Algoritmos y Programación I (95.11) – Curso Kuhn – 3er parcialito – 30/05/2019

Resolver los siguientes problemas en forma clara y legible en código ISO-C99.

1. Se quiere modelar un TDA que represente a un vector en $\mathbb{R}^n$. Se pide:
   a. Declarar la estructura `vector_t` que representa el TDA. Explicar qué modela cada uno de los miembros y sus valores posibles (*invariante*).
   b. Implementar la primitiva `size_t vector_dimension(const vector_t *v);` que retorne la dimensión n del vector.
   c. Implementar la primitiva `double vector_norma(const vector_t *v);` que devuelve la norma del vector, $\|v\|$.
   d. Implementar la primitiva `bool vector_sumar(vector_t *a, const vector_t *b);` que sume `b` en `a`, si es que son sumables. La función devuelve `true` si la operación pudo ser realizada.
2. Se tiene un archivo **binario** que contiene un `size_t` n seguido de n flotantes, según el siguiente esquema:

```
+----+----+----+----+----+-   -+----+
| n  | a1 | a2 | a3 | a4 | ... | an |
+----+----+----+----+----+-   -+----+
```

   a. Escribir una función `bool escribir_arreglo(const char *r, const float *a, size_t n);` que reciba una ruta `r` y un arreglo `a` de `n` elementos y los escriba en un archivo **binario** según el formato anterior.
   b. Escribir una función `float *leer_arreglo(const char *r, size_t *n);` que reciba una ruta `r` a un archivo binario y devuelva por nombre el arreglo contenido en él, y en `n` la cantidad de elementos leídos.
3. Dado el formato (y las funciones) del ejercicio 2 escribir un programa que se ejecute

   ```
   $ ./escalar entrada salida valor
   ```

   que cargue en memoria el arreglo contenido en el archivo **binario** `entrada`, multiplique cada uno de sus elementos por el escalar `valor` y lo guarde en el archivo **binario** `salida`.

¡Suerte! :)

Algoritmos y Programación I (95.11) – Curso Kuhn – 3er parcialito – 30/05/2019

Resolver los siguientes problemas en forma clara y legible en código ISO-C99.

1. Se quiere modelar un TDA que represente a un vector en $\mathbb{R}^n$. Se pide:
   a. Declarar la estructura `vector_t` que representa el TDA. Explicar qué modela cada uno de los miembros y sus valores posibles (*invariante*).
   b. Implementar la primitiva `size_t vector_dimension(const vector_t *v);` que retorne la dimensión n del vector.
   c. Implementar la primitiva `double vector_norma(const vector_t *v);` que devuelve la norma del vector, $\|v\|$.
   d. Implementar la primitiva `bool vector_sumar(vector_t *a, const vector_t *b);` que sume `b` en `a`, si es que son sumables. La función devuelve `true` si la operación pudo ser realizada.
2. Se tiene un archivo **binario** que contiene un `size_t` n seguido de n flotantes, según el siguiente esquema:

```
+----+----+----+----+----+-   -+----+
| n  | a1 | a2 | a3 | a4 | ... | an |
+----+----+----+----+----+-   -+----+
```

   a. Escribir una función `bool escribir_arreglo(const char *r, const float *a, size_t n);` que reciba una ruta `r` y un arreglo `a` de `n` elementos y los escriba en un archivo **binario** según el formato anterior.
   b. Escribir una función `float *leer_arreglo(const char *r, size_t *n);` que reciba una ruta `r` a un archivo binario y devuelva por nombre el arreglo contenido en él, y en `n` la cantidad de elementos leídos.
3. Dado el formato (y las funciones) del ejercicio 2 escribir un programa que se ejecute

   ```
   $ ./escalar entrada salida valor
   ```

   que cargue en memoria el arreglo contenido en el archivo **binario** `entrada`, multiplique cada uno de sus elementos por el escalar `valor` y lo guarde en el archivo **binario** `salida`.

¡Suerte! :)