MANUAL TÉCNICO

VERSION DE G++:

ii g++ 4:7.4.0-1ubu amd64 GNU C++ compiler
ii g++-7 7.4.0-1ubunt amd64 GNU C++ compiler

IDE donde fue trabajado el proyecto:

Netbeans 8.2

Clases del programa:

.CPP:

AVL.cpp
FuncionesHash.cpp
Graphviz.cpp
ListaTablas.cpp
Nodo.cpp
NodoHash.cpp
OperacionesInsercion.cpp
OperacionesSeleccion.cpp
OrdenamientoSeleccion.cpp
ReposicionamientoNodos.cpp
SeparacionPalabras.cpp
TablaHash.cpp
TextoConsultas,cpp
main.cpp

METODOS CLASE AVL:

```
void Insertar(Nodo*);
void Reposicionamiento(Nodo*);
void Insertar2(Nodo*, Nodo*, Nodo*);
void Borrar(Nodo* nodo);
bool Buscar(Nodo* nodo);
void BuscarEntero(string, Nodo*);
void BuscarDecimal(string, Nodo*);
void BuscarCaracter(string, Nodo*);
void BuscarDato(string, Nodo*, bool, Nodo*, string);
void Set_NoArbol(int);
void BuscarCadena(string, Nodo*, Nodo*, string);
```

```
Nodo* Get_Raiz();
Nodo* Get_Actual();
int Get_NoArbol();
void Set_Raiz(Nodo*);
void Set_Actual(Nodo*);
Nodo* BuscarPorFuncion(int, int);
Nodo* VerificarFuncionesHash(int, int);
bool Vacio(Nodo *nodo);
bool EsHoja(Nodo* nodo) { return !nodo->Get_derecha() && !nodo->Get_izquierda(); }
const int NumeroNodos();
const int AlturaArbol();
int Altura(const int dat);
void Raiz() { actual = raiz; }
void InOrden();
void Equilibrar(Nodo* nodo, int, bool);
void RSI(Nodo* nodo);
void RSD(Nodo* nodo);
void RDI(Nodo* nodo);
void RDD(Nodo* nodo);
void auxContador(Nodo*);
void auxAltura(Nodo*, int);
```

METODOS CLASE FuncionesHash:

```
int valorAscii(char);
int funcionString(string, int);
int funcionEntero(int, int);
int funcionChar(char, int);
int funcionDecimal(double, int);
void reHashing(TablaHash*&);
```

METODOS ClASE Graphviz:

```
void dibujoGrafo(TablaHash*&, string);
string convertirIntString(int);
string convertirDoubleString(double);
string convertirCharString(char);
void grafoGlobal(ListaTablas*&, TablaHash*&, TablaHash*&);
string* InOrden(Nodo*, bool, AVL*, string*, int, int);
void sentenciaNodos(TablaHash*&, NodoHash*);
string* InOrden2(Nodo*, bool, AVL*, string*, int);
string verificarTipo(string, Nodo*);
```

METODOS CLASE ListaTablas:

```
void Set_primero(TablaHash*);
void Set_ultimo(TablaHash*);
void Set_TotalColumnas(int);
void Set_TotalElementos(int);
```

int Get_TotalElementos();
int Get_TotalColumnas();
TablaHash* Get_primero();
TablaHash* Get_ultimo();
void verTotalElementos(TablaHash*&, TablaHash*&);
void verColumnasTotales(TablaHash*&, TablaHash*&);
void insertarTabla(TablaHash*&, TablaHash*&, TablaHash*);
TablaHash* buscarTabla(TablaHash*&, TablaHash*&, string);
ListaTablas(const ListaTablas& orig);

METODOS CLASE Nodo:

void Set_entero(int);
void Set_decimal(double);
void Set_cadena(string);
void Set_caracter(char);
void Set_tipo(string);
void Set_tupla(int);
void Set_izquierda(Nodo*);
void Set_derecha(Nodo*);
void Set_padre(Nodo*);
void Set_FE(int);
void Set_id(int);
void Set_Columna(string);
void Set_Bandera(string);
void Set_NombreTabla(string);
void Set_NoColumna(int);
void Set_TotalColumnas(int);
string Get_Bandera();
string Get_NombreTabla();
int Get_NoColumna();
int Get_TotalColumnas();
int Get_entero();
int Get_id();
string Get_Columna();
double Get_decimal();
string Get_cadena();
int Get_FE();
char Get_caracter();
string Get_tipo();
int Get_tupla();
Nodo* Get_izquierda();
Nodo* Get_derecha();
Nodo* Get_padre();

METODOS CLASE NodoHash:

void Set_id(int);
void Set_Siguiente(NodoHash*);

```
    void Set_Anterior(NodoHash*);
    void Set_AVL(AVL*);
    int Get_id();
    NodoHash* Get_Siguiente();
    NodoHash* Get_Anterior();
    AVL* Get_AVL();
    NodoHash(const NodoHash& orig);
```

METODOS CLASE OperacionesInsecion:

```
    bool verificadorColumnas(string, string*, string*, int, TablaHash*&);
    bool verificadorValores(string*, string*, int, string*, string*, TablaHash*&);
```

METODOS CLASE OperacionesSeleccion:

```
    string verificarTipoColumna(string*, string*, string, int);
    void seleccionarNodos(string, string*, TablaHash*&);
    OperacionesSeleccion(const OperacionesSeleccion& orig);
```

METODOS CLASE OrdenamientoSeleccion:

```
    string* busquedaNodoTablas(string, int, TablaHash*&, string*);
    string* busquedaArboles(string, int, TablaHash*&, NodoHash*, string*);
    string* recorridoInOrden(string, int, Nodo*, bool, AVL*, string*);
    string* busquedaDatos(string, int, Nodo*, string*);
```

METODOS CLASE ReposicionamientoNodos:

```
    Nodo* BuscarPorFuncion(int, int, TablaHash*&, NodoHash*);
    void VerificarFuncionesHash(AVL*, int, int, TablaHash*&, int);
    void InOrden(AVL*);
    static int orden;
    void Buscar(string, Nodo*, bool, Nodo*, string, AVL*, TablaHash*&);
    void BuscarCadena(string, Nodo*, Nodo*, string, AVL*, TablaHash*&);
```

METODOS CLASE SeparacionPalabras:

```
    SeparacionPalabras(const SeparacionPalabras& orig);
    string* separacionColumnas(string, string, string);
    string* separacionColumnas2(string, string, string);
    string* separacionColumnas3(string, string, string);
    string* separacionColumnas4(string, string, string);
    void acumuladorColumnas(TablaHash*&, string);
    int totalColumnas(string, string);
    string quitarParentesisA(string);
    string quitarParentesisB(string);
    string quitarParentesis(string);
```

METODOS CLASE TablaHash:

```
void Set_primero(NodoHash*);
void Set_ultimo(NodoHash*);
void Set_siguiente(TablaHash*);
void Set_anterior(TablaHash*);
void Set_nombreTabla(string);
void Set_Columnas(int);
void Set_NombreColumnas(string);
void Set_TotalNodos(int);
void Set_TotalElementos(int);
void Set_UltimaTupla(int);
void Set_TotalColumnas(int);
void Set_ColumnasInt(int);
void Set_ColumnasString(int);
void Set_ColumnasDouble(int);
void Set_ColumnasChar(int);
int Get_ColumnasInt();
int Get_ColumnasString();
int Get_ColumnasDouble();
int Get_ColumnasChar();
int Get_TotalElementos();
int Get_TotalNodos();
int Get_TotalColumnas();
int Get_UltimaTupla();
string Get_NombreColumna();
int Get_Columnas();
string Get_nombreTabla();
NodoHash* Get_primero();
NodoHash* Get_ultimo();
TablaHash* Get_siguiente();
TablaHash* Get_anterior();
void insertarNodo(NodoHash*);
void crearNodosPrincipales();
NodoHash* buscarNodo(int);
void verificarTotalColumnas(TablaHash*&);
void reposicionamientoNodos(int, TablaHash*&);
```

METODOS CLASE TextoConsultas:

```
TextoConsultas(const TextoConsultas& orig);
string* separacionTexto(string);
int totalPalabras(string);
```