

SEMINARIO DE LENGUAJES

Opción C



Práctica 5- 2022

1. Escriba un programa que reciba como argumentos al main nombres de archivos. Para cada archivo recibido, deberá abrirlo, leer cada carácter en él y enviarlo a **stdout**. Utilice las funciones **fgetc** y **fputc**. Para comparar su funcionamiento puede usar el comando `cat (1)`.
2. Modifique el programa anterior para que en vez de escribir en **stdout**, escriba el resultado en el último archivo recibido como parámetro. *Nota: deberá recibir al menos dos argumentos el main.*
 - (a) Supongamos que el programa del ejercicio anterior se llama **prog_a.exe** y el de este ejercicio **prog_b.exe**. Compare las ejecuciones:

```
prog_a.exe entrada1.txt entrada2.txt > salida1.txt
prog_b.exe entrada1.txt entrada2.txt salida2.txt
diff salida1.txt salida2.txt
```

3. Con lo implementado anteriormente escriba un programa para copiar archivos. El mismo recibirá 2(dos) argumentos, el nombre del archivo de entrada y el nombre del archivo de salida. En caso de que el archivo de salida exista lo deberá re-escribir. Para comparar su funcionamiento puede usar el comando `cp (1)`.
4. Escriba un programa que compare dos archivos byte por byte e indique, si difieren, el primer byte en que lo hacen. Use funciones de entrada/salida orientada a chars. Funcionaría de forma similar al comando `cmp (1)`.
5. Escriba un programa que compare dos archivos e imprima las líneas en que difieren. Usar **fgets**. Funcionaría de forma similar al comando `diff (1)`.
6. Reescriba los ejercicios 1, 2, 3 y 4 empleando las funciones **fread** y **fwrite**. Analice cómo sería reescribir los ejercicios con las funciones de bajo nivel **read** y **write**, empleando:
 - (a) Buffer de 1 byte.
 - (b) Buffer de 1024 bytes.
 - (c) Buffer de 4096 bytes.
7. Escriba un programa que reciba como argumento un archivo e imprima las líneas en orden inverso. Es decir desde la última a la primera. Funcionaría de forma similar al comando `tac (1)`.
8. Escriba un programa que reciba como argumento un archivo e imprima cada línea del mismo en orden reverso. Es decir el orden de las líneas se conservaría, pero el orden de los chars de cada línea se invertiría. Funcionaría de forma similar al comando `rev (1)`.
9. Escriba un programa que reciba como argumento un archivo e imprima las últimas 10 líneas del mismo. Funcionaría de forma similar al comando `tail (1)`.
10. Dada una estructura como la siguiente:

```
struct {
    char apellido[100];
    char nombre[100];
    int edad;
}
```

- (a) Escriba un programa que reciba como parámetro el nombre de un archivo, lo cree y escriba en él registros que se irán leyendo desde la entrada estándar. Utilice la función **fwrite**.
- (b) Escriba un programa que reciba como parámetro el nombre de un archivo creado en el punto anterior e imprima su contenido. Utilice la función **fread**.
- (c) Analice qué sucede si los campos apellido y nombre son punteros a **char**. 
- (d) Busque una alternativa para almacenarlos. 

11. Describa qué es lo que hacen las siguientes declaraciones:

```
int *f1();
int (*f1)();
int (*f2)(int, int);
int *(*f1)();
int *x[10];
int (*x)[10];
```

12. Escriba una función que reciba un arreglo de enteros, su longitud y un parámetro más que sea una función que se aplique a cada elemento del arreglo.
 - (a) Pruebe con una función cuadrado, que aplica el cuadrado de un entero.
 - (b) Pruebe con una función que imprima un entero.
13. Agregue a la biblioteca **T_lista_generica**, implementada en *la Práctica 4*, la funcionalidad de:
 - (a) Ordenación de la lista. Pruebe la lista genérica con el tipo **T_alumno** de la práctica anterior, de forma tal de lograr el mismo comportamiento implementado en **T_lista_alumno**.
 - (b) Cree una nueva función que permita agregar varios elementos a la vez empleando argumentos dinámicos. Reuse la función de agregado de un solo elemento.
 - (c) Agregue las nuevas funciones eliminar y existe elemento mejoradas (no eliminando las anteriores) considerando un nuevo parámetro que sea la función de comparación.
14. Escriba una función sumatoria que reciba una lista dinámica de enteros y devuelva la sumatoria. Ejemplos de su uso sería:

```
sumatoria(2, 1, 2); /* retorna 3 */
sumatoria(4, 1, 2, 3, 4); /* retorna 10 */
sumatoria(1, 2); /* retorna 2 */
```
