# TP Conjunto De Enteros

Cassinerio Marcos

Julio 28, 2020*

*Updated July 28, 2020

# 1 Introducción

El objetivo de este trabajo fue implementar un conjunto de numeros enteros. Adicionalmente implemente un intérprete para poder crear conjuntos y manipularlos a partir de sus funciones basicas.

# 2 Compilado y ejecucion

Para compilar el proyecto abro una terminal, y una vez ubicado en el directorio del proyecto, ejecuto el comando `make`. Esto generará el ejecutable del intérprete.

El mismo lo corro con:
`./interprete`

Este programa permitira ingresar comandos para crear conjuntos y manipularlos con sus funciones. Los comando aceptados son:

- `alias = {1,2,3,...,n}`: crea un conjunto con nombre alias y los elementos dados en el. Ej: `A = {1,2,3,5,8,6,0}`.
- `alias = {var : x <= var <= y}`: crea un conjunto con nombre alias y el intervalo de enteros que va de 'x' a 'y' en el. Ej: `A = {x : 1 <= x <= 5}`.
- `alias1 = alias2 l alias3`: crea un conjunto con nombre alias1 a partir de la union de los conjuntos con nombres alias2 y alias3. Ej: `A = B l C`.
- `alias1 = alias2 & alias3`: crea un conjunto con nombre alias1 a partir de la interseccion de los conjuntos con nombres alias2 y alias3. Ej: `A = B & C`.
- `alias1 = alias2 - alias3`: crea un conjunto con nombre alias1 a partir de la diferencia entre los conjuntos con nombres alias2 y alias3. Ej: `A = B - C`.
- `alias1 = ~ alias2`: crea un conjunto con nombre alias1 a partir del complemento del conjunto con nombre alias2. Ej: `A = ~ B`.
- `salir`: destruye todos los conjuntos creados y termina el programa.

# 3 Formatos aceptados para los alias

Los alias no tienen un formato dado, estos pueden contener cualquier caracter y tampoco tienen un tamano maximor.

# 4 Mensajes de error devueltos por el interprete

Al escribir en la terminal, si se intenta insertar un intervalo por comprension y el intervalo es invalido (el extremo izquierdo es mayor al derecho), se imprimira por pantalla "Intervalo Invalido". En caso de intentar hacer un operacion entre conjuntos, si los conjuntos ingresados no se encuentran en la tablahash, dira por ejemplo "No se encontro el conjunto AB". Finalmente, si alguno de los comandos fue ingresado de manera diferente al formato deseado se imprimira "Formato Incorrecto".

# 5 Organizacion de los archivos

El programa se divide en 4 partes: Interval, Set, TablaHash e intérprete.
Por un lado tenemos la implementación y declaración de Interval en los archivos `interval.c` y `interval.h` respectivamente.
Por otro lado tenemos TablaHash hecho de la misma manera, en los archivos `tablahash.c` y `tablahash.h`.
Luego tenemos Set (conjunto de numeros enteros) que hace uso de las dependecias interval. Su implementación y declaración se encuentra en los archivos `set.c` y `set.h`.
Finalmente tenemos en el archivo `interprete.c` el intéprete que es nuestra interfaz del progragama para manipular los los conjuntos de numeros enteros.

# 6 Implementaciones y estructuras

## 6.1 Interval

La declaración de interval es la siguiente:

```
struct _Interval{
  double extremoIzq;
  double extremoDer;
};

typedef struct _Interval Interval;
```

En su cabecera declaramos las siguientes funciones:

```
interval_crear
interval_destruir
interval_extremo_izq
interval_extremo_der
interval_concat
interval_imprimir
interval_valido
interval_interseccion
interval_comparar
interval_copy
```

Sus implentaciones se encuentran en `interval.c`.

## 6.2 Set

El conjunto de numeros enteros se encuentra definido de la siguiente manera:

```
struct _Set {
    int size;
    Interval **intervalArray;
};

typedef struct _Set *Set;
```

En su archivo cabecera se encuentran declaradas las siguientes funciones:

set_crear
set_destruir
set_copia
set_insertar
set_unir
set_intersecar
set_restar
set_complemento
set_imprimir

Sus implementaciones se encuentran en el archivo `itree.c`, junto con las implementaciones de las funciones:

set_insertar_ultimo
buscar_inicio_interseccion

### 6.3 TablaHash

La tablahash se encuentra definida junto a la casillahash y a linkedList de la siguiente manera:

```
struct _LinkedList {
  CasillaHash casilla;
  struct _LinkedList *ant;
  struct _LinkedList *sig;
};

typedef struct _LinkedList LinkedList;

struct _TablaHash {
  LinkedList **tabla;
  unsigned numElems;
  unsigned capacidad;
  unsigned profundidad;
  FuncionHash hash;
};

typedef struct _TablaHash TablaHash;

struct _CasillaHash {
  char *clave;
  void *dato;
};

typedef struct _CasillaHash CasillaHash;
```

En su archivo cabecera se encuentran declaradas las siguientes funciones:
tablahash_crear
tablahash_insertar
tablahash_buscar
tablahash_destruir_entera

Sus implementaciones se encuentran en el archivo `tablahash.c`, junto con las implementaciones de las funciones:

`linked_list_insertar`
`linked_list_buscar`
`linked_list_eliminar`
`tablahash_destruir`

### 6.4 Interprete

El interprete se encuentra el main del programa, este se encarga de leer la entrada estandar, validarla, ejecutar las funciones de Set e intervalo según la entrada y guardar los conjuntos creados a partir de las funciones de TablaHash. En este archivo estan implementadas las siguientes funciones:

`hash`
`leer_cadena`
`leer_extension`
`leer_comprension`
`obtenerConjuntoDestino`
`obtenerPrimerConjunto`
`obtenerUltimoConjunto`

## 7 Desarrollo y complicaciones

Antes de empezar con la implementacion de conjunto tuve que averiguar como representar los elementos ya que si habia elementos contiguos los deberia representar como un intervalo cuando el usuario quisiera imprimir el conjunto. Debido a esto tuve que que implementar interval antes de seguir con el conjunto.

Luego implemente al conjunto como un arbol AVL ya que me permitia mantener los elementos ordenados e insertarlos de una manera eficiente. Mientras hacia la funcion insertar, note que si habian iontervalos dentro del arbol que intersecaran con el intervalo a borrar, en algunos casos, al intentar borrar dichos nodos el arbol se desbalanceaba ya que la busqueda del nodo a eliminar no comenzaba desde la raiz lo que provocaba que el balanceo no se hiciera en todo el arbol. La manera que se me ocurrio de arreglarlo era haciendo una funcion que elimine de un arbol el primer nodo que interseque con el intervalo proporcionado. La desventaja de esto era que si ese intervalo intersecaba con varios intervalos del arbol, deberia llamar n veces a esta funcion lo que conllevaria reccorer n veces el arbol lo cual seria muy ineficiente. Teniendo en cuenta que varias de las funciones a implementar si no es que todas usarian la funcion insertar (la cual era muy ineficiente) decidi cambiar a otro tipo de implementacion. Opte por implementar al conjunto como un tipo de dato que tuviera un array de intervalos y su tamano. El cual despues de haber investigado note que era mucho mas eficiente que el arbol AVL.

Habiendo terminado el conjunto, comence con la implementacion de la tablahash. Para hacer esta tome como punto de partida el archivo proporcionado en el campus. Primero que nada queria saber como guardaria los conjuntos alli teniendo en cuenta que aplicando la tablahash al nombre de 2 conjuntos puede que obtenga el mismo valor, por los que los conjuntos se guardarian en el mismo lugar. Para resolver ese problema implemente una LinkedList dentro de tablahash, de esta manera, en la posicion de la tablahash donde guardaria los cunjuntos habria una LinkedList con todos los conjuntos que aplicandole la funcion de hasheo retornaran el mismo valor.

Habiendo terminado la implementacion de conjunto y de tablahash, decidi hacer el interprete para poder testear mejor todas las funciones y verificar que funcione el programa. Este me tomo

mucho tiempo ya que habia muchos comandos a implementar. Haber tomado un formato para cada comando, hizo que hacer el interprete no tuviese tantas complicaciones.

# 8 Bibliografia

http://www.cplusplus.com/reference/cstdlib/strtol/