

VARIABLES

aux1 = guardar el CURSO.
aux2 = guardar el GRUPO.
sw = permite almacenar el valor de CURSO y del GRUPO en los auxiliares para el primer registro.
gru = acumulador de los cobros de todos los alumnos de un mismo grupo.
cur = acumulador de los cobros de todos los alumnos de un mismo curso.
total = acumulador de los cobros de todos los alumnos del fichero.
pag = contador de las páginas.
fi = contador de las filas.

PSEUDOCODIGO

```
gru = 0
cur = 0
total = 0
fi = 55
sw = 0
aux1 = espacios(12)
aux2 = espacios(2)
pag = 1
activar impresora
abrir ALUMNOS
hacer mientras no sea EOF()
     if sw = 0
          aux1 = curso
```

```
            aux2 = grupo
            sw = 1
        fin del if
        if fi = 55
            hacer CABECERA
        fin del if
        if grupo = aux2
            hacer IMPRIMIR
        else
            hacer GRUPO                    *imprimir el importe del grupo
            aux2 = grupo
            if curso <> aux1
                hacer CURSO                *imprimir el importe del curso
                aux1 = curso
            fin del if
            hacer CABECERA
            hacer IMPRIMIR
        fin del if
        leer siguiente registro
    fin del hacer
    hacer TOTAL
    activar pantalla
    cerrar ALUMNOS
    fin del programa.

    CABECERA.
    hacer salto de pagina
    en 1,60 imprime "Página:", pag
    en 3,25 imprime "Listado de alumnos del curso:"
    en 3,56 imprime aux1
    en 5,40 imprime "Grupo:"
    en 5,48 imprime aux2
    en 7,20 imprime "Nombre"
    en 7,45 imprime "Importe"
    fi = 9
    pag = pag + 1

    IMPRIMIR.
    en fi,15 imprime nombre
    en fi,46 imprime importe
    fi = fi + 1
    gru = gru + importe
    *total = total + importe

    GRUPO.
    fi = fi + 2
```

```
en fi,33 imprime "Importe:"
en fi,45 imprime gru
cur = cur + gru
gru = 0

CURSO.
fi = fi + 2
en fi,20 imprime "Total curso"
en fi,33 imprime aux1
en fi,44 imprime ":"
en fi,45 imprime cur
total = total + cur
cur = 0

TOTAL.
hacer GRUPO
hacer CURSO
en fi+2,15 imprime "Total de todos los alumnos:"
en fi+2,44 imprime total
```

COMPROBACION

Una vez abierto el fichero ALUMNOS, guardamos el valor de los campos curso y grupo en los respectivos auxiliares por estar el *sw* inicializado a cero. Imprime el subprograma CABECERA por valer *fi 55*.

Preguntamos si *grupo=aux2:*

- Lo son, por encontrarnos en el primer registro, realiza el subprograma IMPRIMIR: visualiza la línea de detalle y acumula el valor del campo *importe* en *gru*.

  Leemos el siguiente registro; si el alumno pertenece al mismo grupo, continúa ejecutando el subprograma IMPRIMIR y acumulando el importe en *gru*.

  En el caso de llegar a la línea 55 ejecuta de nuevo el subprograma CABECERA.
- No son iguales: estamos con un alumno perteneciente a un grupo distinto e imprimiremos el importe de ese grupo, para lo cual ejecuta el subprograma GRUPO. El valor del *aux2* debe contener el nuevo valor del grupo. El importe del *gru* lo acumula al del curso. *cur*. Reasignamos *gru* a cero.

  Al cambiar de grupo puede que cambie de curso o puede que no. Preguntamos por el valor de aux1, *if curso< >aux1:*

  – Son distintos: hemos cambiado también de curso. Hay que imprimir el importe del curso en la misma página que el del grupo; por tanto, ejecuta el subprograma CURSO. Acumula en el contador del *total* el valor del importe del curso y se reinicializa el *cur* a cero. El valor del *aux1* toma el valor del nuevo curso.

  – No son distintos: continuamos con el mismo curso. No debe hacer lo descrito en el punto anterior.

  Tanto si cambia de curso como si no, el nuevo grupo debe ir en una página nueva. Ejecuta CABECERA.

Terminados de imprimir todos los registros del fichero, ejecuta el subprograma TOTAL: imprimir el total del último grupo, hacer GRUPO, imprimir el importe del nuevo curso, hacer CURSO, e imprimir el total de todos los alumnos que está guardado en la variable *total*.

Ejercicio 3

Dado el fichero secuencial PERSONAL.DAT, que contiene los datos de los empleados de una empresa, con el formato:

NOMBRE	EDAD	DIRECCION	SUELBRUTO	EC

que está ordenado por el campo EC: estado civil.

Sacar un listado del personal de esta empresa según el formato:

Página: --			
Listado de cobros			
Nombre	Edad	Dirección	S.Neto
----------	--	----------	------
----------	--	----------	------
----------	--	----------	------

Teniendo en cuenta que el salario neto se calcula mediante la siguiente operación:

$$SNETO = SUELBRUTO - DESCUENTO$$

El descuento se busca en la tabla DESCUENTO de dos columnas: la primera almacena el estado civil y la segunda el descuento que se aplica a dicho estado civil.

El cambio de página se realizará en la línea 55 o por cambio de estado civil. Al cambiar el estado civil, imprimir el sueldo neto total.

Imprimir el número de personas que cumplen la condición de edad mayor que una determinada que se introducirá por teclado y su sueldo neto sea mayor que una cantidad también introducida por teclado.

ESTUDIO PREVIO

Para controlar el cambio de estado civil emplearemos, como ya hemos visto, un *sw* y un auxiliar.

El descuento se aplica según el estado civil del empleado. Al estar guardado en una tabla tenemos que buscarlo, para lo cual hay que recorrer la tabla desde la primera posición, pues

no nos dicen nada acerca de la misma y, por tanto, debemos suponer que está desordenada. Para recorrer las distintas posiciones de la tabla nos ayudamos de una variable.

Al cambiar el estado civil se imprimirá el sueldo neto total y hacemos un salto de página.

A la vez que controlamos si cambia o no el estado civil, comprobamos si la edad y el sueldo neto de ese empleado es mayor que el valor de las variables, que previamente, y antes del ciclo de lectura del fichero, habremos introducido por teclado. Una vez finalizado el proceso de lectura del archivo imprimimos el número de personas que cumplen esas dos condiciones.

VARIABLES

aux = guardar el estado civil del empleado.
sw = permite almacenar el campo EC en el auxiliar.
total = acumulador del sueldo neto total de los trabajadores de un determinado estado civil.
pag = contador de las páginas.
fi = contador de las filas.
i = contador para recorrer la tabla.
ed = variable para introducir la edad.
suel = variable para introducir el sueldo neto.
son = contador de los empleados que cumplen las condiciones de la edad y sueldo neto.

PSEUDOCODIGO

```
ed = 0
suel = 0
total = 0
son = 0
fi = 55
sw = 0
aux = espacios(1)
pag = 1
borrar pantalla
en 8,10 imprime "Empleados con una edad superior a:"
en 8,47 introduce ed
en 10,10 imprime "y con un sueldo neto mayor de:"
en 10,47 introduce suel
activar impresora
abrir PERSONAL
hacer mientras no sea EOF()
     if sw = 0
          aux = ec
          hacer BUSCAR
          sw = 1
     fin del if
     if fi = 55
```

```
            hacer CABECERA
        fin del if
        if ec = aux
            hacer IMPRIMIR
            hacer CONTAR
        else
            hacer NETO
            aux = ec
            hacer BUSCAR
            hacer CABECERA
            hacer IMPRIMIR
            hacer CONTAR
        fin del if
        leer siguiente registro
    fin del hacer
    hacer NETO
    hacer CUANTOS
    activar pantalla
    cerrar PERSONAL
    fin del programa.

    CABECERA.
    hacer salto de pagina
    en 1,60 imprime "Página:", pag
    en 3,35 imprime "Listado de cobros"
    en 5,15 imprime "Nombre"
    en 5,35 imprime "Edad"
    en 5,40 imprime "Dirección"
    en 5,65 imprime "S.Neto"
    fi = 9
    pag = pag + 1

    IMPRIMIR.
    en fi,5 imprime nombre
    en fi,35 imprime edad
    en fi,40 imprime direccion
    en fi,63 imprime suelbruto - DESCUENTO(i,2)
    fi = fi + 1
    total = total +  suelbruto - DESCUENTO(i,2)

    BUSCAR.
    i = 1
    hacer mientras DESCUENTO(i,1) <> ec
        i = i + 1
    fin del hacer
```

```
NETO.
fi = fi + 2
en fi,33 imprime "Neto total:"
en fi,47 imprime total
total = 0

CONTAR.
if edad > ed  AND  (suelbruto - DESCUENTO(i,2)) > suel
     son = son + 1
fin del if

CUANTOS.
fi = fi + 2
en fi,15 imprime "Número de empleados con edad superior a ",ed
en fi+1,15 imprime "con un sueldo neto superior a", suel
en fi+2 imprime "son:", son
```

COMPROBACION

Después de inicializar las variables, y antes de activar la impresora, se introducen por teclado los valores de la edad y el sueldo para contar los empleados que cumplen ambas condiciones.

Una vez abierto el fichero PERSONAL, guardamos el valor del campo estado civil en el auxiliar. Ejecuta el subprograma BUSCAR, que busca el descuento correspondiente a dicho estado civil. Imprime la cabecera.

En BUSCAR simplemente tiene que buscar la posición de la tabla en la cual se encuentra dicho estado civil. Comienza desde la fila uno, *i=1*, comprobando si el estado civil guardado en la columna uno coincide con el del fichero, *hacer mientras DESCUENTO(i,1) < > ec.* En caso de no ser iguales, accedemos a la siguiente posición.

Preguntamos si el estado civil y el auxiliar tienen el mismo valor. *if ec=aux*:

- Sí: imprime los datos correspondientes a dicho empleado. El cálculo del sueldo neto se realiza restando al campo *suelbruto* el valor del descuento situado en la columna segunda de la tabla DESCUENTO y en la fila i, que previamente hemos hallado en el subprograma BUSCAR, *suelbruto–DESCUENTO(i,2).*

  Comprueba si la edad y su sueldo neto es mayor que los datos introducidos por teclado, para ello ejecuta CONTAR.

  Al mismo tiempo controlamos si hay que cambiar de página.
- No: estamos en un estado civil distinto. Imprime la suma de los sueldos netos del estado civil anterior, ejecuta NETO y cambia el valor del auxiliar.

  Para no perder los datos de este empleado busca el descuento a aplicar a este nuevo estado, imprime la cabecera, los datos del empleado y comprueba las condiciones de la edad y el sueldo neto.

Una vez finalizado el fichero imprimimos el sueldo neto total del último estado civil. Ejecuta NETO.

En la misma hoja, dos filas más abajo, imprime el número de empleados que cumplen las condiciones especificadas.

RESUMEN

La **ruptura de control** consiste en la modificación de la línea de detalle por dejarse de cumplir alguna condición.

Pueden existir niveles de ruptura **intermedios** dentro de un nivel superior.

Una vez procesados todos los datos, la impresión de los totales se llama **cambio de control final.**

El encabezamiento o el pie que se imprime antes o después de un grupo de líneas de detalle relacionadas con los cambios de control, se llaman **encabezamiento** o **pie de control.**

EJERCICIOS PROPUESTOS

1. En el fichero secuencial CLIENTES.DAT se guardan los datos personales de los clientes de un banco, cuyos campos son:

NCC	NOMBRE	DIRECCION	POBLACION	PROVINCIA	SALDO

donde el campo SALDO almacena la cantidad actual del cliente: puede ser un saldo deudor.

También se dispone de otro fichero secuencial con los clientes que han tenido movimiento en sus cuentas corrientes, cuyos campos son:

NCC	MOVIMIENTO	TIPO

donde:

— MOVIMIENTO almacena la cantidad.

— TIPO puede tomar los valores:

1. Cero: una imposición.
2. Uno: un reintegro.

Este fichero está ordenado por el número de la cuenta corriente.

Hacer un programa que liste por impresora todos los clientes cuyas cuentas corrientes han tenido movimiento, de tal forma que cada cliente vaya en una hoja distinta. Actualizar su saldo.

2. En el fichero secuencial COCHES.DAT se guardan los datos de aquellas personas que tienen coche, así como las características de estos, cuyos campos son:

DNI	NOMBRE	DIRECCION	MATRICULA	COCHE

donde COCHE almacena el tipo de coche.

En una tabla de dos dimensiones, CODIGO, se almacenan el tipo de multa en su primera columna y en la segunda la penalización correspondiente.

En el fichero MULTAS están guardadas las personas que han cometido alguna infracción, cuya estructura es:

MATRICULA	INFRACCION	FECHA

donde INFRACCION almacena el tipo de infracción cometido. Está ordenado por número de matrícula y un conductor puede tener más de una multa.

Hacer un programa que liste por impresora las infracciones de todo el fichero de MULTAS, de tal forma que todas las multas de un mismo conductor irán en la misma hoja.

3. Unos grandes almacenes guardan en el fichero secuencial PERSONAL.DAT, los datos de sus empleados, cuya estructura es la siguiente:

CODIGO-EMPLEADO	NOMBRE	APELLIDOS	DEPARTAMENTO

En el fichero secuencial VENTAS.DAT, las ventas de todos los productos, cuya estructura es:

CODIGO-EMPLEADO	CODIGO-ARTICULO	MES	CANTIDAD

donde el campo MES posee una longitud de dos caracteres para guardar el número del mes. En CANTIDAD se guardan las unidades vendidas del artículo cuyo código está almacenado en CODIGO-ARTICULO.

En un tercer fichero secuencial, ARTICULOS.DAT, están descritas las características de cada uno de los artículos que venden. El formato del fichero es:

CODIGO-ARTICULO	CONCEPTO	PROVEEDOR	PRECIO

Por último, en el array unidimensional MES(12) están los nombres de los meses que se corresponden con la posición que ocupan dentro del vector.

Sacar un listado por impresora:

1. De las ventas totales del almacén si se produce una ruptura de control al cambiar el mes, el departamento o el vendedor por este orden.
2. Las ventas de un departamento en todo el año.
3. Las ventas de un departamento en un mes determinado.
4. Las ventas de un vendedor durante un mes.
5. Las ventas de un vendedor durante un año.
6. Las ventas de un determinado producto.

El mes debe aparecer en letra.

En la línea 55 se produce un salto de página.

Suponer que el fichero de PERSONAL está ordenado por el campo DEPARTAMENTO y dentro de éste por CODIGO-EMPLEADO. A su vez, el fichero de VENTAS, está ordenado por el campo MES y dentro de él por el campo CODIGO-EMPLEADO. Por otro lado, el campo CODIGO-ARTICULO sólo aparece una vez por empleado en VENTAS.

El fichero ARTICULOS está ordenado por CODIGO-ARTICULO.

CAPITULO

Ficheros con organización directa o aleatoria

17.1. INTRODUCCION

Los ficheros con organización directa o **aleatoria** se caracterizan por grabar los registros, que componen el fichero, en unas direcciones físicas obtenidas a través de unas transformaciones matemáticas en sus campos indicativos.

Para obtener la relación campo indicativo-dirección física de cada uno de los registros, se aplican a dichos campos unas fórmulas o algoritmos de transformación, cuyos valores, lógicamente, deben estar comprendidos en el intervalo de direcciones físicas disponibles en el soporte en el que estamos grabando el fichero.

Los ficheros con organización secuencial se caracterizaban porque los registros se grababan secuencialmente, en el mismo orden en que se introducían. Entre dos registros consecutivos no puede haber ningún espacio libre, y para acceder a un registro en particular hay que pasar obligatoriamente por todos los registros que ocupan posiciones físicas anteriores.

En los ficheros con organización directa:

- Los registros se graban en la dirección física resultante de aplicar el algoritmo a su campo clave.
- Sí puede haber huecos entre un registro y otro.
- Se puede acceder directamente al registro deseado sin necesidad de tener que pasar por todas las posiciones físicas anteriores.

Para poder acceder directamente a los registros el fichero debe estar grabado en soportes direccionables.

El algoritmo de transformación de claves debe cumplir una serie de condiciones:

- Establecer la correspondiente relación entre el campo y la dirección física dentro de los límites del soporte de datos.
- Producir el menor número posible de **sinónimos**: varios registros con la misma dirección física.

Estos sinónimos, en caso de producirse, se grabarán en un fichero con organización secuencial.

- Aprovechamiento al máximo del soporte: dejar el menor número posible de huecos, es decir, direcciones físicas vacías.

A la hora de acceder a un determinado registro se le debe aplicar al campo indicativo la misma fórmula o algoritmo utilizado en el proceso de su grabación.

En la mayoría de los lenguajes que trabajan con ficheros de organización directa los campos deben ser todos alfanuméricos.

17.2. METODOS DE TRANSFORMACION DE CLAVES

Los métodos de transformación de claves más utilizados son los siguientes:

- **Direccionamiento directo:** se emplea cuando los campos indicativos son consecutivos o tienen muy pocos huecos. No producen sinónimos.
- **Direccionamiento por conversión a octal:** consiste en asignar a cada dígito del campo indicativo su valor en octal. Posteriormente se suman todos los valores. Produce sinónimos.
- **Direccionamiento por el centro al cuadrado:** se eleva el campo indicativo al cuadrado y se eliminan las cifras extremas, tanto de la derecha como de la izquierda. Produce sinónimos.

17.3. ESTRUCTURA DE LOS FICHEROS ALEATORIOS

La forma siguiente de estructurar los ficheros aleatorios no es única y, por tanto, cada usuario puede crear la suya.

Si en el algoritmo de transformación de claves se emplea el direccionamiento directo, no produce sinónimos y cada registro estará grabado en posiciones físicas diferentes. Para modificar o consultar los registros basta con ir a la posición que ocupan.

Pero no siempre se puede utilizar este tipo de direccionamiento por la cantidad de huecos que deja. Por ejemplo: para grabar el DNI número 3.333.333, lo realiza en dicha posición de memoria, pero esto lleva consigo el tener que reservar espacio para 3.333.332 direcciones físicas anteriores, y si hay muy pocos registros la cantidad de espacio a utilizar es muy grande frente al volumen de datos del fichero.

La mayoría de las ocasiones habrá que utilizar un algoritmo de transformación de claves.

Al someter al campo indicativo o campo clave a un algoritmo para su transformación, es muy probable que produzca sinónimos. Estos también se tienen que grabar, pero su sitio está ocupado por otro registro. Para estos sinónimos utilizamos otro fichero, también con organización directa.

El algoritmo de transformación puede variar de unos programas a otros, en función del campo clave y procurando, en todo momento, que produzca el menor número posible de sinónimos.

A los registros de un determinado fichero *siempre* se les someterá al mismo algoritmo.

Resumiendo, vamos a tener dos ficheros con organización directa. En el primero, DATOS.DAT, se grabarán los registros cuya posición está libre y en el segundo, SINONIMOS.DAT, los sinónimos. Ambos ficheros van a estar conectados entre sí por medio de un *puntero*.

El **puntero** va a ser un campo del fichero DATOS.DAT que contiene la dirección física en la que se encuentra grabado el sinónimo correspondiente en el fichero SINONIMOS.DAT.

Sea el fichero DATOS.DAT, con organización directa, cuyos campos son:

DNI	NOMBRE	APELLIDOS	DIRECCION	PUNTERO

donde el campo *PUNTERO* indica la dirección física en la cual está grabado el sinónimo correspondiente en el segundo fichero.

El otro fichero, SINONIMOS.DAT, cuya estructura es idéntica a los DATOS:

DNI	NOMBRE	APELLIDOS	DIRECCION	PUNTERO

están grabados los sinónimos cuya posición física está ocupada por otro registro en DATOS. El campo *PUNTERO* indica el número de registro en el cual está grabado otro sinónimo.

Los datos en el fichero SINONIMOS no se van a grabar aleatoriamente, sino que se hará en la posición siguiente al último registro ocupado, siempre y cuando no haya posiciones libres intermedias producidas al borrar registros.

A pesar de tener organización directa lo tratamos "casi" como un secuencial en cuanto a que un registro está físicamente a continuación del anterior.

Para lo cual, *el primer registro debe ser un registro de información* o *de cabecera*. Para este primer registro el campo:

- DNI: guarda el número de registros ocupados.
- NOMBRE: almacena la posición de un registro vacío; es decir, en esa posición antes había un registro y en una operación previa lo hemos borrado.

  Para el resto de los registros el campo *NOMBRE* guarda el valor de otro registro vacío. Cuando no haya más posiciones vacías, en dicho campo habrá un cero.

Supongamos que tenemos la siguiente información grabada en los ficheros con organización directa: DATOS.DAT y SINONIMOS.DAT:

DATOS

Dirección física	DNI	NOMBRE	APELLIDOS	PUNTERO
1A1	0			0
1B1	6543	Agueda	Sánchez	7
1C1	0			2
1D1	1234	Francisco	Bartol	0
1E1				
1F1				
200	8976	José	Hernández	0

SINONIMOS

Registro de información o cabecera	DNI	NOMBRE	APELLIDOS	PUNTERO
1	7	4		
2	1021	Marisa	....	6
3		0		
4		3		
5	4563	Luis	....	0
6	1120	David	....	0
7	3456	Javier	....	5

Indica las posiciones libres dentro del fichero en caso que las hubiere.

Nuestro fichero DATOS ocupa desde la posición física 1A1 hasta la 200. Al aplicarle al campo DNI un algoritmo de transformación de claves hay un valor que produce la dirección física 1A1 y otro la 200. El fichero va a ocupar todas las posiciones entre la primera dirección y la última, aunque haya huecos entre ellas.

El primer registro indica que antes había un registro grabado, por ser DNI=0. No hay ningún sinónimo por valer el puntero cero.

En el segundo registro hay datos personales y el puntero señala un sinónimo en la posición siete del fichero SINONIMOS. En SINONIMOS el puntero del registro siete indica que hay otro sinónimo grabado en la posición cinco. Este, por último, nos dice que ya no hay más sinónimos por valer su puntero cero.

En el tercer registro de DATOS había información pero ésta ha sido borrada. DNI=0. No obstante, tiene un sinónimo en la posición dos. Este registro, a su vez, indica la existencia de otro sinónimo en la posición seis.

En el siguiente registro del archivo DATOS hay una información guardada y no tiene sinónimos. PUNTERO=0.

Los otros dos registros están vacíos, DNI=espacios. Se llenarán si el algoritmo de transformación de claves produce dichas direcciones.

El último también posee datos personales y no tiene ningún sinónimo. PUNTERO=0.

Para el fichero de SINONIMOS el campo DNI del primer registro informa que hay siete registros, de los cuales sólo los seis siguientes van a almacenar información personal.

El campo NOMBRE, del primer registro, guarda la primera posición libre, el primer hueco, que se llenará en el caso de producirse algún sinónimo.

Este mismo campo, NOMBRE, para el resto de los registros, en el caso de no albergar el nombre de alguna persona, guarda la posición del siguiente hueco. El último, siguiendo la secuencia, almacena un cero: no hay más registros vacíos.

En un programa aparte hay que preparar la estructura del primer registro del fichero SINONIMOS; será la siguiente:

1	1	0		

Los registros de los ficheros con organización directa deben ser todos de la misma longitud para que el ordenador se pueda posicionar directamente en el registro indicado.

- **Instrucciones de los ficheros con organización directa.**

Abrir ficheros lo denotaremos de igual forma que en los secuenciales:

abrir nombre–fichero

pero, además, hay que indicarle cuál va a ser la longitud de cada campo expresada en bytes, teniendo en cuenta que un carácter es un byte.

Leer registros:

leer nombre–fichero, número–de registro.

Grabar registros:

grabar nombre–fichero, número–de–registro.

Cerrar fichero:

cerrar nombre–fichero.

Longitud del fichero:

longitud nombre–fichero.

Con la longitud del fichero podemos saber en todo momento cuántos registros hay, dividiendo la longitud del archivo por la longitud de un registro.

Ejercicio 1

Hacer un pseudocódigo que nos permita dar ALTAS en el fichero DATOS de organización directa, controlando las altas duplicadas.

ESTUDIO PREVIO

Introduciremos por teclado el campo clave, DNI, y lo sometemos a un algoritmo de transformación. Puede suceder que la dirección física resultante sea:

— Blanco: grabamos el registro en esa posición y ponemos el puntero a cero.
— Cero: comprobamos cuál es el valor del campo puntero. Si es:
  a) Cero: grabamos el registro en esa posición, porque no hay ningún sinónimo.
  b) Distinta de cero: hay sinónimos. Leemos todos ellos siguiendo la secuencia del puntero. Comparamos el valor introducido por teclado con el campo DNI de estos registros:
    1. Son iguales: alta duplicada y dejamos de leer.
    2. No son iguales después de pasar por todos ellos: introducimos el resto de la información y grabamos el registro en DATOS en la dirección generada al transformar la clave.
— Distinto de cero: hay un registro grabado en esa posición. Comprobamos si es igual al dato introducido:
  a) Iguales: alta duplicada.
  b) No son iguales: comprobamos el valor del puntero. Puede tomar los valores:
    1. Cero: grabamos el registro en SINONIMOS.
    2. Distinta de cero: repetir los pasos 1 y 2 del apartado anterior, excepto que el registro se graba ahora en SINONIMOS.

La posición donde hay que grabar los nuevos datos en SINONIMOS la da el registro cabecera. Primero comprobamos si hay alguna posición intermedia libre, para lo cual leeremos el segundo campo del primer registro, el de información: si es:

- Cero o blancos: no hay huecos. Grabamos los datos en la posición posterior al valor indicado en el primer campo del registro cabecera.
- Distinto de cero: grabamos los datos en esa posición.

Tanto en un caso como en otro hay que actualizar el puntero del sinónimo anterior que puede estar en DATOS o en SINONIMOS y el registro cabecera.

Sea la situación de los ficheros como se muestra en la página siguiente:

DATOS		Puntero
0		0
6543	Agueda Sánchez	7
0		2
1234	Francisco Bartol	0
8976	José Hernández	0

SINONIMOS

1	7	4		
2	1021	Marisa		6
3		0		
4		3		
5	4563	Luis		0
6	1120	David		0
7	3456	Javier		5

Supongamos que introducimos el DNI de valor 5346 y al aplicarle el algoritmo produce la dirección del segundo registro de DATOS. Hay información; al comparar vemos que no son iguales. El puntero tiene un valor distinto de cero; hay sinónimos. Leemos todos ellos, siguiendo la secuencia del puntero. Ninguno es igual al DNI introducido.

Su escritura se hará en la posición referenciada por el campo NOMBRE del primer registro.

SINONIMOS

	DNI	NOMBRE	APELLIDOS	PUNTERO
1	7	3		
2	1021	Marisa	. . . .	6
3		0		
4	**5346**	**Ana**	**Fernández**	**0**
5	4563	Luis	. . . .	**4**
6	1120	David	. . . .	0
7	3456	Javier	. . . .	5

En el registro cabecera, en el campo NOMBRE, se ha escrito el valor del campo de la posición en la que se grabará el registro.

En el puntero del último sinónimo se ha escrito la posición en la que se ha grabado la nueva información.

VARIABLES

dn = introducir el DNI.
nom = introducir el nombre.
ape = introducir los apellidos.
dir = introducir la dirección.
swh = controla si la posición en DATOS está vacía, hay un hueco.
sw = controla si el nuevo DNI está en el archivo SINONIMOS.
aux = guarda la posición del último sinónimo.
num = almacena la dirección física producida al aplicar el algoritmo sobre *dn*.
donde = posición del registro donde se graban los datos en SINONIMOS.
hueco = salvar el valor del segundo campo del registro *donde* y modificar el registro cabecera.
i = contador para saber si hay que modificar el puntero en DATOS o en SINONIMOS.
res = para dar más altas.

PSEUDOCODIGO

```
res = "S"
abrir ficheros de DATOS y SINONIMOS
hacer mientras res = "S"
     dn = 0
     nom = espacios(15)
     ape = espacios(30)
     dir = espacios(35)
     aux = espacios(2)
     hueco = espacios(2)
     swh = 0
     sw = 0
     num = 0
     donde = 0
     i = 0
     borrar pantalla
     en 10,20 imprime "D.N.I.:"
     en 10,29 introduce dn
     num = algoritmo sobre dn
     leer de DATOS,num
     if DNI=str(dn)
          hacer ALTA-DUPLICADA
     else
          if val(DNI)=0  OR  DNI=espacios
               swh = 1
          fin del if
          if val(puntero)=0  OR  puntero=espacios
               hacer INTRODUCIR
               puntero = str(0)
               grabar en DATOS,num
```

```
        else
            hacer BUSCAR
            if sw = 0
                hacer INTRODUCIR
                if swh = 1
                    grabar en DATOS,num
                else
                    hacer ENCONTRAR-SITIO
                    grabar en SINONIMOS,I
                    puntero = str(donde)
                    if i = 0
                        grabar en DATOS,num
                    else
                        grabar en SINONIMOS,val(aux)
                    fin del if
                    puntero = str(0)
                    grabar en SINONIMOS,donde
                fin del if
            else
                hacer ALTA-DUPLICADA
            fin del if
        fin del if
    fin del if
    hacer MAS
fin del hacer
cerrar ficheros
fin del programa.

INTRODUCIR.
en 12,20 imprime "Nombre:"
en 14,20 imprime "Apellidos:"
en 16,20 imprime "Dirección:"
en 12,29 introduce nom
en 14,32 introduce ape
en 18,32 introduce dir.

ENCONTRAR-SITIO.
leer de SINONIMOS,1
if val(nombre)<>0  AND  nombre<>espacios
    donde = val(nombre)
    leer de SINONIMOS,donde
    hueco = val(nombre)
    nombre = str(hueco)        * actualizar registro cabecera
else
    donde = val(dni) + 1
    dni = str(donde)           * actualizar registro cabecera
fin del if
```

```
BUSCAR.
aux = puntero
i = 0
hacer mientras val(puntero) <> 0  AND  sw = 0
     leer de SINONIMOS,val(puntero)
     if DNI = str(dn)
          en 20,10 imprime "Alta Duplicada"
          detener-listado
          sw = 1
     else
          if val(puntero) <> 0
               i = i + 1
               aux = puntero
          fin del if
     fin del if
fin del hacer

ALTA-DUPLICADA.
en 20,10 imprime "Alta Duplicada"
detener-listado
```

COMPROBACION

Al valor del DNI introducido por teclado se le aplica un algoritmo de transformación de claves, dando un número *num*.

Leemos ese registro, *leer de DATOS, num*, comparándolo con el valor allí guardado. Al ser las variables de distinto tipo convertimos el valor numérico en alfanumérico (los campos en los ficheros con organización directa, por lo general, es obligatorio que sean alfanuméricos), *if DNI = str(dn)*:

- Son iguales: alta duplicada.
- Distintos: no son iguales por ahora. Comprobamos si hay un hueco en esa posición, *if val(DNI)=0 OR DNI=espacios*, lo hay. Aquí grabaremos los datos en el caso de no ser alta duplicada. Lo controlamos cambiando el estado del sw de los huecos, *swh=1*.

Para todos estos registros swh =1	0		2	Tiene sinónimos
				Hueco
	0		0	No tiene sinónimos

El valor de *puntero* nos dice si tiene sinónimos o no, *if val(puntero)=0 OR puntero=espacios*:

- Se cumple: no tiene. Introducimos el resto de los datos mediante el subprograma

INTRODUCIR. Movemos un cero al campo puntero. Partimos de la base que si un registro no tiene sinónimos, ese campo vale cero, *puntero=str(0)*. Grabamos en DATOS en la posición num:

5555	Bernabé . . .	0

- No se cumple: hay sinónimos. Comprobaremos si es alta duplicada.

Situación del fichero DATOS y valores del *swh* cuando se presentan sinónimos para dicho registro:

swh = 1	0		7
swh = 0	2222	Juncal Artigas	7

Ejecuta BUSCAR: recorre todos los sinónimos siguiendo la secuencia del puntero o, mientras los DNI no sean iguales, realiza el ciclo *hacer mientras val(puntero)< >0 AND sw=0*. Lee del fichero SINONIMOS el registro que señala el puntero y comparamos los DNI:

- Son iguales: es alta duplicada, para salir del bucle *sw=1*.
- No son iguales: comprobamos el valor del puntero en el registro que acabamos de leer; si es distinto de cero, hay más sinónimos, aumentamos el contador *i*. y cambiamos el valor del auxiliar, *aux=puntero*. Si el puntero fuera cero, se ha llegado al último y el auxiliar no debe cambiar.

Para comprobar si el dato estaba o no, preguntamos por el estado del *sw*:

- Uno: alta duplicada.
- Cero: el dato no está. Introducimos el resto de la información en INTRODUCIR.

Dependiendo del valor de *swh* grabaremos la información en un fichero o en otro. *if swh=1:*

- Se cumple: hay un hueco en DATOS, se graba en dicha posición.

grabar en DATOS.num

5555	Bernabé . . .	7

- No se cumple: se graba en SINONIMOS, pero ¿dónde? Nos lo dice ENCONTRAR-SITIO.

Leemos el registro cabecera:

	DNI	NOMBRE		
1	7	4		

4		3		

si el campo NOMBRE es distinto de cero y no está en blanco, significa que hay huecos, por ejemplo en la posición cuatro, actualizamos el registro cabecera, *grabar en SINONIMOS,1:*

	DNI	NOMBRE		
1	7	3		

Hay que actualizar el puntero del sinónimo anterior, pero éste puede estar en DATOS o en SINONIMOS, *puntero=str(donde);* lo dice el valor del contador i, *if i=0:*

— Sí: el sinónimo anterior está en DATOS, *grabar en DATOS,num:*

Dirección física			
200	8976	José Hernández	**4**

— No: está en SINONIMOS, *grabar en SINONIMOS,val(aux):*

6	1120	David		**3**

Por último, grabamos los datos:

puntero = str(0) *grabar en SINONIMOS,val(aux)*

4	**5346**	**Ana**	**Fernández**	**0**

Cuando no deseemos dar más altas cerramos ficheros.

Ejercicio 2

Hacer un pseudocódigo que nos permita dar bajas en el fichero anterior, controlando las bajas inexistentes.

ESTUDIO PREVIO

Al valor del DNI que deseamos dar de baja se le somete al mismo algoritmo de transformación. Leemos dicho registro: puede suceder:

— Son iguales: en el campo DNI se graba un cero.
— No son iguales: comprobamos si tiene sinónimos. Preguntamos por el valor del puntero:

1. Cero o espacios: no está.
2. Distinto de cero: puede que esté en el fichero SINONIMOS.
   Los recorremos siguiendo su secuencia y vamos comparando el DNI con el valor introducido:
   - Son iguales: se le da de baja. Se actualizan los punteros del registro anterior, puede estar en DATOS o en SINONIMOS, y el registro cabecera.
   - Hemos pasado por todos y no son iguales: no está.

Sea la situación de los ficheros la siguiente:

DATOS

		Puntero
0		0
6543	Agueda Sánchez	7
0		2
1234	Francisco Bartol	0
8976	José Hernández	0

SINONIMOS

1	7	4		
2	1021	Marisa		6
3		0		
4		3		
5	4563	Luis		0
6	1120	David		0
7	3456	Javier		5

Supongamos que deseamos borrar el DNI 3456, queda la siguiente situación:

DATOS

		Puntero
0		0
6543	Agueda Sánchez	7
0		2
1234	Francisco Bartol	0
8976	José Hernández	0

SINONIMOS

1	6	4		
2	1021	Marisa		6
3		0		
4		3		
5	4563	Luis		0
6	1120	David		0
7	3456	Javier		5

Al ocupar el registro cuya última posición estamos dando de baja en el archivo SINONIMOS, el campo DNI del registro cabecera disminuye en una unidad. La longitud física de SINONIMOS sigue siendo la misma, mientras que la estructura lógica indica que sólo hay seis registros ocupados.

VARIABLES

dn = introducir el DNI.
sw = controla si el DNI está en el archivo SINONIMOS.
ant = guarda la posición del último sinónimo.
aux = almacena la posición actual.
num = almacena la dirección física producida al aplicar el algoritmo doble *dn*.
i = contador para saber si hay que modificar el puntero en DATOS o en SINONIMOS.
res = dar más bajas.

PSEUDOCODIGO

```
res = "S"
abrir ficheros de DATOS y SINONIMOS
hacer mientras res = "S"
        dn = 0
        aux = espacios(2)
        ant = espacios(2)
        sw = 0
        num = 0
        donde = 0
        borrar pantalla
        en 10,20 imprime "D.N.I.:"
        en 10,29 introduce dn
        num = algoritmo sobre dn
```

```
leer de DATOS,num
if DNI = str(dn)
     dni = str(0)
     grabar en DATOS,num
     sw = 1
fin del if
if sw = 0
     if puntero <> 0  AND  puntero <> espacios
          hacer BUSCAR
          if sw = 1
               if i = 0
                    puntero(DATOS,num) = puntero
                    grabar en DATOS,num
               else
                    puntero(SINONIMOS,val(ant)) = puntero
                    grabar en SINONIMOS,val(ant)
               fin del if
               if dni = aux
                    dni = str(val(dni)-1)
                    grabar en SINONIMOS,1
               else
                    dni = str(0)
                    nombre = nombre(SINONIMOS,1)
                    grabar en SINONIMOS,val(aux)
                    nombre (SINONIMOS,1) = aux
                    grabar en SINONIMOS,1
               fin del if
          else
               hacer NO-ESTA
          fin del if
     else
          hacer NO-ESTA
     fin del if
fin del if
hacer MAS
fin del hacer
cerrar ficheros
fin del programa

BUSCAR.
aux = puntero
ant = aux
i = 0
hacer mientras puntero <> 0  AND  sw = 0
     leer de SINONIMOS,val(puntero)
     if dni = str(dn)
```

```
            sw = 1
      else
            i = i + 1
            ant = aux
            if val(puntero) <> 0
                  aux = puntero
            fin del if
      fin del if
fin del hacer

NO-ESTA
en 20,10 imprime "No está"
detener-listado
```

COMPROBACION

Al DNI a dar de baja, se le somete al mismo algoritmo que en el proceso de *altas* dando una dirección física. Comparamos los valores DNI:

- Son iguales: lo borramos de una forma lógica escribiendo en el campo clave un cero.

DATOS.DAT

6543	Agueda Sánchez	7

para borrarlo haremos:

dni = str(0) grabar en DATOS,num

0	Agueda Sánchez	7

El puntero no se modifica, ya que puede tener sinónimos, como sucede en este caso. Aunque el nombre y los demás datos figuran en el registro, no podemos acceder a ellos; dicho de otra forma, esta posición está vacía por ser el campo clave cero.

- No son iguales: el registro no está en DATOS, pero puede estar en SINONIMOS; lo indica el valor del puntero:

0		2	Tiene sinónimos
			Hueco
0		0	No tiene sinónimos

Si este campo tiene algún valor distinto de cero, puede que esté en SINONIMOS, *if puntero < > 0 AND puntero < > espacios.*

Ejecuta BUSCAR mientras no hayamos leído todos los sinónimos y estos sean distintos al buscado, *hacer mientras puntero < > 0 AND sw=0.* Empezamos a leer el registro indicado por el puntero. Si es el buscado, nos salimos asignando al sw el valor uno; en caso contrario, reasignamos las posiciones: la anterior, *ant=aux*, y la actual siempre que haya más sinónimos, *aux=puntero.*

El estado del sw nos indica si lo hemos encontrado o no. Si es:

- Uno: está. Pero el sinónimo puede que esté en el fichero DATOS; lo dice el valor de la variable *i, if i=0:*

  — Es cero: el sinónimo anterior está en DATOS.

Supongamos que deseamos borrar el registro siete de SINONIMOS:

DATOS.DAT

6543	Agueda Sánchez	7

SINONIMOS.DAT

1	7	4		
7	3456	Javier		5

reasignamos el puntero en DATOS:

puntero(DATOS,num)=puntero *grabar en DATOS,num*

6543	Agueda Sánchez	5

— Posee otro valor: está en SINONIMOS.

Supongamos que deseamos borrar el registro cinco de SINONIMOS:

DATOS.DAT

6543	Agueda Sánchez	7

SINONIMOS.DAT

1	7	4		
5	4563	Luis		0
7	3456	Javier		5

En esta situación, *aux=5 y ant=7*, reasignamos el puntero del registro anterior:

puntero(SINONIMOS,val(ant)) = puntero
grabar en SINONIMOS,val(ant)

7	3456	Javier		**0**

Bien se actualice el puntero en un fichero o en otro, el registro a dar de baja está en SINONIMOS. Hay que borrarlo de una forma lógica y actualizar la cabecera. Antes comprobamos si ocupa la última posición, *if dni=aux*.

- Sí: al disminuir en una unidad el número de registros ocupados estamos dándole de baja, *dni=str(val(dni)-1)*.

1	**6**	4		

- No: ocupa una posición intermedia en el fichero. Actualizamos el campo NOMBRE, de los huecos, en el registro que estamos dando de baja:

dni = str(0)
nombre = nombre(SINONIMOS,1)
grabar en SINONIMOS,val(aux)

y el de cabecera:

nombre (SINONIMOS,1) = aux
grabar en SINONIMOS,1.

Es lo que sucede cuando damos de baja al registro cinco de SINONIMOS. El registro en el que se produce la baja queda de la siguiente forma:

5	**0**	**4**		0

y el de información:

1	7	**5**		

- Cero: no está en el fichero.

Ejercicio 3

Hacer un programa que liste por impresora todos los registros del fichero del ejercicio anterior.

ESTUDIO PREVIO

Primero leeremos todos los del fichero DATOS.DAT. Para saber cuántos registros hay los calculamos dividiendo la longitud del fichero por la longitud de un registro. Sólo imprimiremos aquellos registros que poseen información: es decir, si el DNI es cero o blancos, no contiene datos personales:

0		2
0		0

Después listaremos los de SINONIMOS.DAT. El número de registros que hay nos lo dice el campo DNI del registro cabecera, pero alguno de ellos puede estar vacío, como sucedía en DATOS.

VARIABLES

fi = contador de las filas.
pag = contador de las páginas.
n = almacena el número de registros que hay en DATOS.
i = contador de registros tanto para DATOS como para SINONIMOS.

PSEUDOCODIGO

```
abrir ficheros de DATOS y de SINONIMOS
i = 1
n = 0
n = longitud-fichero / longitud-registro
fi = 55
pag = 0
activar impresora
hacer LEER-DATOS
hacer LEER-SINONIMOS
activar pantalla
cerrar fichero
fin del programa
```

```
LEER-DATOS
repetir desde i=1 hasta n
     leer de DATOS,i
     if fi = 55
          hacer CABECERA
          fi = 5
     fin del if
     if val(dni) <> 0  OR  dni <> espacios
          hacer IMPRIMIR
          fi = fi + 1
     fin del if
     i = i + 1
fin del repetir

LEER-SINONIMOS
leer de SINONIMOS,1
n = val(dni)
i = 2
repetir desde i=2 hasta n
     leer de SINONIMOS,i
     if fi = 55
          hacer CABECERA
          fi = 5
     fin del if
     if val(dni) <> 0  OR  dni <> espacios
          hacer IMPRIMIR
          fi = fi + 1
     fin del if
     i = i + 1
fin del repetir
```

COMPROBACION

Calculado el número de registros del fichero (n = longitud del fichero dividido longitud de un registro), leeremos del fichero de DATOS hasta el final, *repetir desde i=1 hasta n.*

El registro sólo se imprime si en el campo DNI no contiene espacios o un cero, *if val(dni) < > 0 OR dni < > espacios.* Se aumenta el número de las filas.

Sea cual sea la información en el registro leído, aumentamos su contador, *i=i+1.*

Leemos del fichero SINONIMOS el campo DNI del registro cabecera para saber el número de registros que hay. Los almacenamos en *n.* Empezamos a leer desde la segunda posición, por ser la primera que realmente almacena datos personales. El resto es igual al subprograma LEER-DATOS.

Los subprogramas CABECERA e IMPRIMIR quedan como ejercicios.

RESUMEN

Los ficheros con **organización directa** o **aleatoria** graban sus registros en las direcciones físicas obtenidas a través de unas transformaciones matemáticas en sus campos indicativos.

La relación campo indicativo–dirección física se obtiene al aplicar al campo indicativo unas fórmulas o algoritmos de transformación.

Los registros se graban en la dirección física resultante de aplicar el algoritmo a su campo clave.

Puede haber huecos entre un registro y otro.

Podemos posicionarnos directamente en el registro deseado sin necesidad de tener que pasar por todas las posiciones físicas anteriores.

Los ficheros directos deben estar grabados en soportes direccionables.

El algoritmo de transformación de claves debe cumplir una serie de condiciones:

- Relacionar campo con su dirección física en el soporte de datos.
- Producir el menor número posible de sinónimos.
- Dejar el menor número de huecos: direcciones físicas vacías.

Sinónimos: distintos registros a los que les corresponde la misma dirección física.

Para leer un determinado registro hay que someter al campo indicativo la misma fórmula o algoritmo utilizado en el proceso de su grabación.

Los campos son todos alfanuméricos.

Los métodos de transformación de claves más utilizados son:

- **Direccionamiento directo:** se emplea cuando los campos indicativos son consecutivos o tienen muy pocos huecos. No producen sinónimos.
- **Direccionamiento por conversión a octal:** consiste en asignar a cada dígito del campo indicativo su valor en octal. Posteriormente se suman todos los valores. Produce sinónimos.
- **Direccionamiento por el centro al cuadrado:** se eleva el campo indicativo al cuadrado y se eliminan las cifras extremas, tanto de la derecha como de la izquierda. Produce sinónimos.

Cada usuario puede diseñar su propia estructura para los ficheros con organización directa.

Puntero: es un campo o una variable con la dirección física en la que se encuentran grabado el siguiente dato.

Las instrucciones de los ficheros con organización directa son:

- Leer un registro: *leer nombre-fichero, número-de-registro.*
- Grabar un registro: *grabar nombre-fichero, número-de-registro.*
- Hallar la longitud del fichero: *longitud nombre-fichero.*

Las operaciones de abrir y cerrar ficheros son análogas a las de los ficheros con organización secuencial.

EJERCICIOS PROPUESTOS

1. Consultar un determinado registro del fichero DATOS (que lleva asociado el archivo SINONIMOS).

2. Hacer un pseudocódigo que imprima por pantalla los registros del fichero anterior ordenados por orden alfabético.

3. Sea el fichero ARTICULOS.DAT con organización directa, cuya estructura es la siguiente:

NART	CONCEPTO	STOCK	STOCK-MINIMO	PRECIO	PUNTERO

donde PUNTERO indica la posición donde están almacenados los sinónimos de NART en el fichero con organización directa SINONIART.DAT.

Hacer un programa que liste por impresora:

— Todos los registros.
— Los artículos cuyo stock sea menor que el stock mínimo.

4. En el fichero con organización directa ALUMNOS.DAT, se guardan los datos personales de los alumnos de un determinado colegio, cuya estructura es:

NALUMNO	NOMBRE	DIRECCION	TELEFONO	CURSO	PUNTERO

donde el campo CURSO es el curso en el que está matriculado el alumno, y PUNTERO la posición donde están almacenados los sinónimos de NALUMNO en el fichero SINONIALUMNOS.DAT.

En el fichero con organización secuencial CURSOS.DAT, están almacenados los distintos cursos que imparte el colegio, así como las asignaturas de cada curso, cuyos campos son:

CURSO2	ASIG1	ASIG2	ASIG3	ASIG4	ASIG5

El fichero secuencial NOTAS.DAT guarda las notas, en valor numérico, de cada alumno, con la estructura:

NALUM	A1	A2	A3	A4	A5

Por último, en el fichero directo PENDIENTES.DAT, almacena los datos de los alumnos que poseen asignaturas pendientes con la estructura:

NALUPEN	PENDI1	NOTA1	PENDI2	NOTA2	PUNTERO

donde PENDI1 y PENDI2 son los nombres de las asignaturas y PUNTERO la posición donde están los sinónimos NALUPEN en el fichero SINONIMOPENDIENTES.DAT.

Se desea hacer un programa con las siguientes opciones:

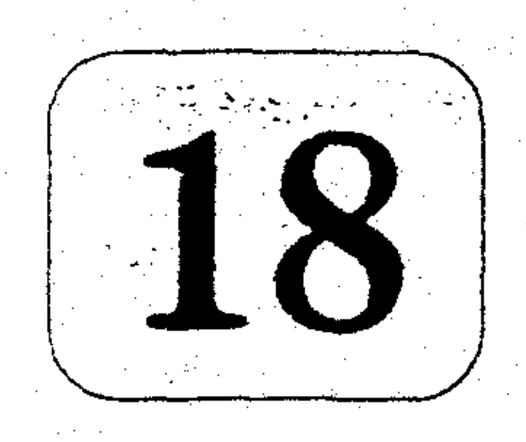
— Introducir la nota, en número, a todos los alumnos de un curso.
— Listado de todos los alumnos de un curso.
— Listado de notas, que tiene las siguientes opciones:
  1. Notas de un alumno determinado.
  2. Notas de todos los alumnos de un curso determinado.
  3. Notas de todos los alumnos del colegio.

El formato de las notas será el siguiente:

Curso Académico: ----/--	
Nombre: ---------- Curso: ------	
Asignaturas	Calificación
---------- ----------	---------- ----------
Nota media:	------

Tanto la calificación como la nota media deben aparecer en letra, es decir: Muy Deficiente, Insuficiente, Aprobado, etc.

CAPITULO



Ficheros con organización secuencial indexada

18.1. INTRODUCCION

Un fichero con organización **secuencial indexada** o, simplemente, **indexado** consta de:

- **Un área de datos:** en ella están almacenados los registros, con todos sus campos, formados durante la creación del fichero de datos.

  Está constituida por conjuntos de registros separados en grupos. Se comporta como un fichero secuencial.
- **Un área o fichero de índices:** organizada de forma secuencial. Puede ser un archivo independiente ordenado por claves.

  Cada registro del área de índices contiene la clave del último registro de cada grupo del área de datos, y la dirección relativa del primer registro del grupo.
- **Un área de overflow o de excedentes:** contiene los registros que no caben en el área de datos cuando se produce una actualización del archivo.

  Las claves de estos registros son intermedias a las existentes en su grupo y, por tanto, no se pueden añadir al grupo correspondiente. Cuando se crea el fichero esta área está vacía.

Cuando deseamos localizar un registro por su clave, primero realiza la búsqueda en la tabla de índices, hasta encontrar la primera clave superior a la dada. Junto a esta clave, se encuentra la dirección inicial del grupo de registros que debe contener el registro a localizar.

El ordenador se posiciona directamente en esa dirección del área de datos, efectuando una búsqueda secuencial del registro por su clave.

Si no se encuentra en dicha área, el puntero del fichero se coloca en el primer registro correspondiente al área de overflow y en forma secuencial, lee los registros y va comparando los campos clave, de todos y cada uno de los registros, con la que deseamos encontrar. Si la halla, se detiene en ese registro; en caso contrario, el puntero se posicionará al final del área de excedentes.

Supongamos que tenemos el fichero NOMBRES.DAT, que está indexado por el campo DNI, cuya estructura es:

DNI	NOMBRE	APELLIDOS

con los siguientes registros, tal como se indica a continuación:

Area de INDICES

20	d1	100	d2	200	d3	. .	d4	. .	d5	. .	d6	900	d7

Area de DATOS

d1	1	Julio Calvo	2	Emilio García	20	Sara González
d2	21	Ana B. Herrera	22	Javier S. José	100	Patricia Rico
d3	101	David Rodríguez	175	José L. Robles	200	Ana Martínez
d4	. . .		. . .		. . .	
d5	. . .		. . .		. . .	
d6	. . .		. . .		. . .	
d7	884	Amparo Calvo	885	Macario Gómez	900	Julia Díez

Area de OVERFLOW

95	Marta	109	Felipe	88	Raquel	---->

En el área de índices los números representan el valor del campo clave mayor de cada uno de los registros del grupo del área de datos y la *d*, seguida de un número, la dirección física en la que se encuentra dicho grupo de registros en el área de datos.

En el área de datos los números indican los campos clave de cada uno de los registros. De igual forma sucede en el área de overflow.

Ejemplo: si deseamos buscar el registro cuyo campo clave es el 88, el ordenador busca primero en el área de índices un valor del campo clave mayor o igual al buscado, en nuestro caso el 100. Al mismo tiempo lee la dirección del grupo en la que se debe encontrar dicho registro, la *d2*. Se posiciona directamente en dicha dirección y comienza la lectura secuencial. Al no encontrarse en dicho grupo, en el cual están los registros cuyos campos clave corresponden a 21, 22 y 100, comienza a leer en el área de overflow hasta encontrarlo.

Esta organización presenta la **ventaja** de un rápido acceso a los datos de una forma lógica, ordenada por el campo en el que está indexado el fichero.

Los lenguajes que trabajan con este tipo de organización secuencial indexada poseen las instrucciones necesarias para construir este tipo de archivos de una forma automática.

El sistema operativo también se encarga de las áreas de índices y de excedentes.

Presenta el inconveniente de que necesita reservar un espacio adicional para el área de índices. Si por algún motivo se apaga el ordenador estando activado el fichero o el área de índices, al estar presente siempre en la memoria del ordenador por comportarse como una tabla, hay que reestructurar, indexar de nuevo el fichero de índices, para que vuelva a haber una correspondencia entre el fichero o área de datos y el de índices.

Frente a los ficheros secuenciales presenta la ventaja de realizar la búsqueda de una forma mucho más rápida y una mayor versatilidad a la hora de efectuar las modificaciones.

En grandes archivos se suele utilizar el método de árboles binarios para realizar la búsqueda en el fichero de índices.

18.2. INSTRUCCIONES DE LOS FICHEROS INDEXADOS

Un fichero con organización indexada puede tener asociado a su área de datos más de un fichero de índices.

Cuantos más ficheros de índices posea, por más campos podemos realizar la búsqueda de una forma directa, pero va a ocupar más espacio en el disco.

Al abrir el fichero indexado le indicaremos el área de índices que tiene que estar asociada al fichero de datos:

abrir nombre–del–fichero asociado al campo–clave–1, campo–clave2, ...

Si le asociamos todos los ficheros de índices, sólo podemos acceder de una forma directa por el primer campo clave especificado, pero las modificaciones realizadas en el fichero de datos automáticamente quedan reflejadas en todos los ficheros de índices; es decir, todos ellos se actualizan.

Las operaciones de lectura pueden ser de dos formas distintas:

- Tratar al fichero indexado como si fuera secuencial. Emplearemos el mismo formato que en los secuenciales:

  leer siguiente registro

- Podemos buscar un registro determinado; en este caso, preguntaremos por el campo clave, es decir, por el campo que está indexado:

  busca variable

  en *variable* está el valor del campo clave a buscar.

  De esta forma le indicamos que lo busque, pero no sabemos si lo ha encontrado o no. Para ello le indicamos:

  if found()

  si lo ha encontrado o no, nos devuelve un valor lógico:

  — TRUE (verdad): lo ha encontrado. Podemos visualizarlo, modificarlo o borrarlo.
  — FALSE (falso): no lo ha encontrado.

Algunos lenguajes, como el COBOL, preguntan si la clave es inválida:

— Lo es: el registro no está.
— No lo es: el registro está en el fichero.

Resumiendo. En un fichero con organización secuencial podemos acceder directamente a un registro en particular o bien tratarlo secuencialmente desde el primer registro.

También se puede tratar de una forma *dinámica:* posicionarnos directamente en un registro y a partir de él hacer una lectura secuencial hasta el final del fichero o mientras se siga cumpliendo alguna condición.

Las operaciones de escritura y regrabar son análogas a las de los ficheros con organización secuencial.

Para cerrar ficheros, además de cerrar el fichero de datos, como si de un secuencial se tratara, hay que cerrar el fichero de índices. Sintaxis:

cerrar índices.

Ejercicio 1

Tenemos el fichero DATOS.DAT, que está indexado por el campo APELLIDOS, cuyos campos son:

DNI	NOMBRE	APELLIDOS	DIRECCION	PROVINCIA

Hacer un programa que nos permita listar, por pantalla, todos los registros del fichero, controlando el salto de página cuando llegue a la línea veinte.

ESTUDIO PREVIO

Leeremos el fichero como si se tratara de un fichero secuencial desde el primer registro al último.

Al estar indexado por el campo APELLIDOS el listado sale ordenado alfabéticamente por apellidos.

VARIABLE

fi = contador de las filas.

PSEUDOCODIGO

```
abrir DATOS asociado a APELLIDOS
fi = 22
hacer mientras no sea EOF()
     if fi = 22
          hacer CABECERA
     fin del if
```

```
en fi,2 imprime dni
en fi,12 imprime nombre
en fi,28 imprime apellidos
en fi,55 imprime direccion
en fi,69 imprime provincia
fi = fi + 1
if fi = 20
     en 22,20 imprime "Pulse RETURN para continuar"
     detener-listado
     fi = 22
     fin del if
     leer siguiente registro
fin del hacer
cerrar DATOS
cerrar indices
fin del programa.

CABECERA.
borrar pantalla
en 3,4 imprime "D.N.I."
en 3,20 imprime "NOMBRE"
en 3,35 imprime "APELLIDOS"
en 3,60 imprime "DIRECCION"
en 3,70 imprime "PROVINCIA"
fi = 5.
```

COMPROBACION

Se trata del mismo pseudocódigo que en la organización secuencial. Todo lo dicho en aquella ocasión es válido ahora. La diferencia estriba en que accedemos a los datos no como los grabamos, sino de una forma lógica, ordenada.

Ejercicio 2

Tenemos el fichero DATOS.DAT que está indexado por el campo DNI y tiene la misma estructura del ejercicio anterior.

Crear un programa que nos permita consultar un registro siempre que queramos.

ESTUDIO PREVIO

Para buscar un determinado registro se introduce por teclado el campo clave: en caso de no hallarlo indicaremos que ese dato no está.

VARIABLES

num = introducir el DNI a buscar.
res = para buscar más datos.

PSEUDOCODIGO

```
res = "S"
abrir DATOS asociado a DNI
hacer mientras res = "S"
     num = 0
     borrar pantalla
     en 8,20 imprime "D.N.I. a buscar:"
     en 8,38 introduce num
     busca num
     if found()
          en 10,12 imprime "Nombre:",nombre
          en 12,28 imprime "Apellidos:",apellidos
          en 14,55 imprime "Dirección:",direccion
          en 16,69 imprime "Provincia:",provincia
     else
          en 12,20 imprime "No está"
          en 16,20 imprime "Pulse RETURN para continuar
          detener-listado
     fin del if
     res = espacios(1)
     hacer MAS
fin del hacer
cerrar DATOS
cerrar indices
fin del programa.
```

COMPROBACION

Antes del ciclo, *hacer mientras res="S"*, abrimos el fichero. La búsqueda la va a realizar siempre y de una forma automática desde el primer registro.

Introducimos por teclado el valor del DNI a consultar y le indicamos que lo busque, *busca num*. Le preguntamos si lo ha hallado, *if found():*

— Sí: imprimimos el registro.
— No: visualiza por pantalla el comentario "No está".

Ejecuta el subprograma MAS. Cuando no deseemos buscar más datos cerramos el fichero.

Ejercicio 3

Tenemos el fichero DATOS.DAT, que está indexado por un lado por el campo DNI y por otro por el campo PROVINCIA, cuya estructura es la misma que la del ejercicio anterior.

Listar por impresora todos los registros cuya provincia sea una determinada que introduciremos por teclado.

El formato de impresión será el siguiente:

Pag: --			
Relación de las personas que viven en la provincia: ------			
D.N.I.	Nombre	Apellidos	Dirección
----	----------	----------	----------
----	----------	----------	----------
----	----------	----------	----------

Cambio de página cuando lleguemos a la línea 55.

Al final del listado imprimir el número de personas que habitan en esa provincia.

ESTUDIO PREVIO

Asociamos al fichero de datos el área de índices adecuado, la PROVINCIA.

Todos los registros con el mismo valor de la provincia van a estar seguidos.

Una vez localizada la provincia que nos interesa, realizaremos un ciclo de lectura secuencial controlando el final del fichero, para imprimir todos los registros de esa provincia.

VARIABLES

pro = introducir la provincia a buscar.
c = contador de las personas que viven en esa provincia.
fi = contador de las filas.
pag = contador del número de páginas.

PSEUDOCODIGO

```
borrar pantalla
fi = 55
c = 0
pag = 1
en 10,20 imprime "Provincia:"
en 10,32 introduce pro
abrir DATOS asociado a PROVINCIA
buscar pro
```

```
if found()
     activar impresora
     hacer mientras provincia = pro  AND  no sea EOF()
          if fi = 55
               hacer CABECERA
          fin del if
          en fi,5 imprime dni
          en fi,15 imprime nombre
          en fi,35 imprime apellidos
          en fi,65 imprime direccion
          fi = fi + 1
          c = c + 1
          leer siguiente registro
     fin del hacer
     en fi+2,20 imprime "Total de personas:",c
     activar pantalla
else
     en 18,20 imprime "No hay datos con esa provincia"
     detener-listado
fin del if
cerrar DATOS
cerrar indices
fin del programa.

CABECERA.
salto de pagina
en 2,65 imprime "Pág:",pag
en 4,10 imprime "Relación de las personas que viven en la provincia:",pro
en 6,7 imprime "DNI"
en 6,18 imprime "Nombre"
en 6,40 imprime "Apellidos"
en 6,68 imprime "Dirección"
en 7,4 imprime "____________________________________________________"
fi = 9
pag = pag + 1.
```

COMPROBACION

Abrimos el fichero de índices y lo relacionamos con el de datos.

Introducimos la provincia a buscar, *busca pro.* Comprobamos si la ha encontrado, *if found():*

- Si: activamos la impresora, pues en caso de no encontrar registros pertenecientes a esa provincia no tiene que imprimir nada por la impresora.

  A partir de este registro debe leer secuencialmente mientras la provincia no cambie y mientras no sea final del fichero, *hacer mientras provincia=pro AND no sea*

EOF(). Realiza la CABECERA, imprime el registro y accedemos al siguiente registro, *leer siguiente registro*.

Cuando termine de sacar todos los registros de esa provincia o haya llegado al final del fichero activamos la pantalla.

- No: imprime por pantalla el comentario que no ha encontrado esa provincia.

Por último cerramos toda clase de ficheros.

Ejercicio 4

Hacer un programa que nos permita dar altas en el fichero DATOS.DAT, que está indexado por los campos DNI, APELLIDOS y PROVINCIA, controlando las altas duplicadas.

ESTUDIO PREVIO

Vamos a dar altas; por tanto, va a cambiar el fichero de datos y todos los ficheros de índices deben recoger esa modificación.

Buscaremos por el campo clave que no se puede repetir, el DNI. Si no está ese DNI, entonces introduciremos los demás campos.

Para dar altas en el fichero de datos, antes de grabar situaremos el puntero al final del mismo.

VARIABLES

num = para introducir el dni del nuevo registro.
res = para seguir introduciendo más registros.

PSEUDOCODIGO

```
res = "S"
abrir DATOS asociado a DNI,APELLIDOS,PROVINCIA
hacer mientras res = "S"
     borrar pantalla
     en 5,10 imprime "D.N.I.:"
     en 5,18 introduce num
     buscar num
     if found()
          en 10,10 imprime "Alta duplicada"
          en 15,10 imprime "Pulse RETURN para continuar"
          detener-listado
     else
          en 7,5 imprime "Nombre:"
          en 9,5 imprime "Apellidos:"
          en 11,5 imprime "Dirección:"
```

```
            en 13,5 imprime "Provincia:"
            en 7,16 introduce nombre
            en 9,16 introduce apellidos
            en 11,16 introduce direccion
            en 13,16 introduce provincia
            dni = num
            posicionarse al final del fichero de DATOS
            escribir registro
        fin del if
        res = espacio(1)
        hacer MAS
    fin del hacer
    cerrar fichero de DATOS
    cerrar ficheros de indices
    fin del programa.
```

COMPROBACION

Abrimos el fichero de DATOS y asociado a él van a estar todos los ficheros de índices por los cuales está indexado, pero sólo podemos hacer la búsqueda indexada por el primer índice.

Introducimos el valor del campo indicativo y busca indexadamente en el fichero de índice DNI. Comprobamos si está o no, *if found()*.

- Lo encontró: es alta duplicada. No hay que introducir más datos.
- No lo encontró: no está, introducimos el resto de los datos. Nos situamos al final del área de datos y grabamos el registro.

Al estar activados todos los ficheros de índices, quedan actualizados todos ellos.

Ejercicio 5

Hacer un programa que nos permita dar bajas en el fichero DATOS.DAT, que está indexado por los campos DNI, APELLIDOS y PROVINCIA, controlando las bajas inexistentes.

ESTUDIO PREVIO

El proceso es análogo al de las altas. Hay que activar al fichero de datos todos los ficheros de índices que recojan las posibles bajas, queden actualizados.

VARIABLES

num = para introducir el dni a dar de baja.
res = para seguir introduciendo más registros.

PSEUDOCODIGO

```
res = "S"
abrir DATOS asociado a DNI,APELLIDOS,PROVINCIA
hacer mientras res = "S"
     borrar pantalla
     en 5,10 imprime "D.N.I.:"
     en 5,18 introduce num
     buscar num
     if found()
          en 7,5 imprime "Nombre:"
          en 9,5 imprime "Apellidos:"
          en 11,5 imprime "Dirección:"
          en 13,5 imprime "Provincia:"
          en 7,16 introduce nombre
          en 9,16 introduce apellidos
          en 11,16 introduce direccion
          en 13,16 introduce provincia
          en 20,20 imprime "Deseas borrarlo (S/N): "
          res = espacio(1)
          hacer mientras res <> "S"  AND  res <> "N"
               res = espacios(1)
               en 20,44 introduce res
          fin del hacer
          if res = "S" or res = "s"
               borrar registro
          fin del if
     else
          en 12,10 imprime "Baja inexistente"
          en 16,10 imprime "Pulse RETURN para continuar"
          detener-listado
     fin del if
     res = espacio(1)
     hacer MAS
fin del hacer
cerrar fichero de DATOS
cerrar fichero de indices
fin del programa.
```

COMPROBACION

Activamos todos los ficheros de índices.
Introducimos el valor del campo clave y comprobamos si está o no, *if found():*

- Está: visualizamos el valor del registro y preguntamos si lo deseamos borrar. Si se desea borrar, se borra.
- No está: baja inexistente.

Ejercicio 6

Dado el fichero DATOS.DAT, que está indexado por los campos DNI, APELLIDOS y PROVINCIA, hacer un programa que nos permita realizar modificaciones por las siguientes opciones:

— por el DNI.
— por los apellidos.

ESTUDIO PREVIO

Nos construiremos primero el menú para elegir por qué campo deseamos hacer las modificaciones.

Según la opción elegida introduciremos en una variable el dato a buscar y haremos las modificaciones oportunas.

Todos los ficheros indexados deben estar activados en cada una de las opciones para que queden actualizados.

VARIABLES

op = elegir la opción deseada.
num = dni del registro a modificar.
nom = introducir el nuevo valor del nombre.
ape = introducir el nuevo valor de los apellidos y buscar el registro.
dir = introducir la nueva dirección.
pro = introducir la nueva provincia.
res = modificar más registros.

PSEUDOCODIGO

```
op = 0
hacer mientras op <> 3
     op = 0
     borrar pantalla
     en 8,20 imprime "Menú de modificaciones"
     en 12,20 imprime "DNI"
     en 14,20 imprime "APELLIDOS"
     en 16,20 imprime "SALIR"
     en 20,20 imprime "Elige opción:"
     en 20,35 introduce op
     hacer case
          case op = 1
               hacer DNI
          case op = 2
               hacer APELLIDOS
     fin del case
fin del hacer
fin del programa.
```

```
DNI.
res = "S"
abrir DATOS asociado a DNI,APELLIDOS,DIRECCION
hacer mientras res = "S"
     num = 0
     borrar pantalla
     en 5,10 imprime "D.N.I.:"
     en 5,18 introduce num
     busca num
     if found()
          hacer IMPRIMIR
          hacer CAMBIOS
     else
          hacer DETENER
     fin del if
     res = espacio(1)
     hacer MAS
fin del hacer
cerrar DATOS
cerrar ficheros de indices
fin del hacer

APELLIDOS.
res = "S"
abrir DATOS asociado a APELLIDOS,DNI,DIRECCION
hacer mientras res = "S"
     ape = espacios(30)
     borrar pantalla
     en 9,5 imprime "Apellidos:"
     en 9,16 introduce ape
     busca ape
     if found()
          en 5,10 imprime "D.N.I.:"
          en 5,18 imprime dni
          hacer IMPRIMIR
          hacer CAMBIOS
     else
          hacer DETENER
     fin del if
     res = espacio(1)
     hacer MAS
fin del hacer
cerrar DATOS
cerrar ficheros de indices
fin del hacer
```

```
IMPRIMIR.
en 7,5 imprime "Nombre:"
en 9,5 imprime "Apellidos:"
en 11,5 imprime "Dirección:"
en 13,5 imprime "Provincia:"
en 7,16 imprime nombre
en 9,16 imprime apellidos
en 11,16 imprime direccion
en 13,16 imprime provincia.

CAMBIOS.
nom = nombre
ape = apellidos
dir = direccion
pro = provincia
en 7,16 introduce nom
en 9,16 introduce ape
en 11,16 introduce dir
en 13,16 introduce pro
nombre = nom
apellidos = ape
direccion = dir
provincia = pro
regrabar registro.

DETENER.
en 10,20 imprime "Registro inexistente"
en 20,18 imprime "Pulse RETURN para continuar"
detener-listado.
```

COMPROBACION

Una vez realizado el menú introducimos la opción deseada.

Si le indicamos la primera, realiza el subprograma DNI. Abrimos el fichero de datos y los de índices. Introducimos el DNI del registro a modificar. Si lo encuentra en el subprograma IMPRIMIR, visualiza los datos de ese registro.

Ejecuta el subprograma de CAMBIOS, en el cual le introducimos los nuevos valores y regrabamos el registro.

El campo clave DNI no lo modificamos porque al darlo de alta se comprobó que es único. En el caso de habernos equivocado al introducirlo y deseáramos modificarlo, le daríamos de baja y luego de alta con el valor deseado. Cuando no deseemos modificar más registros cerramos todos los ficheros.

La opción segunda, modificar por los apellidos. Volvemos a abrir el fichero de datos y de índices. Introducimos los apellidos. Para poder utilizar el subprograma IMPRIMIR visualizamos el DNI y ejecuta el subprograma CAMBIOS. Aquí tampoco modificamos el DNI, por ser una clave única.

Para salir damos la opción tres.

Ejercicio 7

Se desea actualizar el fichero DATOS.DAT que está indexado por los campos DNI, APELLIDOS y DIRECCION mediante el fichero secuencial CAMBIOS.DAT, cuyos campos son:

DNI2	NOMBRE2	APELLI2	DIRE2	PROVIN2	TO

donde TO indica el tipo de operación a realizar y puede tener los valores:

— A: alta.
— B: baja.
— M: modificación.

ESTUDIO PREVIO

La actualización del fichero de DATOS se realiza sobre el mismo fichero.

Leeremos el fichero CAMBIOS secuencialmente desde el primer registro hasta el último. Buscamos en el archivo DATOS el valor de DNI2 correspondiente al fichero secuencial. Puede suceder:

— No está, pero si el tipo de operación es:
  1. A: grabamos el registro en el archivo DATOS.
  2. B: es una baja inexistente; no podemos darle de baja.
  3. M: es una modificación inexistente; tampoco podemos modificar el registro.
— Está el registro, pero si el tipo de operación es:
  1. A: es un alta duplicada y no haremos nada.
  2. B: lo borramos del archivo de DATOS.
  3. M: regrabamos el registro en el fichero de DATOS.

Cuando se haya acabado el fichero de CAMBIOS lo borramos.

PSEUDOCODIGO

```
abrir DATOS asociado a DNI,APELLIDOS,PROVINCIA
abrir CAMBIOS
hacer mientras no sea EOF() de CAMBIOS
     buscar dni2 en DATOS
     if found()
          if to = "M"
               regrabar registro en DATOS
          else
               if to = "B"
                    borrar registro de DATOS
               fin del if
          fin del if
     else
```

```
            if to = "A"
                posicionarse al final de DATOS
                grabar registro en DATOS
            fin del if
        fin del if
        leer siguiente registro de CAMBIOS
fin del hacer
cerrar todos los ficheros de datos
cerrar fichero de indices
borrar fichero CAMBIOS
fin del programa.
```

COMPROBACION

Una vez abiertos todos los ficheros, leemos del de movimientos hasta su final, *hacer mientras no sea EOF() de CAMBIOS.* El DNI que acaba de leer lo busca en DATOS. Si lo encuentra y la operación es:

— M: regrabamos el registro en DATOS.
— B: borramos registro en DATOS.

No hace falta preguntar por la A, ya que no hay que realizar ninguna operación con ese registro; además se están controlando la *M* y la *B*.

En caso de no encontrar al DNI, no está. Lo daremos de alta sólo cuando el tipo de operación sea A. Para *B* y para *M* no se realiza ninguna operación con el registro.

Cuando se hayan ejecutado todos los registros del archivo CAMBIOS, cerramos toda clase de fichero y borramos CAMBIOS.

Ejercicio 8

Tenemos el fichero ARTICULOS.DAT, que está indexado por el campo NART, cuya estructura es:

NART	ARTICULO	PVP	STOCK	MINIMO

donde MINIMO es el stock mínimo de ese artículo.

En otro fichero secuencial, VENTAS, están almacenadas las modificaciones de los productos durante el día, cuyos campos son:

NART2	VENTAS	TIPO

el campo TIPO puede tomar los valores:

— Cero: es una venta.
— Uno: compra del producto.

Puede haber más de un registro para un mismo artículo.
Se desea hacer un programa que realice:

1. Actualización del fichero de ARTICULOS.
2. Un listado por impresora de las entradas y salidas de los artículos, de tal forma que cada producto sólo aparezca una vez, con el siguiente formato:

Pag: --

LISTADO DE LAS ENTRADAS DE ARTICULOS AL DIA: ---

Número	Artículo	Cantidad
--- ---	---------- ----------	---- ----

Pag: --

LISTADO DE SALIDAS DE ARTICULOS AL DIA: ---

Número	Artículo	Cantidad	PVP	Importe
--- * ---	---------- ----------	---- ----	-- --	---- ----
			Total:	------

Si en algún artículo su stock es inferior al mínimo, en el listado de ventas, delante del número de artículo, se imprimirá un asterisco.

Cuando llegue a la línea 55 se producirá un salto de página.

Ambos ficheros están ordenados por número de artículo.

ESTUDIO PREVIO

El enunciado es el mismo que cuando se estudiaron los ficheros con organización secuencial. La diferencia estriba en el fichero de ARTICULOS, que ahora está indexado.

Vamos a procesar el fichero de VENTAS. Empezaremos a leer de él. Para controlar si de un mismo concepto hay más de un registro hay que utilizar un sw y un auxiliar.

Al mismo tiempo que vamos imprimiendo, actualizamos el STOCK del fichero de ARTICULOS.

El proceso de los listados es igual que el ejercicio de los ficheros secuenciales. Nos crearemos un fichero secuencial SALIDAS, donde grabaremos los artículos vendidos que imprimiremos una vez haya terminado el listado de las compras.

La única diferencia respecto al ejercicio mencionado radica en la rapidez con la que buscamos los datos de un artículo en el archivo indexado.

VARIABLES

SALIDAS = fichero con las salidas del almacén. Sus campos son NART3 y VENTAS3.
aux = guardar el NART.
sw = cambiar el valor de *aux*.
entra = acumulador de las entradas de un producto.
sale = acumulador de las salidas de un producto.
total = acumulador de las ventas de todos los artículos.
pag = contador de las páginas.
fi = contador de las filas.

PSEUDOCODIGO

```
entra = 0
sale = 0
total = 0
fi = 55
sw = 0
aux = 0
pag = 1
activar impresora
abrir ARTICULOS asociado a DNI
abrir VENTAS
abrir SALIDAS
hacer mientras no sea EOF() de VENTAS
    if fi = 55
          hacer CABECOMPRAS
    fin del if
    if sw = 0
          aux = nart2
          busca aux en el fichero indice
          sw = 1
    fin del if
    if nart2 = aux
          hacer CALCULOS
    else
          hacer GRABAR
          hacer COMPRA          * imprimir los articulos que entran
          entra = 0
          sale = 0
          aux = nart2
          busca aux en el fichero indice
          hacer CALCULOS
    fin del if
    leer siguiente registro de VENTAS
fin del hacer
hacer GRABAR
```

```
hacer COMPRA
posicionarse en el primer registro de SALIDAS
fi = 55
hacer mientras no sea EOF() de SALIDAS
     if fi = 55
          hacer CABESAL
     fin del if
     buscar nart3 en el fichero indice
     hacer SALE                *imprimir los articulos vendidos
     leer siguiente registro del fichero SALIDAS
fin del hacer
en fi+4,55 imprime "Total:",total
activar pantalla
cerrar todos los ficheros de datos
cerrar fichero de indice
borrar el fichero SALIDAS
fin del programa.


CALCULOS.
if tipo = 0
      entra = entra + ventas
else
      sale = sale + ventas
fin del if


GRABAR.
stock = stock + entra - sale
regrabar registro en ARTICULOS
nart3 = aux
ventas3 = sale
posicionarse al final del fichero SALIDAS
grabar registro en SALIDAS


CABECOMPRAS.
salto de pagina
en 2,65 imprime "Pág:",pag
en 4,20 imprime "LISTADO DE LAS ENTRADAS DE ARTICULOS AL DIA:"
en 4,66 imprime fecha del sistema
en 6,18 imprime "Número"
en 6,35 imprime "Artículo"
en 6,65 imprime "Cantidad"
en 7,15 imprime "_____________________________________________"
fi = 9
pag = pag + 1
```

```
COMPRA.
en fi,16 imprime aux
en fi,30 imprime articulo
en fi,67 imprime entra
fi = fi + 1

CABESAL.
salto de pagina
en 2,65 imprime "Pág:",pag
en 4,20 imprime "LISTADO DE SALIDAS DE ARTICULOS AL DIA:"
en 4,60 imprime fecha del sistema
en 6,8 imprime "Número"
en 6,16 imprime "Artículo"
en 6,40 imprime "Cantidad"
en 6,54 imprime "PVP"
en 6,64 imprime "Importe"
en 7,6 imprime "____________________________________________"
fi = 9
pag = pag + 1

SALE.
salto de pagina
if stock < minimo
     en fi,4 imprime "*"
fin del if
en fi,6 imprime nart3
en fi,14 imprime articulo
en fi,40 imprime ventas
en fi,54 imprime pvp
en fi,65 imprime ventas*pvp
total = total + ventas*pvp
fi = fi + 1
```

COMPROBACION

La única diferencia respecto al ejercicio citado radica en que no es necesario realizar el subprograma BUSCAR; al tratarse de un fichero indexado se posiciona en ese registro.

Tampoco controlamos si está o no está, ya que en el proceso de creación del fichero de VENTAS no grabaremos ningún registro hasta que no haya encontrado en el fichero ARTICULO el número de artículo introducido por teclado (realizarlo como ejercicio).

RESUMEN

Un fichero con organización **secuencial indexada** o simplemente **indexado** consta de:

- **Un área de datos:** en ella están almacenados los registros, con todos sus campos, formados durante la creación del fichero de datos.

  Los registros están agrupados en grupos. Se comporta como un fichero secuencial.
- **Un área de índices:** organizada de forma secuencial. Puede ser un archivo independiente ordenado por claves.

  Cada registro contiene la clave del último registro de cada grupo del área de datos y la dirección relativa del primer registro del grupo.
- **Un área de overflow o de excedentes:** contiene los registros que no caben en el área de datos cuando se produce una actualización del archivo.

  Las claves de estos registros son intermedias a las existentes en su grupo y, por tanto, no se pueden añadir al grupo correspondiente. Cuando se crea el fichero esta área está vacía.

El ordenador se encarga de gestionar las áreas de índices y de excedentes.

Presenta el inconveniente de tener la necesidad de reservar un espacio adicional para el área de índices.

Tienen la ventaja de realizar la búsqueda de un modo rápido.

Un fichero con organización indexada puede tener asociado a su área de datos más de un fichero de índices.

Cuantos más ficheros de índices posea, por más campos podemos realizar la búsqueda de una forma rápida, pero también va a ocupar más espacio en el disco.

Al abrir el fichero le indicamos el área de índices que tiene que estar asociada:

abrir nombre-del-fichero asociado al campo-clave-1, campo-clave-2, ...

Sólo podemos acceder de una forma directa por el primer campo clave especificado. Las modificaciones realizadas en el fichero de datos, automáticamente, quedan reflejadas en todos los ficheros índices.

Se puede tratar al fichero indexado como si fuera secuencial: en este caso, para leer el siguiente registro lo expresamos:

leer siguiente registro

EJERCICIOS PROPUESTOS

1. Del fichero de ARTICULOS.DAT, que está indexado por el campo NART, sacar un listado por impresora de todos los artículos cuyo stock sea menor que el stock mínimo.
2. En el fichero ALUMNOS.DAT, que está indexado por el campo NALUMNO, se guardan los datos personales de los alumnos de un determinado colegio, cuya estructura es:

NALUMNO	NOMBRE	DIRECCION	TELEFONO	CURSO

donde el campo CURSO es el curso en el que está matriculado.

Por otro lado, en el fichero CURSOS.DAT, indexado por CURSO2, están almacenados los distintos cursos que imparte el colegio, así como las asignaturas de cada curso, cuyos campos son:

CURSO2	ASIG1	ASIG2	ASIG3	ASIG4	ASIG5

En el fichero NOTAS.DAT, indexado por NALUM, se guardan las notas, en valor numérico, de cada alumno, con la estructura:

NALUM	A1	A2	A3	A4	A5

Por último, el fichero PENDIENTES.DAT, indexado por NALUPEN, almacena los datos de los alumnos que poseen asignaturas pendientes con la estructura:

NALUPEN	PENDI1	NOTA1	PENDI2	NOTA2

donde PENDI1 y PENDI2 son los nombres de las asignaturas.

Se desea hacer un programa con las siguientes opciones:

— Introducir la nota, en número, a todos los alumnos de un curso.
— Listado de todos los alumnos de un curso.
— Listado de notas, que tiene las siguientes opciones:
  1. Notas de un alumno determinado.
  2. Notas de todos los alumnos de un curso determinado.
  3. Notas de todos los alumnos del colegio.

El formato de las notas será el siguiente:

Curso Académico: ----/--	
Nombre: ---------- Curso: ---	
Asignaturas	Calificación
---------- ----------	---------- ----------
Nota media	------

donde tanto la calificación como la nota media deben aparecer en letra, es decir: Muy Deficiente, Insuficiente, Aprobado, etc.

Glosario

Acceso: forma de acceder a un dato.

Acceso aleatorio: característica de los soportes direccionables que permiten la localización de un dato de forma directa.

Acceso indexado: se accede directamente a un dato por su clave identificativa.

Acceso secuencial: se accede a los datos uno detrás de otro, en secuencia.

Actualización: operación que nos permite mantener vigente la información de un fichero.

Acumulador: variable que permite almacenar el contenido de una suma.

Alfabeto binario: posee dos únicos símbolos, 0 y 1.

Alfanumérico: información compuesta por caracteres alfabéticos, numéricos o especiales.

Algoritmo: conjunto ordenado y finito de reglas y procedimientos que describen la resolución de todos los problemas de un mismo tipo, independientemente de los datos.

Alta: operación que consiste en añadir un nuevo registro al fichero.

AND: operador lógico de conjunción.

Archivo: conjunto de registros relacionados con el mismo tema.

Argumento: porción de información que sigue a un comando.

Array: conjunto de datos homogéneos almacenados en la memoria del ordenador.

Asignación: operación por la cual se almacena un dato en una variable o el contenido de una variable en otra variable.

Auxiliar: variable que guarda un valor determinado de una variable.

Baja: operación que permite desaparecer físicamente un registro de un fichero.

Bandera: switch.

Bit: unidad mínima de información. Puede ser un 1, o un 0.

Byte: conjunto de bits que se manejan como una unidad. Generalmente está formado por 8 bits. Un byte puede almacenar un carácter.

Bucle: repetición de un conjunto de instrucciones.

Borrar: operación que permite hacer desaparecer físicamente un registro de un fichero, o un fichero del disco.

Borrar pantalla: operación que permite que la pantalla aparezca sin ningún tipo de visualización.

Cabecera: comentarios que deben aparecer una sola vez en cada pantalla, o al principio de la hoja de impresora.

Campo: cada uno de los componentes de un registro que está formado por un conjunto de caracteres capaces de expresar por sí mismos una información unitaria e independiente.

Carácter: lo son las letras, figuras, números, signos de puntuación, o cualquier otro símbolo que forma parte de un dato.

Carácter alfanumérico: cualquier carácter. Si son los números, con ellos no se pueden realizar operaciones aritméticas.

Carácter numérico: cualquier número, con ellos sí se pueden realizar operaciones aritméticas.

Carácter especial: son los caracteres no alfanuméricos, como los signos de puntuación.

Ciclo: bucle.

Codificación: transcripción de algoritmo a programa.

Comando: instrucción dada al ordenador.

Concatenar: unir dos o más variables alfanuméricas.

Contador: variable que se utiliza para contar.

Clasificación: reubicación de los elementos en una secuencia determinada.

Clave: campo especial que identifica a un único registro.

Cursor: indicador destelleante que señala el lugar donde aparecerá el siguiente carácter en la pantalla.

Datos: información codificada en un formato aceptable por los ordenadores.

Dimensión: reservar memoria en el ordenador para albergar a los elementos del array.

Disco: soporte de la información de forma circular.

Ejecutar: poner en marcha un programa.

Entrada de datos: acción de introducir datos en el ordenador.

EOF: siglas de *End Of File*, los ficheros finalizan con un carácter particular cuya misión es informar al sistema que se ha alcanzado el final de fichero.

Fichero: archivo.

Formato: reordenación de texto o datos de modo aceptable desde un punto de vista visual.

Flag: switch.

Hardware: componentes físicos que integran un ordenador.

Impresora: unidad de salida de la información que utiliza como soporte el papel.

Indicativo: clave.

Indice: variable que contiene la posición del array de la cual deseamos leer o escribir.

Información: lo que se deduce de los datos.

Informática: ciencia que estudia el tratamiento automático de la información.

Instrucción: mandato que se le da al ordenador para que realice una determinada acción.

Interrupción: señal que interrumpe la ejecución de un programa con una nueva orden.

Interruptor: variable que sólo puede tomar dos valores exclusivos: encendido o apagado, 0 o 1.

Línea de detalle: contiene la información de un registro y/o el resultado de alguna operación efectuada con algún campo del mismo.

Lista: array.

Lenguaje de programación: lenguaje utilizado para escribir programas.

Llamada: orden que se le da a un programa para que se ejecute.

Marcar: operación que permite distinguir un registro, para que en una operación posterior pueda ser dado de baja.

Matriz: array bidimensional.

Memoria: dispositivo físico para el almacenamiento de la información.

Menú: pantalla que permite la selección dirigida de órdenes de un programa.

Merge o mezclar: operación que permite obtener un fichero o un array a partir de otros dos ficheros o arrays.

NOT: operador lógico de negación.

Opción: variación o modificación de un comando que generalmente indica explícitamente el usuario.

Operador: símbolo que representa un enlace entre operandos.

Operador unario: operador que afecta sólo a un operando.

Operador binario: operador que afecta a dos operandos.

Operando: cada uno de los argumentos que intervienen en una operación.

OR: operador lógico de disyunción.

Orden: instrucción, comando.

Ordenador: máquina utilizada para conseguir el tratamiento automático de la información.

Organización: forma como están grabados los registros de un fichero sobre el soporte de datos.

Pantalla: periférico que refleja la información enviada por el ordenador o que ha sido introducida mediante el teclado.

Periférico: dispositivo de entrada, salida o almacenamiento de la información conectado al ordenador.

Pie de página: comentario o resultados que se imprimen al final de una página.

Programa: expresión de un algoritmo en un lenguaje de programación.

Pseudocódigo: descripción de un algoritmo utilizando una mezcla de frases en lenguaje común, junto con instrucciones y palabras claves.

Puntero: variable que contiene la dirección física de un determinado elemento.

Registro: conjunto de campos relacionados entre sí.

Ruptura de control: cambio en el formato de impresión de una línea de detalle.

Salida de datos: visualización de una determinada información en pantalla o en impresora.

Sentencia: instrucción.

Sinónimos: registros con la misma dirección física.

Sistema: es el conjunto de software y hardware capaz de procesar toda la información que el usuario requiere.

Software: conjunto de programas que se ejecutan o pueden ejecutarse sobre un ordenador.

Soporte: medio físico donde se representa la información.

Sumador: acumulador.

Switch: interruptor.

Tabla: array bidimensional.

Usuario: persona que hace uso de un sistema de un ordenador.

Variable: zona de la memoria del ordenador reconocida por un nombre común que almacena un dato.

Vector: array.

Bibliografía

ALCALDE, E.; GARCIA, M. y PEÑUELAS, S.: *Informática básica.* McGraw-Hill.

ALCALDE, E. y GARCIA, M.: *Metodología de la programación. Aplicaciones en Basic, Cobol y Pascal.* McGraw-Hill.

DALE, N. y ORSHALICK, D.: *Pascal.* McGraw-Hill.

DUNTEMANN, J.: *La biblia del Turbo Pascal.* Anaya-Multimedia.

GARCIA DE SOLA, J. F.: *Estructura de la Información.* McGraw-Hill.

GUILLERA AGÜERA, LL.: *Introducción a la informática.* EDUNSA.

NOVEL I CABRE, A.: *Informática básica.* CCUPC.

SANCHEZ, D. y GOMEZ, P.: *Programación en Pascal a través de pseudocódigo.* McGraw-Hill.

TSU-DER CHOU, G.: dBase III. *Guía del programador.* Anaya-Multimedia.

Empaquetar. 189
Escribir registro. 184
Estructuras de datos. 104
  tipos. 104
EOF. 184

Fecha del sistema. 225
Ficheros. 180
  características. 182
  clasificación según su uso. 182
  directos. 250
  indexados. 273
  operaciones sobre los. 182
  organización. 183
  secuenciales. 183
Filas y columnas variables. 45
Final del reporte. 216

Generación informes. 216
Grabar registro. 254

Hacer mientras:
  anidados. 45
  no sea EOF (). 184
  sintaxis. 17

IF. sintasis. 17
IF-ELSE. sintaxis. 18
  anidados. sintaxis. 19
IF FOUND (). sintaxis. 276
Indice:
  array bidimensional. 150
  array tridimensional. 167
  array unidimensional. 105
Inserción. ordenación por. 138
Instrucción. 1
  de entrada. 13
  de salida. 14
INT. sintaxis. 19
Intercalaciones sucesivas. método. 206

Leer registro. 183. 254. 276
Línea de detalle. 216
Longitud:
  de una frase. 49
  del fichero. 254

Marca final fichero. 181
Marcar registro. 189
Memoria central. 1
Menús. 81
  sintasis. 81
Merge. 206
Mezclas. ordenación. 141
Mínimo. ordenación. 136

Nombre de los/as:
  arrays bidimensionales. 150
  arrays tridimensionales. 167
  arrays unidimensionales. 105
  ficheros. 180
  variables. 7
Números aleatorios. 76
  generación. 76
  sintaxis. 76
Número de elementos variables. en los:
  arrays bidimensionales. 163
  arrays unidimensionales. 119
Número de registros. 203

Opción. 82
Operadores. 3
  orden de evaluación. 7
  tipos. 4
Ordenador. 1
Ordenación:
  de arrays bidimensionales. 156
  de ficheros. 201
  de listas. 132
  secuencias de. 207

Palabras reservadas. 8
Pie. 216
  de página. 70
Posición. 106
Posicionarse EOF. 184
Programa. 1
  principal. 54
  tipos de. 12
Puntero. 183
Pseudocódigo. 13
  estructura. 13

Registro. 180
cabecera. 252
inexistente. 192. 286. 287
regrabar. 184
Renombrar fichero. 188
Repetir hasta. sintaxis. 107
Ruptura de control. 236

Salto de página. 221
SHELL. método:
arrays bidimensionales. 163
arrays unidimensionales. 139
ficheros

Sinónimos. 250
Subprograma. 54
sintaxis. 54
Sumador. 19
Switch:
definición. 56
método. 134

Variables. 3. 7
asignación. 9
definición. 7
tipos. 8

Indice

Abrir fichero, 183, 254, 275, 278
Activar:
  impresora, 221
  pantalla, 221
Algoritmo, 11
  características, 12
  tipos, 12
Altas duplicadas, 185, 197, 255, 281
Area de:
  datos, 273
  índices, 273
  overflow, 273
Arrays:
  bidimensionales, 105, 150
  paralelos, 113
  tridimensionales, 105, 167
  unidimensionales, 105
Auxiliar, 38

Bajas inexistentes, 187, 262, 283, 287
Bit, 2, 9
Borrar:
  fichero, 188
  registro, 188
Bucles, 17
  anidados, 45
Burbuja, método, 132
  Switch, 135
Búsqueda:
  binaria, 136
  lineal, 143
    array desordenado, 143
    array ordenado, 145
Byte, 2, 9

Cabecera, 69
  de informe, 216
Campo, 180
  clave, 181, 275
  puntero, 252
Carácter, cálculo de un, 49
CASE, 81
Cerrar:
  fichero, 184, 254, 276
  índices, 276
Ciclo, 13, 17
Clasificación:
  externa, 201, 205
  ficheros, 201
  interna, 201, 202
Comentario, 204
Constantes, 10
Contador, 18
Convertir a mayúsculas, 29
Convertir variables de:
  alfanuméricas a numéricas, 203
  numéricas a alfanuméricas, 203

Datos, 3
  tipos, 3
Detener listado, 70
Dimensionar arrays:
  bidimensionales, 150
  tridimensionales, 167
  unidimensionales, 105
Direccionamiento:
  conversión a octal, 251
  directo, 251
  por el centro al cuadrado, 251

Elemento, 106
  búsqueda de un, 143
Elementos variables, en los:
  arrays bidimensionales, 163
  arrays unidimensionales, 119