

# Minor C

**Manual de Usuario**

Herbert Rafael Reyes Portillo

201612114

# Indice

---

## Contenido

Indice .....	2
Introducción .....	3
Carga de Archivo .....	4
Reportes .....	6
<b>Reporte de Errores</b> .....	7
<b>Reporte de la Tabla de Símbolos</b> .....	8
<b>Reporte de AST</b> .....	8
<b>Reporte Gramatical</b> .....	9
Vistas Principales .....	10
<b>Especificaciones Generales</b> .....	10
<b>Consola</b> .....	13
<b>Debugger</b> .....	14
Flujo de Aplicación .....	15

# Introducción

---

Augus es un lenguaje de programación basado en PHP y en MIPS. Su principal funcionalidad es ser un lenguaje intermedio, ni de alto nivel como PHP ni de bajo nivel como el lenguaje ensamblador MIPS.

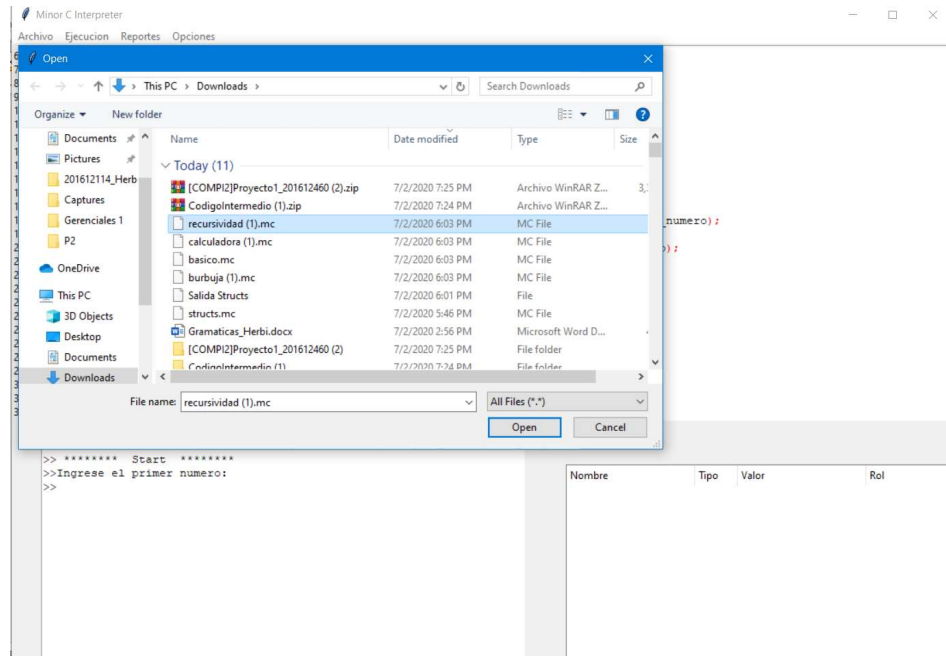
El lenguaje tiene dos restricciones: La primera es que cada instrucción es una operación simple, y la segunda, es que cada instrucción hay un máximo de dos operandos y su asignación.

Es un lenguaje débilmente tipado, sin embargo, si se reconocen cuatro tipos de datos no explícitos: entero, punto flotante, cadena de caracteres y arreglo.

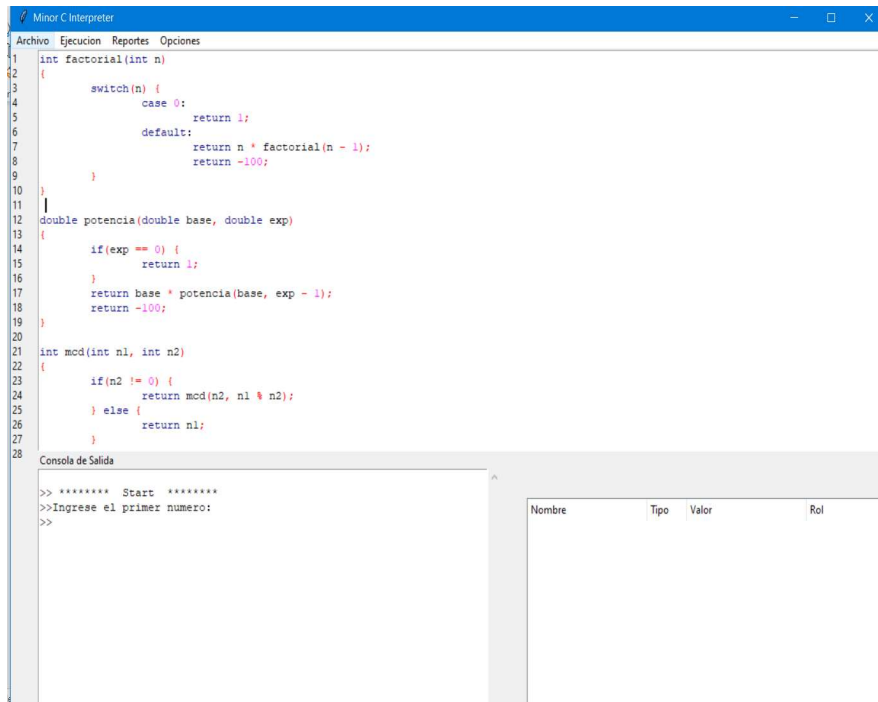
Para manejar el flujo de control se proporciona la declaración de etiquetas, sin tener palabras reservadas para ese uso.

# Carga de Archivo

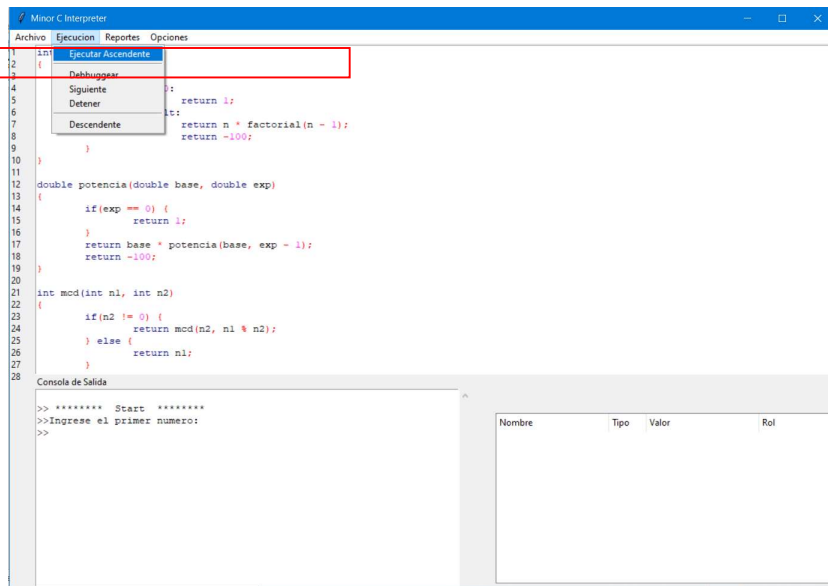
1. En el menú de archivo se encuentra la opción de abrir, se selecciona y esta despliega un cuadro de diálogo donde se debe seleccionar el archivo a analizar.



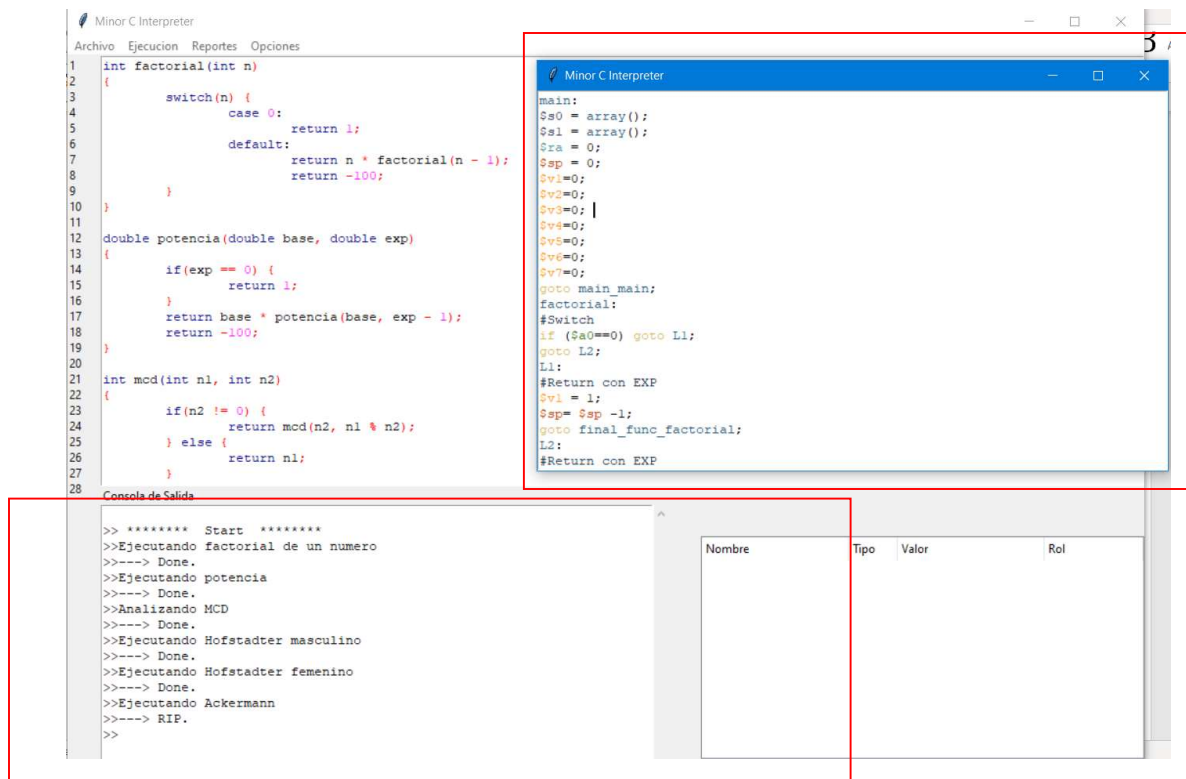
2. En la pestaña del lado izquierdo aparecerá el contenido del archivo seleccionado



### 3. Para su análisis se debe seleccionar analizar

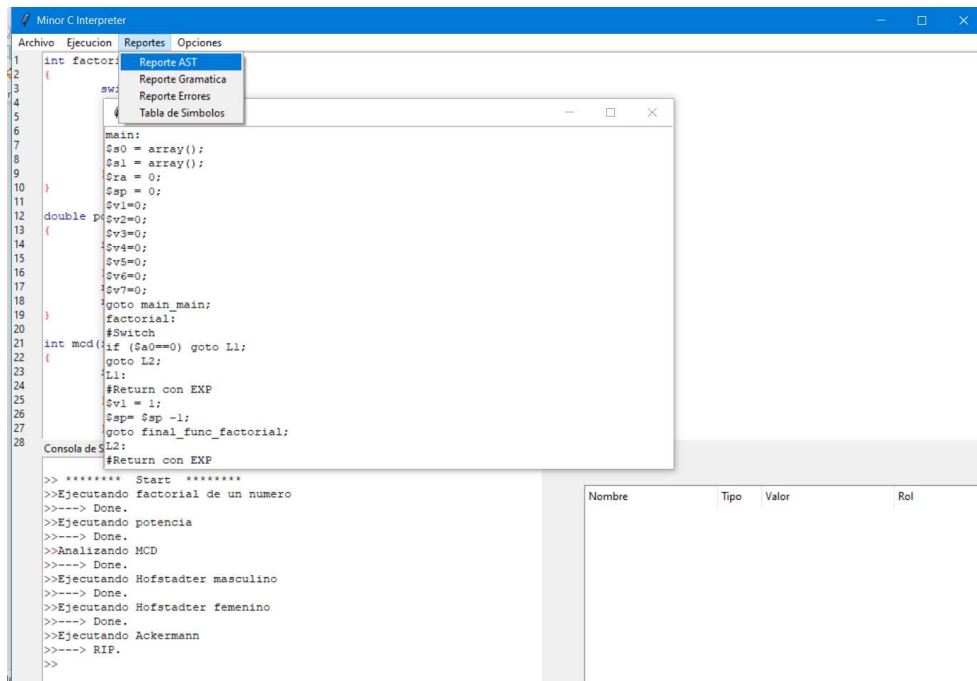


### 4. A continuación, se puede ver la ejecución del código traducido y el código augus generado



# Reportes

Después de Finalizada la Ejecución, el usuario puede seleccionar cualquiera de los reportes que aparecen en el menú



## Reporte de Errores

El reporte de errores muestra los errores léxicos, sintácticos y semánticos encontrados al momento del análisis del archivo.

The screenshot displays the Minor C Interpreter interface. The main window shows the source code of a C program with several errors. An error report window is open, listing the detected errors.

**Source Code:**

```
1 int factorial(int n)
2 {
3     switch(n) {
4         case 0: return 1;
5         default:
6             return n * factorial(n - 1);
7         return -100;
8     }
9 }
10 dfdgad#####
11 double potencia(double base, double exp)
12 {
13     if(exp == 0) {
14         return 1;
15     }
16     return base * potencia(base, exp - 1);
17     return -100;
18 }
19
20 int mod(int n1, int n2)
21 {
22     if(n2 != 0) {
23         return mod(n2, n1 % n2);
24     } else {
25         return n1;
26     }
27 }
28
```

**Error Report Window:**

Tipo	Descripcion	Fila	Columna
Lexico	Caracter ilegal '@'	11	8
Lexico	Caracter ilegal '@'	11	9
Lexico	Caracter ilegal '@'	11	10
Lexico	Caracter ilegal '@'	11	11
Lexico	Caracter ilegal '@'	11	12
Lexico	Caracter ilegal '@'	11	13
Lexico	Caracter ilegal '@'	11	14
Lexico	Caracter ilegal '@'	11	15
Lexico	Caracter ilegal '@'	11	16
Sintactico	double	12	1

**Console de Salida:**

```
>>Error: Verifique errores lexicos y sintacticos
>>
```

**Error Table:**

Nombre	Tipo	Valor	Rol
--------	------	-------	-----

El reporte muestra todas las variables, funciones y procedimientos que fueron declarados, así como su tipo y toda la información relacionada a cada una de ellas.

No	Tipo	Nombre	Ambito	Dimensiones	Temporal Referencia
1	TIPO_DATOS.INT	n	ackermann	0	\$a8
2	TIPO_DATOS.INT	m	ackermann	0	\$a7
3	TIPO_DATOS.INT	n	hofstaderMasculino	0	\$a6
4	TIPO_DATOS.INT	n	hofstaderFemenina	0	\$a5
5	TIPO_DATOS.INT	n2	mcd	0	\$a4
6	TIPO_DATOS.INT	n1	mcd	0	\$a3
7	TIPO_DATOS.DOUBLE	exp	potencia	0	\$a2
8	TIPO_DATOS.DOUBLE	base	potencia	0	\$a1
9	TIPO_DATOS.INT	n	factorial	0	\$a0

No	Tipo	Nombre	Cantidad Parametros
1	INT	factorial	1
2	DOUBLE	potencia	2
3	INT	mcd	2
4	INT	hofstaderFemenina	1
5	INT	hofstaderMasculino	1
6	INT	ackermann	2

El reporte de AST muestra el árbol que se produjo en el análisis sintáctico:





## Reporte Gramatical

El reporte Gramatical muestra las acciones gramaticales realizadas durante la ejecución

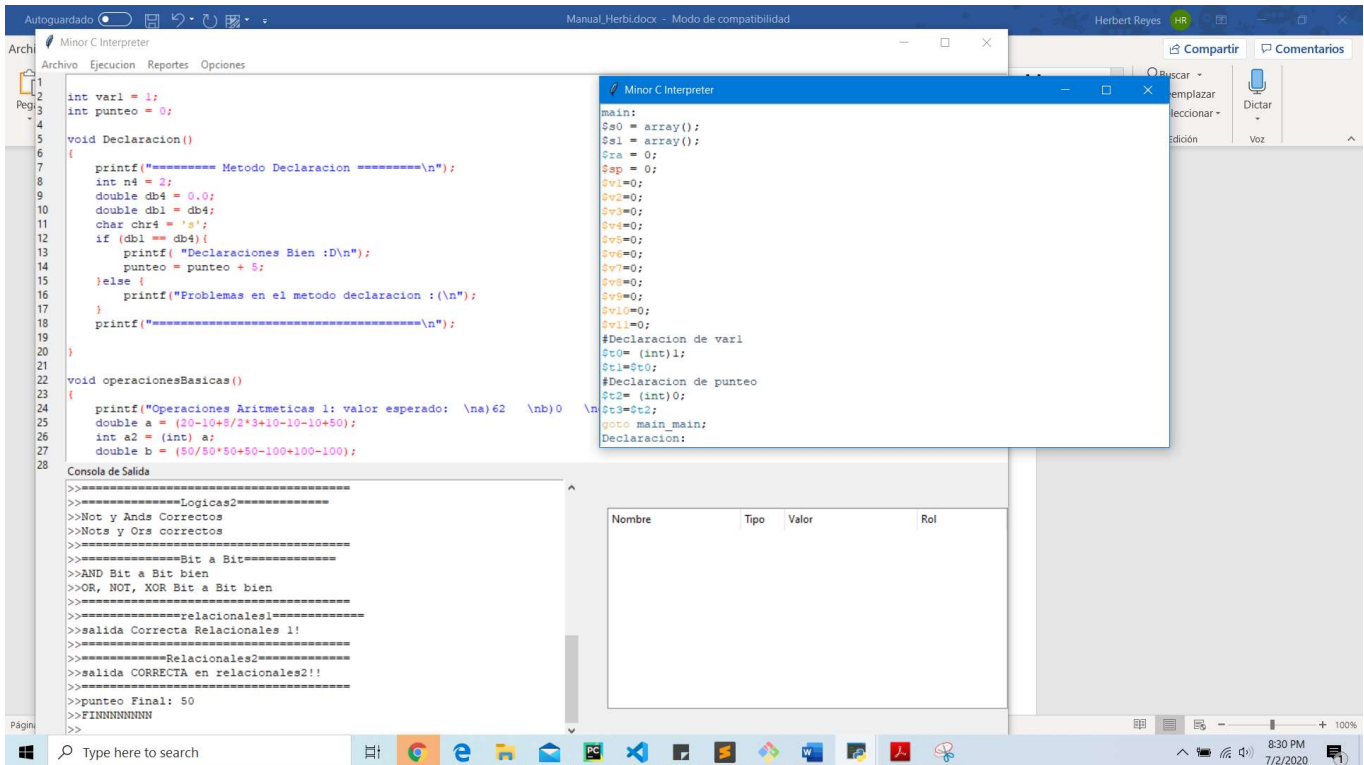
### Reporte Gramatical

Produccion	Regla Semántica
S -> Sentencias_G	t[0]=t[1]
Sentencias_G -> Sentencias_G Sentencia_G	t[1].extend(t[2]) t[0]=t[1]
Sentencia_G -> Declaracion   LASIGNACION pyc   STRUCT pyc   Funcion	t[0]=t[1]
Funcion -> Tipos iden par1 Parametros par2 BLOQUE	t[0]=[Funcion(t[1],t[2],t[4],t[6],t.slice[2].lino,find_column(input,t.slice[2]))]
BLOQUE -> llav1 SS_F llav2	t[0]=t[2]
SS_F -> SS_F S_F	t[1].extend(t[2]) t[0]=t[1]
S_F -> Declaracion   LASIGNACION pyc   STRUCT pyc   IF   LLAMADA pyc   BREAK pyc   CONTINUE pyc   SWITCH   DO_WHILE pyc   WHILE   RETURN pyc   ETIQUETA   GOTO pyc   FOR   INSTANCIA pyc   INC_POST pyc   INC_PRE pyc	t[0]=t[1]
SS_F -> SS_F S_F	t[1].extend(t[2]) t[0]=t[1]

# Vistas Principales

## IDE Minor C

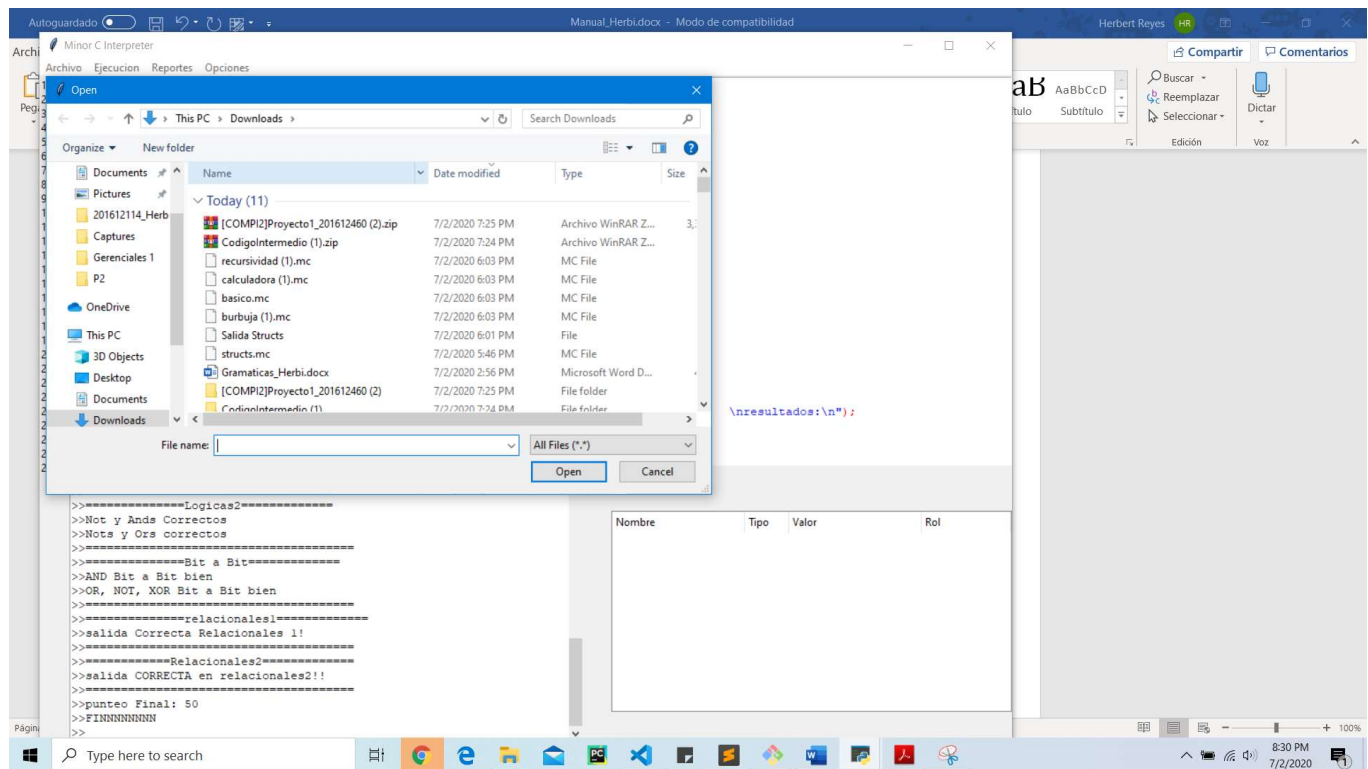
El IDE de MinorC posee varias características para facilitar el análisis del código fuente, cada una de ellas será explicada a continuación.



## Especificaciones Generales

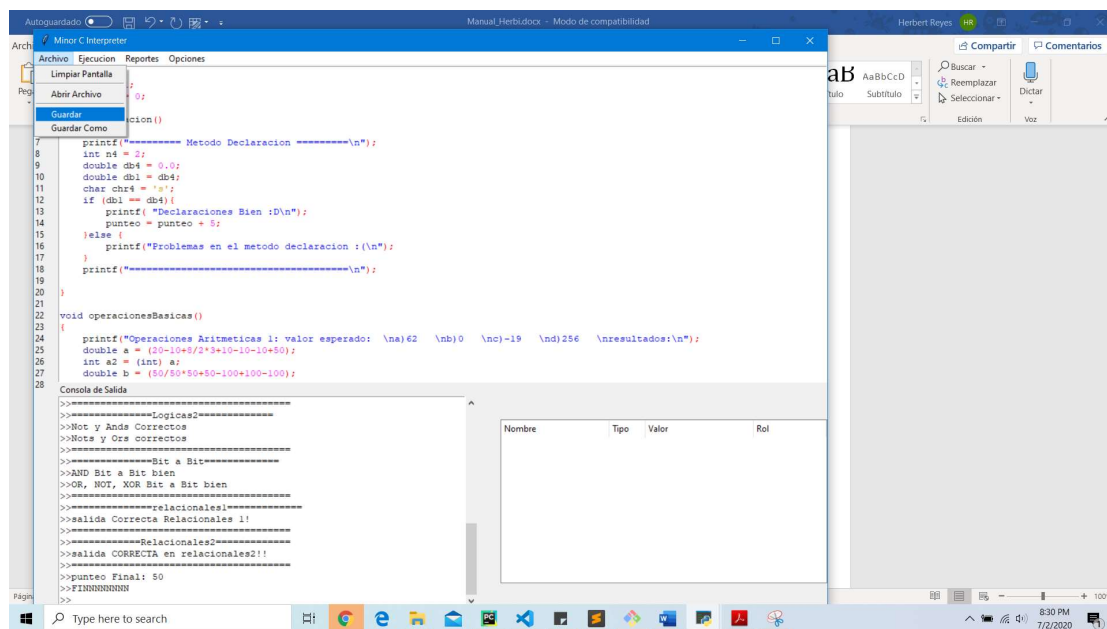
## Abrir:

Este botón abrirá un cuadro de dialogo para seleccionar un archivo existente y mostrará su contenido en la pestaña actual.



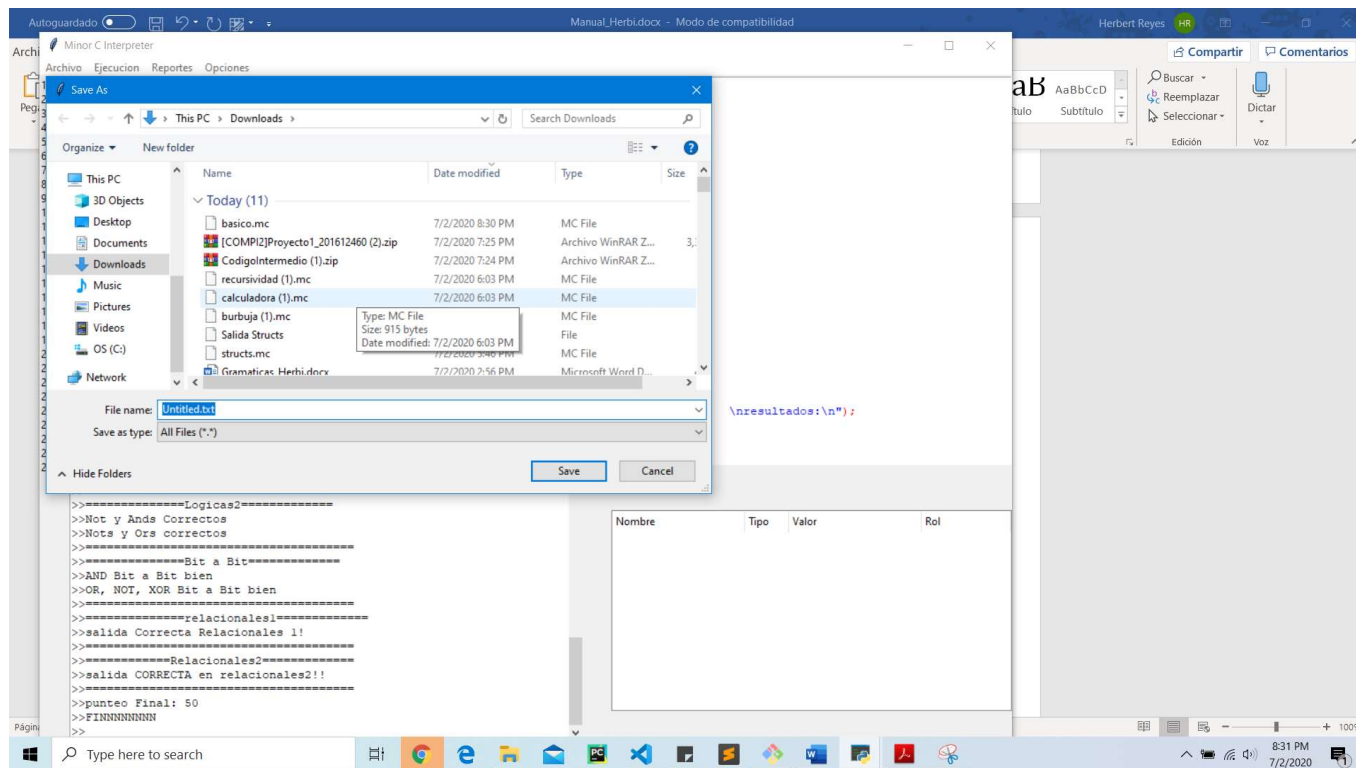
## Guardar:

Este botón permite guardar los cambios realizados en el archivo actual.



## Guardar Como:

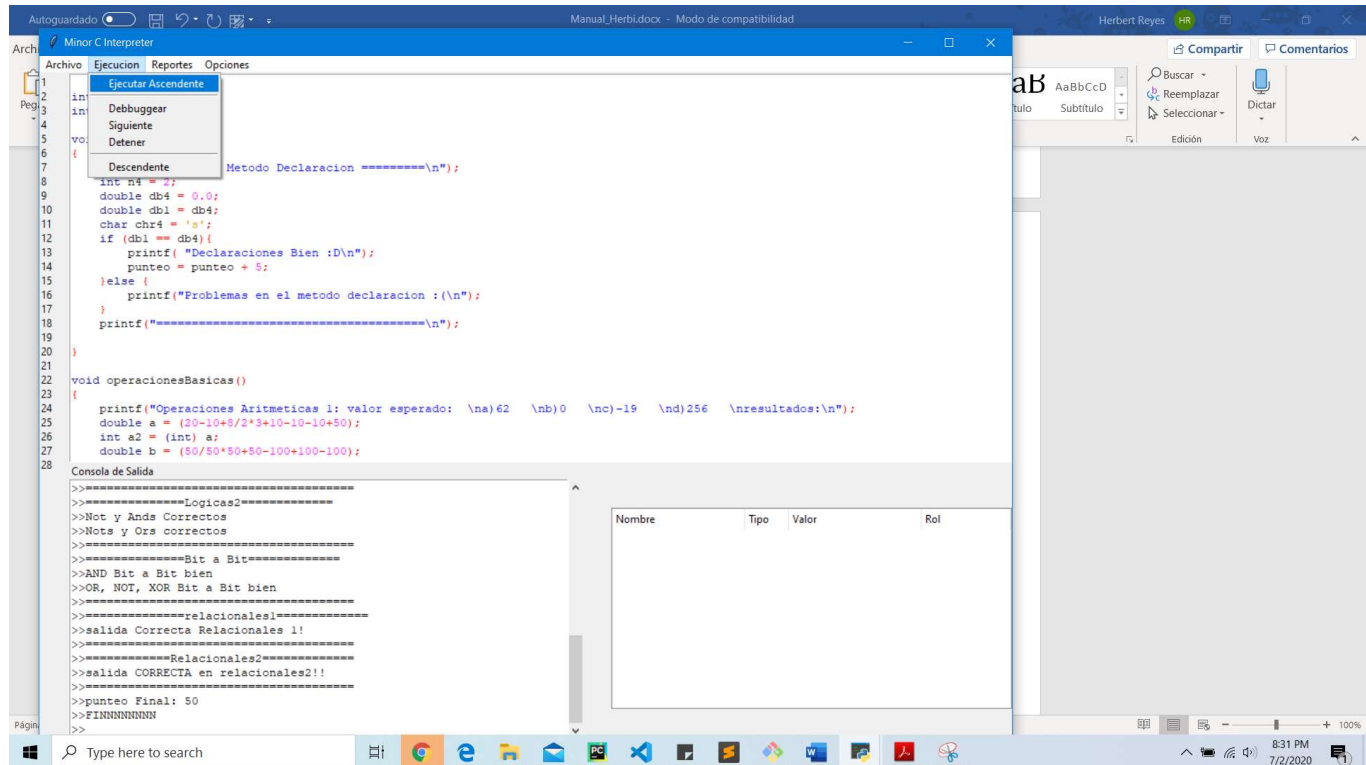
Al presionar este botón se mostrará un cuadro de dialogo para seleccionar la ubicación y el nombre del archivo a guardar.



## Botón de Ejecucion:

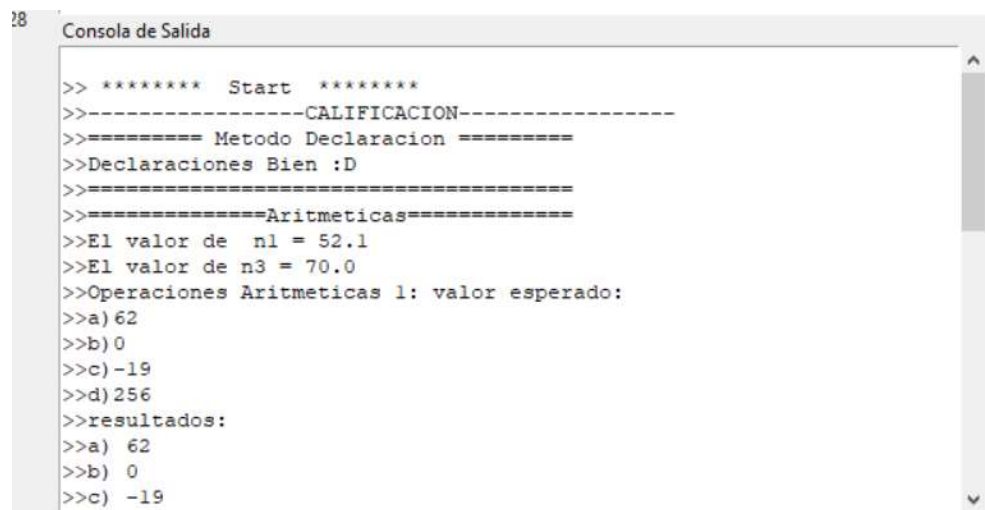
Arit permite la opción de elegir la forma en la que se analizará el archivo:

- Ascendente



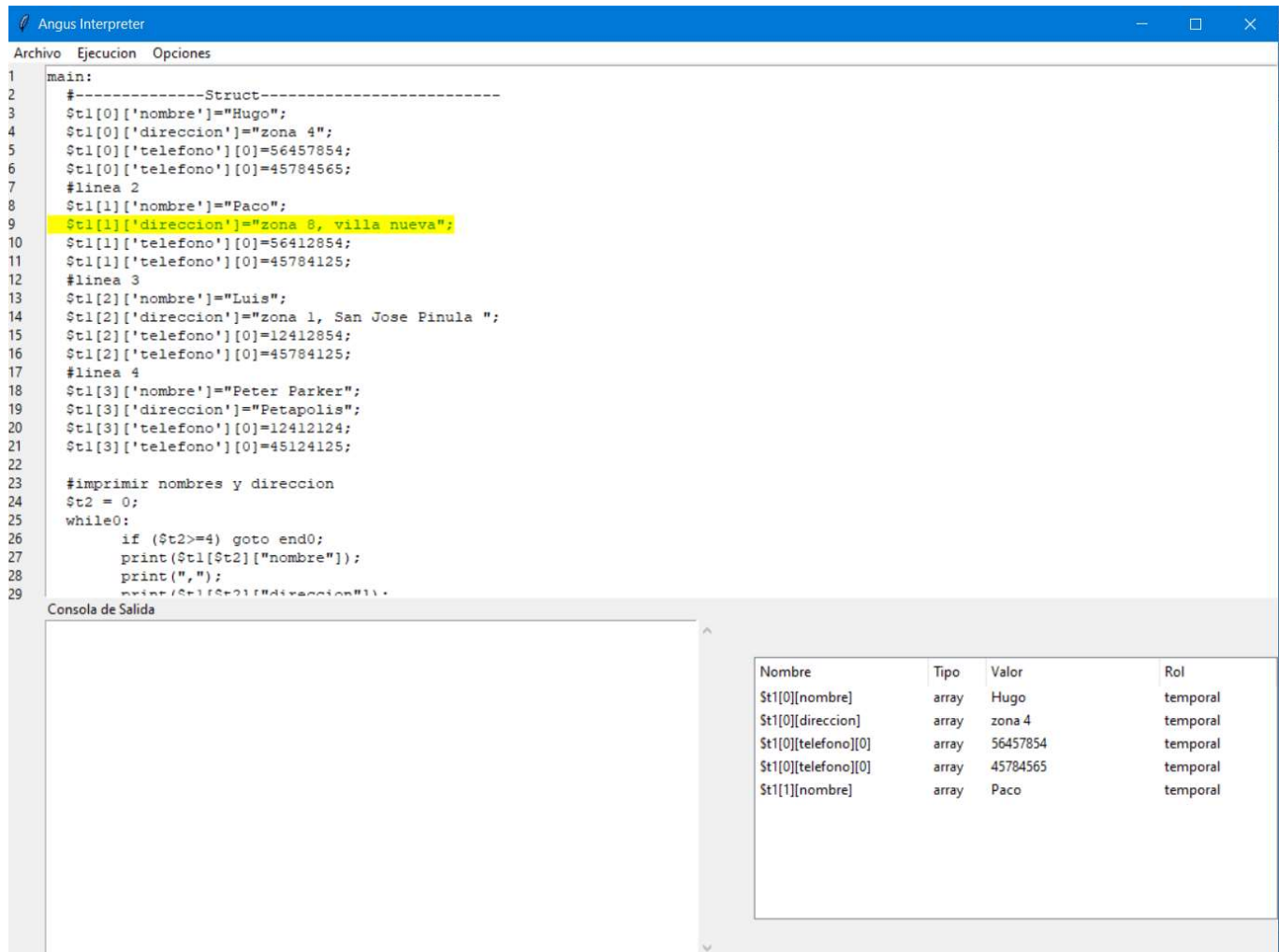
## Consola

Es el área donde se mostrarán los errores, advertencias o impresiones que se detectaron durante el transcurso del análisis de un archivo de entrada.



## Debugger

Cuando deseamos realizar el modo debbuger basta con darle debbuger en el menu.



The screenshot shows the Angus Interpreter window with a menu bar (Archivo, Ejecucion, Opciones) and a code editor. The code is a C program that defines a struct for a person and prints their details. The line number 9 is highlighted, and the variable `$t1[1]['direccion']` is highlighted in yellow. Below the code editor is a console window titled 'Consola de Salida' which is empty. To the right of the console is a table showing the current state of variables.

Nombre	Tipo	Valor	Rol
<code>\$t1[0][nombre]</code>	array	Hugo	temporal
<code>\$t1[0][direccion]</code>	array	zona 4	temporal
<code>\$t1[0][telefono][0]</code>	array	56457854	temporal
<code>\$t1[0][telefono][0]</code>	array	45784565	temporal
<code>\$t1[1][nombre]</code>	array	Paco	temporal

Nos ira indicando paso a paso que línea se va ejecutando y además, en la consola de debbuger tendremos como se muestra el avance de las variables en tiempo real, para ir continuando con el debbuger le damos siguiente y si deseamos pararlo, solamente basta con seleccionar la opción de detener.

## Flujo de Aplicación

---

Arit Software inicia su flujo de aplicación desde la carga del archivo que contiene el programa fuente hasta la creación de reportes o gráficas según lo analizado.

