CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

INTRODUCCIÓN A LA PROGRAMACIÓN II

Ejercicios Propuestos Programa Refuerzo Programación:
Sesión 6 IPR2: Entrada/Salida de ficheros y librería string.h

Contenido

La llamada a strcpy() copia la cadena fuente en la cadena destino

```
char *strcpy ( char *destino, const char *origen );
```

La llamada a strcat() añade la cadena fuente al final de destino,es decir, las concatena

```
char* strcat (char* destino, const char* fuente);
```

La llamada a strchr() devuelve un puntero a la primera ocurrencia del carácter " ch" en la cadena "s1

```
char* strchr (char *s1, int ch);
```

La llamada a strrchr() devuelve un puntero a la última ocurrencia del carácter " ch" en la cadena "s1".

```
char* strrchr (char *s1, int ch);
```

La llamada a strcmp() compara alfabéticamente la cadena s1 con s2

Devuelve:

- **0 si s1 = s2**
- **<0 si s1 < s2**
- **>0 si s1 > s2**

```
int strcmp (const char* s1, const char* s2);
```

La llamada a strcasecmp() misma función que strcmp(), pero sin distinguir entre mayúsculas y minúsculas. Prototipo:

```
int strcasecmp (const char* s1, const char* s2);
```

La llamada a memcpy() reemplaza los primeros n bytes de *s1 con los primeros n bytes de *s2.

```
void* memcpy(void* s1, const void* s2, size_t n);
```

La llamada a strncpy() copia n caracteres de src a dest .

```
char *strncpy(char *dest, const char *src, size_t n)
```

La llamada a strtok() analiza o parsea la cadena s1 en tokens (componentes léxicos), dichos tokens delimitados por cadena s2 .

```
char *strtok (char* s1, const char* s2);
```

La llamada a strstr() busca la cadena s2 en la cadena s1 y devuelve un puntero a los caracteres donde se encuentra s2. Es decir, busca un substring dentro de otro. Si no lo encuentra devuelve NULL. Útil para trocear cadenas.

```
char *strstr (const char* s1, const char* s2);
```

1. Ejercicio 1. Estadísticas de una lista de números enteros pedidos al usuario de tamaño indeterminado

Crea un programa que calcule el máximo, el mínimo y la media de una lista de números decimales de un tamaño indeterminado que se encuentran en un fichero cuyo nombre se le pasa como parámetro del main.

Estos números se le piden al usuario (printf / scanf). La introducción de números finalizará cuando el usuario quiera. Los números se escriben en el fichero cada uno en una línea.

Los números son leídos del fichero y se calculan las estadísticas, presentándolas por pantalla.

Para finalizar se añaden al fichero las estadísticas, cada una en una línea y precedidas de unas palabras que indiquen cual es cual.

El programa principal debe encargarse de:

- Reservar memoria para el primer número
- Bucle mientras que el usuario quiera seguir introduciendo números
    - Pedir al usuario número
    - Almacenar el número
    - Preguntar al usuario si quiere introducir otro número
- Escribir los números en el fichero
- Leer los números del fichero
- Realizar las estadísticas
- Presentar estadísticas
- Añadir estadísticas al fichero
- Liberar memoria
- Cerrar fichero

Utilizar la función con el siguiente prototipo:

```
void calcular_estadisticas(double *numeros, int cantidad, double *min,
double *max, double *media);
```

2. Ejercicio 2. Igualdad y concatenación de Cadenas de caracteres

En un fichero tipo texto (cadenas.txt) están escritas en las dos primeras líneas dos cadenas de caracteres de un tamaño no determinado.

Escribe un programa en lenguaje C que lea cada una de ellas, compruebe si son iguales y si no lo son realizad la concatenación de las dos cadenas, presentándolo por pantalla.

Para concluir añadir en el fichero cadenas.txt un texto que indique si las dos cadenas son o no iguales.

El programa principal debe encargarse de:

- Leer las dos cadenas de caracteres del fichero.
- Comprobar si son iguales
    - Si lo son imprimir un mensaje por pantalla que lo muestre y concatenar las dos cadenas
    - Si no lo son escribir en el fichero las dos cadenas en orden inverso.
- Añadir el mensaje en el fichero
- Liberar memoria
- Cerrar fichero

El programa debe incluir las siguientes funciones:

```
//Función que devuelve una cadena de caracteres leída de
//un fichero y que realiza la reserva de memoria dinámica por
//bloques para la cadena

char* leeLineaDinamica(fd);

// Función que se le pasa como parámetros dos cadenas y devuelve
// verdadero si son iguales y falso si no lo son

int cadenasIguales(char* cadena1, char* cadena2);

// Función que se le pasa como parámetro una cadena y devuelve
//el número de caracteres que tiene

int tamCadena (char* cadena);

// Función que se le pasa como parámetros dos cadenas y devuelve
// la cadena concatenación de las dos que se pasan

char* concatenarCadenas(char* cadena1, char* cadena2);
```

3. Ejercicio 3. Lista de animales

Escribe un programa en C que almacene una lista de un número indeterminado de aves.

El programa solicitará al usuario el nombre de cada ave. Esta acción se repetirá mientras que el ave que escriba el usuario no esté ya incluida en la lista.

Cuando el usuario repita un pájaro tendrá que escribir por orden alfabético cada uno de ellos en un fichero.

El programa debe incluir las siguientes funciones:

```
//Función que devuelve una cadena de caracteres leída del
//buffer I/O y que realiza la reserva de memoria dinámica por
//bloques para la cadena

char* leeLineaDinamica();

// Función que se le pasa como parámetros dos cadenas y devuelve
// verdadero si son iguales y falso si no lo son

int cadenasIguales(char* cadena1, char* cadena2);

// Función que se le pasa como parámetro una cadena y devuelve
//el número de caracteres que tiene

int tamCadena (char* cadena);

// Función que se le pasa como parámetros dos cadenas y devuelve
// la cadena concatenación de las dos que se pasan
```

4. Ejercicio 4. Gestión zapatería

Escribe un programa en C que realice la gestión de una zapatería.

Una zapatería está formada por un conjunto de zapatos.

Los datos de cada uno de los zapatos son:

- Referencia. (Identificador único de zapato entero y consecutivo)
- Color: Por ejemplo: negro
- Tipo. Los tipos de zapatos son: MOCASIN, SANDALIA y ZAPATILLA
- Talla (entero)
- Precio (racional)

La información de los zapatos de la zapatería se encuentra guardada en un fichero (zapatería.txt). en el que se almacena en cada línea la información de un zapato con el siguiente formato: cada uno de los valores de los campos separados por punto y coma.

<referencia>;<color>;<tipo>;<Talla>;<Precio>

El programa debe incluir lo siguiente:

1. Definir un tipo estructura "struct zapato_t", con los datos anteriores.
2. Definir una variable para la zapatería.
3. Cargar la estructura con los valores de los zapatos, sacados del fichero zapatería.txt
4. Implementad un menú, utilizando para ello un enum, cuyas opciones sean:
    1. Introducir zapato. Al usuario se le piden los datos de un zapato y se introduce en un elemento nuevo de la estructura
    2. Imprimir zapatos zapatería
    3. Seleccionar los zapatos que sean de color negro y de la talla 45.
    4. Salvar a fichero. Cuando se seleccione esta opción se deberán guardar los registros que no se hayan guardado antes. Cada zapato se almacena en una línea y cada línea tiene los datos de cada zapato separados por el carácter ‘;’.
    5. Salir

El programa debe incluir las siguientes funciones:

```
// Función que devuelve una cadena de caracteres leída del
// buffer I/O y que realiza la reserva de memoria dinámica por
// bloques para la cadena

char* leeLineaDinamica();

// Función que se le pasa como parámetros la dirección de
// memoria donde comienza la biblioteca y el índice que ocupa el
// zapato dentro de la zapateria e incluye en la zapatería los
// valores del nuevo zapato

void introducirZapato (struct zapato_t* zapateria, int NumZap);
```

```
// Función que reciba una copia de un zapato y lo imprima por pantalla
void ImprimeZapato(struct zapato_t zapato);
```

5. Ejercicio 5. Analizador de líneas con búsqueda de palabras clave y extracción de campos

Se desea crear un programa en C que procese un fichero de texto que contiene múltiples líneas. Cada línea puede tener un formato similar al siguiente:

```
[INFO] Usuario: Juan Perez - Correo: juan@example.com - Edad: 29
[INFO] Usuario: Ana Lopez - Correo: ana.lopez@mail.com - Edad: 34
```

El programa debe recibir como argumento el nombre del fichero a procesar. Su función será:

1. Extraer de cada línea la información específica el nombre del usuario o del correo o de la edad,. Para ello
    a. Buscar en cada línea una palabra clave dada (por ejemplo, "Usuario" o "Correo" o Edad) usando la función **`strstr`**.
    b. Usar la función **`strchr`** para localizar separadores como **`:`** y **`-.`**
2. Mostrar por pantalla la información extraída de cada línea válida.

Requisitos:

- El usuario podrá introducir por consola una palabra clave a buscar.
- El programa debe usar la función **`strstr`** para localizar si una línea contiene una palabra clave.
- Debe usar **`strchr`** para encontrar el carácter **`:`** y extraer el valor que lo sigue.
- Se deben reservar dinámicamente las líneas con una función como `leeLineaDinamicaFichero`, utilizando fgetc.
- El programa debe liberar toda la memoria dinámica utilizada.