Algoritmos y Programación I (95.11) – Curso Essaya – 3er parcialito – 10/06/2022

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

1. Se quiere modelar el TDA polinomio, el cual representa a un polinomio de grado n dado por la ecuación $a_0 + a_1\,x + a_2\,x^2 + \ldots + a_n\,x^n$. 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 polinomio_grado(const polinomio_t *p)` que devuelva el grado del polinomio.

   c. Implementar la primitiva `polinomio_t *polinomio_derivar(const polinomio_t *p);` que dado un polinomio $p(x)$ retorne el polinomio $p(x)'$, su derivada. Puede asumir que se encuentra implementada la función `static polinomio_t *_polinomio_crear(size_t n);` que crea un polinomio de grado n.

2. Se tiene un archivo **binario** que contiene un `size_t` n seguido de n valores flotantes de doble precisión, según el siguiente esquema:

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

   a. Escribir una función `bool escribir_doubles(const char *r, const double a[], size_t n);` que reciba una ruta `r` y un arreglo `a` de `n` doubles y los escriba en un archivo **binario** según el formato anterior.

   b. Escribir una función `double *leer_doubles(const char *r, size_t *n);` que reciba una ruta `r` a un archivo **binario** y devuelva por el nombre el arreglo de doubles 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:

   ```
   $ ./transformar entrada salida a b
   ```

   que cargue en memoria el arreglo de flotantes de doble precisión contenido en el archivo **binario** `entrada` y aplique la transformación $ax + b$ a cada uno de sus elementos y lo guarde en el archivo **binario** `salida`.

   Se deben utilizar las funciones desarrolladas en el punto 2.

¡Suerte! :)

Algoritmos y Programación I (95.11) – Curso Essaya – 3er parcialito – 10/06/2022

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

1. Se quiere modelar el TDA polinomio, el cual representa a un polinomio de grado n dado por la ecuación $a_0 + a_1\,x + a_2\,x^2 + \ldots + a_n\,x^n$. 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 polinomio_grado(const polinomio_t *p)` que devuelva el grado del polinomio.

   c. Implementar la primitiva `polinomio_t *polinomio_derivar(const polinomio_t *p);` que dado un polinomio $p(x)$ retorne el polinomio $p(x)'$, su derivada. Puede asumir que se encuentra implementada la función `static polinomio_t *_polinomio_crear(size_t n);` que crea un polinomio de grado n.

2. Se tiene un archivo **binario** que contiene un `size_t` n seguido de n valores flotantes de doble precisión, según el siguiente esquema:

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

   a. Escribir una función `bool escribir_doubles(const char *r, const double a[], size_t n);` que reciba una ruta `r` y un arreglo `a` de `n` doubles y los escriba en un archivo **binario** según el formato anterior.

   b. Escribir una función `double *leer_doubles(const char *r, size_t *n);` que reciba una ruta `r` a un archivo **binario** y devuelva por el nombre el arreglo de doubles 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:

   ```
   $ ./transformar entrada salida a b
   ```

   que cargue en memoria el arreglo de flotantes de doble precisión contenido en el archivo **binario** `entrada` y aplique la transformación $ax + b$ a cada uno de sus elementos y lo guarde en el archivo **binario** `salida`.

   Se deben utilizar las funciones desarrolladas en el punto 2.

¡Suerte! :)