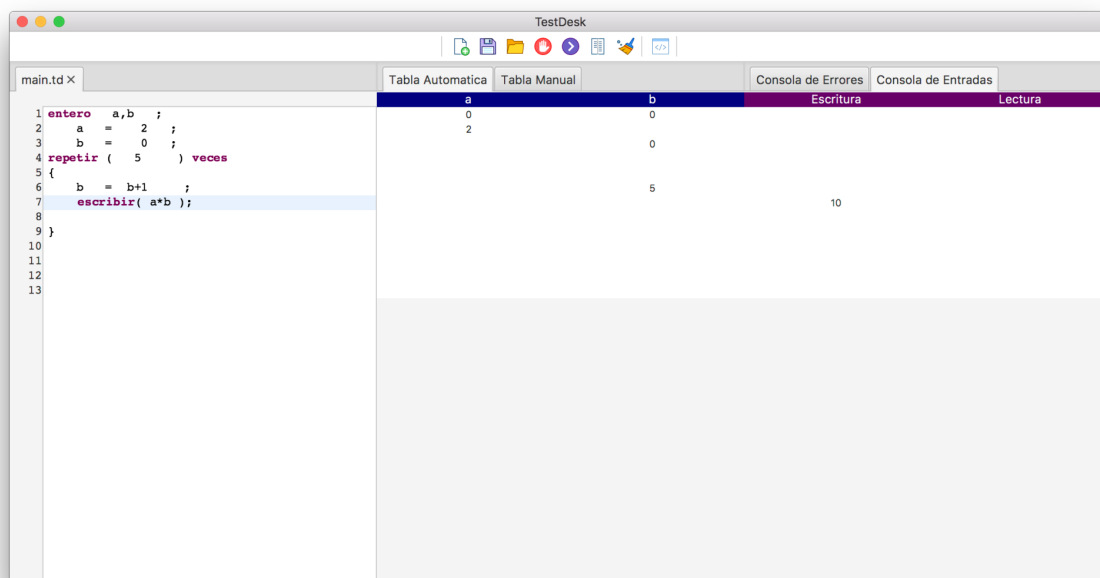


Manual de Usuario - TestDesk

Descripción

TestDesk es un software de aprendizaje de algoritmos estructurados para principiantes o estudiantes que apenas empiezan a programar, este software se comportara como un IDE (Entorno de desarrollo integrado) el cual te ayudará a programar tus primeros algoritmos utilizando un pseudocódigo en español.

Screenshot: Está es una representación de ejemplo del aplicativo; e.j.,



Principales características

- Desarrollo/inserción de estructuras de pseudocódigo mediante un menú de generación de bloques
- Filtro de teclado en el documento para campos no requeridos
- Analizador léxico, sintáctico y semántico de las diferentes expresiones
- Notificación visual de errores en área de codificación
- Notificación detallada de errores en consola
- Ejecución pausada (paso a paso)
- Generación de tablas de prueba de escritorio (manual y automática) perfiladas con las líneas de código
- Registro de cambios en variables en tabla de prueba de escritorio automática (algoritmo en ejecución)
- Almacenamiento de instancias de documentos (pseudocódigo fuente) ~ Guardar código/archivo

- Apertura de instancias de documentos (pseudocódigo fuente) ~ Abrir código/archivo
- Generación de múltiples áreas de código

Pseudocódigo

La herramienta se limita al uso de primitivas de programación:

Asignación

```
■ = ■;
```

Lectura

```
■ = leer( );
```

Escritura

```
escribir( ■ );
```

Estructuras condicionales

```
si ( ■ )  
{  
  
}
```

```
si ( ■ )  
{  
  
}  
sino  
{  
  
}
```

Estructuras cíclicas

```
repetir ( ■ ) veces  
{  
  
}
```

```
mientras que ( ■ )  
{
```

}

Por lo tanto, la complejidad de los algoritmos por los cuales se desarrolló esta propuesta son de carácter simple, sin ser extensos mayormente haciendo uso únicamente de un método principal sin invocar a secundarios (archivo principal de trabajo).

El pseudocódigo al ser independiente es posible delimitarlo a 4 tipos de datos (**entero**, **decimal**, **texto**, **lógico**) con el propósito de acercar al estudiante al contexto de tipación en los lenguajes. Con base a estos tipos de datos se excluyeron vectores o matrices de los mismos.

Los operadores básicos involucrados en esta herramienta son los siguientes símbolos:

Operador	Operación
	ó
&&	y
==	igual que
!=	diferente que
>	mayor que
<	menos que
<=	menor o igual que
>=	mayor o igual que
!	negación
+	suma
-	resta
*	multiplicación
/	división
%	módulo
^	potencia

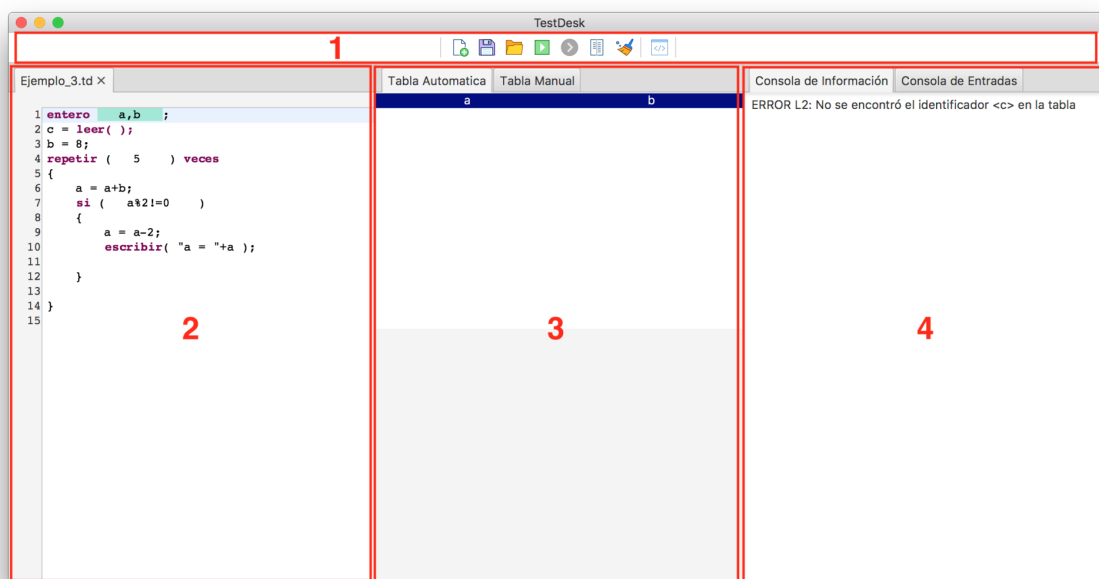
Áreas

El aplicativo se encuentra distribuido en cuatro áreas de trabajo, las cuales se denominan:

- Barra superior
- Área de codificación
- Área de pruebas de escritorio (registro de lo que ocurre en memoria)
- Área de consolas

En la imagen se puede apreciar el la ubicación de estas respecto al índice listado.

Screenshot: Ubicación de las áreas de trabajo; e.j.,



Barra superior

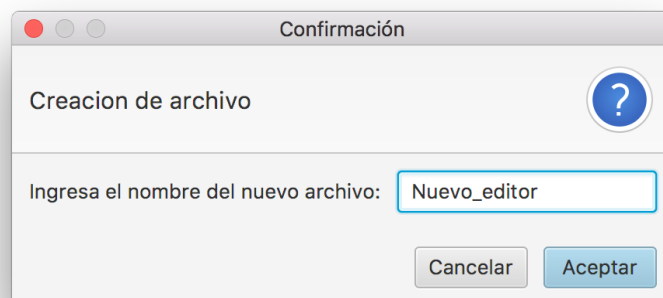
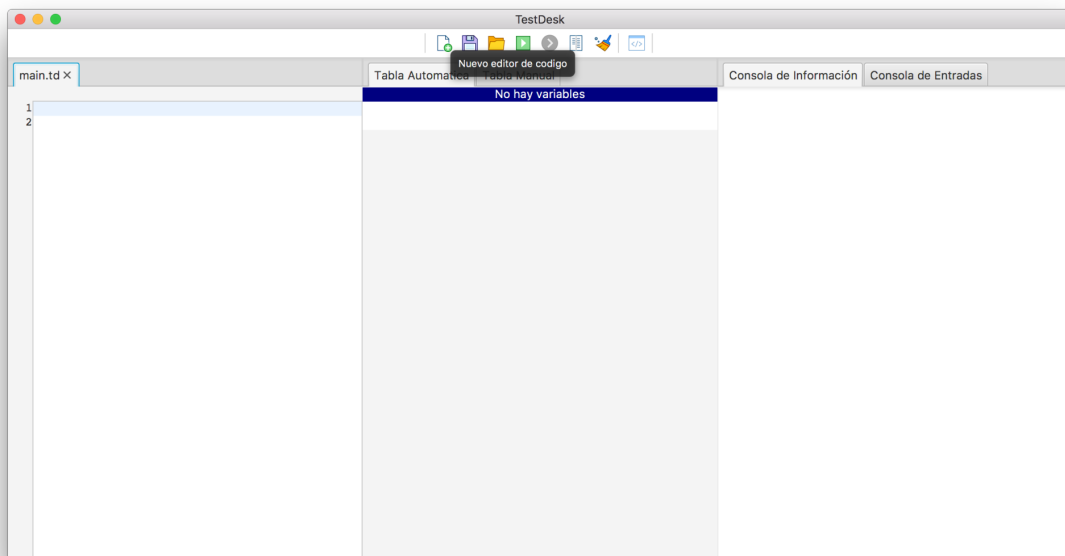
La barra superior constituye una serie de botones con iconos los cuales te permitirán realizar diversas acciones de manera general con tu trabajo.

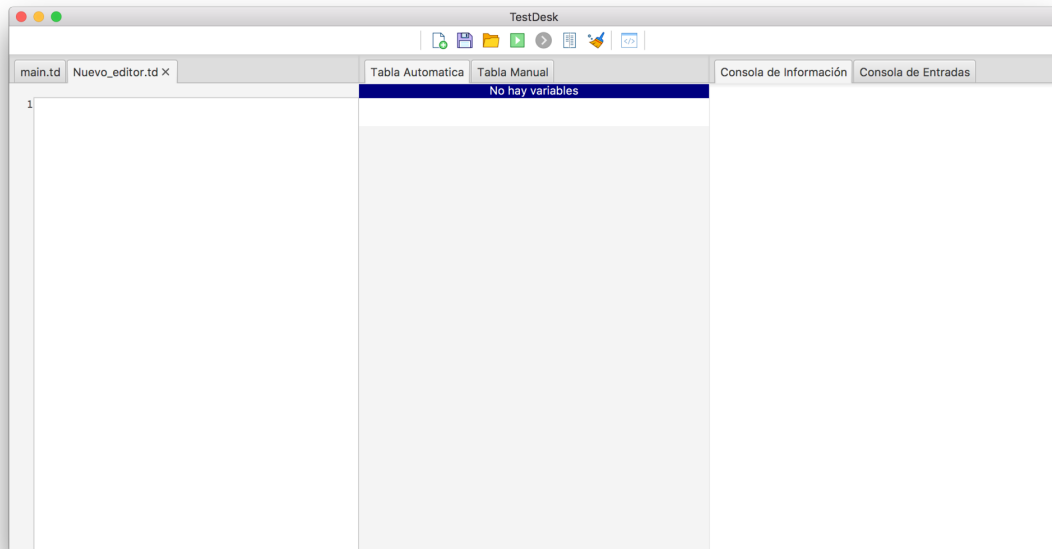
Nuevo editor de código



Al pulsar este icono se creará un nuevo editor de código con el que podrás trabajar. Puedes tener diferentes editores de código pero solo podrás ejecutar uno al mismo tiempo, y por ende solo podrás interactuar con la información (errores, entradas o salidas) del mismo.

A continuación verás una secuencia de imágenes las cuales muestran el proceso para la creación de un nuevo editor de código.

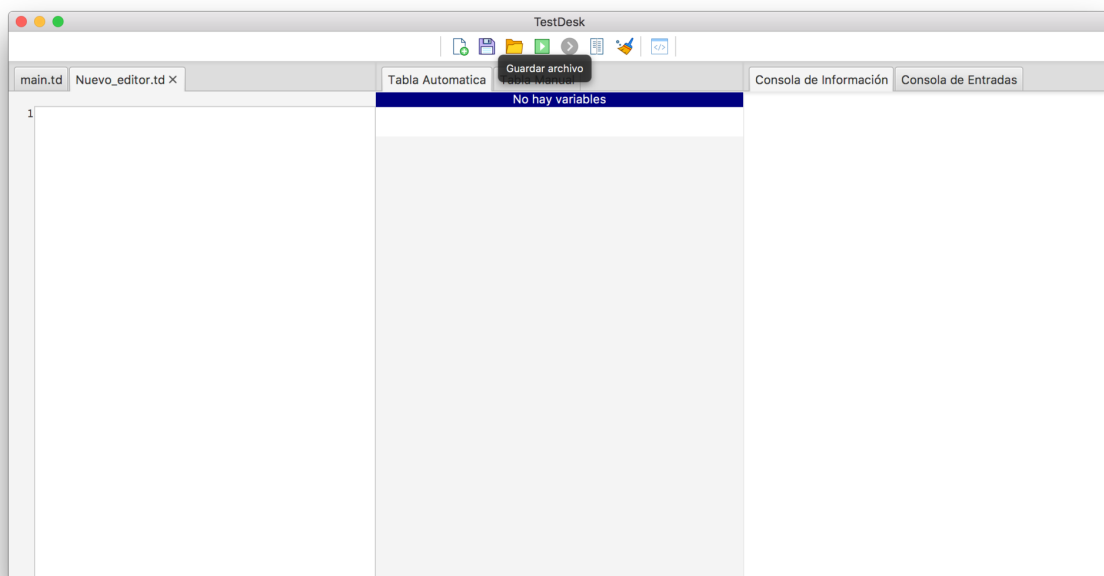


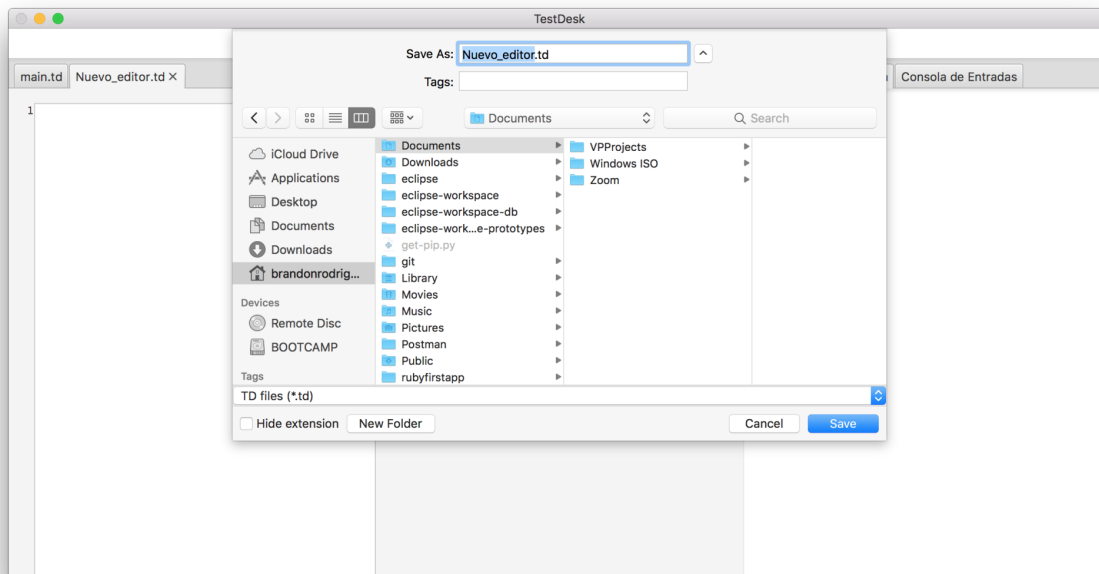


Guardar



Al pulsar este icono se guardará tu algoritmo en un archivo **.td**, el nombre lo podrás cambiar a tu gusto aunque la herramienta siempre te recomendará el nombre con el que creaste el editor. A continuación verás una secuencia de imágenes las cuales muestran el proceso para guardar tu archivo.

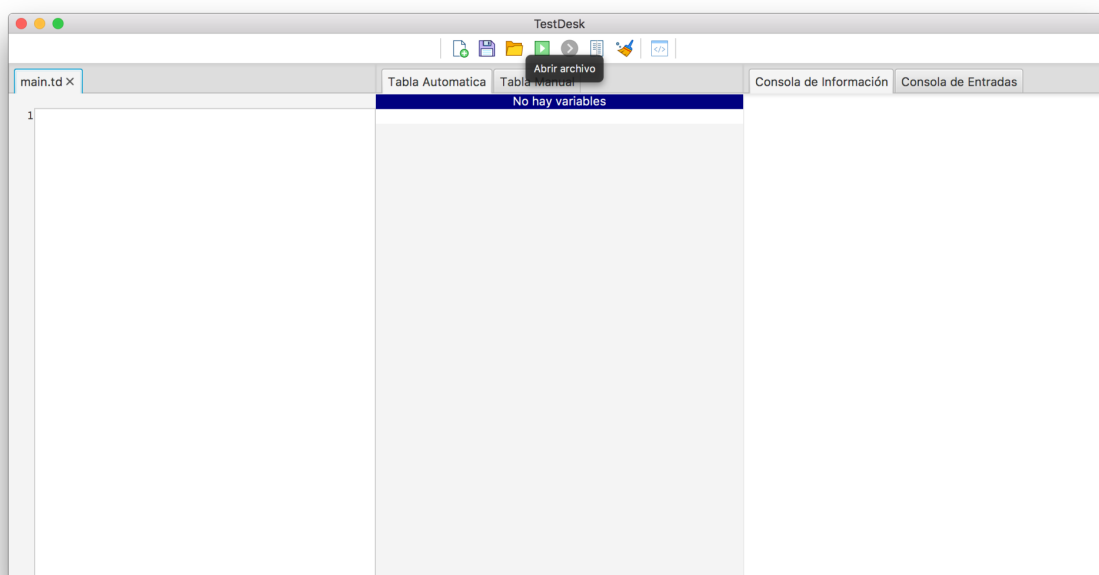


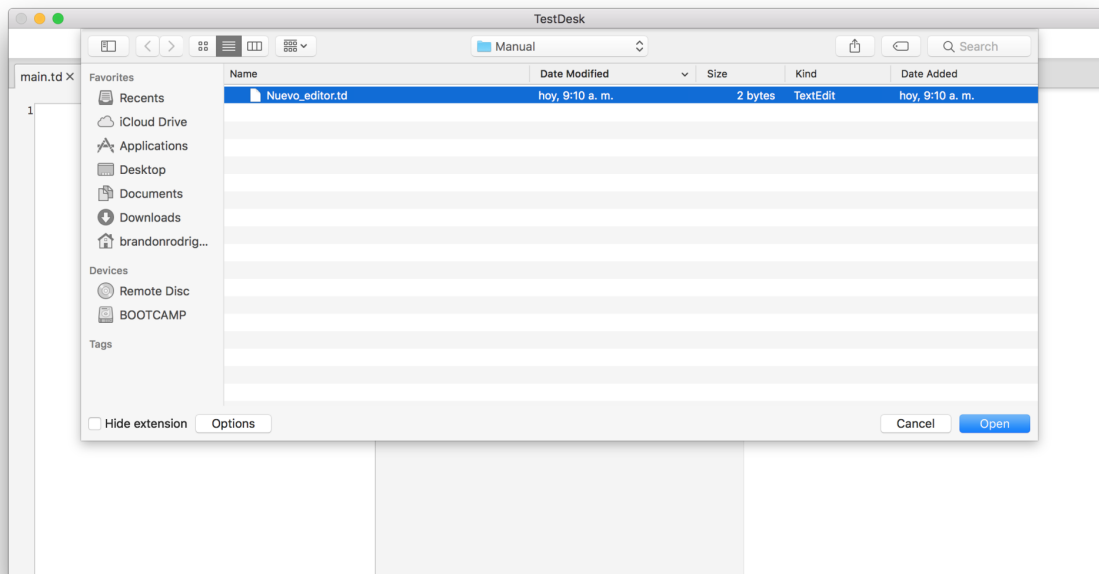


Abrir



Al pulsar este icono se abrirá tu archivo **.td**, solo deberás buscarlo entre tus documentos. A continuación verás una secuencia de imágenes las cuales muestran el proceso para abrir tu archivo.





Ejecutar

Este icono es cambiante dependiendo del modo en que se encuentre la herramienta, por ahora solo veremos qué significa cada uno de ellos. En otro capítulo más adelante se detalla la función.



Cuando este icono está presente, significa que estás en modo edición y podrás editar tu código en el área de codificación de lo contrario no podrás realizar. Si lo pulsas pasarás al modo de ejecución y el icono cambiará al que se muestra a continuación.



Este icono significa que estás en el modo de ejecución el cual te permitirá ejecutar cada instrucción. Si lo pulsas volverás al modo edición y el icono cambiará al que se mostró previamente.

Ejecutar siguiente instrucción

Este icono es cambiante dependiendo del modo (edición o ejecución) y el estado en que se encuentre si está en el modo ejecución.



Cuando este icono está presente, significa que estás en modo ejecución y que podrás ejecutar la siguiente instrucción.



Si este icono está presente y estas en modo edición significa que no podrás ejecutar la siguiente instrucción porque no has entrado en modo ejecución. Por otro lado, si estás en modo ejecución y este icono está presente... significa que la herramienta ha terminado de ejecutar todas las instrucciones en tu código y no puede ejecutar la siguiente ya que no hay más. En este punto solo podrás volver al modo edición pulsando nuevamente el botón de ejecutar.

Comparar



Al pulsar este icono podrás notar los errores al comparar la tabla automática y manual. Más adelante se mostrará un ejemplo.

Limpiar



Al pulsar este icono podrás limpiar la tabla automática y manual.

Área de codificación

Esta es la principal área de trabajo, acá podrás crear algoritmos a partir de las diferentes primitivas. Esta área es un espacio de trabajo filtrado y asistido, por lo cual no tienes que preocuparte de la estructura de las primitivas.

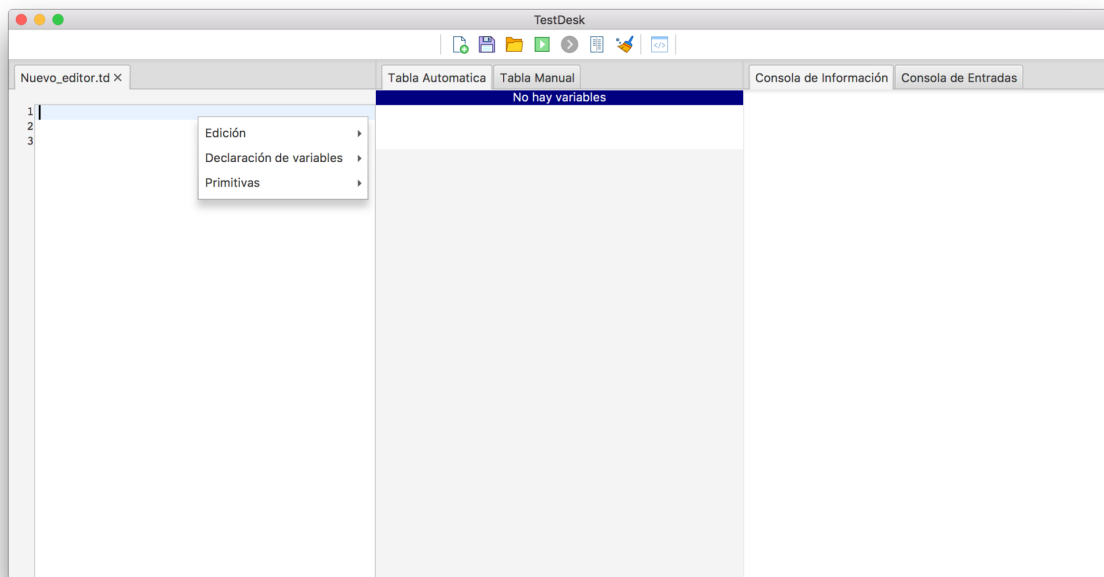
Inserción

Para insertar algo en tu espacio de trabajo solo debes ubicar tu cursor | en la línea en la que deseas insertar la primitiva. Posteriormente haces *click derecho* y aparecerá un menú desglosando las diferentes opciones de inserción. A continuación verás la estructura del menú:

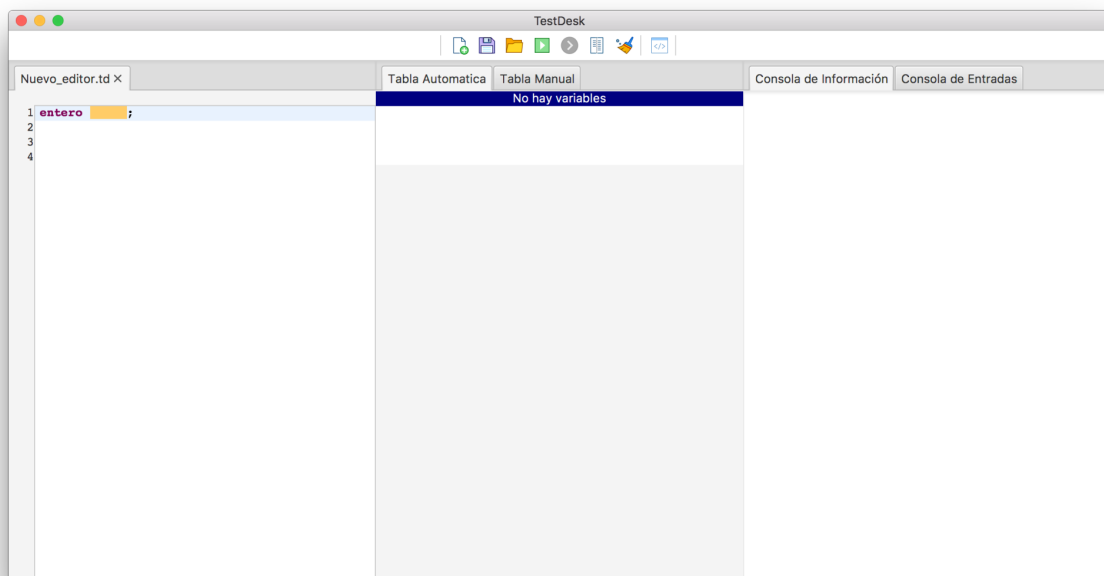
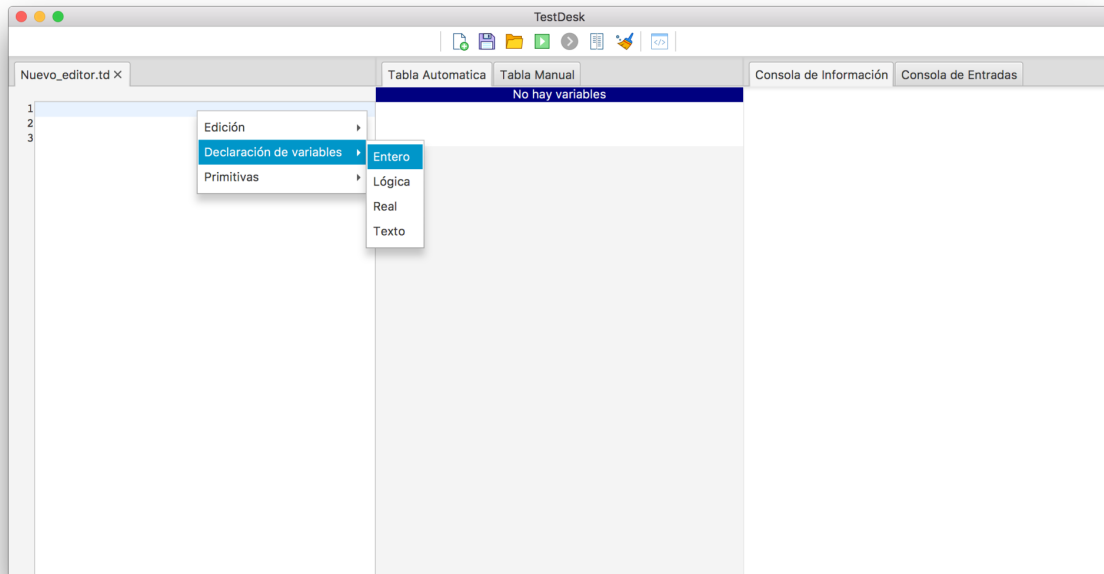
- Edición
 - Nueva línea
 - Eliminar línea o bloque
- Declaración de variables
 - Entero
 - Real
 - Lógica
 - Texto
- Primitivas
 - Simples



- Lectura
- Escritura
- Asignación
- Bloque
 - Condicionales
 - Si simple
 - Si sino
 - Ciclos
 - Repetir
 - Mientras que

En la siguiente imagen verás como se muestra este menú en la herramienta.



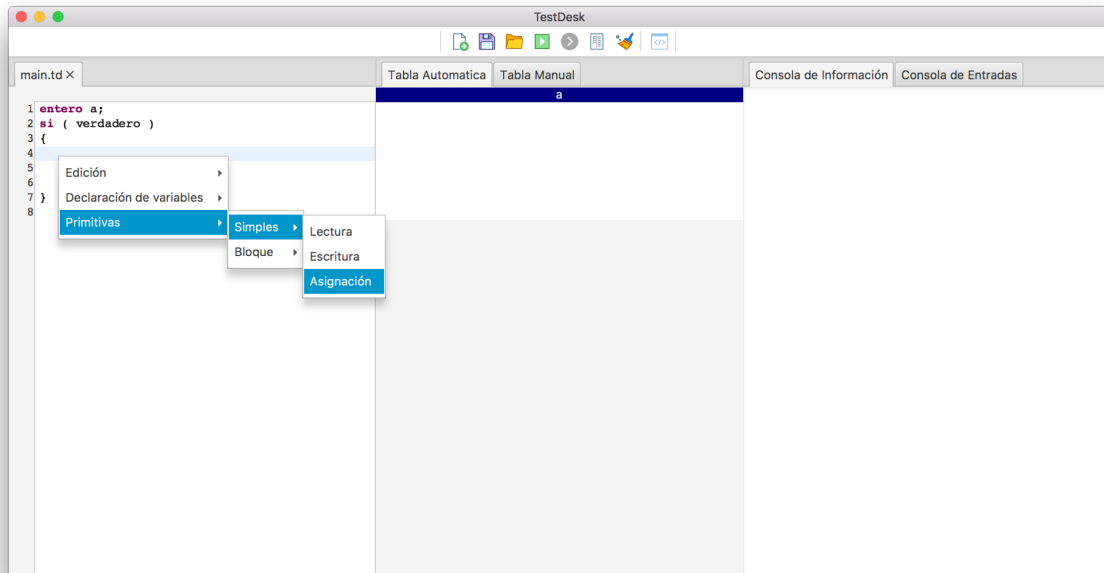
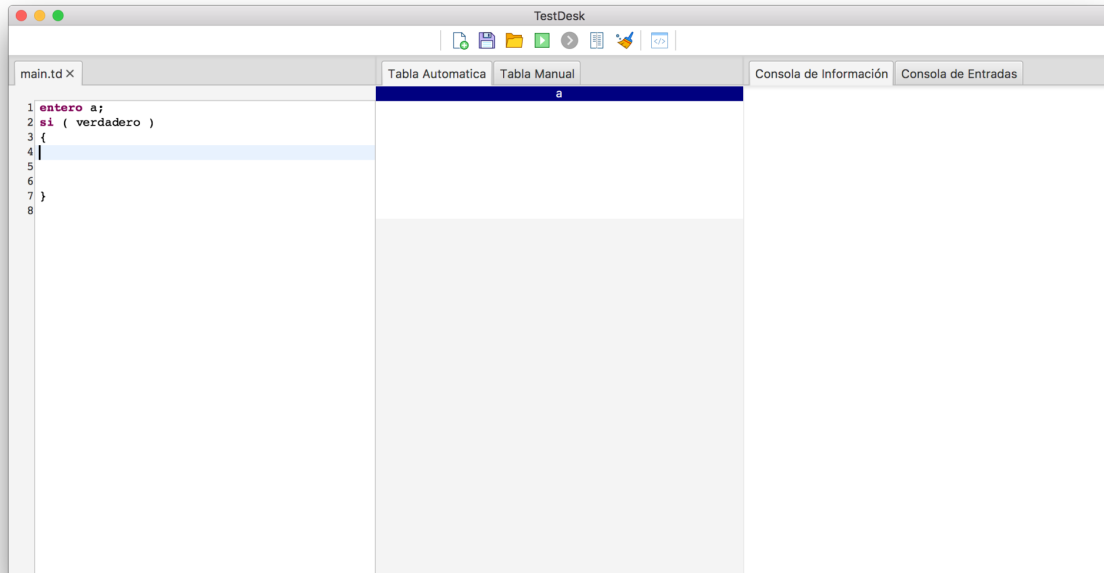
Después solo tienes que desglosar los sub-menús colocando el mouse sobre ellos, allí encontrarás la primitiva según la estructura ya definida y dar click izquierdo sobre el ítem.

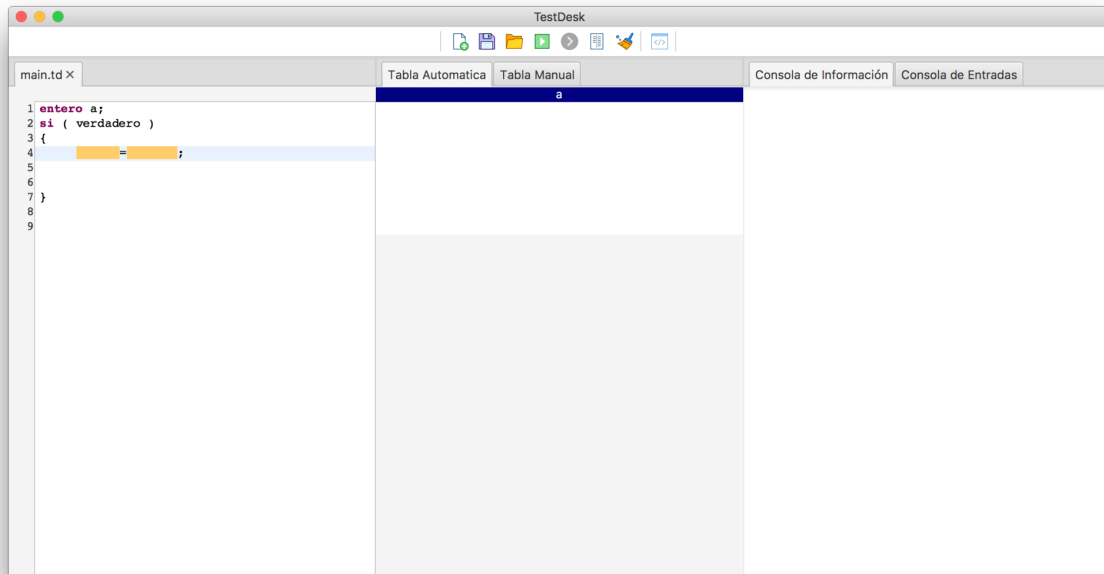


Cada primitiva tiene un espacio donde podrás ingresar tus expresiones libremente, es decir, podrás ingresar el contenido desde tu teclado como es habitual. Este espacio está definido por un resaltado o sombreado el cual aparecerá únicamente cuando el cursor se ubique en la línea. Este sombreado te notificará por medio de colores si la expresión es correcta o incorrecta según la primitiva. La herramienta te notificará los errores en este espacio por medio de color naranja , pero si tu expresión es correcta para la primitiva su color será verde .

Indentación

No te preocupes por la indentación (sangría), este recurso lingüístico es muy importante en programación sobre todo en lenguajes como Python. TestDesk es capaz de calcular la indentación necesaria para la inserción de alguna primitiva dentro de un bloque, a continuación verás un ejemplo de ello.





Área de pruebas de escritorio

En este área podrás ver lo que ocurre con tu programa en memoria, existen 2 pestañas *Tabla automática* y *Tabla manual*.

Tabla automática

En la tabla automática podrás ver lo que ocurre con tu programa en memoria, este es un recurso que suele hacerse de manera manual por los programadores, más sin embargo esta herramienta lo hace automáticamente por ti mientras creas un modelo mental de cómo hacerlo y así no te confundirás en el futuro. Esta tabla se va llenando cuando estes en modo ejecución y recorriendo (ejecutando) cada instrucción.

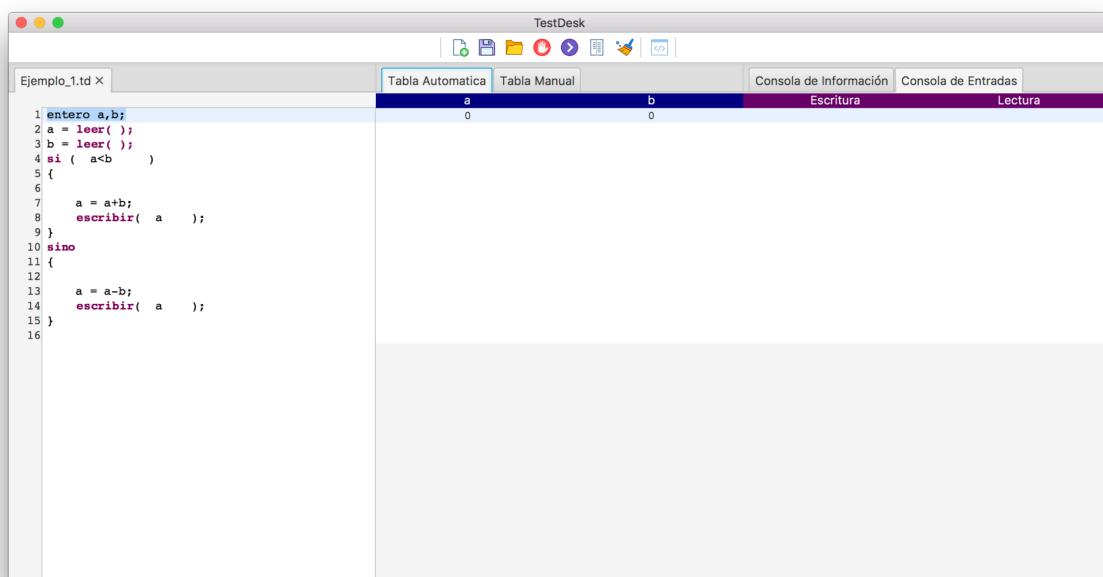
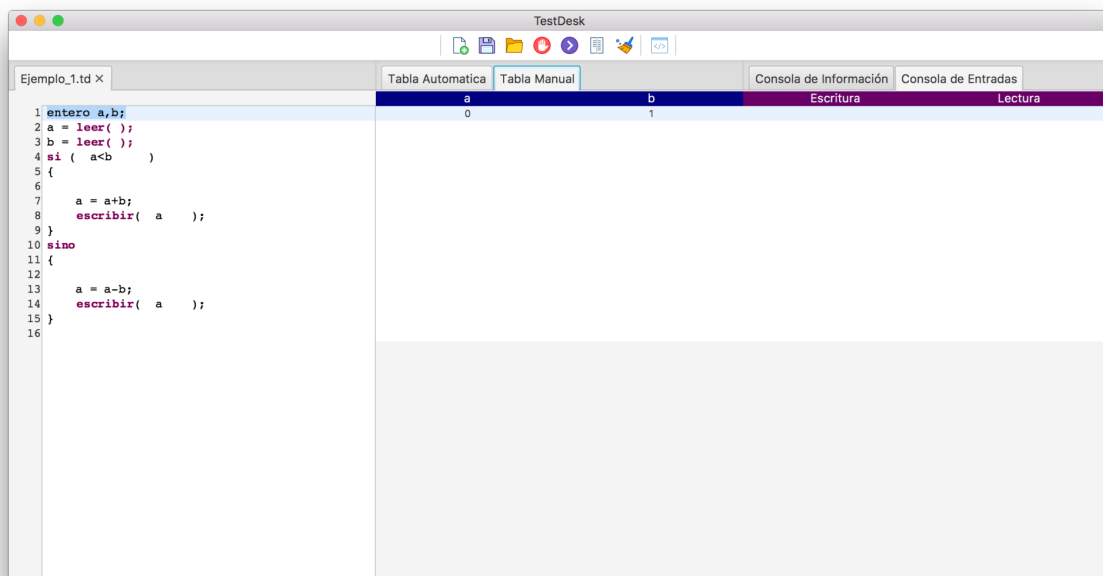
En la barra superior color azul con letras blancas encontrarás las variables declaradas. Cada una de sus filas está alineada con las líneas de código presentes en el área de codificación, así que para encontrar el cambio ocurrido en memoria en la ejecución solo debes ubicar la fila correspondiente a la línea y también la columna a la que corresponda la variable asignada con el cambio. En esta tabla no podrás insertar valores. En el capítulo de ejecución se hará un ejemplo de ello.

Tabla manual

En la tabla manual es posible insertar valores correspondientes a los cambios que tu identifiques, para ello debes entrar en modo ejecución. De esta manera podrás llenarla y a medida de que se ejecuten las líneas podrás identificar las diferencias (errores) al comparar la tabla automática con la tabla manual.

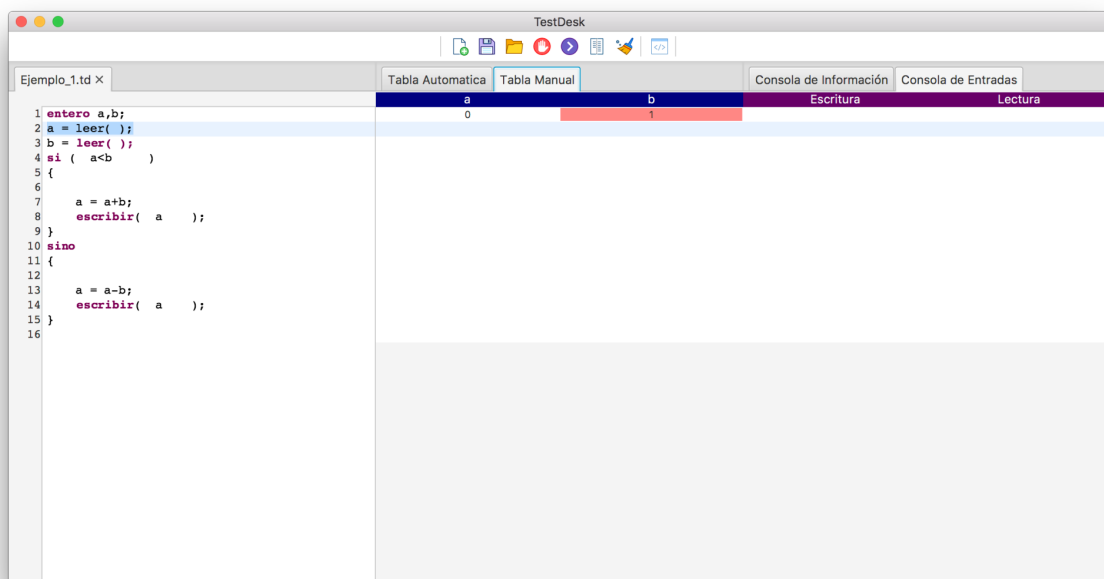
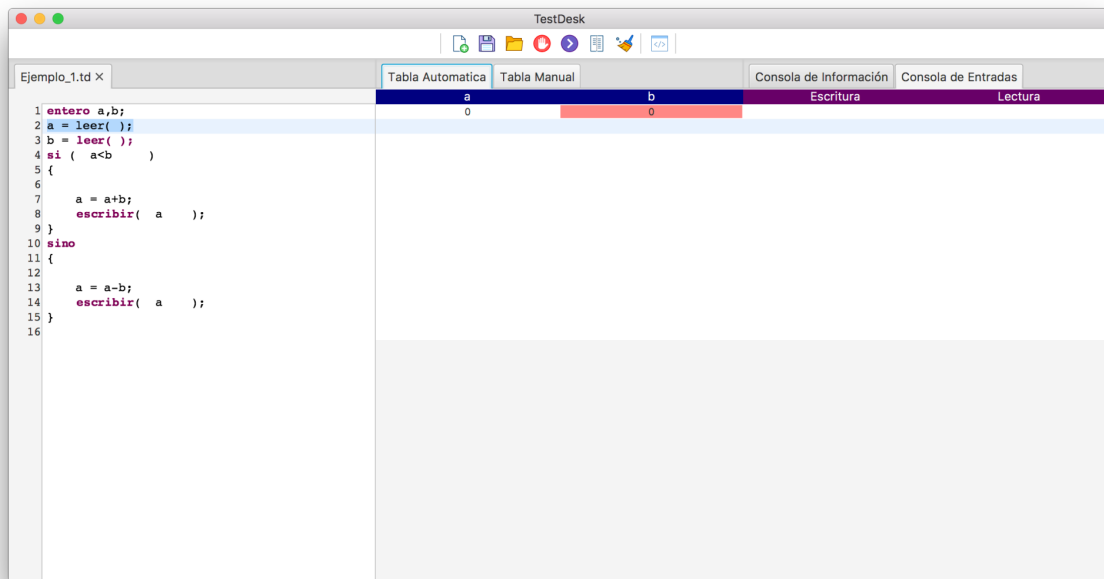
Comparar Tablas

En la siguiente imagen podrás un ejemplo de esta función explicada en el capítulo *BarraSuperior.Comparar*. En este caso la línea 1 para las variables *a* y *b* se denota una declaración, por ende se inicializa la variable con valores predefinidos (en este caso 0). Como se puede apreciar en la *Tabla automática* los valores declarados son 0 y 0 mientras que los valores denotados en la *Tabla manual* son 0 y 1. Al pulsar el botón comparar se cambian los estilos de las celdas, para poder verlos adecuadamente es necesario que *la línea de ejecución* no se encuentre en la misma fila.



Cambio de estilos al comparar las tablas:

<div style="display:flex">



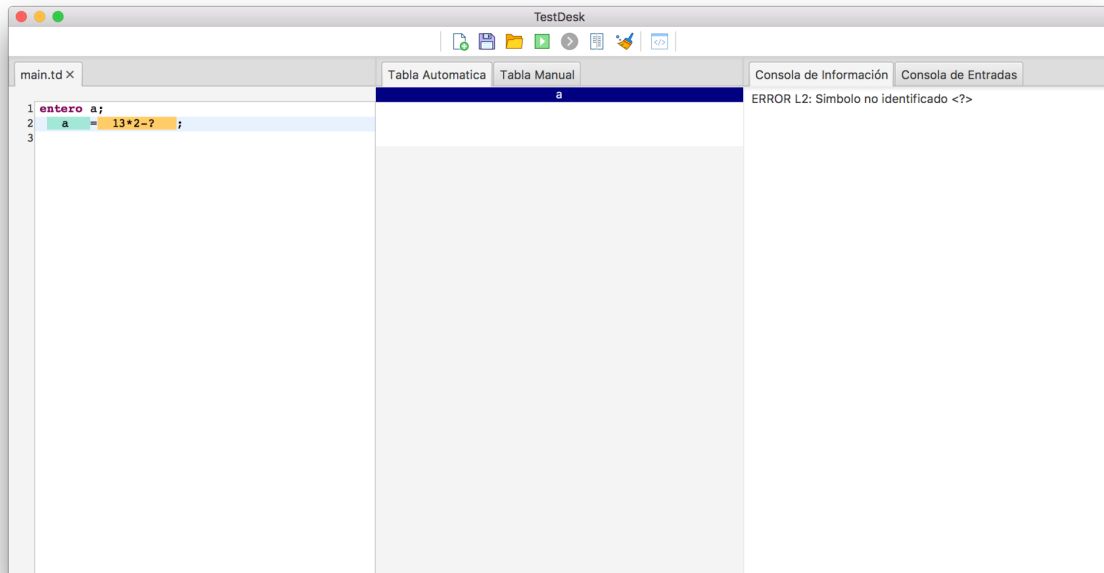
Si se vuelve a comparar las tablas una vez corregido el error se cambiará el color de las celdas a blanco nuevamente.

Área de consolas

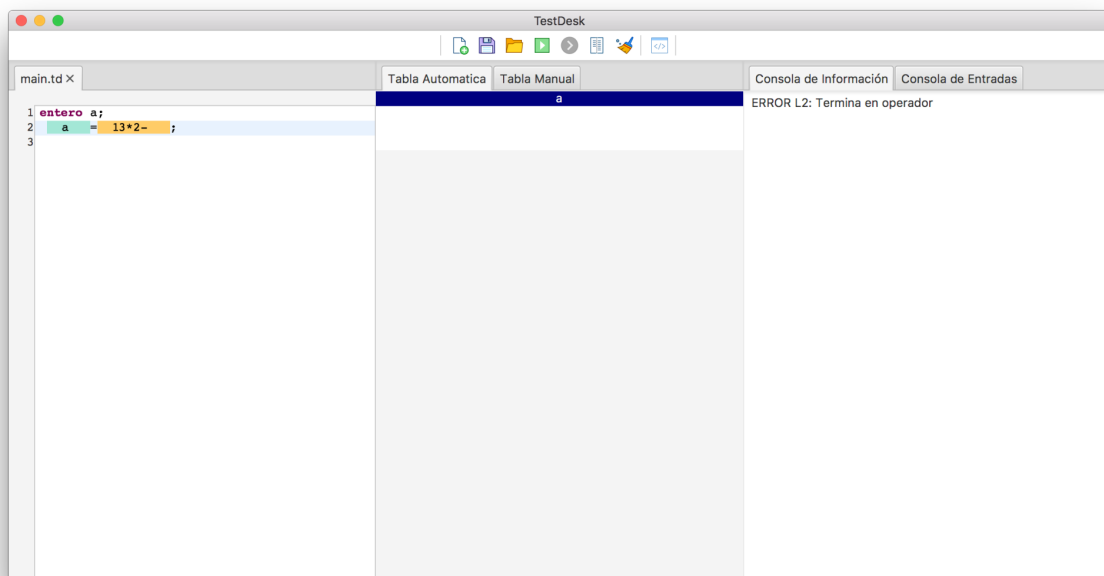
La consola de información es la encargada de detallar los diferentes errores en las expresiones, TestDesk comprobará la expresión de manera léxica, sintáctica y semántica. En caso de encontrar un error lo notificará en la consola de información y aplicará un resaltado color naranja sobre ella. La estructura del error será **ERROR L#:***Detalle del error*. Donde el # indica el numero de linea y el *Detalle del error* mostrará el error con mayor

información. A continuación se muestran unos ejemplos de errores léxicos, sintácticos y semánticos.

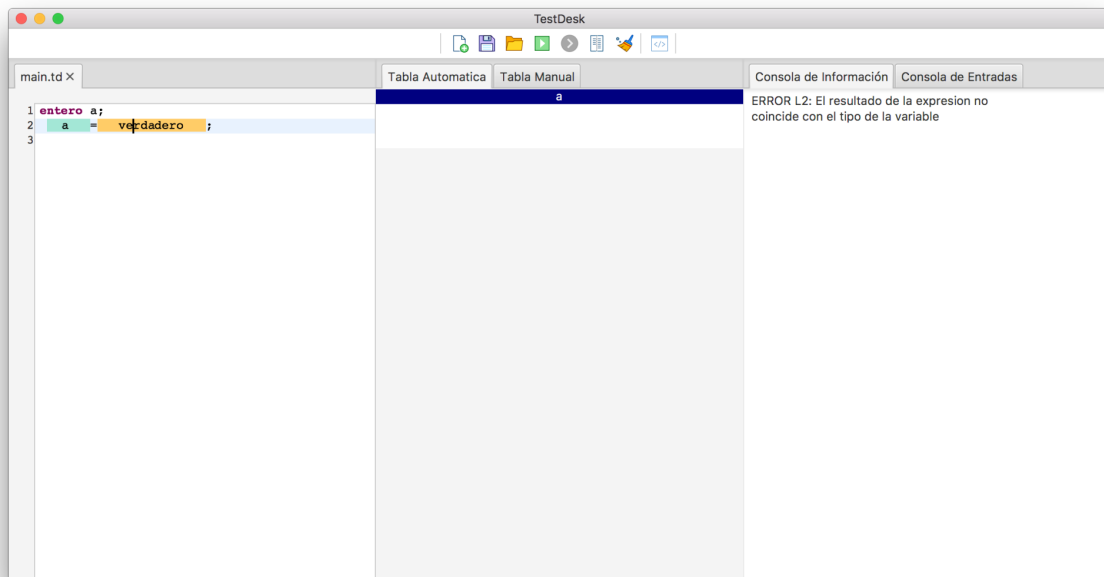
Error léxico



Error sintáctico



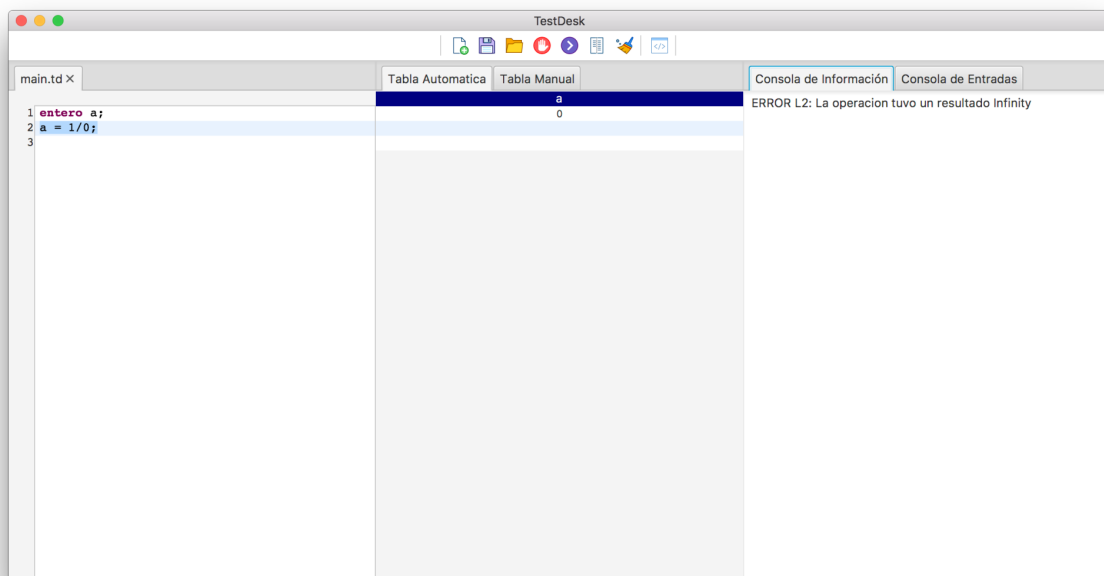
Error semántico



La consola de información también es capaz de mostrar información en ejecución, como errores en la operación de expresiones o salidas. A continuación se muestra un ejemplo de cada una:

Errores en la operación de expresiones

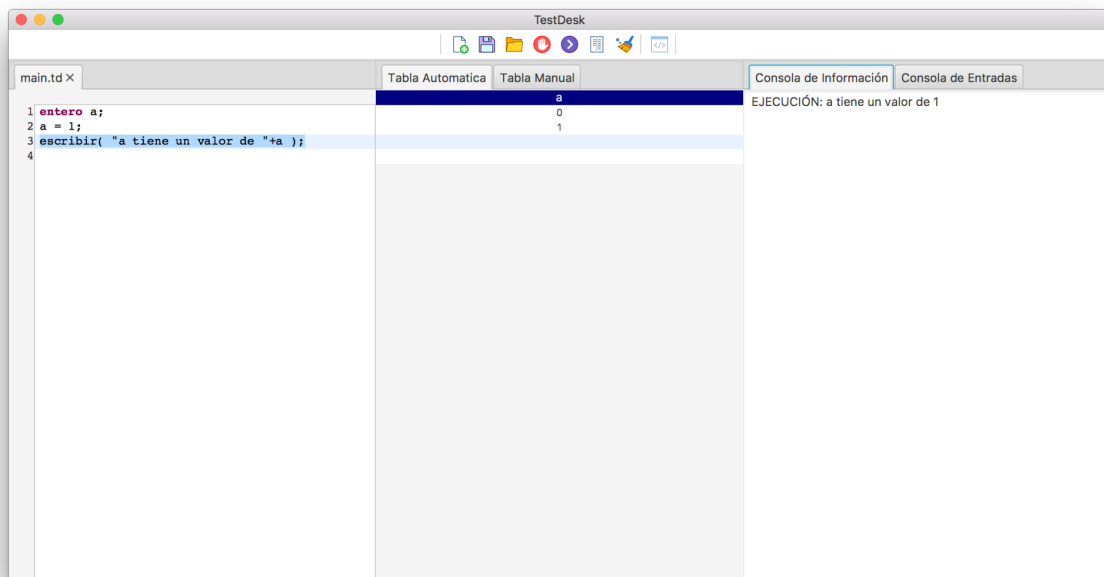
Los errores en ejecución se verán con la misma estructura explicada anteriormente. A continuación un ejemplo:



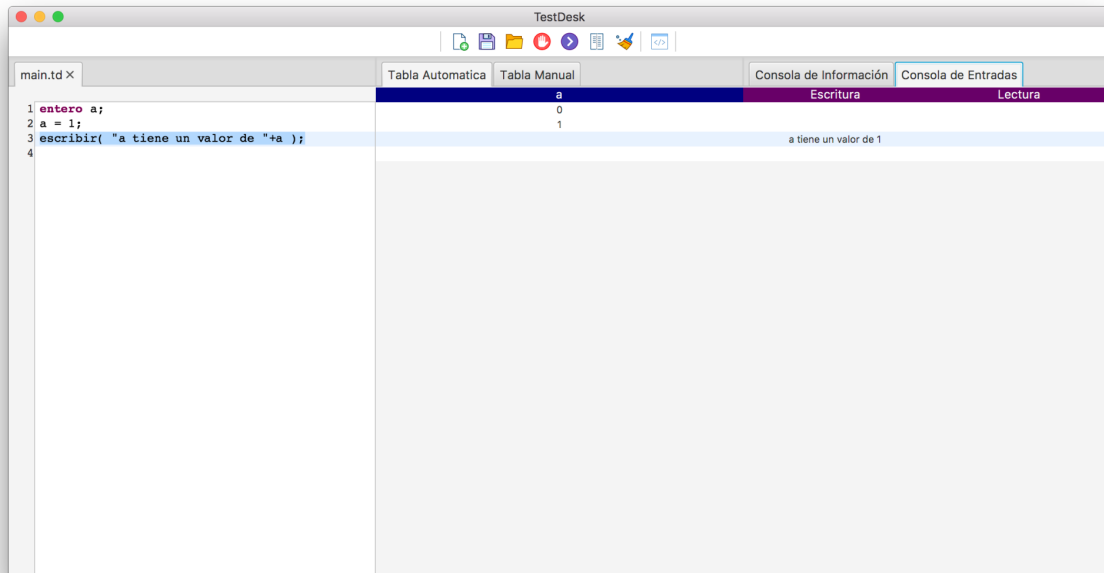
Salidas

Las salidas (ejecutadas por la primitiva simple *escritura*) tendrán la estructura



EJECUCIÓN: *Impresión*





Por otro lado en la *Consola de entradas* se expresa de la siguiente manera en la columna de escritura:



Ejecución

Cuando el programa entre en modo ejecución después de pulsar el botón  se cambiará al icono . De este modo no podrás alterar tu código de ninguna manera, se limpiará la

información en las tablas y consolas, y aparecerá una línea sombreada azul atravesando los paneles de codificación, tablas y consola de entradas y salidas indicando que dicha línea es la actual en ejecución. La línea recorrerá el código en función al comportamiento teórico de las primitivas cada que pulses el botón *ejecutar siguiente instrucción* . Cada que se denote un cambio en memoria (al asignar un nuevo valor en alguna variable) se mostrará en la *Tabla automática* en la línea antes mencionada. En caso de efectuarse un error se presentará en el área de *Consola de información*. Si se expresa una escritura, entonces se denotará está en la *Consola de información* y *Consola de entradas y salidas*. Si se expresa una lectura, el programa ajustará el cursor en la columna de *Lectura* para la *Consola de entradas y salidas*, allí deberas ingresar el valor y continuar ejecutando la siguiente línea hasta llegar a la última de la de la ejecución (donde el icono se encuentra deshabilitado ).

A continuación una serie de videos que muestran su función:

- [Ejemplo 1](#)
- [Ejemplo 2](#)
- [Ejemplo 3](#)