

# Proyecto 1 - Taller: TAD Conjunto Finito

Algoritmos y Estructuras de Datos II 2020

## Preliminares

En este proyecto deberán implementar el TAD *Conjunto Finito* en el lenguaje de programación C. Deben basarse en la solución elaborada para el teórico / práctico, intentando que sea una simple traducción de un lenguaje a otro. Dadas las diferencias entre C y el lenguaje utilizado en el práctico es inevitable realizar algunos cambios.

En C las declaraciones de las funciones del TAD Conjunto finito deben estar en el archivo `set.h`, donde es necesario además definir el tipo `set`. La implementación de estas funciones estarán en `set.c`. En el ejercicio práctico, la definición de `Set` se hace de la siguiente manera:

```
type Set of T = List of T
```

Si queremos seguir al pie de la letra esta definición, en `set.h` deberíamos definir a `set` como:

```
typedef list set;
```

Aquí `set` es un sinónimo de `list` (al igual que en el práctico) pero al incluir esta definición en el `.h` estaríamos exponiendo detalles de la implementación al usuario de nuestro TAD (no habría encapsulamiento). Recordar que todo lo que está en el `.h` es público pues es la interfaz.

Entonces, si el usuario sabe que `set` es en realidad el tipo `list`, podría usar las funciones de listas sobre los conjuntos y romper propiedades de la representación (por ejemplo agregando un elemento desordenadamente o un elemento repetido).

Otro problema es que si más adelante queremos cambiar la representación usada para conjuntos (para en lugar de listas usar arreglos, árboles, hash, etc...) tendremos que modificar la definición de `set` en `set.h`, pero esa no es la idea. **Queremos separar la interfaz de la implementación.** Todas las implementaciones del tipo `set` deben mantener intacto el archivo `set.h`.

Para lograr esto, vamos a definir al tipo `set` como un puntero a una estructura:

```
typedef struct s_set * set;
```

Luego, quien desarrolle una implementación de `set` debe establecer en `set.c` qué componentes hay en la estructura `struct s_set`. En nuestro caso particular usaremos un campo del tipo `list`, por ejemplo:

```
struct s_set {  
    list sorted_list;  
};
```

Esta es la manera en que logramos encapsulación en C. Es importante lograr esto ya que nos permitirá cambiar la implementación de los conjuntos finitos sin tener que modificar los programas que usen el TAD.

# Actividades

Para programar el TAD `set` van a contar con una implementación opaca provista por la cátedra. Su interfaz está definida en `list.h`. No tendrán el código fuente de la implementación sino que les daremos el código ya compilado como archivo objeto. Se proveen los siguientes archivos:

- `list.h` : Interfaz y especificación del TAD Lista
- `list-x86_64.o` : Implementación del TAD Lista compilada para sistemas linux de 64 bits
- `list-i386.o` : Implementación del TAD Lista compilada para sistemas linux de 32 bits
- `list-macos.o` : Implementación del TAD Lista compilada para MAC
- `set.h` : Interfaz del TAD Conjunto finito en C
- `type_elem.h` : Definición del tipo para los elementos del TAD conjunto finito
- `test_set.c` : Archivo para probar las funciones del TAD `set`
- `test_list.c` : Archivo de ejemplo que prueba funciones del TAD `list`

**Ejercicio 1:** Completar la especificación del TAD Conjunto Finito que se encuentra en `set.h`. Pueden usar como guía las especificaciones incluidas en `list.h`. No es necesario especificar en inglés.

**Ejercicio 2:** Basados en la solución elaborada para el teórico / práctico, implementar el TAD Conjunto Finito utilizando el TAD Lista, cuya interfaz está detallada en `list.h`. Chequear precondiciones mediante `assert()` en las funciones implementadas.

**Ejercicio 3:** Completar el archivo `test_set.c` donde deberán definir una función `main()` que pruebe las funciones del TAD Conjunto finito cuya implementación no fue incluida en el ejercicio práctico. Para compilarlo:

```
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -c test_set.c
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -c set.c
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -o test_s list-x86_64.o set.o
test_set.o
```

en el caso de utilizar un sistema distinto a linux de 64 bits, reemplazar `list-x86_64.o` por el archivo objeto correspondiente a su plataforma. Avisar si usan una plataforma no contemplada en los archivos provistos por la cátedra. Para ejecutar el programa deben escribir:

```
$ ./test_s
```

Para tener una guía, pueden basarse en el archivo `test_list.c` que realiza algunas pruebas sobre las funciones del TAD `list`. Para compilarlo:

```
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -c test_list.c
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -o test_l list-x86_64.o test_list.o
```

Y para ejecutarlo:

```
$ ./test_l
```