Una introducción a la programación. Un enfoque algorítmico

Anexo I. Programas en C

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

21 de octubre de 2005

Índice

1. Introducción

Este documento incluye los programas C 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 la versión 4.9.7.0 del entorno Dev-C++ (compilador gcc) de Bloodshed, sobre los sistemas operativos Windows 98, 2000 y XP. Si el lector está más habituado a entornos Linux, puede utilizar el compilador gcc para Linux. Dev-C++ es un entorno de programación moderno potente y amigable, y se puede descargar desde el sitio web de Bloodshed http://www.bloodshed.net/devcpp.html.

El lector encontrará los programas en el fichero `C.zip` que puede obtener en el sitio web donde aparece este documento. Los programas están distribuidos en carpetas, una por cada capítulo. El nombre de los archivos hace referencia al identificador en el libro del algoritmo que implementa, por ejemplo `ALG6_4.C`, es el programa para el Algoritmo 6.4 del Capítulo 6.

Puesto que C 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 `lib` están los archivos cabecera, los archivos con el código fuente y las librerías compiladas de seis máquinas secuenciales, que corresponden con los dos primeros modelos de cada uno de los tres tipos de datos: caracteres, enteros y reales.

Los archivos cabecera y el código fuente de cada unidad se han nombrado con el prefijo "ms" seguido de una letra que indica el tipo de elemento ("c", "e" o "r") y un "1" o "2" para señalar el modelo. Por ejemplo, `mse2.h` es el archivo cabecera y `mse2.c` es el archivo con el código fuente para secuencias de enteros del segundo 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 el archivo cabecera de cada librería. El lector interesado también puede escudriñar el código de los archivos que contiene el código fuente de las librerías con el objetivo de aprender sobre estructuras de datos dinámicas (se han implementado como listas).

Para ejecutar los programas se han creado archivos que almacenan secuencias que se utilizan como ejemplo de entrada, 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.C`. 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.C` y `GENF6_4.C`) que crean los archivos de datos C 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.C` y `VERF6_4.C`).

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 la 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 y la implementación de las librerías del tercer y cuarto modelo de acceso secuencial.

1.1. Guía de usuario para Dev-C++

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):

- Establecer el directorio en el que se encuentran las librerías (opción **Opciones del compilador** del menú **Herramientas**). En la pestaña **Directorios**, seleccionar e introducir, si procede, los directorios donde se encontrarán los archivos cabecera (include) y las librerías. Este paso sólo sería necesario realizarlo una vez. Incluir las opciones del compilador necesarias para enlazar con las librerías que utilice el programa (opción **Opciones del compilador** del menú **Herramientas**). En la pestaña **Compilador**, activar "añadir los siguientes comandos al llamar al compilador" e incluir en el cuadro de texto las opciones que correspondan. Por ejemplo, si es preciso enlazar con la librería **libconio.a** y **libmsc1.a**, se introducirá la cadena "`-lconio -lmsc1 -lstdc++`". Para ejecutar nuestros programas siempre se deben enlazar con las librerías **libconio.a** y **libstdc++.a**, la primera incluye la función clrscr para limpiar la pantalla que se utiliza en todos los programas y la segunda es la librería estándar de C++ que es necesario por un error del compilador. En http://conio.sourceforge.net/ se pueden encontrar enlaces para descargar la librería libconio, así como una breve introducción y documentación sobre ella.
- Cargar el programa (opción **Abrir Proyecto** o **Archivo del menú Archivo**). Hay que asegurarse que en el nombre del archivo fuente la extensión es "C" mayúscula, ya que si fuese minúscula al compilar se producirán errores al enlazar con unidades.
- Compilar el programa (opción **Compilar** del menú **Ejecutar**).
- Ejecutar el programa (opción **Ejecutar** del menú **Ejecutar**).

Para generar una librería se deben seguir los siguientes pasos:

- Crear un proyecto con el nombre de la librería (por ejemplo libmsc1). Seleccionar **Static Library** de la opción **Nuevo proyecto** del menú **File**.
- Cargar el archivo que contiene el código fuente de la librería, opción **Añadir a proyecto** del menú **Proyecto**.
- Compilar el código fuente mediante la opción **Compile** del menú **Execute**. El resultado es una librería estática (en el ejemplo **libmsc1.a**) que será preciso incluir en el directorio de las librerías para poder enlazarla con un programa.

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 $\texttt{t}_k$ `= Tvieja`$_k$ en vez de `tk = Tviejak`.

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) ≠ 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

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main () {

    float notaTeoria;
    float notaPractica;
    float notaFinal;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.1. Calculo de la nota final de una asignatura *****\n");
    printf("*****\n");
    printf("\n");
    printf("Nota de teoria: ");
    scanf("%f",&notaTeoria);
    while (getchar() != '\n');
    printf("\n");
    printf("Nota de practicas: ");
    scanf("%f",&notaPractica);
    while (getchar() != '\n');
    printf("\n");
    notaFinal = notaTeoria * 0.7 + notaPractica * 0.3;
    printf("La nota final es: %.2f\n", notaFinal);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
```

```c
 * 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
 */

#include <stdio.h>
#include <conio.h>

float tempFahrenheit; // dato, temperatura en grados Fahrenheit
float tempCelsius;   // resultado, temperatura en grados Celsius

void ConvertirFahrenheitCelsius()
// PRE 0 <= tempFahrenheit <= 200
// POST convierte a grados celsius la temperatura de tempFahrenheit
{
    tempCelsius = (5.0 / 9.0) * (tempFahrenheit - 32.0);
}

int main()
{
     clrscr();
     printf("\n");
     printf("*****\n");
     printf("***** Algoritmo 2.2. Convertir temperatura Fahrenheit en *****\n");
     printf("*****              temperatura Celsius              *****\n");
     printf("*****\n");
     printf("\n");
     printf("Temperatura Fahrenheit (>= 0) y (<= 200): ");
     scanf("%f", &tempFahrenheit);
     while (getchar() != '\n');
     ConvertirFahrenheitCelsius();
     printf("La temperatura en grados Celsius es: %.2f\n", tempCelsius);
     printf("\nPulse enter para continuar");
     getchar();
     return 0;
}
```

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define IMPUESTO 0.20
#define SEGURO_MEDICO 0.05
#define COMPLEMENTO_QUINQUENIO 60
#define COMPLEMENTO_ANYO 6

long int salarioBase; // dato, sueldo base del trabajador
long int antiguedad; // dato, anyos en la empresa
double salarioNeto;  // resultado, salario percibido por el trabajador
long int salarioBruto; // salario bruto del trabajador
double descuentos;   // descuentos aplicados

void CalcularSalarioBruto()
// PRE salarioBase y antiguedad tienen un valor valido
// POST salarioBruto contiene el salario bruto del trabajador
{
    long int numeroQuinquenios;
    long int numeroAyos;
    long int pagoQuinquenios;
    long int pagoAyos;

    numeroQuinquenios = antiguedad / 5;
    numeroAyos = antiguedad % 5;
    pagoQuinquenios = numeroQuinquenios * COMPLEMENTO_QUINQUENIO;
    pagoAyos = numeroAyos * COMPLEMENTO_ANYO;
    salarioBruto = salarioBase + pagoQuinquenios + pagoAyos;
}

void 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
{
    descuentos = salarioBruto * (IMPUESTO + SEGURO_MEDICO);
}

void 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 */
{
    salarioNeto = salarioBruto - descuentos;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.3. Calculo del salario neto de un *****\n");
    printf("*****              trabajador (version 1)      *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el salario base: ");
    scanf("%ld",&salarioBase);
    while (getchar() != '\n');
    printf("Introduzca la antiguedad: ");
    scanf("%ld",&antiguedad);
    while (getchar() != '\n');
    CalcularSalarioBruto();
    CalcularDescuentos();
    CalcularSalarioNeto();
    printf("El sueldo neto es: %.2lf\n", salarioNeto);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

#define IMPUESTO 0.20
#define SEGURO_MEDICO 0.05
```

```
#define COMPLEMENTO_QUINQUENIO 60
#define COMPLEMENTO_ANYO 6

long int salarioBase; // dato, sueldo base del trabajador
int antiguedad; // dato, anyos en la empresa
double salarioNeto;      // resultado, salario percibido por el trabajador
long int salarioBruto;  // salario bruto del trabajador
double descuentos;       // descuentos aplicados
int numeroQuinquenios;
int numeroAnyos;
long int pagoQuinquenios;
long int pagoAnyos;

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.4. Calculo del salario neto de *****\n");
    printf("*****              un trabajador (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el salario base: ");
    scanf("%ld", &salarioBase);
    while (getchar() != '\n');
    printf("Introduzca la antiguedad: ");
    scanf("%ld", &antiguedad);
    while (getchar() != '\n');

    // Cslculo del salario bruto
    numeroQuinquenios = antiguedad / 5;
    numeroAnyos = antiguedad % 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;

    printf ("El sueldo neto es: %.2lf\n", salarioNeto);

    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

long int n; // dato, numero de bytes a descomponer
long int mb; // resultado, numero de megabytes
int kb;     // 0..1024, resultado, numero de kilobytes
int b;      // 0..1024, resultado, numero de bytes

void Convertir ()
    // PRE 0 <= n
    // POST (n = 1048576mb + 1024kb + b) y (0 <= kb < 1024) y (0 <= b < 1024)
{
    long int rb; // 0..1048575, resto de bytes
    mb = n / 1048576;
    rb = n % 1048576;
    // E1: n = 1048576 mb + rb y 0 <= rb < 1048576
    kb = rb / 1024;
    b = rb % 1024;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.5. Expresar una cantidad de bytes *****\n");
    printf("*****              en megabytes y kilobytes    *****\n");
    printf("*****\n");
    printf("\n");
    printf("Descomposicion en bytes, kilobytes y megabytes\n");
    printf("Introduzca un entero (>= 0): ");
    scanf("%ld", &n);
    while (getchar() != '\n');
    Convertir();
    printf("La descomposicion es:\n");
```

```
    printf("   Megabytes: %ld\n", mb);
    printf("   Kilobytes: %d\n", kb);
    printf("   Bytes: %d\n", b);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

/* No codificado */
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int nota; // 0..20, nota en la universidad extranjera, 0 <= nota <= 20

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.7. Correspondencia entre calificaciones (version 1) *****\
        n");
    printf("*****\n");
```

```
    printf("\n");
    printf("Introduzca la nota (>= 0) y (<= 20): ");
    scanf("%d", &nota);
    while (getchar() != '\n');
    printf("La calificacion es ");
    switch (nota)
    {
          case 20 : printf("matricula de honor"); break;
          case 19:
          case 18 : printf("sobresaliente"); break;
          case 17:
          case 16: printf("notable"); break;
          case 15:
          case 14: printf("aprobado"); break;
          default: if (nota < 14) printf("suspenso"); break;
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int nota;

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.8. Correspondencia entre calificaciones (version 2) *****\
        n");
    printf("*****\n");
```

```
    printf("\n");
    printf("Introduzca la nota (>= 0) y (<= 20): ");
    scanf("%d",&nota);
    while (getchar() != '\n');
    printf("La calificacion es ");
    switch (nota)
    {
        case 20: printf("matricula de honor"); break;
        case 19:
        case 18: printf("sobresaliente"); break;
        case 17:
        case 16: printf("notable"); break;
        case 15:
        case 14: printf("aprobado"); break;
        case 13:
        case 12:
        case 11:
        case 10:
        case 9:
        case 8:
        case 7:
        case 6:
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0: printf("suspenso"); break;
        default: printf("no valida"); break;
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */
```

```c
#include <stdio.h>
#include <conio.h>

int operando1, operando2;
char operador;

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.9. Simulacion de una calculadora simple *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el primer operando entero: ");
    scanf("%d", &operando1);
    while (getchar() != '\n');
    printf("Introduzca el segundo operando entero: ");
    scanf("%d", &operando2);
    while (getchar() != '\n');
    printf("Introduzca el operador (+, *, -, /): ");
    scanf("%c", &operador);
    while (getchar() != '\n');
    printf("El resultado de la operacion es: ");
    switch (operador)
    {
        case '+': printf("%d", operando1 + operando2); break;
        case '*': printf("%d", operando1 * operando2); break;
        case '-': printf("%d", operando1 - operando2); break;
        case '/': printf("%d", operando1 / operando2); break;
        default: printf("operador incorrecto"); break;
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int x, z;
int mayor;

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.10. Obtener el mayor de dos numeros enteros *****\n");
    printf("*****\n");
    printf("\n");
// PRE (x = X) y (z = Z)
// POST ((x >= z) => (mayor = X)) y ((z >= x) => (mayor = Z))
    printf("Introduzca el primer entero: ");
    scanf("%d", &x);
    while (getchar() != '\n');
    printf("Introduzca el segundo entero: ");
    scanf("%d", &z);
    while (getchar() != '\n');
    if (x >= z) mayor = x;
    else mayor = z;
    printf("El numero mayor es: %d\n", mayor);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
```

```
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0

unsigned int dia;
unsigned int mes;
long int anyo;
Booleano esBisiesto; // indicador de anyo bisiesto
Booleano fechaValida; // indicador de fecha valida

void AnyoBisiesto()
// POST esBisiesto es Verdadero si el anyo es bisiesto y Falso en caso contrario *)
{
       esBisiesto = (anyo % 4 == 0) && (anyo % 100 != 0) ||
                    (anyo % 400 == 0) && (anyo != 3600);
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.11. Comprobar si una fecha es correcta (version 1) *****\n
        ");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el dia: ");
    scanf("%u", &dia);
    while (getchar() != '\n');
    printf("Introduzca el mes: ");
    scanf("%u", &mes);
    while (getchar() != '\n');
    printf("Introduzca el anyo: ");
    scanf("%ld", &anyo);
    while (getchar() != '\n');

    fechaValida = Verdadero;
    if (dia < 1) fechaValida = Falso;
    else
        switch (mes)
        {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
```

```
            case 10:
            case 12: // meses de 31 dias
                        if (dia > 31) fechaValida = Falso; break;
                case 4:
            case 6:
            case 9:
            case 11: // meses de 30 dias *)
                        if (dia > 30) fechaValida = Falso; break;
            case 2:  // mes de febrero
                     AnyoBisiesto();
                     if ((dia > 29) || (!esBisiesto && (dia > 28)))
                        fechaValida = Falso; break;
            default: fechaValida = Falso; break;
          }
    printf("%u/%u/%ld", dia, mes, anyo);
    if (fechaValida) printf(" es una fecha valida\n");
    else printf(" no es una fecha valida\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define TRUE 1
#define FALSE 0

unsigned int dia;
unsigned int mes;
long int anyo;
int esBisiesto;  // indicador de anyo bisiesto
int fechaValida; // indicador de fecha valida

               // igual que en Algoritmo 2.11
```

```
void AnyoBisiesto()
// POST esBisiesto es Verdadero si el anyo es bisiesto y Falso en caso contrario *)
{
       esBisiesto = (anyo % 4 == 0) && (anyo % 100 != 0) ||
                    (anyo % 400 == 0) && (anyo != 3600);
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.12. Comprobar si una fecha es correcta (version 2) *****\n
        ");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el dia: ");
    scanf("%u", &dia);
    while (getchar() != '\n');
    printf("Introduzca el mes: ");
    scanf("%u", &mes);
    while (getchar() != '\n');
    printf("Introduzca el anyo: ");
    scanf("%ld", &anyo);
    while (getchar() != '\n');
    if (dia < 1) fechaValida = FALSE;
    else
        switch (mes)
        {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12: fechaValida = dia <= 31; break;
            case 4:
            case 6:
            case 9:
            case 11: fechaValida = dia <= 30; break;
            case 2: AnyoBisiesto();
                    fechaValida = (dia <= 29) && (esBisiesto || (dia <= 28));
                    break;
            default: fechaValida = FALSE; break;
        }
    printf("%u/%u/%ld", dia, mes, anyo);
    if (fechaValida) printf(" es una fecha valida\n");
```

```
    else printf(" no es una fecha valida\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define HORA_NORMAL 10
#define HORA_EXTRA 12
#define UMBRAL_HORAS_EXTRAS 140
#define IMPUESTO_BAJO 0.16
#define IMPUESTO_ALTO 0.22
#define UMBRAL_SUELDO 1300
#define COMPLEMENTO_QUINQUENIO 60
#define COMPLEMENTO_ANYO 6

unsigned int numeroHoras;  // dato, horas trabajadas en un mes por el empleado id
unsigned int id;                    // dato, identificador del empleado
unsigned int antiguedad;       // dato, anyos en la empresa
float salarioNeto;             // resultado, salario percibido por el trabajador
long int salarioBase;
long int salarioBruto;
float descuentos;

void CalcularSalarioBruto()
// PRE numeroHoras y antiguedad tienen un valor
// POST salarioBruto contiene el salario bruto del trabajador
{
    unsigned int numeroQuinquenios;
    unsigned int numeroAnyos;
    unsigned int pagoQuinquenios;
    unsigned int pagoAnyos;

// calcular sueldo base por horas trabajadas
```

```
    if (numeroHoras > UMBRAL_HORAS_EXTRAS)
      salarioBase = (numeroHoras - UMBRAL_HORAS_EXTRAS) * HORA_EXTRA
                  + UMBRAL_HORAS_EXTRAS * HORA_NORMAL;
       else salarioBase = numeroHoras * HORA_NORMAL;

// calcular gratificacion por antiguedad
     numeroQuinquenios = antiguedad / 5;
     numeroAnyos = antiguedad % 5;
     pagoQuinquenios = numeroQuinquenios * COMPLEMENTO_QUINQUENIO;
     pagoAnyos = numeroAnyos * COMPLEMENTO_ANYO;

// calcular salario bruto
     salarioBruto = salarioBase + pagoQuinquenios + pagoAnyos;
}

void CalcularDescuentos()
// PRE salarioBruto almacena el salario bruto calculado
// POST descuentos almacena el valor total de los descuentos sobre el salario bruto
{
    if (salarioBruto > UMBRAL_SUELDO)
      descuentos = salarioBruto * IMPUESTO_ALTO;
    else descuentos = salarioBruto * IMPUESTO_BAJO;
}

void 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 */
{
     salarioNeto = salarioBruto - descuentos;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.13. Calculo del salario neto de *****\n");
    printf("*****              un trabajador (version 3) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Identificador del empleado: ");
    scanf("%u", &id);
    while (getchar() != '\n');
    printf("Numero de horas trabajadas en un mes por el empleado: ");
    scanf("%u", &numeroHoras);
```

```
    while (getchar() != '\n');
    printf("Antiguedad del empleado: ");
    scanf("%u", &antiguedad);
    while (getchar() != '\n');
    CalcularSalarioBruto();
    CalcularDescuentos();
    CalcularSalarioNeto();
    printf("El salario del empleado %u es: %.2f\n", id, salarioNeto);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

/* No codificado */
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0

typedef struct {
               unsigned int dia; // 1..31
               unsigned int mes; // 1..12
```

```
                long int anyo;
                } Fecha;

Fecha f;             // dato, fecha dia/mes/anyo
Booleano esBisiesto; // indicador de ñao bisiesto
Booleano fechaValida; // indicador de fecha ávlida

void AnyoBisiesto()
// POST esBisiesto es Verdadero si f.anyo es bisiesto y Falso en caso contrario
{
       esBisiesto = (f.anyo % 4 == 0) && (f.anyo % 100 != 0) ||
                    (f.anyo % 400 == 0) && (f.anyo != 3600);
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 2.15. Comprobar si una fecha es correcta (version 3) *****\n
        ");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el dia: ");
    scanf("%u", &f.dia);
    while (getchar() != '\n');
    printf("Introduzca el mes: ");
    scanf("%u", &f.mes);
    while (getchar() != '\n');
    printf("Introduzca el anyo: ");
    scanf("%ld", &f.anyo);
    while (getchar() != '\n');
    switch (f.mes)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12: fechaValida = Verdadero; break;
        case 4:
        case 6:
        case 9:
        case 11: fechaValida = f.dia != 31; break;
        case 2: AnyoBisiesto();
                fechaValida = (f.dia <= 29) && (esBisiesto || (f.dia != 29));
```

```
                break;
    }
    printf("%u/%u/%ld", f.dia, f.mes, f.anyo);
    if (fechaValida) printf(" es una fecha valida\n");
    else printf(" no es una fecha valida\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

3.2. Capítulo 3

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#include <math.h>

typedef float Coordenada;
typedef struct {
               Coordenada x;
               Coordenada y;
               } Punto;

Punto p1, p2, p3, p4;
Punto po, pd;
float dist, perimetro;

void Distancia()
// PRE po y pd almacenan dos puntos
// POST dist almacena la distancia entre los puntos po y pd
{
     dist = sqrt(pow(pd.x - po.x, 2) + pow(pd.y - po.y, 2));
}

void LeerPunto (float *x, float *y, short int n)
{
    printf(" Coordenada x del punto %d: ", n);
    scanf("%f", x);
    while (getchar() != '\n');
    printf(" Coordenada y del punto %d: ", n);
    scanf("%f", y);
    while (getchar() != '\n');
    printf("\n");
}

int main()
```

```
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.1. Calcular el perimetro de un  *****\n");
    printf("*****              cuadrilatero irregular (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca los puntos del cuadrilatero (en orden de adyacencia): \n\n");
    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) */

    printf("El perimetro del cuadrilatero irregular es: %.2f\n", perimetro);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#include <math.h>

typedef float Coordenada;
typedef struct {
              Coordenada x;
              Coordenada y;
              } Punto;

Punto p1, p2, p3, p4;
float d1, d2, d3, d4;
float perimetro;

void Distancia(Punto po, Punto pd, float *dist)
// PRE po y pd almacenan dos puntos
// POST dist retorna la distancia entre los puntos po y pd

{
    *dist = sqrt(pow(pd.x - po.x, 2) + pow(pd.y - po.y, 2));
}

void LeerPunto(float *x, float *y, short int n)
{
   printf(" Coordenada x del punto %d: ", n);
   scanf("%f", x);
   while (getchar() != '\n');
   printf(" Coordenada y del punto %d: ", n);
   scanf("%f", y);
   while (getchar() != '\n');
   printf("\n");
}

int main()
{
```

```
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.2. Calcular el perimetro de un  *****\n");
    printf("*****              cuadrilatero irregular (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca los puntos del cuadrilatero: \n\n");
    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;

    printf("El perimetro del cuadrilatero irregular es: %.2f\n", perimetro);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int x, y;

void Intercambiar(int *a, int *b)
// PRE a, b : Entero, a = A, b = B
// POST a = B, b = A
{
```

```
    int t;

    t = *a;
    *a = *b;
    *b = t;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.3. Accion para el intercambio de dos variables *****\n");
    printf("*****\n");
    printf("\n");
    printf("Valor inicial de x: ");
    scanf("%d", &x);
    while (getchar() != '\n');
    printf("Valor inicial de y: ");
    scanf("%d", &y);
    printf("\n");
    while (getchar() != '\n');
    Intercambiar(&x, &y);
    printf("Valor final de x: %d\n", x);
    printf("Valor final de y: %d\n", y);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * Algoritmo 3.4. Desplazamiento circular de tres 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
 */

#include <stdio.h>
#include <conio.h>

int a, b, c;
```

```
void Intercambiar(int *x, int *y)
// PRE x, y : Entero, x = X, y = Y *)
// POST x = Y, y = X *)
{
    int t;

    t = *x;
    *x = *y;
    *y = t;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.4. Desplazamiento circular de tres variables *****\n");
    printf("*****\n");
    printf("\n");
// PRE a, b, c : Entero, a = A, b = B, c = C
// POST a, b, c : Entero, a = B, b = C, c = A
    printf("Introduzca tres enteros separados por espacios\n");
    printf("y termine pulsando enter: ");
    scanf("%d %d %d", &a, &b, &c);
    while (getchar() != '\n');
    printf("\n");
    printf("Valor inicial de a, b y c: %d, %d, %d\n", a, b, c);
    printf("\n");
// 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

    printf("Desplazamiento circular a la izquierda\n");
    printf("\n");
    printf("Valor final de a, b y c: %d, %d, %d\n", a, b, c);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

/* No codificado */
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#include <math.h>

typedef float Coordenada;
typedef struct {
               Coordenada x;
               Coordenada y;
               } Punto;
Punto p1, p2, p3, p4;
float perimetro;

float Distancia(Punto po, Punto pd)
// PRE po y pd almacenan dos puntos
// POST retorna la distancia entre los puntos po y pd
{
    return sqrt(pow(pd.x - po.x, 2) + pow(pd.y - po.y, 2));
}
void LeerPunto(float *x, float *y, short int n)
{
    printf(" Coordenada x del punto %d: ", n);
    scanf ("%f", x);
    while (getchar() != '\n');
```

```
    printf(" Coordenada y del punto %d: ", n);
    scanf ("%f", y);
    while (getchar() != '\n');
    printf("\n");
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.6. Calcular el perimetro de un  *****\n");
    printf("*****              cuadrilatero irregular (version 3) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca los puntos del cuadrilatero:\n");
    printf("\n");
    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);

    printf("El perimetro del cuadrilatero irregular es: %.2f", perimetro);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

long int x, y, z;
```

```
int Max3(long int a, long int b, long int c)
{
    int m;

    if ((a >= b) && (a >= c)) m = a;
    else if ((b >= a) && (b >= c)) m = b;
    else if ((c >= a) && (c >= b)) m = c;

    return m;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.7. Funcion que obtiene el mayor *****\n");
    printf("*****              de tres numeros (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca tres enteros separados por espacios\n");
    printf("y termine pulsando enter: ");
    scanf("%ld %ld %ld", &x, &y, &z);
    while (getchar() != '\n');
    printf("\n");
    printf("El maximo de %ld, %ld y %ld es: %ld", x, y, z, Max3(x, y, z));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
```

```
long int x, y, z;

int Max2 (long int a, long int b)
{
    return (a + b + abs (a - b)) / 2;
}

int Max3 (long int a, long int b, long int c)
{
    return Max2( a, Max2 (b, c));
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.8. Funcion que obtiene el mayor de *****\n");
    printf("*****              tres numeros (version 2)     *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca tres enteros separados por espacios\n");
    printf("y termine pulsando enter: ");
    scanf("%ld %ld %ld", &x, &y, &z);
    while (getchar() != '\n');
    printf("\n");
    printf("El maximo de %ld, %ld y %ld es: %ld", x, y, z, Max3 (x, y, z));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
```

```
typedef struct {
               long int hora;
               unsigned int min, seg; // 0..59
               } Duracion;

Duracion t1, t2; // entrada de datos
Duracion t3;    // salida de datos

Duracion SumarDuraciones(Duracion d1, Duracion d2)
{
    unsigned int x, y;
    Duracion res;

    y = (d1.seg + d2.seg) / 60;      // acarreo de segundos
    x = (d1.min + d2.min + y) / 60;  // acarreo de minutos
    res.hora = d1.hora + d2.hora + x;
    res.min = (d1.min + d2.min + y) % 60;
    res.seg = (d1.seg + d2.seg) % 60;
    return res;
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.9. Calcular la suma de dos    *****\n");
    printf("*****              duraciones de tiempo (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca dos duraciones:\n\n");
    printf(" Horas de la duracion 1: ");
    scanf("%d", &t1.hora);
    while (getchar() != '\n');
    printf(" Minutos de la duracion 1: ");
    scanf("%d", &t1.min);
    while (getchar() != '\n');
    printf(" Segundos de la duracion 1: ");
    scanf("%d", &t1.seg);
    while (getchar() != '\n');
    printf("\n");
    printf(" Horas de la duracion 2: ");
    scanf("%d", &t2.hora);
    while (getchar() != '\n');
    printf(" Minutos de la duracion 2: ");
    scanf("%d", &t2.min);
    while (getchar() != '\n');
```

```
    printf(" Segundos de la duracion 2: ");
    scanf("%d", &t2.seg);
    while (getchar() != '\n');
    t3 = SumarDuraciones(t1, t2);
    printf("\nLa suma de las duraciones es : %d-%d-%d", t3.hora, t3.min, t3.seg);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

typedef struct {
               long int hora;
               unsigned int min, seg; // 0..59
               } Duracion;

Duracion t1, t2; // entrada de datos
Duracion t3;    // salida de datos

long int ConvertirDS(Duracion d)
{
    return (3600 * d.hora + 60 * d.min + d.seg);
}
Duracion ConvertirSD(long int n)
{
    Duracion d;
    unsigned int rh; // 0..3599

    d.hora = n / 3600;
    rh = n % 3600;
    d.min = rh / 60;
    d.seg = rh % 60;
    return d;
```

```
}

Duracion SumarDuraciones(Duracion a, Duracion b)
{
    return ConvertirSD(ConvertirDS(a) + ConvertirDS(b));
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.10. Calcular la suma de dos    *****\n");
    printf("*****              duraciones de tiempo (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca dos duraciones:\n\n");
    printf(" Horas de la duracion 1: ");
    scanf("%d", &t1.hora);
    while (getchar() != '\n');
    printf(" Minutos de la duracion 1: ");
    scanf("%d", &t1.min);
    while (getchar() != '\n');
    printf(" Segundos de la duracion 1: ");
    scanf("%d", &t1.seg);
    while (getchar() != '\n');
    printf("\n");
    printf(" Horas de la duracion 2: ");
    scanf("%d", &t2.hora);
    while (getchar() != '\n');
    printf(" Minutos de la duracion 2: ");
    scanf("%d", &t2.min);
    while (getchar() != '\n');
    printf(" Segundos de la duracion 2: ");
    scanf("%d", &t2.seg);
    while (getchar() != '\n');

    t3 = SumarDuraciones(t1, t2);

    printf("\nLa suma de las duraciones es : %d-%d-%d", t3.hora, t3.min, t3.seg);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0

int diasMes[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

typedef struct {
               unsigned int dia; // 1..31;
               unsigned int mes; // 1..12;
               unsigned int anyo;
               } Fecha;

Fecha f; // fecha dia/mes/anyo

Booleano AnyoBisiesto(int anyo)
// PRE anyo es un entero
/* POST AnyoBisiesto es verdad si anyo es bisiesto
        y falso en caso contrario */
{
    return ((anyo % 4) == 0) && ((anyo % 100) != 0) ||
          ((anyo % 400) == 0) && (anyo != 3600);
}

int main()
{
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 3.11. Comprobar si una fecha es valida (version 3) *****\n")
        ;
    printf("*****\n");
    printf("\n");
    printf("Introduzca una fecha\n");
    printf("Introduzca el dia: ");
    scanf("%d", &f.dia);
```

```
    while (getchar() != '\n');
    printf("Introduzca el mes: ");
    scanf("%d", &f.mes);
    while (getchar() != '\n');
    printf("Introduzca el anyo: ");
    scanf("%d", &f.anyo);
    while (getchar() != '\n');
    if (AnyoBisiesto(f.anyo)) diasMes[1] = 29;
    printf("%d/%d/%d ", f.dia, f.mes, f.anyo);
    if (f.dia <= diasMes[f.mes-1]) printf("es una fecha valida\n");
    else printf("no es una fecha valida\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

3.3. Capítulo 4

```c
/*
 * Algoritmo 4.1. Calcular el valor medio de 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{

    float v; // nota leida
    float s; // lleva cuenta de la suma total
    int n;      // lleva cuenta del numero de notas

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.1. Calcular el valor medio de *****\n");
    printf("*****              las notas de una asignatura *****\n");
    printf("*****\n");
    printf("\n");
    s = 0;
    n = 0;
    printf("Introduzca una nota (valor <0 para terminar): ");
    scanf("%f", &v);
    while (getchar() != '\n');
    // E0 : s = 0 y n = 0
    while (v >= 0) {
          s = s + v;
          n = n + 1;
          printf("Introduzca una nota (valor <0 para terminar): ");
          scanf("%f", &v);
          while (getchar() != '\n');
     /* Ei : s = suma de las n notas introducidas hasta este paso
             y n = numero de notas introducidas hasta este paso */
     }
     /* Ef : s = suma de todas las notas y n = numero total de
```

```
            notas introducidas */
    if (n > 0) printf("Valor medio de las notas = %.2f\n", s/n);
    else printf("No hay valores\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * Algoritmo 4.2. Calcular el numero de ceros y 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_2.TXT
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"

int main()
{
    Msc1 s;         // secuencia de enteros
    int numCeros;   // contador del numero de ceros
    int numUnos;    // contador del numero de unos
    char c;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.2. Calcular el numero de ceros y unos *****\n");
    printf("*****              en una secuencia de caracteres  *****\n");
    printf("*****\n");
    printf("\n");

    s = Init_MSC1();
    Cargar_Fichero_MSC1(s,"datos4_2.txt");
    numCeros = 0;
    numUnos = 0;
    printf("La secuencia de entrada es: ");
    Comenzar_MSC1(s);
    while (EA_MSC1(s) != MSC1_MarcaFin()) {
```

```
            if (EA_MSC1(s) == '0') numCeros = numCeros + 1;
            else numUnos = numUnos + 1;
            printf("%c ", EA_MSC1(s));
            Avanzar_MSC1(s);
    }
    printf("\n");
    printf(" Numero de ceros = %d\n", numCeros);
    printf(" Numero de unos = %d\n", numUnos);

    printf("\nPulse enter para continuar");
    getchar();

    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"

int main()
{
    Msc1 s, t;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.3. Crear una secuencia de caracteres *****\n");
    printf("*****              a partir de otra existente     *****\n");
    printf("*****\n");
    printf("\n");
    s = Init_MSC1();
    t = Init_MSC1();
    Cargar_Fichero_MSC1(s,"datos4_3.txt");
    printf("La secuencia de entrada es: ");
```

```
    Comenzar_MSC1(s);
    Arrancar_MSC1(t);
    while (EA_MSC1(s) != MSC1_MarcaFin()) {
          if (EA_MSC1(s) == '*') Registrar_MSC1(t, '+');
          else Registrar_MSC1(t, EA_MSC1(s));
          printf("%c", EA_MSC1(s));
          Avanzar_MSC1(s);
    }
    Marcar_MSC1(t);
    printf("La secuencia de salida es: ");
    Comenzar_MSC1(t);
    while (EA_MSC1(t) != MSC1_MarcaFin()) {
          printf("%c", EA_MSC1(t));
          Avanzar_MSC1(t);
    }
    Salvar_Fichero_MSC1(t,"sal4_3.txt");
    printf("\n");
    printf("Grabado\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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 entrada: datos4_4.txt
 */

#include <stdio.h>
#include <conio2.h>
#include "msr1.h"

int main()
{
    Msr1 s;
    float suma; // lleva cuenta de la suma total de las notas
    int numElem; // lleva cuenta del numero de notas
```

```
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.4. Calcular el valor medio de una secuencia *****\n");
    printf("*****              con las notas de una asignatura      *****\n");
    printf("*****\n");
    printf("\n");

    s = Init_MSR1();
    Cargar_Fichero_MSR1(s,"datos4_4.txt");
    Comenzar_MSR1(s);
    printf("La secuencia de entrada es: ");
    suma = 0.0;
    numElem = 0;

// Eini : suma = 0 y numElem = 0 y ea = Primero (S) y INV = Verdadero
    while (EA_MSR1(s) != MSR1_MarcaFin()) {
    // 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;
           printf("%.2f ", EA_MSR1(s));
           Avanzar_MSR1(s);
    }
// Efin : INV y (EA (S) = MarcaFin)
    printf("\n");
    printf("\n");
    if (numElem > 0) printf("Valor medio de las notas = %.2f", suma/numElem);
    else printf("Secuencia vacia");
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
```

```
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"

int main()
{
    Msc1 s;
    int numCeros; // lleva cuenta del numero de '0'
    int numPares; // lleva cuenta del numero de pares

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.5. Calcular el numero de pares ceros-unos *****\n");
    printf("*****              en una secuencia de caracteres     *****\n");
    printf("*****\n");
    printf("\n");
    s = Init_MSC1();
    Cargar_Fichero_MSC1(s,"datos4_5.txt");
    Comenzar_MSC1(s);
    printf("La secuencia de entrada es: ");
    numCeros = 0;
    numPares = 0;
    // INV y (EA = Primero (S))
    while (EA_MSC1(s) != MSC1_MarcaFin()) {
    // INV : numPares = NumPares (Piz), numCeros = NumCeros (Piz)
    // INV y (EA_MSC1 <> MarcaFin)
          switch (EA_MSC1(s)) {
               case '0': numCeros = numCeros + 1; break;
               case '1': numPares = numPares + numCeros; break;
          }
          printf("%c", EA_MSC1(s));
          Avanzar_MSC1(s);
     }
     // INV y (EA = MarcaFin), numPares = NumPares (S) *)
    printf("\n");
    printf("\n");
    printf("El numero de pares 0-1 es: %d\n", numPares);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    int n, i;
    long int factorial;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.6. Calculo del factorial (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (< 17): ") ;
    scanf("%d", &n);
    while (getchar() != '\n');
    i = 0;
    factorial = 1;
    while (i < n) {
    // INV: factorial = i!
          i = i + 1;
          factorial = factorial * i;
    }
    printf("\n");
    printf("El factorial de %d es: %ld\n", n, factorial);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    int n, i;
    long int factorial;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.7. Calculo del factorial (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (< 17): ");
    scanf("%d", &n);
    while (getchar() != '\n');
    factorial = 1;
    for (i=2; i<=n; i++)
        factorial = factorial * i;
    // INV factorial = i!
    printf("\n");
    printf("El factorial de %d es: %ld\n", n, factorial);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
```

```c
#include <conio.h>

int main()
{
    int a, b, producto;
    int i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.8. Calculo del producto mediante sumas sucesivas *****\n")
        ;
    printf("*****\n");
    printf("\n");
    printf("Introduzca dos numeros enteros separados por blancos: ");
    scanf("%d %d", &a, &b);
    while (getchar() != '\n');
    producto = 0;
    for (i=1; i<=b; i++)
         producto = producto + a;
     // INV producto = a * i y   1 <= i <= b
     // INV y (i = b), producto = a * b = POST
    printf("\n");
    printf("  %d*%d=%d\n", a, b, producto);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    int p, c, r;
```

```
    int q;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.9. Calculo del cociente y resto de una division entera
        *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca dividendo (>= 0) y divisor (> 0) separados por blancos: ");
    scanf("%d %d", &p, &q);
    while (getchar() != '\n');
    r = p; c = 0;
    while (r >= q) {
    // INV (p = q * c + r) y (r >= 0)
          r = r - q;
          c = c + 1;
    }
    // INV y (r < q) => POST
    printf("\n");
    printf("Cociente = %d Resto = %d\n", c, r);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    unsigned int a, b, u, v;

    clrscr();
    printf("\n");
```

```
    printf("*****\n");
    printf("***** Algoritmo 4.10. Calculo del maximo comun divisor *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca dos enteros (> 0) separados por blancos: ");
    scanf("%u %u", &a, &b);
    while (getchar() != '\n');
    u = a; v = b;
    while (u != v)
    // INV (* mcd (a,b) = mcd (u,v)
        if (u > v) u = u - v;
        else v = v - u;
    //INV y (u = v)
    printf("\n");
    printf("Maximo comun divisor de %u y %u es: %u\n", a, b, u);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

long int Potencia2(int n)
// PRE n : Entero >= 0
// POST Potencia2 = 2^n
{
    long int p;
    int j;

    p = 1;
    for (j = 1; j<=n; j++)
        p = p * 2;
    // INV p = 2^j
    return p;
```

```
}

int main()
{
    int i;
    int x;       // indice inferior del intervalo de potencias
    int y;       // indice superior del intervalo

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.11. Calculo de las potencias de 2 (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca dos enteros (>= 0) en orden creciente separados por blancos: ");
    scanf("%d %d", &x, &y);
    while (getchar() != '\n');
    printf("\n");
    for (i = x; i <= y; i++)
        printf("La potencia %d-esima de 2 es: %ld\n", i, Potencia2(i));
        /* INV se han calculado y mostrado las potencias de 2
           en el intervalo [x,i] */
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    int x;            // indice inferior
    int y;            // indice superior
    long int potencia2; // resultado
```

```
    int i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.12. Calculo de las potencias de 2 (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca dos enteros (>= 0) en orden creciente separados por blancos: ");
    scanf("%d %d", &x, &y);
    while (getchar() != '\n');
    printf("\n");
    potencia2 = 1;
    for (i=1; i<=x; i++)
        potencia2 = potencia2 * 2;
    // INV potencia2 = 2^i
    printf("La potencia %d-esima de 2 es: %ld\n", x, potencia2);
    for (i=x+1; i<=y; i++) {
        potencia2 = potencia2 * 2;
        printf("La potencia %d-esima de 2 es: %ld\n", i, potencia2);
    /* INV potencia2 = 2^i y se han mostrado las potencias de 2
       en el intervalo [x,i] */
    }
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    long int ult, penult; // ultimo y penultimo termino de Fibonacci
    long int copiaFibo;  // valor Fibonacci en el paso i-1
```

```
    long int fibo;       // resultado
    int n;               // dato
    int i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.13. Calculo del termino enesimo *****\n");
    printf("*****              de la sucesion de Fibonacci *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (>= 0): ");
    scanf("%d", &n);
    while (getchar() != '\n');
    printf("\n");
    switch (n)
    {
         case 0 :
         case 1 : printf("El termino %d-esimo de la sucesion (posicion %d) es: 1\n", n,
              n+1);
                  break;
         default : penult = 1; ult = 1; fibo = 2;
                  for (i=3; i<=n; i++) {
                       copiaFibo = fibo;
                       fibo = fibo + ult;
                       penult = ult;
                       ult = copiaFibo;
                       /* INV (fibo = fi) y (ult = fi-1) y (penult = fi-2),
                               3 <= i <= n */
                  }
                  // INV y (i = n)
                  printf("El termino %d-esimo de la sucesion (posicion %d) es: %d\n", n,
                       n+1, fibo);
     }
     printf("\nPulse enter para continuar");
     getchar();
     return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    int n;   // dato
    int j;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.14. Comprobar si un numero es primo (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (> 0): ");
    scanf("%d", &n);
    while (getchar() != '\n');
    printf("\n");
    if (n == 1) printf("1 es primo\n");
    else
    {
        j = 2;
        while ((n % j) != 0)
               j = j + 1;
               // INV ( i : 2 <= i < j : (n % i) <> 0)
               // INV y (n % j = 0)
        if (j == n) printf("%d es primo\n", n);
        else printf("%d no es primo\n", n);
    }
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main() {
    int n;
    int j;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.15. Comprobar si un numero es primo (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (> 0): ");
    scanf("%d", &n);
    while (getchar() != '\n');
    printf("\n");
    if ((n == 1) || (n == 2) || (n == 3)) printf("%d es primo\n", n);
    else if ((n > 3) && ((n % 2) == 0)) printf("%d es par, no es primo\n", n);
    else if ((n > 3) && ((n % 2) != 0)) {
                                    j = 3;
                                    while (((n % j) != 0) && (j < (n / j)))
                                           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 % j) != 0) printf("%d es primo\n", n);
                                    else printf("%d no es primo\n", n);
    }
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
```

```
 */

#include <stdio.h>
#include <conio.h>

int main() {
    int n, rc; // n dato de entrada, rc parte entera de la raiz
    int i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.16. Calcular la parte entera de la raiz cuadrada *****\n")
        ;
    printf("*****                de un entero (version 1)                 *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (>= 0): ");
    scanf("%d", &n);
    printf("\n");
    while (getchar() != '\n');
    i = 1;
    while (n >= i * i)
    // INV ( j : 1 <= j < i : (j^2 <= n))
        i = i + 1;
    // INV y (n < i^2)
    rc = i - 1;
    printf("La parte entera de la raiz cuadrada de %d es: %d\n",n, rc);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
```

```c
#include <conio.h>

int main() {

    int n, rc;     // n dato, rc resultado
    int imp, i, c;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.17. Calcular la parte entera de la raiz cuadrada *****\n")
        ;
    printf("*****                de un entero (version 2)                  *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero (>= 0): ");
    scanf("%d", &n);
    while (getchar() != '\n');
    printf("\n");
    i = 1; c = 1; imp = 1;
    while (c <= n) {
    /* 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;
    }
    // INV y (n < c)
    rc = i - 1;
    printf("La parte entera de la raiz cuadrada de %d es: %d\n", n ,rc);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */
```

```c
#include <stdio.h>
#include <conio.h>

int main() {

    float x;          // dato entre 0 y 1
    float a;          // resultado
    float c;          // para el calculo de la sucesion
    float eps;        // error

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.18. Calcular 1/x a partir de una sucesion recurrente
        *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un valor real (<= 1) y (> 0): ");
    scanf("%f", &x);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca el error permitido: ");
    scanf("%f", &eps);
    while (getchar() != '\n');
    printf("\n");
    a = 1; c = 1 - x;
    while (c > eps) {
          a = a * (1 + c);
          c = c * c;
    }
    printf("El resultado de evaluar la expresion 1/%f es: %f\n", x, a);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */
```

```c
#include <stdio.h>
#include <conio.h>

int main()
{
    int x;        // dato
    int i;        // numero de termino
    float t, s;   // termino y suma de la serie
    float eps;    // error

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 4.19. Aproxima e elevado a x      *****\n");
    printf("*****              a partir de su desarrollo en serie *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un entero: ");
    scanf("%d", &x);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca el error permitido: ");
    scanf("%f", &eps);
    while (getchar() != '\n');
    t = 1; s = t; i = 0;
    while (t > eps) {
          i = i + 1;
          t = t * ((float) x / (float) i);
          s = s + t;
    }
    printf("\n");
    printf("e elevado a %d es: %f\n", x, s);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

3.4. Capítulo 5

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "mse1.h"

int main()
{
    int a, b;
    int numA, numB;
    int numPares;
    Mse1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.1. Calcular el numero de pares *****\n");
    printf("*****              (primer modelo, tercer esquema) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un par de enteros:\n");
    printf("  primer entero: ");
    scanf("%d", &a);
    while (getchar() != '\n');
    printf("  segundo entero: ");
    scanf("%d", &b);
    while (getchar() != '\n');
    S = Init_MSE1();
    Cargar_Fichero_MSE1(S,"datos5_1.txt");
    Comenzar_MSE1(S);
    if (EA_MSE1(S) == MSE1_MarcaFin()) printf("Secuencia vacia");
    else {
          printf("\n");
          printf("La secuencia de entrada es: ");
```

```
        printf("%d ", EA_MSE1(S));
        numPares = 0;
        if (EA_MSE1(S) == a) { numA = 1; numB = 0; }
        else if (EA_MSE1(S) == b) { numA = 0; numB = 1; }
        else { numA = 0; numB = 0; }
        // Se utiliza la equivalencia ITERAR-MIENTRAS
        Avanzar_MSE1(S);
        while (EA_MSE1(S) != MSE1_MarcaFin()) {
              if (EA_MSE1(S) == a) {
                                  numPares = numPares + numB;
                                  numA = numA + 1;
              }
              else if (EA_MSE1(S) == b) {
                                       numPares = numPares + numA;
                                       numB = numB + 1;
              }
              printf("%d ", EA_MSE1(S));
              Avanzar_MSE1(S);
        }
        printf("\n\n");
        printf("El numero de pares (%d, %d) (%d, %d) es: %d", a, b, b, a, numPares);
   }
   printf("\n");
   printf("\nPulse enter para continuar");
   getchar();
   return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "mse2.h"

int main()
{
```

```
    int a, b;
    int numA, numB;
    int numPares;
    Mse2 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.2. Calcular el numero de pares *****\n");
    printf("*****              (segundo modelo, tercer esquema) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un par de enteros:\n");
    printf("  primer entero: ");
    scanf("%d", &a);
    while (getchar() != '\n');
    printf("  segundo entero: ");
    scanf("%d", &b);
    while (getchar() != '\n');
    S = Init_MSE2();
    Cargar_Fichero_MSE2(S,"datos5_2.txt");
    Iniciar_MSE2(S);
    if (EsVacia_MSE2(S)) printf("Secuencia vacia");
    else {
          printf("\n");
          printf("La secuencia de entrada es: ");
          Avanzar_MSE2(S);
          printf("%d ", EA_MSE2(S));
          numPares = 0;
          if (EA_MSE2(S) == a) { numA = 1; numB = 0; }
          else if (EA_MSE2(S) == b) { numA = 0; numB = 1; }
          else { numA = 0; numB = 0; }
          while (!EsUltimo_MSE2(S)) {
                Avanzar_MSE2(S);
                printf("%d ", EA_MSE2(S));
                if (EA_MSE2(S) == a) {
                                  numPares = numPares + numB;
                                  numA = numA + 1;
                }
                else if (EA_MSE2(S) == b) {
                                       numPares = numPares + numA;
                                       numB = numB + 1;
                }
          }
          printf("\n");
          printf("\n");
          printf("El numero de pares (%d, %d) (%d, %d) es: %d", a, b, b, a, numPares);
```

```
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "mse1.h"
#define Booleano int
#define Verdadero 1
#define Falso 0


/*
 * Maquina abstracta del tercer modelo a partir de
 * una maquina secuencial del primer modelo
 */
    int aux;

void Comenzar_MSE3(Mse1 S)
{
    Comenzar_MSE1(S);
    aux = EA_MSE1(S);
    Avanzar_MSE1(S);
}

void Avanzar_MSE3(Mse1 S)
{
    aux = EA_MSE1(S);
    Avanzar_MSE1(S);
}
```

```
int EA_MSE3(Mse1 S)
{
    return aux;
}

void Cargar_Fichero_MSE3(Mse1 S, char *nombre)
{
    Cargar_Fichero_MSE1(S, nombre);
}

Booleano EsVacia_MSE3(Mse1 S)
{
    return EA_MSE1(S) == MSE1_MarcaFin();
}

Booleano EsUltimo_MSE3(Mse1 S)
{
    return EA_MSE1(S) == MSE1_MarcaFin();
}

Mse1 Init_MSE3()
{
    return Init_MSE1();
}

int main()
{
    int a, b;
    int numA, numB;
    int numPares;
    Mse1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.3. Calcular el numero de pares *****\n");
    printf("*****              (tercer modelo, segundo esquema) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un par de enteros:\n");
    printf("  primer entero: ");
    scanf("%d", &a);
    while (getchar() != '\n');
    printf("  segundo entero: ");
    scanf("%d", &b);
    while (getchar() != '\n');
    S = Init_MSE3();
```

```
    Cargar_Fichero_MSE3(S,"datos5_3.txt");
    Comenzar_MSE3(S);
    if (EsVacia_MSE3(S)) printf("Secuencia vacia");
    else {
          printf("\n");
          printf("La secuencia de entrada es: ");
          printf("%d ", EA_MSE3(S));
          numPares = 0;
              numA = 0;
              numB = 0;
          // Se utiliza la equivalencia ITERAR-MIENTRAS
          if (EA_MSE3(S) == a) {
                              numPares = numPares + numB;
                              numA = numA + 1;
          }
          else if (EA_MSE3(S) == b) {
                                   numPares = numPares + numA;
                                   numB = numB + 1;
          }
          while (!EsUltimo_MSE3(S)) {
                 Avanzar_MSE3(S);
                 printf("%d ", EA_MSE3(S));
                 if (EA_MSE3(S) == a) {
                                     numPares = numPares + numB;
                                     numA = numA + 1;
                 }
                 else if (EA_MSE3(S) == b) {
                                          numPares = numPares + numA;
                                          numB = numB + 1;
                 }
          }
          printf("\n");
          printf("\n");
          printf("El numero de pares (%d, %d) (%d, %d) es: %d", a, b, b, a, numPares);
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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: dat5_4.txt
 */

#include <stdio.h>
#include <conio2.h>
#include "mse1.h"
#define Booleano int
#define Verdadero 1
#define Falso 0

/*
 * Maquina abstracta del cuarto modelo a partir de
 * una maquina secuencial del primer modelo
 */


TipoBase_MSE1 MSE4_MarcaFin()
{
    return MSE1_MarcaFin();
}


    Booleano comenzar;

Mse1 Init_MSE4()
{
    return Init_MSE1();
}

void Iniciar_MSE4(Mse1 S)
{
     Comenzar_MSE1(S);
     comenzar = Falso;
}

void Avanzar_MSE4(Mse1 S)
{
     if (!comenzar) comenzar = Verdadero;
     else Avanzar_MSE1(S);
}

int EA_MSE4(Mse1 S)
{
```

```
    return EA_MSE1(S);
}

void Cargar_Fichero_MSE4(Mse1 S, char *nombre)
{
    Cargar_Fichero_MSE1(S, nombre);
}

Booleano EsVacia_MSE4(Mse1 S)
{
    return EA_MSE1(S) == MSE1_MarcaFin();
}

int main()
{
    int a, b;
    int numA, numB;
    int numPares;
    Mse1 S;


    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.4. Calcular el numero de pares *****\n");
    printf("*****              (cuarto modelo, segundo esquema) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un par de enteros:\n");
    printf("  primer entero: ");
    scanf("%d", &a);
    while (getchar() != '\n');
    printf("  segundo entero: ");
    scanf("%d", &b);
    while (getchar() != '\n');
    S = Init_MSE4();
    Cargar_Fichero_MSE4(S,"datos5_4.txt");
    Iniciar_MSE4(S);
    if (EsVacia_MSE4(S)) printf("Secuencia vacia");
    else {
         printf("\n");
         printf("La secuencia de entrada es: ");
         numPares = 0;
         numA = 0;
         numB = 0;
         Avanzar_MSE4(S);
         do {
```

```
                if (EA_MSE4(S) == a) {
                                numPares = numPares + numB;
                                numA = numA + 1;
                }
                else if (EA_MSE4(S) == b) {
                                numPares = numPares + numA;
                                numB = numB + 1;
                }
                printf("%d ", EA_MSE4(S));
                Avanzar_MSE4 (S);
        }
        while (EA_MSE4(S) != MSE4_MarcaFin());
        printf("\n");
        printf("\n");
        printf("El numero de pares (%d, %d) (%d, %d) es: %d", a, b, b, a, numPares);
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msr1.h"

int main()
{
    Msr1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
```

```
    printf("***** Algoritmo 5.5. Comprobar si hay una nota mayor o igual que 9 *****\n")
        ;
    printf("*****              (primer modelo)                          *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSR1();
    Cargar_Fichero_MSR1(S,"datos5_5.txt");
    printf("La secuencia de entrada es: ");
    Comenzar_MSR1(S);
    while (EA_MSR1(S) != MSR1_MarcaFin()) {
          printf("%.2f ", EA_MSR1(S));
          Avanzar_MSR1(S);
    }
    printf("\n");
    Comenzar_MSR1(S);
    while ((EA_MSR1(S) != MSR1_MarcaFin()) && (!(EA_MSR1(S) >= 9)))
          Avanzar_MSR1(S);
    // INV "No existe en Piz una nota superior o igual a 9"
    // fin while
    printf("\n");
    if (EA_MSR1(S) == MSR1_MarcaFin())
        printf("No existe ninguna nota superior o igual a 9");
    else
        printf("Al menos existe una nota superior o igual a 9");
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msr2.h"
```

```
int main()
{
    Msr2 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.6. Comprobar si hay una nota mayor o igual que 9 *****\n")
        ;
    printf("*****              (segundo modelo)                          *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSR2();
    Cargar_Fichero_MSR2(S,"datos5_6.txt");
    printf("La secuencia de entrada es: ");
    Iniciar_MSR2(S);
      Avanzar_MSR2(S);
      printf("%.2f ", EA_MSR2(S));

    do {
      // INV "No existe en Piz una nota superior o igual a 9"
      Avanzar_MSR2(S);
      printf("%.2f ", EA_MSR2(S));
    } while (!(EsUltimo_MSR2(S)));
    Iniciar_MSR2(S);
    if (EsVacia_MSR2(S)) printf("Nadie se ha presentado al examen");
    else {
          do
          // INV "No existe en Piz una nota superior o igual a 9"
                Avanzar_MSR2(S);
          while (!(EsUltimo_MSR2(S) || (EA_MSR2(S) >= 9)));
          printf("\n");
          printf("\n");
          if (EA_MSR2(S) >= 9)
              printf("Al menos existe una nota superior o igual a 9");
          else
              printf("No existe ninguna nota superior o igual a 9");
    };
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"
#define ESPACIO ' '

int main()
{
    int posicion;
    char anterior;
    Msc1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.7. Calculo posicion de la primera palabra que comienza
        *****\n");
    printf("*****              por 'a' (primer modelo, primer esquema)        *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S,"datos5_7.txt");
    printf("La secuencia de entrada es: ");
    Comenzar_MSC1 (S);
    while (EA_MSC1 (S) != MSC1_MarcaFin()) {
           printf("%c", EA_MSC1 (S));
           Avanzar_MSC1(S);
     };
    Comenzar_MSC1 (S);
    posicion = 0;
    anterior = ESPACIO;
    while ((EA_MSC1 (S) != MSC1_MarcaFin()) &&
           !((EA_MSC1 (S) == 'a') && (anterior == ESPACIO))) {
               posicion = posicion + 1;
               anterior = EA_MSC1 (S);
               Avanzar_MSC1 (S);
     // INV "En Piz ninguna palabra empieza por 'a'" y posicion = Long (Piz)
```

```
    };

   printf("\n");
   printf("\n");
   if (EA_MSC1 (S) == MSC1_MarcaFin())
      printf("El texto no contiene ninguna palabra que comience por la letra a");
   else
      printf("Posicion de la primera palabra que comienza por a es: %d", posicion+1);
   printf("\n");
   printf("\nPulse enter para continuar");
   getchar();
   return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"

#define ESPACIO ' '
#define Booleano int
#define Verdadero 1
#define Falso 0

int main()
{
   int posicion;
   char anterior;
   Msc1 S;

   clrscr();
   printf("\n");
   printf("*****\n");
```

```
    printf("***** Algoritmo 5.8. Calculo posicion de la primera palabra que comienza
        *****\n");
    printf("*****              por 'a' (primer modelo, tercer esquema)        *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S,"datos5_8.txt");
    printf("La secuencia de entrada es: ");
    Comenzar_MSC1 (S);
    while ((EA_MSC1 (S) != MSC1_MarcaFin())) {
           printf("%c", EA_MSC1 (S));
           Avanzar_MSC1(S);
    };
    printf("\n");
    Comenzar_MSC1 (S);
    if (EA_MSC1(S) == MSC1_MarcaFin()) printf("\nLa secuencia esta vacia");
    else if (EA_MSC1 (S) == 'a')
           printf("Posicion de la primera palabra que comienza por a es: %d", 1);
    else {
         posicion = 1;
         anterior = EA_MSC1 (S);
         // Se utiliza la equivalencia ITERAR-MIENTRAS
         // INV "En Piz ninguna palabra empieza por 'a'" y posicion = Long (Piz)
         do {
             Avanzar_MSC1 (S);
             if ((EA_MSC1(S) == MSC1_MarcaFin()) ||
                ((EA_MSC1(S) == 'a') &&
                (anterior == ESPACIO))) break;
             posicion = posicion + 1;
             anterior = EA_MSC1(S);
             }
         while (Verdadero);
         printf("\n");
         if (EA_MSC1(S) == MSC1_MarcaFin())
             printf("El texto no contiene ninguna palabra que comience por la letra a");
         else
             printf("Posicion de la primera palabra que comienza por a es: %d",
                 posicion+1);
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"
#define ESPACIO ' '

int main()
{
    char ultLetraPrimPal, anterior;
    Msc1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.9. Comprobar si todas las palabras acaban con la misma
        *****\n");
    printf("*****              letra (primer modelo, sin secuencia intermedia) *****\n");
    printf("*****\n");
    printf("\n");
/* Esquema de recorrido y busqueda.
   Encontrar letra final de la primera palabra */
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S,"datos5_9.txt");
    printf("La secuencia de entrada es: ");
    Comenzar_MSC1(S);
    while (EA_MSC1(S) != MSC1_MarcaFin()) {
          printf("%c", EA_MSC1(S));
          Avanzar_MSC1(S);
    };
    printf("\n");
    Comenzar_MSC1(S);
    anterior = ESPACIO;
    while ((EA_MSC1(S) != MSC1_MarcaFin()) &&
          !((EA_MSC1(S) == ESPACIO) && (anterior != ESPACIO))) {
            anterior = EA_MSC1(S);
            Avanzar_MSC1(S);
```

```
    /* INV anterior = "ultimo elemento de Piz" y
          "no hay ningun par (letra, espacio) en Piz" */
    };
    if ((EA_MSC1(S) == MSC1_MarcaFin()) && (anterior == ESPACIO))
            printf("No hay palabras en la secuencia");
    else if ((EA_MSC1(S) == MSC1_MarcaFin()))
          /* Solo hay una palabra y anterior <> ESPACIO */
         printf("Todas las palabras terminan en la misma letra");
    else {
       /* 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()) &&
                 !((EA_MSC1(S) == ESPACIO) && (anterior != ESPACIO) &&
                 (anterior != ultLetraPrimPal))) {
                        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" */
          };
          printf("\n");
          if ((EA_MSC1(S) == MSC1_MarcaFin()) && ((anterior == ultLetraPrimPal)
              || (anterior == ESPACIO)))
              printf("Todas las palabras terminan en la misma letra");
          else
              printf("Todas las palabras no terminan en la misma letra");
     };
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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: datos5_10.txt
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"
#define ESPACIO ' '
#define Booleano int
#define Verdadero 1
#define Falso 0

char ultLetra;
Booleano FinDeSecuenciaUltLetra;
char ultLetraPrimPal;
Msc1 S;

void IgnorarEspacios()
// POST EA es la siguiente letra, si la hay
{
    while ((EA_MSC1(S) != MSC1_MarcaFin()) && (EA_MSC1(S) == ESPACIO))
          Avanzar_MSC1(S);
}

void ObtenerUltimo()
/* POST ultLetra es la £ltima letra de la siguiente palabra,
       si la hay, y se actualiza FinDeSecuenciaUltLetra */
{
    IgnorarEspacios();
    if ((EA_MSC1(S) == MSC1_MarcaFin()))
        FinDeSecuenciaUltLetra = Verdadero;
    else {
         FinDeSecuenciaUltLetra = Falso;
         do {
             ultLetra = EA_MSC1(S);
             Avanzar_MSC1(S);
         } while (!((EA_MSC1(S) == MSC1_MarcaFin()) || (EA_MSC1(S) == ESPACIO)));
    };
}

void ComenzarUltLetra()
/* POST ultLetra es la ultima letra de la primera palabra,
       si la hay, y se actualiza FinDeSecuenciaUltLetra */
{
    Comenzar_MSC1(S);
    ObtenerUltimo();
```

```
}

void AvanzarUltLetra()
/* POST ultLetra es la última letra de la siguiente palabra,
       si la hay, y se actualiza FinDeSecuenciaUltLetra */
{
    ObtenerUltimo();
}

int main ()
{

   clrscr();
   printf("\n");
   printf("*****\n");
   printf("***** Algoritmo 5.10. Comprobar si todas las palabras acaban con la *****\n"
       );
   printf("*****              misma letra (primer modelo, con secuencia *****\n");
   printf("*****              intermedia formada por las ultimas letras de *****\n");
   printf("*****              cada palabra)                              *****\n");
   printf("*****\n");
   printf("\n");
   S = Init_MSC1();
   Cargar_Fichero_MSC1(S,"dat5_10.txt");
   printf("La secuencia de entrada es: ");
   Comenzar_MSC1(S);
   while (EA_MSC1(S) != MSC1_MarcaFin()) {
        printf("%c", EA_MSC1(S));
        Avanzar_MSC1(S);
   };
   printf("\n");
   ComenzarUltLetra();
   if (FinDeSecuenciaUltLetra) printf("No hay palabras");
   else {
        ultLetraPrimPal = ultLetra;
        do
          AvanzarUltLetra();
      // INV Todas las letras de Piz son iguales a ultLetraPrimPal
        while (!(FinDeSecuenciaUltLetra || (ultLetra != ultLetraPrimPal)));
        printf("\n");
        if (FinDeSecuenciaUltLetra)
            printf("Todas las palabras terminan en la misma letra");
        else
            printf("Todas las palabras no terminan en la misma letra");
   }
   printf("\n");
   printf("\nPulse enter para continuar");
```

```
    getchar();
    return 0;
}
```

```
/*
 * 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'                                  */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"
#define ESPACIO ' '
#define Booleano int
#define Verdadero 1
#define Falso 0

typedef struct {
               int longi;
               char primeraLetra;
} TKAbecedario;

TKAbecedario kABC;
Booleano EsUltimokABC, EsVaciakABC;
char anterior;
Msc1 S;

void 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 */
{
    do {
       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 */
    } while (!((EA_MSC1(S) == MSC1_MarcaFin()) || (EA_MSC1(S) == anterior+1)));
    EsUltimokABC = EA_MSC1(S) == MSC1_MarcaFin();
}

void 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 */
{
    Comenzar_MSC1(S);
    if (EA_MSC1(S) == MSC1_MarcaFin()) {
                                    EsUltimokABC = Verdadero;
                                    EsVaciakABC = Verdadero;
    }
    else {
         EncontrarInikABC();
         EsVaciakABC = EA_MSC1(S) == MSC1_MarcaFin();
    }
}

void 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 */
{
    kABC.longi = 1;
    kABC.primeraLetra = anterior;
    do {
        kABC.longi = kABC.longi + 1;
        anterior = EA_MSC1(S);
        Avanzar_MSC1(S);
    // INV kABC.long mantiene la longitud del k-abecedario actual
    } while (!((EA_MSC1(S) == MSC1_MarcaFin()) || (EA_MSC1(S) != anterior+1)));
    if ((EA_MSC1(S) == MSC1_MarcaFin())) EsUltimokABC = Verdadero;
    else EncontrarInikABC();
}

int main ()
{
    TKAbecedario mayorkABC;
    int i;
```

```
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.11. Obtener el k-abecedario de mayor longitud *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S,"dat5_11.txt");
    printf("La secuencia de entrada es: ");
    Comenzar_MSC1(S);
    while (EA_MSC1(S) != MSC1_MarcaFin()) {
          printf("%c", EA_MSC1(S));
          Avanzar_MSC1(S);
    };
    printf("\n");
    IniciarkABC();
    if (EsVaciakABC) printf("No hay k-abecedarios");
    else {
          AvanzarkABC();
          mayorkABC = kABC;
          while (!EsUltimokABC) {
              AvanzarkABC();
              if (mayorkABC.longi < kABC.longi) mayorkABC = kABC;
          /* INV mayorkABC.long mantiene la longitud del maximo
                  k-abecedario de Piz */
          };
          printf("\n");
          printf("El k-abecedario de mayor longitud es: \n");
          printf("\n");
          printf("  longitud: %d\n", mayorkABC.longi);
          printf("\n");
          printf("  k-abecedario: ");
          for (i = 1; i <= mayorkABC.longi; i++)
               printf("%c", mayorkABC.primeraLetra+i-1);
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "mse1.h"

int main ()
{
    int numSegTotal;
        // Numero de segundos transcurridos durante el control
    int numVehEnt;
        // Numero de vehiculos que han entrado en la autopista
    int numSegInterMasLargo;
        // Intervalo de tiempo mas largo sin entrar vehiculos
    int numSegInterActual;
        // Contador del numero de segundos del intervalo actual
    Mse1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 5.12. Control de trafico en una autopista *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSE1();
    Cargar_Fichero_MSE1(S,"dat5_12.txt");
    printf("La secuencia de entrada es: ");
    Comenzar_MSE1(S);
    numSegTotal = 0;
    numVehEnt = 0;
    numSegInterMasLargo = 0;
    while (EA_MSE1(S) != 0) {
       printf("%d ", EA_MSE1(S));
       switch (EA_MSE1(S)) {
            case 2: numVehEnt = numVehEnt + 1;
                    Avanzar_MSE1(S);
                    break;
            case 1: numSegInterActual = 1;
                    Avanzar_MSE1(S);
                    while (!((EA_MSE1(S) == 0) ||
                             (EA_MSE1(S) == 2))) {
                          printf("%d ", EA_MSE1(S));
```

```
                        if (EA_MSE1(S) == 1)
                            numSegInterActual = numSegInterActual + 1;
                        Avanzar_MSE1(S);
                    }
                    if (numSegInterActual > numSegInterMasLargo)
                        numSegInterMasLargo = numSegInterActual;
                    numSegTotal = numSegTotal + numSegInterActual;
                    break;
            default: Avanzar_MSE1(S);
        }
    }
    printf("\n\n");
    printf("El tiempo total transcurrido es: %d\n", numSegTotal);
    printf("\n");
    printf("El numero de vehiculos que han circulado por la autopista es: %d\n",
           numVehEnt);
    printf("\n");
    printf("El intervalo de tiempo mas largo sin entrar un vehiculo es: %d\n",
           numSegInterMasLargo);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

3.5. Capítulo 6

```
/*
 * 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.c
 * Para consultar el fichero datos6_1.dat emplear el programa verf6_1.c
 * en este algoritmo se ha sustituido la secuencia de NotaAlumno
 * por un fichero secuencial, 'datos6_1.dat' */

#include <stdio.h>
#include <conio.h>

#define MAX_ALUMNOS 200
typedef struct {
               int codigo; /* 1..MAX_ALUMNOS; */
       float nota;
} NotaAlumno;

int main ()
{
    FILE *f;
    NotaAlumno alum;
    int cod; /* 1..MAX_ALUMNOS; */

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.1. Mostrar la nota de un alumno (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    f = fopen("datos6_1.dat", "rb");
    printf("Introduzca el codigo del alumno (>= 1) (<= %d ): ",MAX_ALUMNOS);
    scanf("%d", &cod);
    while (getchar() != '\n');
    fread (&alum, sizeof(NotaAlumno), 1, f);
    while (!(feof(f)) && (alum.codigo != cod))
```

```
            fread (&alum, sizeof(NotaAlumno), 1, f);
            /* Si se ha llegado al final, feof sera verdadero,
             * por ello debemos consultar la funcion feof
             * para determinar la ausencia del codigo */
    printf("\n");
    if (feof(f)) printf("No existe ese codigo\n");
    else printf("La nota de este alumno es: %.2f\n", alum.nota);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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 */
/* En C las tablas comienzan en cero y por esa razon el codigo del alumno */
/* comienza en cero en lugar de en uno, como en el Algoritmo 6.1 */

#define MAX_ALUMNOS 100
#include <stdio.h>
#include <conio.h>

float notas[MAX_ALUMNOS] =
            {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, 3.4, 2.2, 1.2, 8.2, 5.4, 3.2,
             5.3, 6.2, 2.2, 1.2, 8.2, 3.5, 5.2, 1.2, 2.7, 3.4,
             2.2, 1.2, 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, 6.2, 2.2, 1.2, 8.2, 3.5, 5.2, 1.2, 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, 1.2, 2.7, 3.4, 2.2, 1.2, 8.2, 5.4, 3.2,
             9.0, 6.2, 2.2, 1.2, 8.2, 3.5, 5.2, 1.2, 2.7, 3.4,
             2.4, 1.2, 8.2, 5.4, 3.2, 5.3, 6.2, 2.2, 1.2, 8.2};


int main()
{
```

```
    int cod; /* 1..MAX_ALUMNOS; */

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.2. Mostrar la nota de un alumno (version 2) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el codigo del alumno (>= 0) y (<=%d): ", MAX_ALUMNOS-1);
    scanf("%d", &cod);
    while (getchar() != '\n');
    printf("\n");
    if (notas[cod] != -1.0)
        printf("La nota del alumno %d es: %.2f\n", cod, notas[cod]);
    else printf("No existe una calificacion para ese alumno");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"

#define ESPACIO ' '
#define Booleano int
#define Verdadero 1
#define Falso 0
#define TAM_HISTO 'Z' - 'A' + 1
#define elemHisto(h,c) h[(c) - 'A']

typedef int FrecuenciaLetras[TAM_HISTO];
```

```
void InicializarTablaFrecuencias (int *t)
// Inicializacion de la tabla con ceros
// POST: t[c] = 0, c: 'A' <= c <= 'Z'
{
    char c; // 'A'..'Z';

    for (c = 'A'; c <= 'Z'; c++) {
        elemHisto(t, c) = 0;
    }
}

void EscribirTablaFrecuencias (int *t)
// POST: Se escriben los valores de la tabla t
{
    char c; // 'A'..'Z';

    printf("\n");
    for (c = 'A'; c <= 'Z'; c++) {
        printf(" */* %c : %d", c, elemHisto(t, c));
        if (((c-'A'+1) % 6) == 0) printf("\n");
    }
    printf("\n");
}

Booleano EsMayuscula (char car)
// POST: Retorna Verdadero si y solo si car es una letra mayuscula
{
    return (car >= 'A') && (car <= 'Z');
}

int main()
{
    FrecuenciaLetras tf;
    Msc1 S;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.3. Calculo de la frecuencia de aparicion de las *****\n");
    printf("*****              letras mayusculas en un texto           *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S,"datos6_3.txt");
    Comenzar_MSC1(S);
    if (EA_MSC1(S) == MSC1_MarcaFin())
       printf("Secuencia vacia");
```

```
    else {
          InicializarTablaFrecuencias(tf);
          printf("La secuencia de entrada es: ");
              do {
                  printf("%c", EA_MSC1(S));
                  if (EsMayuscula (EA_MSC1(S)))
                 elemHisto(tf, EA_MSC1(S)) = elemHisto(tf, EA_MSC1(S)) + 1;
                  Avanzar_MSC1(S);
              /* INV: tf almacena la frecuencia de aparicion de cada
                  letra mayuscula en Piz */
          } while (!(EA_MSC1(S) == MSC1_MarcaFin()));
          printf("\n");
              EscribirTablaFrecuencias(tf);
    }
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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.c */
/* Para consultar todo el fichero datos6_4.dat emplear el programa verf6_4.c */

#include <stdio.h>
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0
#define NUM_NOTAS 5
#define MAX_ESTUDIANTES 100

typedef struct {
               char nombre[30];
                      int edad;
                      Booleano sexo;
```

```
                    float nota;
                  } Estudiante;

typedef FILE *Festudiantes;
typedef Estudiante Curso[MAX_ESTUDIANTES+1];

void Escribe(int i)
{
    switch (i) {
      case 0 : printf("Numero de suspensos: "); break;
      case 1 : printf("Numero de aprobados: "); break;
      case 2 : printf("Numero de notables: "); break;
      case 3 : printf("Numero de sobresalientes: "); break;
      case 4 : printf("Numero de matriculas de honor: "); break;
    }
}

int main()
{
    Festudiantes f;
    Curso c;
    int fNotas[NUM_NOTAS];
    int i, k; // 0..MAX_ESTUDIANTES
    int j;    // 0..NUM_NOTAS

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.4. Calcular distribucion de las notas *****\n");
    printf("*****\n");
    printf("\n");
    f = fopen("datos6_4.dat", "rb");
    k = 0;
    fread (&c[k], sizeof(Estudiante), 1, f);
    while ((!feof(f)) && (k <= MAX_ESTUDIANTES-1)) {
          k = k+1;
          fread(&c[k], sizeof(Estudiante), 1, f);
    }
    fclose(f);
    // Comenzar
    i = 0;

    // Tratamiento inicial
    for (j = 0; j <= NUM_NOTAS-1; j++)
        fNotas[j] = 0;

    // Tratamiento de cada elemento de la tabla
```

```
    while (i != k) {
          if ((c[i].nota) < 5) fNotas[0] = fNotas[0] + 1;
          else if ((c[i].nota >= 5) && (c[i].nota < 7)) fNotas[1] = fNotas[1] + 1;
          else if ((c[i].nota >= 7) && (c[i].nota < 9)) fNotas[2] = fNotas[2] + 1;
          else if (c[i].nota == 9) fNotas[3] = fNotas[3] + 1;
          else fNotas[4] = fNotas[4] + 1;
          // Avanzar
          i = i+1;
    }

    // Tratamiento final
    for (j = 0; j <= NUM_NOTAS-1; j++) {
         Escribe(j);
         printf("%d\n", fNotas[j]);
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * Algoritmo 6.5. Porcentaje de mujeres y hombres, y nota media de cada sexo
 * 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.c */
/* Para consultar todo el fichero datos6_4.dat emplear el programa verf6_4.c */

#include <stdio.h>
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0
#define MAX_ESTUDIANTES 100

typedef struct {
               char nombre[30];
                      int edad;
```

```
                    Booleano sexo; // verdadero si es mujer, falso si es hombre
                    float nota;
                   } Estudiante;

typedef FILE *Festudiantes;
typedef Estudiante Curso[MAX_ESTUDIANTES+1];
typedef int NumEstudiantes; // 0..MAX_ESTUDIANTES

void Leer(Curso c, NumEstudiantes *l)
{
    Festudiantes f;

    f = fopen("datos6_4.dat", "rb");
    *l = 0;
    fread(&c[*l], sizeof(Estudiante), 1, f);
    while ((!feof(f)) && (*l < MAX_ESTUDIANTES)) {
          *l = *l + 1;
          fread(&c[*l], sizeof(Estudiante), 1, f);
    }
    fclose(f);
}

int main()
{

    Curso c;
    int contaM, contaH; // 0..MAX_ESTUDIANTES contadores numero de mujeres y hombres
    float sumaM, sumaH;                    // contadores notas de mujeres y hombres
    float mujeres, hombres;                    // porcentajes de mujeres y hombres
    NumEstudiantes longi;                  // longitud de la secuencia almacenada
    int i; // 1..MAX_ESTUDIANTES           indice para la tabla

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.5. Porcentaje de mujeres y hombres, *****\n");
    printf("*****              y nota media de cada sexo     *****\n");
    printf("*****\n");
    printf("\n");
    // Se introducen los datos de long estudiantes en la tabla c
    Leer (c, &longi);
    switch (longi) {
          case 0 : printf("No hay alumnos\n"); break;
          default : sumaM = 0; sumaH = 0;
                    contaM = 0; contaH = 0;
                for (i = 0; i <= longi-1; i++)
                     if (c[i].sexo) {
```

```
                                        sumaM = sumaM + c[i].nota;
                                        contaM = contaM + 1;
                    }
                    else {
                        sumaH = sumaH + c[i].nota;
                                    contaH = contaH + 1;
                    };
                mujeres = contaM / (float) longi * 100;
                hombres = contaH / (float) longi * 100;
                printf("Porcentaje de mujeres: %.2f%%\n", mujeres);
                printf("\n");
                printf("Porcentaje de hombres: %.2f%%\n", hombres);
                if (contaM != 0) printf("\nNota media de las mujeres: %.2f\n", sumaM/
                    contaM);
                if (contaH != 0) printf("\nNota media de los hombres: %.2f\n", sumaH/
                    contaH);
    }
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef unsigned int Rango; // 1..LMAX
typedef float Vector[LMAX];

void Leer(float *a, Rango n)
{
    Rango i;

    for (i = 0; i <= n-1; i++) {
        printf("Introduzca componente %d: ", i+1);
        scanf("%f", &a[i]);
```

```
        while (getchar() != '\n');
    };
}

int main()
{
    Vector a, b;
    float producto;
    Rango longitud;
    Rango i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.6. Producto escalar de dos vectores *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduce la longitud de los vectores: ");
    scanf("%d", &longitud);
    while (getchar() != '\n');
    printf("\n");
    printf("Vector 1: \n");
    Leer(a, longitud);
    printf("\n");
    printf("Vector 2: \n");
    Leer(b, longitud);
    producto = 0;
    for (i = 0; i <= longitud-1; i++)
         producto = producto + a[i] * b[i];
         // INV producto = a[1]* b[1] + a[2]* b[2] + .. a[i] * b[i]
    printf("\n");
    printf("El producto escalar es: %.2f\n", producto);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef unsigned int Rango; // 1..LMAX
typedef int Vector[LMAX];

void Leer(int *a, Rango n)
{
    Rango i;

    for (i = 0; i <= n-1; i++) {
        printf(" Introduzca componente %d: ", i+1);
        scanf("%d", &a[i]);
        while (getchar() != '\n');
    };
}

int main()
{
    Rango longitud;
    Vector t;
    int mayor;
    Rango numApa;
    Rango pri, ult;
    Rango i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.7. Obtener el mayor, su numero de ocurrencias y *****\n");
    printf("*****              su primera y ultima posicion            *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos: ");
    scanf("%d", &longitud);
    while (getchar() != '\n');
    Leer(t, longitud);
    mayor = t[0]; numApa = 1; pri = 0; ult = 0;
    for (i = 1; i <= longitud-1; i++)
        // Se cumple INV definido en la descripcion del algoritmo
        if (t[i] > mayor) {
                        mayor = t[i];
                        numApa = 1;
                        pri = i;
```

```
                        ult = i;
        }
        else if (t[i] == mayor) {
                              numApa = numApa + 1;
                              ult = i;
        }
    printf("\n");
    printf("El mayor es: %d\n", mayor);
    printf("\n");
    printf("El numero de apariciones es: %d\n", numApa);
    printf("\n");
    printf("Las posiciones de la primera y la ultima aparicion son: %d y %d\n", pri+1,
         ult+1);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

// Calcular la fecha a partir de la posicion del dia en el anyo (version 1)

#include <stdio.h>
#include <conio.h>
typedef int MesAnyo; // 1..12
typedef int DiaAnyo; // 1..365
typedef int Diames; // 1..31

int main()
{
    int tp[12] = { 1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 };
    int p;
    int i;
    int mes;
    int dia;
```

```
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.8. Calcular la fecha a partir de la posicion *****\n");
    printf("*****              del dia en el anyo (version 1)        *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca la posicion del dia en el anyo: ");
    scanf("%d", &p);
    while (getchar() != '\n');
    i = 1;
    while ((i != 11) && (tp[i] <= p)) {
          // INV (j: 1 <= j < i : p >= tp[j])
          i = i + 1;
    }
    if (tp[i] > p) mes = i;
    else mes = 12;
    dia = p - tp[mes-1] + 1;
    printf("\n");
    printf("Dia: %d del mes: %d\n", dia, mes);

    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

// Calcular la fecha a partir de la posicion del dia en el anyo (version 2)

#include <stdio.h>
#include <conio.h>
typedef int MesAyo; // 1..12
typedef int DiaAyo; // 1..365
typedef int Diames; // 1..31
```

```c
int main()
{
    int tp[12] = {1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335};
    int p;
    int i;
    int mes;
    int dia;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.9. Calcular la fecha a partir de la posicion *****\n");
    printf("*****              del dia en el anyo (version 2)        *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca la posicion del dia en el anyo: ");
    scanf("%d", &p);
    while (getchar() != '\n');
    if (p >= tp[11]) mes = 12;
    else {
          i = 1;
          while (tp[i] <= p)
          // INV j: 1 <= j < i : p >= tp[j]
                i = i + 1;
          mes = i;
    };
    dia = p - tp[mes-1] + 1;
    printf("\n");
    printf("Dia: %d del mes %d\n", dia, mes);

    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */
```

```c
#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef int Rango;
typedef int Vector[LMAX];

void Leer(int *a, Rango n)
{
   Rango i;

   for (i = 0; i <= n-1; i++) {
        printf(" Introduzca componente %d: ", i+1);
        scanf("%d", &a[i]);
        while (getchar() != '\n');
   }
}

int main()
{
    Vector t;
    int pesoMaximo;     // peso mayor de los segmentos de la secuencia
    int pesoMaximoSegAct; // peso mayor de los segmentos actuales
    Rango i, longitud;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.10. Calculo del peso del segmento de mayor peso *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos (maximo %d): ", LMAX);
    scanf("%d", &longitud);
    while (getchar() != '\n');
    Leer(t, longitud);
    if (longitud == 0) printf("Secuencia vacia\n");
    else {
         pesoMaximo = t[0];
         pesoMaximoSegAct = t[0];
         for (i = 1; i <= longitud-1; i++) {
              if (t[i] > t[i] + pesoMaximoSegAct) pesoMaximoSegAct = t[i];
              else pesoMaximoSegAct = pesoMaximoSegAct + t[i];
              if (pesoMaximo < pesoMaximoSegAct) pesoMaximo = pesoMaximoSegAct;
         }
         printf("\n");
         printf("El peso del segmento de mayor peso es: %d\n", pesoMaximo);
    }
    printf("\nPulse enter para continuar");
```

```
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef int Rango; // 1..LMAX
typedef int Vector[LMAX];

void Leer(int *a, Rango n)
{
    Rango i;

    for (i = 0; i <= n-1; i++) {
         printf(" Introduzca el elemento %d: ",i+1);
         scanf("%d", &a[i]);
         while (getchar() != '\n');
    }
}

int main()
{

    Vector t;
    Rango longitud;
    Rango i;
    int noCerosSLC;
    int pesoNoCeros;
    int ceros;

    clrscr();
    printf("\n");
    printf("*****\n");
```

```
    printf("***** Algoritmo 6.11. Suma de enteros distintos de cero en *****\n");
    printf("*****                segmentos con cero en los extremos *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca la longitud de la secuencia (maximo %d): ", LMAX);
    scanf("%d", &longitud);
    while (getchar() != '\n');
    Leer(t, longitud);
    noCerosSLC = 0;
    pesoNoCeros = 0;
    ceros = 0;
    for (i = 0; i <= longitud-1; i++)
         if (t[i] == 0) {
                        noCerosSLC = noCerosSLC + pesoNoCeros;
                        ceros = ceros + 1;
         }
         else pesoNoCeros = pesoNoCeros + ceros;
    printf("\n");
    printf("La suma de los elementos distintos de cero en \n");
    printf("segmentos con cero en los extremos es: %d\n", noCerosSLC);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef int Rango; // 1..LMAX
typedef int Vector[LMAX];

void Leer(int *a, Rango n)
{
   Rango i;
```

```
   for (i = 0; i <= n-1; i++) {
        printf(" Introduzca elemento %d: ",i+1);
        scanf("%d", &a[i]);
        while (getchar() != '\n');
    }
}

int main()
{
    Vector t;
    int numPares;
    Rango i, j, longitud;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.12. Calcular numero de pares cero-uno *****\n");
    printf("*****              (version fuerza bruta)         *****\n");
    printf("*****\n");
    printf("\n");

    printf("Introduzca la longitud de la secuencia (maximo %d): ", LMAX);
    scanf("%d", &longitud);
    while (getchar() != '\n');
    Leer(t, longitud);

    numPares = 0;
    for (i = 0; i <= longitud-1; i++)
        if (t[i] == 0)
           for (j = i + 1; j <= longitud; j++)
               if (t[j] == 1) numPares = numPares + 1;
    printf("\n");
    printf("El numero de pares 0-1 es: %d\n", numPares);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef int Rango; // 0..LMAX
typedef int Vector[LMAX+1];

void Leer(int *a, Rango n)
{
    Rango i;

    for (i = 1; i <= n; i++) {
         printf(" Introduzca elemento %d: ", i);
         scanf("%d", &a[i]);
         while (getchar() != '\n');
    }
}

void Escribir (int *a, Rango n)
{
    Rango i;

    for (i = 1; i <= n; i++)
         printf("%d ", a[i]);
    printf("\n");
}

int main()
{
    Vector t;
    Rango q, j;
    Rango n;
    Rango x;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.13. Metodo de ordenacion de insercion directa *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos (maximo %d): ", LMAX);
    scanf("%d", &n);
    while (getchar() != '\n');
    Leer(t, n);
    printf("\n");
```

```
    printf("Los elementos iniciales son: ");
    Escribir(t, n);
    for (q = 2; q <= n; q++) {
         t[0] = t[q];                // centinela
         x = t[q];
         j = q - 1;
             while (x < t[j]) {
                   t[j+1] = t[j];    // desplazamiento
                   j = j - 1;
         }
             t[j+1] = x;
    };
    printf("\n");
    printf("Los elementos ordenados son: ");
    Escribir(t, n);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100
typedef int Rango; // 1..LMAX
typedef int Vector[LMAX];

void Leer(int *a, Rango n)
{
    Rango i;

    for (i = 0; i <= n-1; i++) {
         printf(" Introduzca elemento %d: ", i+1);
         scanf("%d", &a[i]);
         while (getchar() != '\n');
    }
```

```
}

void Escribir (int *a, Rango n)
{
    Rango i;

    for (i = 0; i <= n-1; i++)
        printf("%d ", a[i]);
    printf("\n");
}

int main()
{
    Vector t;
    Rango pos, q, j;
    Rango n;
    int min;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.14. Metodo de ordenacion de seleccion directa *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos (maximo %d): ", LMAX);
    scanf("%d", &n);
    while (getchar() != '\n');
    Leer(t, n);
    printf("\n");
    printf("Los elementos iniciales son: ");
    Escribir(t, n);
    for (q = 0; q <= n - 2; q++) {
        pos = q;
        min = t[q];
        for (j = q; j <= n-1; j++)
            if (min > t[j]) {
                        pos = j;
                                    min = t[j];
            }
            t[pos] = t[q];
            t[q] = min;
    }
    printf("\n");
    printf("Los elementos ordenados son: ");
    Escribir(t, n);
    printf("\nPulse enter para continuar");
    getchar();
```

```
    return 0;
}
```

```
/*
 * Algoritmo 6.15. Busqueda binaria en un tabla ordenada
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define LMAX 100

typedef int Rango; // 1..LMAX
typedef int Vector[LMAX];


void Leer(int *a, Rango n)
{
    Rango i;

    for (i = 0; i <= n-1; i++) {
         printf(" Introduzca elemento %d: ", i+1);
         scanf("%d", &a[i]);
         while (getchar() != '\n');
    }
}

int main()
{
    Rango n;
    Vector t;
    int i; // 0..LMAX
    int j; // 1..LMAX+1
    Rango h;
    int x;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.15. Busqueda binaria en un tabla ordenada *****\n");
```

```
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos (maximo %d): ", LMAX);
    scanf("%d", &n);
    while (getchar() != '\n');
    Leer(t, n);
    printf("\n");
    printf("Introduzca el elemento a buscar: ");
    scanf("%d", &x);
    while (getchar() != '\n');
    printf("\n");
    printf("La secuencia de entrada es: ");
    for (i = 0; i <= n-1; i++)
        printf("%d ", t[i]);
    printf("\n");
    i = -1;
    j = n;
    while ((i + 1) != j) {
           h = (i + j) / 2;
           if (t[h] <= x) i = h;
           else j = h;
    }
    printf("\n");
    if ((i > -1) && (x == t[i]))
        printf("Valor encontrado en posicion: %d\n", i+1 );
    else printf("Valor no encontrado\n");

    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define NFilas 20   // constante con el numero de filas
```

```
#define NColumnas 20 // constante con el numero de columnas
#define n 20

typedef int RangoFilas; // 1..NFilas
typedef int RangoColumnas; // 1..NColumnas
typedef int Vector[NFilas][NColumnas];

int nf, i;
int nc, j;

void Leer(Vector t, int nf, int nc)
{
    int i;
    int j;

    for (i = 0; i <= nf-1; i++) {
        printf("\n");
        printf("Fila: %d\n", i+1);
        for (j = 0; j <= nc-1; j++) {
            printf(" Introduzca elemento (%d, %d): ",i+1, j+1);
            scanf("%d", &t[i][j]);
            while (getchar() != '\n');
        }
    }
}

int main()
{
    Vector t;
    int nf, i;
    int nc, j;
    int sm, mayor;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.16. Suma de los valores mayores de cada fila de *****\n");
    printf("*****              una matriz de enteros                    *****\n");
    printf("*****\n");
    printf("\n");

    printf("Introduzca el numero de filas (<= %d): ", n);
    scanf("%d", &nf);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca el numero de columnas (<= %d): ", n);
    scanf("%d", &nc);
```

```
    while (getchar() != '\n');
    Leer(t, nf, nc);
    sm = 0;
    for (i = 0; i <= nf-1; i++) {
         mayor = t[i][0];
         for (j = 0; j <= nc-1; j++)
              if (mayor < t[i][j]) mayor = t[i][j];
             /* INVint mayor = mayor de los j elementos ya recorridos
                       de la fila i, 1 <= j <= nc */
         sm = sm + mayor;
         /* INVext sm = suma de los mayores de las primeras
                        i filas de T, 1 <= i <= nf */
    }
    // INVext y (i = nf) => POST
    printf("\n");
    printf("La suma de los valores mayores de cada fila de \n");
    printf("la matriz de enteros es: %d\n", sm);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
#include "msc1.h"
#define Booleano int
#define Verdadero 1
#define Falso 0
#define N 5
#define ESPACIO ' '

typedef char Palabra[30];
typedef char MatrizCar[N][N];
typedef struct {
```

```
                char p;
                char s;
                } ParLetras;
typedef struct {
                int f, c;
               } Posicion;

Palabra clave;          // palabra clave
MatrizCar mc;           // matriz de codificacion
Msc1 S;                         // texto original
// lexico de una secuencia intermedia de pares de caracteres
ParLetras parActual;
Booleano ultimoPar;

Booleano FinPares()
{
    return (EA_MSC1(S) == MSC1_MarcaFin()) && (!ultimoPar);
}

void IgnorarBlancos()
// Salta blancos del texto original
{
    while (EA_MSC1(S) == ESPACIO)
          Avanzar_MSC1(S);
}

void sustituir(char *l)
{
    switch (*l) {
        case 'j': *l = 'i'; break;
        case 'ñ': *l = 'n'; break;
    }
}

void SigPar()
{
    IgnorarBlancos();
    if (EA_MSC1(S) != MSC1_MarcaFin()) {
        parActual.p = EA_MSC1(S);
        sustituir(&parActual.p);
        Avanzar_MSC1(S);
        IgnorarBlancos();
        if (EA_MSC1(S) == MSC1_MarcaFin()) {
                                    ultimoPar = Verdadero;
                                    parActual.s = 'x';
        }
        else {
```

```
            parActual.s = EA_MSC1(S);
            sustituir(&parActual.s);
            if (parActual.s == parActual.p) parActual.s = 'x';
        };
    };
}

void ComenzarPar()
{
    Comenzar_MSC1(S);
    ultimoPar = Falso;
    SigPar();
}

void AvanzarPar()
{
    if (ultimoPar) ultimoPar = Falso;
    else {
         Avanzar_MSC1(S);
         SigPar();
    }
}

Booleano CarEnPalabra(Palabra p, int longi, char c)
// PRE: longi > 0
/* POST retorna Verdadero si el caracter c se encuentra en la tabla p,
        Falso en otro caso */
{
    int i; // 1..N+1 En el rango [1, long]

    i = 0;
    while ((i != longi) && !(p[i] == c))
          i = i + 1;
    return (i != longi);
}

void CrearMatrizCodificacion()
{
    char letra; // letra actual
    int i, j;  // 1..N

    // rellenar la primera fila con la clave
    for (i = 0; i <= N-1; i++)
        mc[0][i] = clave[i];
    // rellenar resto de filas
    // inicializar recorrido intervalo de letras
    letra = 'a';
```

```
    for (i = 1; i <= N-1; i++)
        for (j = 0; j <= N-1; j++) {
            while (CarEnPalabra(clave, N, letra) ||
                   (letra == 'j') || (letra == 'ñ'))
                       letra = letra+1;
            mc[i][j] = letra;
            letra = letra+1;
        }
}

void PosEnTabla(MatrizCar m, char car, Posicion *pos)
// PRE: car es una letra y se encuentra en la tabla
{
    int i, j; // 1..N

    i = 0;
    j = 0;
    while (m[i][j] != car)
         if (j < N-1) j = j + 1;
            else {
              i = i + 1;
              j = 0;
         }
    pos->f = i;
    pos->c = j;
}

void Codificacion(ParLetras par, ParLetras *nuevoPar)
{
    Posicion p1, p2;
    int fila, col; // 1..N

    PosEnTabla(mc, par.p, &p1);
    PosEnTabla(mc, par.s, &p2);
    if (p1.f == p2.f) {
                    p1.c = (p1.c + 1) % N;
                              p2.c = (p2.c + 1) % N;

                    nuevoPar->p = mc[p1.f][p1.c];
                    nuevoPar->s = mc[p2.f][p2.c];
    }
    else if (p1.c == p2.c) {
                         p1.f = (p1.f + 1) % N;
                         p2.f = (p2.f + 1) % N;
                         nuevoPar->p = mc[p1.f][p1.c];
                         nuevoPar->s = mc[p2.f][p2.c];
    }
```

```
    else {
         nuevoPar->p = mc[p1.f][p2.c];
         nuevoPar->s = mc[p2.f][p1.c];
    }
}

int main()
{
    ParLetras parCifrado; // par codificado
    Msc1 R;                          // texto cifrado

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.17. Cifrado PlayFair *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca la palabra clave de 5 letras: ");
    scanf("%s", clave);
    while (getchar() != '\n');
    printf("\n");
    printf("Texto original: ");
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S,"dat6_17.txt");
    Comenzar_MSC1(S);
    while (EA_MSC1(S) != MSC1_MarcaFin()) {
          printf("%c",EA_MSC1(S));
          Avanzar_MSC1(S) ;
    }
    printf("\n");
    CrearMatrizCodificacion();
    ComenzarPar();
    R = Init_MSC1();
    Arrancar_MSC1(R);
    while (!FinPares()) {
         Codificacion(parActual, &parCifrado);
            Registrar_MSC1 (R, parCifrado.p);
         Registrar_MSC1 (R, parCifrado.s);
         AvanzarPar();
    }
    Marcar_MSC1(R) ;
    printf("\n");
    printf("El texto cifrado es: ");
    Comenzar_MSC1(R);
    while (EA_MSC1(R) != MSC1_MarcaFin()) {
          printf("%c",EA_MSC1(R));
          Avanzar_MSC1(R) ;
```

```
    }
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#include "msc1.h"

#define Booleano int
#define Verdadero 1
#define Falso 0

#define ESPACIO ' '
#define SANGRADOMAXIMO 15
#define LONGMAXPAL 20
#define ANCHOMINLINEA LONGMAXPAL+SANGRADOMAXIMO
#define ANCHOMAXLINEA 80

typedef int LongitudPalabra; // 1..LONGMAXPAL
typedef struct {
               char contenido[LONGMAXPAL];
               LongitudPalabra longitud;
               } Palabra;
typedef int PosEnLinea;   // 1..ANCHOMAXLINEA;
typedef int LongitudLinea; // 0..ANCHOMAXLINEA;
typedef struct {
               LongitudLinea longitud;
               char contenido[ANCHOMAXLINEA];
              } Linea;

typedef int LongSangrado; // 0..SANGRADOMAXIMO;
```

```
typedef int LimAnchoLinea; // ANCHOMINLINEA..ANCHOMAXLINEA;

LongSangrado sangrado;
LimAnchoLinea anchoLinea;
Palabra palabraActual;
Linea lineaActual;
Booleano FinDeSecuenciaPalabra;
Msc1 S;
Palabra MarcaParrafo;

/* **************                                    ********************
 * Operaciones secuencia intermedia de palabras
 * **************                                    ********************/

void ObtenerPalabra()
{
    int i;
    /* Ignora los espacios en blanco */
    while ((EA_MSC1(S) != MSC1_MarcaFin()) &&
          ((EA_MSC1(S) == ESPACIO) || (EA_MSC1(S) == MSC1_MarcaFinLinea)))
             Avanzar_MSC1 (S);
    FinDeSecuenciaPalabra = (EA_MSC1(S) == MSC1_MarcaFin());
    if (!(EA_MSC1(S) == MSC1_MarcaFin())) {
    /* Almacena en el campo palabraActual.contenido la siguiente palabra */
          i = -1;
          do {
              i = i + 1;
                  palabraActual.contenido[i] = EA_MSC1 (S);
                  Avanzar_MSC1 (S);
          } while (!((EA_MSC1(S) == MSC1_MarcaFin()) ||
                    ((EA_MSC1(S) == ESPACIO) ||
                    ((EA_MSC1(S) == MSC1_MarcaFinLinea)))));
          palabraActual.longitud = i + 1;
     }
}

/* *********************                                  *************/
void ComenzarPalabra()
{
     S = Init_MSC1();
     Cargar_Fichero_MSC1(S,"dat6_18.txt");
     Comenzar_MSC1(S);
     ObtenerPalabra();
}

/* *********************                                  *************/
void AvanzarPalabra()
```

```
{
     ObtenerPalabra();
}

/* **************                                    ********************
 * Operaciones relacionadas con el tratamiento de palabras
 * **************                                    ********************/
LongitudPalabra Longitud (Palabra pal)
{
     return (pal.longitud);
}
/* *********************                                        *************/
Booleano Igual(Palabra p1, Palabra p2)
{
     LongitudPalabra longi, i;
     Booleano igualLong;

     igualLong = p1.longitud == p2.longitud;
     if (igualLong) {
                         longi = p1.longitud;
                i = 0;
                while ((i != longi-1) && (p1.contenido[i] == p2.contenido[i]))
                       i = i + 1;
                igualLong = (p1.contenido[i] == p2.contenido[i]);
     }
     return igualLong;
}

/* **************                                    ********************
 * Grupo de operaciones de tratamiento de lineas
 * **************                                    ********************/
void AnyadirCaracterALinea(char car)
{
     lineaActual.contenido[lineaActual.longitud] = car;
     lineaActual.longitud = lineaActual.longitud + 1;
}

/* *********************                                        *************/
void AnyadirPalabraALinea()
{
   LongitudPalabra i;

   for (i = 0; i<=palabraActual.longitud-1; i++)
         AnyadirCaracterALinea(palabraActual.contenido[i]);
};

/* *********************                                        *************/
```

```
void InicializarPropLinea (LongSangrado *longSangrado,
                           LimAnchoLinea *longLinea)
{
    int sangradoLinea, anchoLinea;

    printf("Introduzca el ancho de la linea (entre %d y %d): ", ANCHOMINLINEA,
        ANCHOMAXLINEA);
    scanf("%d", &anchoLinea);
    while (getchar() != '\n');
    if (anchoLinea < ANCHOMINLINEA) *longLinea = ANCHOMINLINEA;
    else if (anchoLinea > ANCHOMAXLINEA) *longLinea = ANCHOMAXLINEA;
    else *longLinea = anchoLinea;
    printf("\n");
    printf("Introduzca el sangrado de la primera linea (entre 0 y %d): ",
        SANGRADOMAXIMO);
    scanf("%d", &sangradoLinea);
    while (getchar() != '\n');
    if (sangradoLinea < 0) longSangrado = 0;
    else if (sangradoLinea > SANGRADOMAXIMO) *longSangrado = SANGRADOMAXIMO;
    else *longSangrado = sangradoLinea;
    MarcaParrafo.longitud = 5;
    MarcaParrafo.contenido[0] = '/';
    MarcaParrafo.contenido[1] = 'P';
    MarcaParrafo.contenido[2] = 'A';
    MarcaParrafo.contenido[3] = 'R';
    MarcaParrafo.contenido[4] = '/';
}

/* ********************                                    *************/
void SangrarLinea (int longSangrado)
{
    int i; // 1..SangradoMaximo;

    for (i = 1; i <= longSangrado; i++)
        AnyadirCaracterALinea(ESPACIO);
}

/* ********************                                    *************/
void InicializarLinea (int longSangrado)
{
    lineaActual.longitud = 0;
    if (longSangrado > 0) SangrarLinea(longSangrado);
}

/* ********************                                    *************/
Booleano CabeEnLinea (LongitudPalabra espacioSolicitado,
                      int longLinea)
```

```
{
    return (espacioSolicitado <= (longLinea - lineaActual.longitud));
}

/* ********************                                    *************/
void EscribirEspacios (PosEnLinea numEspacios)
{
    PosEnLinea i;
    for (i = 1; i <= numEspacios; i++)
         printf("%c", ESPACIO);
}

/* ********************                                    *************/
void EscribirLineaJustificada (int longLinea, int longSangrado)
{
    LongitudLinea numHuecos, blancosExtraMin;
    LongitudLinea numHuecosAct, numHuecosMin;
    PosEnLinea limInf, limSup, pos;

    /* Calcular limite inferior y superior */
    if (lineaActual.contenido[0] == ESPACIO)
    /* Si es primera linea de un parrafo */
       limInf = longSangrado;
    else limInf = 0;

/*
    printf("limInf %d", limInf);
    printf("\n");
    getchar();
*/

    if (lineaActual.contenido[lineaActual.longitud-1] == ESPACIO)
       limSup = lineaActual.longitud - 2;
    else limSup = lineaActual.longitud - 1;

  /*
    printf("limSup %d", limSup);
    printf("\n");
    getchar();
    */

    EscribirEspacios(limInf);

    /* Calcular numero de huecos */
    numHuecos = 0;
    for (pos = limInf; pos <= limSup; pos++)
        if (lineaActual.contenido[pos] == ESPACIO)
```

```
            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 */

/*
    printf("numHuecos %d", numHuecos);
    printf("\n");
    printf("longlinea %d", longLinea);
    printf("\n");
    getchar();

  */
    if (numHuecos != 0) {
            blancosExtraMin = (longLinea - limSup - 1) / numHuecos;
            numHuecosMin = numHuecos -
                          (longLinea - limSup - 1) % numHuecos;
    }
    /* Recorrido para insertar espacios en los huecos */

/*
    printf("numHuecosMin %d", numHuecosMin);
    printf("\n");
    getchar();
*/

    numHuecosAct = 0;
    for (pos = limInf; pos <= limSup; pos++) {
        if ((lineaActual.contenido[pos] == ESPACIO) &&
          (numHuecosAct < numHuecosMin)) {
          EscribirEspacios (blancosExtraMin + 1);
          numHuecosAct = numHuecosAct + 1;
        }
        else if ((lineaActual.contenido[pos] == ESPACIO) &&
                (numHuecosAct >= numHuecosMin))
                EscribirEspacios(blancosExtraMin+2);
        else if (lineaActual.contenido[pos] != ESPACIO)
                 printf("%c", lineaActual.contenido[pos]);
    }
}

/* *********************                                    *************/
void EscribirLineaSinJustificar()
{
    PosEnLinea i;
    for (i = 0; i <= lineaActual.longitud - 2; i++)
        printf("%c",lineaActual.contenido[i]);
```

```
    if (lineaActual.contenido[lineaActual.longitud-1] != ESPACIO)
       printf("%c",lineaActual.contenido[lineaActual.longitud-1]);
}

/* ********************                                    *************/
void EscribirLinea (Booleano justificacion,
                    int longLinea, int longSangrado)
{
    if (justificacion) EscribirLineaJustificada(longLinea, longSangrado);
    else EscribirLineaSinJustificar();
    printf("\n");
}

int main ()
{
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 6.18. Justificacion de un texto *****\n");
    printf("*****\n");
    printf("\n");
    InicializarPropLinea(&sangrado, &anchoLinea);
    InicializarLinea(sangrado);
    printf("\n");
    ComenzarPalabra();
    while (!FinDeSecuenciaPalabra) {

/*
    printf("long linea actual %d", lineaActual.longitud);
    printf("\n");


    for (int i = 1; i <= palabraActual.longitud; i++)
    printf("%c", palabraActual.contenido[i-1]);
    printf("\n");
  getchar();

  */


         if (Igual(palabraActual, MarcaParrafo)) {
            EscribirLinea(Falso, anchoLinea, sangrado);
            printf("\n");
            InicializarLinea(sangrado);
         }
         else {
               if (!CabeEnLinea(Longitud(palabraActual), anchoLinea)) {
                  EscribirLinea(Verdadero, anchoLinea, sangrado);
```

```
                InicializarLinea(0);
             }
             AnyadirPalabraALinea();
             if (CabeEnLinea(1, anchoLinea))
                AnyadirCaracterALinea(ESPACIO);
          }
          AvanzarPalabra();
    }
    EscribirLinea(Falso, anchoLinea, sangrado);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
int main()
{
    adfadf();
}
```

```
/*
 * 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' */

#include <stdio.h>
#include <conio.h>
#define MAX_ALUMNOS 200
typedef struct {
             int codigo; /* 1..MAX_ALUMNOS; */
             float nota;
} NotaAlumno;

int main ()
{
    FILE *f;
    NotaAlumno alum;
    char x;
```

```
 //   clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo GenF6_1. Genera el fichero datos6_1.dat *****\n");
    printf("*****\n");
    printf("\n");
    f = fopen("datos6_1.dat", "wb");
    do {
        printf("\n");
        printf("Introduzca el codigo del alumno (>= 1) (<= %d ): ",MAX_ALUMNOS);
        scanf("%d", &alum.codigo);
        while (getchar() != '\n');
        printf("Introduzca la nota del alumno: ");
        scanf("%f", &alum.nota);
        while (getchar() != '\n');
        fwrite(&alum, sizeof(NotaAlumno), 1, f);
        printf("¨Desea introducir mas notas? s/S (si),n/N (no): ");
        scanf("%c", &x);
        while (getchar() != '\n');
    } while ((x != 'n') && (x != 'N'));
    fclose(f);
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define MAX_ALUMNOS 200
typedef struct {
               int codigo; /* 1..MAX_ALUMNOS; */
               float nota;
} NotaAlumno;
```

```c
int main ()
{
    FILE *f;
    NotaAlumno alum;

// clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo VerF6_1. Consulta todo el fichero datos6_1.dat *****\n");
    printf("*****\n");
    printf("\n");
    f = fopen("datos6_1.dat", "rb");
    fread (&alum, sizeof(NotaAlumno), 1, f);
    printf("** Notas de los alumnos (pulsa enter para ver siguiente) **\n");
    printf("\n");
    while (!feof(f)) {
          printf(" codigo: %d\n", alum.codigo);
          printf(" nota: %.2f\n", alum.nota);
          getchar();
          fread (&alum, sizeof(NotaAlumno), 1, f);
    }
    fclose(f);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0
#define MAX_ESTUDIANTES 100
```

```
#define MarcaFinNotas -1

typedef struct {
               char nombre[30];
                      int edad;
                      Booleano sexo;
                      float nota;
                     } Estudiante;

typedef FILE *Festudiantes;
typedef Estudiante Curso[MAX_ESTUDIANTES];

int main()
{
    Festudiantes f;
    Estudiante est;
    char x;

// clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo GenF6_4. Genera el fichero datos6_4.dat *****\n");
    printf("*****\n");
    printf("\n");
    f = fopen("datos6_4.dat", "wb");
    // Se introducen los datos de los estudiantes en la tabla
    printf(" Introduzca los datos de los estudiantes (nota=-1 para terminar)\n");
    do {
        printf("\n");
        printf(" Introduzca nota: ");
        scanf("%f", &est.nota);
        while (getchar() != '\n');
     if (est.nota == MarcaFinNotas) break;
        printf(" Introduzca el nombre: ");
        scanf("%s", &est.nombre);
        while (getchar() != '\n');
        printf(" Introduzca la edad: ");
        scanf("%d", &est.edad);
        while (getchar() != '\n');
        do {
            printf(" M/m mujer, H/h hombre: ");
            scanf("%c", &x);
            while (getchar() != '\n');
        } while ((x != 'm') && (x != 'M') && (x != 'h') && (x != 'H'));
            est.sexo = (x == 'M') || (x == 'm');
        fwrite(&est, sizeof(Estudiante), 1, f);
    } while (Verdadero);
```

```
    fclose(f);
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
#define Booleano int
#define Verdadero 1
#define Falso 0
#define MarcaFinNotas -2
#define MAX_ESTUDIANTES 100

typedef struct {
               char nombre[30];
                      int edad;
                      Booleano sexo;
                      float nota;
                     } Estudiante;

typedef FILE *Festudiantes;
typedef Estudiante Curso[MAX_ESTUDIANTES];

int main()
{
    Festudiantes f;
    Estudiante est;

// clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo VerF6_4. Consulta todo el fichero datos6_4.dat *****\n");
```

```
    printf("*****\n");
    printf("\n");
    f = fopen("datos6_4.dat", "rb");
    printf("** Datos de los estudiantes (pulse enter para ver siguiente) **\n");
    printf("\n");
    do {
        fread(&est, sizeof(Estudiante), 1, f);
    if (feof(f)) break;
        printf(" nombre: %s\n", est.nombre);
        printf(" edad: %d\n", est.edad);
        printf(" sexo: ");
        switch (est.sexo) {
             case Verdadero: printf("mujer\n"); break;
             case Falso: printf("hombre\n"); break;
        }
        printf(" nota: %.2f\n", est.nota);
        getchar();
    } while (Verdadero);
    fclose(f);

    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

3.6. Capítulo 7

```c
/*
 * 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
 */

#include <stdio.h>
#include <conio2.h>
long int Factorial (int n)
{
/* PRE: n >= 0 */
/* POST: Factorial (n) = n! */
   long int resultado;
   if (n == 0) resultado = 1;
   else if (n > 0) resultado = n * Factorial (n-1);
   return (resultado);
}

main ()
{
    int n;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.1. Funcion factorial recursiva *****\n");
    printf("*****\n");
    printf("\n");
    printf ("Numero (>= 0) y (<= 16): ");
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\n");
    printf("El factorial de %d es: %ld\n", n, Factorial (n));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```c
/*
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>
typedef struct {
               int c;
               int r;
               } Par;
Par DivisionNatural (int num, int den)
// PRE: (num >= 0) y (den > 0)
// POST: DivisionNatural (num,den) = <c, r> y num = c * den + r y 0 <= r < den
{
    Par p;
    if (num < den) {
                   p.c = 0;
                   p.r = num;
    }
    else {
          p = DivisionNatural(num-den, den);
          p.c = p.c + 1;
    }
    return p;
}

int main ()
{
    int dividendo, divisor;
    Par p;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.2. Funcion division natural recursiva *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el dividendo (>= 0): ");
    scanf("%d", &dividendo);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca el divisor (> 0): ");
```

```
    scanf("%d", &divisor);
    while (getchar() != '\n');
    printf("\n");
    p = DivisionNatural(dividendo, divisor);
    printf("Resultado: cociente = %d resto = %d\n", p.c, p.r);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>
#define MAX 10
typedef float Vector[MAX];

float pei (Vector v1, Vector v2, int a, int b)
       // 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]
{
    float peiaux;
        if (a == b) peiaux = v1[a] * v2[a];
        else if (a < b) peiaux = v1[a] * v2[a] + pei(v1, v2, a+1, b);
        return peiaux;
}

main ()
{
    Vector vv1, vv2;
    int n, i;


    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.3. Funcion producto escalar recursiva lineal *****\n");
    printf("*****\n");
```

```
    printf("\n");
    printf("Introduzca el numero de elementos de los vectores (> 0) y (<=%d): ", MAX);
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\nPrimer vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv1[i]);
         while (getchar() != '\n');
    };
    printf("\nSegundo vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv2[i]);
         while (getchar() != '\n');
    };
    printf("\nEl producto escalar de los vectores es: %.2f\n", pei(vv1, vv2, 0, n-1));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>
#define MAX 10
typedef float Vector[MAX];

float peii(Vector v1, Vector v2, int a, int b, float c)
// 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]
{
    float peiiaux;

        if (a == b) peiiaux = c + v1[a] * v2[a];
        else /* a < b */ peiiaux = peii (v1, v2, a+1, b, c + v1[a] * v2[a]);
        return peiiaux;
```

```
}

int main()
{
    Vector vv1, vv2;
    int n, i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.4. Funcion producto escalar recursiva lineal final *****\n
        ");
    printf("*****\n");
    printf("\n");

    printf("Introduzca el numero de elementos de los vectores (> 0) y (<=%d): ",MAX);
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\nPrimer vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv1[i]);
         while (getchar() != '\n');
    };
    printf("\nSegundo vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv2[i]);
         while (getchar() != '\n');
    };
    printf("\nEl producto escalar de los vectores es: %.2f\n", peii (vv1, vv2, 0, n-1,
        0));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * Capitulo 7. Disenyo recursivo
```

```
 */

#include <stdio.h>
#include <conio.h>
#define MAX 10
typedef float Vector[MAX];

float peii (Vector v1, Vector v2, int a, int b, float c)
       // 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]
{
       int a_loc;      // corresponde al parametro a
       float c_loc; // corresponde al parametro c
                              // el parametro b no varia, no necesita copia local

       a_loc = a;
    c_loc = c;
       while (a_loc < b) {
                 c_loc = c_loc + v1[a_loc] * v2[a_loc];
                 a_loc = a_loc + 1;
       }
       return c_loc + v1[a_loc] * v2[a_loc];
}

int main()
{
    Vector vv1, vv2;
    int n, i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.5. Funcion producto escalar iterativa, *****\n");
    printf("*****              equivalente a la del Algoritmo 7.4 *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos de los vectores (> 0) y (<=%d): ",MAX);
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\nPrimer vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv1[i]);
         while (getchar() != '\n');
    };
    printf("\nSegundo vector\n");
```

```
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv2[i]);
         while (getchar() != '\n');
    };
    printf("\nEl producto escalar de los vectores es: %.2f\n", peii (vv1, vv2, 0, n-1,
        0));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * í                 Mara J. Majado Rosales
 * Capitulo 7. ñDiseo recursivo
 */

#include <stdio.h>
#include <conio.h>
#define MAX 10
typedef float Vector[MAX];
int n;

float pe (Vector v1, Vector v2)
{
    int a, b;
        float c;

        a = 0; b = n-1; c = 0;
        while (a < b) {
               c = c + v1[a] * v2[a];
               a = a + 1;
        }
        return c + v1[a] * v2[a];
}


int main()
{
    Vector vv1, vv2;
```

```
    int i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.6. Funcion producto escalar iterativa sin *****\n");
    printf("*****              parametros de inmersion (version 1) *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos de los vectores (> 0) y (<=%d): ",MAX);
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\nPrimer vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv1[i]);
         while (getchar() != '\n');
    };
    printf("\nSegundo vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv2[i]);
         while (getchar() != '\n');
    };
    printf("\nEl producto escalar de los vectores es: %.2f\n", pe (vv1, vv2));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * í                   Mara J. Majado Rosales
 * Capitulo 7. ñDiseo recursivo
 */

#include <stdio.h>
#include <conio.h>
#define MAX 10
typedef float Vector[MAX];
int n;
```

```
float pe (Vector v1, Vector v2)
{
    int a, b;
        float c;

    a = 0; b = n-1; c = 0;
        while (a <= b) {
                c = c + v1[a] * v2[a];
                a = a + 1;
        }
        return c;
}

int main()
{
    Vector vv1, vv2;
    int i;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.7. Funcion producto escalar iterativa (version 2) *****\n"
        );
    printf("*****\n");
    printf("\n");
    printf("Introduzca el numero de elementos de los vectores (> 0) y (<=%d): ",MAX);
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\nPrimer vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv1[i]);
         while (getchar() != '\n');
    };
    printf("\nSegundo vector\n");
    for (i = 0; i <= n-1; i++) {
         printf("  Componente %d: ", i+1);
         scanf("%f", &vv2[i]);
         while (getchar() != '\n');
    };
    printf("\nEl producto escalar de los vectores es: %.2f\n", pe(vv1, vv2));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
```

```c
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>

float PotenciaNatural (float a, int n)
// PRE (n >= 0) y (n = 0 => a <> 0)
// POST PotenciaNatural(a, n) = a elevado a n
{
    float pnaux;

    if (n == 0)
        pnaux = 1.0;
    else /* n > 0 */
        pnaux = a * PotenciaNatural(a, n-1);
    return pnaux;

}

int main()
{
    float base;
    int exponente;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.8. Funcion potencia natural recursiva *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un real como base de la potencia: ");
    scanf("%f",&base);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca un entero como exponente de la potencia (>= 0): ");
    scanf("%d",&exponente);
    while (getchar() != '\n');
    printf("\n");
    printf("Resultado: %.2f\n", PotenciaNatural(base, exponente));
```

```
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * Algoritmo 7.9. Calculo recursivo de Fibo(n) con inmersion de resultados por
    eficiencia
 * 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. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>

typedef struct {
               int p;
               int s;
               } Par;

Par iFibo (int n)
// PRE n > 0
// POST Devuelve el par < Fibo(n), Fibo(n-1) >
{
    int fib, fibAnt;
    Par par, parAux;
        if (n == 1) {
          par.p = 1;
          par.s = 1; /* iF = < Fibo(1), Fibo(0) > */
    }
        else /* n > 1 */ {
               parAux = iFibo (n - 1);
                      /*
                       * tenemos: parAux.p = Fibo(n-1) y parAux.s = Fibo(n-2), luego:
                       * Fibo(n) = Fibo(n-1) + Fibo(n-2) = parAux.p + parAux.s
                       * Fibo(n-1) = parAux.p :
                       */
               par.p = parAux.p + parAux.s;
        par.s = parAux.p;
        }
        return par;
}
```

```
int main()
{
    int n, fib, fibAnt;
    Par par;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.9. Calculo recursivo de Fibo(n) con inmersion *****\n");
    printf("*****              de resultados por eficiencia           *****\n");
    printf("*****\n");
    printf("\n");
    printf("Termino de la sucesion que se desea calcular (> 0): ");
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\n");
    par = iFibo (n);
    printf("El termino %d-esimo (posicion %d) de la sucesion es: %d\n", n, n+1, par.p);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>
typedef struct {
               int p;
               int s;
               } Par;


Par iFibo (int n)
// PRE n > 0
// POST Devuelve el par < Fibo(n), Fibo(n-1) >
{
```

```
    int fib, fibAnt;
    Par par, parAux;

        if (n == 1) {
                par.p = 1;
                par.s = 1; // a = Fibo(1), b = Fibo(0)
    }
        else { /* n > 1 */
               parAux = iFibo (n - 1);
                      /*
                       * tenemos: parAux.p = Fibo(n-1) y parAux.s = Fibo(n-2), luego:
                       * Fibo(n) = Fibo(n-1) + Fibo(n-2) = parAux.p + parAux.s
                       * Fibo(n-1) = parAux.p :
                       */
               par.p = parAux.p + parAux.s; par.s = parAux.p;
        }
        return par;
}

int Finterf (int n)
// PRE n > 0
// POST Finterf (n) = Fibo(n)
{
    int fib, fibAnt;
    int aux;
    Par par;

        if (n == 0) aux = 1;
        else { /* n > 0 */
               // se llama a iFibo y se descarta su segundo resultado
               par = iFibo (n);
               aux = par.p;
    }
    return aux;
}

int main()
{
    int n;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.10. Funcion interfaz para la funcion del Algoritmo 7.9
        *****\n");
    printf("*****\n");
    printf("\n");
```

```
    printf("Termino de la sucesion que se desea calcular (>= 0): ");
    scanf("%d",&n);
    while (getchar() != '\n');
    printf("\n");
    printf("El termino %d-esimo (posicion %d) de la sucesion: %d\n", n, n+1, Finterf (n
        ));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>

float PotNat (float a, int n)
// PRE (n >= 0) y (n = 0 => a <> 0)
// POST PotNat(a, n) = a elevado a n
{
    float p, potAux;


    if (n == 0)
        potAux = 1.0;
    else { /* n > 0 */
        p = PotNat (a, n / 2);
        if (n % 2 == 0) potAux = p * p;
        else potAux = a * p * p;
    }
    return potAux;
}

int main()
{
    float base;
    int exponente;
```

```
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.11. Funcion potencia natural mejorada *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca un real como base de la potencia: ");
    scanf("%f",&base);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca un entero como exponente de la potencia (>= 0): ");
    scanf("%d",&exponente);
    while (getchar() != '\n');
    printf("\n");
    printf("Resultado: %.2f\n", PotNat(base, exponente));
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>
typedef struct {
               int coc;
               int res;
               } Par;

Par DivNat(int num, int den)

 // PRE: (num >= 0) y (den > 0)
 // POST: DivNat (num, den, c, r) y num = coc * den + res y 0 <= res < den
{
    Par par;

    if (num < den) {
         par.coc = 0;
```

```
          par.res = num;
    }
    else {
          par = DivNat(num, den * 2);
          par.coc = par.coc * 2;
          if (par.res >= den) {
               par.coc = par.coc + 1;
               par.res = par.res - den;
          }
    }
    return par;
}

int main()
{
    int dividendo, divisor;
    Par par;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.12. Funcion division natural mejorada *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el dividendo (>= 0): ");
    scanf("%d",&dividendo);
    while (getchar() != '\n');
    printf("\n");
    printf("Introduzca el divisor (> 0): ");
    scanf("%d",&divisor);
    while (getchar() != '\n');
    par = DivNat(dividendo, divisor);
    printf("\n");
    printf("Resultado: cociente = %d resto = %d\n", par.coc, par.res);
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
```

```
 * Capitulo 7. Disenyo recursivo
 */

#include <stdio.h>
#include <conio.h>
#define MAX 15
typedef char Estaca; // 'A'..'C'; El tipo Estaca sera un rango de caracteres

void mover(Estaca X, Estaca Y)
// PRE ...              la explicada en el texto
// POST ...             la explicada en el texto
{

                      /*
                       * 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.
                       */

    printf(" mover disco de la estaca %c a la estaca %c\n", X, Y);
}

void Hanoi(int n, Estaca origen, Estaca intermedio, Estaca destino)
// PRE ...               la explicada en el texto
// POST ...              la explicada en el texto
{
     switch (n)
     {
        case 1 : mover(origen, destino); break;
        default :
            Hanoi(n-1, origen, destino, intermedio);
            mover(origen, destino);
            Hanoi(n-1, intermedio, origen, destino);
            break;
     }
}

int main()
{
    int ndiscos;

    clrscr();
    printf("\n");
```

```
    printf("*****\n");
    printf("***** Algoritmo 7.13. Accion recursiva que resuelve el *****\n");
    printf("*****              problema de las torres de Hanoi *****\n");
    printf("*****\n");
    printf("\n");
    printf("Numero de discos (maximo %d): ", MAX);
    scanf("%d",&ndiscos);
    while (getchar() != '\n');
    printf("\n");
    Hanoi(ndiscos, 'A', 'B', 'C');
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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. Disenyo 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.
 */

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 3000
#define Booleano int
#define Verdadero 1
#define Falso 0
typedef int TipoBase;
```

```
typedef int indice; // 1..MAX
typedef TipoBase TDatos[MAX];


void Comprueba(TDatos v, indice mr)
/*
 * 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].
 */
{
    indice i;
    Booleano error;

    i = 0;
    error = Falso;
    while ((i < mr-1) && (!error)) {
          if (v[i] > v[i+1]) {
              printf("<<< ERROR >>> (i=%d)", i);
                        error = Verdadero;
          }
          else i = i + 1;
    }
}

int randn(int n)
{
   return rand()%n + 1;
}

void InitTDatos(TipoBase *v, indice *mr)
{
    indice i;
        int seed, maximo;

        printf("Semilla aleatoria: ");
    scanf("%d",&seed);
    while (getchar() != '\n');
    srand(seed);
    printf("\n");
    printf("Elementos en el vector (minimo 1, maximo %d): ", MAX);
    scanf("%d",mr);
    while (getchar() != '\n');
    printf("\n");
    printf("Generar valores entre cero y: ");
    scanf("%d",&maximo);
```

```
    while (getchar() != '\n');
    for (i = 0; i <= *mr-1 ; i++)
        v[i] = randn(maximo);
}

void MostrarTDatos(TipoBase *v, indice *mr)
{
    indice i;

    i = 0;
    while (1) { // ITERAR
        printf("%d", v[i]);
    if (i == *mr-1) break; // DETENER i = mr;
        printf(", ");
        i = i + 1;
     };         // FIN_ITERAR
     printf("\n");
}

void Intercambio(TipoBase *v, indice i, indice j)
{
    TipoBase t;

    t = v[i];
    v[i] = v[j];
    v[j] = t;
}

void Particion(TipoBase *v, indice inf, indice sup, indice *pivote)
{
    indice iz, dr;

    iz = inf + 1;
    dr = sup;
    while (iz != dr + 1) {
          while ((iz <= dr) && (v[iz] <= v[inf]))
                iz = iz + 1;
          while ((iz <= dr) && (v[dr] >= v[inf]))
                dr = dr - 1;
          if (iz < dr) {
              Intercambio(v, iz, dr);
              iz = iz + 1;
              dr = dr - 1;
          }
    }
    Intercambio(v, inf, dr);
    *pivote = dr;
```

```
}

void ClasificacionRapidaI(TipoBase *v, int inf, int sup)
    /*
     * 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.
     */
{
    indice p;

    if (inf < sup) {
        Particion(v, inf, sup, &p);
        ClasificacionRapidaI(v, inf, p-1);
        ClasificacionRapidaI(v, p+1, sup);
    }
}

void ClasificacionRapida(TipoBase *v, indice elementos)
{
    ClasificacionRapidaI(v, 0, elementos-1);
}

int main()
{
    TDatos t;
    indice max_real;

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 7.14. Algoritmo de ordenacion rapida de Hoare *****\n");
    printf("***** Algoritmo 7.15. Accion de particionamiento para la accion de *****\n")
        ;
    printf("*****              clasificacion rapida                    *****\n");
    printf("*****\n");
    printf("\n");
    InitTDatos(t, &max_real);
    printf("\n");
    printf("\nVector antes de la clasificacion:\n");
    MostrarTDatos(t, &max_real);
    printf("Enter para ordenar...");
    getchar();
```

```
    ClasificacionRapida(t, max_real);
    printf("\n");
    printf("Vector despues de la clasificacion:\n");
    MostrarTDatos(t, &max_real);
    Comprueba(t, max_real);

    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

3.7. Capítulo 8

```
/*
 * 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 correto 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 serviran 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.
```

```
 */

#include <stdio.h>
#include <stdlib.h>
#include <conio2.h>
#include "msc1.h"
#define Booleano int
#define Verdadero 1
#define Falso 0

typedef struct {
               char signo;
               int linea, columna;
              } datos_pila;
typedef struct np {
                  datos_pila datos;
                  struct np *sig;
                 } nodo_signos;
typedef nodo_signos *pila_signos;

// ---------------- Operaciones de la Pila -----------------

pila_signos PilaVacia()
{
     return NULL;
}

Booleano EsPilaVacia(pila_signos p)
{
     return p == PilaVacia();
}

pila_signos Apilar(pila_signos p, datos_pila d)
{
    pila_signos q;

    q = (pila_signos) malloc(sizeof(nodo_signos));
    q->datos = d;
    q->sig  = p;

    return q;
}

void Cima(pila_signos p, datos_pila *c)
{
     *c = p->datos;
}
```

```
pila_signos Desapilar(pila_signos p)
{
    pila_signos q;

    q = p->sig;
    free(p);
    return q;
}

pila_signos DestruirPila(pila_signos p)
{
     while (!EsPilaVacia(p)) p = Desapilar(p);
     return p;
}

// ---------------- Operaciones del Programa -----------------

char SignoCierre(char apertura)
/*
 * Devuelve el signo de cierre correspondiente al signo
 * de apertura 'apertura'.
 *
 * Pre: el caracter 'apertura' es un signo de apertura
 */
{
    char res;

    switch (apertura) {
        case '[' : res = ']'; break;
        case '(' : res = ')'; break;
        case '{' : res = '}'; break;
     };
     return res;
}

// ---------------- Programa Principal -----------------

int main ()
{
    Msc1 S;
    pila_signos p;
    datos_pila d;
    Booleano error_signo, error_pila;
    int linea_actual;  // linea actual del recorrido
    int columna_actual; // columna actual del recorrido
```

```
    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 8.1. Comprobar el equilibrio de los signos *****\n");
    printf("*****              puntuacion de apertura/cierre      *****\n");
    printf("*****\n");
    printf("\n");
    S = Init_MSC1();
    Cargar_Fichero_MSC1(S, "dat8_1A.txt");
    printf("La secuencia de entrada es: ");
    printf("\n");
    printf("\n");
    Comenzar_MSC1 (S);
    while ((EA_MSC1 (S) != MSC1_MarcaFin())) {
           printf("%c", EA_MSC1 (S));
           Avanzar_MSC1(S);
    };
    printf("\n");
    printf("\n");
    Comenzar_MSC1(S);
    if (EA_MSC1(S) == MSC1_MarcaFin())
       printf("Secuencia vacia\n");
    else {
         p = PilaVacia();
         linea_actual  = 1;
         columna_actual = 1;
         error_signo   = Falso;
         error_pila    = Falso;
         do {
             if ((EA_MSC1(S) == '[') ||
                 (EA_MSC1(S) == '(') ||
                 (EA_MSC1(S) == '{')) {
                    d.signo  = EA_MSC1(S);
                    d.linea  = linea_actual;
                    d.columna = columna_actual;
                    p = Apilar(p, d);
               }
             else if ((EA_MSC1(S) == ']') ||
                      (EA_MSC1(S) == ')') ||
                      (EA_MSC1(S) == '}')) {
                       if (EsPilaVacia(p))
                           error_pila = Verdadero;
                       else {
                            Cima(p, &d);
                            p = Desapilar(p);
                            error_signo = EA_MSC1(S) != SignoCierre(d.signo);
                       }
```

```
    }
    else if (EA_MSC1(S) == MSC1_MarcaFinLinea) {
            linea_actual = linea_actual + 1;
            columna_actual = 0;
    };
    /*
     * 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 (!(error_signo || error_pila))
    {
     Avanzar_MSC1(S);
     columna_actual = columna_actual + 1;
    }
} while (!((EA_MSC1(S) == MSC1_MarcaFin()) || error_signo || 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 p}ientes 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
 *     enontrado (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)
{
 printf("Error: signo de cierre '%c' en ", EA_MSC1(S));
 printf("linea %d, columna %d,\n", linea_actual, columna_actual);
 printf("no tiene signo de apertura correspondiente\n");
}
else if (error_signo) {
        printf("Error: signo de cierre '%c' en ", EA_MSC1(S));
        printf("linea %d, columna %d,\n", linea_actual, columna_actual);
        printf("no se corresponde con signo de apertura ");
        printf("'%c' en linea %d", d.signo, d.linea);
        printf(", columna %d\n", d.columna);
}
```

```
        else if (!EsPilaVacia(p))
        {
             printf("Error: se agoto el texto con los siguientes \n");
             printf("signos de apertura pendientes de cerrar:\n\n");
             do {
                  Cima(p, &d);
                  p = Desapilar(p);
                  printf(" signo: '%c'", d.signo);
                  printf(" linea %d, columna %d", d.linea, d.columna);
             } while (!EsPilaVacia(p));
        }
        else {
            printf("Texto correcto\n");
        }
        p = DestruirPila(p);
    }
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    FILE *a;
    int i;
    char nombreFichero[20];

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 8.2. Crear un archivo secuencial que almacena enteros *****\
        n");
```

```
    printf("*****\n");
    printf("\n");
    printf("Introduzca el nombre del fichero de entrada: ");
    scanf("%s", &nombreFichero);
    while (getchar() != '\n');
    printf("\n");
    a = fopen(nombreFichero, "wb");
    while (1) {
              printf(" Introduzca un numero entero (0 para terminar): ");
              scanf("%d",&i);
              while (getchar() != '\n');
     if (i == 0) break;
              fwrite (&i, sizeof(int), 1, a); // se almacena en formato binario
    }
    fclose(a);
    printf("\n");
    printf("Fichero %s creado", nombreFichero);
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    FILE *a;
    int i;
    char nombreFichero[20];

    clrscr();
    printf("\n");
    printf("*****\n");
```

```
    printf("***** Algoritmo 8.3. Recorrido de un archivo secuencial de enteros *****\n")
        ;
    printf("*****              para mostrarlos                          *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el nombre del fichero de entrada: ");
    scanf("%s", &nombreFichero);
    while (getchar() != '\n');
    printf("\nEl fichero consta de los siguientes elementos: ");
    a = fopen(nombreFichero, "rb");

       /*
     * Notese que este algoritmo sigue el primer esquema de
         * recorrido del primer modelo de acceso secuencial
         * en el que se tiene:
         * Comenzar: fread
         * FinDeSecuencia: feof
         * Avanzar: fread
     */

    fread(&i, sizeof(int), 1, a);
    while (!feof(a)) {
          printf("%d ", i);
          fread(&i, sizeof(int), 1, a); // se almacena en formato binario
    }
    fclose(a);
    printf("\n");

    printf("\nPulse enter para continuar");
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>
```

```c
#include <stdlib.h>

int main()
{
    FILE *a;
    int i, p;
    int tamanyo;
    char nombreFichero[20];

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 8.4. Acceso directo a un archivo para *****\n");
    printf("*****              modificar un elemento          *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el nombre del fichero de entrada: ");
    scanf("%s", &nombreFichero);
    while (getchar() != '\n');
    a = fopen(nombreFichero, "r+b");
    printf("\n");
    printf("Posicion que desea modificar (>= 0): ");
    scanf("%d",&p);
    while (getchar() != '\n');
    printf("\n");
    fseek(a, 0, SEEK_END);
             // se posiciona al final del archivo
    tamanyo = ftell(a)/sizeof(int);
             // se obtiene el únmero de enteros que contiene el archivo
    if ((p < 0) || (p > tamanyo-1))
          printf("Posicion fuera del archivo");
    else {
          fseek(a, p * sizeof(int), SEEK_SET);
          fread(&i, sizeof(int), 1, a);
          printf("Valor actual en la posicion %d: %d\n", p, i);
          printf("\n");
          printf("Introduzca el nuevo valor: ");
          scanf("%d",&i);
          while (getchar() != '\n');
          fseek (a, p * sizeof(int), SEEK_SET);
          fwrite(&i, sizeof(int), 1, a);
          printf("\n");
          printf("Fichero %s actualizado", nombreFichero);
    }
    fclose(a);
    printf("\n");
    printf("\nPulse enter para continuar");
```

```
    getchar();
    return 0;
}
```

```
/*
 * 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
 */

#include <stdio.h>
#include <conio.h>

int main()
{
    FILE *a;
    int i;
    char nombreFichero[20];

    clrscr();
    printf("\n");
    printf("*****\n");
    printf("***** Algoritmo 8.5. Anyadir un elemento al final de un archivo *****\n");
    printf("*****\n");
    printf("\n");
    printf("Introduzca el nombre del fichero de entrada: ");
    scanf("%s", &nombreFichero);
    while (getchar() != '\n');
    a = fopen(nombreFichero, "ab");
    printf("\n");
    printf("Dato entero para anyadir al final del fichero: ");
    scanf("%d",&i);
    while (getchar() != '\n');
    fwrite(&i, sizeof(int), 1, a);
    fclose(a);
    printf("\n");
    printf("Fichero %s actualizado", nombreFichero);
    printf("\n");
    printf("\nPulse enter para continuar");
    getchar();
    return 0;
```

```
}
```