Examen Fundamentos de Programación

15 de enero de 2016 **Curso 2015/16**

Duración: 2,45 horas 10 PUNTOS

Publicación de notas 26 de enero
Revisión de examen 1 de febrero
En el horario y lugar que se publicará con las notas del examen

Problema 1: [1 punto]

La sociedad secreta informática utiliza la sucesión de Fibonacci definida como:

$$fibonacci(n) = \begin{cases} 1 \, si \, n = 1 \\ 2 \, \text{si} \, n = 2 \\ fibonacci(n-1) + fibonacci(n-2) \, para \, todo \, n > 2 \end{cases}$$

para enmascarar sus mensajes secretos de forma que solo los caracteres que ocupan una posición que coincide con cada uno de los términos de esta sucesión tienen que ser considerados para descifrar la palabra oculta.

SE PIDE:
Escribir un **programa** en C que lea caracteres desde la entrada estándar hasta detectar el fin de fichero y que escriba por pantalla la palabra secreta oculta en el mensaje.

Por ejemplo, para la entrada:
***APR**RO**OOB**BBBB**A**AAAAAAA**D**DDDDDDDDDDD**O**OPQREST*
La palabra oculta es:
APROBADO

SOLUCIÓN:

```
#include <stdio.h>

int fibonacci(int);

void main(void)
{
    char c;
    int contadorCaracteres = 0;
    int posicionSiguienteEnMensaje = 1;
```

```
    while ((c=getchar()) != EOF)
    {
        contadorCaracteres++;
        if (contadorCaracteres ==
            fibonacci(posicionSiguienteEnMensaje))
        {
            putchar(c);
            posicionSiguienteEnMensaje++;
        }
    }
}

int fibonacci (int n)
{
    if (n == 1 || n ==2)
        return n;
    return fibonacci(n-1)+fibonacci(n-2);
}
```

Problema 2: [2,5 puntos]

En una matriz cuadrada ***m*** de orden *N* se dan las definiciones siguientes:

1. Son **adyacentes** a un elemento a_{ij} de la matriz ***m*** aquellos a_{kl} tales que:

$$\begin{cases} i-1 \le k \le i+1,\ con\ i-1 \ge 0,\ i+1 < N \\ j-1 \le l \le j+1,\ con\ j-1 \ge 0,\ j+1 < N \end{cases}, \text{ con } i \ne k\ ||\ j \ne l$$

2. Se dice que un elemento a_{ij} de la matriz *m* es un ***acumulador*** si y solo si a_{ij} es la **suma** de todos los valores de sus adyacentes.

Ejemplo: Dada la matriz $m = \begin{pmatrix} \mathbf{6} & 2 & -1 & 5 \\ 2 & 2 & \mathbf{6} & 2 \\ 1 & 1 & -6 & 1 \\ 0 & 2 & 3 & \mathbf{-2} \end{pmatrix}$

Acumuladores de m = {6 posición (0, 0), 6 posición (1, 2) y -2 posición (3, 3)}

SE PIDE:

a) (2 puntos) Escribir una **función** en C con el prototipo siguiente:

int esAcumulador (int i, int j, int m[][N])

que devuelva ***1*** si el elemento ***m[i][j]*** es un acumulador en la matriz ***m***, y ***0,*** en otro caso.

b) (0,5 puntos) Escribir un **programa** en C que escriba todos los elementos de una matriz cuadrada de orden *N* que son acumuladores indicando la posición que ocupan en la matriz como en el ejemplo utilizando la función anterior. Si la matriz *m* no tuviese acumuladores se escribirá el mismo mensaje sin ningún valor entre las llaves.

NOTA: NO se pueden utilizar estructuras auxiliares para la resolución del problema.

SOLUCIÓN:

Apartado a)

```
int esAcumulador (int i, int j, int m[][N])
{
    int k, l;
    int suma = 0;

    for (k = i - 1; k <= i + 1; k++)
    {
        for (l = j - 1; l <= j + 1; l++)
          if (k >= 0 && k < N && l>= 0 && l < N && (i != k || j != l))
                    suma += m[k][l];
    }

    return suma == m[i][j];
}
```

Apartado b)

```
#include <stdio.h>

#define N 4

int esAcumulador (int, int, int [][N]);

void main()
{
    int i,j;
    int m[N][N] = {
                    {6, 2, -1, 5},
                    {2, 2, 6, 2},
                    {1, 1, -6, 1},
                    {0, 2, 3, -2}
                };

    printf("Acumuladores de m = {");
    for (i=0; i<N; i++)
        for(j=0; j<N; j++)
            if (esAcumulador(i,j,m))
                printf("%d posicion(%d, %d), ",m[i][j],i,j);

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

Problema 3: [3 puntos]

Sea *N* una constante entera positiva.
Se dice que una matriz cuadrada de orden N de números enteros ***es una Z*** sí y solo sí los únicos elementos no nulos de la matriz son los de su primera y última fila y los de su diagonal secundaria.
Se dice que una matriz cuadrada de orden N de números enteros en una N sí y solo sí los únicos elementos no nulos de la matriz son los de su primera y última columna y los de su diagonal principal.

Se pide:

a) Escribir una función en C con el prototipo

int esUnaZeta (int A[][N])

que devuelva 1 si la matriz *A* es una Z, y 0, en caso contrario. (1,5 puntos)

b) Escribir un programa en C que determine si una matriz cuadrada de orden N es una Z, y en caso de que lo sea, transforme la matriz en una N. En este paso no se permite el uso de arrays auxiliares para realizar la transformación. (1,5 puntos).

SOLUCIÓN:

```
#include <stdio.h>

#define N 5

int esUnaZ (int [][N]);
void transformarMatrizN(int [][N]);

int main(int argc, const char * argv[]) {

    int matriz[N][N] = {{2, 3, 4, 5, 6},
                        {0, 0, 0, 5, 0},
                        {0, 0, 4, 0, 0},
                        {0, 3, 0, 0, 0},
                        {1, 2, 3, 4, 5}};

    if (esUnaZ(matriz))
        transformarMatrizN(matriz);

    return 0;
}
```

```
int esUnaZ(int m[N][N])
{
    int i,j;

    for (j = 0; j< N; j++)
        if (!m[0][j] || !m[N-1][j] || !m[j][N-1-j])
            return 0;

    for (i=1; i<N-1; i++)
        for (j=0; j<N; j++)
            if (m[i][j] && j!=N-1-i)
                return 0;

    return 1;
}

void transformarMatrizN(int m[N][N])
{
    int i,aux;

    for (i=1; i<N; i++)
    {
        aux = m[0][i];
        m[0][i] = m[i][0];
        m[i][0] = aux;
    }
    for (i=1; i<N; i++)
    {
        aux = m[N-1][i];
        m[N-1][i] = m[i][N-1];
        m[i][N-1] = aux;
    }
    for (i=1; i<N-1; i++)
    {       m[i][i] = m[i][N-1-i];
            if (i != (N-1-i))
                m[i][N-1-i] = 0;
    }
}
```

Problema 4: [3,5 puntos]

Se ha solicitado a los 50 críticos de cine más influyentes de *Chikitistán,* las valoraciones de los estrenos de películas de las pasadas navidades. Estas están recogidas en el fichero "***Estrenos.txt***".
Cada línea del fichero contiene el nombre de una película sin espacios en blanco y las 50 valoraciones que ha recibido de los críticos.

Por ejemplo: LosJuegosDelHambreSinsajo 2 5 1 8 4 7 … 8

Considerando los siguientes tipos de datos:

```
#define MAXNOM 30
struct Tinfo
{
        char    nombrePelicula[MAXNOM];
        int     valoracion;
}
```

SE PIDE: Escribe un programa en C que: (2,5 puntos)

a) **Abra** y **cierre** el fichero con la información de las valoraciones emitidas por los críticos de manera adecuada, una única vez, para **leer** y **procesar** la información que se necesite de él.
b) **Define** la estructura adecuada para mantener la información de **cinco** películas con la valoración acumulada de la película.
c) **Lea**, **busque** en el fichero, y **muestre** por pantalla las **cinco películas** con **mejor valoración**.

Competencia de resolución de problemas: (1 punto)

Justifica la estructura de datos que eliges para resolver el apartado b) de este problema.
Describe los pasos fundamentales del algoritmo que utilizas para resolver el apartado c) del problema.

SOLUCIÓN:

```
#include <stdio.h>

#define TOPE 5
#define MAXNOM 30
#define NUM 50

struct Tinfo
{
 char nombrePelicula[MAXNOM];
 int  valoracion;
};
```

```
struct TLista
{
   struct Tinfo Peliculas[TOPE];
   int          cont;
};

void aniadir (struct Tinfo, struct TLista *);
void print   (struct TLista);

int main()
{
     FILE * fp;

     struct Tinfo  actual;
     struct TLista lista;
     int i; int valor;
     lista.cont = 0;

     if ((fp = fopen("Estrenos.txt","r")) == NULL)
          {
           printf ("\nNo se puede abrir el fichero\n");
           return (-1);
          }

    while(fscanf(fp,"%s",actual.nombrePelicula) != EOF)
    {
               actual.valoracion = 0;
               for (i = 0; i < NUM; i++)
                 {   fscanf(fp, "%d", &valor );
                     actual.valoracion += valor;
                 }
               aniadir (actual, &lista);
    }

    fclose(fp);

    print(lista);
    return 0;
}
```

```
void print (struct TLista lista)
{
  int i;

  printf("\nLas cinco peliculas con mejor valoracion
                en navidades\n");
  printf("************************************************\n");

  for (i = 0; i < lista.cont; i++)
      printf ("Pelicula: %s valoracion: %d\n",
                lista.Peliculas[i].nombrePelicula,
                lista.Peliculas[i].valoracion);

}
void aniadir (struct Tinfo actual, struct TLista *lista)
{
  int i = 0;
  int j = lista -> cont;

  if (j == 0)
  {
    lista->Peliculas[i] = actual;
    lista->cont++;
  }

  for(i = 0; i < j; i++)
      if (actual.valoracion > lista->Peliculas[i].valoracion)
      break;

  if (i == j && j<TOPE)
    lista->Peliculas[i] = actual;
  else
  {
    for (; i < j; j--)
      lista->Peliculas[j] = lista->Peliculas[j-1];
    lista->Peliculas[i] = actual;
  }

  lista->cont++;
  if (lista->cont > TOPE) lista->cont = TOPE;

}
```

Competencia de Resolución de Problemas

Se pide definir una estructura que mantenga la información de 5 películas con la valoración acumulada. La estructura que nos proporciona el enunciado del problema ***struct Tinfo*** permite almacenar la información de una película con la valoración acumulada que ha obtenido de los críticos de cine como puede apreciarse.

```
struct Tinfo
{
 char nombrePelicula[MAXNOM];
 int  valoracion;
};
```

Por lo tanto, lo que hay que definir es una lista que nos permita manejar hasta un máximo de 5 películas.

```
#define TOPE 5

struct TLista
{
   struct Tinfo Peliculas[TOPE];
   int          cont;
};
```

Una vez definida esta estructura la vamos a utilizar para almacenar las 5 películas con mejor valoración acumulada de los críticos de cine.

Para saber qué películas son las mejor valoradas hemos de procesar la información de todo el fichero ***Estrenos.txt*** que contiene, en cada línea, la película de estreno junto con las valoraciones obtenidas por parte de los 50 críticos.

En este caso, el proceso consiste en leer en una estructura auxiliar, ***struct Tinfo actual,*** el nombre de la película y, acumular en la valoración las valoraciones de los críticos para esa película. La valoración acumulada se obtiene sumando los 50 valores enteros que aparecen en la línea del fichero a continuación del nombre de la película.

Para resolver el problema hay que declarar una variable de tipo ***struct Tlista lista*** que nos va a permitir almacenar en ella las 5 películas con mejor valoración acumulada.

Se lee del fichero una película y su valoración acumulada en la estructura ***aux*** hasta que no hay más películas. Cada película leída en ***actual*** se trata de insertar en la lista. La inserción de las películas en la ***lista*** se hace en orden decreciente considerando las valoraciones acumuladas. El proceso de lectura del fichero se realiza hasta que se alcanza el fin de fichero.

El proceso de inserción de una película en la ***lista*** es el siguiente:

1. Si es la primera película se inserta directamente.
2. Se busca el lugar en el que hay que insertar la película y se pueden dar los casos siguientes:
    a. Que sea la última en valoración y hay sitio en la lista, en cuyo caso la película se almacena a continuación en la lista.

b. Que su valoración esté entre dos valoraciones ya almacenadas en la lista, en este caso se le hace un hueco desplazando las películas almacenadas y perdiendo la situada en la última posición.
c. Que la valoración de la película sea menor que las 5 previamente almacenadas, en cuyo caso la película no se guarda en la estructura.

Al acabar la lectura del fichero, la lista contiene las 5 películas mejor valoradas por los críticos, y son estas las que se muestran escribiendo por pantalla los títulos y las valoraciones de las películas.