Algoritmos y Programación I (95.11) – Curso Essaya – 3er parcialito, 2do recuperatorio – 15/03/2021

Resolver los siguientes problemas en un único archivo de código ISO-C99.

1. Una palabra puede tener múltiples sinónimos. Se quiere modelar un TDA sinónimos de palabra que permita almacenar la palabra y sus sinónimos. Se pide:

   a. Declarar la estructura que encapsula el TDA. Explicar qué representa cada miembro y documentar el invariante de representación.

   b. Implementar la primitiva `size_t sinonimos_cantidad_sinonimos(const sinonimos_t *s);` que devuelva la cantidad de sinónimos que tiene asociada la palabra.

   c. Implementar la primitiva `bool sinonimos_es_sinonimo(const sinonimos_t *sinonimos, const sinonimos_t *palabra);` que diga si la palabra que representa `palabra` está contenida entre los sinónimos de `sinonimos`.

      Por ejemplo (palabra: sinónimos):

      - Programar: Exponer, planificar, plantear, proyectar, sistematizar, esquematizar, bosquejar.
      - Exponer: Explicar, explanar, manifestar, referir, plantear, formular, afirmar, alegar, declarar, describir, razonar.

      Entonces `sinonimos_es_sinonimo(programar, exponer) -> true` y `sinonimos_es_sinonimo(exponer, programar) -> false`.

2. Se tiene un producto definido como

```
typedef struct {
    unsigned int id;
    char nombre[MAX_NOMBRE];
    float precio;
} producto_t;
```

   a. Escribir una función `bool escribir_producto_csv(const producto_t *p, FILE *f);` que dado un producto `p` y un archivo `f` vuelque el producto en una línea de archivo de texto en formato CSV.

   b. Escribir una función `bool leer_productos(const char *r, producto_t ps[], size_t max, size_t *n);` que reciba una ruta `r` y un vector `ps` con `max` productos. El archivo guardado en `r` contiene el volcado **binario** de `n` estructuras de tipo `producto_t`. Se deben leer todos los productos del archivo hasta un máximo `max` y guardarlos en `ps`. Se debe devolver por `n` la cantidad de productos leídos y por el nombre `false` en caso de falla (que el archivo no

contuviera ningún producto o que hubiera menos de `max` productos no es una falla) o `true` de haber podido realizar la operación.

El archivo binario es tan sólo una sucesión de productos:

```
+----+----+----+----+-   -+----+
| p1 | p2 | p3 | p4 | ... | pn |
+----+----+----+----+-   -+----+
```

3. Escribir un programa que se ejecute como:

   ```
   $ ./convertir_productos [archivo_entrada] <[archivo_salida]>
   ```

   donde el parámetro `[archivo_entrada]` es obligatorio mientras que el parámetro `<[archivo_salida]>` es optativo.

   El programa deberá levantar los `producto_t` del archivo binario `[archivo_entrada]` y convertirlos a CSV. En el caso de que se haya especificado un `<[archivo_salida]>` se volcarán en él, en el caso de que no se haya especificado se volcarán en `stdout`.

   El programa reutiliza las funciones del ejercicio 2.

Se deberán codificar todas las funciones en un único archivo de código fuente que será el que se entregue.

El examen es de elaboración personal, todo el código entregado debe ser realizado por el alumno.

El examen se envía a través del sistema de entregas de trabajos prácticos.

¡Suerte! :)