T.P. N° 10: Punteros a función y T.A.D.

Ejercicio 1

Dado un arreglo de números enteros, se desea presentar un menú (por pantalla) que ofrezca las siguientes opciones :

- ☞ **Promedio**
- ☞ **Mínimo**
- ☞ **Máximo**

para lo cual se deberán definir las siguientes estructuras de datos:

- Un arreglo de enteros sobre el cual se realizarán las funciones antedichas (puede estar inicializado por programa).
- Un arreglo donde cada elemento contiene una cadena de caracteres que representa la opción y un puntero a la función correspondiente. Todas las funciones devuelven valores reales.

Hacer un programa que muestre un menú permitiendo al usuario elegir funciones para aplicar sobre un arreglo de enteros y mostrando el resultado de cada caso. Diseñe un mecanismo de menúes que luego pueda ser utilizado en otros programas.

Ejercicio 2

Escribir una función genérica **tieneDuplicados** que determine si un vector de elementos desordenados tiene o no duplicados. La función debe poder trabajar con cualquier tipo de arreglos. Analizar los parámetros necesarios para la función, bajo qué tipo de datos se recibirán los arreglos y qué funciones auxiliares serán necesarias para manipular los datos.

Ejemplos:

- Para el vector {1, 3, 5, 7} debería devolver falso.
- Para el vector {"Juan", "juan"} podría devolver falso o verdadero, según el criterio del usuario de la función.
- Para el vector {{50, "Perez"}, {50, "Gonzales"}} podría devolver verdadero si se considera que dos estructuras son iguales si los primeros elementos de cada estructura coinciden o podría devolver falso si considera que dos estructuras son iguales si y solo si todos sus campos son iguales.

Ejercicio 3

Se tienen dos arreglos de estructuras de un mismo tipo (no importa cual) ordenados y se desea obtener a partir de los mismos un nuevo arreglo que contenga la unión ordenada de ambos (no debe contener elementos repetidos).

Escribir una función **genérica** que pueda trabajar con cualquier tipo de arreglos. Analizar los parámetros necesarios para la función, bajo qué tipo de datos se recibirán los arreglos y qué funciones auxiliares serán necesarias para manipular los datos.

Ejercicio 4

Escribir un programa para hallar las raíces de una función matemática en un intervalo cerrado, recorriéndolo de forma tal que el intervalo quede dividido en 100000 (cien mil) particiones o subintervalos. Ejemplo: si el intervalo es [1, 50000] deberá evaluar la función en los puntos 1, 1.5, 2, etc. (también se tomarán como válidos los puntos 1, 1.49999, etc.).

El programa ofrecerá un menú de funciones matemáticas y deberá solicitar los extremos del intervalo, imprimiendo los resultados en la salida estándar. Todas las funciones reciben y devuelven un valor real.

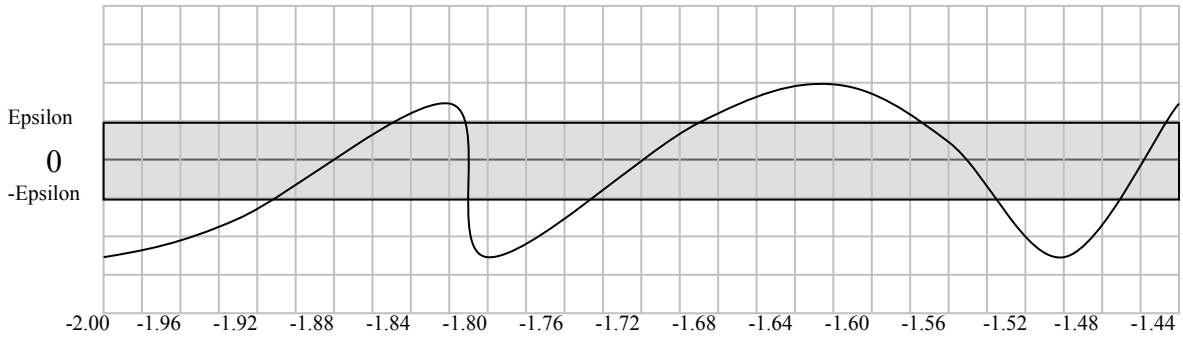
La función que realice la búsqueda de las raíces debe recibir como parámetros una estructura que represente al intervalo y una función a la cual se le quieren hallar las raíces, regresando en su nombre una estructura que empaquete un arreglo con aquellas particiones donde haya raíces y la dimensión de dicho arreglo.

Para detectar una raíz se deben considerar dos casos:

- Que la función cambie de signo entre dos puntos: En ese caso se agrega al arreglo una partición con ambos puntos.
- Que la función se haga cero en un punto (considerando un error de EPSILON): En ese caso la partición que se agrega al arreglo de resultados está formada por el punto anterior al que se detectó como raíz y el próximo que no lo sea.

Ejemplo:

Considerando un intervalo cuyo paso es de 0.02, se grafica un fragmento del mismo indicando que raíces se encuentran:



Se detectarán raíces en :

- **-1.90**, porque el valor absoluto de la imagen en ese punto es menor que EPSILON. Pero los dos puntos siguientes (-1.88 y -1.86) también se consideran raíces por el mismo motivo. Por lo tanto en este caso la partición que se agrega con resultado toma los valores **-1.92** (por ser el anterior a la primera raíz) y **-1.84** (por ser el posterior a la última raíz).
- luego se detecta un cambio de signo de la función entre los puntos **-1.82** y **-1.80**, por lo que se agrega dicha partición.
- **-1.74** es raíz por comparación con EPSILON, al igual que los dos puntos siguientes (-1.72, -1.70) por lo que se agrega la partición **-1.76**, **-1.68**.
- **-1.56** es raíz por comparación con EPSILON, al igual que el punto siguiente -1.54, por lo que se agrega la partición **-1.58, -1.52**.
- **-1.46** es raíz por comparación con EPSILON, y el punto siguiente ya no lo es , por lo que se agrega la partición **-1.48, -1.44**.

Por lo tanto nuestra función debería retornar una estructura conteniendo los siguientes valores:

{ 5, { { -1.92, -1.84}, { -1.82, -1.80}, { -1.76, -1.68}, { -1.58, -1.52}, { -1.48, -1.44} } }

Ejercicio 5

¿Qué indican las siguientes declaraciones complejas?

```
a) int *(*p) (int (*a)[ ]);
b) int *p[10];
c) int (*p)[10];
d) int *p(void);
e) p (char *a);
f) int *p(char *a);
g) int (*p) (char *a);
h) int (*p(char *a))[10];
i) int p(char (*a)[ ]);
j) int p(char *a[ ]);
k) int *p(char a[ ]);
l) int *p(char (*a)[ ]);
m) int *p(char *a[ ]);
n) int (*p)(char(*a)[ ]);
o) int *(*p)(char *a[ ]);
p) int (*p[10])(void);
q) int (*p[10])(char a);
r) int *(*p[10])(char a);
s) int *(*p[10])(char *a);
t) int (*p) (char *a[ ]);
```

Ejercicio 6

Escribir las siguientes declaraciones complejas:

a) **p** es un arreglo de punteros a funciones que no reciben parámetros y devuelven punteros a double.
b) **p** es una función que recibe un puntero a char y devuelve un puntero a un arreglo de 3x5 enteros.
c) **p** es un puntero a un arreglo de N punteros a función que reciben un entero y retornan un puntero a entero.

Ejercicio 7

Crear un T.A.D. que maneje números complejos. Debe ofrecer como mínimo las siguientes funcionalidades:

- obtener la parte real
- obtener la parte imaginaria
- sumar dos números complejos (retorna un nuevo número complejo)

Ejercicio 8

Agregar al TAD de listas genéricas provisto por la Cátedra una función que reciba un número entero i y devuelve el i-ésimo elemento, donde el primer elemento tiene el índice cero.

Ejercicio 9

Crear un T.A.D. que permita operar con un conjunto de números enteros.

Ejercicio 10

Extender el T.A.D. anterior para que opere con tipos de datos genéricos. Al crear el conjunto se recibe la función de comparación y el tamaño de cada elemento, para realizar la copia de elementos en forma superficial (*shallow copy).*