

## Algoritmos y Programación I (95.11) – Curso Kuhn – 3<sup>er</sup> parcialito – xx/xx/xxxx

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  $a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$ . Se pide:
  - a. Declarar la estructura que encapsula el TDA. Explicar qué representa cada uno de los miembros.
  - b. Implementar la primitiva `double polinomio_evaluar(const polinomio_t *p, double x)`; que evalúe el polinomio en la  $x$  dada.
  - 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 una primitiva privada del TDA `static polinomio_t *_polinomio_crear(size_t n)`; que crea un polinomio de grado  $n$ .
2. Se tiene una estructura que representa a una persona como `typedef struct { char nombre[MAX]; unsigned int dni; float altura; } persona_t`; se pide:
  - a. Escribir una función `bool leer_persona(FILE *f, persona_t *p)`; que lea una persona del archivo **binario** `f` y la guarde en la estructura `p`. La función debe devolver `false` de no poder realizar la operación.
  - b. Escribir una función `void escribir_persona(FILE *f, const persona_t *p)`; que dado un archivo de **texto** `f` escriba en él a la persona `p` en formato CSV “nombre;dni;altura”.
  - c. Escribir una función `bool leer_personas(const char *r, persona_t v[], size_t max, size_t *n)`; que reciba una ruta a un archivo **binario** `r` y un vector `v` de `max` elementos y cargue en él todas las estructuras contenidas en el archivo hasta el máximo. La función debe retornar por `n` la cantidad de estructuras leídas y `false` en caso de falla.
3. Escribir un programa que se ejecute “\$ ./convertir entrada” donde **entrada** es el nombre de un archivo **binario** de personas que imprima por `stdout` cada una de las personas del archivo en formato CSV. Si el programa se ejecutara “\$ ./convertir --help” se debe imprimir una pequeña ayuda y terminar.

## Algoritmos y Programación I (95.11) – Curso Kuhn – 3<sup>er</sup> parcialito – xx/xx/xxxx

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  $a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$ . Se pide:
  - a. Declarar la estructura que encapsula el TDA. Explicar qué representa cada uno de los miembros.
  - b. Implementar la primitiva `double polinomio_evaluar(const polinomio_t *p, double x)`; que evalúe el polinomio en la  $x$  dada.
  - 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 una primitiva privada del TDA `static polinomio_t *_polinomio_crear(size_t n)`; que crea un polinomio de grado  $n$ .
2. Se tiene una estructura que representa a una persona como `typedef struct { char nombre[MAX]; unsigned int dni; float altura; } persona_t`; se pide:
  - a. Escribir una función `bool leer_persona(FILE *f, persona_t *p)`; que lea una persona del archivo **binario** `f` y la guarde en la estructura `p`. La función debe devolver `false` de no poder realizar la operación.
  - b. Escribir una función `void escribir_persona(FILE *f, const persona_t *p)`; que dado un archivo de **texto** `f` escriba en él a la persona `p` en formato CSV “nombre;dni;altura”.
  - c. Escribir una función `bool leer_personas(const char *r, persona_t v[], size_t max, size_t *n)`; que reciba una ruta a un archivo **binario** `r` y un vector `v` de `max` elementos y cargue en él todas las estructuras contenidas en el archivo hasta el máximo. La función debe retornar por `n` la cantidad de estructuras leídas y `false` en caso de falla.
3. Escribir un programa que se ejecute “\$ ./convertir entrada” donde **entrada** es el nombre de un archivo **binario** de personas que imprima por `stdout` cada una de las personas del archivo en formato CSV. Si el programa se ejecutara “\$ ./convertir --help” se debe imprimir una pequeña ayuda y terminar.

¡Suerte! :)

¡Suerte! :)