APRENDERAPROGRAMAR.COM

LEER RECUPERAR EXTRAER DATOS DE FICHEROS O ARCHIVOS EN C. FGETC, GETC, FGETS, FSCANF. EJERCICIOS (CU00538F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

Fecha revisión: 2031

Resumen: Entrega nº38 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

LEER ARCHIVOS EN LENGUAJE C

Ya hemos estudiado cómo abrimos y cerramos la comunicación con un archivo. Ahora nos estamos centrando en ver cómo accedemos a los datos. Para ello nos valemos de las funciones disponibles. En el caso de lectura o recuperación de información desde archivos tenemos las siguientes.

Para la recuperación de datos desde archivos tenemos las siguientes funciones disponibles.

Función o macro	Significado y ejemplo aprenderaprogramar.com
fgetc	Recupera un carácter del archivo fgetc (nombreInternoFichero);
getc	Igual que fgetc
fgets (arg1, arg2, arg3)	Recupera el contenido del archivo en la variable arg1 hasta que encuentra o bien un carácter de fin de línea (\n) o bien hasta extraer arg2-1 caracteres siendo arg2 un valor especificado en la llamada de la función y arg3 el nombre interno del fichero. Ejemplo: fgets (cadena1, n, nombreInternoFichero);
fscanf (arg1, "%format1 %format2 ...", &var1, &var2 ...)	Recupera el contenido del archivo con nombre interno arg1 (hasta el primer espacio en blanco) en las variables var1, var2 ... con los formatos %format1, %format2... Ejemplo: fscanf (fichero, "%c", &cadena1[i]); Ejemplo: fscanf (fichero, "%d %d", &numero1, &numero2);

Tener en cuenta que los datos extraídos pueden ser mostrados directamente en pantalla (p.ej. usando *printf*), o almacenados en una variable cuyo contenido representa lo que hemos leído del fichero.

Veamos varios ejemplos de código que logran el mismo resultado usando *fgetc*, *fgets* y *fscanf*. Escribe el código y comprueba los resultados. Antes de ejecutar debes crear un archivo de nombre *cursoAF1.txt* en el directorio donde tengas el proyecto de Code::Blocks. Nosotros hemos trabajado con un archivo que tenía 3 líneas, concretamente estas:

```
Aprender a programar (linea 1)

requiere esfuerzo (linea 2)

y dedicacion (linea 3)
```

© aprenderaprogramar.com, 2006-2031

Los programas que presentamos extraen exactamente el contenido de 3 líneas, de modo que si existen más líneas no son extraídas. Veremos más adelante cómo se puede explorar un fichero y extraer reiteradamente contenidos hasta llegar al final del fichero sin conocer de antemano cuántas líneas o qué extensión tiene.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define LIMITE 50
// Ejemplo curso C aprenderaprogramar.com
int main() {
   char cadena1 [LIMITE];    char cadena2 [LIMITE];
   char cadena3 [LIMITE];
   FILE* fichero;
   fichero = fopen("cursoAF1.txt", "rt");
   fgets (cadena1, LIMITE, fichero);
   fgets (cadena2, LIMITE, fichero);
   fgets (cadena3, LIMITE, fichero);
   fclose(fichero);
   puts ("Extraido de fichero lo siguiente: \n");
   puts (cadena1); puts (cadena2); puts (cadena3);
   puts("Proceso completado");
   return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define LIMITE 50
// Ejemplo aprenderaprogramar.com
int main() {
   int i;
   char cadena1 [LIMITE]; char cadena2 [LIMITE];
   char cadena3 [LIMITE];
   FILE* fichero;
   fichero = fopen("cursoAF1.txt", "rt");
   i=0;
   do{ fscanf (fichero, "%c", &cadena1[i]); i++;
   } while (cadena1[i-1]>=32 && cadena1[i-1]<=126);
   cadena1[i]='\0';
   i=0;
   do{ fscanf (fichero, "%c", &cadena2[i]); i++;
   } while (cadena2[i-1]>=32 && cadena2[i-1]<=126);
   cadena2[i]='\0';
   i=0;
   do{ fscanf (fichero, "%c", &cadena3[i]); i++;
   } while (cadena3[i-1]>=32 && cadena3[i-1]<=126);
   cadena3[i]='\0';
   fclose(fichero);
   puts ("Extraido de fichero lo siguiente: \n");
   puts (cadena1); puts (cadena2); puts (cadena3);
   puts("Proceso completado");
   return 0;
}
```

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

int main() {
   char cadena1 [50];
   char cadena2 [50];
   char cadena3 [50];
   char control= 'a';
   int i;
   FILE* fichero;
   fichero = fopen("cursoAF1.txt", "rt");

   i=1;
   while (control >=32 && control<=126)  {
      control = fgetc(fichero);
      cadena1[i]='\0';
      if (control >=32 && control<=126)
      {cadena1[i-1] = control;}
      i++;
   }

   i=1;
   control= 'a';
   while (control >=32 && control<=126)  {
      control = fgetc(fichero);
      cadena2[i]='\0';
      if (control >=32 && control<=126)
     {cadena2[i-1] = control;}
      i++;
   }

   i=1;
   control= 'a';
   while (control >=32 && control<=126)  {
      control = fgetc(fichero);
      cadena3[i]='\0';
      if (control >=32 && control<=126)
     {cadena3[i-1] = control;}
      i++;
   }

   fclose(fichero);
   printf ("Extraido de fichero lo siguiente: \n");
   printf ("cadena1: %s \n", cadena1);
   printf ("cadena2: %s \n", cadena2);
   printf ("cadena3: %s \n", cadena3);
   printf("Proceso completado");

   return 0; // Ejemplo aprenderaprogramar.com
}
```

© aprenderaprogramar.com, 2006-2031

Algunos comentarios aclaratorios de los programas anteriores son: hemos usado expresiones como *control >=32 && control<=126* para localizar los caracteres imprimibles (letras y caracteres comunes) dentro del archivo. En concreto en el juego de caracteres ASCII los caracteres imprimibles están asociados a los números 32 a 126. Un carácter no imprimible como un salto de línea está fuera de este rango. Analizando estos valores podemos detectar por ejemplo un salto de línea. Quedan sin embargo fuera del rango de caracteres las letras con tildes, por lo que esto no es una solución "ideal" para tratamiento de texto, únicamente lo ponemos aquí a modo de ejemplo relativo a las distintas posibilidades que se nos abren para el manejo de archivos.

Hemos usado expresiones como *cadena1[i]='\0';* para añadir el carácter no imprimible de fin de cadena a una cadena extraida con la sentencia *fgetc*, de modo que la adecuamos al formato que utiliza el lenguaje C para poder mostrarla por pantalla.

La sentencia *fscanf* extrae un dato (numérico o cadena de texto) hasta que encuentra un espacio, omitiendo el espacio. Puede resultar útil para extraer datos en un fichero delimitados por espacios, pero conviene tener en cuenta esta circunstancia si pretendemos extraer texto que contiene espacios, ya que únicamente nos extraerá la primera palabra o cadena, hasta que se presente un espacio. También sería posible especificar longitudes exactas a ocupar con *fprintf* y a extraer con *fscanf*.

Indicar que con *fgets* extraemos una línea completa hasta el salto de línea (siendo éste incluido dentro de la cadena extraída).

Hemos visto ejemplos de apertura de ficheros para escritura y para lectura por separado. Si fuera necesario, ambos procesos pueden producirse en un mismo acceso al archivo. Escribe el siguiente código y comprueba cómo se puede escribir y recuperar datos en un mismo bloque de código.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int dato[3]; dato[0] = 322; dato[1]= 112; dato[2]=567;    // 3 datos numerados del 0 al 2
   int datoExtraido[3]; char textoExtraido [50];
   FILE* fichero;
   //ESCRITURA
   fichero = fopen("misdatos.txt", "wt");
   fprintf (fichero, "%s", "Datos de pesos en kgs");  fprintf (fichero, "%c", '\n');
   fprintf (fichero, "%d %c", dato[0], '\n');
   fprintf (fichero, "%d %c", dato[1], '\n');
   fprintf (fichero, "%d %c", dato[2], '\n');
   fclose(fichero);
   printf("Proceso de escritura completado \n");

   //LECTURA
   fichero = fopen("misdatos.txt", "rt");
   fgets (textoExtraido, 50, fichero);
   fscanf (fichero, "%d", &datoExtraido[0] );
   fscanf (fichero, "%d", &datoExtraido[1] );
   fscanf (fichero, "%d", &datoExtraido[2] );
   printf ("Texto extraido es: %s", textoExtraido);
   printf ("Dato extraido indice 0 vale %d \n", datoExtraido[0]);
   printf ("Dato extraido indice 1 vale %d \n", datoExtraido[1]);
   printf ("Dato extraido indice 2 vale %d \n", datoExtraido[2]);
   fclose(fichero);
   printf("Proceso de lectura completado");
   return 0; // Ejemplo curso C aprenderaprogramar.com
}
```

© aprenderaprogramar.com, 2006-2031

El código comienza con la declaración de las variables que van a intervenir en el programa. Se asignan valores a las variables *Dato[0], Dato[1] y Dato[2]* (tipo entero). El nombre interno del fichero se establece en "fichero" con la línea *FILE* fichero;*. A continuación se abre el archivo *C:\misdatos.txt* (en caso de que no existiera previamente el archivo es creado) para escritura (modo wt). A través de *fprintf* se escriben cuatro líneas y se cierra el fichero. Usamos el carácter no imprimible \n para introducir saltos de línea.

Se vuelve a abrir el fichero para lectura de datos (modo rt), se asignan los datos a las variables *textoextraido* y *datoExtraido[0]*, *[1]* y *[2]* y se cierra el fichero.

La información extraída del archivo se muestra en pantalla, donde aparecerá:

```
Proceso de escritura completado
Texto extraido es: Datos de pesos en kgs
Dato extraido indice 0 vale 322
Dato extraido indice 1 vale 112
Dato extraido indice 2 vale 567
Proceso de lectura completado
```

Si abrimos directamente el archivo *C:\misdatos.txt* con un visor como el bloc de notas, el contenido que observaremos es el siguiente:

```
Datos de pesos en kgs
322
112
567
```

El tamaño del archivo es de 41 bytes (aunque esto podría variar) desglosados en:

- 21 bytes al texto de la primera línea (21 caracteres).
- 2 bytes al salto de línea y retorno de carro de la primera línea.
- 4 bytes a la segunda línea (3 caracteres y uno no visible).
- 2 bytes al salto de línea y retorno de carro de la segunda línea.
- 4 bytes a la tercera línea (3 caracteres y uno no visible).
- 2 bytes al salto de línea y retorno de carro de la tercera línea.
- 4 bytes a la cuarta línea (3 caracteres y uno no visible).
- 2 bytes al salto de línea y retorno de carro de la cuarta línea.

Se comprueba que con acceso secuencial a pesar de guardar variables tipo entero, el espacio ocupado no es el correspondiente a este tipo de variables (2 bytes por entero) sino el equivalente a que fueran un texto en el archivo (1 byte por carácter).

© aprenderaprogramar.com, 2006-2031

Existen numerosas instrucciones, posibilidades adicionales y consideraciones relacionadas con el manejo de Ficheros en C que no vamos a estudiar, ya que nuestro objetivo es explicar cómo se aplica el pseudocódigo y algoritmia básica a un lenguaje y no entrar en los detalles del mismo. Entre estas cuestiones que no estudiaremos está la interpretación de punteros, los archivos binarios, funciones fwrite y fread, el acceso aleatorio, funciones fseek y ftell, etc..

La señal de Final de Archivo (EOF) la estudiaremos más adelante, ya que se constituye en una herramienta de gran interés para poder extraer datos desde un archivo hasta llegar al final del mismo cuando desconocemos la cantidad o longitud de datos existentes.

EJERCICIO

Crea un archivo denominado almacen.txt que contendrá una palabra en cada línea (para un total de 6 líneas) como se muestra a continuación:

El

gaucho

es

equilibrio

y

belleza

Crea un programa que cree un array de palabras de modo que se lea el contenido del archivo y se almacene en los elementos del array. Por ejemplo palabra[0] contendrá "El", palabra[1] contendrá "gaucho" y así sucesivamente. Usando un bucle, muestra por pantalla la frase (intercalando los espacios necesarios). En este caso el resultado del programa será como este:

Tras extraer la información del archivo construimos esta frase: El gaucho es equilibrio y belleza

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00539F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210

© aprenderaprogramar.com, 2006-2031