Ejercicios del Tema 7

Ficheros

1. Vectores a columnas

Escribe un programa que almacene 3 vectores de 5 elementos en 3 columnas de un fichero separadas por una tabulación.

```c
#include <stdio.h>
#define N 5

void main ()
{ // Definimos tres vectores cualesquiera
  int v1[N]={-1, 3, 5, 0, 4};
  int v2[N]={4, 9, -8, 2, 3};
  int v3[N]={1, 2, 3, 4, 5};

  int i;
  FILE *pf;
  // Abrimos un fichero para escritura: si ya existe su contenido será
  // borrado
  pf = fopen("columnas.txt", "w");
  // Escribimos línea a línea
  for(i = 0; i < N; i++)
    // Separando por tabuladores, y añadiendo un salto de línea al
    // final
    fprintf(pf, "%d\t%d\t%d\n",
           v1[i], v2[i], v3[i]);
  fclose(pf);
}
```

2. Copia de ficheros

Escribe un programa que copie el contenido de un fichero en otro cambiando los espacios en blanco por un guión bajo _. Emplea el fichero `lorem_ipsum.txt` como ejemplo. (Sugerencia: revisa el ejercicio «Espacios por guiones» del capítulo 4).

```c
#include <stdio.h>
int main ()
{
  FILE *fOrigen, *fDestino;
  char c;

  // Abre ficheros:
  // El origen en modo lectura:
  fOrigen = fopen("lorem_ipsum.txt", "r");
  if (fOrigen == NULL) {
    printf("Error al abrir el archivo origen.\n");
    return -1;
  } else {
    // El destino en modo escritura:
```

```
   fDestino = fopen("lorem_ipsum2.txt", "w");
   if (fDestino == NULL) {
    printf("Error al abrir el archivo destino.\n");
    return -1;
   } else { // Los dos ficheros están correctamente abiertos
    // Leemos el origen caracter por caracter, parando al encontrar
    // el EOF
    while (fscanf(fOrigen, "%c", &c) != EOF)
     {
      if (c == ' ') c = '_'; // Cambiamos espacios por guiones
      fprintf(fDestino, "%c", c); // Escribimos el caracter en el destino
     }
    // Cerramos ficheros
    fclose(fOrigen);
    fclose(fDestino);
    printf("Copia finalizada con exito.\n");
    return 0;
   }
  }
}
```

3. Número de caracteres, palabras y líneas

Escribe un programa que calcule el número de caracteres, palabras y líneas de un fichero de texto. Comprueba su funcionamiento con el fichero `lorem_ipsum.txt`.

```
#include <stdio.h>

int main()
{
 FILE *pf;
 char c;
 // Contadores
 int iChar = 0, iWord = 0, iLine = 0;
 // Abrimos archivo en modo lectura
 pf = fopen("lorem_ipsum.txt", "r");
 if (pf == NULL) {
  printf("Error al abrir el archivo.\n");
  return -1;
 } else {
  // Leemos carácter a carácter hasta encontrar EOF
  while (fscanf(pf, "%c", &c) != EOF)
   {// En cada paso incrementamos el contador de caracteres
    iChar++;
    // El espacio y el salto de línea definen una nueva palabra,
    // luego incrementamos el contador de palabras
    if (c == ' ' || c == '\n')
     iWord++;
    // Si lo leído es un salto de línea, incrementamos su contador
    if (c == '\n')
     iLine++;
   }
  printf("Caracteres: %d \t Palabras: %d \t Líneas: %d.\n",
        iChar, iWord, iLine);

  fclose(pf);
 }
}
```

4. Cuenta letras

Escribe un programa que analice el texto almacenado en un fichero representando el número de vocales que contiene mediante líneas de asteriscos. Este programa debe estar construido de manera similar al ejercicio planteado en el capítulo 4, empleando dos funciones, `cuentaLetra` y `pintaAsteriscos`. Puedes usar el fichero `lorem_ipsum.txt` como ejemplo.

- Versión con asignación dinámica de memoria

```c
#include <stdio.h>
#include <stdlib.h> // malloc y free

// En el capítulo 4, el texto era una variable global. Aquí no, y por
// tanto hay que pasarla como argumento a la función
int cuentaLetra(char x, char texto[]);

void pintaAsteriscos(int n);

int main()
{
  FILE *f;
  char vocales[5] = {'a', 'e', 'i', 'o', 'u'};
  char *texto; //Puntero a una cadena de caracteres
  int fSize; // Tamaño del fichero de texto
  int i = 0, nLetra[5]; //Vector de resultados

  // Abrimos el fichero para leer
  f = fopen("lorem_ipsum.txt", "r");
  if (f == NULL) {
    printf("Error al abrir el archivo origen.\n");
    return -1;
  } else {
    // Calculamos el tamaño del fichero
    fseek(f, 0, SEEK_END);
    fSize = ftell(f);
    fseek(f, 0, SEEK_SET);
    // Reservamos memoria para leer este fichero, y obtenemos un
    // puntero para almacenar los caracteres
    texto = malloc(sizeof(char) * fSize);
    // Recorremos el fichero leyendo caracter a caracter hasta
    // encontrar EOF
    while (fscanf(f, "%c", &texto[i]) != EOF)
       i++;
    // Cerramos el fichero tras terminar la lectura
    fclose(f);

  }
  // Cuenta las apariciones de cada letra, y almacena resultados en un
  // vector
  for (i = 0; i < 5; i++)
    nLetra[i] = cuentaLetra(vocales[i], texto);
  for (i = 0; i < 5; i++)
    { // Recorre el vector de resultados, y pinta asteriscos
      // correspondientes a cada letra
      printf("%c: ", vocales[i]);
      pintaAsteriscos(nLetra[i]);
      printf("\n");
    }
  // Liberamos la memoria asociada a la cadena de caracteres
  free(texto);
  return 0;
}
```

```
int cuentaLetra(char x, char texto[])
{
  int i = 0, n = 0;

  while (texto[i] != '\0')
    {
      if (texto[i] == x)
        n++; // Contador de apariciones
      i++;
    }
  return n;
}

void pintaAsteriscos(int n)
{
  int i;
  if (n > 0)
    for (i = 1; i <= n; i++)
      printf("*");
}
```

- Versión sin asignación dinámica de memoria (procesando carácter a carácter)

```
#include <stdio.h>

void pintaAsteriscos(int n);

int main()
{
  FILE *f;
  char vocales[5] = {'a', 'e', 'i', 'o', 'u'};
  char c;
  int i, nLetra[5] = {0, 0, 0, 0, 0}; //Vector de resultados

  // Abrimos el fichero para leer
  f = fopen("lorem_ipsum2.txt", "r");
  if (f == NULL) {
    printf("Error al abrir el archivo origen.\n");
    return -1;
  } else {
    // Recorremos el fichero hasta encontrar EOF
    while (fscanf(f, "%c", &c) != EOF)
      {
        // Cuenta las apariciones de cada letra, y almacena resultados en un
        // vector
        for (i = 0; i < 5; i++)
          {
            if (c == vocales[i])
               ++nLetra[i];
          }
      }
    // Cerramos el fichero tras terminar la lectura
    fclose(f);

  for (i = 0; i < 5; i++)
    { // Recorre el vector de resultados, y pinta asteriscos
      // correspondientes a cada letra
      printf("%c: ", vocales[i]);
      pintaAsteriscos(nLetra[i]);
      printf("\n");
    }
  return 0;
  }
```

```
}

void pintaAsteriscos(int n)
{
  int i;
  if (n > 0)
    for (i = 1; i <= n; i++)
      printf("*");
}
```

5. Máximo, mínimo y promedio de variables

El fichero `aranjuez.csv` es un fichero que almacena valores numéricos en formato CSV (valores separados por comas). Contiene 4 columnas (variables) y 2898 filas (registros). Cada fila corresponde a un valor diario de una variable meteorológica registrada en la estación localizada en Aranjuez. Estas variables son: temperatura ambiente, humedad, velocidad del viento, radiación solar.

Escribe un programa que lea este fichero almacenando el contenido de cada columna en un vector. A continuación, el programa debe calcular el valor máximo, mínimo y promedio de cada vector empleando una función separada para cada cálculo (*revisa el ejercicio «Máximo, mínimo y promedio de una colección de números» del capítulo 4*). Finalmente, el programa mostrará el resultado en pantalla y escribirá estos cálculos en un fichero con un formato similar al siguiente:

Variable	Max	Min	Media
Temp	XX	XX	XX
Humedad	XX	XX	XX
Viento	XX	XX	XX
Rad	XX	XX	XX

Ampliación: escribe una versión de este programa que realice la misma tarea pero sin conocer a priori el número de filas del fichero.

```
#include <stdio.h>
#define N 2898

double maximo(double x[], int n);
double minimo(double x[], int n);
double promedio(double x[], int n);

int main()
{
  FILE *pf, *wf;
  int i;
  // Defino un vector para cada variable a leer del fichero
  double TempAvg[N], HumidAvg[N], WindAvg[N],Radiation[N];
  double minTa, minH, minW, minR;
  double maxTa, maxH, maxW, maxR;
  double promTa, promH, promW, promR;

  pf = fopen("aranjuez.csv", "r");
  if (pf == NULL)
    {
      printf("Error al abrir el fichero");
      return -1;
    }
  else
    { // Leemos el fichero línea a línea
      for (i = 0; i < N; i++)
        { // El formato es de números separados por comas
          fscanf(pf, "%lf,%lf,%lf,%lf",
                 &TempAvg[i],
```

```
            &HumidAvg[i],
            &WindAvg[i],
            &Radiation[i]);
     }
    // Cerramos el fichero
    fclose(pf);
    // Calcula el minimo, maximo y promedio usando las funciones
    minTa = minimo(TempAvg, N);
    maxTa = maximo(TempAvg, N);
    promTa = promedio(TempAvg, N);

    minH = minimo(HumidAvg, N);
    maxH = maximo(HumidAvg, N);
    promH = promedio(HumidAvg, N);

    minW = minimo(WindAvg, N);
    maxW = maximo(WindAvg, N);
    promW = promedio(WindAvg, N);

    minR = minimo(Radiation, N);
    maxR = maximo(Radiation, N);
    promR = promedio(Radiation, N);

    // Mostramos resultados
    printf("Variable: \t Max \t Min \t Prom \n");
    printf("Temp: \t %lf \t %lf \t %lf \n",
          minTa, maxTa, promTa);
    printf("Humedad: \t %lf \t %lf \t %lf \n",
          minH, maxH, promH);
    printf("Viento: \t %lf \t %lf \t %lf \n",
          minW, maxW, promW);
    printf("Radiacion: \t %lf \t %lf \t %lf \n",
          minR, maxR, promR);

    // Escribimos fichero
    wf = fopen("resumen.txt", "w");

    fprintf(wf, "Variable: \t Max \t Min \t Prom \n");
    fprintf(wf, "Temp: \t %lf \t %lf \t %lf \n",
          minTa, maxTa, promTa);
    fprintf(wf, "Humedad: \t %lf \t %lf \t %lf \n",
          minH, maxH, promH);
    fprintf(wf, "Viento: \t %lf \t %lf \t %lf \n",
          minW, maxW, promW);
    fprintf(wf, "Radiacion: \t %lf \t %lf \t %lf \n",
          minR, maxR, promR);
    fclose(wf);

    return 0;
   }
}

double maximo(double x[], int n)
{
  int i;
  double mx = x[0]; //El maximo inicial es el primer elemento
  // Busca en el vector algún valor mayor
  for (i = 1; i < n; i++)
    if (x[i] > mx) mx = x[i]; // y lo sustituye si lo encuentra
  return mx;
}

double minimo(double x[], int n)
```

```
{
  int i;
  double mn = x[0];

  for (i = 1; i < n; i++)
    if (x[i] < mn) mn = x[i];
  return mn;
}

double promedio(double x[], int n)
{
  int i;
  double media = 0.0;

  for (i = 0; i < n; i++)
    media += x[i];
  return media / n;
}
```