INFORMÁTICA APLICADA curso 2006-2007

PRÁCTICA 3: Entrada/Salida. Ficheros

Los objetivos de esta práctica son: (1) realizar una programación correcta y estructurada; (2) manejar el acceso a ficheros de datos desde un programa en C.

En la práctica se deberán realizar dos programas. Cada uno de ellos se encuentra descrito a continuación:

Programa 1

Para abrir un fichero, es necesario pasarle a la función fopen el nombre de ese fichero. El siguiente código ilustra el funcionamiento de la función fopen. manera.

```
FILE *fp;
fp = fopen("prueba1.txt", "w");
fprintf(fp, "%s", "hola");
fclose(fp);
```

El programa anterior abre un fichero con el nombre `prueba1.txt` en modo de escritura, a continuación escribe la cadena `hola` en él y, finalmente cierra el fichero. Lo siguiente que nos deberíamos preguntar es dónde se abre ese fichero. Por defecto, la función fopen abrirá el fichero en el mismo directorio donde se encuentre el fichero de nuestro programa ejecutable. Sin embargo, podemos hacer que la función fopen abra el fichero en una ubicación diferente:

```
FILE *fp;
fp = fopen("c:\\temp\\prueba2.txt", "w");
fprintf(fp, "%s", "hola");
fclose(fp);
```

Este trozo de código abre el fichero `prueba2.txt` en el directorio `c:\temp`. Recordemos que para escribir el carácter `\` en C debemos indicarle `\\`. En resumen, le estamos pasando a la función `fopen` una **ruta absoluta** para abrir el fichero, que es `c:\temp\prueba2.txt`. El programa abrirá el fichero en ese lugar independientemente de donde se encuentre el fichero ejecutable del programa. Altenativamente, podemos pasarle una **ruta relativa** a la función `fopen`

```
FILE *fp;
fp = fopen("..\\out\\prueba3.txt", "w");
fprintf(fp, "%s", "hola");
fclose(fp);
```

Observemos la estructura de directorios a continuación:

```
\practica0
          \exe → ejecutable
```

```
\inc
\src
\out
```

En este ejemplo, al utilizar "`..`" le estamos diciendo a `fopen` que se dirija al directorio inmediatamente superior a `exe`, y que, a continuación se introduzca en el directorio `out` y abra el fichero con nombre `prueba3.txt`.

Se pide al alumno que escriba el código siguiente y que compruebe su funcionamiento.

```
    char nombre[LONG_CADENA];
    char nombreLargo[LONG_CADENA];
    int numero=100;
    char carac = 'A';
    FILE *fp;

1:  printf("Intro nombre sin extension");
2:  scanf("%s", nombre);

3:  strcpy(nombreLargo, "..\\out\\");
4:  strcat(nombreLargo, nombre);
5:  strcat(nombreLargo, ".txt");

    printf("\nAbriendo fichero %s", nombreLargo);

6:  fp = fopen(nombreLargo, "w");
    if(fp==NULL){
        printf("\nError abriendo fichero");
         return;
      }

7:  fprintf(fp, "%d\n", numero);
8:  fprintf(fp, "%c\n", numero);

9:  fprintf(fp, "%d\n", carac);
10: fprintf(fp, "%c\n", carac);

11: fclose(fp);
```

El programa recoge el nombre del fichero por teclado (sin extensión, líneas 1, 2). A continuación añade la ruta relativa donde se encuentra el fichero, así como la extensión (líneas 3 a 5). Por ejemplo, si el usuario teclea `prueba`, forma la cadena: `..\out\prueba.txt`. En la línea 6 se abre efectivamente el fichero. En las líneas 7-10 se escribe información en el fichero.

El alumno deberá ejecutar el programa anterior y observar el fichero de salida generado (p.e. usando el programa Notepad de Windows). ¿Cómo se explican los datos del fichero?

Programa 2

El alumno deberá escribir el programa 2 siguiendo las instrucciones que se dan a continuación.

El programa 2 deberá realizar las siguientes funciones:

a) Leer cadenas de texto por teclado grabándolas en un fichero con extensión **.txt**. Se guardará cada cadena en una línea. El nombre del fichero donde se guardarán las cadenas también se introducirá por teclado (sin extensión). El programa leerá cadenas hasta que se teclee "**salir**", en mayúscula o minúscula.
b) Leer todas las cadenas del fichero anterior imprimiendo en pantalla cada cadena junto con su longitud y generando un nuevo fichero en el que se guardará cada cadena junto con su longitud en una línea. Este fichero se creará también con extensión **.txt**, y su nombre se introducirá por teclado.
c) Concatenar los dos ficheros generados en la práctica creando uno nuevo con extensión **.txt**. El nombre de este tercer fichero se introducirá por teclado.
d) Debe indicarse si se ha producido algún error en el proceso.

A continuación se muestra un **ejemplo** de ejecución del programa (en negrita aparecen los datos introducidos por el usuario):

```
Introducir el nombre del fichero sin extension:
fichero1

Introducir las cadenas (salir para terminar):
a
ab
abc
abcd
salir

Introducir el nombre del fichero donde incluir las longitudes:
fichero2

Cadenas junto con longitudes:
a:1
ab:2
abc:3
abcd:4

Introducir el nombre del fichero concatenado:
fichero3

Programa finalizado con exito.
```

Se deberán seguir los siguientes pasos:

1) Crear y configurar un proyecto en Visual C++ que se llamará **practica3**. El fichero fuente se llamará *practica3.c*.

2) Definir la función ***int GenerarFichero(char *nombrefich)***, que debe:

   a) Abrir el fichero cuyo nombre viene indicado en el parámetro (nombrefich).

b) Leer cadenas de texto por el teclado (se recomienda utilizar la función ***gets***) grabándolas en el fichero (se recomienda utilizar la función ***fprintf***). Debe existir un comando para finalizar ("**salir**").
c) Cerrar el fichero.
d) La función devolverá 0 si no se ha producido ningún error, y -1 en caso contrario.

3) Definir la función ***int CrearFicheroLongitudes(char *fich1, char *fich2)***, que debe

a) Abrir el fichero ***fich1*** para lectura y el fichero ***fich2*** para escritura.
b) Leer las cadenas que contenga el primer fichero (utilizar la función ***feof*** para detectar cuando se ha llegado al final del fichero).
c) Para cada cadena leída (se recomienda hacerlo con ***fgets*** para poder leer cadenas con espacios) del primer fichero se calculará su longitud y se mostrará en pantalla la cadena junto con su longitud. Así mismo se escribirá en el segundo fichero cada cadena junto con su longitud en una línea. Se debe actuar con precaución con la función ***fgets*** comprobando en todo momento si se ha producido un error de lectura antes de realizar cualquier operación. Para calcular la longitud de una cadena se recomienda utilizar la función ***strlen***.
d) Esta función devolverá 0 si no se ha producido ningún error, y –1 en caso contrario.

4) Para concatenar dos ficheros se recomienda implementar la función ***int ConcatenarFicheros(char *nombrefich1, char *nombrefich2, char *nombrefich3)*** donde los dos primeros parámetros es el nombre de los ficheros a concatenar (deben abrirse en *modo lectura*) y el último es el fichero resultado (abrir en *modo escritura*). El esquema de esta función es como sigue:

a) Se irán leyendo líneas del primer fichero (se recomienda utilizar la función ***fgets***), escribiéndolas en el fichero resultado (se recomienda utilizar la función ***fputs***). Como se indicó anteriormente, se debe actuar con precaución con la función ***fgets*** comprobando en todo momento si se ha producido un error de lectura antes de realizar cualquier operación.
b) Posteriormente se leerán líneas desde el segundo fichero (***fgets***) y se escribirán en el fichero resultado (***fputs***), a continuación de las cadenas escritas procedentes del primer fichero.
c) Se cerrarán todos los ficheros.

Si se ha producido algún error en esta función, devolver el valor –1. En caso contrario, devolver un 0.

5) Definir la función principal del programa (***main***).

Notas:

- El programa debe estar estructurado dividiendo cada tarea en funciones.

- *Se recomienda diseñar y comprobar cada función independientemente del resto del programa.*
- Se considerará que el tamaño máximo de una cadena introducida por el usuario es de 100 caracteres.
- Utilizar las funciones descritas en la librería estándar de Entrada /Salida (cabecera **<stdio.h>**) descritas en las transparencias para el manejo de ficheros.
- Para el manejo de cadenas puede utilizarse la librería de funciones *string* (cabecera **<string.h>**). A continuación se resumen las funciones más comunes (utilizar la ayuda de Visual C++ para obtener más información).

Prototipo de la función	Descripción
size_t strlen(const char *string);	Devuelve la longitud de una cadena
char *strcat(char *strDestination, const char *strSource);	Añade la cadena strSource al final de la cadena strDestination
int strncmp(const char *string1, const char *string2, size_t count);	Compara los *count* primeros caracteres de dos cadenas (0 SI SON IGUALES)
int strcmp(const char *string1, const char *string2);	Compara dos cadenas (0 SI SON IGUALES)
char *strcpy(char *strDestination, const char *strSource);	Copia el contenido de una cadena en otra