Trabajo práctico integrador

Seminario de Lenguajes - Opción “C” 2021

Desarrollar un programa “ordenar” que permita ordenar líneas de un archivo de texto.

Las fechas de entrega, coloquio de defensa y re-entrega se publicarán por la plataforma de Cátedras.

Para la aprobación del trabajo, el mismo debe cumplir con lo solicitado en el enunciado, estar correctamente modularizado y no presentar errores (de lógica, de acceso a memoria, de acceso a archivos, etc). Notar que hay puntos del enunciado que deben realizarse únicamente en el caso de haber desaprobado algún parcial durante la cursada.

Para el desarrollo del programa deberá implementar un biblioteca (*library*) str_vector_t correctamente modularizada, tratando de respetar reglas para ocultar la implementación y separarla de la interfaz. La misma definirá los tipos:

- `str_vector_t`: Almacena una secuencia de strings, la usarán para almacenar las líneas del archivo.
- `enum sort_mode`: Identifica el modo de ordenación a utilizar:
  - `Secuencial`: En `str_vector_append_sorted` simplemente agrega al final, en `str_vector_sort` este modo no hace nada.
  - `Invertido`: Invierte el orden.
  - `Aleatorio`: Su uso no es válido para `str_vector_append_sorted`, en `str_vector_sort` este modo ordena los elementos del vector en posiciones aleatorias.

El módulo *str_vector* debe cumplir con una interfaz similar a la siguiente (los parámetros y/o sus nombres podrían variar acorde a cómo se implemente el módulo):

```
/*****************************************************************
 *                            Macros                             *
 *****************************************************************/

// Retorna el elemento en la posición index del vector, si la
// posición index no existe retorna NULL.
#define str_vector_get(vector, index) // ...COMPLETAR...

// Guarda el elemento en la posición index del vector, si el
// vector no tiene posiciones disponibles para tantos elementos,
```

```c
// se lo debe agrandar para poder insertar el elemento.
#define str_vector_set(vector, index, elemento) // ...COMPLETAR...


/*****************************************************************
 *                            Funciones                          *
 *****************************************************************/

/**
 * Retorna un vector vacío.
 */
str_vector_t str_vector_new();

/**
 * Agrega un nuevo elemento al vector. La cadena no debe copiarse
 * sino que se almacena el puntero.
 * Si no hay espacio alocado para el nuevo elemento es necesario
 * agrandar el vector.
 */
void str_vector_append(str_vector_t *vector, char *string);

/**
 * Agrega un nuevo elemento al vector siguiendo el orden especificado
 * por `mode` (excepto el modo random o aleatorio). La cadena no debe
 * copiarse sino que se almacena el puntero.
 * Si no hay espacio alocado para el nuevo elemento es necesario
 * agrandar el vector.
 */
void str_vector_append_sorted(str_vector_t *vector, char *string,
enum sort_mode mode);

/**
 * Redimensiona el vector para contener `elements` elementos.
 * Si el vector tiene más de `elements` elementos también
 * cambia el tamaño lógico del vector para que sea igual a
 * `elements`.
 */
char *str_vector_resize(str_vector *vector, unsigned elements);

/**
 * Ordena el vector de acuerdo a el modo elegido.
 */
void str_vector_sort(str_vector_t *vector, enum sort_mode mode);

/**
 * Libera la memoria alocada para el vector.
 */
```

```
void str_vector_free(str_vector_t *vector);
```

El programa a desarrollar debe admitir los siguientes argumentos y funcionalidades:

- `-r o --reverse`
  Invierte la condición de ordenación.
- `-s o --shuffle`
  Ordena de forma aleatoria (no imprime repetidos).
- `-i <archivo> o --input <archivo>`
  Lee las líneas desde el archivo indicado. Si se omite se leen desde stdin.
- `-o <archivo> o --output <archivo>`
  Escribe los resultados en el archivo indicado. Si se omite se escriben en stdout.
- `--help`
  Imprime la ayuda que indica cómo utilizar el programa en salida estándar y termina.
- `-c o --count`
  Cuenta la cantidad de líneas en el archivo de entrada e imprime el resultado en la salida estándar.
- `-u o --unique`
  Solo guarda en el archivo las líneas que no se repiten con sus adyacentes. **(Quienes hayan desaprobado al menos un parcialito deben implementar esta opción)**.
- `-R o --repeated`
  Se usa en combinación con -u (o su equivalente largo), al guardar o imprimir cada línea le agrega la cantidad de veces que esa línea se repite de acuerdo al criterio usado con -u. **(Quienes hayan desaprobado los dos parcialitos deben implementar esta opción)**.

Si se omiten -r y -s (o sus equivalentes largos) el archivo es ordenado en el orden dado por los valores ASCII de sus elementos (como con strcmp).
Si se invoca el programa con -r y -s (o sus equivalentes largos) al mismo tiempo el programa deberá mostrar un mensaje de error y terminar sin leer el archivo de entrada.
Si se invoca el programa con -s y -u (o sus equivalentes largos) al mismo tiempo el programa deberá mostrar un mensaje de error y terminar sin leer el archivo de entrada.
Si se invoca el programa con -c y una o más de los argumentos -o, -r ó -s (o sus equivalentes largos) al mismo tiempo el programa deberá mostrar un warning indicando que se ignorará el argumento (-o, -r ó -s), realizará el conteo e imprimirá la cantidad de líneas en la salida estándar.

Hint: ver man page sort(1) para comparar resultados.