

Una introducción a la programación. Un enfoque algorítmico

Anexo I. Programas en Pascal

Jesús J. García Molina
Francisco J. Montoya Dato
José L. Fernández Alemán
María J. Majado Rosales

21 de octubre de 2005

Índice

1. Introducción	3
1.1. Guía de usuario para Turbo Pascal 5.5	4
1.2. Guía de usuario para Dev-Pascal	4
2. Fe de erratas	6
3. Listados	8
3.1. Capítulo 2	8
3.2. Capítulo 3	27
3.3. Capítulo 4	43
3.4. Capítulo 5	63
3.5. Capítulo 6	85
3.6. Capítulo 7	126
3.7. Capítulo 8	147

1. Introducción

Este documento incluye los programas Pascal que se han obtenido al codificar en este lenguaje de programación todos los algoritmos que aparecen en el libro “Una Introducción a la Programación. Un enfoque algorítmico”, según la conversión descrita en el Capítulo 9 del libro. Por tanto, el lector del libro dispone de los programas que le permiten analizar el comportamiento de los algoritmos durante la ejecución.

Los programas han sido creados y probados con las versiones 5.5 y 7.0 de Turbo Pascal de Borland y con la versión 1.9.2 del entorno Dev-Pascal (compilador Free Pascal) de Bloodshed, sobre los sistemas operativos Windows 98, 2000 y XP. Si el lector está más habituado a entornos Linux, puede optar por emplear Free Pascal (fpc) o GNU Pascal (gpc) para Linux.

Las versiones de Turbo Pascal se ejecutan sobre DOS y ya no son soportadas por Borland; la versión 5.5 se puede descargar de forma gratuita del sitio web de Borland

<http://community.borland.com/article/0,1410,20803,00.html> pero no la versión 7.0 que es más cómoda de usar (entorno multiventana, manejo del ratón, etc.). Dev-Pascal es un entorno de programación moderno, más potente y amigable, y se puede descargar desde el sitio web de Bloodshed <http://www.bloodshed.net/devpascal.html>.

El lector puede ejecutar los programas a partir del fichero **pascal.zip** que se encuentra en el sitio web junto a este documento. Los programas están distribuidos en carpetas, una por cada capítulo. El nombre de los archivos de cada capítulo hace referencia al identificador en el libro del algoritmo que implementa, por ejemplo, **ALG6_4.pas**, es el programa para el **Algoritmo 6.4** del Capítulo 6.

Puesto que Pascal no dispone del tipo secuencia propiamente dicho, se han implementado unas unidades que permiten el manejo de secuencias de caracteres, enteros y reales para el primer y segundo modelos de acceso secuencial descritos en el Capítulo 5. Los dos últimos modelos se han construido apoyándonos en el concepto de secuencia intermedia. En la carpeta **unit** están los archivos fuentes y los archivos objeto de seis unidades, que corresponden con los dos primeros modelos de cada uno de los tres tipos de datos: caracteres, enteros y reales.

Cada archivo que contiene una unidad se ha nombrado con el prefijo “unitms” seguido de una letra que indica el tipo de elemento (“c”, “e” o “r”) y un “1” o “2” para indicar el modelo. Por ejemplo, **Unitmse2.pas** es la unidad para secuencias de enteros del segundo modelo y **Unitmsc1.pas** es la unidad para secuencias de caracteres del primer modelo.

Las primitivas para el manejo de secuencias se nombran igual que en el libro (**EA**, **Comenzar**, **Avanzar**, etc.) pero con un sufijo que indica el tipo de elemento y el modelo secuencial (**EA_MSC1**, **EA_MSE2**, **Avanzar_MSR2**, **Avanzar_MSC2**, etc.). Para conocer el léxico que puede utilizar se puede consultar la parte interface del código fuente de cada unidad. El lector interesado también puede escudriñar el código de las unidades con el objetivo de aprender sobre estructuras de datos dinámicas (se han implementado como listas).

Se han creado archivos que almacenan secuencias que se utilizan como ejemplo de entrada en la ejecución de los programas, y que están preparados para que el lector pueda modificarlos. Estos archivos de datos se identifican con el prefijo “datos” seguido del identificador del algoritmo. Por ejemplo, **datos4.3.txt** corresponde con la entrada de datos del programa **ALG4_3.pas**. Las secuencias de datos se almacenan como líneas de caracteres y las secuencias de números con un número por línea de entrada. En la carpeta con los programas del Capítulo 6 se incluyen dos programas (**GENF6_1.pas** y **GENF6_4.pas**) que crean los archivos de datos Pascal para el **Algoritmo 6.1** y el **Algoritmo 6.4** (también se proporcionan los programas que permiten la visualización de los dos archivos creados, **VERF6_1.pas** y **VERF6_4.pas**).

En los programas no se ha realizado la comprobación de que los datos de entrada son válidos, por lo

que en el caso de una entrada incorrecta, el resultado es impredecible.

En una próxima versión de este documento se incluirá el código de todas las acciones y funciones del Capítulo 8, así como de la aplicación gestor de archivos indexados que se discute en dicho capítulo. También estarán disponibles en el sitio web del libro, algoritmos y programas que son soluciones a los ejercicios del libro, así como la implementación de las unidades del tercer y cuarto modelo de acceso secuencial.

1.1. Guía de usuario para Turbo Pascal 5.5

Existen numerosos libros que ofrecen información sobre el uso de los entornos de Borland, por ejemplo: “Programación en Turbo Pascal. Versiones 5.5, 6.0, 7.0” de L. Joyanes, McGraw-Hill, 1993, y “Programación en Turbo/Borland Pascal 7” de L. Joyanes, Mc Graw Hill, 1998. Aquí ofrecemos una breve guía para ejecutar los programas, pero el lector puede consultar esos libros para obtener más detalles. Los pasos para ejecutar un programa serían:

1. Abrir el entorno desde la ventana del sistema operativo (`c:\tp>turbo`).
2. Establecer el directorio en el que se encuentran los programas (opción **Change dir** del menú **File**).
3. Indicar el directorio en el que se encuentran las unidades (añadir directorio en el cuadro de texto **Unit Directories** de la opción **Directories** del menú **Options**). Este paso sólo sería necesario realizarlo una vez.
4. Cargar el programa (opción **Load** del menú **File**).
5. Ejecutar el programa (opción **Run** menú **Run**).

La carpeta **unit** incluye tanto los archivos fuente de las unidades como los archivos compilados, pero si fuese necesario compilar una unidad se debe proceder del siguiente modo:

1. Introducir el directorio en el que se almacenan las unidades (añadir directorio en el cuadro de texto **EXE & TPU Directory** de la opción **Directories** del menú **Options**).
2. Cargar el archivo con el código fuente de la unidad, opción **Load** del menú **File**.
3. Indicar que la compilación se almacenará en disco con la opción **Destination** del menú **Compile**.
4. Compilar con la opción **Compile** del menú **Compile**. Si no se indica el directorio en el paso 1, el archivo objeto (extensión TPU) se almacenará en el mismo directorio que el archivo fuente.

1.2. Guía de usuario para Dev-Pascal

El entorno dispone de una ayuda y un tutorial. Los pasos para ejecutar un programa, una vez abierto el entorno, serían (existen botones para ejecutar las órdenes, además de teclas rápidas):

1. Establecer el directorio en el que se encuentran las unidades en el cuadro de texto “**Add the directory below ...**” de la opción **Compiler options** del menú **Options**. Este paso sólo sería necesario realizarlo una vez.
2. Cargar el programa (opción **Open file** del menú **File**).

3. Compilar el programa (opción **Compile** del menú **Execute**).
4. Ejecutar el programa (opción **Run** del menú **Execute**).

Para ejecutar los programas que utilizan las máquinas secuenciales bastarían los pasos anteriores si en el directorio de las unidades se encuentran el archivo fuente de la unidad empleada, ya que se generaría automáticamente el archivo objeto de la unidad. Si por alguna razón se necesitase compilar las unidades de forma independiente, los pasos para crear el archivo objeto de una unidad son los siguientes:

1. Crear un proyecto con el nombre de la unidad. Seleccionar **Empty project** de la opción **New project** del menú **File**. Eliminar el archivo incluido por defecto en el proyecto y cargar el archivo que contiene el código fuente de la unidad, opción **Add to project** del menú **Project**.
2. Habilitar la creación de una DLL en la opción **Project options** del menú **Project**.
3. Compilar la unidad mediante la opción **Compile** del menú **Execute**.

Sobre los sistemas operativos Windows 2000 y XP, durante la ejecución del programa con la última versión de Dev-Pascal, el directorio actual es el directorio en que se instala DevPascal (por defecto `c:\Dev-Pas`) y no funciona la opción de establecer como directorio actual aquel en el que se encuentra el ejecutable. Por tanto, cuando en un programa hay referencias a ficheros, o bien se indica la ruta completa, o bien se ejecuta el programa desde fuera del entorno, por ejemplo, desde el explorador de Windows. Este problema no se tiene con el sistema operativo Windows 98.

2. Fe de erratas

Hemos encontrado en el libro las erratas que indicamos abajo organizadas por capítulos. En el sitio web del libro iremos actualizando esta fe de erratas en un fichero aparte.

Capítulo 4

Algoritmo 4.4

El segundo argumento de la llamada a la acción **Escribir** que escribe el resultado debe ser **suma/numElem** en vez de **suma/num**.

Capítulo 6

Algoritmo 6.17

La función **PosEnTabla** no coincide en el texto y en el algoritmo. El algoritmo válido es el que aparece en el texto.

En la función **Codificacion** hay que cambiar **nuevoPar.c** por **nuevoPar.s** en la segunda asignación de la cláusula **EN_OTRO_CASO** de la instrucción **SEGÚN**.

Capítulo 7

Algoritmo 7.13

El tipo **Estaca** debe definirse fuera de la acción **Hanoi**, ya que se utiliza como tipo de los parámetros **origen**, **intermedio** y **destino**.

La segunda llamada a la acción **Hanoi** debería ser **Hanoi(n-1, intermedio, origen, destino)** en vez de **Hanoi(n-1, intermedio, destino, origen)**.

Sección 7.5.4 (Página 377)

En la postcondición de la acción **OrdenarI** debería decir $t_k = T_{vieja_k}$ en vez de $t_k = T_{vieja_k}$.

Capítulo 8

Sección 8.3 (Página 395)

En la tercera línea debería decir “una secuencia de cuatro nodos:” en vez de “una secuencia de tres nodos:”.

Sección 8.7.4 (Página 464)

En la acción **InsertarArchivoIndexado** hay que cambiar:

SI (**BuscarPosicionIndice**(**arch**↑.**indice**, **p.clave**) : **POS_NULA**)

por:

SI (**BuscarPosicionIndice**(**arch**↑.**indice**, **p.clave**) $\neq$ **POS_NULA**)

Capítulo 9

Sección 9.7.12 (Página 565)

En la Figura 9.4 el elemento `[0][29]` debería ser `t[0][29]`

3. Listados

3.1. Capítulo 2

```

program Algoritmo2_1;
(*
  * Algoritmo 2.1. Calculo de la nota final de una asignatura
  * Titulo del libro: Una introduccion a la programacion.
  *
    Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
    Francisco J. Montoya Dato
  *
    Jose L. Fernandez Aleman
  *
    Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 2. Secuenciacion y analisis de casos
  *)

uses crt;

var
  notaTeoria : real;
  notaPractica : real;
  notaFinal : real;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 2.1. Calculo de la nota final de una asignatura *****');
  writeln('*****');
  writeln;
  write('Nota de teoria: ');
  readln(notaTeoria);
  writeln;
  write('Nota de practicas: ');
  readln(notaPractica);
  notaFinal := notaTeoria * 0.7 + notaPractica * 0.3;
  writeln;
  write('La nota final es: ');
  writeln(notaFinal:1:2);
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo2_2;
(*

```

```
* Algoritmo 2.2. Convertir temperatura Fahrenheit en temperatura Celsius
* Titulo del libro: Una introduccion a la programacion.
*                               Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*                               Francisco J. Montoya Dato
*                               Jose L. Fernandez Aleman
*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;

var
    tempFahrenheit : real; (* dato, temperatura en grados Fahrenheit *)
    tempCelsius    : real; (* resultado, temperatura en grados Celsius *)

procedure ConvertirFahrenheitCelsius;
(* PRE 0 <= tempFahrenheit <= 200 *)
(* POST convierte a grados celsius la temperatura de tempFahrenheit *)
begin
    tempCelsius := (5.0 / 9.0) * (tempFahrenheit - 32.0)
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.2. Convertir temperatura Fahrenheit en *****');
    writeln('*****          temperatura Celsius          *****');
    writeln('*****');
    writeln;

    write('Temperatura Fahrenheit (<= 0) y (>= 200): ');
    readln(tempFahrenheit);
    ConvertirFahrenheitCelsius;
    writeln('La temperatura en grados Celsius es: ', tempCelsius:1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo2_3;
(*
* Algoritmo 2.3. Calculo del salario neto de un trabajador (version 1)
* Titulo del libro: Una introduccion a la programacion.
*                               Un enfoque algoritmico
```

```

* Autores del libro: Jesus J. Garcia Molina
*                   Francisco J. Montoya Dato
*                   Jose L. Fernandez Aleman
*                   Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;
const
    IMPUESTO = 0.20;
    SEGURO_MEDICO = 0.05;
    COMPLEMENTO_QUINQUENIO = 60;
    COMPLEMENTO_ANYO = 6;

var
    salarioBase : longint; (* dato, sueldo base del trabajador *)
    antiguedad : word;     (* dato, anyos en la empresa *)
    salarioNeto : real;    (* resultado, salario percibido por el trabajador *)
    salarioBruto : longint; (* salario bruto del trabajador *)
    descuentos : real;     (* descuentos aplicados *)

procedure CalcularSalarioBruto;
(* PRE salarioBase y antiguedad tienen un valor valido *)
(* POST salarioBruto contiene el salario bruto del trabajador *)
var
    numeroQuinquenios : word;
    numeroAnyos : word;
    pagoQuinquenios : word;
    pagoAños : word;

begin
    numeroQuinquenios := antiguedad div 5;
    numeroAnyos := antiguedad mod 5;
    pagoQuinquenios := numeroQuinquenios * COMPLEMENTO_QUINQUENIO;
    pagoAños := numeroAnyos * COMPLEMENTO_ANYO;
    salarioBruto := salarioBase + pagoQuinquenios + pagoAños
end;

procedure CalcularDescuentos;
(* PRE se ha calculado el salario bruto y se ha asignado a salarioBruto *)
(* POST descuentos almacena el valor total de los descuentos sobre el salario bruto *)

begin
    descuentos := salarioBruto * (IMPUESTO + SEGURO_MEDICO)
end;

```

```
procedure CalcularSalarioNeto;
(* PRE salarioBruto y descuentos almacenan, respectivamente, el salario bruto y el
   descuento que le corresponde *)
(* POST salarioNeto contiene el salario recibido por el trabajador,
   salarioNeto = salarioBruto - descuentos *)

begin
    salarioNeto := salarioBruto - descuentos
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.3. Calculo del salario neto de un *****');
    writeln('*****          trabajador (version 1)          *****');
    writeln('*****');
    writeln;
    write('Introduzca el salario base: ');
    readln(salarioBase);
    write('Introduzca la antigüedad: ');
    readln(antigüedad);
    CalcularSalarioBruto;
    CalcularDescuentos;
    CalcularSalarioNeto;
    writeln('El sueldo neto es: ', salarioNeto:1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo2_4;
(*
 * Algoritmo 2.4. Calculo del salario neto de un trabajador (version 2)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *          Francisco J. Montoya Dato
 *
 *          Jose L. Fernandez Aleman
 *
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 2. Secuenciacion y analisis de casos
 *)

uses crt;
const
    IMPUESTO = 0.20;
```

```

SEGURO_MEDICO = 0.05;
COMPLEMENTO_QUINQUENIO = 60;
COMPLEMENTO_ANYO = 6;
var
    salarioBase : word;      (* dato, sueldo base del trabajador *)
    antiguedad : word;       (* dato, anyos en la empresa *)
    salarioNeto : real;      (* resultado, salario percibido por el trabajador *)
    salarioBruto : word;     (* salario bruto del trabajador *)
    descuentos : real;      (* descuentos aplicados *)
    numeroQuinquenios : word;
    numeroAnyos : word;
    pagoQuinquenios : word;
    pagoAnyos : word;
begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.4. Calculo del salario neto de *****');
    writeln('***** un trabajador (version 2) *****');
    writeln('*****');
    writeln;
    write('Introduzca el salario base: ');
    readln(salarioBase);
    write('Introduzca la antiguedad: ');
    readln(antiguedad);

    (* Calculo del salario bruto *)
    numeroQuinquenios := antiguedad div 5;
    numeroAnyos := antiguedad mod 5;

    pagoQuinquenios := numeroQuinquenios * COMPLEMENTO_QUINQUENIO;
    pagoAnyos := numeroAnyos * COMPLEMENTO_ANYO;
    salarioBruto := salarioBase + pagoQuinquenios + pagoAnyos;

    (* Calculo de los descuentos y el salario neto *)
    descuentos := salarioBruto * (IMPUESTO + SEGURO_MEDICO);
    salarioNeto := salarioBruto - descuentos;

    writeln('El sueldo neto es: ', salarioNeto:1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo2_5;
(*

```

```
* Algoritmo 2.5. Expresar una cantidad de bytes en megabytes y kilobytes
* Titulo del libro: Una introduccion a la programacion.
*
*           Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*                   Francisco J. Montoya Dato
*                   Jose L. Fernandez Aleman
*                   Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;

var
  n : longint;           (* dato, numero de bytes a descomponer *)
  mb : longint;          (* resultado, numero de megabytes *)
  kb : 0..1024;          (* resultado, numero de kilobytes *)
  b : 0..1024;           (* resultado, numero de bytes *)

procedure Convertir;
  (* PRE 0 <= n *)
  (* POST (n = 1048576mb + 1024kb + b) y (0 <= kb < 1024) y (0 <= b < 1024) *)
  var
    rb : 0..1048575; (* resto de bytes *)

  begin
    mb := n div 1048576;
    rb := n mod 1048576;
    (* E1: n = 1048576 mb + rb y 0 <= rb < 1048576 *)
    kb := rb div 1024;
    b := rb mod 1024
  end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 2.5. Expresar una cantidad          *****');
  writeln('*****           de bytes en megabytes y kilobytes *****');
  writeln('*****');
  writeln;
  writeln('Descomposicion en bytes, kilobytes y megabytes');
  write('Introduzca un entero (>= 0): ');
  readln(n);
  Convertir;
  writeln('La descomposicion es:');
  writeln('  Megabytes: ', mb);
```

```

    writeln('    Kilobytes: ', kb);
    writeln('    Bytes: ', b);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo2_6;
(*
  * Algoritmo 2.6. Dibujar un cuadrado
  * Titulo del libro: Una introduccion a la programacion.
  *
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 2. Secuenciacion y analisis de casos
  *)

uses Graph3, crt;

var
  lado : integer;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 2.6. Dibujar un cuadrado *****');
  writeln('*****');
  writeln;
  write ('Introduzca la longitud de lado del cuadrado:');
  readln (lado);

  (* E0: Indiferente *)
  GraphMode;          (* 320 x 200 modo grafico blanco-negro *)
  ClearScreen;
  ShowTurtle;         (* Muestra la tortuga *)
  (* E1: pos = (0, 0), dir = 0, trazo = Verdadero *)
  Forwd(lado); TurnRight (90);
  (* E2: pos = (0, lado), dir = 90, trazo = Verdadero,
  se ha dibujado el primer lado *)
  Forwd(lado); TurnRight (90);
  (* E3: pos = (lado, lado), dir = 180, trazo = Verdadero,
  se han dibujado dos lados *)
  Forwd(lado); TurnRight (90);

```

```
(* E4: pos = (lado, 0), dir = 270, trazo = Verdadero,
se han dibujado tres lados *)
  Forwd(lado); TurnRight (90);
(* Ef: pos = (0, 0), dir = 0, trazo = Verdadero,
se ha dibujado el cuadrado *)
  writeln;
  write ('Pulse enter para continuar');
  readln
end.
```

```
program Algoritmo2_7;
(*
* Algoritmo 2.7. Correspondencia entre calificaciones (version 1)
* Titulo del libro: Una introduccion a la programacion.
*
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
* Francisco J. Montoya Dato
*
* Jose L. Fernandez Aleman
*
* Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;

var
  nota : 0..20;  (* nota en la universidad extranjera, 0 <= nota <= 20 *)

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 2.7. Correspondencia entre calificaciones (version 1)
  *****');
  writeln('*****');
  writeln;
  write('Introduzca la nota (>= 0) y (<= 20): ');
  readln(nota);
  write('La calificacion es ');
  case nota of
    20 :      write('matricula de honor');
    19, 18 :  write('sobresaliente');
    17, 16 :  write('notable');
    15, 14 :  write('aprobado');
    else if nota < 14 then write('suspenso')
  end;
  writeln;
```

```

    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo2_8;
(*
  * Algoritmo 2.8. Correspondencia entre calificaciones (version 2)
  * Titulo del libro: Una introduccion a la programacion.
  *
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 2. Secuenciacion y analisis de casos
  *)

uses crt;

var
    nota : integer;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.8. Correspondencia entre calificaciones (version 2)
    *****');
    writeln('*****');
    writeln;
    write('Introduzca la nota (>= 0) y (<= 20): ');
    readln (nota);
    write('La calificacion es ');
    case nota of
        20 :      write('matricula de honor');
        19, 18 :  write('sobresaliente');
        17, 16 :  write('notable');
        15, 14 :  write('aprobado');
        0..13:    write('suspense');
        else      write('no valida');
    end;
    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```
program Algoritmo2_9;
(*
 * Algoritmo 2.9. Simulacion de una calculadora simple
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 2. Secuenciacion y analisis de casos
 *)

uses crt;

var
    operando1, operando2 : longint;
    operador : char;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.9. Simulacion de una calculadora simple *****');
    writeln('*****');
    writeln;
    write ('Introduzca el primer operando entero: ');
    readln (operando1);
    write ('Introduzca el segundo operando entero: ');
    readln (operando2);
    write ('Introduzca el operador (+, *, -, /): ');
    readln (operador);
    write ('El resultado de la operacion es: ');
    case operador of
        '+' : writeln(operando1 + operando2);
        '*' : writeln(operando1 * operando2);
        '-' : writeln(operando1 - operando2);
        '/' : writeln(operando1 div operando2);
        else writeln('operador incorrecto')
    end;
    writeln;
    write ('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo2_10;
(*
```

```

* Algoritmo 2.10. Obtener el mayor de dos numeros enteros
* Titulo del libro: Una introduccion a la programacion.
*
      Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
      Francisco J. Montoya Dato
*
      Jose L. Fernandez Aleman
*
      Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;

var
  x, z : longint;
  mayor : longint;

begin
  clrscr;
  (* PRE (x = X) y (z = Z) *)
  (* POST ((x >= z) => (mayor = X)) y ((z >= x) => (mayor = Z)) *)
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 2.10. Obtener el mayor de dos numeros enteros *****');
  writeln('*****');
  writeln;
  write ('Introduzca el primer entero: ');
  readln (x);
  write ('Introduzca el segundo entero: ');
  readln (z);
  if x >= z then mayor := x
  else mayor := z;
  writeln ('El numero mayor es: ', mayor);
  writeln;
  write ('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo2_11;
(*
* Algoritmo 2.11. Comprobar si una fecha es correcta (version 1)
* Titulo del libro: Una introduccion a la programacion.
*
      Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
      Francisco J. Montoya Dato
*
      Jose L. Fernandez Aleman
*
      Maria J. Majado Rosales

```

```
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;

var
    dia : word;
    mes : word;
    anyo : word;
    esBisiesto : boolean;      (* indicador de anyo bisiesto *)
    fechaValida : boolean;    (* indicador de fecha valida *)

procedure AnyoBisiesto;
(* POST esBisiesto es Verdadero si el anyo es bisiesto y Falso en caso contrario *)
begin
    esBisiesto := (anyo mod 4 = 0) and (anyo mod 100 <> 0) or
                  (anyo mod 400 = 0) and (anyo <> 3600)
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.11. Comprobar si una fecha es correcta (version 1) *****');
    writeln('*****');
    writeln;
    write('Introduzca el dia: ');
    readln(dia);
    write('Introduzca el mes: ');
    readln(mes);
    write('Introduzca el anyo: ');
    readln(anyo);
    fechaValida := true;
    if dia < 1 then fechaValida := false
    else
        case mes of
            1, 3, 5, 7, 8, 10, 12: (* meses de 31 dias *)
                if dia > 31 then fechaValida := false;
            4, 6, 9, 11:          (* meses de 30 dias *)
                if dia > 30 then fechaValida := false;
            2:                    (* mes de febrero *)
                begin
                    AnyoBisiesto;
                    if (dia > 29) or (not esBisiesto and (dia > 28))
                        then fechaValida := false
                end
        end
end
```

```

        end
        else fechaValida := false
    end;
    write (dia, '/', mes, '/', anyo);
    if fechaValida then writeln(' es una fecha valida')
        else writeln(' no es una fecha valida');
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo2_12;
(*
* Algoritmo 2.12. Comprobar si una fecha es correcta (version 2)
* Titulo del libro: Una introduccion a la programacion.
*
*          Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
*          Francisco J. Montoya Dato
*
*          Jose L. Fernandez Aleman
*
*          Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt;

var
    dia : word;
    mes : word;
    anyo : word;
    esBisiesto : boolean;    (* indicador de anyo bisiesto *)
    fechaValida : boolean;  (* indicador de fecha valida *)

    (* igual que en Algoritmo 2.11 *)

procedure AnyoBisiesto;
(* POST esBisiesto es Verdadero si el anyo es bisiesto y Falso en caso contrario *)
begin
    esBisiesto := (anyo mod 4 = 0) and (anyo mod 100 <> 0) or
        (anyo mod 400 = 0) and (anyo <> 3600)
end;

begin
    clrscr;
    writeln;
    writeln('*****');

```

```
writeln('***** Algoritmo 2.12. Comprobar si una fecha es correcta (version 2) *****');
writeln('*****');
writeln;
write('Introduzca el dia: ');
readln(dia);
write('Introduzca el mes: ');
readln(mes);
write('Introduzca el anyo: ');
readln(anyo);
if dia < 1 then fechaValida := false
else
    case mes of
        1, 3, 5, 7, 8, 10, 12: fechaValida := dia <= 31;
        4, 6, 9, 11: fechaValida := dia <= 30;
        2: begin
            AnyoBisiesto;
            fechaValida := (dia <= 29) and (esBisiesto or (dia <= 28));
        end
        else fechaValida := false
    end;
write (dia, '/', mes, '/', anyo);
if fechaValida then writeln( ' es una fecha valida')
else writeln( ' no es una fecha valida');
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo2_13;
(*
 * Algoritmo 2.13. Calculo del salario neto de un trabajador (version 3)
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 * Jose L. Fernandez Aleman
 * Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 2. Secuenciacion y analisis de casos
 *)

uses crt;

const
    HORA_NORMAL = 10;
    HORA_EXTRA = 12;
```

```

    UMBRAL_HORAS_EXTRAS = 140;
    IMPUESTO_BAJO = 0.16;
    IMPUESTO_ALTO = 0.22;
    UMBRAL_SUELDO = 1300;
    COMPLEMENTO_QUINQUENIO = 60;
    COMPLEMENTO_ANYO = 6;

var
    numeroHoras : word;    (* dato, horas trabajadas en un mes por el empleado id *)
    id : word;             (* dato, identificador del empleado *)
    antiguedad : word;     (* dato, anyos en la empresa *)
    salarioNeto : real;    (* resultado, salario percibido por el trabajador *)
    salarioBase : word;
    salarioBruto : word;
    descuentos : real;

procedure CalcularSalarioBruto;
(* PRE numeroHoras y antiguedad tienen un valor *)
(* POST salarioBruto contiene el salario bruto del trabajador *)

var
    numeroQuinquenios : word;
    numeroAnyos : word;
    pagoQuinquenios : word;
    pagoAnyos : word;

begin
(* calcular sueldo base por horas trabajadas *)
    if numeroHoras > UMBRAL_HORAS_EXTRAS
    then salarioBase := (numeroHoras - UMBRAL_HORAS_EXTRAS) * HORA_EXTRA
        + UMBRAL_HORAS_EXTRAS * HORA_NORMAL
    else salarioBase := numeroHoras * HORA_NORMAL;

(* calcular gratificacion por antiguedad *)
    numeroQuinquenios := antiguedad DIV 5;
    numeroAnyos := antiguedad MOD 5;
    pagoQuinquenios := numeroQuinquenios * COMPLEMENTO_QUINQUENIO;
    pagoAnyos := numeroAnyos * COMPLEMENTO_ANYO;

(* calcular salario bruto *)
    salarioBruto := salarioBase + pagoQuinquenios + pagoAnyos
end;

procedure CalcularDescuentos;
(* PRE salarioBruto almacena el salario bruto calculado *)
(* POST descuentos almacena el valor total de los descuentos sobre el salario
    bruto *)

```

```
begin
    if salarioBruto > UMBRAL_SUELDO
    then descuentos := salarioBruto * IMPUESTO_ALTO
    else descuentos := salarioBruto * IMPUESTO_BAJO
end;

procedure CalcularSalarioNeto;
    (* PRE salarioBruto y descuentos almacenan, respectivamente, el salario bruto y el
       descuento correspondiente *)
    (* POST salarioNeto contiene el salario percibido por el trabajador,
       salarioNeto = salarioBruto -descuentos *)

begin
    salarioNeto := salarioBruto - descuentos
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 2.13. Calculo del salario neto de *****');
    writeln('***** un trabajador (version 3) *****');
    writeln('*****');
    writeln;
    write('Identificador del empleado: ');
    readln(id);
    write('Numero de horas trabajadas en un mes por el empleado: ');
    readln(numeroHoras);
    write('Antigüedad del empleado: ');
    readln(antigüedad);
    CalcularSalarioBruto;
    CalcularDescuentos;
    CalcularSalarioNeto;
    writeln('El salario del empleado ', id, ' es: ', salarioNeto:1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo2_14;
(*
 * Algoritmo 2.14. Dibujar un cuadrado a partir de la posicion de los vertices
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 *
 * Jose L. Fernandez Aleman
*)
```

```

*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 2. Secuenciacion y analisis de casos
*)

uses crt, graph;

type
  Punto = record
    abs : integer; (* valores enteros en lugar de reales *)
    ord : integer;
  end;
  Cuadrado = record
    v1, v2, v3, v4 : Punto;
  end;

var
  c : Cuadrado;
  cg, mg : integer;      (* controlador y modo grafico *)

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 2.14. Dibujar un cuadrado a partir *****');
  writeln('***** de la posicion de los vertices *****');
  writeln('*****');
  writeln;

  (* E0: Indiferente *)

  writeln('Introduzca los puntos de un cuadrado');
  write(' Punto 1 (abscisa ordenada): ');
  readln(c.v1.abs, c.v1.ord);
  write(' Punto 2 (abscisa ordenada): ');
  readln(c.v2.abs, c.v2.ord);
  write(' Punto 3 (abscisa ordenada): ');
  readln(c.v3.abs, c.v3.ord);
  write(' Punto 4 (abscisa ordenada): ');
  readln(c.v4.abs, c.v4.ord);

  InitGraph(cg,mg,'c:\tp'); (* inicializa modo grafico*)

  (* E1: pos = (0, 0), dir = 0, trazo = Verdadero;
  los puntos de c son vertices de un cuadrado *)

  MoveTo(c.v1.abs, c.v1.ord); (* mover al vertice v1 *)

```

```
(* E2: pos = c.v1, dir = 0, trazo = Verdadero *)
  LineTo(c.v2.abs,c.v2.ord); (* mover al vertice v2 y traza linea *)
  LineTo(c.v3.abs,c.v3.ord); (* mover al vertice v3 y traza linea *)
  LineTo(c.v4.abs,c.v4.ord); (* mover al vertice v4 y traza linea *)
  LineTo(c.v1.abs,c.v1.ord); (* mover al vertice v1 y traza linea *)
(* Ef: "cuadrado dibujado ", pos = c.v1, dir = 0, trazo = Verdadero *)
  readln;
  closeGraph;
end.
```

```
program Algoritmo2_15;
(*
  * Algoritmo 2.15. Comprobar si una fecha es correcta (version 3)
  * Titulo del libro: Una introduccion a la programacion.
  *
    Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
    Francisco J. Montoya Dato
  *
    Jose L. Fernandez Aleman
  *
    Maria J. Majado Rosales
  * Capitulo 2. Secuenciacion y analisis de casos
  *)

uses crt;

(*$R+*)

type
  Fecha = record
    dia : 1..31;
    mes : 1..12;
    anyo : word
  end;
var
  f : Fecha; (* dato, fecha dia/mes/anyo *)
  esBisiesto : boolean; (* indicador de anyo bisiesto *)
  fechaValida : boolean; (* indicador de fecha valida *)

  procedure AnyoBisiesto;
    (* POST esBisiesto es Verdadero si f.anyo es bisiesto y Falso en caso contrario *)

  begin
    esBisiesto := (f.anyo mod 4 = 0) and (f.anyo mod 100 <> 0) or
      (f.anyo mod 400 = 0) and (f.anyo <> 3600)
  end;

begin
  clrscr;
```

```

writeln;
writeln('*****');
writeln('***** Algoritmo 2.15. Comprobar si una fecha es correcta (version 3) *****
');
writeln('*****');
writeln;
write('Introduzca el dia: ');
readln(f.dia);
write('Introduzca el mes: ');
readln(f.mes);
write('Introduzca el anyo: ');
readln(f.anyo);
case f.mes of
  1, 3, 5, 7, 8, 10, 12: fechaValida := true;
  4, 6, 9, 11: fechaValida := f.dia <> 31;
  2: begin
      AnyoBisiesto;
      fechaValida := (f.dia <= 29) and (esBisiesto or (f.dia <> 29))
    end;
end;
write (f.dia, '/', f.mes, '/', f.anyo);
if fechaValida then writeln(' es una fecha valida')
  else writeln(' no es una fecha valida');
writeln;
write('Pulse enter para continuar');
readln
end.

```

3.2. Capítulo 3

```
program Algoritmo3_1;
(*
 * Algoritmo 3.1. Calcular el perimetro de un cuadrilatero irregular (version 1)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 3. Acciones y funciones
 *)

uses crt;

type
  Coordenada = real;
  Punto = record
    x : Coordenada;
    y : Coordenada;
  end;
var
  p1, p2, p3, p4 : Punto;
  po, pd : Punto;
  dist, perimetro : real;

procedure Distancia;
(* PRE po y pd almacenan dos puntos *)
(* POST dist almacena la distancia entre los puntos po y pd *)
begin
  dist := sqrt(sqr(pd.x - po.x) + sqr(pd.y - po.y))
end;

procedure LeerPunto(var x, y : real; n : shortint);
begin
  write(' Coordenada x del punto ', n, ': ');
  readln(x);
  write(' Coordenada y del punto ', n, ': ');
  readln(y);
  writeln
end;

begin
  clrscr;
  writeln;
```

```

writeln('*****');
writeln('***** Algoritmo 3.1. Calcular el perimetro de un *****');
writeln('*****          cuadrilatero irregular (version 1) *****');
writeln('*****');
writeln;
writeln('Introduzca los puntos del cuadrilatero (en orden de adyacencia):');
writeln;
LeerPunto(p1.x, p1.y, 1);
LeerPunto(p2.x, p2.y, 2);
LeerPunto(p3.x, p3.y, 3);
LeerPunto(p4.x, p4.y, 4);
perimetro := 0.0;
(* E0 : perimetro = 0 *)

po := p1;
pd := p2;
Distancia;
perimetro := dist;
(* E1 : perimetro = Distancia (p1, p2) *)

po := p2;
pd := p3;
Distancia;
perimetro := perimetro + dist;
(* E2 : perimetro = Distancia (p1, p2) + Distancia (p2, p3) *)

po := p3;
pd := p4;
Distancia;
perimetro := perimetro + dist;
(* E3 : perimetro = Distancia (p1, p2) + Distancia (p2, p3) + Distancia (p3, p4) *)

po := p4;
pd := p1;
Distancia;
perimetro := perimetro + dist;
(* E4 : perimetro = Distancia (p1, p2) + Distancia (p2, p3) +
          Distancia (p3, p4) + Distancia (p4, p1) *)

writeln('El perimetro del cuadrilatero irregular es: ', perimetro:1:2);
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo3_2;
(*

```

```
* Algoritmo 3.2. Calcular el perimetro de un cuadrilatero irregular (version 2)
* Titulo del libro: Una introduccion a la programacion.
*                               Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*                               Francisco J. Montoya Dato
*                               Jose L. Fernandez Aleman
*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 3. Acciones y funciones
*)

uses crt;

type
    Coordenada = real;
    Punto = record
        x : Coordenada;
        y : Coordenada;
    end;
var
    p1, p2, p3, p4 : Punto;
    d1, d2, d3, d4 : real;
    perimetro : real;

procedure Distancia(po, pd : Punto; var dist : real);
    (* PRE po y pd almacenan dos puntos *)
    (* POST dist retorna la distancia entre los puntos po y pd *)
begin
    dist := sqrt(sqr(pd.x - po.x) + sqr(pd.y - po.y))
end;

procedure LeerPunto(var x, y : real; n : shortint);
begin
    write(' Coordenada x del punto ', n, ':');
    readln(x);
    write(' Coordenada y del punto ', n, ':');
    readln(y);
    writeln
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.2. Calcular el perimetro de un *****');
    writeln('*****                               cuadrilatero irregular (version 2) *****');
```

```

    writeln('*****');
    writeln;

    writeln('Introduzca los puntos del cuadrilatero:');
    writeln;
    LeerPunto(p1.x, p1.y, 1);
    LeerPunto(p2.x, p2.y, 2);
    LeerPunto(p3.x, p3.y, 3);
    LeerPunto(p4.x, p4.y, 4);

    Distancia(p1, p2, d1);
    Distancia(p2, p3, d2);
    Distancia(p3, p4, d3);
    Distancia(p4, p1, d4);
    perimetro := d1 + d2 + d3 + d4;

    writeln('El perimetro del cuadrilatero irregular es:', perimetro:1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo3_3;
(*
  * Algoritmo 3.3. Accion para el intercambio de dos variables
  * Titulo del libro: Una introduccion a la programacion.
  *
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 3. Acciones y funciones
  *)

uses crt;

var
    x, y: integer;

procedure Intercambiar(var a, b : integer);
(* PRE a, b : Entero, a = A, b = B *)
(* POST a = B, b = A *)
var
    t : integer;
begin
    t := a;

```

```
    a := b;
    b := t
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.3. Accion para el intercambio de dos variables *****');
    writeln('*****');
    writeln;
    write('Valor inicial de x: ');
    readln(x);
    write('Valor inicial de y: ');
    readln(y);
    writeln;
    Intercambiar(x, y);
    writeln('Valor final de x: ', x);
    writeln('Valor final de y: ', y);
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo3_4;
(*
  * Algoritmo 3.4. Desplazamiento circular de tres variables
  * Título del libro: Una introduccion a la programacion.
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 3. Acciones y funciones
  *)

uses crt;

var
    a, b, c : integer;

procedure Intercambiar(var x, y : integer);
(* PRE x, y : Entero, x = X, y = Y *)
(* POST x = Y, y = X *)
var
```

```

    t : integer;
begin
    t := x;
    x := y;
    y := t
end;
begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.4. Desplazamiento circular de tres variables *****');
    writeln('*****');
    writeln;
    (* PRE a, b, c : Entero, a = A, b = B, c = C *)
    (* POST a, b, c : Entero, a = B, b = C, c = A *)
    writeln('Introduzca tres enteros separados por espacios');
    write('y termine pulsando enter: ');
    readln(a, b, c);
    writeln;
    writeln('Valor inicial de a, b y c: ', a, ', ', b, ', y ', c);
    (* E0 : a = A, b = B, c = C *)
    (* E0 = PRE *)

    Intercambiar(a, b);
    (* E1 : a = B, b = A, c = C *)

    Intercambiar(b, c);
    (* Ef : a = B, b = C, c = A => Post *)
    writeln;
    writeln('Desplazamiento circular a la izquierda');
    writeln;
    writeln('Valor final de a, b y c: ', a, ', ', b, ', y ', c);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo3_5;
(*
* Algoritmo 3.5. Algoritmo para dibujar dos cuadrados encajados
* Titulo del libro: Una introduccion a la programacion.
*
*           Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
*           Francisco J. Montoya Dato
*
*           Jose L. Fernandez Aleman
*
*           Maria J. Majado Rosales
* Fecha: 1/9/2005

```

```
* Capitulo 3. Acciones y funciones
*)

uses crt, graph3;

type
  Coordenada = integer; (* En Pascal las coordenadas son valores enteros *)
  Punto = record
    x : Coordenada;
    y : Coordenada;
  end;
  Cuadrado = record
    pos : Punto;
    lado, ang : integer
  end;
var
  ce : Cuadrado;

procedure DibujarCuadrado(c : Cuadrado);
  (* PRE Pantalla y pluma indiferentes *)
  (* POST "cuadrado c dibujado", PE = Baja, PP = c.pos, ORI = c.ang *)
var
  i : integer;
begin
  PenUp;
  SetPosition(c.pos.x, c.pos.y);
  SetHeading(c.ang);
  PenDown;
  for i := 1 to 4 do begin
    Forwd(c.lado);
    TurnRight(90);
  end;
end;

procedure Leer(var c : Cuadrado);
begin
  writeln(' CUADRADO');
  write(' Coordenada x del punto origen: ');
  readln(c.pos.x);
  write(' Coordenada y del punto origen: ');
  readln(c.pos.y);
  write(' Lado del cuadrado: ');
  readln(c.lado);
  write(' Angulo del cuadrado (grados): ');
  readln(c.ang);
  writeln;
```

```

end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.5. Algoritmo para dibujar dos cuadrados encajados *****'
    );
    writeln('*****');
    writeln;
    Leer(ce);
    GraphMode;      (* 320 x 200 modo grafico blanco-negro *)
    ClearScreen;
    ShowTurtle;
    DibujarCuadrado(ce);
    ce.lado := ce.lado div 2;
    DibujarCuadrado(ce);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo3_6;
(*
 * Algoritmo 3.6. Calcular el perimetro de un cuadrilatero irregular (version 3)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *                   Francisco J. Montoya Dato
 *                   Jose L. Fernandez Aleman
 *                   Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 3. Acciones y funciones
 *)

uses crt;

type
    Coordenada = real;
    Punto = record
        x : Coordenada;
        y : Coordenada;
    end;

var
    p1, p2, p3, p4 : Punto;
    perimetro : real;

```

```
function Distancia(po, pd : Punto) : real;
(* PRE po y pd almacenan dos puntos *)
(* POST retorna la distancia entre los puntos po y pd *)
begin
    Distancia := sqrt(sqr(pd.x - po.x) + sqr(pd.y - po.y))
end;
procedure LeerPunto(var x, y : real; n : shortint);
begin
    write(' Coordenada x del punto ', n, ':');
    readln(x);
    write(' Coordenada y del punto ', n, ':');
    readln(y);
    writeln;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.6. Calcular el perimetro de un *****');
    writeln('*****          cuadrilatero irregular (version 3) *****');
    writeln('*****');
    writeln;

    writeln('Introduzca los puntos del cuadrilatero:');
    writeln;
    LeerPunto(p1.x, p1.y, 1);
    LeerPunto(p2.x, p2.y, 2);
    LeerPunto(p3.x, p3.y, 3);
    LeerPunto(p4.x, p4.y, 4);

    perimetro := Distancia(p1, p2) + Distancia(p2, p3) + Distancia(p3, p4) + Distancia(
        p4, p1);

    writeln('El perimetro del cuadrilatero irregular es:', perimetro:1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo3_7;
(*
* Algoritmo 3.7. Funcion que obtiene el mayor de tres numeros (version 1)
* Titulo del libro: Una introduccion a la programacion.
*           Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*           Francisco J. Montoya Dato
*)
```

```

*                               Jose L. Fernandez Aleman
*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 3. Acciones y funciones
*)

uses crt;

var
  x, y, z : longint;

function Max3(a, b, c : longint) : longint;
begin
  if (a >= b) and (a >= c) then Max3 := a
  else if (b >= a) and (b >= c) then Max3 := b
  else if (c >= a) and (c >= b) then Max3 := c
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 3.7. Funcion que obtiene el mayor de tres numeros (version
    1) *****');
  writeln('*****');
  writeln;

  writeln('Introduzca tres enteros separados por espacios');
  write('y termine pulsando enter: ');
  readln(x, y, z);
  writeln;
  writeln('El maximo de ', x, ', ', y, ' y ', z, ' es: ', Max3(x, y, z));
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo3_8;
(*
* Algoritmo 3.8. Funcion que obtiene el mayor de tres numeros (version 2)
* Titulo del libro: Una introduccion a la programacion.
*                               Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*                               Francisco J. Montoya Dato
*                               Jose L. Fernandez Aleman
* í                               Mara J. Majado Rosales
* Fecha: 1/9/2005

```

```
* Capitulo 3. Acciones y funciones
*)
uses crt;

var
    x, y, z : longint;

function Max2 (a, b : longint) : longint;
begin
    Max2 := (a + b + abs (a - b)) div 2
end;

function Max3 (a, b, c : longint) : longint;
begin
    Max3 := Max2( a, Max2 (b, c));
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.8. Funcion que obtiene el mayor de *****');
    writeln('*****          tres numeros (version 2)          *****');
    writeln('*****');
    writeln;
    writeln('Introduzca tres enteros separados por espacios');
    write('y termine pulsando enter: ');
    readln (x, y, z);
    writeln;
    writeln('El maximo de ', x, ', ', y, ', y ', z, ' es: ', Max3 (x, y, z));
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo3_9;
(*
* Algoritmo 3.9. Calcular la suma de dos duraciones de tiempo (version 1)
* Titulo del libro: Una introduccion a la programacion.
*          Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*          Francisco J. Montoya Dato
*          Jose L. Fernandez Aleman
*          Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 3. Acciones y funciones
*)
```

```

uses crt;

(*$R+*) (* Habilita la comprobacion de intervalos *)
{(*$Q+*) Habilita la comprobacion de desbordamiento (solo en version 7.0)}

type
    Duracion = record
        hora : longint;
        min, seg : 0..59
    end;
var
    t1, t2 : Duracion; (* entrada de datos *)
    t3 : Duracion; (* salida de datos *)

procedure SumarDuraciones(d1, d2 : Duracion; var res : Duracion);
(* La funcion SumarDuraciones mostrada en el libro se codifica
   mediante un procedimiento con un parametro paso por referencia,
   porque en Pascal una funcion no puede devolver un valor de un
   tipo registro *)
var
    x, y : 0..1;
begin
    y := (d1.seg + d2.seg) div 60; (* acarreo de segundos *)
    x := (d1.min + d2.min + y) div 60; (* acarreo de minutos *)
    res.hora := d1.hora + d2.hora + x;
    res.min := (d1.min + d2.min + y) mod 60;
    res.seg := (d1.seg + d2.seg) mod 60;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.9. Calcular la suma de dos *****');
    writeln('***** duraciones de tiempo (version 1) *****');
    writeln('*****');
    writeln;
    writeln('Introduzca dos duraciones:');
    writeln;
    write(' Horas de la duracion 1: ');
    readln(t1.hora);
    write(' Minutos de la duracion 1: ');
    readln(t1.min);
    write(' Segundos de la duracion 1: ');
    readln(t1.seg);
    writeln;
    write(' Horas de la duracion 2: ');

```

```

    readln(t2.hora);
    write(' Minutos de la duracion 2: ');
    readln(t2.min);
    write(' Segundos de la duracion 2: ');
    readln(t2.seg);

    SumarDuraciones(t1, t2, t3);

    writeln;
    writeln('La suma de las duraciones es: ',
            t3.hora, '-', t3.min, '-', t3.seg);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo3_10;
(*
 * Algoritmo 3.10. Calcular la suma de dos duraciones de tiempo (version 2)
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 3. Acciones y funciones
 *)

uses crt;

(*$R*) (* Habilita la comprobacion de intervalos *)
(*$Q*) (* Habilita la comprobacion de desbordamiento (solo en version 7.0) *)

type
    Duracion = record
        hora : longint;
        min, seg : 0..59
    end;
var
    t1, t2 : Duracion; (* entrada de datos *)
    t3 : Duracion; (* salida de datos *)

procedure SumarDuraciones(a, b : Duracion; var res : Duracion);
(* Las funciones SumarDuraciones y ConvertirSD del Algoritmo 3.10
se codifican mediante un procedimiento con un parametro paso por
referencia, porque en Pascal una funcion no puede devolver un

```

```

valor de un tipo registro *)
function ConvertirDS(d : Duracion) : longint;
begin
    ConvertirDS := 3600 * d.hora + 60 * d.min + d.seg
end;
procedure ConvertirSD(n : longint; var res : Duracion);
var
    d : Duracion;
    rh : 0..3599;
begin
    d.hora := n div 3600;
    rh := n mod 3600;
    d.min := rh div 60;
    d.seg := rh mod 60;
    res := d
end;

begin (* de SumarDuraciones *)
    ConvertirSD(ConvertirDS(a) + ConvertirDS(b), res);
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 3.10. Calcular la suma de dos      *****');
    writeln('*****                      duraciones de tiempo (version 2) *****');
    writeln('*****');
    writeln;

    writeln('Introduzca dos duraciones:');
    writeln;
    write(' Horas de la duracion 1: ');
    readln(t1.hora);
    write(' Minutos de la duracion 1: ');
    readln(t1.min);
    write(' Segundos de la duracion 1: ');
    readln(t1.seg);
    writeln;
    write(' Horas de la duracion 2: ');
    readln(t2.hora);
    write(' Minutos de la duracion 2: ');
    readln(t2.min);
    write(' Segundos de la duracion 2: ');
    readln(t2.seg);

    SumarDuraciones(t1, t2, t3);

```

```
writeln;
writeln('La suma de las duraciones es: ',
        t3.hora, '-', t3.min, '-', t3.sec);
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo3_11;
(*
 * Algoritmo 3.11. Comprobar si una fecha es valida (version 3)
 * Titulo del libro: Una introduccion a la programacion.
 *               Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *                   Francisco J. Montoya Dato
 *                   Jose L. Fernandez Aleman
 *                   Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 3. Acciones y funciones
 *)

uses crt;

(*$R+*) (* Habilita la comprobacion de intervalos *)
(* $Q+ Habilita la comprobacion de desbordamiento (solo en version 7.0) *)

const
    diasMes : array [1..12] of integer =
        (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);

type
    Fecha = record
        dia : 1..31;
        mes : 1..12;
        anyo : word;
    end;

var
    f : Fecha; (* fecha dia/mes/anyo *)

function AnyoBisiesto(anyo : integer) : boolean;
    (* PRE anyo es un entero *)
    (* POST AnyoBisiesto es verdad si anyo es bisiesto
        y falso en caso contrario *)
begin
    AnyoBisiesto := (anyo mod 4 = 0) and (anyo mod 100 <> 0) or
        (anyo mod 400 = 0) and (anyo <> 3600);
```

```
end;

begin
    clrscr;
    writeln('Introduzca una fecha');
    writeln;
    write('Introduzca el dia: ');
    readln(f.dia);
    write('Introduzca el mes: ');
    readln(f.mes);
    write('Introduzca el anyo: ');
    readln(f.anyo);

    if AnyoBisiesto(f.anyo) then diasMes[2] := 29;
    write(f.dia, '/', f.mes, '/', f.anyo, ' ');
    if (f.dia <= diasMes[f.mes]) then writeln('es una fecha valida')
    else writeln('no es una fecha valida');
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

3.3. Capítulo 4

```

program Algoritmo4_1;
(*
  * Algoritmo 4.1. Calcular el valor medio de las notas de una asignatura
  * Título del libro: Una introduccion a la programacion.
  *
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
  v : real;           (* nota leida *)
  s : real;           (* lleva cuenta de la suma total *)
  n : integer;        (* lleva cuenta del numero de notas *)

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.1. Calcular el valor medio de *****');
  writeln('*****           las notas de una asignatura *****');
  writeln('*****');
  writeln;
  s := 0;
  n := 0;
  write('Introduzca una nota (valor <0 para terminar): ');
  readln(v);
  (* E0 : s = 0 y n = 0 *)
  while v >= 0 do begin
    s := s + v;
    n := n + 1;
    write('Introduzca una nota (valor <0 para terminar): ');
    readln(v);
    (* Ei : s = suma de las n notas introducidas hasta este paso
      y n = numero de notas introducidas hasta este paso *)
  end;
  (* Ef : s = suma de todas las notas y n = numero total de
    notas introducidas *)
  if n > 0 then writeln('Valor medio de las notas = ', s/n:1:2)
  else writeln('No hay valores');

```

```

    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo4_2;
(*
 * Algoritmo 4.2. Calcular el numero de ceros y unos en una secuencia
 * de caracteres ceros y unos
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 4. La iteracion
 * Fichero de entrada: datos4_2.TXT
 *)

uses crt, unitmsc1;

var
    s : msc1;                (* secuencia de enteros *)
    numCeros : integer;      (* contador del numero de ceros *)
    numUnos : integer;       (* contador del numero de unos *)
    c : char;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.2. Calcular el numero de ceros      *****');
    writeln('*****                y unos en una secuencia de caracteres *****');
    writeln('*****');
    writeln;
    TratamientoInicial_MSC1(s);
    Cargar_Fichero_MSC1(s, 'datos4_2.TXT');
    numCeros := 0;
    numUnos := 0;
    write('La secuencia de entrada es: ');
    Comenzar_MSC1(s);
    while (EA_MSC1(s) <> MSC1_MarcaFin) do begin
        write(EA_MSC1(s), ' ');
        if EA_MSC1(s) = '0' then numCeros := numCeros + 1
        else numUnos := numUnos + 1;
        Avanzar_MSC1(s);
    end;

```

```
end;  
writeln;  
writeln(' Numero de ceros = ', numCeros);  
writeln(' Numero de unos = ', numUnos);  
writeln;  
write('Pulse enter para continuar');  
readln  
end.
```

```
program Algoritmo4_3;  
(*  
  * Algoritmo 4.3. Crear una secuencia de caracteres a partir de otra existente  
  * Titulo del libro: Una introduccion a la programacion.  
  *           Un enfoque algoritmico  
  * Autores del libro: Jesus J. Garcia Molina  
  *           Francisco J. Montoya Dato  
  *           Jose L. Fernandez Aleman  
  *           Maria J. Majado Rosales  
  * Fecha: 1/9/2005  
  * Capitulo 4. La iteracion  
  * Fichero de prueba: datos4_3.txt  
  *)  
  
uses crt, unitmsc1;  
  
var  
  s, t : msc1;  
  
begin  
  clrscr;  
  writeln;  
  writeln('*****');  
  writeln('***** Algoritmo 4.3. Crear una secuencia de caracteres *****');  
  writeln('*****           a partir de otra existente           *****');  
  writeln('*****');  
  writeln;  
  TratamientoInicial_MSC1(s);  
  TratamientoInicial_MSC1(t);  
  Cargar_Fichero_MSC1(s, 'datos4_3.TXT');  
  write('La secuencia de entrada es: ');  
  Comenzar_MSC1(s);  
  Arrancar_MSC1(t);  
  while EA_MSC1(s) <> MSC1_MarcaFin do begin  
    if EA_MSC1(s) = '*' then Registrar_MSC1(t, '+')  
    else Registrar_MSC1(t, EA_MSC1(s));  
    write(EA_MSC1(s));  
    Avanzar_MSC1(s);
```

```

end;
Marcar_MSC1(t);
Salvar_Fichero_MSC1(t, 'sal4_3.TXT');
writeln;
write('La secuencia de salida es: ');
Comenzar_MSC1(t);
while EA_MSC1(t) <> MSC1_MarcaFin do begin
    write(EA_MSC1(t));
    Avanzar_MSC1(t);
end;
writeln;
writeln('Grabado');
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo4_4;
(*
  * Algoritmo 4.4. Calcular el valor medio de una secuencia con las notas de una
  * asignatura
  * Titulo del libro: Una introduccion a la programacion.
  * Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  * Francisco J. Montoya Dato
  * Jose L. Fernandez Aleman
  * Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  * Fichero de entradas: datos4_4.txt
  *)

uses crt, unitmsr1;

var
  s : msr1;
  suma : real; (* lleva cuenta de la suma total de las notas *)
  numElem : integer; (* lleva cuenta del numero de notas *)

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.4. Calcular el valor medio de una secuencia *****');
  writeln('***** con las notas de una asignatura *****');
  writeln('*****');
  writeln;

```

```

    TratamientoInicial_MSR1(s);
    Cargar_Fichero_MSR1(s, 'datos4_4.txt');
    write('La secuencia de entrada es: ');
    Comenzar_MSR1(s);
    suma := 0.0;
    numElem := 0;

(* Eini : suma = 0 y numElem = 0 y ea = Primero (S) y INV = Verdadero *)
    while EA_MSR1(s) <> MSR1_MarcaFin do begin
        (* INV 1 <= i <= Long (Piz) y suma tiene el *)
        (* sumatorio desde i=1 hasta Long(Piz) de Si *)
        (* y numElem = Long (Piz) y (EA (S) <> MarcaFin) *)
            suma := suma + EA_MSR1(s);
            numElem := numElem + 1;
            write(EA_MSR1(s):1:2, ' ');
            Avanzar_MSR1(s);
        end;
(* Efin : INV y (EA (S) = MarcaFin) *)
    writeln;
    writeln;
    if (numElem > 0) then write('Valor medio de las notas = ', suma/numElem:1:2)
    else write('Secuencia vacia');
    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo4_5;
(*
 * Algoritmo 4.5. Calcular el numero de pares ceros-unos en una secuencia de caracteres
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 * Jose L. Fernandez Aleman
 * Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 4. La iteracion
 * Fichero de entrada: datos4_5.txt
 *)

uses crt, unitmscl;

var
    s : MSC1;
    numCeros : integer; (* lleva cuenta del numero de '0' *)

```

```

numPares : integer; (* lleva cuenta del numero de pares *)

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.5. Calcular el numero de pares ceros-unos *****');
  writeln('*****          en una secuencia de caracteres          *****');
  writeln('*****');
  writeln;
  TratamientoInicial_MSC1(s);
  Cargar_Fichero_MSC1(s, 'datos4_5.txt');
  write('La secuencia de entrada es: ');
  Comenzar_MSC1(s);
  numCeros := 0;
  numPares := 0;
  (* INV y (EA = Primero (S)) *)
  while EA_MSC1(s) <> MSC1_MarcaFin do begin
    (* INV : numPares = NumPares (Piz), numCeros = NumCeros (Piz) *)
    (* INV y (EA_MSC1 <> MarcaFin) *)
    case EA_MSC1(s) of
      '0': numCeros := numCeros + 1;
      '1': numPares := numPares + numCeros
    end;
    write(EA_MSC1(s));
    Avanzar_MSC1(s)
  end;
  writeln;
  writeln;
  (* INV y (EA = MarcaFin), numPares = NumPares (S) *)
  writeln('El numero de pares 0-1 es: ', numPares);
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo4_6;
(*
  * Algoritmo 4.6. Calculo del factorial (version 1)
  * Titulo del libro: Una introduccion a la programacion.
  *          Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *          Francisco J. Montoya Dato
  *          Jose L. Fernandez Aleman
  *          Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion

```

```
*)

uses crt;

var
  n, i : integer;
  factorial : longint;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.6. Calculo del factorial (version 1) *****');
  writeln('*****');
  writeln;
  write('Introduzca un entero (< 17): ');
  readln(n);
  i := 0;
  factorial := 1;
  while i < n do begin
    (* INV: factorial = i! *)
    i := i + 1;
    factorial := factorial * i
  end;
  writeln;
  writeln('El factorial de ', n, ' es: ', factorial);
  writeln;
  write('Pulse enter para continuar');
  readln
end.
```

```
program algoritmo4_7;
(*
  * Algoritmo 4.7. Calculo del factorial (version 2)
  * Titulo del libro: Una introduccion a la programacion.
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
```

```

n, i : integer;
factorial : longint;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.7. Calculo del factorial (version 2) *****');
    writeln('*****');
    writeln;
    write('Introduzca un entero (< 17): ');
    readln(n);
    factorial := 1;
    for i:= 2 to n do
        factorial := factorial * i;
        (* INV factorial = i! *)
    writeln;
    writeln('El factorial de ', n, ' es: ', factorial);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo4_8;
(*
  * Algoritmo 4.8. Calculo del producto mediante sumas sucesivas
  * Titulo del libro: Una introduccion a la programacion.
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
    a, b, producto : longint;
    i : longint;

begin
    clrscr;
    writeln;
    writeln('*****');

```

```
writeln('***** Algoritmo 4.8. Calculo del producto mediante sumas sucesivas *****')
;
writeln('*****');
writeln;
write('Introduzca dos numeros enteros separados por blancos: ');
readln (a, b);
producto := 0;
for i := 1 to b do
    producto := producto + a;
    (* INV producto = a * i y 1 <= i <= b *)
    (* INV y (i = b), producto = a * b = POST *)
writeln;
writeln(' ', a, '*', b, '=', producto);
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo4_9;
(*
* Algoritmo 4.9. Calculo del cociente y resto de una division entera
* Titulo del libro: Una introduccion a la programacion.
*
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
* Francisco J. Montoya Dato
*
* Jose L. Fernandez Aleman
*
* Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 4. La iteracion
*)

uses crt;

var
    p, c, r : longint;
    q : longint;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.9. Calculo del cociente y resto de una division entera
    *****');
    writeln('*****');
    writeln;
    write('Introduzca dividendo (>= 0) y divisor (> 0) separados por blancos: ');
    readln(p, q);
```

```

r := p; c := 0;
while r >= q do begin
  (* INV (p = q * c + r) y (r >= 0) *)
  r := r - q;
  c := c + 1;
end;
(* INV y (r < q) => POST *)
writeln;
writeln('Cociente = ', c, ' Resto = ', r);
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo4_10;
(*
  * Algoritmo 4.10. Calculo del maximo comun divisor
  * Titulo del libro: Una introduccion a la programacion.
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
  a, b, u, v : word;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.10. Calculo del maximo comun divisor *****');
  writeln('*****');
  writeln;
  write('Introduzca dos enteros (> 0) separados por blancos: ');
  readln(a, b);
  u := a; v := b;
  while u <> v do begin
    (* INV (* mcd (a,b) = mcd (u,v) *)
      if (u > v) then u := u - v
      else v := v - u;
    end;

```

```

    (* INV y (u = v) *)
    writeln;
    writeln('Maximo comun divisor de ', a, ' y ', b, ' es: ', u);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo4_11;
(*
  * Algoritmo 4.11. Calculo de las potencias de 2 (version 1)
  * Titulo del libro: Una introduccion a la programacion.
  *               Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *               Francisco J. Montoya Dato
  *               Jose L. Fernandez Aleman
  *               Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
  x : integer;      (* indice inferior del intervalo de potencias *)
  y : integer;      (* indice superior del intervalo *)
  i : integer;

function Potencia2(n : integer):longint;
(* PRE: n : Entero >= 0 *)
(* POST: Potencia2 = 2^n *)
var
  p : longint;
  j : integer;
begin
  p := 1;
  for j := 1 to n do
    p := p * 2;
    (* INV p = 2^j *)
  Potencia2 := p
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.11. Calculo de las potencias de 2 (version 1) *****');

```

```

writeln('*****');
writeln;
write('Introduzca dos enteros (>= 0) en orden creciente separados por blancos: ');
readln(x, y);
writeln;
for i := x to y do
    writeln('La potencia ', i, '-esima de 2 es ', Potencia2(i));
    (* INV se han calculado y mostrado las potencias de 2
       en el intervalo [x,i] *)
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo4_12;
(*
 * Algoritmo 4.12. Calculo de las potencias de 2 (version 2)
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 * Jose L. Fernandez Aleman
 * Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 4. La iteracion
 *)

uses crt;

var
    x : integer;          (* indice inferior *)
    y : integer;          (* indice superior *)
    potencia2 : longint;   (* resultado *)
    i : integer;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.12. Calculo de las potencias de 2 (version 2) *****');
    writeln('*****');
    writeln;
    write('Introduzca dos enteros (>= 0) en orden creciente separados por blancos: ');
    readln(x, y);
    writeln;
    potencia2 := 1;
    for i:= 1 to x do

```

```

    potencia2 := potencia2 * 2;
    (* INV potencia2 = 2^i *)
    writeln('La potencia ', x, '-esima de 2 es ', potencia2);
    for i:= x+1 to y do begin
        potencia2 := potencia2 * 2;
        writeln('La potencia ', i, '-esima de 2 es: ', potencia2)
        (* INV potencia2 = 2^i y se han mostrado las potencias de 2
           en el intervalo [x,i] *)
    end;
    writeln;
    write('Pulse enter para continuar');
    readln

end.

```

```

program Algoritmo4_13;
(*
 * Algoritmo 4.13. Calculo del termino enesimo de la sucesion de Fibonacci
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 4. La iteracion
 *)

uses crt;

var
    ult, penult : longint;      (* ultimo y penultimo termino de Fibonacci *)
    copiaFibo : longint;       (* valor Fibonacci en el paso i-1 *)
    fibo : longint;            (* resultado *)
    n : integer;               (* dato *)
    i : integer;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.13. Calculo del termino enesimo *****');
    writeln('*****           de la sucesion de Fibonacci *****');
    writeln('*****');
    writeln;
    write('Introduzca un entero (>= 0): ');
    readln(n);

```

```

writeln;
case n of
  0 : writeln('El termino ',n,'-esimo de la sucesion (posicion ', n+1,') es: 1'
    );
  1 : writeln('El termino ',n,'-esimo de la sucesion (posicion ', n+1,') es: 1'
    );
  else penult := 1; ult := 1; fibo := 2;
      for i:=3 to n do begin
        copiaFibo := fibo;
        fibo := fibo + ult;
        penult := ult;
        ult := copiaFibo
        (* INV (fibo = f(i)) y (ult = f(i-1)) y (penult = f(i-2)),
          3 <= i <= n *)
      end;
      (* INV y (i = n) *)
      writeln('El termino ',n,'-esimo de la sucesion (posicion ', n+1,') es: ',
        fibo)
end;
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo4_14;
(*
  * Algoritmo 4.14. Comprobar si un numero es primo (version 1)
  * Titulo del libro: Una introduccion a la programacion.
  *                   Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
  n : longint;      (* dato *)
  j : integer;

begin
  clrscr;
  writeln;
  writeln('*****');

```

```
writeln('***** Algoritmo 4.14. Comprobar si un numero es primo (version 1) *****');
writeln('*****');
writeln;
write('Introduzca un entero (> 0): ');
readln(n);
writeln;
if n = 1 then writeln ('1 es primo')
else begin
    j := 2;
    while (n mod j) <> 0 do
        j := j + 1;
        (* INV ( i : 2 <= i < j : (n MOD i) <> 0) *)
        (* INV y (n MOD j = 0) *)
    if j = n then writeln (n, ' es primo')
    else writeln (n, ' no es primo')
end;
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo4_15;
(*
 * Algoritmo 4.15. Comprobar si un numero es primo (version 2)
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 * Jose L. Fernandez Aleman
 * Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 4. La iteracion
 *)

uses crt;

var
    n : longint;
    j : integer;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.15. Comprobar si un numero es primo (version 2) *****');
    writeln('*****');
    writeln;
```

```

write('Introduzca un entero (> 0): ');
readln(n);
writeln;
if (n = 1) or (n = 2) or (n = 3) then writeln(n, ' es primo')
else if (n > 3) and ((n mod 2) = 0) then writeln(n, ' es par, no es primo')
else if (n > 3) and ((n mod 2) <> 0) then begin
    j := 3;
    while ((n mod j) <> 0) and (j < (n div j)) do
        j := j + 2;
        (* INV (i : 3 <= i < j : (n MOD i) <> 0) *)
        (* INV y (n MOD j = 0) o (j = (n DIV j))* *)
    if (n mod j) <> 0 then writeln(n, ' es primo')
    else writeln(n, ' no es primo');
end;
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo4_16;
(*
* Algoritmo 4.16. Calcular la parte entera de la raiz cuadrada de un entero (version
  1)
* Titulo del libro: Una introduccion a la programacion.
*
  Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
  Francisco J. Montoya Dato
*
  Jose L. Fernandez Aleman
*
  Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 4. La iteracion
*)

uses crt;

var
  n, rc : longint; (* n dato de entrada, rc parte entera de la raiz *)
  i : longint;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.16. Calcular la parte entera de la raiz cuadrada *****');
  ;
  writeln('***** de un entero (version 1) *****');
  writeln('*****');

```

```
writeln;
write('Introduzca un entero: ');
readln (n);
writeln;
i := 1;
while (n >= i * i) do
    (* INV ( j : 1 <= j < i : (j^2 <= n)) *)
    i := i + 1;
    (* INV y (n < i^2) *)
rc := i - 1;
writeln('La parte entera de la raiz cuadrada de ', n, ' es: ', rc);
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo4_17;
(*
* Algoritmo 4.17. Calcular la parte entera de la raiz cuadrada de un entero (version
* 2)
* Titulo del libro: Una introduccion a la programacion.
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
* Francisco J. Montoya Dato
* Jose L. Fernandez Aleman
* Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 4. La iteracion
*)

uses crt;

var
    n, rc : longint;          (* n dato, rc resultado *)
    imp, i, c : longint;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.17. Calcular la parte entera de la raiz cuadrada *****');
    ;
    writeln('***** de un entero (version 2) *****');
    writeln('*****');
    writeln;
    write('Introduzca un entero (>= 0): ');
    readln (n);
```

```

writeln;
i := 1; c := 1; imp := 1;
while c <= n do begin
  (* INV (j : 1 <= j < i : (j^2 <= n) <> 0) y (c = i^2) y
    c es el i-esimo cuadrado, e imp es el i-esimo impar *)
  imp := imp + 2;
  c := c + imp;
  i := i + 1
end;
(* INV y (n < c) *)
rc := i - 1;
writeln('La parte entera de la raiz cuadrada de ',n,' es ',rc);
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo4_18;
(*
  * Algoritmo 4.18. Calcular 1/x a partir de una sucesion recurrente
  * Titulo del libro: Una introduccion a la programacion.
  *
  * Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
  * Francisco J. Montoya Dato
  * Jose L. Fernandez Aleman
  * Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 4. La iteracion
  *)

uses crt;

var
  x : real;          (* dato entre 0 y 1 *)
  a : real;          (* resultado *)
  c : real;          (* para el calculo de la sucesion *)
  eps : real;        (* error *)

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 4.18. Calcular 1/x a partir de una sucesion recurrente
    *****');
  writeln('*****');
  writeln;
  write('Introduzca un valor real (<= 1) y (> 0): ');

```

```

    readln (x);
    writeln;
    write('Introduzca el error permitido: ');
    readln(eps);
    a := 1; c := 1 - x;
    while (c > eps) do begin
        a := a * (1 + c);
        c := c * c;
    end;
    writeln;
    writeln('El resultado de evaluar la expresion 1/',x, ' es: ',a);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo4_19;
(*
 * Algoritmo 4.19. Aproxima e elevado a x a partir de su desarrollo en serie
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 4. La iteracion
 *)

uses crt;

var
    x : integer;      (* dato *)
    i : integer;      (* numero de termino *)
    t, s : real;      (* termino y suma de la serie *)
    eps : real;       (* error *)

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 4.19. Aproxima e elevado a x a partir *****');
    writeln('*****           de su desarrollo en serie           *****');
    writeln('*****');
    writeln;
    write('Introduzca un entero: ');
    readln(x);

```

```
writeln;  
write('Introduzca el error permitido: ');  
readln(eps);  
t := 1; s := t; i := 0;  
while t > eps do begin  
    i := i + 1;  
    t := t * (x / i);  
    s := s + t;  
end;  
writeln;  
writeln('e elevado a ',x,' es: ',s);  
writeln;  
write('Pulse enter para continuar');  
readln  
end.
```

3.4. Capítulo 5

```
program Algoritmo5_1;
(*
 * Algoritmo 5.1. Calcular el numero de pares (primer modelo, tercer esquema)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 5. Tecnicas de disenyo de algoritmos
 * Fichero de entrada: datos5_1.txt
 *)

uses crt, unitmse1;

var
  a, b : integer;
  numA, numB : integer;
  numPares : integer;
  S : mse1;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 5.1. Calcular el numero de pares *****');
  writeln('***** (primer modelo, tercer esquema) *****');
  writeln('*****');
  writeln;
  writeln('Introduzca un par de enteros distintos:');
  write(' primer entero: ');
  readln(a);
  write(' segundo entero: ');
  readln(b);
  writeln;
  write('La secuencia de entrada es: ');
  TratamientoInicial_MSE1(S);
  Cargar_fichero_MSE1(S, 'datos5_1.txt');
  Comenzar_MSE1(S);
  if EA_MSE1(S) = MSE1_MarcaFin then write ('Secuencia vacia')
  else begin
    numPares := 0;
    if EA_MSE1(S) = a then begin numA := 1; numB := 0 end
    else if EA_MSE1(S) = b then begin numA := 0; numB := 1 end
```

```

        else begin numA := 0; numB := 0 end;
(* Se utiliza la equivalencia ITERAR-MIENTRAS *)
    write(EA_MSE1(S), ' ');
    Avanzar_MSE1(S);
    while EA_MSE1(S) <> MSE1_MarcaFin do begin
        if EA_MSE1(S) = a then begin
            numPares := numPares + numB;
            numA := numA + 1
        end
        else if EA_MSE1(S) = b then begin
            numPares := numPares + numA;
            numB := numB + 1
        end;
        write(EA_MSE1(S), ' ');
        Avanzar_MSE1(S)
    end;
    writeln;
    writeln;
    writeln('El numero de pares (', a, ', ', b, '), (', b, ', ', a, ') es: ',
        numPares);
end;
writeln;
write ('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo5_2;
(*
* Algoritmo 5.2. Calcular el numero de pares (segundo modelo, tercer esquema)
* Titulo del libro: Una introduccion a la programacion.
*
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
* Francisco J. Montoya Dato
*
* Jose L. Fernandez Aleman
*
* Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 5. Tecnicas de disenyo de algoritmos
* Fichero de entrada: datos5_2.txt
*)

uses crt, unitmse2;

var
    a, b : integer;
    numA, numB : integer;
    numPares : integer;
    S : mse2;

```

```

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 5.2. Calcular el numero de pares *****');
  writeln('***** (segundo modelo, tercer esquema) *****');
  writeln('*****');
  writeln;
  writeln('Introduzca un par de enteros distintos:');
  write(' primer entero: ');
  readln(a);
  write(' segundo entero: ');
  readln(b);
  writeln;
  write('La secuencia de entrada es: ');
  TratamientoInicial_MSE2(s);
  Cargar_fichero_MSE2(S, 'datos5_2.txt');
  InicAcceso_MSE2(S);
  if EsVacia_MSE2(S) then write('Secuencia vacia')
  else begin
    Avanzar_MSE2(S);
    write(EA_MSE2(S), ' ');
    numPares := 0;
    if EA_MSE2(S) = a then begin numA := 1; numB := 0 end
    else if EA_MSE2(S) = b then begin numA := 0; numB := 1 end
    else begin numA := 0; numB := 0 end;
    while not EsUltimo_MSE2(S) do begin
      Avanzar_MSE2(S);
      write(EA_MSE2(S), ' ');
      if EA_MSE2(S) = a then begin
        numPares := numPares + numB;
        numA := numA + 1
      end
      else if EA_MSE2(S) = b then begin
        numPares := numPares + numA;
        numB := numB + 1
      end
    end;
    writeln;
    writeln;
    writeln('El numero de pares (', a, ', ', b, '), (', b, ', ', a, ') es: ',
      numPares);
  end;
  writeln;
  write('Pulse enter para continuar');
  readln

```

end.

```

program Algoritmo5_3;
(*
 * Algoritmo 5.3. Calcular el numero de pares (tercer modelo, segundo esquema)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 5. Tecnicas de disenyo de algoritmos
 * Fichero de entrada: datos5_3.txt
 *)

uses crt, unitmse1;

var
  a, b : integer;
  numA, numB : integer;
  numPares : integer;
  S : mse1;

(*
 * Maquina abstracta del tercer modelo a partir de
 * una maquina secuencial del primer modelo
 *)
  aux : integer;
procedure TratamientoInicial_MSE3(VAR S : mse1);
begin
  TratamientoInicial_MSE1(S);
end;

procedure Comenzar_MSE3(VAR S : mse1);
begin
  Comenzar_MSE1(S);
  aux := EA_MSE1 (S);
  Avanzar_MSE1(S);
end;

procedure Avanzar_MSE3(VAR S : mse1);
begin
  aux := EA_MSE1 (S);
  Avanzar_MSE1(S);
end;

```

```

function EA_MSE3(VAR S : mse1) : integer;
begin
    EA_MSE3 := aux;
end;

procedure Cargar_Fichero_MSE3(var S : mse1; nombre : string);
begin
    Cargar_Fichero_MSE1(S, nombre);
end;

function EsVacia_MSE3(S : mse1) : boolean;
begin
    EsVacia_MSE3 := EA_MSE1(S) = MSE1_MarcaFin;
end;

function EsUltimo_MSE3(S : mse1) : boolean;
begin
    EsUltimo_MSE3 := EA_MSE1(S) = MSE1_MarcaFin
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 5.3. Calcular el numero de pares *****');
    writeln('***** (tercer modelo, segundo esquema) *****');
    writeln('*****');
    writeln;
    writeln('Introduzca un par de enteros distintos:');
    write(' primer entero: ');
    readln(a);
    write(' segundo entero: ');
    readln(b);
    writeln;
    write('La secuencia de entrada es: ');
    TratamientoInicial_MSE3(s);
    Cargar_fichero_MSE3(S, 'datos5_3.txt');
    Comenzar_MSE3(S);
    if EsVacia_MSE3(S) then write('Secuencia vacia')
    else begin
        numPares := 0;
        numA := 0;
        numB := 0;
        (* Se utiliza la equivalencia ITERAR-MIENTRAS *)
        if EA_MSE3(S) = a then begin
            numPares := numPares + numB;
            numA := numA + 1
        end
    end
end

```

```

end
else if EA_MSE3(S) = b then begin
    numPares := numPares + numA;
    numB := numB + 1
end;
write(EA_MSE3(S), ' ');
while not EsUltimo_MSE3(S) do begin
    Avanzar_MSE3(S);
    write(EA_MSE3(S), ' ');
    if EA_MSE3(S) = a then begin
        numPares := numPares + numB;
        numA := numA + 1
    end
    else if EA_MSE3(S) = b then begin
        numPares := numPares + numA;
        numB := numB + 1
    end;
end;
writeln;
writeln;
writeln('El numero de pares (', a, ', ', b, '), (', b, ', ', a, ') es: ',
    numPares);

end;
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo5_4;
(*
* Algoritmo 5.4. Calcular el numero de pares (cuarto modelo, segundo esquema)
* Titulo del libro: Una introduccion a la programacion.
*
*           Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
*           Francisco J. Montoya Dato
*
*           Jose L. Fernandez Aleman
*
*           Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 5. Tecnicas de disenyo de algoritmos
* Fichero de entrada: datos5_4.txt
*)

uses crt, unitmse1;

const
    MSE4_MarcaFin = MSE1_MarcaFin;
var

```

```
a, b : integer;
numA, numB : integer;
numPares : integer;
S : mse1;

(*
 * Maquina abstracta del cuarto modelo a partir de
 * una maquina secuencial del primer modelo
 *)
comenzar : boolean;
procedure TratamientoInicial_MSE4(VAR S : mse1);
begin
    TratamientoInicial_MSE1(S);
end;

procedure Iniciar_MSE4(VAR S : mse1);
begin
    Comenzar_MSE1(S);
    comenzar := false;
end;

procedure Avanzar_MSE4(VAR S : mse1);
begin
    if not comenzar then comenzar := true
    else Avanzar_MSE1(S);
end;

function EA_MSE4(VAR S : mse1) : integer;
begin
    EA_MSE4 := EA_MSE1 (S);
end;

procedure Cargar_Fichero_MSE4(var S : mse1; nombre : string);
begin
    Cargar_Fichero_MSE1(S, nombre);
end;

function EsVacia_MSE4(S : mse1) : boolean;
begin
    EsVacia_MSE4 := EA_MSE1(S) = MSE1_MarcaFin;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
```

```

writeln('***** Algoritmo 5.4. Calcular el numero de pares *****');
writeln('*****          (cuarto modelo, segundo esquema) *****');
writeln('*****');
writeln;
writeln('Introduzca un par de enteros distintos:');
write(' primer entero: ');
readln(a);
write(' segundo entero: ');
readln(b);
writeln;
write('La secuencia de entrada es: ');
TratamientoInicial_MSE4(s);
Cargar_fichero_MSE4(S, 'datos5_4.txt');
Iniciar_MSE4(S);
if EsVacia_MSE4(S) then write('Secuencia vacia')
else begin
    numPares := 0;
    numA := 0;
    numB := 0;
    Avanzar_MSE4(S);
    REPEAT
        if EA_MSE4(S) = a then begin
            numPares := numPares + numB;
            numA := numA + 1
        end
        else if EA_MSE4(S) = b then begin
            numPares := numPares + numA;
            numB := numB + 1
        end;
        write(EA_MSE4(S), ' ');
        Avanzar_MSE4 (S);
    UNTIL EA_MSE4(S) = MSE4_MarcaFin;
    writeln;
    writeln;
    writeln('El numero de pares (', a, ', ', b, '), (', b, ', ', a, ') es: ',
        numPares);
end;
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo5_5;
(*
* Algoritmo 5.5. Comprobar si hay una nota mayor o igual que 9 (primer modelo)
* Titulo del libro: Una introduccion a la programacion.
*          Un enfoque algoritmico

```

```
* Autores del libro: Jesus J. Garcia Molina
*                      Francisco J. Montoya Dato
*                      Jose L. Fernandez Aleman
*                      Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 5. Tecnicas de disenyo de algoritmos
* Fichero de entrada: datos5_5.txt
*)

uses crt, unitmsr1;
var
  S : msr1;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 5.5. Comprobar si hay una nota mayor o igual que 9 *****')
    ;
  writeln('*****                (primer modelo)                *****');
  writeln('*****');
  writeln;
  TratamientoInicial_MSR1(S);
  Cargar_fichero_MSR1(S, 'datos5_5.txt');
  write('La secuencia de entrada es: ');
  Comenzar_MSR1(S);
  while (EA_MSR1(S) <> MSR1_MarcaFin) do begin
    write(EA_MSR1(S):1:2, ' ');
    Avanzar_MSR1(S);
  end;
  Comenzar_MSR1(S);
  while (EA_MSR1(S) <> MSR1_MarcaFin) and not (EA_MSR1(S) >= 9) do begin
    Avanzar_MSR1(S);
  end;
  (* INV "No existe en Piz una nota superior o igual a 9" *)
  (* fin while *)
  writeln;
  writeln;
  if EA_MSR1(S) = MSR1_MarcaFin then
    write('No existe ninguna nota superior o igual a 9')
  else
    write('Al menos existe una nota superior o igual a 9');
  writeln;
  writeln;
  write('Pulse enter para continuar');
  readln
end.
```

```

program Algoritmo5_6;
(*
  * Algoritmo 5.6. Comprobar si hay una nota mayor o igual que 9 (segundo modelo)
  * Titulo del libro: Una introduccion a la programacion.
  *
    Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
    Francisco J. Montoya Dato
  *
    Jose L. Fernandez Aleman
  *
    Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 5. Tecnicas de disenyo de algoritmos
  * Fichero de entrada: datos5_6.txt
  *)

uses crt, unitmsr2;

var
  S : msr2;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 5.6. Comprobar si hay una nota mayor o igual que 9 *****');
  ;
  writeln('*****          (segundo modelo)          *****');
  writeln('*****');
  writeln;
  TratamientoInicial_MSR2(S);
  Cargar_fichero_MSR2(S, 'datos5_6.txt');
  write('La secuencia de entrada es: ');
  InicAcceso_MSR2(S);
  if not EsVacía_MSR2(S) then
    REPEAT
      (* INV "No existe en Piz una nota superior o igual a 9" *)
      Avanzar_MSR2(S);
      write(EA_MSR2(S):1:2, ' ');
    UNTIL (EsUltimo_MSR2(S));
  InicAcceso_MSR2(S);
  if EsVacía_MSR2(S) then write('Nadie se ha presentado al examen')
  else begin
    REPEAT
      (* INV "No existe en Piz una nota superior o igual a 9" *)
      Avanzar_MSR2(S);
    UNTIL EsUltimo_MSR2(S) or (EA_MSR2(S) >= 9);
    writeln;
    writeln;
  end;
end;

```

```
        if EA_MSR2(S) >= 9 then
            write('Al menos existe una nota superior o igual a 9')
        else
            write('No existe ninguna nota superior o igual a 9')
        end;
    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo5_7;
(*
 * Algoritmo 5.7. Calculo posicion de la primera palabra que comienza por 'a'
 *          (primer modelo, primer esquema)
 * Titulo del libro: Una introduccion a la programacion.
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *          Francisco J. Montoya Dato
 *          Jose L. Fernandez Aleman
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 5. Tecnicas de disenyo de algoritmos
 * Fichero de entrada: datos5_7.txt
 *)

uses crt, unitmscl;

const
    ESPACIO = ' ';
var
    posicion : integer;
    anterior : char;
    S : msc1;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 5.7. Calculo posicion de la primera palabra *****');
    writeln('*****          que comienza por "a" (primer modelo, primer esquema) *****');
    writeln('*****');
    writeln;
    write('La secuencia de entrada es: ');
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'datos5_7.txt');
```

```

Comenzar_MSC1 (S);
while (EA_MSC1 (S) <> MSC1_MarcaFin) do begin
    write(EA_MSC1(S));
    Avanzar_MSC1 (S)
end;
writeln;
Comenzar_MSC1 (S);
posicion := 0;
anterior := ESPACIO;
while (EA_MSC1 (S) <> MSC1_MarcaFin) and
    not ((EA_MSC1 (S) = 'a') and (anterior = ESPACIO)) do begin
    posicion := posicion + 1;
    anterior := EA_MSC1 (S);
    Avanzar_MSC1 (S)
(* INV "En Piz ninguna palabra empieza por 'a'" y posicion = Long (Piz) *)
end;
if EA_MSC1 (S) = MSC1_MarcaFin then
    write('El texto no contiene ninguna palabra que comience por la letra a')
else
    write('Posicion de la primera palabra que comienza por "a" es: ', posicion+1);
writeln;
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo5_8;
(*
* Algoritmo 5.8. Calculo posicion de la primera palabra que comienza por 'a'
* (primer modelo, tercer esquema)
* Titulo del libro: Una introduccion a la programacion.
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
* Francisco J. Montoya Dato
* Jose L. Fernandez Aleman
* Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 5. Tecnicas de disenyo de algoritmos
* Fichero de entrada: datos5_8.txt
*)

uses crt, unitmscl;

const
    ESPACIO = ' ';

var

```

```

    posicion : integer;
    anterior : char;
    S : msc1;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 5.8. Calculo posicion de la primera palabra      *****');
    writeln('*****                               que comienza por "a" (primer modelo, tercer esquema)
              *****');
    writeln('*****');
    writeln;
    write('La secuencia de entrada es: ');
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'datos5_8.txt');
    Comenzar_MSC1 (S);
    while (EA_MSC1 (S) <> MSC1_MarcaFin) do begin
        write(EA_MSC1(S));
        Avanzar_MSC1 (S)

    end;
    writeln;
    Comenzar_MSC1 (S);
    if EA_MSC1(S) = MSC1_MarcaFin then write('La secuencia esta vacia')
    else if EA_MSC1 (S) = 'a' then
        write('Posicion de la primera palabra que comienza por a es: ', 1)
    else begin
        posicion := 1;
        anterior := EA_MSC1 (S);
        (* Se utiliza la equivalencia ITERAR-MIENTRAS *)
        (* INV "En Piz ninguna palabra empieza por 'a'" y posicion = Long (Piz) *)
        Avanzar_MSC1 (S);
        while not ((EA_MSC1(S) = MSC1_MarcaFin) or
                    ((EA_MSC1(S) = 'a') and
                     (anterior = ESPACIO))) do begin
            posicion := posicion + 1;
            anterior := EA_MSC1(S);
            Avanzar_MSC1 (S);

        end;
        writeln;
        if EA_MSC1(S) = MSC1_MarcaFin then
            write('El texto no contiene ninguna palabra que comience por la letra
                  a')
        else
            write('Posicion de la primera palabra que comienza por "a" es: ',
                  posicion+1)

    end;
end;

```

```

    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo5_9;
(*
 * Algoritmo 5.9. Comprobar si todas las palabras acaban con la misma letra
 *           (primer modelo, sin secuencia intermedia)
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 5. Tecnicas de disenyo de algoritmos
 * Fichero de entrada: datos5_9.txt
 *)

uses crt, unitmsc1;

const
    ESPACIO = ' ';
var
    ultLetraPrimPal, anterior : char;
    S : msc1;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 5.9. Comprobar si todas las palabras acaban con la misma
        *****');
    writeln('*****          letra (primer modelo, sin secuencia intermedia) *****')
        ;
    writeln('*****');
    writeln;

    (* Esquema de recorrido y busqueda.
    Encontrar letra final de la primera palabra *)
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'datos5_9.txt');
    write('La secuencia de entrada es: ');
    Comenzar_MSC1 (S);
    while (EA_MSC1 (S) <> MSC1_MarcaFin) do begin

```

```

        write(EA_MSC1(S));
        Avanzar_MSC1 (S)
    end;
    writeln;
    Comenzar_MSC1(S);
    anterior := ESPACIO;
    while (EA_MSC1(S) <> MSC1_MarcaFin) and
        not ((EA_MSC1(S) = ESPACIO) and (anterior <> ESPACIO)) do begin
        anterior := EA_MSC1(S);
        Avanzar_MSC1(S)
        (* INV anterior = "ultimo elemento de Piz" y
           "no hay ningun par (letra, espacio) en Piz" *)
    end;
    writeln;
    if (EA_MSC1(S) = MSC1_MarcaFin) and (anterior = ESPACIO) then
        write('No hay palabras en la secuencia')
    else if (EA_MSC1(S) = MSC1_MarcaFin) then
        (* (* óSlo hay una palabra y anterior <> ESPACIO *)
           write('Todas las palabras terminan en la misma letra')
        else begin
        (* EA (S) es un espacio en blanco
           Esquema de recorrido y busqueda.
           Comprobar en el resto de la secuencia *)
        ultLetraPrimPal := anterior;
        anterior := EA_MSC1(S);
        Avanzar_MSC1(S);
        while (EA_MSC1(S) <> MSC1_MarcaFin) and
            not ((EA_MSC1(S) = ESPACIO) and (anterior <> ESPACIO) and
                (anterior <> ultLetraPrimPal)) do begin
            anterior := EA_MSC1(S);
            Avanzar_MSC1(S)
            (* INV anterior = "ultimo elemento de Piz" y
               "no hay ningun par (letra, espacio) en Piz
               tal que letra <> ultLetraPrimPal" *)
        end;
        if (EA_MSC1(S) = MSC1_MarcaFin) and ((anterior = ultLetraPrimPal)
            or (anterior = ESPACIO)) then
            write('Todas las palabras terminan en la misma letra')
        else
            write('Todas las palabras no terminan en la misma letra')
        end;
        writeln;
        writeln;
        write('Pulse enter para continuar');
        readln
    end.

```

```

program Algoritmo5_10;
(*
  * Algoritmo 5.10. Comprobar si todas las palabras acaban con la misma letra
  *                   (primer modelo, con secuencia intermedia formada por las
  *                   ultimas letras de cada palabra)
  * Titulo del libro: Una introduccion a la programacion.
  *                   Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 5. Tecnicas de disenyo de algoritmos
  * Fichero de entrada: dat5_10.txt
  *)

uses crt, unitmsc1;

const
  ESPACIO = ' ';
var
  ultLetra : char;
  FinDeSecuenciaUltLetra : boolean;
  ultLetraPrimPal : char;
  S : msc1;

procedure ObtenerUltimo;
  (* POST ultLetra es la última letra de la siguiente palabra,
     si la hay, y se actualiza FinDeSecuenciaUltLetra *)
  procedure IgnorarEspacios;
    (* POST EA es la siguiente letra, si la hay *)
  begin
    while (EA_MSC1(S) <> MSC1_MarcaFin) and (EA_MSC1(S) = ESPACIO) do
      Avanzar_MSC1(S);
    end;
  begin
    IgnorarEspacios;
    if (EA_MSC1(S) = MSC1_MarcaFin) then FinDeSecuenciaUltLetra := true
    else begin
      FinDeSecuenciaUltLetra := False;
      REPEAT
        ultLetra := EA_MSC1(S);
        Avanzar_MSC1(S)
      UNTIL (EA_MSC1(S) = MSC1_MarcaFin) or (EA_MSC1(S) = ESPACIO);
    end;
  end;
end;

```

```

procedure ComenzarUltLetra;
    (* POST ultLetra es la ultima letra de la primera palabra,
       si la hay, y se actualiza FinDeSecuenciaUltLetra *)
begin
    Comenzar_MSC1(S);
    ObtenerUltimo
end;
procedure AvanzarUltLetra;
    (* POST ultLetra es la última letra de la siguiente palabra,
       si la hay, y se actualiza FinDeSecuenciaUltLetra *)
begin
    ObtenerUltimo
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 5.10. Comprobar si todas las palabras acaban con *****');
    writeln('***** la misma letra (primer modelo, con secuencia *****');
    writeln('***** intermedia formada por las ultimas letras *****');
    writeln('***** de cada palabra *****');
    writeln('*****');
    writeln;
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'dat5_10.txt');
    write('La secuencia de entrada es: ');
    Comenzar_MSC1 (S);
    while (EA_MSC1 (S) <> MSC1_MarcaFin) do begin
        write(EA_MSC1(S));
        Avanzar_MSC1 (S)
    end;
    writeln;

    ComenzarUltLetra;
    if FinDeSecuenciaUltLetra then write('No hay palabras')
    else begin
        ultLetraPrimPal := ultLetra;
        repeat
            AvanzarUltLetra
        (* INV Todas las letras de Piz son iguales a ultLetraPrimPal *)
        until FinDeSecuenciaUltLetra or (ultLetra <> ultLetraPrimPal);
        writeln;
        if FinDeSecuenciaUltLetra then
            write('Todas las palabras terminan en la misma letra')
        else
            write('Todas las palabras no terminan en la misma letra')
    end
end

```

```

    end;
    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo5_11;
(*
 * Algoritmo 5.11. Obtener el k-abecedario de mayor longitud
 * Titulo del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *          Francisco J. Montoya Dato
 *
 *          Jose L. Fernandez Aleman
 *
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 5. Tecnicas de disenyo de algoritmos
 * Fichero de entrada: dat5_11.txt
 *)

(* El codigo ASCII de la enye no se encuentra entre el codigo ASCII *)
(* de la letra 'n' y la letra 'o' *)

uses crt, unitmsc1;

type
    TKAbecedario = record
        long : integer;
        primeraLetra : char
    end;

var
    kABC : TKAbecedario;
    EsUltimokABC, EsVaciakABC : boolean;
    anterior : char;
    mayorkABC : TKAbecedario;
    i : integer;
    S : msc1;

procedure EncontrarInikABC;
    (* PRE EA es el primer caracter de la secuencia o
       el siguiente despues del ultimo
       k-abecedario en Piz, si lo hay *)
    (* POST EA es la segunda letra del siguiente k-abecedario,
       si lo hay *)
begin

```

```

repeat
    anterior := EA_MSC1(S);
    Avanzar_MSC1(S);
    (* INV No hay ningun k-abecedario entre EA y
        el ultimo k-abecedario en Piz, si lo hay *)
until (EA_MSC1(S) = MSC1_MarcaFin) or (EA_MSC1(S) = Succ(anterior));
EsUltimokABC := EA_MSC1(S) = MSC1_MarcaFin
end;

procedure IniciarkABC;
    (* POST Se actualizan EsVaciakABC y EsUltimokABC,
        y anterior y EA contienen el primer y el segundo caracter,
        respectivamente del primer k-abecedario, si lo hay *)
begin
    Comenzar_MSC1(S);
    if EA_MSC1(S) = MSC1_MarcaFin then begin
        EsUltimokABC := true;
        EsVaciakABC := true;
    end
    else begin
        EncontrarInikABC;
        EsVaciakABC := EA_MSC1(S) = MSC1_MarcaFin;
    end;
end;

procedure AvanzarkABC;
    (* PRE EsUltimokABC = Falso y EA es el segundo caracter
        del actual k-abecedario *)
    (* POST kABC contiene la longitud y el primer caracter del actual
        k-abecedario, EA es el segundo caracter del siguiente
        k-abecedario, si lo hay, y se actualiza EsUltimokABC *)
begin
    kABC.long := 1;
    kABC.primeraLetra := anterior;
    REPEAT
        kABC.long := kABC.long + 1;
        anterior := EA_MSC1(S);
        Avanzar_MSC1(S);
        (* INV kABC.long mantiene la longitud del k-abecedario actual *)
    until (EA_MSC1(S) = MSC1_MarcaFin) or (EA_MSC1(S) <> Succ(anterior));
    if (EA_MSC1(S) = MSC1_MarcaFin) then EsUltimokABC := true
    else EncontrarInikABC
end;

begin
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'dat5_11.txt');

```

```

clrscr;
writeln;
writeln ('*****');
writeln ('***** Algoritmo 5.11 Obtener el k-abecedario de mayor longitud');
writeln ('*****');
writeln;
write('La secuencia de entrada es: ');
Comenzar_MSC1 (S);
while (EA_MSC1 (S) <> MSC1_MarcaFin) do begin
    write(EA_MSC1(S));
    Avanzar_MSC1 (S)
end;
writeln;
IniciarkABC;
if EsVaciakABC then write('No hay k-abecedarios')
else begin
    AvanzarkABC;
    mayorkABC := kABC;
    while not EsUltimokABC do begin
        AvanzarkABC;
        if mayorkABC.long < kABC.long then mayorkABC := kABC
        (* INV mayorkABC.long mantiene la longitud del maximo
           k-abecedario de Piz *)
    end;
    writeln;
    writeln('El k-abecedario de mayor longitud es: ');
    writeln;
    writeln(' longitud: ', mayorkABC.long);
    writeln;
    write(' k-abecedario: ');
    for i := 1 to mayorkABC.long do
        write(chr(ord(mayorkABC.primeraletra)+i-1));
    end;
    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo5_12;
(*
* Algoritmo 5.12. Control de trafico en una autopista
* Titulo del libro: Una introduccion a la programacion.
*
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
* Francisco J. Montoya Dato
*
* Jose L. Fernandez Aleman

```

```
*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 5. Tecnicas de disenyo de algoritmos
* Fichero de entrada: dat5_12.txt
*)

uses crt, unitmse1;

var
  numSegTotal : integer;
               (* Numero de segundos transcurridos durante el control *)
  numVehEnt : integer;
               (* Numero de vehiculos que han entrado en la autopista *)
  numSegInterMasLargo : integer;
               (* Intervalo de tiempo mas largo sin entrar vehiculos *)
  numSegInterActual : integer;
               (* Contador del numero de segundos del intervalo actual *)
  S : mse1;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 5.12 Control de trafico en una autopista *****');
  writeln('*****');
  writeln;
  TratamientoInicial_MSE1(S);
  Cargar_fichero_MSE1(S, 'dat5_12.txt');
  Comenzar_MSE1(S);
  numSegTotal := 0;
  numVehEnt := 0;
  numSegInterMasLargo := 0;
  write('La secuencia de entrada es: ');
  while EA_MSE1(S) <> 0 do begin
    write(EA_MSE1(S), ' ');
    case EA_MSE1(S) of
      2 : begin
          numVehEnt := numVehEnt + 1;
          Avanzar_MSE1(S)
        end;
      1 : begin
          numSegInterActual := 1;
          Avanzar_MSE1(S);
          while not ((EA_MSE1(S) = 0) or
                     (EA_MSE1(S) = 2)) do begin
            write(EA_MSE1(S), ' ');
            if EA_MSE1(S) = 1 then
```

```

        numSegInterActual := numSegInterActual + 1;
        Avanzar_MSE1(S);
    end;
    if numSegInterActual > numSegInterMasLargo then
        numSegInterMasLargo := numSegInterActual;
        numSegTotal := numSegTotal + numSegInterActual
    end;
    else Avanzar_MSE1(S);
end;
writeln;
writeln;
writeln('El tiempo total transcurrido es: ', numSegTotal);
writeln;
writeln('El numero de vehiculos que han circulado por la autopista es: ',
        numVehEnt);
writeln;
writeln('El intervalo de tiempo mas largo sin entrar un vehiculo es: ',
        numSegInterMasLargo);
writeln;
write('Pulse enter para continuar');
readln
end.

```

3.5. Capítulo 6

```

program Algoritmo6_1;
(*
 * Algoritmo 6.1. Mostrar la nota de un alumno (version 1)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 * Fichero de entrada: datos6_1.dat
 *)

(* Para generar el fichero datos6_1.dat emplear el programa genf6_1.pas *)
(* Para consultar el fichero datos6_1.dat emplear el programa verf6_1.pas *)
(* En este algoritmo se ha sustituido la secuencia de NotaAlumno
   por un fichero secuencial, 'datos6_1.dat' *)

uses crt;

const
    MAX_ALUMNOS = 200;

type
    NotaAlumno = record
        codigo : 1..MAX_ALUMNOS;
        nota : real;
    end;
var
    f : file of NotaAlumno;
    alum : NotaAlumno;
    cod : 1..MAX_ALUMNOS;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.1. Mostrar la nota de un alumno (version 1) *****');
    writeln('*****');
    writeln;
    assign(f, 'datos6_1.dat');
    reset(f);
    write('Introduzca el codigo del alumno (>= 1) (<= ', MAX_ALUMNOS, '): ');
    readln(cod);

```

```

if eof(f) then writeln('El fichero esta vacio')
else begin
    repeat
        read(f,alum);
    until (eof(f)) or (alum.codigo = cod);
    (* Si el codigo es el ultimo, eof sera TRUE,
       por ello debemos preguntar por alum.codigo
       para determinar su presencia *)
    writeln;
    if (alum.codigo <> cod) then writeln('No existe ese codigo')
    else writeln('La nota de este alumno es: ',alum.nota:1:2);
end;
writeln;
write('Pulse enter para continuar');
readln

end.

```

```

program Algoritmo6_2;
(*
 * Algoritmo 6.2. Mostrar la nota de un alumno (version 2)
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

(* En lugar de solicitar las notas, se ha declarado la tabla constante *)

uses crt;

const
    MAX_ALUMNOS = 100 ;

    notas : array [1..MAX_ALUMNOS] of real =
        (1.0, 3.5, 5.2, 1.2, 2.7, 5.3, 6.2, 2.2, 1.2, 8.2,
         3.5, 5.2, 1.2, 2.7, 3.7, 2.2, 1.2, 8.2, 5.4, 3.2,
         5.3, 6.7, 2.2, 1.2, 8.2, 3.5, 5.2, 1.2, 2.7, 3.4,
         2.2, 1.9, 8.2, 5.4, 3.2, 5.3, 6.2, 2.2, 1.2, 8.2,
         5.5, 5.2, 1.2, 2.7, 3.4, 2.2, 1.2, 8.2, 5.4, 3.2,
         5.3, 6.9, 2.2, 1.2, 8.2, 3.5, 5.2, 9.2, 2.7, 3.4,
         2.6, 1.2, 8.2, 5.4, 3.2, 5.3, 6.2, 2.2, 1.2, 8.2,
         3.4, 5.2, 9.2, 2.7, 3.4, 2.2, 1.9, 8.2, 5.4, 3.2,

```

```
          9.0, 6.2, 7.2, 1.2, 8.2, 3.9, 5.2, 1.2, 2.7, 3.4,
          2.4, 1.2, 7.2, 5.4, 3.9, 5.3, 6.2, 2.2, 1.2, 8.2);

var
    cod : 1..MAX_ALUMNOS;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.2. Mostrar la nota de un alumno (version 2) *****');
    writeln('*****');
    writeln;
    write('Introduzca el codigo del alumno (>= 1) y (<= ', MAX_ALUMNOS, '): ');
    readln(cod);
    writeln;
    if (notas[cod] <> -1.0) then
        writeln('La nota del alumno ',cod,' es: ', notas[cod]:1:2)
    else writeln('No existe una calificacion para ese alumno');
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo6_3;
(*
  * Algoritmo 6.3. Calculo de la frecuencia de aparicion de las letras mayusculas en un
  * texto
  * Titulo del libro: Una introduccion a la programacion.
  *                      Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                      Francisco J. Montoya Dato
  *                      Jose L. Fernandez Aleman
  *                      Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 6. Tablas
  * Fichero de entrada: datos6_3.txt
  *)

uses crt, unitmscl;

type
    FrecuenciaLetras = array ['A'..'Z'] of integer;
var
    tf : FrecuenciaLetras;
    s : msc1;
```

```

procedure InicializarTablaFrecuencias (var t : FrecuenciaLetras);
(* Inicializacion de la tabla con ceros *)
(* POST: t[c] = 0, c: 'A' <= c <= 'Z' *)

var
    c : 'A'..'Z';

begin
    for c := 'A' to 'Z' do
        t[c] := 0;
    end;

procedure EscribirTablaFrecuencias (t : FrecuenciaLetras);
(* POST: Se escriben los valores de la tabla t *)
var
    c : 'A'..'Z';

begin
    writeln;
    for c := 'A' to 'Z' do begin
        write(' /* ', c, ' : ', t[c]);
        if ((ord(c)-ord('A')+1) mod 6 = 0) then writeln;
    end;
    writeln
end;

function EsMayuscula (car : char) : Boolean;
(* POST: Retorna Verdadero si y solo si car es una letra mayuscula *)

begin
    EsMayuscula := (car >= 'A') and (car <= 'Z');
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.3. Calculo de la frecuencia de aparicion de las *****');
    writeln('*****          letras mayusculas en un texto          *****');
    writeln('*****');
    writeln;
    TratamientoInicial_MSC1(s);
    Cargar_fichero_MSC1(s, 'datos6_3.txt');
    Comenzar_MSC1(s);
    if EA_MSC1(s) = MSC1_MarcaFin then
        writeln('Secuencia vacia')
    else begin

```

```

        InicializarTablaFrecuencias(tf);
        write('La secuencia de entrada es: ');
        repeat
            if EA_MSC1(s) = MSC1_MarcaFinLinea then writeln
            else write(EA_MSC1(s));
            if EsMayuscula(EA_MSC1(s)) then
                tf[EA_MSC1(s)] := tf[EA_MSC1(s)] + 1;
            Avanzar_MSC1(s);
            (* INV: tf almacena la frecuencia de aparicion de cada
                letra mayuscula en Piz *)
        until EA_MSC1(s) = MSC1_MarcaFin;
        writeln;
        EscribirTablaFrecuencias(tf);

    end;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo6_4;
(*
 * Algoritmo 6.4. Calcular distribucion de las notas
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 * Jose L. Fernandez Aleman
 * Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 * Fichero de entrada: datos6_4.dat
 *)

(* Para generar el fichero datos6_4.dat emplear el programa genf6_4.pas *)
(* Para consultar el fichero datos6_4.dat emplear el programa verf6_4.pas *)

uses crt;

const
    NUM_NOTAS = 5;
    MAX_ESTUDIANTES = 100;
type
    Estudiante = record
        nombre : string;
        edad : integer;
        sexo : boolean;
        nota : real;
    end;

```

```

    end;
    Festudiantes = file of Estudiante;
    Curso = array [1..MAX_ESTUDIANTES] of Estudiante;
var
    f : Festudiantes;
    c : Curso;
    fNotas : array [1..NUM_NOTAS] of integer;
    i, k : 1..MAX_ESTUDIANTES;
    j : 1..NUM_NOTAS;

procedure Escribe(i : integer);
begin
    case i of
        1 : write('Numero de suspensos: ');
        2 : write('Numero de aprobados: ');
        3 : write('Numero de notables: ');
        4 : write('Numero de sobresalientes: ');
        5 : write('Numero de matriculas de honor: ');
    end;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.4. Calcular distribucion de las notas *****');
    writeln('*****');
    writeln;
    assign(f, 'datos6_4.dat');
    reset(f);
    k := 1;
    while (not eof(f)) and (k <= MAX_ESTUDIANTES) do begin
        read(f, c[k]);
        k := k + 1;
    end;
    close(f);
    (* Comenzar *)
    i := 1;

    (* Tratamiento inicial *)
    for j := 1 to NUM_NOTAS do
        fNotas[j] := 0;

    (* Tratamiento de cada elemento de la tabla *)
    while i <> k do begin
        if (c[i].nota) < 5 then fNotas[1] := fNotas[1] + 1
        else if (c[i].nota >= 5) and (c[i].nota < 7) then fNotas[2] := fNotas[2] + 1

```

```

        else if (c[i].nota >= 7) and (c[i].nota < 9) then fNotas[3] := fNotas[3] + 1
        else if c[i].nota = 9 then fNotas[4] := fNotas[4] + 1
        else fNotas[5] := fNotas[5] + 1;
        (* Avanzar *)
        i := i+1;
    end;

    (* Tratamiento final *)
    for j := 1 to NUM_NOTAS do begin
        Escribe(j);
        writeln(fNotas[j]);
    end;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo6_5;
(*
 * Algoritmo 6.5. Porcentaje de mujeres y hombres, y nota media de cada sexo
 * Título del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *          Francisco J. Montoya Dato
 *
 *          Jose L. Fernandez Aleman
 *
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 * Fichero de entrada: datos6_4.dat
 *)

(* Para generar el fichero datos6_4.dat emplear el programa genf6_4.pas *)
(* Para consultar el fichero datos6_4.dat emplear el programa verf6_4.pas *)

uses crt;
const
    MAX_ESTUDIANTES = 100;
type
    Estudiante = record
        nombre: string;
        edad : integer;
        sexo : Boolean;          (* verdadero si es mujer, falso si es hombre *)
        nota : Real;
    end;
    Festudiantes = file of Estudiante;
    NumEstudiantes = 0..MAX_ESTUDIANTES;

```

```

Curso = array [1.. MAX_ESTUDIANTES] of Estudiante;
var
  c : Curso;
  contaM, contaH : 0..MAX_ESTUDIANTES;      (* contadores numero de mujeres y hombres *)
  sumaM, sumaH : Real;                      (* contadores notas de mujeres y hombres *)
  mujeres, hombres : Real;                  (* porcentajes de mujeres y hombres *)
  long : NumEstudiantes;                    (* longitud de la secuencia almacenada *)
  i : 1..MAX_ESTUDIANTES;                   (* indice para la tabla *)

procedure Leer(var c : Curso; var l : NumEstudiantes);
var
  f : FEstudiantes;
begin
  assign(f, 'datos6_4.dat');
  reset(f);
  l := 0;
  while (not eof(f)) and (l < MAX_ESTUDIANTES) do begin
    l := l + 1;
    read(f, c[l]);
  end;
  close(f);
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 6.5. Porcentaje de mujeres y hombres, *****');
  writeln('***** y nota media de cada sexo *****');
  writeln('*****');
  writeln;
  (* Se introducen los datos de long estudiantes en la tabla c *)
  Leer (c, long);
  case long of
    0 : writeln('No hay alumnos');
    else begin
      sumaM := 0; sumaH := 0;
      contaM := 0; contaH := 0;
      for i := 1 to long do
        if c[i].sexo then begin
          sumaM := sumaM + c[i].nota;
          contaM := contaM + 1;
        end
        else begin
          sumaH := sumaH + c[i].nota;
          contaH := contaH + 1;
        end;
      end;
    end;
  end;
end;

```

```
        mujeres := contaM / long * 100;
        hombres := contaH / long * 100;
        writeln('Porcentaje de mujeres: ', mujeres:1:2, '%');
        writeln;
        writeln('Porcentaje de hombres: ', hombres:1:2, '%');
        if contaM <> 0 then begin
            writeln;
            writeln('Nota media de las mujeres: ', sumaM/contaM:1:2);
        end;
        if contaH <> 0 then begin
            writeln;
            writeln('Nota media de los hombres: ', sumaH/contaH:1:2);
        end;
    end;
end;
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo6_6;
(*
 * Algoritmo 6.6. Producto escalar de dos vectores
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

uses crt;

const
    LMAX = 100;
type
    Rango = 1..LMAX;
    Vector = array [Rango] of Real;
var
    a, b : Vector;
    producto : Real;
    long : Rango;
    i : Rango;

procedure Leer(VAR a:Vector; n:Rango);
```

```

var
  i:Rango;
begin
  for i:=1 to n do begin
    write('Introduzca el componente ',i,': ');
    readln(a[i]);
  end
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 6.6. Producto escalar de dos vectores *****');
  writeln('*****');
  writeln;
  write('Introduzca la longitud de los vectores: ');
  readln(long);
  writeln;
  writeln('Vector 1: ');
  Leer(a, long);
  writeln;
  writeln('Vector 2: ');
  Leer(b, long);

  producto := 0;
  for i := 1 to long do
    producto := producto + a[i] * b[i];
    (* INV producto = a[1]* b[1] + a[2]* b[2] + .. a[i] * b[i] *)

  writeln;
  writeln('El producto escalar es: ', producto:1:2);
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo6_7;
(*
  * Algoritmo 6.7. Obtener el mayor, su numero de ocurrencias y su primera y ultima
    posicion
  * Titulo del libro: Una introduccion a la programacion.
  *               Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *               Francisco J. Montoya Dato
  *               Jose L. Fernandez Aleman
  *               Maria J. Majado Rosales
*)

```

```

* Fecha: 1/9/2005
* Capitulo 6. Tablas
*)

uses crt;

const
    LMAX = 100;
type
    Rango = 1..LMAX;
    Vector = array [Rango] of integer;
var
    long : Rango;
    t : Vector;
    mayor : integer;
    numApa : Rango;
    pri, ult : Rango;
    i : Rango;

procedure Leer(VAR a:Vector; n:Rango);
var
    i:Rango;
begin
    for i:=1 to n do begin
        write('Introduzca el componente ',i,': ');
        readln(a[i]);
    end;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.7. Obtener el mayor, su numero de ocurrencias y *****');
    writeln('***** su primera y ultima posicion *****');
    writeln('*****');
    writeln;
    write('Introduzca el numero de elementos (> 0): ');
    readln(long);
    Leer(t, long);
    mayor := t[1]; numApa := 1; pri := 1; ult := 1;
    for i := 2 to long do begin
        (* Se cumple INV definido en la descripcion del algoritmo *)
        if t[i] > mayor then begin
            mayor := t[i];
            numApa := 1;
            pri := i;

```

```

        ult := i;

    end
    else if t[i] = mayor then begin
        numApa := numApa + 1;
        ult := i;
    end
end;
writeln;
writeln('El mayor es: ', mayor);
writeln('El numero de apariciones es: ', numApa);
writeln('La primera aparicion y la ultima son: ', pri, ' y ', ult);
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo6_8;
(*
  * Algoritmo 6.8. Calcular la fecha a partir de la posicion del dia en el anyo (version
  * 1)
  * Titulo del libro: Una introduccion a la programacion.
  *                   Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 6. Tablas
  *)

uses crt;

type
  MesAnyo = 1..12;
  DiaAnyo = 1..365;
  Diames = 1..31;
const
  tp : array [MesAnyo] of integer = ( 1, 32, 60, 91, 121, 152, 182, 213, 244, 274,
    305, 335 );
var
  p : DiaAnyo;
  i : MesAnyo;
  mes : MesAnyo;
  dia : Diames;

begin
  clrscr;

```

```
writeln;
writeln('*****');
writeln('***** Algoritmo 6.8. Calcular la fecha a partir de la posicion *****');
writeln('***** del dia en el anyo (version 1) *****');
writeln('*****');
writeln;
write('Introduzca la posicion del dia en el anyo: ');
readln (p);
i := 2;
while (i <> 12) and (tp [i] <= p) do
    (* INV (j: 1 <= j < i : p >= tp[j]) *)
    i := i + 1;

if tp[i] > p then mes := i - 1
               else mes := 12;
dia := p - tp[mes] + 1;
writeln;
writeln('Dia ',dia,' del mes ', mes);
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo6_9;
(*
 * Algoritmo 6.9. Calcular la fecha a partir de la posicion del dia en el anyo (version
   2)
 * Titulo del libro: Una introduccion a la programacion.
 *                   Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *                   Francisco J. Montoya Dato
 *                   Jose L. Fernandez Aleman
 *                   Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

uses crt;

type
    MesAyo = 1..12;
    DiaAyo = 1..365;
    Diames = 1..31;

const
    tp : array [MesAyo] of integer = (1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305,
    335);
```

```

var
  p : DiaAyo;
  i : MesAyo;
  mes : MesAyo;
  dia : Diames;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 6.9. Calcular la fecha a partir de la posicion *****');
  writeln('*****          del dia en el año (version 2)          *****');
  writeln('*****');
  writeln;
  write('Introduzca la posicion del dia en el año: ');
  readln(p);
  if p >= tp[12] then mes := 12
  else begin
    i := 2;
    while tp[i] <= p do
      (* INV: j: 1 <= j < i : p >= tp[j] *)
      i := i + 1;
    mes := i - 1;
  end;
  dia := p - tp[mes] + 1;
  writeln;
  writeln('Dia ',dia,' del mes ', mes);
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo6_10;
(*
  * Algoritmo 6.10. Calculo del peso del segmento de mayor peso
  * Titulo del libro: Una introduccion a la programacion.
  *          Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *          Francisco J. Montoya Dato
  *          Jose L. Fernandez Aleman
  *          Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 6. Tablas
  *)
uses crt;

```

```
const
    LMAX = 100;
type
    Rango = 1..LMAX;
    Vector = array [Rango] of integer;
var
    t : Vector;
    pesoMaximo : integer; (* peso mayor de los segmentos de la secuencia *)
    pesoMaximoSegAct : integer; (* peso mayor de los segmentos actuales *)
    i, long : Rango;

procedure Leer(VAR a : Vector; n : Rango);
var
    i : Rango;
begin
    for i := 1 to n do begin
        write(' Introduzca componente ', i, ': ');
        readln(a[i]);
    end;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.10. Calculo del peso del segmento de mayor peso *****');
    writeln('*****');
    writeln;
    write('Introduce el numero de elementos: ');
    readln(long);
    Leer(t, long);
    if long = 0 then writeln('Secuencia vacia')
    else begin
        pesoMaximo := t[1];
        pesoMaximoSegAct := t[1];
        for i:= 2 to long do begin
            if t[i] > t[i] + pesoMaximoSegAct
            then pesoMaximoSegAct := t[i]
            else pesoMaximoSegAct := pesoMaximoSegAct + t[i];
            if pesoMaximo < pesoMaximoSegAct
            then pesoMaximo := pesoMaximoSegAct;
        end;
        writeln;
        writeln('El peso del segmento de mayor peso es: ', pesoMaximo);
    end;
    writeln;
```

```

    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo6_11;
(*
  * Algoritmo 6.11. Suma de enteros distintos de cero en segmentos con cero en los
    extremos
  * Titulo del libro: Una introduccion a la programacion.
  *                   Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 6. Tablas
  *)

uses crt;

const
  LMAX = 100;
type
  Rango = 1..LMAX;
  Vector = array [Rango] of integer;
var
  t : Vector;
  long : Rango;
  i : Rango;
  noCerosSLC : integer;
  pesoNoCeros : integer;
  ceros : integer;

procedure Leer(VAR a:Vector; n:Rango);
var
  i:Rango;
begin
  for i:=1 to n do begin
    write(' Introduzca el elemento ',i,': ');
    readln(a[i]);
  end;
end;

begin
  clrscr;
  writeln;
  writeln('*****');

```

```
writeln('***** Algoritmo 6.11. Suma de enteros distintos de cero en *****');
writeln('***** segmentos con cero en los extremos *****');
writeln('*****');
writeln;
write('Introduzca la longitud de la secuencia: ');
readln(long);
Leer(t, long);
noCerosSLC := 0;
pesoNoCeros := 0;
ceros := 0;
for i := 1 to long do
    if t[i] = 0 then begin
        noCerosSLC := noCerosSLC + pesoNoCeros;
        ceros := ceros + 1;
    end
    else pesoNoCeros := pesoNoCeros + ceros;
writeln;
writeln('La suma de los elementos distintos de cero en');
writeln('segmentos con cero en los extremos es: ',noCerosSLC);
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo6_12;
(*
 * Algoritmo 6.12. Calcular numero de pares cero-uno (version fuerza bruta)
 * Titulo del libro: Una introduccion a la programacion.
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 * Francisco J. Montoya Dato
 * Jose L. Fernandez Aleman
 * Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

uses crt;

const
    LMAX = 100;
type
    Rango = 1..LMAX;
    Vector = array [Rango] of 0..1;
var
    t : Vector;
    numPares : integer;
```

```

i, j, long : Rango;

procedure Leer(VAR a : Vector; n : Rango);
var
  i : Rango;
begin
  for i := 1 to n do begin
    write(' Introduzca elemento ',i,': ');
    readln(a[i]);
  end;
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 6.12. Calcular numero de pares cero-uno *****');
  writeln('***** (version fuerza bruta) *****');
  writeln('*****');
  writeln;
  write('Introduzca la longitud de la secuencia: ');
  readln(long);
  Leer(t, long);
  numPares := 0;
  for i := 1 to long do
    if t[i] = 0 then
      for j := i + 1 to long do
        if t[j] = 1 then numPares := numPares + 1;
      end;
    writeln;
  writeln('El numero de pares 0-1 es: ',numPares);
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo6_13;
(*
  * Algoritmo 6.13. Metodo de ordenacion de insercion directa
  * Titulo del libro: Una introduccion a la programacion.
  *
  * Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
  * Francisco J. Montoya Dato
  *
  * Jose L. Fernandez Aleman
  *
  * Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 6. Tablas
  *)

```

```
(* Metodo de ordenacion de insercion directa *)

uses crt;

const
    LMAX = 100;

type
    Rango = 0..LMAX;
    Vector = array [Rango] of integer;

var
    t: Vector;
    q, j : Rango;
    n : Rango;
    x : integer;

procedure Leer(VAR a : Vector; n : Rango);
var
    i : Rango;
begin
    for i := 1 to n do begin
        write(' Introduzca elemento ',i,': ');
        readln(a[i]);
    end
end;

procedure Escribir(a : Vector; n : Rango);
var
    i : Rango;
begin
    for i := 1 to n do
        write(a[i], ' ');
    writeln
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.13. Metodo de ordenacion de insercion directa *****');
    writeln('*****');
    writeln;
    write('Introduzca el numero de elementos: ');
    readln(n);
    Leer(t, n);
```

```

    writeln;
    write('Los elementos iniciales son: ');
    Escribir(t,n);

    for q := 2 to n do begin
        t[0] := t[q];          (* centinela *)
        x := t[q];
        j := q - 1;
        while x < t[j] do begin
            t[j+1] := t[j];    (* desplazamiento *)
            j := j - 1;
        end;
        t[j+1] := x;
    end;

    writeln;
    write('Los elementos ordenados son: ');
    Escribir(t,n);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo6_14;
(*
 * Algoritmo 6.14. Metodo de ordenacion de seleccion directa
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

(* Metodo de ordenacion de seleccion directa *)

uses crt;

const
    LMAX = 100;
type
    Rango = 1..LMAX;
    Vector = array [Rango] of integer;
var

```

```

t : Vector;
pos, q, j : Rango;
n : Rango;
min : integer;

procedure Leer(VAR a:Vector; n:Rango);
var
    i:Rango;
begin
    for i:=1 to n do begin
        write(' Introduzca elemento ',i,' : ');
        readln(a[i]);
    end;
end;

procedure Escribir(a:Vector; n:Rango);
var
    i:Rango;
begin
    for i:=1 to n do
        write(a[i], ' ');
    writeln;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.14. Metodo de ordenacion de seleccion directa *****');
    writeln('*****');
    writeln;

    write('Introduzca el numero de elementos: ');
    readln(n);
    Leer(t, n);

    writeln;
    write('Los elementos iniciales son: ');
    Escribir(t,n);

    for q := 1 to n - 1 do begin
        pos := q;
        min := t[q];
        for j := q+1 to n do
            if min > t[j] then begin
                pos := j;
                min := t[j];
            end;
        end;
    end;

```

```

        end;
        t[pos] := t[q];
        t[q] := min;
    end;

    writeln;
    write('Los elementos ordenados son: ');
    Escribir(t,n);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo6_15;
(*
 * Algoritmo 6.15. Búsqueda binaria en un tabla ordenada
 * Título del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

(* Búsqueda binaria en un tabla ordenada *)

uses crt;

const
    LMAX = 100;
type
    Rango = 1..LMAX;
    Vector = array [Rango] of integer;
var
    n : Rango;
    t : Vector;
    i : 0..LMAX;
    j : 1..LMAX+1;
    h : Rango;
    x: integer;

procedure Leer(VAR a : Vector; n : Rango);
var
    i:Rango;

```

```
begin
  for i := 1 to n do begin
    write(' Introduzca elemento ',i,': ');
    readln(a[i]);
  end;
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 6.15. Busqueda binaria en un tabla ordenada *****');
  writeln('*****');
  writeln;
  write('Introduzca el numero de elementos (maximo ', LMAX, '): ');
  readln(n);
  Leer(t, n);
  writeln;
  write('Introduzca el elemento a buscar: ');
  readln(x);
  i := 0;
  j := n + 1;
  while (i + 1) <> j do begin
    h := (i + j) div 2;
    if t[h] <= x then i := h
    else j := h;
  end;
  writeln;
  write('La secuencia de entrada es: ');
  for h := 1 to n do begin
    write (t[h], ' ');
  end;
  writeln;
  writeln;
  if (i > 0) and (x = t[i]) then
    writeln('Valor encontrado en la posicion: ', i )
  else
    writeln('Valor no encontrado');
  writeln;
  write('Pulse enter para continuar');
  readln
end.
```

```
program Algoritmo6_16;
(*
  * Algoritmo 6.16. Suma de los valores mayores de cada fila de una matriz de enteros
  * Titulo del libro: Una introduccion a la programacion.
```

```

*                               Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*                               Francisco J. Montoya Dato
*                               Jose L. Fernandez Aleman
*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 6. Tablas
*)

uses crt;

const
    n = 20;
    NFilas = n; (* constante con el numero de filas *)
    NColumnas = n; (* constante con el numero de columnas *)

type
    RangoFilas = 1..NFilas;
    RangoColumnas = 1..NColumnas;
    Vector = array [RangoFilas, RangoColumnas] of integer;
var
    t : Vector;
    nf, i : RangoFilas;
    nc, j : RangoColumnas;
    sm, mayor: integer;

procedure Leer(VAR a : Vector; nf : RangoFilas; nc : RangoColumnas);
var
    i : RangoFilas;
    j : RangoColumnas;

begin
    for i:=1 to nf do begin
        writeln;
        writeln('Fila: ',i);
        for j:=1 to nc do begin
            write('Introduzca elemento (' ,i, ', ', j, '): ');
            readln(a[i, j]);
        end;
    end;

end;

begin
    clrscr;
    writeln;
    writeln('*****');

```

```

writeln('***** Algoritmo 6.16. Suma de los valores mayores de cada fila *****');
writeln('*****          de una matriz de enteros          *****');
writeln('*****');
writeln;
write('Introduzca el numero de filas (<= ',n,'): ');
readln(nf);
writeln;
write('Introduzca el numero de columnas (<= ',n,'): ');
readln(nc);
Leer(t, nf, nc);
sm := 0;
for i := 1 to nf do begin
    mayor := t[i,1];
    for j := 2 to nc do begin
        if mayor < t[i,j] then mayor := t[i,j]
        (* INVint mayor = mayor de los j elementos ya recorridos
           de la fila i, 1 <= j <= nc *)
    end;
    sm := sm + mayor
    (* INVext sm = suma de los mayores de las primeras
       i filas de T, 1 <= i <= nf *)
end;
(* INVext y (i = nf) => POST *)
writeln;
writeln('La suma de los valores mayores de cada fila de');
writeln('la matriz de enteros es: ', sm);
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo6_17;
(*
 * Algoritmo 6.17. Cifrado PlayFair
 * Titulo del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *          Francisco J. Montoya Dato
 *
 *          Jose L. Fernandez Aleman
 *
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 * Fichero de entrada: dat6_17.txt
 *)
uses crt, unitmscl;
const

```

```

    N = 5;
    ESPACIO = ' ';
type
    Palabra = string;
    MatrizCar = array [1..N,1..N] of char;
    ParLetras = record
        p, s : Char;
    end;
    Posicion = record
        f, c : integer;
    end;
var
    clave : Palabra;                (* palabra clave *)
    parCifrado : ParLetras;         (* par codificado *)
    mc : MatrizCar;                (* matriz de codificacion *)
    S : msc1;                      (* texto original *)
    R : msc1;                      (* texto cifrado *)
    (* lexico de una secuencia intermedia de pares de caracteres *)
    parActual : ParLetras;
    ultimoPar : boolean;

function FinPares : Boolean;
begin
    FinPares := (EA_MSC1(S) = MSC1_MarcaFin) and not ultimoPar
end;

procedure IgnorarBlancos;
(* Salta blancos del texto original *)
begin
    while EA_MSC1(S) = ESPACIO do
        Avanzar_MSC1(S);
    end;

procedure sustituir(var l : char);
begin
    case l of
        'j': l:='i';
        'i': l:='n';
    end;
end;

procedure SigPar;
begin
    IgnorarBlancos;
    if EA_MSC1(S) <> MSC1_MarcaFin then begin
        parActual.p := EA_MSC1(S);
        sustituir(parActual.p);
    end;
end;

```

```

        Avanzar_MSC1(S);
        IgnorarBlancos;
        if EA_MSC1(S) = MSC1_MarcaFin then begin
                                                    ultimoPar := true;
                                                    parActual.s:='x';

        end
        else begin
            parActual.s := EA_MSC1(S);
            sustituir(parActual.s);
            if parActual.s=parActual.p then
                parActual.s:='x'
            end;
        end;
    end;
end;

procedure ComenzarPar;

begin
    Comenzar_MSC1(S);
    ultimoPar := false;
    Sigpar;
end;

procedure AvanzarPar;
begin
    if ultimoPar then ultimoPar := false
    else begin
        Avanzar_MSC1(S);
        SigPar;
    end;
end;

function CarEnPalabra(p : Palabra; long : integer; c : Char) : Boolean;
(* PRE: long > 0 *)
(* POST retorna Verdadero si el caracter c se encuentra en la tabla p,
    Falso en otro caso *)
var
    i : 1..N+1;          (* En el rango [1, long] *)

begin
    i := 1;
    while (i <> long+1) and not (p[i] = c) do
        i := i + 1;
    CarEnPalabra := i <> long+1;
end;

procedure CrearMatrizCodificacion;

```

```

var
  letra : char;      (* letra actual *)
  i, j : 1..N;

begin
  (* rellenar la primera fila con la clave *)
  for i := 1 to N do
    mc[1,i] := clave[i];
  (* rellenar resto de filas *)
  (* inicializar recorrido intervalo de letras *)
  letra := 'a';
  for i := 2 to N do
    for j := 1 to N do begin
      while CarEnPalabra(clave, N, letra) or
        (letra = 'j') or (letra = 'i') do
        letra := Succ(letra);
      mc[i,j] := letra;
      letra := Succ(letra);
    end;
  end;

procedure PosEnTabla(m : MatrizCar; car : char; var pos : Posicion);
(* PRE: car es una letra y se encuentra en la tabla *)
var
  i, j : 1..N;
begin
  i := 1;
  j := 1;
  while (m[i,j] <> car) do
    if j < N then j := j + 1
    else begin
      i := i + 1;
      j := 1;
    end;
  pos.f := i;
  pos.c := j;
end;

procedure Codificacion(par : ParLetras; var nuevoPar : ParLetras);
var
  p1, p2 : Posicion;
  fila, col : 1..N;
begin
  PosEnTabla(mc, par.p, p1);
  PosEnTabla(mc, par.s, p2);
  if p1.f = p2.f then begin

```

```

        p1.c := p1.c mod N + 1;
        p2.c := p2.c mod N + 1;
        nuevoPar.p := mc[p1.f, p1.c];
        nuevoPar.s := mc[p2.f, p2.c]

    end
    else if p1.c = p2.c then begin
        p1.f := p1.f mod N + 1;
        p2.f := p2.f mod N + 1;
        nuevoPar.p := mc[p1.f, p1.c] ;
        nuevoPar.s := mc[p2.f, p2.c]

    end
    else begin
        nuevoPar.p := mc[p1.f, p2.c];
        nuevoPar.s := mc[p2.f, p1.c]

    end;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.17. Cifrado PlayFair *****');
    writeln('*****');
    writeln;

    write('Introduzca la palabra clave de 5 letras: ');
    readln(clave);
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'dat6_17.txt');
    writeln;
    write('El texto original es: ');
    Comenzar_MSC1(S);
    while EA_MSC1(S) <> MSC1_MarcaFin do begin
        write(EA_MSC1(S));
        Avanzar_MSC1(S);
    end;
    CrearMatrizCodificacion;
    ComenzarPar;
    TratamientoInicial_MSC1(R);
    Arrancar_MSC1(R);
    while not FinPares do begin
        Codificacion(parActual, parCifrado);
        Registrar_MSC1(R, parCifrado.p);
        Registrar_MSC1(R, parCifrado.s);
        AvanzarPar;
    end;
    Marcar_MSC1(R);

```

```

    writeln;
    writeln;
    write('El texto cifrado es: ');
    Comenzar_MSC1(R);
    while (EA_MSC1(R) <> MSC1_MarcaFin) do begin
        write(EA_MSC1(R));
        Avanzar_MSC1(R) ;
    end;
    writeln;
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo6_18;
(*
  * Algoritmo 6.18. Justificacion de un texto
  * Titulo del libro: Una introduccion a la programacion.
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *           Francisco J. Montoya Dato
  *           Jose L. Fernandez Aleman
  *           Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 6. Tablas
  * Fichero de entrada: dat6_18.txt
  *)

uses crt, unitmsc1;

const
    Espacio = ' ';
    SangradoMaximo = 15;
    LongMaxPal = 20;
    AnchoMinLinea = LongMaxPal + SangradoMaximo;
    AnchoMaxLinea = 80;

type
    LongitudPalabra = 1..LongMaxPal;
    Palabra = record
        contenido : array [LongitudPalabra] of char;
        longitud : LongitudPalabra
    end;
    PosEnLinea = 1..AnchoMaxLinea;
    LongitudLinea = 0..AnchoMaxLinea;
    Linea = record
        longitud : LongitudLinea;

```

```

        contenido : array [PosEnLinea] of char;
    end;
    LongSangrado = 0..SangradoMaximo;
    LimAnchoLinea = AnchoMinLinea..AnchoMaxLinea;
var
    sangrado : LongSangrado;
    anchoLinea : LimAnchoLinea;
    palabraActual : Palabra;
    lineaActual : Linea;
    FinDeSecuenciaPalabra : boolean;
    S : MSC1;
    MarcaParrafo : Palabra;

(* *****
 * Operaciones secuencia intermedia de palabras
 * ***** *)

procedure ObtenerPalabra;
var
    i : integer;
begin
    (* Ignora los espacios en blanco *)
    while (EA_MSC1(S) <> MSC1_MarcaFin) and
        ((EA_MSC1(S) = Espacio) or (EA_MSC1(S) = MSC1_MarcaFinLinea)) do
        Avanzar_MSC1 (S);
    FinDeSecuenciaPalabra := EA_MSC1(S) = MSC1_MarcaFin;
    if not (EA_MSC1(S) = MSC1_MarcaFin) then begin
        (* Almacena en el campo palabraActual.contenido la siguiente palabra *)
        i := 0;
        repeat
            i := i + 1;
            palabraActual.contenido[i] := EA_MSC1 (S);
            Avanzar_MSC1 (S);
        until (EA_MSC1(S) = MSC1_MarcaFin) or ((EA_MSC1(S) = Espacio) or
            ((EA_MSC1(S) = MSC1_MarcaFinLinea)));
        palabraActual.longitud := i;
    end
end;

(* ***** *)

procedure ComenzarPalabra;
begin
    TratamientoInicial_MSC1(S);
    Cargar_fichero_MSC1(S, 'dat6_18.txt');
    Comenzar_MSC1(S);
    ObtenerPalabra;
end;

```

```

(* ***** *)
procedure AvanzarPalabra;
begin
    ObtenerPalabra;
end;

(* ***** *)
* Operaciones relacionadas con el tratamiento de palabras
(* ***** *)
function Longitud (pal: Palabra) : LongitudPalabra;
begin
    Longitud := pal.longitud;
end;

(* ***** *)
function Igual (p1, p2 : Palabra) : boolean;
var
    long, i : LongitudPalabra;
    igualLong : boolean;
begin
    igualLong := p1.longitud = p2.longitud;
    if igualLong then begin
        long := p1.longitud;
        i := 1;
        while (i <> long) and (p1.contenido[i] = p2.contenido[i]) do
            i := i + 1;
            igualLong := (p1.contenido[i] = p2.contenido[i]);
        end;
        Igual := igualLong
    end;
end;

(* ***** *)
* Grupo de operaciones de tratamiento de lineas
(* ***** *)
procedure AnyadirCaracterALinea (car : char);
begin
    lineaActual.longitud := lineaActual.longitud + 1;
    lineaActual.contenido[lineaActual.longitud] := car
end;

(* ***** *)
procedure AnyadirPalabraALinea;
var
    i : LongitudPalabra;
begin
    for i := 1 to palabraActual.longitud do

```

```

        AnyadirCaracterALinea (palabraActual.contenido[i])
end;

(* ***** *)
procedure InicializarPropLinea (var longSangrado: LongSangrado;
                                var longLinea : LimAnchoLinea);
var
    sangradoLinea, anchoLinea : integer;
begin
    write('Introduzca el ancho de la linea (entre ', AnchoMinLinea,
          ' y ', AnchoMaxLinea, '): ');
    readln(anchoLinea);
    if anchoLinea < AnchoMinLinea then longLinea := AnchoMinLinea
    else if anchoLinea > AnchoMaxLinea then longLinea := AnchoMaxLinea
    else longLinea := anchoLinea;
    writeln;
    write('Introduzca el sangrado de la primera linea (entre 0',
          ' y ', SangradoMaximo, '): ');
    readln(sangradoLinea);
    if sangradoLinea < 0 then longSangrado := 0
    else if sangradoLinea > SangradoMaximo then longSangrado := SangradoMaximo
    else longSangrado := sangradoLinea;

    MarcaParrafo.longitud := 5;
    MarcaParrafo.contenido[1] := '/';
    MarcaParrafo.contenido[2] := 'P';
    MarcaParrafo.contenido[3] := 'A';
    MarcaParrafo.contenido[4] := 'R';
    MarcaParrafo.contenido[5] := '/';

end;

(* ***** *)
procedure SangrarLinea (longSangrado : integer);
var
    i : 1..SangradoMaximo;
begin
    for i := 1 to longSangrado do
        AnyadirCaracterALinea(Espacio)
    end;

end;

(* ***** *)
procedure InicializarLinea (longSangrado : integer);
begin
    lineaActual.longitud := 0;
    if longSangrado > 0 then SangrarLinea(longSangrado)
end;

```

```

(* ***** *)
function CabeEnLinea (espacioSolicitado: LongitudPalabra;
                    longLinea : integer) : boolean;
begin
    CabeEnLinea := espacioSolicitado <= (longLinea - lineaActual.longitud)
end;

(* ***** *)
procedure EscribirLinea (justificacion : boolean;
                        longLinea, longSangrado : integer);
    (* ***** *)
    procedure EscribirLineaJustificada (longLinea, longSangrado : integer);
    var
        numHuecos, blancosExtraMin : LongitudLinea;
        numHuecosAct, numHuecosMin : LongitudLinea;
        limInf, limSup, pos : PosEnLinea;
        (* ***** *)
        procedure EscribirEspacios (numEspacios : PosEnLinea);
        var
            i : PosEnLinea;
        begin (* EscribirEspacios *)
            for i := 1 to numEspacios do
                write(Espacio);
            end; (* EscribirEspacios *)

        begin (* EscribirLineaJustificada
            * Calcular limite inferior y superior *)
            if lineaActual.contenido[1] = Espacio
                (* Si es primera linea de un parrafo *)
            then limInf := longSangrado + 1
            else limInf := 1;
            if lineaActual.contenido[lineaActual.longitud] = Espacio
                then limSup := lineaActual.longitud - 1
            else limSup := lineaActual.longitud;
            EscribirEspacios(limInf - 1);

            (* Calcular número de huecos *)
            numHuecos := 0;
            for pos := limInf to limSup do
                if lineaActual.contenido[pos] = Espacio then
                    numHuecos := numHuecos + 1;

            (* Calcular numero de espacios a insertar y numero de huecos
            * en los que se inserta ese numero de espacios, en el resto
            * se inserta uno mas *)
            if (numHuecos <> 0) then begin

```

```

        blancosExtraMin := (longLinea - limSup) div numHuecos;
        numHuecosMin := numHuecos -
                        (longLinea - limSup) mod numHuecos
    end;

    (* Recorrido para insertar espacios en los huecos *)
    numHuecosAct := 0;
    for pos := limInf to limSup do
        if (lineaActual.contenido[pos] = Espacio) and
            (numHuecosAct < numHuecosMin) then begin
            EscribirEspacios (blancosExtraMin + 1);
            numHuecosAct := numHuecosAct + 1;
        end
        else if (lineaActual.contenido[pos] = Espacio) and
            (numHuecosAct >= numHuecosMin)
            then EscribirEspacios(blancosExtraMin+2)
        else if (lineaActual.contenido[pos] <> Espacio)
            then write(lineaActual.contenido[pos])
    end; (* EscribirLineaJustificada *)

    (* ***** *)
    procedure EscribirLineaSinJustificar;
    var
        i : PosEnLinea;
    begin (* EscribirLineaSinJustificar *)
        for i := 1 to lineaActual.longitud - 1 do
            write(lineaActual.contenido[i]);
            if lineaActual.contenido[lineaActual.longitud] <> Espacio then
                write(lineaActual.contenido[lineaActual.longitud])
        end; (* EscribirLineaSinJustificar *)
    end;

begin (* EscribirLinea *)
    if justificacion then EscribirLineaJustificada(longLinea, longSangrado)
    else EscribirLineaSinJustificar;
    writeln
end; (* EscribirLinea *)

begin (* Principal *)
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 6.18. Justificacion de un texto *****');
    writeln('*****');
    writeln;
    InicializarPropLinea(sangrado, anchoLinea);
    InicializarLinea(sangrado);
    writeln;

```

```

write('El texto justificado es:');
writeln;
writeln;
ComenzarPalabra;
while not FinDeSecuenciaPalabra do begin
    if Igual(palabraActual, MarcaParrafo) then begin
        EscribirLinea(false, anchoLinea, sangrado);
        writeln;
        InicializarLinea(sangrado)
    end
    else begin
        if not CabeEnLinea(Longitud(palabraActual), anchoLinea) then begin
            EscribirLinea(true, anchoLinea, sangrado);
            InicializarLinea (0);
        end;
        AnyadirPalabraALinea;
        if CabeEnLinea(1, anchoLinea) then
            AnyadirCaracterALinea(Espacio)
        end;
        AvanzarPalabra
    end;
EscribirLinea(false, anchoLinea , sangrado);
writeln;
write('Pulse enter para continuar');
readln

end.

```

```

program GenF6_1;
(*
 * Algoritmo GenF6_1. Genera el fichero datos6_1.dat
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 *)

(* En este algoritmo se ha sustituido la secuencia de NotaAlumno
   por un fichero secuencial, 'datos6_1.dat' *)

uses crt;

const

```

```

    MAX_ALUMNOS = 200;

type
    NotaAlumno = record
        codigo : 1..MAX_ALUMNOS;
        nota : Real;
    end;
var
    f : file of NotaAlumno;
    alum : NotaAlumno;
    x: char;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo GenF6_1. Genera el fichero datos6_1.dat *****');
    writeln('*****');
    writeln;
    assign(f, 'datos6_1.dat');
    rewrite(f);
    repeat
        writeln;
        write('Introduzca el codigo del alumno (>= 1) (<= ', MAX_ALUMNOS, '): ');
        readln(alum.codigo);
        write('Introduzca la nota del alumno: ');
        readln(alum.nota);
        write(f, alum);
        write('""Desea introducir mas notas s/S (si),n/N (no)? ');
        readln(x);
    until x in ['n', 'N'];
    close(f);
    writeln;
    write('Pulse enter para continuar');
    readln

end.

```

```

program VerF6_1;
(*
* Algoritmo VerF6_1. Consulta todo el fichero datos6_1.dat
* Titulo del libro: Una introduccion a la programacion.
*
*           Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
*           Francisco J. Montoya Dato
*
*           Jose L. Fernandez Aleman
*
*           Maria J. Majado Rosales

```

```

* Fecha: 1/9/2005
* Capitulo 6. Tablas
*)

uses crt;

const
    MAX_ALUMNOS = 200;
type
    NotaAlumno = record
        codigo : 1..MAX_ALUMNOS;
        nota : real;
    end;
var
    f : file of NotaAlumno;
    alum : NotaAlumno;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo VerF6_1. Consulta todo el fichero datos6_1.dat *****');
    writeln('*****');
    writeln;
    assign (f, 'datos6_1.dat');
    reset(f);
    writeln('** Notas de los alumnos (pulsa enter para ver siguiente) **');
    writeln;
    while not eof(f) do begin
        read(f, alum);
        write(' codigo: '); writeln(alum.codigo);
        write(' nota: '); writeln (alum.nota:1:2);
        readln;
    end;
    close(f);
    writeln;
    write('Pulse enter para continuar');
    readln

end.

```

```

program GenF6_4;
(*
* Algoritmo GenF6_4. Genera el fichero datos6_4.dat
* Titulo del libro: Una introduccion a la programacion.
* Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina

```

```

*           Francisco J. Montoya Dato
*           Jose L. Fernandez Aleman
*           Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 6. Tablas
* Fichero de salida: datos6_4.dat
*)

uses crt;

const
    MarcaFinNotas = -1;
type
    Estudiante = record
        nombre : string;
        edad : integer;
        sexo : boolean;
        nota : real;
    end;
    Festudiantes = file of Estudiante;

var
    f : Festudiantes;
    est : Estudiante;
    x: char;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo GenF6_4. Genera el fichero datos6_4.dat *****');
    writeln('*****');
    writeln;
    assign (f, 'datos6_4.dat');
    rewrite(f);
    (* Se introducen los datos de los estudiantes en la tabla *)
    writeln('Introduzca los datos de los estudiantes (nota=-1 para terminar)');
    writeln;
    write(' Introduzca nota: '); readln(est.nota);
    while est.nota <> MarcaFinNotas do begin
        write(' Introduzca el nombre: '); readln(est.nombre);
        write(' Introduzca la edad: '); readln(est.edad);
        repeat
            write(' M/m mujer, H/h hombre: ');
            readln(x);
        until x in ['m', 'M', 'h', 'H'];
        est.sexo := (x='M') or (x='m');
    end;
end;

```

```

        write(f,est);
        writeln;
        write(' Introduzca nota: '); readln(est.nota);
    end;
    close(f);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program VerF6_4;
(*
 * Algoritmo VerF6_4. Consulta todo el fichero datos6_4.dat
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 6. Tablas
 * Fichero de entrada: datos6_4.dat
 *)

uses crt;
type
    Estudiante = record
        nombre : string;
        edad : integer;
        sexo : boolean;
        nota : real;
    end;
    Festudiantes = file of Estudiante;

var
    f : Festudiantes;
    est : Estudiante;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo VerF6_4. Consulta todo el fichero datos6_4.dat *****');
    writeln('*****');
    writeln;
    assign (f, 'datos6_4.dat');

```

```
reset(f);
writeln('** Datos de los estudiantes (pulse enter para ver siguiente) **');
writeln;
while not eof(f) do begin
    read(f,est);
    writeln(' nombre: ', est.nombre);
    writeln(' edad: ', est.edad);
    write(' sexo: ');
    case est.sexo of
        TRUE: writeln('mujer');
        FALSE: writeln('hombre');
    end;
    writeln(' nota: ', est.nota:1:2);
    readln
end;
close(f);
writeln;
write('Pulse enter para continuar');
readln
end.
```

3.6. Capítulo 7

```

program Algoritmo7_1;
(*
  * Algoritmo 7.1. Funcion factorial recursiva
  * Titulo del libro: Una introduccion a la programacion.
  *
    Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
    Francisco J. Montoya Dato
  *
    Jose L. Fernandez Aleman
  *
    Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 7. Disenyo recursivo
  *)

uses crt;

var
  n : integer;

function Factorial (n : integer): longint;
(* PRE: n >= 0 *)
(* POST: Factorial (n) = n! *)
var
  resultado : longint;
begin
  if n = 0 then resultado := 1
  else if n > 0 then resultado := n * Factorial (n-1);
  Factorial := resultado;
end;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 7.1. Funcion factorial recursiva *****');
  writeln('*****');
  writeln;
  write('Introduzca un entero (>= 0) y (<= 16): ');
  readln(n);
  writeln;
  writeln('El factorial de ', n, ' es: ', Factorial (n));
  writeln;
  write('Pulse enter para continuar');
  readln;
end.

```

```
program Algoritmo7_2;
(*
 * Algoritmo 7.2. Funcion division natural recursiva
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 7. Disenyo recursivo
 *)

uses crt;

var
    dividendo, divisor, cociente, resto : integer;

(*
 * Pascal no admite que una funcion devuelve un valor de un tipo registro,
 * Por este motivo la funcion DivisionNatural se ha convertido en un procedimiento
 * con dos parametros paso por variable
 *)
procedure DivisionNatural (num, den : integer; VAR coc, res : integer);
(* PRE: (num >= 0) y (den > 0) *)
(* POST: DivisionNatural (num,den) = <c, r> y num = c * den + r y 0 <= r < den *)
begin
    if num < den then begin
        coc := 0;
        res := num
    end
    else begin
        DivisionNatural(num-den, den, coc, res);
        coc := coc + 1
    end
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.2. Funcion division natural recursiva *****');
    writeln('*****');
    writeln;
    write('Introduzca el dividendo (>= 0): ');
    readln(dividendo);
    writeln;
```

```

write('Introduzca el divisor (> 0): ');
readln(divisor);
DivisionNatural(dividendo, divisor, cociente, resto);
writeln;
writeln('Resultado: cociente = ', cociente, ', resto = ', resto);
writeln;
write('Pulse enter para continuar');
readln;
end.

```

```

program Algoritmo7_3;
(*
  * Algoritmo 7.3. Funcion producto escalar recursiva lineal
  * Titulo del libro: Una introduccion a la programacion.
  *
  *           Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 7. Disenyo recursivo
  *)

uses crt;

const
  MAX = 10;

type
  Vector = array [1..MAX] of real;

var
  vv1, vv2 : Vector;
  n, i : integer;

function pei (v1, v2 : Vector; a, b : integer) : real;
  (* PRE 1 <= a <= b <= MAX, y v1 y v2 tienen valores significativos en [a, b] *)
  (* POST pei (v1, v2, a, b) = sumatorio desde i = a hasta b de v1[i] * v2[i] *)
begin
  if a = b then pei := v1[a] * v2[a]
  else (* a < b *) pei := v1[a] * v2[a] + pei (v1, v2, a+1, b)
end;

begin
  clrscr;
  writeln;
  writeln('*****');

```

```
writeln('***** Algoritmo 7.3. Funcion producto escalar recursiva lineal *****');
writeln('*****');
writeln;
write('Introduzca el numero de elementos de los vectores (> 0) y (<=,MAX,'): ');
readln(n);
writeln;
writeln('Primer vector');
for i := 1 to n do begin
    write(' Componente ', i, ': ');
    readln(vv1[i]);
end;
writeln;
writeln('Segundo vector');
for i := 1 to n do begin
    write(' Componente ', i, ': ');
    readln(vv2[i]);
end;
writeln;
writeln('El producto escalar de los vectores es: ', pei (vv1, vv2, 1, n):1:2);
writeln;
write('Pulse enter para continuar');
readln;
end.
```

```
program Algoritmo7_4;
(*
 * Algoritmo 7.4. Funcion producto escalar recursiva lineal final
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 7. Disenyo recursivo
 *)

uses crt;

const
    MAX = 10;

type
    Vector = array [1..MAX] of real;

var
    vv1, vv2 : Vector;
```

```

n, i : integer;

function peii (v1, v2 : Vector; a, b : integer; c : real) : real;
(* PRE 1 <= a <= b <= MAX, y v1 y v2 tienen valores significativos en [a, b] *)
(* POST pei (v1, v2, a, b, c) = c + sumatorio desde i = a hasta b de v1[i] * v2[i] *)
begin
    if a = b then peii := c + v1[a] * v2[a]
    else (* a < b *) peii := peii (v1, v2, a+1, b, c + v1[a] * v2[a])
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.4. Funcion producto escalar recursiva lineal final *****');
    writeln('*****');
    writeln;
    write('Introduzca el numero de elementos de los vectores (> 0) y (<=,MAX,'): ');
    readln(n);
    writeln;
    writeln('Primer vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv1[i]);
    end;
    writeln;
    writeln('Segundo vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv2[i]);
    end;
    writeln;
    writeln('El producto escalar de los vectores es: ', peii (vv1, vv2, 1, n, 0):1:2);
    writeln;
    write('Pulse enter para continuar');
    readln;
end.

```

```

program Algoritmo7_5;
(*
* Algoritmo 7.5. Funcion producto escalar iterativa, equivalente a la del Algoritmo
  7.4
* Titulo del libro: Una introduccion a la programacion.
*                   Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*                   Francisco J. Montoya Dato

```

```

*                               Jose L. Fernandez Aleman
*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 7. Disenyo recursivo
*)

uses crt;

const
    MAX = 10;

type
    Vector = array [1..MAX] of real;

var
    vv1, vv2 : Vector;
    n, i : integer;

function peii (v1, v2 : Vector; a, b : integer; c : real) : real;
    (* PRE 1 <= a <= b <= MAX, y v1 y v2 tienen valores significativos en [a, b] *)
    (* POST pei (v1, v2, a, b, c) = c + sumatorio desde i = a hasta b de v1[i] * v2[
        i] *)
var
    a_loc : integer;      (* corresponde al parametro a *)
    c_loc : real;         (* corresponde al parametro c *)
                           (* el parametro b no varia, no necesita copia local *)

begin
    a_loc := a; c_loc := c;
    while a_loc < b do begin
        c_loc := c_loc + v1[a_loc] * v2[a_loc];
        a_loc := a_loc + 1
    end;
    peii := c_loc + v1[a_loc] * v2[a_loc]
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.5. Funcion producto escalar iterativa, *****');
    writeln('*****                               equivalente a la del Algoritmo 7.4 *****');
    writeln('*****');
    writeln;
    write('Introduzca el numero de elementos de los vectores (> 0) y (<=,MAX,'): ');
    readln(n);
    writeln;

```

```

    writeln('Primer vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv1[i]);
    end;
    writeln;
    writeln('Segundo vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv2[i]);
    end;
    writeln;
    writeln('El producto escalar de los vectores es: ', peii (vv1, vv2, 1, n, 0):1:2);
    writeln;
    write('Pulse enter para continuar');
    readln;
end.

```

```

program Algoritmo7_6;
(*
  * Algoritmo 7.6. Funcion producto escalar iterativa sin parametros de inmersion (
    version 1)
  * Titulo del libro: Una introduccion a la programacion.
  *
    Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
    Francisco J. Montoya Dato
  *
    Jose L. Fernandez Aleman
  *
    Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 7. Disenyo recursivo
  *)

uses crt;

const
    MAX = 10;

type
    Vector = array [1..MAX] of real;

var
    vv1, vv2 : Vector;
    n, i : integer;

function pe (v1, v2 : Vector) : real;
var
    a, b : integer;

```

```

        c : real;
begin
    a := 1; b := n; c := 0;
    while a < b do begin
        c := c + v1[a] * v2[a];
        a := a + 1
    end;
    pe := c + v1[a] * v2[a]
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.6. Funcion producto escalar iterativa sin *****');
    writeln('*****          parametros de inmersion (version 1) *****');
    writeln('*****');
    writeln;
    write('Introduzca el numero de elementos de los vectores (> 0) y (<=,MAX,'): ');
    readln(n);
    writeln;
    writeln('Primer vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv1[i]);
    end;
    writeln;
    writeln('Segundo vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv2[i]);
    end;
    writeln;
    writeln('El producto escalar de los vectores es: ', pe (vv1, vv2):1:2);
    writeln;
    write('Pulse enter para continuar');
    readln;
end.

```

```

program Algoritmo7_7;
(*
* Algoritmo 7.7. Funcion producto escalar iterativa (version 2)
* Titulo del libro: Una introduccion a la programacion.
*
*          Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
*          Francisco J. Montoya Dato
*
*          Jose L. Fernandez Aleman

```

```

*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 7. Disenyo recursivo
*)

uses crt;

const
    MAX = 10;

type
    Vector = array [1..MAX] of real;

var
    vv1, vv2 : Vector;
    n, i : integer;

function pe (v1, v2 : Vector) : real;
var
    a, b : integer;
    c : real;
begin
    a := 1; b := MAX; c := 0;
    while a <= b do begin
        c := c + v1[a] * v2[a];
        a := a + 1
    end;
    pe := c
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.7. Funcion producto escalar iterativa (version 2) *****');
    writeln('*****');
    writeln;
    write('Introduzca el numero de elementos de los vectores (> 0) y (<=,MAX,'): ');
    readln(n);
    writeln;
    writeln('Primer vector');
    for i := 1 to n do begin
        write(' Componente ', i, ': ');
        readln(vv1[i]);
    end;
    writeln;

```

```
writeln('Segundo vector');
for i := 1 to n do begin
    write(' Componente ', i, ': ');
    readln(vv2[i]);
end;
writeln;
writeln('El producto escalar de los vectores es: ', pe (vv1, vv2):1:2);
writeln;
write('Pulse enter para continuar');
readln;
end.
```

```
program Algoritmo7_8;
(*
 * Algoritmo 7.8. Funcion potencia natural recursiva
 * Titulo del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *          Francisco J. Montoya Dato
 *
 *          Jose L. Fernandez Aleman
 *
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 7. Disenyo recursivo
 *)

uses crt;

var
    base : real;
    exponente : integer;

function PotenciaNatural (a : real; n : integer) : real;
(* PRE (n >= 0) y (n = 0 => a <> 0) *)
(* POST PotenciaNatural(a, n) = a elevado a n *)

begin
    if n = 0 then
        PotenciaNatural := 1.0
    else (* n > 0 *)
        PotenciaNatural := a * PotenciaNatural(a, n-1);
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.8. Funcion potencia natural recursiva *****');
```

```

writeln('*****');
writeln;
write('Introduzca un real como base de la potencia: ');
readln(base);
writeln;
write('Introduzca un entero como exponente de la potencia (>= 0): ');
readln(exponente);
writeln;
writeln('Resultado: ', PotenciaNatural(base, exponente):1:2);
writeln;
write('Pulse enter para continuar');
readln

```

end.

```

program Algoritmo7_9;
(*
  * Algoritmo 7.9. Calculo recursivo de Fibo(n) con inmersión de resultados por
  * eficiencia
  * Título del libro: Una introducción a la programación.
  *                   Un enfoque algorítmico
  * Autores del libro: Jesus J. Garcia Molina
  *                   Francisco J. Montoya Dato
  *                   Jose L. Fernandez Aleman
  *                   Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 7. Diseño recursivo
  *)

uses crt;

var
  n, fib, fibAnt : integer;

(*
  * Pascal no admite que una función devuelve un valor de un tipo registro,
  * Por este motivo la función iFibo se ha convertido en un procedimiento con
  * dos parámetros paso por variable
  *)
procedure iFibo (n : integer; VAR a, b : integer);
(* PRE n > 0 *)
(* POST <a, b> = < Fibo(n), Fibo(n-1) > *)
var
  fib, fibAnt : integer;

begin
  if n = 1 then begin

```

```

        a := 1;
        b := 1; (* iF = < Fibo(1), Fibo(0) > *)
    end
    else (* n > 1 *) begin
        iFibo (n - 1, fib, fibAnt);
        (*
            * tenemos: fib = Fibo(n-1) y fibAnt = Fibo(n-2), luego:
            * Fibo(n) = Fibo(n-1) + Fibo(n-2) = fib + fibAnt
            * Fibo(n-1) = fib :
            *)
        a := fib + fibAnt;
        b := fib;
    end
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.9. Calculo recursivo de Fibo(n) con immersion *****');
    writeln('*****           de resultados por eficiencia           *****');
    writeln('*****');
    writeln;
    write('Termino de la sucesion que se desea calcular (> 0): ');
    readln(n);
    writeln;
    iFibo (n, fib, fibAnt);
    writeln('El termino ', n, '-esimo (posicion ', n+1, ') de la sucesion es: ', fib);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo7_10;
(*
    * Algoritmo 7.10. Funcion interfaz para la funcion del Algoritmo 7.9
    * Titulo del libro: Una introduccion a la programacion.
    *
    *           Un enfoque algoritmico
    * Autores del libro: Jesus J. Garcia Molina
    *
    *           Francisco J. Montoya Dato
    *
    *           Jose L. Fernandez Aleman
    *
    *           Maria J. Majado Rosales
    * Fecha: 1/9/2005
    * Capitulo 7. Disenyo recursivo
    *)

uses crt;

```

```

var
  n : integer;

(*
  * Pascal no admite que una funcion devuelve un valor de un tipo registro,
  * Por este motivo la funcion iFibo se ha convertido en un procedimiento con
  * dos parametros paso por variable
  *)
procedure iFibo (n : integer; VAR a, b : longint);
(* PRE n > 0 *)
(* POST <a, b> = < Fibo(n), Fibo(n-1) > *)
var
  fib, fibAnt : longint;
begin
  if n = 1 then begin
    a := 1;
    b := 1; (* a = Fibo(1), b = Fibo(0) *)
  end
  else begin (* n > 1 *)
    iFibo (n - 1, fib, fibAnt);
    (*
      * tenemos: fib = Fibo(n-1) y fibAnt = Fibo(n-2), luego:
      * Fibo(n) = Fibo(n-1) + Fibo(n-2) = fib + fibAnt
      * Fibo(n-1) = fib :
      *)
    a := fib + fibAnt; b := fib;
  end
end;

function Finterf (n : integer) : longint;
(* PRE n > 0 *)
(* POST Finterf (n) = Fibo(n) *)
var
  fib, fibAnt : longint;
begin
  if n = 0 then Finterf := 1
  else begin (* n > 0 *)
    (* se llama a iFibo y se descarta su segundo resultado *)
    iFibo (n, fib, fibAnt);
    Finterf := fib
  end
end;

begin
  clrscr;

```

```
writeln;
writeln('*****');
writeln('***** Algoritmo 7.10. Funcion interfaz para la funcion *****');
writeln('***** del Algoritmo 7.9 *****');
writeln('*****');
writeln;
write('Termino de la sucesion que se desea calcular (>= 0): ');
readln(n);
writeln;
writeln('El termino ', n, '-esimo (posicion ', n+1, ') de la sucesion es: ',
      Finterf (n));
writeln;
write ('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo7_11;
(*
 * Algoritmo 7.11. Funcion potencia natural mejorada
 * Titulo del libro: Una introduccion a la programacion.
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *           Francisco J. Montoya Dato
 *           Jose L. Fernandez Aleman
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 7. Disenyo recursivo
 *)

uses crt;

var
  base : real;
  exponente : integer;

function PotNat (a : real; n : integer) : real;
(* PRE (n >= 0) y (n = 0 => a <> 0) *)
(* POST PotNat(a, n) = a elevado a n *)

var
  p : real;
begin
  if n = 0 then
    PotNat := 1.0
  else (* n > 0 *) begin
    p := PotNat (a, n div 2);
    if n mod 2 = 0 then PotNat := p * p
```

```

        else PotNat := a * p * p
    end;
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.11. Funcion potencia natural mejorada *****');
    writeln('*****');
    writeln;
    write('Introduzca un real como base de la potencia: ');
    readln(base);
    writeln;
    write('Introduzca un entero como exponente de la potencia (>= 0): ');
    readln(exponente);
    writeln;
    writeln('Resultado: ', PotNat(base, exponente):1:2);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo7_12;
(*
 * Algoritmo 7.12. Funcion division natural mejorada
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 7. Disenyo recursivo
 *)

uses crt;

var
    dividendo, divisor, cociente, resto : integer;
(*
 * Pascal no admite que una funcion devuelve un valor de un tipo registro,
 * Por este motivo la funcion DivNat se ha convertido en un procedimiento
 * con dos parametros paso por variable
 *)
procedure DivNat(num, den : integer; VAR coc, res : integer);
(*

```

```

* PRE: (num >= 0) y (den > 0)
* POST: DivNat (num, den, c, r) y num = coc * den + res y 0 <= res < den
*)
begin
    if num < den then
        begin
            coc := 0;
            res := num
        end
    else
        begin
            DivNat(num, den * 2, coc, res);
            coc := coc * 2;
            if res >= den then
                begin
                    coc := coc + 1;
                    res := res - den
                end
            end
        end
    end;
begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.12. Funcion division natural mejorada *****');
    writeln('*****');
    writeln;
    write('Introduzca el dividendo (>= 0): ');
    readln(dividendo);
    writeln;
    write('Introduzca el divisor (> 0): ');
    readln(divisor);
    DivNat(dividendo, divisor, cociente, resto);
    writeln;
    writeln('Resultado: cociente = ', cociente, ', resto = ', resto);
    writeln;
    write('Pulse enter para continuar');
    readln
end.

```

```

program Algoritmo7_13;
(*
* Algoritmo 7.13. Accion recursiva que resuelve el problema de las torres de Hanoi
* Titulo del libro: Una introduccion a la programacion.
*
* Un enfoque algoritmico

```

```

* Autores del libro: Jesus J. Garcia Molina
*                   Francisco J. Montoya Dato
*                   Jose L. Fernandez Aleman
*                   Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 7. Disenyo recursivo
*)

uses crt;

const
    MAX = 15;

type
    Estaca = 'A'..'C'; (* El tipo Estaca sera un rango de caracteres *)

var
    ndiscos : integer;

procedure Hanoi(n : integer; origen, intermedio, destino: Estaca);
    (* PRE ...          la explicada en el texto *)
    (* POST ...         la explicada en el texto *)
    procedure mover(X, Y: Estaca);
        (* PRE ...          la explicada en el texto *)
        (* POST ...         la explicada en el texto *)
        begin (* de la accion mover *)
            (*
                * En nuestro caso la accion mover se limitara a
                * mostrar por la salida el movimiento realizado.
                * En general podria involucrar la modificacion del
                * estado de las estacas, si quisieramos llevar cuenta
                * del estado del juego en cada momento. Por esta
                * razon los parametros de tipo Estaca de la accion
                * Hanoi son de tipo dato-resultado.
                *)
            writeln(' mover disco de la estaca ', X, ' a la estaca ', Y)
        end;
    begin (* de la accion Hanoi *)
        case n of
            1 : mover(origen, destino)
            else (* n > 1 *) begin
                Hanoi(n-1, origen, destino, intermedio);
                mover(origen, destino);
                Hanoi(n-1, intermedio, origen, destino)
            end
        end
    end
end;

```

```
begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.13. Accion recursiva que resuelve el *****');
    writeln('***** problema de las torres de Hanoi *****');
    writeln('*****');
    writeln;
    write('Numero de discos (maximo ', MAX, '): ');
    readln(ndiscos);
    writeln;
    Hanoi(ndiscos, 'A', 'B', 'C');
    writeln;
    write('Pulse enter para continuar');
    readln
end.
```

```
program Algoritmo7_14_15;
(*
 * Algoritmo 7.14. Algoritmo de ordenacion rapida de Hoare
 * Algoritmo 7.15. Accion de particionamiento para la accion de clasificacion rapida
 * Titulo del libro: Una introduccion a la programacion.
 *
 * Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 * Francisco J. Montoya Dato
 *
 * Jose L. Fernandez Aleman
 *
 * Maria J. Majado Rosales
 * Capitulo 7. Diseno recursivo
 *)

(*
 * HOARE
 *
 * Implementa el algoritmo de clasificacion rapida de C. A. R. Hoare.
 * Solicita los datos para la generacion de los elementos del vector
 * (valores enteros): semilla aleatoria, numero de elementos que se
 * desean generar (con un maximo determinado por la constante MAX), y
 * valor maximo que se generara. Muestra el vector antes y despues de
 * la clasificacion, y comprueba que efectivamente esta ordenado tras
 * la llamada a la funcion de clasificacion.
 *)

uses crt;

const
    MAX = 3000;
```

```

type
  TipoBase = integer;
  indice = 1..MAX;
  TDatos = array [indice] of TipoBase;

var
  t : TDatos;
  max_real : indice;

procedure Comprueba(v : TDatos; mr : indice);
(*
  * Comprueba que los 'mr' primeros elementos del vector 'v' esten
  * efectivamente ordenados. En caso contrario imprime un mensaje
  * de error indicando la posicion 'i' donde hay una inversion, es
  * decir, donde v[i] > v[i+1].
  *)
var
  i : indice;
  error : boolean;
begin
  i := 1;
  error := false;
  while (i < mr) and not error do
    if v[i] > v[i+1] then begin
      write('<<< ERROR >>> (i=', i, ')');
      error := true
    end
    else
      i := i + 1
  end;

procedure InitTDatos(VAR v : TDatos; VAR mr : indice);
var
  i : indice;
  seed, maximo : integer;
begin
  write('Semilla aleatoria: ');
  readln(seed);
  RandSeed := seed;
  writeln;
  write('Elementos en el vector (minimo 1, maximo ', MAX, '): ');
  readln(mr);
  writeln;
  write('Generar valores entre cero y: ');
  readln(maximo);
  for i := 1 to mr do v[i] := Random(maximo+1)

```

```

end;

procedure MostrarTDatos(v : TDatos; mr : indice);
var
    i : indice;
begin
    i := 1;
    write(v[i]);
    while i <> mr do begin
        write(', ');
        i := i + 1;
        write(v[i])
    end;
    writeln
end;

procedure ClasificacionRapida(VAR v : TDatos; elementos : indice);
    procedure ClasificacionRapidaI(VAR v : TDatos; inf, sup : integer);
    (*
        * Los parametros 'inf' y 'sup' son de tipo 'integer' en lugar de
        * ser de tipo 'indice' porque en las llamadas recursivas que se
        * hacen en esta funcion con los valores ('inf', 'p-1') y
        * ('p+1', 'sup') el valor de 'p+1' podria superar el limite
        * superior del tipo 'indice', mientras que el de 'p-1' podria
        * caer por debajo de su limite inferior, como asi ocurre, de
        * hecho.
        *)
        var
            p : indice;
        procedure Particion(VAR v : TDatos; inf, sup : indice;
            VAR pivote : indice);
            var
                iz, dr : indice;

                procedure Intercambio(VAR v : TDatos; i, j : indice);
                var
                    t : TipoBase;
                begin
                    t := v[i];
                    v[i] := v[j];
                    v[j] := t
                end;

            begin
                iz := inf + 1;
                dr := sup;
                while iz <> dr + 1 do begin
                    while (iz <= dr) and (v[iz] <= v[inf]) do iz := iz + 1;

```

```

        while (iz <= dr) and (v[dr] >= v[inf]) do dr := dr - 1;
        if iz < dr then begin
            Intercambio(v, iz, dr);
            iz := iz + 1;
            dr := dr - 1
        end
    end;
    Intercambio(v, inf, dr);
    pivote := dr
end;

begin
    if inf < sup then begin
        Particion(v, inf, sup, p);
        ClasificacionRapidaI(v, inf, p-1);
        ClasificacionRapidaI(v, p+1, sup)
    end
end;

begin
    ClasificacionRapidaI(v, 1, elementos)
end;

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 7.14_15. óOrdenacin rpida de Hoare');
    writeln('*****');
    writeln;
    InitTDatos(t, max_real);
    writeln;
    writeln('Vector antes de la clasificacion:');
    MostrarTDatos(t, max_real);
    write ('Enter para ordenar...');
    readln;
    ClasificacionRapida(t, max_real);
    writeln;
    writeln('Vector despues de la clasificacion:');
    MostrarTDatos(t, max_real);
    Comprueba(t, max_real);
    writeln;
    write ('Pulse enter para continuar');
    readln
end.

```

3.7. Capítulo 8

```
program Algoritmo8_1;
(*
 * Algoritmo 8.1. Comprobar el equilibrio de los signos puntuacion de apertura/cierre
 * Titulo del libro: Una introduccion a la programacion.
 *
 *           Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *           Francisco J. Montoya Dato
 *
 *           Jose L. Fernandez Aleman
 *
 *           Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 8. Estructuras de datos dinamicas
 * Ficheros de prueba: dat8_1A.txt, dat8_1B.txt, dat8_1C.txt y dat8_1D.txt
 *
 * SIGNOS
 *
 * Comprueba el equilibrio de los signos de puntuacion parentizados
 * (parentesis, corchetes y llaves) de un fichero cuyo nombre se le
 * solicita por teclado al usuario.
 *
 * A diferencia del algoritmo presentado en el capitulo de EE.DD.
 * dinamicas, este programa no solo dice si el texto es correcto o
 * no, sino que en este ultimo caso informa ademas del punto del
 * texto exacto y los signos que causaron el error.
 *
 * Para ello, al recorrer la secuencia de entrada se va llevando
 * cuenta de la posicion por la que en cada momento se va en el reco-
 * rrido (numero de linea y columna dentro de esta). Cuando se en-
 * cuentra un signo de apertura, no solo se guarda en la pila el
 * signo en cuestion, sino tambien el punto (linea y columna) donde
 * este se encontro. De este modo, ante la ocurrencia de un error
 * es posible notificar exactamente los signos que lo produjeron y
 * las posiciones del texto donde estos se encuentran.
 *
 * Para detectar la condicion de error, en lugar de llevar una unica
 * variable booleana llamada 'error', como en el algoritmo del libro,
 * utilizaremos dos variables distintas que nos sirvan para identi-
 * ficar el tipo de error exacto. Los nombres y significados de estas
 * variables son los siguientes:
 *
 * error_pila: esta variable se pondra a VERDADERO cuando se encuentre
 *             un signo de cierre y la pila este vacia (es decir, no
 *             haya signos de apertura pendientes de cerrar).
 *
 * error_signo: esta variable se pondra a VERDADERO cuando se encuen-
 *             tre un signo de cierre que no se corresponda con el
```

```

*           ultimo signo de apertura encontrado en el texto.
*)

uses crt, unitmsc1;

const
  APERTURAS = [ '[', '(', '{' ]; (* conjunto de signos de apertura *)
  CIERRES   = [ ']', ')', '}' ]; (* conjunto de signos de cierre *)

type
  datos_pila =
    record
      signo : char;
      linea, columna : integer
    end;

  pila_signos = ^nodo_signos;
  nodo_signos =
    record
      datos : datos_pila;
      sig : pila_signos
    end;

var
  S : msc1;
  p : pila_signos;
  d : datos_pila;
  nombre_fichero : string;
  error_signo, error_pila : boolean;
  linea_actual,          (* linea actual del recorrido *)
  columna_actual : integer; (* columna actual del recorrido *)

(* ----- Operaciones de la Pila ----- *)

function PilaVacía : pila_signos;
begin
  PilaVacía := NIL
end;

function EsPilaVacía(p : pila_signos) : boolean;
begin
  EsPilaVacía := p = PilaVacía
end;

function Apilar(p : pila_signos; d : datos_pila) : pila_signos;
var
  q : pila_signos;

```

```

begin
    new(q);
    q^.datos := d;
    q^.sig := p;

    Apilar := q
end;

procedure Cima(p : pila_signos; VAR c : datos_pila);
begin
    c := p^.datos
end;

function Desapilar(p : pila_signos) : pila_signos;
var
    q : pila_signos;
begin
    q := p^.sig;
    dispose(p);

    Desapilar := q
end;

function DestruirPila(p : pila_signos) : pila_signos;
begin
    while not EsPilaVacía(p) do p := Desapilar(p);

    DestruirPila := p
end;

(* ----- Operaciones del Programa ----- *)

function SignoCierre(apertura : char) : char;
(*
 * Devuelve el signo de cierre correspondiente al signo
 * de apertura 'apertura'.
 *
 * Pre: el caracter 'apertura' es un signo de apertura
 *)
var
    res : char;
begin
    case apertura of
        '[' : res := ']';
        '(' : res := ')';
        '{' : res := '}';
    end;
end;

```

```

    SignoCierre := res
end;

(* ----- Programa Principal ----- *)

begin
    clrscr;
    writeln;
    writeln('*****');
    writeln('***** Algoritmo 8.1. Comprobar el equilibrio de los signos puntuacion
        *****');
    writeln('*****          de apertura/cierre          *****');
    writeln('*****');
    writeln;
    TratamientoInicial_MSC1(S);
    Cargar_Fichero_MSC1(S, 'dat8_1A.txt');
    writeln('La secuencia de entrada es:');
    writeln;
    Comenzar_MSC1 (S);
    while (EA_MSC1 (S) <> MSC1_MarcaFin) do begin
        if (EA_MSC1(S) = MSC1_MarcaFinLinea) then writeln
        else write(EA_MSC1(S));
        Avanzar_MSC1 (S)
    end;
    writeln;
    writeln;
    Comenzar_MSC1(S);
    if EA_MSC1(S) = MSC1_MarcaFin then
        writeln('Secuencia vacia.')
    else begin
        p := PilaVacía;
        linea_actual := 1;
        columna_actual := 1;
        error_signo := false;
        error_pila := false;
        repeat
            if EA_MSC1(S) in APERTURAS then begin
                d.signo := EA_MSC1(S);
                d.linea := linea_actual;
                d.columna := columna_actual;
                p := Apilar(p, d)
            end
            else if EA_MSC1(S) in CIERRES then begin
                if EsPilaVacía(p) then
                    error_pila := true
                else begin
                    Cima(p, d);

```

```

        p := Desapilar(p);
        error_signo := EA_MSC1(S) <> SignoCierre(d.signo)
    end
end
else if EA_MSC1(S) = MSC1_MarcaFinLinea then
begin
    linea_actual := linea_actual + 1;
    columna_actual := 0
end;
(*
 * Avanzamos solo si no ha habido error, para preservar
 * en ese caso los valores de la posicion actual y poder
 * mostrarlos en el mensaje de error
 *)
if not (error_signo or error_pila) then begin
    Avanzar_MSC1(S);
    columna_actual := columna_actual + 1
end
until (EA_MSC1(S) = MSC1_MarcaFin) or error_signo or error_pila;
(*
 * Analizamos la razon por la que se detuvo la iteracion:
 *
 * error_pila: se encontro (en el EA) un signo de cierre cuando
 * no habia pendientes de cierre ningun signo de apertura.
 *
 * error_signo: se encontro (en el EA) un signo de cierre que
 * no se corresponde con el ultimo signo de apertura
 * encontrado (que esta en 'd').
 *
 * no hay error, pero la pila no es vacia: el texto se agoto y
 * los signos de apertura que quedan en la pila no se
 * cerraron.
 *
 * en otro caso: se alcanzo la marca de fin sin errores, y en
 * la pila no quedan signos de apertura sin cerrar, luego
 * el texto es correcto.
 *)
if error_pila then begin
    write('Error: signo de cierre ', EA_MSC1(S), ' en ');
    writeln('linea ', linea_actual, ', columna ', columna_actual, ',');
    writeln('no tiene signo de apertura correspondiente.')
end
else if error_signo then begin
    write('Error: signo de cierre ', EA_MSC1(S), ' en ');
    writeln('linea ', linea_actual, ', columna ', columna_actual, ',');
    write('no se corresponde con signo de apertura ');
    write('', d.signo, ' en linea ', d.linea);

```

```

        writeln(', columna ', d.columna);
    end
    else if not EsPilaVacía(p) then begin
        writeln('Error: se agoto el texto con los siguientes ');
        writeln('signos de apertura pendientes de cerrar:');
        writeln;
        repeat
            Cima(p, d);
            p := Desapilar(p);
            write(' signo ', d.signo, ': linea ');
            writeln(d.linea, ', columna ', d.columna)
        until EsPilaVacía(p)
    end
    else
        writeln('Texto correcto.');
```

p := DestruirPila(p)
 end;
 writeln;
 write('Pulse enter para continuar');
 readln
end.

```

program Algoritmo8_2;
(*
 * Algoritmo 8.2. Crear un archivo secuencial que almacena enteros
 * Titulo del libro: Una introduccion a la programacion.
 *
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *
 *          Francisco J. Montoya Dato
 *
 *          Jose L. Fernandez Aleman
 *
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 8. Estructuras de datos dinamicas
 *)

uses crt;

var
    a : file of integer;
    i : integer;
    nombreFich : string;

begin
    clrscr;
    writeln;
    writeln('*****');
```

```
writeln('***** Algoritmo 8.2. Crear un archivo secuencial que *****');
writeln('*****          almacena enteros          *****');
writeln('*****');
writeln;
write('Introduzca el nombre del fichero: ');
readln(nombreFich);
writeln;
assign(a, nombreFich);
rewrite(a);
write(' Introduzca un numero entero (0 para terminar): ');
readln(i);
while i <> 0 do begin
    write(a, i);
    write(' Introduzca un numero entero (0 para terminar): ');
    readln(i)
end;
close(a);
writeln;
writeln('Fichero ', nombreFich, ' creado');
writeln;
write('Pulse enter para continuar');
readln
end.
```

```
program Algoritmo8_3;
(*
 * Algoritmo 8.3. Recorrido de un archivo secuencial de enteros para mostrarlos
 * Titulo del libro: Una introduccion a la programacion.
 *          Un enfoque algoritmico
 * Autores del libro: Jesus J. Garcia Molina
 *          Francisco J. Montoya Dato
 *          Jose L. Fernandez Aleman
 *          Maria J. Majado Rosales
 * Fecha: 1/9/2005
 * Capitulo 8. Estructuras de datos dinamicas
 *)

uses crt;

var
    a : file of integer;
    i : integer;
    nombreFich : string;
begin
    clrscr;
    writeln;
    writeln('*****');
```

```

writeln('***** Algoritmo 8.3. Recorrido de un archivo secuencial de enteros *****')
;
writeln('*****          para mostrarlos          *****');
writeln('*****');
writeln;
write('Introduzca el nombre del fichero de entrada: ');
readln(nombreFich);
assign(a, nombreFich);
writeln;
write('El fichero consta de los siguientes elementos: ');
(*
  * Notese que este algoritmo sigue el primer esquema de
  * recorrido del segundo modelo de acceso secuencial
  * en el que se tiene:
  * Iniciar: Reset(a)
  * EsUltimo: eof(a)
  * Avanzar: read(a, i)
  *)
reset(a);
while not eof(a) do begin
    read(a, i);
    write(i, ' ')
end;
close(a);
writeln;
writeln;
write('Pulse enter para continuar');
readln
end.

```

```

program Algoritmo8_4;
(*
  * Algoritmo 8.4. Acceso directo a un archivo para modificar un elemento
  * Titulo del libro: Una introduccion a la programacion.
  *
  *          Un enfoque algoritmico
  * Autores del libro: Jesus J. Garcia Molina
  *
  *          Francisco J. Montoya Dato
  *
  *          Jose L. Fernandez Aleman
  *
  *          Maria J. Majado Rosales
  * Fecha: 1/9/2005
  * Capitulo 8. Estructuras de datos dinamicas
  *)

uses crt;

var
    a : file of integer;

```

```

p, i : integer;
nombreFich : string;

begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 8.4. Acceso directo a un archivo para *****');
  writeln('*****          modificar un elemento          *****');
  writeln('*****');
  writeln;
  write('Introduzca el nombre del fichero: ');
  readln(nombreFich);
  assign(a, nombreFich);
  reset(a);
  writeln;
  write('Introduzca la posicion que quiere modificar (>= 0): ');
  readln(p);
  writeln;
  if (p < 0) or (p > FileSize(a)-1) then
    writeln('Posicion fuera del archivo')
  else (* (p >= 0) and (p < FileSize(a)) *) begin
    seek(a, p);
    read(a, i);
    writeln('Valor actual en la posicion ', p, ': ', i);
    writeln;
    write('Introduce nuevo valor: ');
    readln(i);
    seek(a, p);
    write(a, i);
    writeln;
    writeln('Fichero ', nombreFich, ' actualizado');
  end;
  close(a);
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```

```

program Algoritmo8_5;
(*
* Algoritmo 8.5. Anyadir un elemento al final de un archivo
* Titulo del libro: Una introduccion a la programacion.
*
*          Un enfoque algoritmico
* Autores del libro: Jesus J. Garcia Molina
*
*          Francisco J. Montoya Dato
*
*          Jose L. Fernandez Aleman

```

```

*                               Maria J. Majado Rosales
* Fecha: 1/9/2005
* Capitulo 8. Estructuras de datos dinamicas
*)

uses crt;

var
  a : file of integer;
  i : integer;
  nombreFich : string;
begin
  clrscr;
  writeln;
  writeln('*****');
  writeln('***** Algoritmo 8.5. Anyadir un elemento al final de un archivo *****');
  writeln('*****');
  writeln;
  write('Introduzca el nombre del fichero: ');
  readln(nombreFich);
  assign(a, nombreFich);
  reset(a);
  writeln;
  write('Introduzca un dato entero para anyadir al final del fichero: ');
  readln(i);
  seek(a, FileSize (a));
  write(a, i);
  close(a);
  writeln;
  writeln('Fichero ', nombreFich, ' actualizado');
  writeln;
  write('Pulse enter para continuar');
  readln
end.

```