Manual tecnico

En este documento encontrara la informacion basica de como funciona el programa y basarse para modificaciones futuras.

Al inicio del codigo fuente se encuentran tres tipos de estructuras las cuales son para las tablas, columnas y datos de cada columan, son los siguientes.

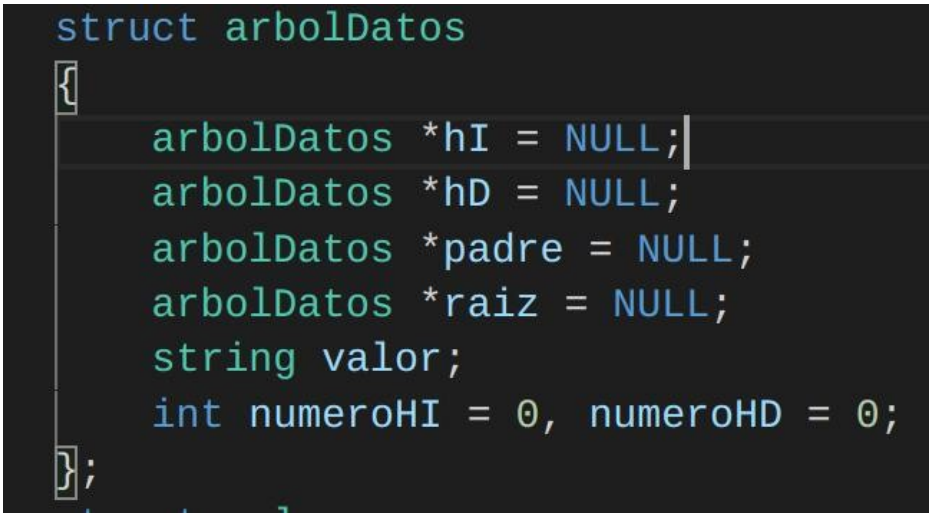
```
struct tabla
{
    tabla *anterior, *siguiente;
    string nombre;
    int numeroColumnas;
    columna *columnas = NULL;
};
```

La estructura de tabla esta enlazada con las tablas siguientes y anteriores, asi mismo tienen un nombre y el numero de columnas por cada tabla, tambien esta enlazada con las columnas que son otro tipo de estructura que se define en la siguiente imangen.

```
struct columna
{
    columna *anterior;
    columna *siuiente;
    int numeroColumnas = 5;
    arbolDatos *arregloArboles [5];
    int posicionFila = 0;
    string nombre;
    double factorCarga;
    double factorCargaPermitido = .6;
    int tipo;
};
```

La estructura columna esta enlzada con otras columnas dentro de la tabla, las cuales tiene nombre, numero de filas iniciales, un tabla hash que inicialmente esta con 5 posisciones que apuntan a los datos contenidos en cada columna, tambien tendra el factor de carga que ira incrementando según se vallan ingresando datos, y el factor de

carga permitido esta definido por el 60%, tambien se define el tipo de dato que sera ingreasdo para poder saber el tipo de dato que almacenara.



```
struct arbolDatos
{
    arbolDatos *hI = NULL;
    arbolDatos *hD = NULL;
    arbolDatos *padre = NULL;
    arbolDatos *raiz = NULL;
    string valor;
    int numeroHI = 0, numeroHD = 0;
};
```

Esta estructura contendra los datos ingresados, los cuales se visualizan de una manera de arbol tipo avl.

- **Funcio hash:** La funcion hash se baso en separar el texto ingresado en caracteres y sacarles su codigo ascii, el cual es sumado y se obtiene el residuo de la division de la suma de los ascci entre la cantidad aceptada de columnas, el cual retornara la posicion en arreglo en donde sera guardado el dato. El algoritmo es el siguiente:

la funcion hash esta diseñada para datos generales en este programa, si desea implementear
datos especificos para un manejo posterior de dichos debera crear mas funciones hash, en esta
ocacion se realizo una suma de los valores ascii de cada entrada evaluando cada entrada en sus
caracteres y sumando el valor ascii de cada caracter y obtendra el valor del residuo entre los
espacios disponibles para la posiscion la cual apunta al arbol que contiene esos datos

```
int ascii = 0;
int retorno = 0;
for (int d = 0; d < entrada.length(); d++)
{
char aux = entrada[d];
```

```
cout << aux << " ";
ascii += (int)aux;
}
retorno = ascii%numeroFilasPermitido;
if(retorno<0)retorno = retorno * -1;
```

- **Algoritmo insertar (avl):**
  - Revisar el arbol
  - si (arbol = NULL)
  - insertar raiz
  - sino si(arbol != NULL)
  - si(datoInsertar > arbol→dato)
  - insertar en lado izquierdo del arbolDato
  - verificar balance
  - salir
  - fin si
  - si (datoInsertar < arbol→dato)
  - insertar en lado derecho de arbolDato
  - verificar balance
  - salir
  - fin si
  - fin si

- **Algoritmo para balance:**
  - **rotacion derecha**
    - entero diferencia = numeroHijosIzquierda - numeroHijosDerecha

- si (diferencia > 0)
    - si(nodo→hijoIzquierda→hijoDerecha != NULL)
    - guardar puntero de arbol→ hijoizQuierdo→hijoDerecho
    - arbol apunta a hijo izquierdo
    - arbol→hijo→ derecho apunta a padre
    - arbol→ hijo derecho → hijo izquierdo apunta al guardado
    - sino
    - guardar padre
    - arbol apunta a hijo izquierdo
    - arbol→ hijo derecho apunta a guardado

- **rotacion izquierda**
    - entero diferencia = numeroHijosIzquierda - numeroHijosDerecha
    - si (diferencia < 0)
        - si(nodo→hijoDerecha→hijoIzquierda!= NULL)
        - guardar puntero de arbol→ hijoDerecha→hijoIzquierda
        - arbol apunta a hijo derecho
        - arbol→hijo→ izquierdo apunta a padre
        - arbol→ hijo izquierdo → hijo derecho apunta al guardado
        - sino
        - guardar padre
        - arbol apunta a hijo derecho
        - arbol→ hijo izquierdo apunta a guardado