CLASE PARA LA CREACIÓN DE TABLA DE SÍMBOLOS

MANUAL DE USUARIO

INTRODUCCIÓN:

Este documento contiene el manual de usuario correspondiente a la presente librería, que consiste en un Tipo Abstracto de Datos (TAD) para representar la organización de una Tabla de Símbolos necesaria para la construcción de cualquier Compilador. Este TAD está implementado en lenguaje C++ y la organización interna de la Tabla de Símbolos se ha realizado mediante un árbol binario.

FICHEROS ADJUNTOS:

- CArbolSimbolos.h: Fichero que contiene la definición de la clase CArbolSimbolos; dicho fichero será el que los usuarios deberán incluir con `#include "CArbolSimbolos.h"`
- CArbolSimbolos.cpp: Contiene el código que implementa todos los métodos definidos en CArbolSimbolos.h
- CNodoSimbolo.h: Fichero que contiene la definición de la clase CNodoSimbolo
- CNodoSimbolo.cpp: Contiene el código que implementa todos los métodos definidos en CNodoSimbolo.h
- CEntrada.h: Fichero que contiene la definición de la clase CEntrada
- CEntrada.cpp: Contiene el código que implementa todos los métodos definidos en CEntrada.h
- CAtributo.h: Fichero que contiene la definición de la clase CAtributo.
- CAtributo.cpp: Contiene el código que implementa todos los métodos definidos en CAtributo.h
- Manual.doc: Manual de usuario en formato Word (este documento)

CREACIÓN DE UNA NUEVA TABLA DE SÍMBOLOS:

Para la creación de una nueva Tabla de Símbolos primeramente deberemos definir una nueva variable cuyo tipo es CArbolSimbolos, por ejemplo:

```
CArbolSimbolo nuevaTabla;
```

A continuación habrá que invocar al método CrearTabla(), es decir la siguiente sentencia creará una nueva Tabla de Símbolos:

```
nuevaTabla.CrearTabla();
```

RECOMENDACIONES:

Para el correcto funcionamiento del sistema se deberá primeramente crear mediante el uso del método CrearTabla() una nueva Tabla de Símbolos. Deberemos crear la estructura que tendrá cada una de las entradas en función del tipo de éstas, siempre antes de la utilización de los métodos: InsertarTipo (ya que este método asociará a cada entrada una de las estructuras anteriormente definidas), ObtenerTipo, InsertarValor, DevolverValor, InsertarParametro, ObtenerTipoParametro, ObtenerPasoParametro e IntroducirPalabraReservada.

MÉTODOS APORTADOS POR EL TAD:

void CrearTabla (): Crea una nueva Tabla de Símbolos**.**

void EliminarTabla (): Elimina una Tabla de Símbolos liberándose los recursos asignados.

void IntroducirPalabraReservada (char *palabraReservada): Introduce un lexema identificado por el usuario como palabra reservada, asignándole la estructura definida anteriormente a la llamada a esta función, de las palabras reservadas.

int EsPalabraReservada (char *nombre): Comprueba si un determinado lexema esta almacenado en la Tabla de Símbolos y su tipo corresponde con las palabras reservadas.

CNodoSimbolo *InsertarSimbolo (char *lexema): Introduce un nuevo lexema en la Tabla de Símbolos, devolviéndose un puntero a la posición que ocupa en la tabla para posteriores accesos.

void CrearEntrada (TipoEntrada tent, char *nombre, Tipo tipo): Introduce un nuevo atributo a la estructura de cada entrada.

CNodoSimbolo *BuscarSimbolo (char *lexema): Busca un lexema en la Tabla de Símbolos; si lo encuentra devolverá un puntero a la posición en la que se encuentra almacenado.

void InsertarTipo (CNodoSimbolo *simb, TipoEntrada tent): Inserta el tipo a una determinada entrada; en este momento se le asignará a dicha entrada la correspondiente estructura definida por el usuario para ese determinado tipo de entrada.

TipoEntrada ObtenerTipo (CNodoSimbolo *simb): Obtiene el tipo almacenado de una determinada entrada.

void InsertarValor (CNodoSimbolo *simb, char *nombre, int valor): Da valor a un atributo de tipo entero asociado a un lexema almacenado en la posición a la que apunta el parámetro pasado.

void InsertarValor (CNodoSimbolo *simb, char *nombre, char *valor): Da valor a un atributo de tipo cadena asociado a un lexema almacenado en la posición a la que apunta el parámetro pasado.

void InsertarValor (CNodoSimbolo *simb, char *nombre, void *valor): Da valor a un atributo de tipo puntero asociado a un lexema almacenado en la posición a la que apunta el parámetro pasado.

DevolverValor (CNodoSimbolo *simb, char *nombre, int &valor): Devuelve el valor de tipo entero almacenado en el atributo solicitado.

DevolverValor (CNodoSimbolo *simb, char *nombre, punt &valor): Devuelve el valor de tipo puntero almacenado en el atributo solicitado.

DevolverValor (CNodoSimbolo *simb, char *nombre, palabra &valor): Devuelve el valor de tipo cadena almacenado en el atributo solicitado.

void InsertarParametro (CNodoSimbolo *simb, char *nombre, PasoParametro pparam, TipoParam tipo): Inserta un parámetro en un atributo de tipo listaParametros el cual representará los parámetros de una función o procedimiento.

TipoParam ObtenerTipoParametro (CNodoSimbolo *simb, char *nombre, int numeroparam): Obtiene la forma de paso de una parámetro de una función o procedimiento almacenado en la Tabla de Símbolos.

PasoParametro ObtenerPasoParametro (CNodoSimbolo *simb, char *nombre, int numeroparam): Obtiene el tipo de un parámetro de una función o procedimiento almacenado en la Tabla de Símbolos.

void ImprimirTabla (char *salida): Imprime el contenido de la Tabla de Símbolos en el fichero especificado.

TIPOS ENUMERADOS:

Los tipos enumerados definidos son los siguientes:

- Enumerado que representa los posibles valores del paso de un parámetro:

enum PasoParametro {valor=0, referencia};

- Enumerado que representa los posibles tipos de un parámetro:

enum TipoParam {ent, cad, real, vec, clas, reg, ptr};

- Enumerado que representa los posibles tipos de un atributo:

enum Tipo {entero=0, cadena, puntero, listaParametros};

- Enumerado que representa los diferentes posibles tipos que podrá tomar un identificador:

enum TipoEntrada {funcion=0, identificador, clase, procedimiento, etiqueta, registro, palabraReserv, vector};

DESCRIPCIÓN DE LOS MÉTODOS PÚBLICOS APORTADOS:

1. CrearTabla ()

PROPÓSITO: Crea una nueva Tabla de Símbolos.
PARÁMETROS: No.
SALIDA: No.
EXCEPCIONES: No provoca ningún tipo de excepción.

EJEMPLO.

```
void main (void)
{
    CArbolSimbolos tabla;

    tabla.CrearTabla();
}
```

2. EliminarTabla ()

PROPÓSITO: Elimina una Tabla de Símbolos.
PARÁMETROS: No.
SALIDA: No.
EXCEPCIONES: No provoca ningún tipo de excepción.

EJEMPLO.

```
void main (void)
{
    CArbolSimbolos tabla;

    tabla.CrearTabla();
    tabla.EliminarTabla();
}
```

3. void ImprimirTabla (char *salida)

PROPÓSITO: Imprime una Tabla de Símbolos en un fichero de salida; dicho fichero de salida será creado en el caso de no existir. El contenido de cada entrada de la Tabla de Símbolos será impreso en una línea diferente del fichero de salida.
PARÁMETROS: Nombre del fichero de salida en el que se imprimirá el contenido de la Tabla de Símbolos.
SALIDA: No.
EXCEPCIONES: Se producirá una excepción en el caso de que el fichero de salida no pueda ser abierto o generado. Su formato es *EXCEPCION En ImprimirTabla: Error abriendo fichero de salida.*

EJEMPLO.

```
void main (void)
{
    CArbolSimbolos tabla;
    CNodoSimbolo *nodo, *nodo1, *nodo2, *nodo3, *nodo4, *nodo5, *nodo6, *nodo7, *nodo8, *nodo9;

    tabla.CrearTabla();

    nodo=tabla.InsertarSimbolo ("suma");
    nodo1=tabla.InsertarSimbolo ("resta");
    nodo2=tabla.InsertarSimbolo ("resultado");
    nodo3=tabla.InsertarSimbolo ("encontrado");
    nodo4=tabla.InsertarSimbolo ("producto");
    nodo5=tabla.InsertarSimbolo ("ptr");
    nodo6=tabla.InsertarSimbolo ("salto");
    nodo7=tabla.InsertarSimbolo ("vector");
```

```
    nodo8=tabla.InsertarSimbolo ("escribir");
    nodo9=tabla.InsertarSimbolo ("palabra");

    tabla.CrearEntrada (identificador,"posicionmemoria",puntero);
    tabla.CrearEntrada (identificador,"inicializada",entero);

    tabla.CrearEntrada (vector,"tipoelemento",cadena);
    tabla.CrearEntrada (vector,"posicion",entero);
    tabla.CrearEntrada (vector,"numeroelementos",entero);
    tabla.CrearEntrada (vector,"indiceinf",entero);
    tabla.CrearEntrada (vector,"indicesup",entero);

    tabla.CrearEntrada (funcion,"numparam",entero);
    tabla.CrearEntrada (funcion,"tiporetorno",cadena);
    tabla.CrearEntrada (funcion,"parametros",listaParametros);
    tabla.CrearEntrada (funcion,"direccion",puntero);

    tabla.CrearEntrada (procedimiento,"numparam",entero);
    tabla.CrearEntrada (procedimiento,"parametros",listaParametros);
    tabla.CrearEntrada (procedimiento,"direccion",puntero);

    tabla.CrearEntrada (etiqueta,"posicionsalto",puntero);

    tabla.CrearEntrada (registro,"posicion",puntero);

    tabla.InsertarTipo (nodo,funcion);
    tabla.InsertarTipo (nodo1,funcion);
    tabla.InsertarTipo (nodo2,identificador);
    tabla.InsertarTipo (nodo3,identificador);
    tabla.InsertarTipo (nodo4,funcion);
    tabla.InsertarTipo (nodo5,identificador);
    tabla.InsertarTipo (nodo6,etiqueta);
    tabla.InsertarTipo (nodo7,vector);
    tabla.InsertarTipo (nodo8,procedimiento);
    tabla.InsertarTipo (nodo9,identificador);

    tabla.IntroducirPalabraReservada ("while");
    tabla.IntroducirPalabraReservada ("for");
    tabla.IntroducirPalabraReservada ("if");
    tabla.IntroducirPalabraReservada ("then");
    tabla.IntroducirPalabraReservada ("break");
    tabla.IntroducirPalabraReservada ("case");
    tabla.IntroducirPalabraReservada ("define");

    tabla.ImprimirTabla ("salida.txt");
}
```

El contenido de salida.txt para este ejemplo sería:

```
IMPRESION DE LA TABLA DE SÍMBOLOS
suma:direccion(0x00000000)->parametros( NULL )->tiporetorno()->numparam( )->NULL
resta:direccion(0x00000000)->parametros( NULL )->tiporetorno()->numparam( )->NULL
encontrado:inicializada( )->posicionmemoria(0x00000000)->NULL
break:NULL
case:NULL
define:NULL
producto:direccion(0x00000000)->parametros( NULL )->tiporetorno()->numparam( )->NULL
escribir:direccion(0x00000000)->parametros( NULL )->numparam( )->NULL
palabra:inicializada( )->posicionmemoria(0x00000000)->NULL
for:NULL
```

if:NULL
ptr:inicializada( )->posicionmemoria(0x00000000)->NULL
resultado:inicializada( )->posicionmemoria(0x00000000)->NULL
salto:posicionsalto(0x00000000)->NULL
vector:indicesup( )->indiceinf( )->numeroelementos( )->posicion( )->tipoelemento()->NULL
then:NULL
while:NULL

4. CNodoSimbolo *InsertarSimbolo (char *lexema)

PROPÓSITO: Inserta un nuevo lexema en la Tabla de Símbolos.
PARÁMETROS: Lexema que se almacenará.
SALIDA: Puntero a la posición en la que se ha almacenado dicho lexema.
EXCEPCIONES: Generará dos posibles excepciones:

- *EXCEPCION En InsertarSimbolo: Este lexema ya esta insertado*: Provocada por la existencia en la Tabla de Símbolos de ese mismo lexema.
- *EXCEPCION En InsertarSimbolo: No hay memoria suficiente*: Se produce cuando no hay recursos para almacenar un nuevo lexema.

EJEMPLO.

```
void main (void)
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo, *nodo1;

 tabla.CrearTabla();

 nodo=tabla.InsertarSimbolo("suma");
 nodo1=tabla.InsertarSimbolo("resta");
}
```

EJEMPLO.

```
void main (void)
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo, *nodo1;

 tabla.CrearTabla();

 nodo=tabla.InsertarSimbolo("suma");
 nodo1=tabla.InsertarSimbolo("suma");
}
```

Produce en la salida:
EXCEPCION En InsertarSimbolo: Este lexema ya esta insertado

5. void IntroducirPalabraReservada (char *palabraReservada)

PROPÓSITO: Introducir en la Tabla de Símbolos un lexema y asociarle el tipo de palabra reservada.
PARÁMETROS: Lexema que se introducirá como palabra reservada.
SALIDA: No.
EXCEPCIONES:

- *EXCEPCION En InsertarSimbolo: Este lexema ya esta insertado:* Provocada por la existencia en la Tabla de Símbolos de ese mismo lexema.
- *EXCEPCION En InsertarSimbolo: No hay memoria suficiente:* Se produce cuando no hay recursos para almacenar un nuevo lexema.

EJEMPLO.

```
void main (void)
{
  CArbolSimbolos tabla;

  tabla.CrearTabla();
  tabla.IntroducirPalabraReservada("while");
  tabla.IntroducirPalabraReservada("for");
}
```

EJEMPLO.

```
void main (void)
{
   CArbolSimbolos tabla;

  tabla.CrearTabla();
  tabla.IntroducirPalabraReservada("while");
  tabla.IntroducirPalabraReservada("while");
}
```

Produce en la salida:
EXCEPCION En InsertarSimbolo: Este lexema ya esta insertado

6. int EsPalabraReservada (char *lexema)

PROPÓSITO: Conocer si un determinado lexema pasado como parámetro está almacenado en la Tabla de Símbolos como palabra reservada.
PARÁMETROS: Lexema sobre el que se realizará la consulta.
SALIDA: Un valor entero de forma que si el valor devuelto es un cero se interpretará como que no es una palabra reservada; en caso contrario (uno), se considerará como tal.
EXCEPCIONES: No se produce.

EJEMPLO.

```
void main (void)
{
   CArbolSimbolos tabla;

  tabla.CrearTabla();
  tabla.IntroducirPalabraReservada("while");
  tabla.IntroducirPalabraReservada("for");

if (tabla.EsPalabraReservada("while"))
     cout<<"Es palabra reservada"<<endl;
  else
     cout<<"No es palabra reservada"<<endl;
}
```

7. void CrearEntrada (TipoEntrada tent, char *nombre, Tipo tipo)

PROPÓSITO: Crear la estructura para los diferentes tipos de posibles entradas; estas estructuras serán creadas por el usuario mediante sucesivas llamadas a este método.
PARÁMETROS:

-tent: parámetro de tipo TipoEntrada; dicho tipo es un tipo enumerado el cual toma los siguientes posibles valores: funcion, identificador, clase, procedimiento,etiqueta, registro, palabraReserv, vector.

-nombre: parámetro de tipo cadena de caracteres, que representa el nombre del atributo que estamos añadiendo a la estructura para un determinado tipo de entrada.

-tipo: Tipo de datos que almacenará dicho atributo, sus posibles valores son: entero, cadena, puntero, listaParametros.

SALIDA: No.

EXCEPCIONES:

- *EXCEPCION En CrearEntrada: No hay memoria suficiente:* Se produce cuando no hay recursos para crear la nueva entrada.
- *EXCEPCION En CrearEntrada: Tipo del atributo no definido:* El parámetro tipo no tiene un valor correcto.
- *EXCEPCION En CrearEntrada: Tipo de entrada no definido:* el parámetro tent no tiene un valor correcto.

EJEMPLO:

```
void main (void)
{

   CArbolSimbolos tabla;

   tabla.CrearTabla();

  tabla.CrearEntrada(identificador,"posicionmemoria",puntero);
  tabla.CrearEntrada(identificador,"inicializada",entero);

  tabla.CrearEntrada(vector,"tipoelemento",cadena);
  tabla.CrearEntrada(vector,"posicion",entero);
  tabla.CrearEntrada(vector,"numeroelementos",entero);
  tabla.CrearEntrada(vector,"indiceinf",entero);
  tabla.CrearEntrada(vector,"indicesup",entero);

 tabla.CrearEntrada(funcion,"numparam",entero);
  tabla.CrearEntrada(funcion,"tiporetorno",cadena);
  tabla.CrearEntrada(funcion,"parametros",listaParametros);
  tabla.CrearEntrada(funcion,"direccion",puntero);

  tabla.CrearEntrada(procedimiento,"numparam",entero);
  tabla.CrearEntrada(procedimiento,"parametros",listaParametros);
  tabla.CrearEntrada(procedimiento,"direccion",puntero);

  tabla.CrearEntrada(etiqueta,"posicionsalto",puntero);

  tabla.CrearEntrada(registro,"posicion",puntero);
}
```

Este ejemplo crea los atributos “posicionmemoria” e “inicializada” para las entradas de tipo “identificador. Crea los atributos “tipoelemento”, “posicion”, “numeroelementos”, “indiceinf” e “indicesup” para las entradas de tipo “vector”, etc. Además, se define el tipo de datos que se almacenará en cada uno de esos atributos.

Es importante hacer notar que para poder utilizar una Tabla de Símbolos, en primer lugar es necesario definir su estructura.

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
```

```
  tabla.CrearTabla();
  tabla.CrearEntrada(identificador,"posicionmemoria",9);
  tabla.CrearEntrada(identificador,"inicializada",entero);
}
```

Produce en la salida:
EXCEPCION En CrearEntrada: Tipo del atributo no definido

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;

  tabla.CrearTabla();
  tabla.CrearEntrada(identificador,"posicionmemoria",9);
  tabla.CrearEntrada(12,"inicializada",entero);
}
```

Produce en la salida:
EXCEPCION En CrearEntrada: Tipo del atributo no definido
Produce en la salida:
EXCEPCION En CrearEntrada: Tipo de entrada no definido

8. CNodoSimbolo BuscarSimbolo (char *lex)

PROPÓSITO: Buscar la posición de la Tabla de Símbolos en la que está almacenado el lexema pasado como parámetro.
PARÁMETROS: Lexema que se buscará.
SALIDA: La posición en la que se encuentra almacenada el lexema, o NULL en caso de no encontrarse almacenado en la Tabla de Símbolos.
EXCEPCIONES: No se producen.

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
   CNodoSimbolo *nodo;
   CNodoSimbolo *pos;

  tabla.CrearTabla();

  nodo=tabla.InsertarSimbolo("escribir");
  tabla.CrearEntrada(procedimiento,"numparam",entero);
  tabla.CrearEntrada(procedimiento,"parametros",listaParametros);
  tabla.CrearEntrada(procedimiento,"direccion",puntero);

  tabla.InsertarTipo(nodo,procedimiento);

  pos=tabla.BuscarSimbolo("escribir");
  tabla.ObtenerTipo(pos);
}
```

9. void InsertarTipo (CNodoSimbolo *simb, TipoEntrada tent)

PROPÓSITO: Asignar un tipo a una determinada entrada en la que está almacenado un lexema; en dicho momento será asignado a dicha entrada la estructura que el usuario había definido para ese determinado tipo de entrada.

PARÁMETROS:

-simb: Posición de la Tabla de Símbolos en la que se encuentra almacenado el lexema al que se le quiere asignar un tipo.

-tent: parámetro de tipo TipoEntrada; dicho tipo es un tipo enumerado el cual toma los siguientes posibles valores: funcion, identificador, clase, procedimiento, etiqueta, registro, palabraReserv, vector.

SALIDA: No.

EXCEPCIONES:

- *EXCEPCION En InsertarTipo: TipoEntrada no definido:*el parámetro tent no tiene un valor correcto.

EJEMPLO:

```
void main (void)
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo, *nodo1, *nodo2, *nodo3, *nodo4, *nodo5, *nodo6, *nodo7, *nodo8, *nodo9;

 tabla.CrearTabla();

 nodo=tabla.InsertarSimbolo("suma");
 nodo1=tabla.InsertarSimbolo("resta");
 nodo2=tabla.InsertarSimbolo("resultado");
 nodo3=tabla.InsertarSimbolo("encontrado");
 nodo4=tabla.InsertarSimbolo("producto");
 nodo5=tabla.InsertarSimbolo("ptr");
 nodo6=tabla.InsertarSimbolo("salto");
 nodo7=tabla.InsertarSimbolo("vector");
 nodo8=tabla.InsertarSimbolo("escribir");
 nodo9=tabla.InsertarSimbolo("palabra");

 tabla.CrearEntrada(identificador,"posicionmemoria",puntero);
 tabla.CrearEntrada(identificador,"inicializada",entero);

 tabla.CrearEntrada(vector,"tipoelemento",cadena);
 tabla.CrearEntrada(vector,"posicion",entero);
 tabla.CrearEntrada(vector,"numeroelementos",entero);
 tabla.CrearEntrada(vector,"indiceinf",entero);
 tabla.CrearEntrada(vector,"indicesup",entero);

 tabla.CrearEntrada(funcion,"numparam",entero);
 tabla.CrearEntrada(funcion,"tiporetorno",cadena);
 tabla.CrearEntrada(funcion,"parametros",listaParametros);
 tabla.CrearEntrada(funcion,"direccion",puntero);

 tabla.CrearEntrada(procedimiento,"numparam",entero);
 tabla.CrearEntrada(procedimiento,"parametros",listaParametros);
 tabla.CrearEntrada(procedimiento,"direccion",puntero);

 tabla.CrearEntrada(etiqueta,"posicionsalto",puntero);

 tabla.CrearEntrada(registro,"posicion",puntero);

 tabla.InsertarTipo(nodo,funcion);
 tabla.InsertarTipo(nodo1,funcion);
 tabla.InsertarTipo(nodo2,identificador);
 tabla.InsertarTipo(nodo3,identificador);
 tabla.InsertarTipo(nodo4,funcion);
 tabla.InsertarTipo(nodo5,identificador);
 tabla.InsertarTipo(nodo6,etiqueta);
```

```
 tabla.InsertarTipo(nodo7,vector);
 tabla.InsertarTipo(nodo8,procedimiento);
 tabla.InsertarTipo(nodo9,identificador);

 tabla.Imprimir("sal.txt");
}
```

En el fichero sal.txt podemos ver como se ha asignado la estructura creada a cada lexema:

```
IMPRESION DE LA TABLA DE SÍMBOLOS
suma:direccion(0x00000000)->parametros( NULL )->tiporetorno()->numparam( )->NULL
resta:direccion(0x00000000)->parametros( NULL )->tiporetorno()->numparam( )->NULL
encontrado:inicializada( )->posicionmemoria(0x00000000)->NULL
producto:direccion(0x00000000)->parametros( NULL )->tiporetorno()->numparam( )->NULL
escribir:direccion(0x00000000)->parametros( NULL )->numparam( )->NULL
palabra:inicializada( )->posicionmemoria(0x00000000)->NULL
ptr:inicializada( )->posicionmemoria(0x00000000)->NULL
resultado:inicializada( )->posicionmemoria(0x00000000)->NULL
salto:posicionsalto(0x00000000)->NULL
vector:indicesup( )->indiceinf( )->numeroelementos( )->posicion( )->tipoelemento()->NULL
```

10. TipoEntrada ObtenerTipo (CNodoSimbolo *simb)

PROPÓSITO: Obtener el tipo que habia sido introducido para un determinado lexema.
PARÁMETROS: simb: Posición de la Tabla de Símbolos en la que se encuentra almacenado el lexema del que queremos obtener un tipo.
SALIDA: resultado TipoEntrada que representa el tipo almacenado para el lexema de esa determinada entrada.
EXCEPCIONES:

- *EXCEPCION En ObtenerTipo: Dicha entrada no tiene introducido ningun tipo:* No hay ningun tipo almacenado en la entrada especificada como parámetro.

EJEMPLO:

```
void main (void)
{
  CArbolSimbolos tabla;
 TipoEntrada tent;
 CNodoSimbolo *nodo;

 tabla.CrearTabla();

 nodo=tabla.InsertarSimbolo("resultado");
 tabla.CrearEntrada(identificador,"posicionmemoria",puntero);
 tabla.CrearEntrada(identificador,"inicializada",entero);
 tabla.InsertarTipo(nodo,identificador);
 tabla.ObtenerTipo(nodo);
}
```

EJEMPLO:

```
void main (void)
{
  CArbolSimbolos tabla;
 TipoEntrada tent;
 CNodoSimbolo *nodo;

 tabla.CrearTabla();

 nodo=tabla.InsertarSimbolo("vector");
```

```
  tabla.CrearEntrada(vector,"tipoelemento",cadena);
  tabla.CrearEntrada(vector,"posicion",entero);
  tabla.CrearEntrada(vector,"numeroelementos",entero);
  tabla.CrearEntrada(vector,"indiceinf",entero);
  tabla.CrearEntrada(vector,"indicesup",entero);
  tabla.ObtenerTipo(nodo);
}
```

Produce en la salida:
EXCEPCION En ObtenerTipo: Dicha entrada no tiene introducido ningun tipo

11. void InsertarValor (CNodoSimbolo *simb, char *nombre, int valor),
12. void InsertarValor (CNodoSimbolo *simb, char *nombre, char *valor),
13. void InsertarValor (CNodoSimbolo *simb, char *nombre, void *valor)

PROPÓSITO: Con estos tres métodos sobrecargados insertaremos un valor de tipo entero, cadena o puntero a un atributo de una determinada entrada.
PARÁMETROS:
- simb: Posición de la Tabla de Símbolos en la que se encuentra almacenado el lexema al que le vamos a dar valor.
- nombre. Nombre del atributo al que daremos valor.
- valor: valor que almacenaremos para el atributo.

SALIDA: No.
EXCEPCIONES:
- *EXCEPCION En InsertarValor: El atributo nombre no pertenece a esa entrada:* No puede insertar valor a un atributo que dicha entrada no contiene.

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
   CNodoSimbolo *nodo;

  tabla.CrearTabla();

  nodo=tabla.InsertarSimbolo("resta");
  tabla.CrearEntrada(funcion,"numparam",entero);
  tabla.CrearEntrada(funcion,"tiporetorno",cadena);
  tabla.CrearEntrada(funcion,"parametros",listaParametros);
  tabla.CrearEntrada(funcion,"direccion",puntero);
  tabla.InsertarTipo(nodo,funcion);
  tabla.InsertarValor(nodo,"numparam",2);
  tabla.InsertarValor(nodo,"tiporetorno","entero");
  valor_d=(void *)nodo;
  tabla.InsertarValor(nodo,"direccion",valor_d);

  tabla.ImprimirTabla("salida.txt");
}
```

En salida.txt podemos comprobar los resultados de insertar valor a dichos atributos:

```
IMPRESION DE LA TABLA DE SÍMBOLOS
resta:direccion(0x00b81108)->parametros(NULL)->tiporetorno(entero)->numparam(2)->NULL
```

EJEMPLO:

```
void main (void)
```

```
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo;

 tabla.CrearTabla();
 nodo=tabla.InsertarSimbolo("resta");
 tabla.CrearEntrada(funcion,"numparam",entero);
 tabla.CrearEntrada(funcion,"tiporetorno",cadena);
 tabla.CrearEntrada(funcion,"parametros",listaParametros);
 tabla.CrearEntrada(funcion,"direccion",puntero);
 tabla.InsertarTipo(nodo,funcion);
 tabla.InsertarValor(nodo,"num",2);
 tabla.InsertarValor(nodo,"tipo","entero");
 valor_d=(void *)nodo;
 tabla.InsertarValor(nodo,"dir",valor_d);

 tabla.ImprimirTabla("salida.txt");
}
```

Obtenemos en la salida:
EXCEPCION En InsertarValor: El atributo num no pertenece a esa entrada
EXCEPCION En InsertarValor: El atributo tipo no pertenece a esa entrada
EXCEPCION En InsertarValor: El atributo dir no pertenece a esa entrada

14. DevolverValor (CNodoSimbolo *simb, char *nombre, int &valor)
15. DevolverValor (CNodoSimbolo *simb, char *nombre, punt &valor)
16. DevolverValor (CNodoSimbolo *simb, char *nombre, palabra &valor)

PROPÓSITO: Con estos tres métodos sobrecargados obtendremos el valor de un atributo de una determinada entrada.
PARÁMETROS:
- simb: Posición de la Tabla de Símbolos en la que se encuentra almacenado el lexema que queremos consultar.
- nombre. Nombre del atributo cuyo valor queremos consultar.
- valor: parámetro de salida en el que almacenaremos el valor para el atributo especificado.

SALIDA: No.
EXCEPCIONES:
- *EXCEPCION En ObtenerValor: El atributo nombre no pertenece a esa entrada:* No puede obtener el valor de un atributo que dicha entrada no contiene.
- *EXCEPCION En DevolverValor: Este atributo no tiene valor alguno:* Se produce al intentar conseguir el contenido de un atributo el cual no tiene valor alguno introducido.

EJEMPLO:

```
void main (void)
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo, *nodo1, *nodo2, *nodo3, *nodo4, *nodo5, *nodo6, *nodo7, *nodo8, *nodo9;

 tabla.CrearTabla();

 nodo=tabla.InsertarSimbolo("suma");
 nodo1=tabla.InsertarSimbolo("resta");
 nodo2=tabla.InsertarSimbolo("resultado");
 nodo3=tabla.InsertarSimbolo("encontrado");
 nodo4=tabla.InsertarSimbolo("producto");
 nodo5=tabla.InsertarSimbolo("ptr");
 nodo6=tabla.InsertarSimbolo("salto");
 nodo7=tabla.InsertarSimbolo("vector");
```

```
  nodo8=tabla.InsertarSimbolo("escribir");
  nodo9=tabla.InsertarSimbolo("palabra");

  tabla.CrearEntrada(identificador,"posicionmemoria",puntero);
  tabla.CrearEntrada(identificador,"inicializada",entero);
  tabla.CrearEntrada(vector,"tipoelemento",cadena);
  tabla.CrearEntrada(vector,"posicion",entero);
  tabla.CrearEntrada(vector,"numeroelementos",entero);
  tabla.CrearEntrada(vector,"indiceinf",entero);
  tabla.CrearEntrada(vector,"indicesup",entero);
  tabla.CrearEntrada(funcion,"numparam",entero);
  tabla.CrearEntrada(funcion,"tiporetorno",cadena);
  tabla.CrearEntrada(funcion,"parametros",listaParametros);
  tabla.CrearEntrada(funcion,"direccion",puntero);
  tabla.CrearEntrada(procedimiento,"numparam",entero);
  tabla.CrearEntrada(procedimiento,"parametros",listaParametros);
  tabla.CrearEntrada(procedimiento,"direccion",puntero);
  tabla.CrearEntrada(etiqueta,"posicionsalto",puntero);
  tabla.CrearEntrada(registro,"posicion",puntero);

  tabla.InsertarTipo(nodo,funcion);
  tabla.InsertarTipo(nodo1,funcion);
  tabla.InsertarTipo(nodo2,identificador);
  tabla.InsertarTipo(nodo3,identificador);
  tabla.InsertarTipo(nodo4,funcion);
  tabla.InsertarTipo(nodo5,identificador);
  tabla.InsertarTipo(nodo6,etiqueta);
  tabla.InsertarTipo(nodo7,vector);
  tabla.InsertarTipo(nodo8,procedimiento);
  tabla.InsertarTipo(nodo9,identificador);

  tabla.DevolverValor(nodo7,"numeroelementos",resultado);
  tabla.DevolverValor(nodo2,"inicializada",resultado);
  tabla.DevolverValor(nodo1,"direccion",resultadoptr);
  tabla.DevolverValor(nodo,"tiporetorno",resultadocad);
}
```

RESULTADO OBTENIDO:

```
EXCEPCION En DevolverValor: Este atributo no tiene valor alguno
EXCEPCION En DevolverValor: Este atributo no tiene valor alguno
EXCEPCION En DevolverValor: Este atributo no tiene valor alguno
EXCEPCION En DevolverValor: Este atributo no tiene valor alguno
```

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
   CNodoSimbolo *nodo, *nodo1, *nodo2, *nodo3, *nodo4, *nodo5, *nodo6, *nodo7, *nodo8, *nodo9;

  tabla.CrearTabla();

  nodo=tabla.InsertarSimbolo("suma");
  nodo1=tabla.InsertarSimbolo("resta");
  nodo2=tabla.InsertarSimbolo("resultado");
  nodo3=tabla.InsertarSimbolo("encontrado");
  nodo4=tabla.InsertarSimbolo("producto");
```

```
 nodo5=tabla.InsertarSimbolo("ptr");
 nodo6=tabla.InsertarSimbolo("salto");
 nodo7=tabla.InsertarSimbolo("vector");
 nodo8=tabla.InsertarSimbolo("escribir");
 nodo9=tabla.InsertarSimbolo("palabra");

 tabla.CrearEntrada(identificador,"posicionmemoria",puntero);
 tabla.CrearEntrada(identificador,"inicializada",entero);
 tabla.CrearEntrada(vector,"tipoelemento",cadena);
 tabla.CrearEntrada(vector,"posicion",entero);
 tabla.CrearEntrada(vector,"numeroelementos",entero);
 tabla.CrearEntrada(vector,"indiceinf",entero);
 tabla.CrearEntrada(vector,"indicesup",entero);
 tabla.CrearEntrada(funcion,"numparam",entero);
 tabla.CrearEntrada(funcion,"tiporetorno",cadena);
 tabla.CrearEntrada(funcion,"parametros",listaParametros);
 tabla.CrearEntrada(funcion,"direccion",puntero);
 tabla.CrearEntrada(procedimiento,"numparam",entero);
 tabla.CrearEntrada(procedimiento,"parametros",listaParametros);
 tabla.CrearEntrada(procedimiento,"direccion",puntero);
 tabla.CrearEntrada(etiqueta,"posicionsalto",puntero);
 tabla.CrearEntrada(registro,"posicion",puntero);

 tabla.InsertarTipo(nodo,funcion);
 tabla.InsertarTipo(nodo1,funcion);
 tabla.InsertarTipo(nodo2,identificador);
 tabla.InsertarTipo(nodo3,identificador);
 tabla.InsertarTipo(nodo4,funcion);
 tabla.InsertarTipo(nodo5,identificador);
 tabla.InsertarTipo(nodo6,etiqueta);
 tabla.InsertarTipo(nodo7,vector);
 tabla.InsertarTipo(nodo8,procedimiento);
 tabla.InsertarTipo(nodo9,identificador);

 tabla.InsertarValor(nodo,"tiporetorno","entero");
 valor_d=(void *)nodo1;
 tabla.InsertarValor(nodo1,"direccion",valor_d);
 tabla.InsertarValor(nodo2,"inicializada",0);
 tabla.InsertarValor(nodo7,"numeroelementos",10);
 tabla.InsertarValor(nodo6,"posicionsalto",nodo3);

 tabla.DevolverValor(nodo7,"numero",resultado);
 tabla.DevolverValor(nodo2,"ini",resultado);
 tabla.DevolverValor(nodo1,"dire",resultadoptr);
 tabla.DevolverValor(nodo,"retorno",resultadocad);
}
```

RESULTADO OBTENIDO:

```
EXCEPCION En DevolverValor: El atributo numero no pertenece a esa entrada
EXCEPCION En DevolverValor: El atributo ini no pertenece a esa entrada
EXCEPCION En DevolverValor: El atributo dire no pertenece a esa entrada
EXCEPCION En DevolverValor: El atributo retorno no pertenece a esa entrada
```

17. void InsertarParametro (CNodoSimbolo *simb, char *nombre, PasoParametro pparam, TipoParam tipo)

PROPÓSITO: Con este método almacenaremos un nuevo parámetro al que dotaremos de tipo y forma de paso en una entrada, dicha entrada contendrá un lexema de tipo función o prodecimiento que posee un atributo de tipo listaParametros.
PARÁMETROS:
- simb: Posición de la Tabla de Símbolos en la que está almacenado el lexema al cual se le añade un parámetro.
- nombre. Nombre del atributo listaParametros donde queremos insertar un parámetro.
- pparam: Forma de paso del parametro (valor o referencia).
- tipo: Tipo del parámetro (ent, cad, real, vec, clas, reg, ptr).

SALIDA: No
EXCEPCIONES:
- *EXCEPCION En InsertarParametro: El atributo nombre no pertenece a esa entrada:* No puede insertar un parámetro en un atributo que dicha entrada no contiene.
- *EXCEPCION En Insertar Parametro: No hay memoria suficiente:* Se produce cuando no hay recursos para almacenar un nuevo parámetro.

EJEMPLO:

```
void main (void)
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo1;

 tabla.CrearTabla();

 nodo1=tabla.InsertarSimbolo("resta");
 tabla.CrearEntrada(funcion,"numparam",entero);
 tabla.CrearEntrada(funcion,"tiporetorno",cadena);
 tabla.CrearEntrada(funcion,"parametros",listaParametros);
 tabla.CrearEntrada(funcion,"direccion",puntero);
 tabla.InsertarTipo(nodo1,funcion);

 tabla.InsertarParametro(nodo1,"parametros",valor,real);
 tabla.InsertarParametro(nodo1,"parametr",valor,real);
}
```

RESULTADO OBTENIDO:

EXCEPCION En InsertarParametro: El atributo parametr no pertenece a esa entrada

18. TipoParam ObtenerTipoParametro (CNodoSimbolo *simb, char *nombre, int numeroparam)

PROPÓSITO: Con este método obtendremos el tipo de un parámetro que hemos almacenado para un lexema de tipo función o procedimiento.
PARÁMETROS:
- simb: Posición de la Tabla de Símbolos en la que se encuentra almacenado el lexema que tiene un parámetro cuyo tipo deseamos consultar.
- nombre. Nombre del atributo listaParametros que contiene el parámetro cuyo tipo queremos consultar.
- numeroparam: Orden del parámetro requerido en la lista de parámetros almacenada.

SALIDA: TipoParam del parámetro requerido.
EXCEPCIONES:
- *EXCEPCION En ObtenerTipoParametro: El atributo nombre no pertenece a esa entrada:* No puede obtener el parámetro almacenado en un atributo que dicha entrada no contiene.
- *EXCEPCION En ObtenerTipoParametro: El parametro numeroparam no existe:* El número de parámetro especificado sobrepasa al número de parámetros almacenados para ese atributo.

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
   CNodoSimbolo *nodo;

  tabla.CrearTabla();
  nodo=tabla.InsertarSimbolo("suma");
  tabla.InsertarTipo(nodo,funcion);

  tabla.ObtenerTipoParametro(nodo,"parametros",2);
  tabla.ObtenerTipoParametro(nodo,"parametros",1);
}
```

RESULTADO OBTENIDO:

EXCEPCION En ObtenerTipoParametro: El parametro 2 no existe
EXCEPCION En ObtenerTipoParametro: El parametro 1 no existe

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
   CNodoSimbolo *nodo;

  tabla.CrearTabla();
  nodo=tabla.InsertarSimbolo("suma");
  tabla.InsertarTipo(nodo,funcion);
  tabla.InsertarValor(nodo,"numpara",2);
  tabla.InsertarValor(nodo,"tiporetorn","entero");
  tabla.InsertarParametro(nodo,"parametros",valor,ptr);
  tabla.InsertarParametro(nodo,"parametros",referencia,ent);

  tabla.ObtenerTipoParametro(nodo,"parametros",2);
  tabla.ObtenerTipoParametro(nodo,"parametros",1);
}
```

19. PasoParametro ObtenerPasoParametro (CNodoSimbolo *simb, char *nombre, int numeroparam)

PROPÓSITO: Con este método obtendremos el tipo de paso (valor o refernecia) de un parámetro que hemos almacenado para un lexema de tipo función o procedimiento.
PARÁMETROS:
- simb: Posición de la Tabla de Símbolos en la que está el lexema cuya información queremos consultar.
- nombre. Nombre del atributo listaParametros que contiene el párametro cuyo modo de paso queremos obtener.
- numeroparam: Orden del parámetro requerido en la lista de parámetros almacenada.

SALIDA: PasoParametro del parámetro requerido.
EXCEPCIONES:
- *EXCEPCION En ObtenerPasoParametro: El atributo nombre no pertenece a esa entrada:* No puede obtener valor alguno de un parámetro almacenado en un atributo que dicha entrada no contiene.
- *EXCEPCION En ObtenerPasoParametro: El parametro numeroparam no existe:* El número de parámetro especificado sobrepasa al número de parámetros almacenados para ese atributo.

EJEMPLO:

```
void main (void)
{
   CArbolSimbolos tabla;
```

```
  CNodoSimbolo *nodo;

 tabla.CrearTabla();
 nodo=tabla.InsertarSimbolo("cuadrado");
 tabla.InsertarTipo(nodo,funcion);
 tabla.ObtenerPasoParametro(nodo,"parametros",1);
}
```

RESULTADO OBTENIDO:
EXCEPCION En ObtenerPasoParam: El parametro 1 no existe

EJEMPLO:

```
void main (void)
{
  CArbolSimbolos tabla;
  CNodoSimbolo *nodo;

 tabla.CrearTabla();
 nodo=tabla.InsertarSimbolo("cuadrado");
 tabla.InsertarTipo(nodo,funcion);
 tabla.InsertarParametro(nodo,"parametros",valor,ent);
 tabla.ObtenerPasoParametro(nodo,"parametros",1);
}
```

EJEMPLO CON LLAMADAS A TODOS LOS MÉTODOS:

```
void main (void)
{
  CArbolSimbolos tabla;
  int resultado;
  void *resultadoptr;
  char *resultadocad;
  void *valor_d;
  CNodoSimbolo *nodo, *nodo1, *nodo2, *nodo3, *nodo4, *nodo5, *nodo6, *nodo7, *nodo8, *nodo9;

 cout<<"Creamos la Tabla de Símbolos"<<endl;
 tabla.CrearTabla();

 cout<<"INSERTAMOS suma, resta, resultado, encontrado, producto, ptr, salto, vector, escribir,palabra"<<endl;

 nodo=tabla.InsertarSimbolo("suma");
 nodo1=tabla.InsertarSimbolo("resta");
 nodo2=tabla.InsertarSimbolo("resultado");
 nodo3=tabla.InsertarSimbolo("encontrado");
 nodo4=tabla.InsertarSimbolo("producto");
 nodo5=tabla.InsertarSimbolo("ptr");
 nodo6=tabla.InsertarSimbolo("salto");
 nodo7=tabla.InsertarSimbolo("vector");
 nodo8=tabla.InsertarSimbolo("escribir");
 nodo9=tabla.InsertarSimbolo("palabra");

 cout<<"Damos formato a las entradas"<<endl;

 tabla.CrearEntrada(identificador,"posicionmemoria",puntero);
 tabla.CrearEntrada(identificador,"inicializada",entero);

 tabla.CrearEntrada(vector,"tipoelemento",cadena);
```

```
tabla.CrearEntrada(vector,"posicion",entero);
tabla.CrearEntrada(vector,"numeroelementos",entero);
tabla.CrearEntrada(vector,"indiceinf",entero);
tabla.CrearEntrada(vector,"indicesup",entero);

tabla.CrearEntrada(funcion,"numparam",entero);
tabla.CrearEntrada(funcion,"tiporetorno",cadena);
tabla.CrearEntrada(funcion,"parametros",listaParametros);
tabla.CrearEntrada(funcion,"direccion",puntero);

tabla.CrearEntrada(procedimiento,"numparam",entero);
tabla.CrearEntrada(procedimiento,"parametros",listaParametros);
tabla.CrearEntrada(procedimiento,"direccion",puntero);

tabla.CrearEntrada(etiqueta,"posicionsalto",puntero);

tabla.CrearEntrada(registro,"posicion",puntero);

cout<<"Insertamos el tipo de cada entrada"<<endl;

tabla.InsertarTipo(nodo,funcion);
tabla.InsertarTipo(nodo1,funcion);
tabla.InsertarTipo(nodo2,identificador);
tabla.InsertarTipo(nodo3,identificador);
tabla.InsertarTipo(nodo4,funcion);
tabla.InsertarTipo(nodo5,identificador);
tabla.InsertarTipo(nodo6,etiqueta);
tabla.InsertarTipo(nodo7,vector);
tabla.InsertarTipo(nodo8,procedimiento);
tabla.InsertarTipo(nodo9,identificador);

cout<<"Inicializamos la Tabla de Símbolos: while, for, if, then, break ,case,"
    << "define,int,char,void,throw,catch"<<endl;

tabla.IntroducirPalabraReservada("while");
tabla.IntroducirPalabraReservada("for");
tabla.IntroducirPalabraReservada("if");
tabla.IntroducirPalabraReservada("then");
tabla.IntroducirPalabraReservada("break");
tabla.IntroducirPalabraReservada("case");
tabla.IntroducirPalabraReservada("define");
tabla.IntroducirPalabraReservada("int");
tabla.IntroducirPalabraReservada("char");
tabla.IntroducirPalabraReservada("void");
tabla.IntroducirPalabraReservada("throw");
tabla.IntroducirPalabraReservada("catch");

cout<<"es palabra reservada if"<<endl;
if (tabla.EsPalabraReservada("if"))
  cout<<"Es palabra reservada"<<endl;
else
  cout<<"No es palabra reservada"<<endl;

cout<<"es palabra reservada void"<<endl;
if (tabla.EsPalabraReservada("void"))
  cout<<"Es palabra reservada"<<endl;
else
  cout<<"No es palabra reservada"<<endl;
```

```
cout<<"es palabra reservada end"<<endl;
if (tabla.EsPalabraReservada("end"))
   cout<<"Es palabra reservada"<<endl;
else
   cout<<"No es palabra reservada"<<endl;

cout<<"es palabra reservada while"<<endl;
if (tabla.EsPalabraReservada("while"))
   cout<<"Es palabra reservada"<<endl;
else
   cout<<"No es palabra reservada"<<endl;

cout<<"es palabra reservada read"<<endl;
if (tabla.EsPalabraReservada("read"))
   cout<<"Es palabra reservada"<<endl;
else
   cout<<"No es palabra reservada"<<endl;

cout<<"damos valor a los atributos de suma(8,9)"<<endl;
tabla.InsertarValor(nodo,"numparam",2);
tabla.InsertarValor(nodo,"tiporetorno","entero");
valor_d=(void *)nodo;
tabla.InsertarValor(nodo,"direccion",valor_d);
tabla.InsertarParametro(nodo,"parametros",valor,ent);
tabla.InsertarParametro(nodo,"parametros",referencia,ent);

cout<<"damos valor a los atributos de resta(15.7,9.6)"<<endl;
tabla.InsertarValor(nodo1,"numparam",2);
tabla.InsertarValor(nodo1,"tiporetorno","real");
valor_d=(void *)nodo1;
tabla.InsertarValor(nodo1,"direccion",valor_d);
tabla.InsertarParametro(nodo1,"parametros",valor,real);
tabla.InsertarParametro(nodo1,"parametros",valor,real);

cout<<"damos valor a los atributos de resultado"<<endl;
valor_d=(void *)nodo2;
tabla.InsertarValor(nodo2,"posicionmemoria",valor_d);
tabla.InsertarValor(nodo2,"inicializada",0);

cout<<"damos valor a los atributos de vector[9]"<<endl;
valor_d=(void *)nodo7;
tabla.InsertarValor(nodo7,"tipoelemento","enteros");
tabla.InsertarValor(nodo7,"posicion",9);
tabla.InsertarValor(nodo7,"numeroelementos",10);
tabla.InsertarValor(nodo7,"indiceinf",0);
tabla.InsertarValor(nodo7,"indicesup",9);

cout<<"damos valor a los atributos salto a encontrado"<<endl;
valor_d=(void *)nodo3;
tabla.InsertarValor(nodo6,"posicionsalto",nodo3);

cout<<"Dame valor de numeroelementos de vector[9]"<<endl;
tabla.DevolverValor(nodo7,"numeroelementos",resultado);
cout<<"OBTENGO:"<<resultado<<endl;

cout<<"Dame valor de inicializada de resultado=0"<<endl;
tabla.DevolverValor(nodo2,"inicializada",resultado);
```

```
  cout<<"OBTENGO:"<<resultado<<endl;

  cout<<"Dame valor de direccion de resta"<<endl;
  tabla.DevolverValor(nodo1,"direccion",resultadoptr);
  cout<<"OBTENGO:"<<resultadoptr<<endl;
  cout<<"HABIA INTRODUCIDO"<<(void *)nodo1<<endl;

  cout<<"Dame valor de tiporetorno de suma"<<endl;
  tabla.DevolverValor(nodo,"tiporetorno",resultadocad);
  cout<<"OBTENGO:"<<resultadocad<<endl;
  cout<<"HABÍA INTRODUCIDO"<<"entero"<<endl;
  cout<<"Dame valores del segundo parámetro de suma (era entero->0,valor->0)"<<endl;
  cout<<tabla.ObtenerTipoParametro(nodo,"parametros",2)<<endl;
  cout<<tabla.ObtenerPasoParametro(nodo,"parametros",2)<<endl;
  cout<<"Dame valores del primer parámetro de suma (era entero->0,referencia->1)"<<endl;
  cout<<tabla.ObtenerTipoParametro(nodo,"parametros",1)<<endl;
  cout<<tabla.ObtenerPasoParametro(nodo,"parametros",1)<<endl;

  cout<<"Dime la posición en el árbol de escribir; era"<<nodo8<<endl;
  cout<<"OBTENGO "<<tabla.BuscarSimbolo("escribir")<<endl;
  cout<<"Dime el tipo de escribir (procemiento->3)"<<endl;
  cout<<tabla.ObtenerTipo(nodo8)<<endl;

  cout<<"Impresión en salida"<<endl;
  tabla.ImprimirTabla("salida.txt");

  tabla.EliminarTabla();
}
```

DIRECTRICES DE COMPILACIÓN

- Microsoft Visual C++ versión 6.0:
  modificar Project | Settings -> pestaña General y donde pone como etiqueta Microsoft Foundation classes elegir "Use MFC in a shared DLL". La aplicación que se ha creado para desarrollar y probar la librería es "Win32ConsoleApplication".

- Borland C++ Builder:
  modificar Tools | Debugger Options y en la pestaña "Language Exceptions" quitar el tick de Stop en C++ Exceptions (al no ser que se quiera que cada vez que se produzca una excepción el programa pare su ejecución y te lo comunique). La aplicación que se ha creado para desarrollar y probar la librería es "Applicarion Wizard"con la opción "VCL".

- g++ versión de GNU:
  se ha compilado siguiendo los siguientes pasos sin tener ningún problema:
  $ g++ -c catributo.cpp
  este comando no genera un ejecutable, genera un .o
  $ g++ -c centrada.cpp
  $ g++ -c cnodosimbolo.cpp
  $ g++ -c carbolsimbolos.cpp
  $ g++ -c pruebav1.cpp
  $ g++ -o p1 carbolsimbolos.cpp centrada.cpp cnodosimbolo.cpp carbolsimbolos.cpp
  este comando ya genera un ejecutable, hay que cambiarle los permisos de ejecución
  $ chmod –x p1
  $./p1

  Puede haber algún problema con los nombres de las librerías (en algunos g++ hay que incluir .h y en otros no) comprobar los nombres en el directorio /include, /include/g++ o el que se corresponda. Otro pequeño problema es el ftp, en algunos casos cambia los nombres de los ficheros y los #include comprobarlo antes de compilar.

De manera general hay que hacer notar que se trata de una librería por lo que lo contiene “main” lo que provoca en todos los casos un “FATAL ERROR” al linkar, no compila por sí solo.