



TECNOLÓGICO NACIONAL DE MÉXICO  
**INSTITUTO TECNOLÓGICO DE VERACRUZ**  
VERACRUZ, VERACRUZ



CARRERA  
**INGENIERÍA EN SISTEMAS COMPUTACIONALES**

MATERIA  
**LENGUAJES Y AUTOMATAS I**

DOCENTE  
**ING. OFELIA GUTIERREZ GIRALDI**

TITULO  
**Manual de Usuario**

ESTUDIANTE  
**JOSÉ MANUEL MIRANDA VILLAGRÁN**  
**LUIS ENRIQUE ESCAMILLA ÁLVAREZ**  
**BRYAN CASTILLO MARÍN**

**24 de agosto de 2019**

# Indice

|   |           |
|---|-----------|
| <b>Introducción.....</b>                  | <b>3</b>  |
| <b>Objetivo.....</b>                      | <b>3</b>  |
| <b>Dirigido a.....</b>                    | <b>3</b>  |
| <b>Manejo e interfaz de usuario.....</b>  | <b>4</b>  |
| <b>1.-Acceder a la aplicación.....</b>    | <b>4</b>  |
| <b>2.-Estructura de la interfaz.....</b>  | <b>4</b>  |
| 2.1.-Barra de menús.....                  | 5         |
| 2.2.-Barra de herramientas.....           | 7         |
| 2.3.-Editor de código.....                | 8         |
| 2.4.-Despliegue de resultados.....        | 9         |
| <b>3.-Uso del compilador.....</b>         | <b>12</b> |
| 3.1.-Escribir y analizar un programa..... | 12        |
| 3.2.-Guardar un archivo.....              | 13        |
| 3.3.-Abrir un archivo.....                | 14        |
| 3.4.-Atajos del teclado.....              | 15        |
| <b>Conclusiones.....</b>                  | <b>16</b> |
| <b>Bibliografía.....</b>                  | <b>17</b> |

## **Introducción**

En este manual se plasma de la manera más sencilla y práctica el uso del compilador 'MJU'. Dicho compilador es un traductor de texto que ha sido desarrollado durante el curso de 'Lenguajes y Autómatas I' perteneciente al mapa curricular de la carrera Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Veracruz.

Este compilador es una solución para el desarrollo de programas basados en el lenguaje con el que comparte el mismo nombre 'MJU', el cual es un lenguaje de alto nivel y desarrollado exclusivamente para su uso en este compilador.

## **Objetivo**

La finalidad con la que se realiza este manual es llevar de la mano al usuario para el manejo del compilador 'MJU' ofreciendo no solo ayuda textual sino también ayuda gráfica.

## **Dirigido a**

Este manual está orientado tanto a usuarios sin experiencia en el manejo de compiladores como a usuarios con conocimientos más amplios de este campo.

# Manejo e Interfaz de usuario

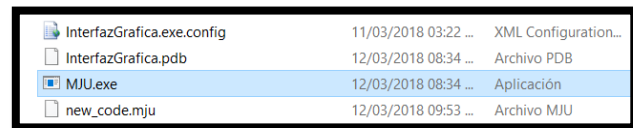
## 1.-Acceder a la aplicación



**Figura 1**

Se puede acceder al compilador desde un acceso directo como se muestra en la figura 1.

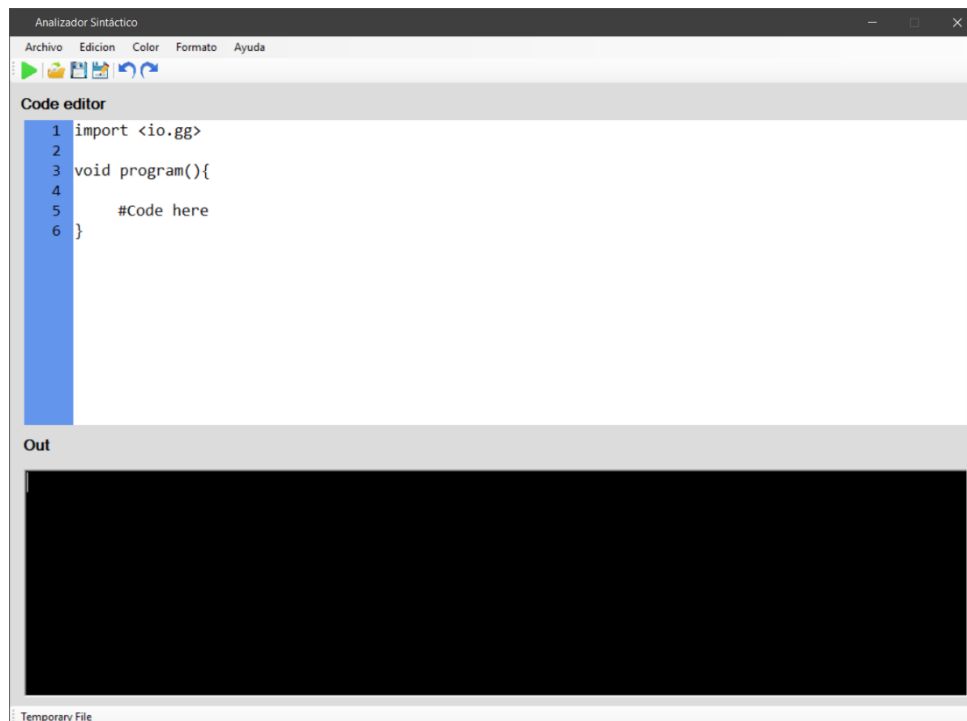
También desde la aplicación principal que se encuentra ubicada dentro de la carpeta principal del compilador en las siguiente dirección: *bin\Debug* tal como se muestra en la figura 1.1



**Figura 1.1**

## 2.-Estructura de la interfaz

La figura 2 muestra la interfaz de 'MJU', dicha interfaz se divide en 4 partes principales las cuales se irán detallando en los próximos puntos.

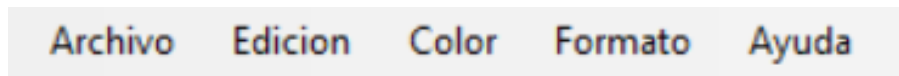


**Figura 2**

La interfaz en general tiene un diseño ligeramente transparente, pero para ilustrar mejor todos los elementos que la componen la veremos sin dicho efecto.

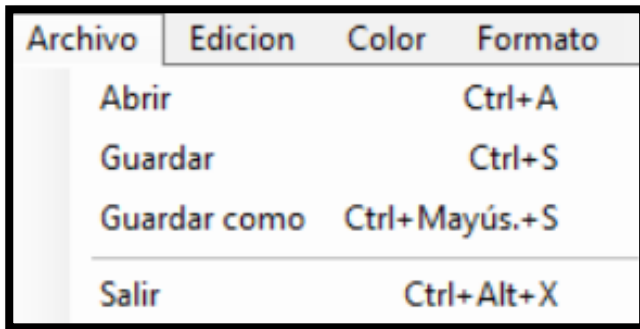
## 2.1.-Barra de menús

La parte superior lleva como nombre “Barra de menús” la cual destaca en la figura 2.1



**Figura 2.1**

La barra provee 5 diferentes opciones las cuales el usuario puede elegir libremente.



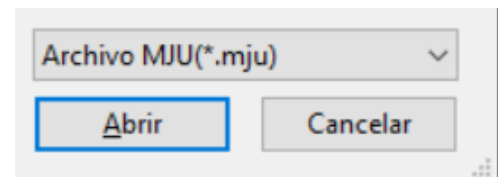
**Figura 2.1.1 Menú ‘Archivo’**

### Archivo:

Este es el primer menú desplegable, contiene diferentes opciones las cuales están enfocadas a la manipulación de archivos empezando con “Abrir” el cuál da a entender su función por su mismo nombre, permite abrir un archivo con una extensión “.mju” es importante tener claro esto porque es el

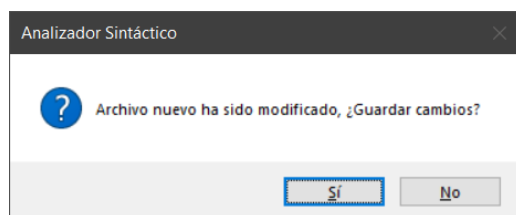
formato de ficheros que maneja MJU(Figura 2.1.2).

Los proximos 2 botones llamados “Guardar” y “Guardar Como” comparten la característica que permiten conservar los cambios que se van realizando en un archivo mientras se está programando, con la diferencia que cada uno está hecho con un propósito que los difiere, “Guardar como” permite crear un nuevo archivo o sustituirlo mientras que “Guardar” simplemente sobrescribe los cambios hechos en un archivo que se esté trabajando.



**Figura 2.1.2**

Cuando se inicia por primera vez el programa y el usuario crea algún código, si trata de cerrar el compilador, MJU desplegará un mensaje diciendo lo siguiente (Figura 2.1.3).



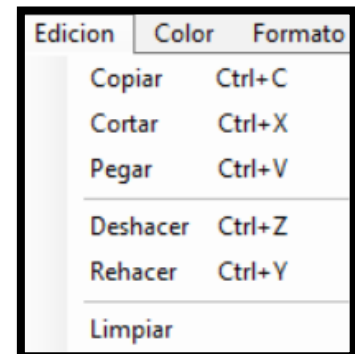
**Figura 2.1.3**

Esto permitirá al usuario guardar su código en un nuevo archivo “.mju”, en dado caso que el elija la opción de “no” simplemente saldrá del compilador sin guardar sus cambios, esta utilidad estará siempre que el usuario cierre el compilador sin guardar sus cambios.

Cada una de estas opciones cuentan con un método abreviado de teclas con las cuales el usuario puede acceder a ellas sin necesidad de seleccionarlal con el cursor, dichas combinaciones se mostrarán en una sección más adelante.

### **Edición:**

En este menú encontramos una variedad de opciones relacionadas a la manipulación de texto, desde lo más básico como (Copiar, Cortar y pegar), así como también herramientas para deshacer o rehacer algún cambio en la escritura, por último tenemos “Limpiar” el cual su función es eliminar todo el texto del archivo.



**Figura 2.1.4 Menú ‘Edición’**

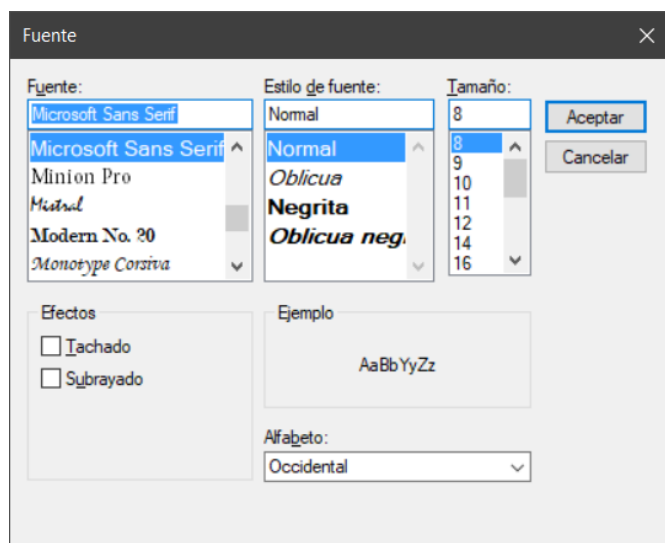
### **Color y Formato:**

En el siguiente menú titulado como ‘Color’ (Figura 2.1.5) desplegará una ventana emergente donde podemos escoger cual es el color que le queremos asignar a nuestro texto dentro del *editor de código*.

Al seleccionar el menú ‘formato’ (Figura 2.1.6) también nos desplegará una ventana emergente que sirve para seleccionar que tipografía queremos para el *editor de código* y el *contador de líneas*.



**Figura 2.1.5 Menú ‘Color’**



**Figura 2.1.6 Menú ‘Formato’**

### **Ayuda:**

Por último, pero no menos importante se encuentra este menú, donde hallará la información de los desarrolladores y diseñadores de MJU.



**Figura 2.1.7**  
**Acerca de.**

## **2.2.-Barra de herramientas**

La barra de herramientas (Figura 2.2) se encuentra ubicada debajo de la barra de menús y por encima del editor de código, esta provee un acceso más rápido e intuitivo a diferentes herramientas que se encuentran distribuidas entre los menús de la barra de menús, otro de sus propósitos es proveer un acceso más rápido a las herramientas que consideramos de uso más común a la hora de editar código.



**Figura 2.2**  
**Barra de Herramientas**

La barra se compone por 6 íconos separados en 3 pequeñas secciones.



Empezando de izquierda a derecha tenemos el botón de “Analizar Código”, su función es tal y como se entiende, inicia el análisis del código escrito dentro del

**Figura 2.2.1** editor de código.  
**“Analizar**

Continuando hacia la derecha está la siguiente sección que contiene algunas de las herramientas del menú “Archivo” (Figura 2.2.1), la carpeta amarilla cumple con la función de “Abrir” el disquete azul cumple con la función de “Guardar” y el disquete con el lápiz es para la función “Guardar Como...”, la explicación de cada una de estas funciones se encuentran en el subtema “Barra de menús” en la página 5.



**Figura 2.2.2**  
**Abrir, Guardar y Guardar**



**Figura 2.2.3**  
**Deshacer v**

Para finalizar tenemos las siguientes flechas azules (Figura 2.2.3), la flecha de la izquierda es para “Deshacer” un cambio hecho, una manera más fácil para identificar esta función es mencionarla con su combinación de teclas, la muy conocida (Ctrl+Z) y su contraparte “Rehacer” (Ctrl+Y) que permite regresar a un estado antes de retroceder usando (Ctrl+Z).

### 2.3.-Editor de Código

En esta parte del compilador es donde se ingresa todo el código que se desea analizar. (Figura 2.3)



**Figura 2.3**  
**Code Editor**

El usuario es libre de escribir lo que guste en esta sección, pero eso no garantiza que lo que escriba esté correcto y respete el lenguaje, eso se demostrará cuando decida 'Analizar su código' para así ver que elementos pertenecen al lenguaje y cuales sobran.

Es importante mencionar que el *editor de código* o también *Code Editor* cuenta a su izquierda con un *contador de líneas* el cual es útil para visualizar como ingresamos nuestro código, saber en qué línea estamos además que es una ayuda extra para desplazarnos por el código.

Cuando inicie el compilador MJU le proporcionará la sintaxis básica para empezar a codificar en él, tal como puede ver en la figura anterior (2.3).

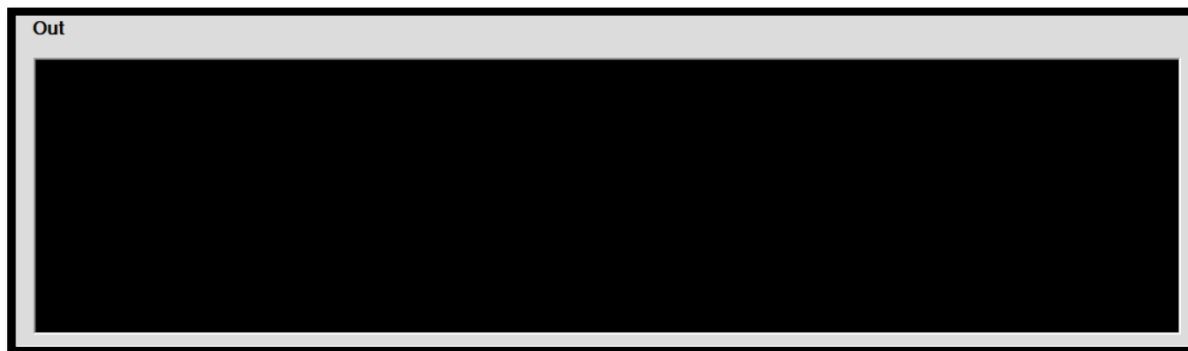
Lo primero que se requiere es colocar las cabeceras que se requieran para llevar a cabo un programa, por default se coloca `<io.gg>` el cual sirve para manipular la entrada y salida (input/output).



Todo este código lo puede quitar el usuario, pero es importante aclarar que es lo más básico para programar en nuestro lenguaje, así que de cualquier manera será necesario que lo coloque si quieres realizar un programa, esto solo es una ayuda para evitar tener que teclear tanto y hacer a un lado el trabajo monótono de estar escribiendo lo mismo cada que se quiera crear un programa nuevo.

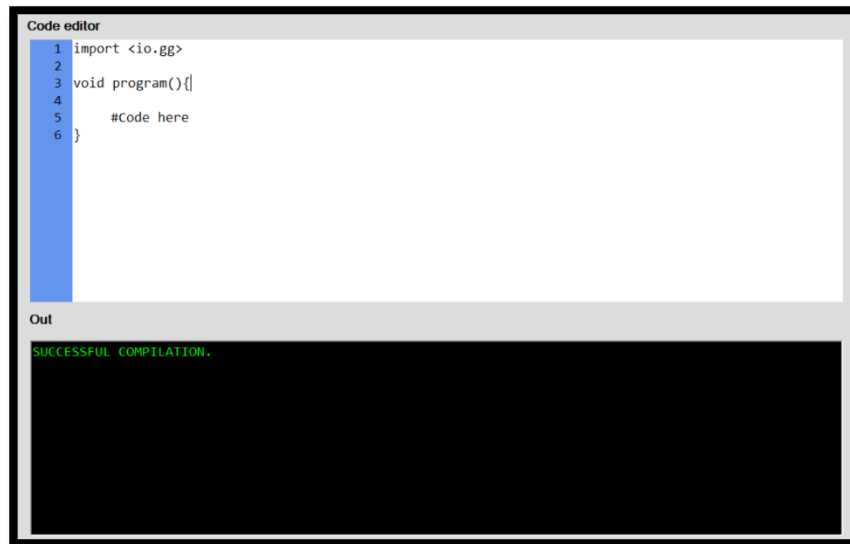
## **2.4.-Despliegue de resultados**

La parte de *salida* o también conocida por *out* (Figura 2.4) sirve para arrojar información útil para el usuario, como lo son los resultados del análisis además de otra parte por mencionar más adelante, esta parte es controlada por el compilador y no por el usuario, lo que significa que el usuario no puede controlar de manera voluntaria el texto que despliega.



**Figura 2.4**  
**Out**

Existen dos opciones que puede arrojar dependiendo a la sintaxis del código escrito, la primera es que devuelva un análisis realizado con éxito “SUCCESSFUL COMPILATION” (Figura 2.4.1), la otra posible salida es que se encuentra un error <syntax error> acompañado de la línea donde se encuentra dicho error, así como también el tipo de error, ya sea porque hace falta algún elemento en la sintaxis del programa(Figura 2.4.2) o porque se encuentra un elemento que no es perteneciente al lenguaje(Figura 2.4.3).

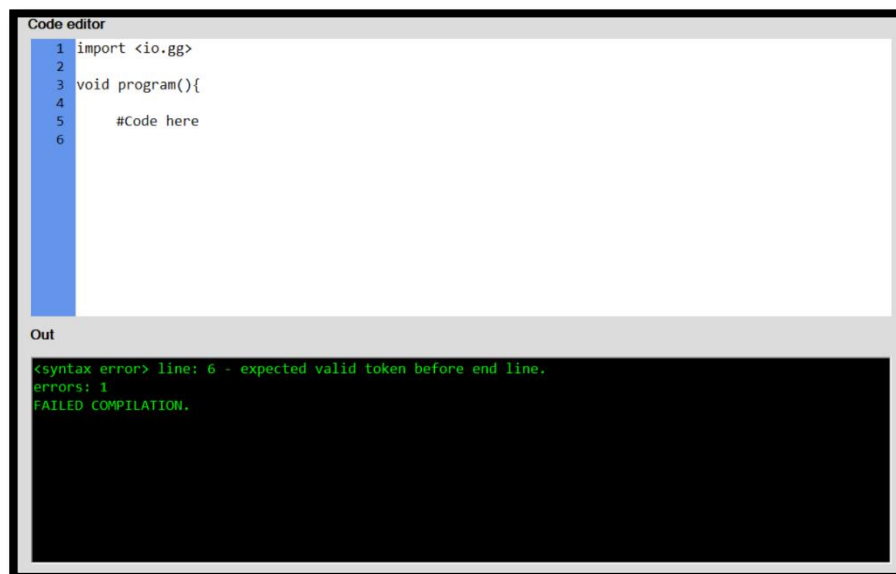


The screenshot shows a code editor window with the title "Code editor". The code is as follows:

```
1 import <io.gg>
2
3 void program(){
4
5     #Code here
6 }
```

Below the code editor is an "Out" panel. It displays the text "SUCCESSFUL COMPILATION." in green, indicating that the code was compiled without any errors.

**Figura 2.4.1**  
**Código sin errores**



The screenshot shows a code editor window with the title "Code editor". The code is identical to the one in Figure 2.4.1:

```
1 import <io.gg>
2
3 void program(){
4
5     #Code here
6 }
```

Below the code editor is an "Out" panel. It displays the following error message in green:

```
<syntax error> line: 6 - expected valid token before end line.
errors: 1
FAILED COMPILATION.
```

This indicates a syntax error on line 6, where the compiler expected a valid token before the end of the line.

**Figura 2.4.2**  
**Código con error sintáctico**

```
Code editor
1 import <io.gg>
2
3 void program(){
4 |
5 #Code here
6 }

Out
<lexico error> line: 4 - unrecognized token '|'
errors: 1
FAILED COMPILATION.

Temporary File
```

**Figura 2.4.3**  
**Código con error léxico**

Preste atención en el mensaje que arroja cada tipo de error para así poder identificarlos de manera correcta a la hora de corregir los errores.

En la página anterior mencionamos que esta sección sirve para desplegar información útil para el usuario, bueno, es importante decir que debajo de la zona de los resultados del análisis está la barra de ubicación, esta solo indica el nombre del archivo que estamos manipulando (Figura 2.4.3), En el tema 3 explicaremos como cambia.

### 3.-Uso del compilador

Con el conocimiento que se ha adquirido a lo largo de la lectura de este manual se procederá a realizar un ejemplo ocupando los procesos principales del compilador 'MJU'.

Para ello realizaremos un programa que diga "¡Hola mundo!".

#### 3.1.-Escribir y analizar un programa

Empezamos escribiendo nuestro programa en la sección *editor de código* (Figura 3.1), posteriormente pasamos a analizar lo ingresado dándole click al *botón analizar* ubicado en la *Barra de herramientas*.



**Figura 3.1**  
**Programa "Hola mundo"**

Como se puede notar la salida del análisis arroja que el código está elaborado perfectamente, debajo de eso vemos que la barra de ubicación dice "Temporary File" esto es porque se está trabajando sobre el archivo temporal que MJU ocupa para analizar.

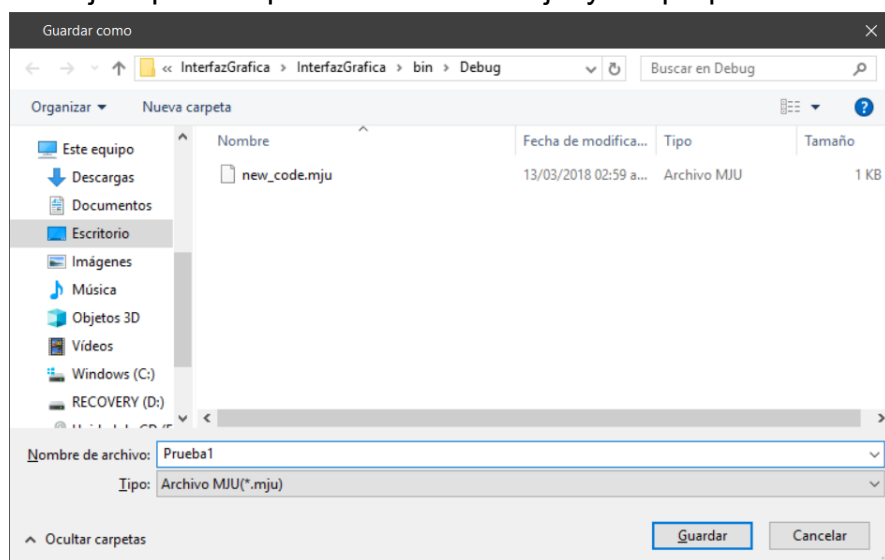
### ¿Para qué sirve y qué es el archivo temporal?

Bueno, explicado de una forma sencilla para el usuario, para que MJU pueda analizar un código este debe estar dentro de un archivo “.mju”, para ello la opción más lógica sería guardar el código antes de analizarlo, pero no es algo que se quiera hacer siempre, muchas veces como usuarios queremos resultados rápidos, para ello existe el “Temporary File”, simplemente crea un archivo “.mju” para así poder analizar cualquier código que se escriba dentro del *editor de código* sin necesidad de tener que guardarlo antes en un archivo.

Este archivo temporal no puede ser modificado, para un usuario que ha leído todo este manual seguro piense ¿Y si decido “Guardar” mientras estoy en el Temporary File y no “Guardar Como”? Para evitar el conflicto de modificar directamente el archivo temporal al pulsar “Guardar” mientras estamos en él, MJU lo tomará de la manera en la que tomaría un “Guardar Como”, obligando al usuario a crear un nuevo archivo para así guardar sus avances.

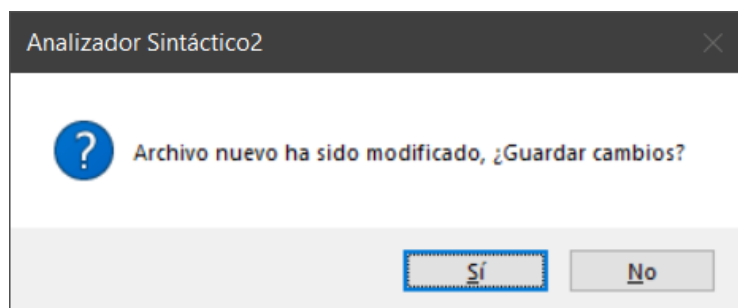
## **3.2.-Guardar un archivo**

Una vez analizado el código y estar conforme con lo que se ha ingresado podemos escoger la opción de *Guardar como*, ya sea mediante el menú ‘Archivo’ o con la combinación de teclas ya mencionada, al querer guardar se despliega una ventana emergente donde podemos asignar un nombre a nuestro archivo sin preocuparse por especificar el tipo de archivo que manejará puesto que la extensión ‘.mju’ ya la proporciona el compilador.

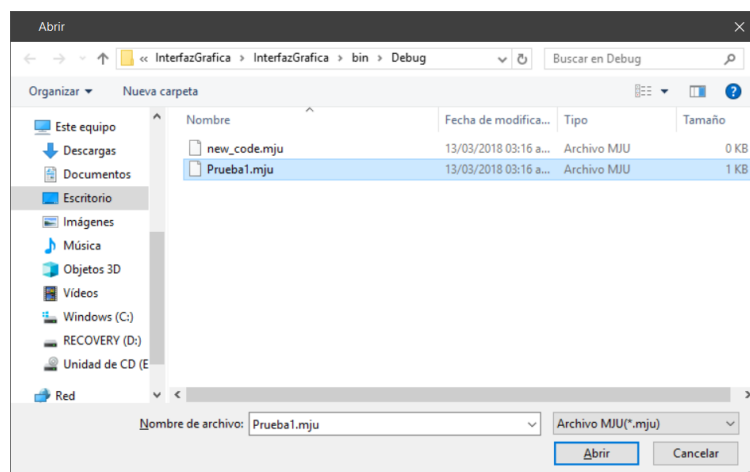


**Figura 3.3**  
**‘Guardar como’**

Es importante mencionar que, si el usuario realiza algún cambio en un archivo y decide cerrar el programa sin guardar cambios, MJU desplegará el siguiente mensaje:



### 3.3.-Abrir un archivo



Teniendo abierto el compilador accedemos a la opción de *abrir* ya sea por el menú 'Archivo' o con su combinación de teclas correspondiente. Es importante aclarar que el compilador 'MJU' sólo nos permitirá abrir archivos cuya extensión sea homónima a él.

**Figura 3.4**  
**'Abrir'**

### **3.4.-Atajos de teclado**

A lo largo del manual hemos tocado varias veces el tema de la combinación de teclas, explicamos que dichas combinaciones están diseñadas para facilitar el uso del compilador, aquí proporcionamos las posibles combinaciones de teclas que se pueden realizar.

#### *Archivo*

|                     |              |
|---------------------|--------------|
| <i>Abrir</i>        | Ctrl+A       |
| <i>Guardar</i>      | Ctrl+S       |
| <i>Guardar Como</i> | Ctrl+Shift+S |
| <i>Salir</i>        | Ctrl+Alt+X   |

#### *Edición*

|                 |        |
|-----------------|--------|
| <i>Copiar</i>   | Ctrl+C |
| <i>Cortar</i>   | Ctrl+X |
| <i>Pegar</i>    | Ctrl+V |
| <i>Deshacer</i> | Ctrl+Z |
| <i>Rehacer</i>  | Ctrl+Y |

## **Conclusiones**

El desarrollo de este compilador ha sido posible por el uso de diferentes medios de información que han ayudado a entender y comprender los conceptos necesarios para la implementación de herramientas de programación para llevar a cabo este proyecto.

Visto de otra forma la elaboración de este proyecto ayudó a que nosotros comprendiéramos todo el proceso que existe detrás de un compilador, partiendo desde el diseño de un lenguaje hasta su implementación.



## Bibliografía

- 4o Ingeniería Informática II26 Procesadores de lenguaje Analizador léxico.
- El generador de analizadores léxicos lex. Teoría de Autómatas y lenguajes formales Federico Simmross Wattenberg (fedesim@infor.uva.es) Universidad de Valladolid.
- Introducción a Flex y Bison-Prácticas de Lenguajes, Gramáticas y Autómatas, cuarto cuatrimestre (primavera) de Ingeniería en Informática.
- Alfred V. Aho, Jeffrey D. Ullman, C. (s.f.). *Compiladores principios, técnicas y herramientas*.
- Gálvez Rojas, S., & Mora Mata, M. (2005). Java a tope: Traductores y Compiladores con Lex/Yacc, JFlex/CUP y JavaCC. Málaga, España: Universidad de Málaga.