Fundamentos de Programación (1º ASI)
Ejercicios Resueltos del Tema 4 - Archivos de texto

4.2. Leer un texto por teclado y descomponerlo en palabras, guardando el resultado en un archivo de texto (una palabra en cada línea)

```
int main(void)
{
      int i;
      FILE *fich;
      char txt[500];

      printf("Este programa lee un texto por teclado y lo envía a un archivo de texto\n");
      printf("sustituyendo todos los espacios por saltos de línea.\n\n");

      printf("Introduzca un texto: ");
      gets(txt);

      // Abrir el archivo para escritura
      fich = fopen("quitar_espacios.txt", "wt");
      if (fich == NULL) {
            printf("Error al abrir el archivo");
            exit(1);
      }

      // Recorremos la cadena carácter por carácter
      for (i=0; i<strlen(txt); i++)
      {
            if (txt[i] == ' ')       // Si el carácter es un espacio...
                  fputc('\n', fich);  // ... escribimos un salto de línea
            else                     // Si es cualquier otro carácter...
                  fputc(txt[i], fich);       // ... escribimos ese carácter
      }
      fclose(fich);

      printf("Se ha generado el archivo quitar_espacios.txt\n\n");

      return 0;
}
```

4.3. Leer el contenido del archivo de texto del ejercicio anterior y mostrarlo por la pantalla recomponiendo su aspecto original (es decir, volviendo a sustituir los saltos de línea por espacios)

```
int main(void)
{
      FILE *fich;
      char car;

      printf("\nEste programa lee un archivo de texto y lo muestra en la pantalla\n");
      printf("sustituyendo los saltos de línea por espacios.\n\n");

      // Abrir el archivo para lectura
      printf("Mostrando el contenido del archivo quitar_espacios.txt:\n\n");
      fich = fopen("quitar_espacios.txt", "rt");
      if (fich == NULL) {
            printf("Error al abrir el archivo");
            exit(1);
      }

      // Recorremos el archivo carácter por carácter
      while (!feof(fich))
      {
```

```
            car = fgetc(fich);        // Leemos un carácter del archivo
            if (car == '\n')          // Si el carácter es un salto de línea...
                  printf(" ");        // ... escribimos un espacio en la pantalla
            else                      // Si es cualquier otro carácter...
                  printf("%c", car);  // ... escribimos ese carácter en la pantalla
      }
      fclose(fich);

      printf("\n\nSe ha terminado de procesar el archivo\n\n");

      return 0;
}
```

4.4. Concatenar dos archivos de texto.

```
int main(void)
{
      FILE *f1, *f2, *fdest;
      char arch1[50], arch2[50], arch_dest[50];
      char car;

      printf("\nEste programa concatena dos archivos de texto\n\n");

      // Leemos los nombres de los archivos
      printf("Archivos de ORIGEN:\n");
      printf("  Introduzca el nombre del primer archivo: ");
      gets(arch1);
      printf("  Introduzca el nombre del segundo archivo: ");
      gets(arch2);
      printf("Archivo de DESTINO:\n");
      printf("  Introduzca el nombre del archivo donde se realizará la concatenación: ");
      gets(arch_dest);

      // Abrimos el primer archivo de origen (lectura) y el de destino (escritura)
      f1 = fopen(arch1, "rt");
      fdest = fopen(arch_dest, "wt");
      if ((f1 == NULL) || (fdest == NULL))
      {
            if (f1 == NULL) printf("Error al abrir el archivo %s\n", arch1);
            if (fdest == NULL) printf("Error al abrir el archivo %s\n", arch_dest);
            exit(1);
      }

      // Recorremos el archivo 1 carácter por carácter, enviándolos todos al archivo de destino
      while (!feof(f1))
      {
            car = fgetc(f1);
            if (car != EOF) fputc(car, fdest);
      }
      // Cerramos el archivo 1
      fclose(f1);

      // Abrimos el segundo archivo de origen (lectura) SIN CERRAR el archivo de destino
      f2 = fopen(arch2, "rt");
      if (f2 == NULL)
      {
            printf("Error al abrir el archivo %s\n", arch2);
            exit(1);
      }

      // Recorremos el archivo 2 carácter por carácter, enviándolos todos al archivo de destino
      while (!feof(f2))
      {
            car = fgetc(f2);
            if (car != EOF) fputc(car, fdest);
      }
      // Cerramos el archivo 2 y el archivo de destino
```

```
      fclose(f2);
      fclose(fdest);

      printf("El archivo %s se ha generado correctamente\n\n", arch_dest);

      return 0;
}
```

4.5. Calcular la frecuencia de aparición de letras en un archivo de texto cualquiera.

```
int main(void)
{
   FILE* f;
   char nombre_arch[50], c;
   int cont[29];                    // Contadores para las letras
   int total, otros, i;

   printf("\nEste programa cuenta la frecuencia de aparción de letras en un archivo\n\n");
   printf("Nombre del archivo: ");
   gets(nombre_arch);

   // Inicializamos los contadores
   total = 0;
   otros = 0;
   for (i=0; i<=28; i++)
      cont[i] = 0;

   // Abrimos el archivo
   f = fopen(nombre_arch, "rt");
   if (f == NULL)
   {
         printf("Error al abrir el archivo %s\n", nombre_arch);
         system("pause");
         exit(1);
   }

   printf("Procesando el archivo...\n");

   // Recorremos el archivo
   while (!feof(f))
   {
      c = fgetc(f);                       // Leemos un carácter
      total++;
      if ((c >= 'A') && (c <= 'Z'))       // Contabilizamos una letra mayúscula
            cont[c-65] = cont[c-65] + 1;  // (el código ASCII de la 'A' es 65)
      else if ((c >= 'a') && (c <= 'z'))  // Contabilizamos una letra minúscula
            cont[c-97] = cont[c-97] + 1;  // (el código ASCII de la 'a' es 97)
      else if ((c == 'ñ') || (c == 'Ñ'))  // Contabilizamos una eñe
            cont[28] = cont[28] + 1;
      else                                // Contabilizamos "otros" caracteres
            otros++;
   }

   // Mostramos resultados
   printf("\nRECUENTO FINALIZADO\n\n");
   printf("Total de caracteres en el archivo: %i\n\n", total);
   printf("Letra     Num. Apariciones    Frecuencia\n");
   printf("---------------------------------------\n");
   for (i=0; i<26; i++)
   {
      printf("  %c           %5i              %2.2f\n", i+65, cont[i], (float)cont[i]/total*100);
   }
   printf("  Ñ           %5i              %2.2f\n", cont[28], (float)cont[28]/total*100);
   printf("Otros         %5i             %2.2f\n", otros, (float)otros/total*100);

   return 0;
}
```

4.6. Mostrar sólo las líneas de un archivo de texto que contengan una palabra dada por el usuario.

```
int main(void)
{
      FILE *f;
      char arch[50], palabra[50], linea[1000], *aux;
      int cont = 0, cont_lin = 0;

      printf("\nEste programa abre un archivo de texto y muestra por la pantalla\n");
      printf("sólo las líneas que contengan una palabra introducida por el usuario\n\n");

      // Leemos los nombres de los archivos
      printf("Introduzca el nombre del archivo: ");
      gets(arch);
      printf("Introduzca la palabra buscada: ");
      gets(palabra);

      // Abrimos el archivo de origen (modo lectura)
      f = fopen(arch, "rt");
      if (f == NULL)
      {
            printf("Error al abrir el archivo %s\n", arch);
            exit(1);
      }

      // Recorremos el archivo línea por línea
      printf("Procesando el archivo...\n\n");
      while (!feof(f))
      {
            fgets(linea, 999, f);           // Leemos una línea
            cont_lin++;
            aux = strstr(linea, palabra);   // Comprobamos si "palabra" está contenida en "línea"
            if (aux != NULL) {              // ¡Sí que lo está!
                  // Mostramos la línea en la pantalla
                  printf("Línea %i ---> %s", cont_lin, linea);
                  cont++;
            }
      }
      fclose(f);

      printf("\n\nFin del archivo\nLa palabra '%s' se ha encontrado %i veces.\n\n", palabra, cont);

      return 0;
}
```

4.7. (Primera parte) Leer 5 números enteros por teclado y calcular la suma y la media de ellos, escribiéndolo todo en un archivo de texto con este formato:

```
Los números son:
5
4
5
4
2
La suma es: 20
La media es: 4.0
```

```
int main()
{
      int i, suma, n[5];
      float media;
      FILE *fich;

      // Parte 1: introducir datos por teclado
      printf("Introduzca cinco números enteros");
```

```
     suma = 0;
     for (i=0; i<5; i++) {
          scanf("%i", &n[i]);
          suma = suma + n[i];
     }
     media = (float)suma / 5;

     // Parte 2: abrir el archivo en modo "w"
     printf("Creando el archivo 4-07.txt...\n");
     fich = fopen("6-04b.txt", "wt");
     if (fich == NULL) {
          printf("Error al abrir el archivo para escritura");
          exit(1);
     }

     // Parte 3: escribir los datos en el archivo
     fprintf(fich, "Los números son:\n");
     for (i=0; i<5; i++) {
          fprintf(fich, "%i\n", n[i]);
     }
     fprintf(fich, "La suma es: %i\n", suma);
     fprintf(fich, "La media es: %2.2f\n", media);

     printf("Archivo escrito con éxito\n");
     fclose(fich);

     return 0;
}
```

4.7. (Segunda parte) Leer el archivo generado en el ejercicio anterior y mostrar su contenido por la pantalla con este formato:

```
5 - 4 - 5 - 4 - 2 - 20 - 4.0
```

```
int main(void)
{
     int i, suma, n[5];
     float media;
     FILE *fich;
     char txt[50];

     // Parte 1: abrir el archivo para lectura
     printf("Abriendo el archivo 4-07.txt...");
     fich = fopen("4-07.txt", "rt");
     if (fich == NULL) {
          printf("Error al abrir el archivo para lectura");
          exit(1);
     }

     // Parte 2: leer los datos del archivo
     fgets(txt, 49, fich);              // Lee el texto "Los números son" (incluido "\n")
     for (i=0; i<5; i++) {
          fscanf(fich, "%i\n", &n[i]);  // Lee los números (cada uno seguido de "\n")
     }
     fgets(txt, 12, fich);              // Lee el texto "La suma es" (12 caracteres)
     fscanf(fich, "%i\n", &suma);       // Lee el valor de la suma (seguido de "\n")
     fgets(txt, 13, fich);              // Lee el texto "La media es" (13 caracteres)
     fscanf(fich, "%f", &media);        // Lee el valor de la media
     fclose(fich);
```

```
     // Parte 3: mostrar los datos en la pantalla
     printf("Los datos leídos del archivo son:\n");
     for (i=0; i<4; i++) {
          printf("%i - ", n[i]);
     }
     printf("%i - %f\n", suma, media);

     return 0;
}
```

4.8. Generar un archivo HTML simple que sea correctamente visualizable en un navegador web.

```
#define ARCHIVO "página_web.html"

int main(void)
{
   FILE* f;
   char cad[2000], cad2[100], c;
   int terminar;

   printf("Programa para generar un documento HTML sencillo\n\n\n");

   f = fopen(ARCHIVO, "wt");
   if (f == NULL)
   {
         printf("No puedo crear el archivo HTML\n");
         exit(-1);
   }

   // Encabezado
   printf("ENCABEZADO:\n");
   fputs("<html>\n<head>\n<title>Ejercicio de Fund.Prog.</title>\n</head>\n<body>\n", f);
   // Atributos del encabezado
   printf("Introduzca los atributos del encabezado: \n");
   printf("Color: "); gets(cad2);
   fprintf(f, "<font color = '%s'>", cad2);
   printf("Tamaño: "); gets(cad2);
   fprintf(f, "<font size = '%s'>", cad2);
   printf("Tipo de letra: "); gets(cad2);
   fprintf(f, "<font face = '%s'>", cad2);
   printf("Modificador (N = negrita, K = cursiva, S = subrayado): ");
   getchar(c);
   if (c == 'N') fputs("<b>", f);
   if (c == 'K') fputs("<i>", f);
   if (c == 'S') fputs("<u>", f);

   // Texto del encabezado
   printf("Introduzca el texto del encabezado: ");
   gets(cad);
   fputs(cad, f);            // Envía el texto del encabezado al archivo
   fputs("<br>", f);         // Envía un salto de línea

   // Quitamos los atributos del encabezado
   if (c == 'N') fputs("</b>", f);
   if (c == 'K') fputs("</i>", f);
   if (c == 'S') fputs("</u>", f);
   fputs("</font></font></font>\n", f);

   // Repetimos la jugada con el cuerpo de la página
   printf("CUERPO:\n");
```

```
   // Atributos del cuerpo
   printf("Introduzca los atributos del cuerpo: \n");
   printf("Color: "); gets(cad2);
   fprintf(f, "<font color = '%s'>", cad2);
   printf("Tamaño: "); gets(cad2);
   fprintf(f, "<font size = '%s'>", cad2);
   printf("Tipo de letra: "); gets(cad2);
   fprintf(f, "<font face = '%s'>", cad2);
   printf("Modificador (N = negrita, K = cursiva, S = subrayado): ");
   getchar(c);
   if (c == 'N') fputs("<b>", f);
   if (c == 'K') fputs("<i>", f);
   if (c == 'S') fputs("<u>", f);

   // Texto del cuerpo. Puede tener varias líneas. Terminaremos al teclear "fin"
   printf("Introduzca el texto del cuerpo (teclee fin para terminar):\n");
   terminar = 0;
   while (terminar == 0)
   {
         gets(cad);
         if (strcmp(cad, "fin") != 0)
         {
            fputs (cad, f);     // Envía el texto al archivo
            fputs ("<br>", f);  // Envía un salto de línea
         }
         else
            terminar = 1;
   }

   // Quitamos los atributos del cuerpo
   if (c == 'N') fputs("</b>", f);
   if (c == 'K') fputs("</i>", f);
   if (c == 'S') fputs("</u>", f);
   fputs("</font></font></font>\n", f);
   fputs("</body></html>\n", f);


   fclose(f);

   printf("El documento HTML ha sido generado correctamente.\n");
   return 0;
}
```

4.9. (Primera parte) Encriptar un archivo de texto mediante el método César.

```
char encriptar(char c, int clave);  // Prototipo de la función

int main(void)
{
   FILE *f, *fd;
   char arch[50], arch_dest[50], aux[50];
   int clave;
   char c, c_enc;

   // Leemos los nombres de los archivos
   printf("\nEste programa encripta archivos de texto mediante el Método César\n\n");
   printf("Nombre del archivo a encriptar: ");
   gets(arch);
   strcpy(arch_dest, arch);  // El archivo de destino tendrá el mismo nombre que el original...
   strcat(arch_dest, ".cfr");// ...añadiendo la extensión ".cfr"

   // Leemos la clave de encriptación
```

```
   printf("Clave de encriptación: ");
   gets(aux);
   clave = atoi(aux);

   // Abrimos el archivo de entrada en modo lectura
   // y el de salida en modo escritura
   f = fopen(arch, "rt");
   fd = fopen(arch_dest, "wt");
   if ((f == NULL) || (fd == NULL))
   {
           if (f == NULL) printf("Error al abrir el archivo %s\n", arch);
           if (fd == NULL) printf("Error al abrir el archivo %s\n", arch_dest);
           exit(1);
   }

   printf("Encriptando el archivo...\n");

   // Recorremos el archivo de origen carácter por carácter
   while (!feof(f))
   {
      c = fgetc(f);                    // Leemos un carácter
      c_enc = encriptar(c, clave);     // Encriptamos ese carácter
      fputc(c_enc, fd);                // Escribimos el carácter encriptado
   }

   printf("El archivo %s ha sido generado correctamente\n", arch_dest);

   return 0;
}


// Esta función recibe un carácter y una clave de encriptación, y devuelve el carácter
// encriptado según el método César. Para variar el método de encriptación, basta con
// modificar esta función (el resto del programa puede seguir igual)
char encriptar(char c, int clave)
{
      char cifr;
      int n;

      // Primero comprobaremos que el carácter es alfabético. Sólo encriptaremos
      // caracteres alfabéticos (aunque esto se puede cambiar si se quiere)
      if ((c >= 'A') && (c <= 'Z'))  // Mayúsculas (hay que encriptar)
      {
            n = c + clave;
            if (n > 'Z') {
                  n = n - 'A';
                  n = (n % 26) + 'A';
            }
            cifr = n;
      }
      else if  ((c >= 'a') && (c <= 'z'))   // Minúsculas (encriptar)
      {
            n = c + clave;
            if (n > 'z') {
                  n = n - 'a';
                  n = (n % 26) + 'a';
            }
            cifr = n;
      }
      else  // Otros caracteres (no hay que encriptar)
      {
            cifr = c;  // Lo dejamos igual
      }

      return cifr;
}
```

4.9. (Segunda parte) Desencriptar el archivo de texto del ejercicio anterior y mostrarlo por la pantalla.

```
char desencriptar(char c, int clave);


int main(void)
{
   FILE *f, *fd;
   char arch[50], aux[50];
   int clave;
   char c, c_des;

   // Leemos los nombres de los archivos
   printf("\nEste programa desencripta archivos de texto mediante el Método César\n\n");
   printf("Nombre del archivo encriptado: ");
   gets(arch);

   // Leemos la clave de encriptación
   printf("Clave de desencriptación: ");
   gets(aux);
   clave = atoi(aux);

   // Abrimos el archivo en modo lectura
   f = fopen(arch, "rt");
   if (f == NULL)
   {
         printf("Error al abrir el archivo %s\n", arch);
         exit(1);
   }

   printf("Desencriptando el archivo...\n");

   // Recorremos el archivo carácter por carácter
   while (!feof(f))
   {
      c = fgetc(f);                    // Leemos un carácter
      c_des = desencriptar(c, clave);        // Desencriptamos ese carácter
      printf("%c", c_des);            // Mostramos en la pantalla el carácter desencriptado
   }

   printf("\n\nFin del archivo\n\n");

   return 0;
}


// Esta función recibe un carácter encriptado y una clave de desencriptación, y devuelve el
carácter
// desencriptado según el método César. Para variar el método de desencriptación, basta con
// modificar esta función (el resto del programa puede seguir igual)
char desencriptar(char c, int clave)
{
      char cifr;
      int n;

      // Primero comprobaremos que el carácter es alfabético. Sólo desencriptaremos
      // caracteres alfabéticos (aunque esto se puede cambiar si se quiere)
      if ((c >= 'A') && (c <= 'Z'))  // Mayúsculas (hay que desencriptar)
      {
            n = c - clave;
            if (n < 'A') {
                  n = n - 'Z';
                  n = (n % 26) + 'Z';
            }
            cifr = n;
      }
      else if  ((c >= 'a') && (c <= 'z'))   // Minúsculas (desencriptar)
```

```
	{
		n = c - clave;
		if (n < 'a') {
			n = n - 'z';
			n = (n % 26) + 'z';
		}
		cifr = n;
	}
	else  // Otros caracteres (no hay que desencriptar)
	{
		cifr = c;  // Lo dejamos igual
	}

	return cifr;
}
```

Ejercicio extra 1. Leer el archivo de texto de la biblioteca y cargarlo en un vector, mostrando su contenido por la pantalla.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define LONG_LIN 500
#define MAX_LIBROS 1600

struct s_libro {
	char autor[60];
	char titulo[100];
	char editorial[50];
	char cdu[20];
};


int main(void)
{
	struct s_libro biblioteca[MAX_LIBROS];
	char linea[LONG_LIN];
	int i, j, k, pos, campo;
	FILE *f;

	// ********** ABRIMOS EL ARCHIVO **************
	f = fopen("libros.txt", "rt");
	if (f == NULL) { printf("\n\n** Error al abrir el archivo libros.txt **\n\n"); exit(1); }

	// ******** PROCESAMOS EL ARCHIVO, METIENDO LOS DATOS EN EL VECTOR ********
	pos = 0;
	while (!feof(f))
	{
		// Leemos una línea del archivo de texto
		printf("Procesando el archivo... %i%%", pos * 100 / MAX_LIBROS);
		printf("%c", 13);
		for (k = 0; k < 1000000; k++);
		fgets(linea, LONG_LIN-1, f);
		if (strlen(linea) >= LONG_LIN-1)
			printf("** AVISO: ¡la línea %i es demasiado larga! **\n", pos + 1);

		// Procesamos la línea del archivo de texto
		j = 0;
		campo = 0;
		for (i = 0; i <= strlen(linea); i++)
		{
			if ((linea[i] != '*') && (linea[i] != '\0'))
			{
				switch(campo)
				{
					case 0: // Apellidos del autor
```

```
                                    biblioteca[pos].autor[j] = linea[i]; break;
                              case 1: // Nombre del autor
                                    biblioteca[pos].autor[j] = linea[i]; break;
                              case 2: // Titulo del libro
                                    biblioteca[pos].titulo[j] = linea[i]; break;
                              case 3: // Editorial
                                    biblioteca[pos].editorial[j] = linea[i]; break;
                              case 4: // Código CDU
                                    biblioteca[pos].cdu[j] = linea[i]; break;
                        }
                        j++;
                  }
                  else  // Se ha leido '*' o '\0', es decir, un campo acaba de terminar
                  {
                        switch(campo)
                        {
                              case 0: // Apellidos del autor
                                    biblioteca[pos].autor[j] = ' '; j++; break;
                              case 1: // Nombre del autor
                                    biblioteca[pos].autor[j] = '\0'; j = 0; break;
                              case 2: // Titulo del libro
                                    biblioteca[pos].titulo[j] = '\0'; j = 0; break;
                              case 3: // Editorial
                                    biblioteca[pos].editorial[j] = '\0'; j = 0; break;
                              case 4: // Código CDU
                                    biblioteca[pos].cdu[j] = '\0'; j = 0; break;
                        }
                        campo++;      // Pasamos al siguiente campo
                  }
            }
            pos++;
      }

      fclose(f);
      printf("Fin de la lectura del archivo\nPulse Intro para mostrar los datos.");
      getchar();

      // ************ MOSTRAMOS EL CONTENIDO DEL VECTOR EN LA PANTALLA ************
      printf("\n\n");
      printf("AUTOR                     TITULO                        EDITORIAL          CDU\n");
     printf("-------------------------------------------------------------------------------\n");
      for (i = 0; i < pos; i++)
            printf("%-30s %-30s %-15s %-10s\n",
            biblioteca[i].autor, biblioteca[i].titulo, biblioteca[i].editorial, biblioteca[i].cdu);

      return 0;
}
```

Ejercicio extra 2. Escribe un programa que abra un archivo de texto (cuyo nombre debe teclear el usuario por teclado) y sustituya todas las vocales por asteriscos. Haz dos versiones del programa: una que emplee exclusivamente acceso secuencial y otra que utilice acceso directo.

VERSIÓN 1: CON ACCESO SECUENCIAL

```
int main(void)
{
      FILE *f, *f_dest;
      char nom_arch[100], c;

      printf("Introduce el nombre del fichero: ");
      gets(nom_arch);

      // Abrimos el archivo de origen
      f = fopen(nom_arch, "rt");
      if ( f == NULL ) { printf("Error al abrir el archivo %s\n", nom_arch); exit(1); }
```

```
      // Creamos un archivo temporal donde iremos construyendo el archivo de destino
      f_dest = fopen("temporal", "wt");
      if ( f_dest == NULL ) { printf("Error al crear archivo temporal\n"); exit(1); }

      // Recorremos el archivo
      while (!feof(f))
      {
            c = fgetc(f);     // Leemos un carácter del archivo
            if (c != EOF)
            {
                  if ((c == 'a') || (c == 'e') || (c == 'i') || (c == 'o') || (c == 'u') ||
                      (c == 'A') || (c == 'E') || (c == 'I') || (c == 'O') || (c == 'U'))
                        // Si en el arch. original hay una vocal, escribimos '*' en el arch. nuevo
                        fputc('*', f_dest);
                  else
                        // Si hay cualquier otro carácter, lo copiamos igual al archivo nuevo
                        fputc(c, f_dest);
            }
      }
      fclose(f); fclose(f_dest);

      // Borramos el archivo de origen y renombramos el temporal
      remove(nom_arch);
      rename("temporal", nom_arch);
      printf("Se han sustituido las vocales por asteriscos con éxito\n");

      return 0;
}
```

VERSIÓN 2: CON ACCESO DIRECTO

```
int main(void)
{
      FILE *f;
      char nom_arch[100], c;

      printf("Introduce el nombre del fichero: ");
      gets(nom_arch);

      // Abrimos el archivo de origen
      f = fopen(nom_arch, "r+t");
      if ( f == NULL ) { printf("Error al abrir el archivo %s\n", nom_arch); exit(1); }

      // Recorremos el archivo
      while (!feof(f))
      {
            c = fgetc(f);     // Leemos un carácter del archivo
            if (c != EOF)
            {
                  if ((c == 'a') || (c == 'e') || (c == 'i') || (c == 'o') || (c == 'u') ||
                      (c == 'A') || (c == 'E') || (c == 'I') || (c == 'O') || (c == 'U'))
                  {
                        // Hemos leido una vocal. Vamos a retroceder y a sobreescribirla con un '*'
                        fseek(f, -1, SEEK_CUR);
                        fputc('*', f);
                  }
            }
      }
      fclose(f);
      printf("Se han sustituido las vocales por asteriscos con éxito\n");

      return 0;
}
```