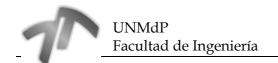
Práctica 3 - TDA

- 1. Codificar un TDA para representar una "dupla". Una dupla es un conjunto ordenado de tamaño fijo de 2 elementos. Para acotar la implementación se pide realizar una Dupla de dos enteros (int). Con lo cual no será una Dupla genérica que pueda tener cualquier tipo de elemento. Las operaciones que deberá soportar el TDA son:
 - crear(entero primero, entero segundo) retorna dupla.
 - primero(dupla d) retorna entero.
 - segundo(dupla d) retorna entero.
 - multiplicar(dupla d, entero múltiplo) retorna dupla. Genera una nueva dupla que es el resultado de multiplicar ambos elementos por el número múltiplo que se pasa como parámetro.
 - adicionar(dupla d, entero adición) retorna dupla. Genera una nueva dupla resultado de sumarle el parámetro adición a ambos elementos.
 - sumar(dupla a, dupla b) retorna dupla. Genera una nueva dupla resultado de la suma de las dos recibidas como parámetro.
 - restar(dupla a, dupla b) retorna dupla. Idem anterior, pero restando de a, los valores de la dupla b.
- 2. Implementar un TDA para el **tipo booleano** en C. Recordar que en C no existe este tipo como tipo básico, por lo cual a veces el código pierde legibilidad y el uso de números conlleva a errores. En C se asume que el número 0 es falso y el resto verdadero. Las operaciones que deberá soportar el TDA son:
 - crear(entero valor) retorna booleano.
 - and(booleano b1, booleano b2) retorna booleano.
 - or(booleano b1, booleano b2) retorna booleano.
 - not(booleano b) retorna booleano.
 - comoEntero(booleano b) retorna entero.
- 3. Realizar un TDA para **números fraccionarios**, es decir aquellos que se expresan como el cociente de dos números enteros. Las operaciones que deberá soportar el TDA son:
 - · crear(entero numerador, entero denominador) retorna fracción.
 - numerador(fracción f) retorna entero.
 - denominador(fracción f) retorna entero.
 - sumar(fracción f1, fracción f2) retorna fracción.
 - restar(fracción f1, fracción f2) retorna fracción.
 - multiplicar(fracción f1, fracción f2) retorna fracción.
 - dividir(fracción f1, fracción f2) retorna fracción.
 - simplificar(fracción f) retorna fracción simplificada.
 - iguales(fracción f1, fracción f2) retorna verdadero o falso.
- 4. Desarrollar un TDA que represente la idea de "texto". En C no existe este tipo básico, con lo cual las aplicaciones siempre quedan acopladas a la idea de que un *string* es una cadena de caracteres, por lo que se manejan con punteros y posiciones. Las operaciones que deberá soportar el TDA son (se propone no utilizar funciones *strxxxx*):
 - crear(char* c) retorna Texto
 - destruir(Texto t) sin valor de retorno. Libera la memoria reservada en crear.
 - tamaño(Texto t) retorna entero.
 - caracterEn(Texto texto, entero posición) retorna caracter.
 - concatenar(Texto t1, Texto t2) retorna Texto.
 - reemplazar(Texto texto, char caracter, char nuevoCaracter) retorna Texto.

Práctica 3 – TDA Página 1 / 2



Ej: reemplazar('Balsa', 'a', 'o') >> 'Bolso'

- subTexto(Texto texto, entero desde, entero hasta) retorna Texto.
 - Ej: subTexto('HolaMundo', 4, 7) retorna 'Mun'
- comienzaCon(Texto texto, Texto prefijo) retorna entero | booleano.
- terminaCon(Texto texto, Texto sufijo) retorna entero | booleano.
- 5. Desarrollar una **agenda de contactos** mediante un TDA. Para cada contacto se almacena *nombre* y *teléfono*. Los contactos deben estar ordenados alfabéticamente. El TDA debe contar con los siguientes operadores: agregar contacto, listar agenda, buscar por nombre (búsqueda lineal).

Se solicitan dos implementaciones: una con arreglos paralelos y otra con arreglo de registros. Mejorar el operador buscar por nombre, recodificándolo como búsqueda binaria.

Práctica 3 – TDA Página 2 / 2