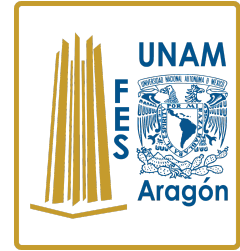




UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

Ingeniería en Computación

Compiladores

Proyecto Compiladores

Prof. Martin Romero Ugalde

Integrantes:

ESTRADA BARBOSA ERICK EDUARDO

HERNANDEZ SOLEDAD ANGEL AGUSTÍN

IBIETA OLVERA JAZMIN SARAI

PRADO GONZÁLEZ JOSÉ LEONARDO

ÍNDICE

INTRODUCCIÓN	4
CAPÍTULO 1: GENERALIDADES	4
Sobre:	4
Objetivo:	4
Requerimientos del sistema:	4
Ayuda adicional:	5
Aho, A. V. (1990). COMPILADORES PRINCIPIOS TÉCNICAS Y HERRAMIENTAS (1a. ed.). WILMINGTON: ADDISON WESLEY IBEROAMERICANA.	5
CAPÍTULO 2: LENGUAJE	5
Sobre el lenguaje:	5
Palabras reservadas y detalles específicos:	5
Estructura general de un programa:	8
CAPÍTULO 3: COMPILADOR	12
Sobre el compilador:	12
Vista general:	12
Vista general, especificaciones:	12
Botones:	12
Tabla de Tokens:	15
Área de programación:	15
Area de errores y resultado de compilación:	16
Vistas adicionales	16
Guardar	16
Abrir	17
Tripletas	18
Código Objeto	19
Ensamblador	20

CAPÍTULO 4: CRUD	21
Crear un nuevo proyecto:	21
Guardar un proyecto:	22
Abrir un proyecto existente:	23
Ingresar Código:	24
Compilación:	25
Generación de Código intermedio	26
Generacion deCodigo Objeto	27
Optimización	28
Generación de código ensamblador	29
Apéndices:	29
Apéndice A:	29
Apéndice B:	32
B.1: Ejemplo Funcional:	32
B.2: VideoTutorial:	36

INTRODUCCIÓN

Este documento contiene las primeras especificaciones para el desarrollo del anteproyecto “CodeChair”, con el objetivo principal de mejorar la calidad de vida general de las personas con discapacidad física o aquejadas. Aspira a hacerlo implementando ciertas características y capacidades a una silla de ruedas. Estas características incluyen la capacidad de mover la silla de ruedas de forma autónoma a un lugar específico, proporcionando y recibiendo información sobre el entorno del usuario, así como facilitar las interacciones con los electrodomésticos con la ayuda de dispositivos tecnológicos inalámbricos.

CAPÍTULO 1: GENERALIDADES

Sobre:

El principal enfoque de nuestro proyecto consta en la automatización de una silla de ruedas común y corriente, con lo que lograremos conseguir que el usuario de la silla de ruedas tenga una experiencia más cómoda al permitirle ciertas funciones que le facilitarán la realización de necesidades básicas, tales como: movimiento de la silla, activación de freno, activación de alertas para facilitar el traslado del usuario, equipo de iluminación para facilitar la movilidad, envío de mensaje de auxilio a contacto en caso de incidente, desplazamiento a ciertas áreas definidas por el usuario entre otras funciones.

Objetivo:

Diseñar una guía para usuarios comunes, que sea capaz de sintetizar las instrucciones y explicaciones para el correcto funcionamiento del proyecto presentado anteriormente, el cual emplea un lenguaje de programación creado por nosotros mismos.

Requerimientos del sistema:

Procesador	Intel Pentium 2 226MHz.
RAM	4 GB
Espacio disponible en disco	300 MB
Sistema Operativo	Windows 7, Windows 10
Ambiente de Ejecución (JRE + Update)	126 MB de espacio disponible en disco.

JDK (Herramientas de desarrollo + código fuente)	272 MB de espacio disponible en disco.

Ayuda adicional:

Aho, A. V. (1990). *COMPILADORES PRINCIPIOS TÉCNICAS Y HERRAMIENTAS* (1a. ed.). WILMINGTON: ADDISON WESLEY IBEROAMERICANA.

CAPÍTULO 2: LENGUAJE

Sobre el lenguaje:

El compilador maneja un lenguaje simple y poco preciso el cual nos permite principalmente llevar a cabo la programación del sistema utilizando estructuras de control y operaciones que nos sirvan para este propósito.

Palabras reservadas y detalles específicos:

El programa cuenta con 11 tipos de tokens y 25 palabras reservadas.

Tokens.

Token	Descripción	Ejemplo
Palabras Reserva	Son las cadenas que identifican las funciones reservadas	IMPORT, STRING, INT
Operadores	Caracteres +, -, /, *, =	+ ó - ó / ó * ó =
Comparación	Caracteres ==, !=, <, >, >=, <=	== ó != ó < ó > ó >= ó <=
Separadores	Caracteres ., :, ;, ' "	. ó , ó : ó ; ó ' ó "
AbiertoCerrado	Caracteres (,), {, }, [,]	(ó) ó { ó } ó [ó]

COMPILADOR CHAIRCODE

Pasos	Caracteres ++, -- (incremento decremento)	++ ó --
id	Letra seguida por letras y numeros	Contador1, ResultadoMul
número	Cualquier dígito	1, 4892, 57, 192
SaltoLinea	Caracteres \n indicando el salto de línea	\n
Comentario	Comentario de línea indicado por //	//Comentario, // Ola
Error	Cualquier caracter no valido	ñ, , ~

Palabras Reservadas:

Palabra reservada	Descripción	Ejemplo
IMPORT	Cadena mayus y min I,M,P,O,R,T	IMPORT ó import

DEF	Cadena mayus y min D, E, F	DEF ó def
CLASS	Cadena mayus y min C, L, A, S, S	CLASS ó class
IF	Cadena mayus y min I, F	IF ó if

COMPILADOR CHAIRCODE

ELSE	Cadena mayus y min E, L, S, E	ELSE ó else
FOR	Cadena mayus y min F, O, R	FOR ó for
IN	Cadena mayus y min I, N	IN ó in
RANGE	Cadena mayus y min R, A, N, G, E	RANGE ó range
SELF	Cadena mayus y min S, E, L, F	SELF ó self
WHILE	Cadena mayus y min W, H, I, L, E	WHILE ó while
TRY	Cadena mayus y min T, R, Y	TRY ó try
EXCEPT	Cadena mayus y min E, X, C, E, P, T	EXCEPT ó except
RETURN	Cadena mayus y min R, E, T, U, R, N	RETURN ó return
BREAK	Cadena mayus y min B, R, E, A, K	BREAK ó break
NEXT	Cadena mayus y min N, E, X, T	NEXT ó next
INPUT	Cadena mayus y min I, N, P, U, T	INPUT ó input
PRINT	Cadena mayus y min P, R, I, N, T	PRINT ó print
INT	Cadena mayus y min I, N, T	INT ó int
FLOAT	Cadena mayus y min F, L, O, A, T	FLOAT ó float

STRING	Cadena mayus y min S, T, R, I, N, G	STRING ó string
POWER	Cadena mayus y min P, O, W, E, R	POWER ó power
SQRT	Cadena mayus y min S, Q, R, T	SQRT ó sqrt
AND	Cadena mayus y min A, N, D	AND ó and
OR	Cadena mayus y min O, R	OR ó or
NOT	Cadena mayus y min N, O, T	NOT ó not

Estructura general de un programa:

Reglas del Lenguaje:

1. Las palabras reservadas deben ser rigurosamente escritas ó todas en mayúsculas ó todas en minúsculas para considerarse como tal.
2. Los caracteres inválidos (No añadidos al alfabeto) son catalogados como caracteres inválidos
3. Los identificadores deben comenzar por una letra mayúscula o minúscula, seguida de letras y/o números.
4. Todo carácter abierto ((, {, [) deberá tener su carácter para cerrar.
5. Los operadores, deben de ser seguidos de números menos el ++(incremento) y el -- (decremento).

Patrones Válidos:

```

/* Variables básicas de comentarios y espacios */
TerminadorDeLinea = \r|\n|\r\n
EntradaDeCaracter = [^\r\n]
EspacioEnBlanco = {TerminadorDeLinea} | [ \t\f]
ComentarioTradicional = "/*" [^*] ~"*/" | "/*" "*" + "/"
FinDeLineaComentario = "/*" {EntradaDeCaracter}*
{TerminadorDeLinea}?
ContenidoComentario = ( [^*] | \*+ [^/*] ) *
ComentarioDeDocumentacion = "/*" {ContenidoComentario} "*" + "/"

```



```

/* Comentario */
Comentario = {ComentarioTradicional} | {FinDeLineaComentario} |
{ComentarioDeDocumentacion}

/* Identificador */
Letra = [A-Za-zÑñ_ÁÉÍÓÚáéíóúÜü]
Digito = [0-9]
Identificador = {Letra}({Letra}|{Digito})*
Numero = {Digito} ({Digito})*

/* Número */
Numero = 0 | [1-9][0-9]*
%%

/* Comentarios o espacios en blanco */
{Comentario}|{EspacioEnBlanco} { /*Ignorar*/ }

/*Numero*/
REAL {Numero} "." {Numero}
NUMERO {Numero}

/*Operadores*/
"+" "SUMA"
 "-" "RESTA"
 "/" "DIVISION"
 "*" "MULTIPLICACION"

/*Logicos*/
"==" "IGUAL"
"!=" "DIFERENTE"
">" "MAYORQUE"
"<" "MENORQUE"
">=" "MAYORIGUALQUE"
"<=" "MENORIGUALQUE"

/*puntuacion*/
"." "PUNTO"
"," "COMA"
":" "DOSPUNTOS"
";" "PUNTOCOMA"
\" "COMILLASIMPLE"
\" [a-zA-Z0-9_.-]* \" "CADENA"
"=" "ASIGNACION"

\" "COMILLADOBLE"
\" ( "PARENTESISABIERTO"
\" ) "PARENTESISCERRADO"
\" { "LLAVEABIERTO"

```

```

\} "LLAVECERRADO"
\[ "CORCHETEABIERTO"
\[ "CORCHETECERRADO"

"++" "INCREMENTO"
"--" "DECREMENTO"

/*Palabras reservadas*/
"IMPORT" | "import" | "Import"
"DEF" | "def" | "Def"
"CLASS" | "class" | "Class"
"IF" | "if" | "If"
"ELSE" | "else" | "Else"
"FOR" | "for" | "For"
"IN" | "in" | "In"
"RANGE" | "range" | "Range"
"SELF" | "self" | "Self"
"WHILE" | "while" | "While"
"TRY" | "try" | "Try"
"EXCEPT" | "except" | "Except"
"RETURN" | "return" | "Return"
"BREAK" | "break" | "Break"
"NEXT" | "next" | "Next"
"INPUT" | "input" | "Input"
"OUTPUT" | "output" | "Output"
"PRINT" | "print" | "Print"
"INT" | "int" | "Int"
"FLOAT" | "float" | "Float"
"STRING" | "string" | "String"
"TRUE" | "true" | "True"
"FALSE" | "false" | "False"
"POWER" | "power" | "Power"
"SQRT" | "sqrt" | "Sqrt"
"AND" | "and" | "And"
"OR" | "or" | "Or"
"NOT" | "not" | "Not"
"BEGIN" | "begin" | "Begin"
"END" | "end" | "End"

/* IDs */
{Identificador} "ID"

```

CAPÍTULO 3: COMPILADOR

Sobre el compilador:

Un compilador es un programa que traduce un programa escrito en lenguaje fuente y produce otro equivalente escrito en un lenguaje destino. Lenguaje de alto nivel. Por ejemplo: C, Pascal, C++.

Vista general:

Al ejecutar el compilador aparecerá la siguiente pantalla:



El programa cuenta con 5 secciones visibles, la zona superior que cuenta con los logos del instituto tecnológico de tepic así como el título del proyecto, un menú con las opciones disponibles para realizar en el proyecto, la tabla de tokens que nos brinda información sobre el código ingresado al compilador, el panel de texto principal donde ingresamos el código que deseamos compilar y por último una zona de mensajes donde se mostrará el resultado de la compilación.

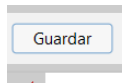
Vista general, especificaciones:

Botones:

En la parte superior del panel principal se encuentran los 7 botones que otorgan diversas funcionalidades a nuestro compilador.

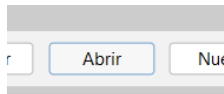


1. Guardar



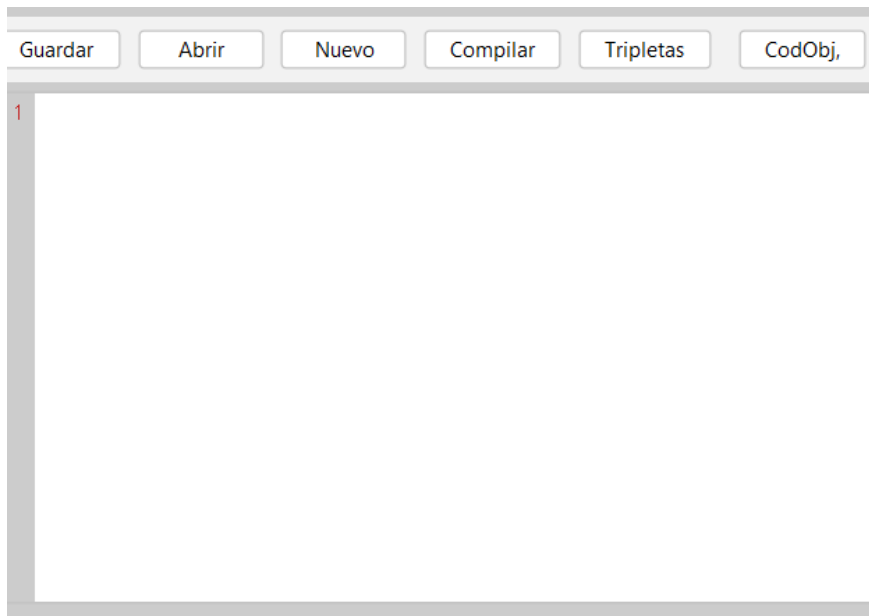
Al seleccionar este botón, se abrirá una ventana con el explorador de archivos que nos permitirá elegir el nombre y ubicación donde deseamos guardar el archivo donde estemos trabajando actualmente.

2. Abrir



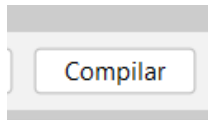
Al seleccionar el botón abrir nos mostrará una ventana emergente con el explorador de archivos, en la cual seleccionaremos el archivo que deseamos abrir en nuestro compilador

3. Nuevo



Al seleccionar este botón, la pantalla de texto será renovada por una en blanco, de manera que podamos trabajar en un archivo nuevo sin editar.

4. Compilar



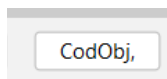
El botón compilar, como su nombre indica, compilara lo que se encuentre en el campo de texto, de manera que nos brinde diversa información en la tabla de tokens así como en la zona de mensajes de consola.

5. Tripletas



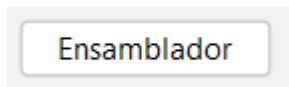
El botón tripletas nos brinda el código intermedio de lo que se encuentre en el campo de texto, mientras se tenga una estructura correcta en el código del mismo.

6. CodObj



El botón CodObj es una abreviatura de Código objeto, el mismo que se nos brinda al seleccionar dicho botón en forma de texto en una ventana emergente.

7. Ensamblador

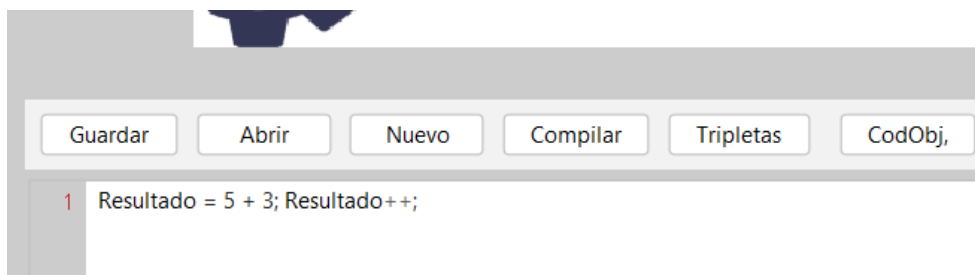


El botón ensamblador genera el mismo código mostrándolo a través de una ventana emergente.

Tabla de Tokens:

Tabla de Tokens		
Componente léxico	Lexema	[Línea, Columna]
ID	Resultado	[1, 1]
ASIGNACION	=	[1, 11]
NUMERO	5	[1, 13]
SUMA	+	[1, 15]
NUMERO	3	[1, 17]
PUNTOCOMA	;	[1, 18]
ID	Resultado	[1, 20]
INCREMENTO	++	[1, 29]
PUNTOCOMA	;	[1, 31]

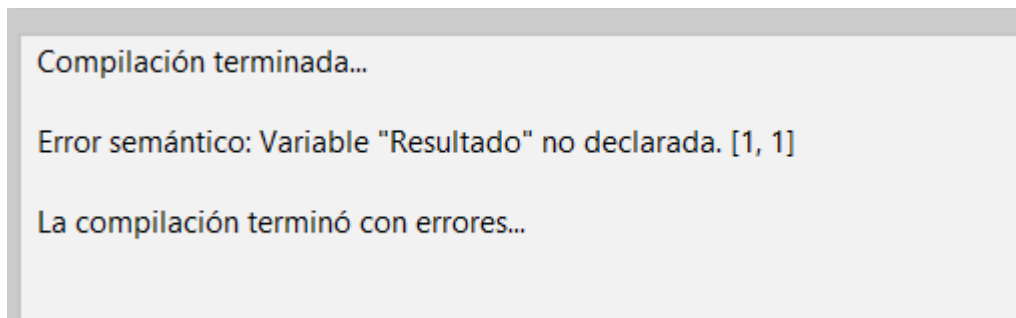
La tabla de tokens es un componente vital en todo compilador. Al declarar un identificador (normalmente una sola vez), éste es insertado en la tabla. Cada vez que se utilice el identificador se realizará una búsqueda en la tabla para obtener la información asociada (el valor). La tabla de tokens nos brindará información sobre el componente léxico utilizado, el lexema al que pertenece así como la línea y columna donde se encuentra con respecto al campo de texto (Área de programación)

Área de programación:

El área de programación, o sección de código, corresponde al cuarto espacio del entorno visual del compilador. Se encuentra al centro y abarca la mayor parte de la pantalla. Aquí, es donde el programador escribe el código del programa que realiza.

Está dividido en dos secciones, la parte izquierda donde se realiza el conteo de las líneas utilizadas y el resto del área de programación, donde el programador escribe el código.

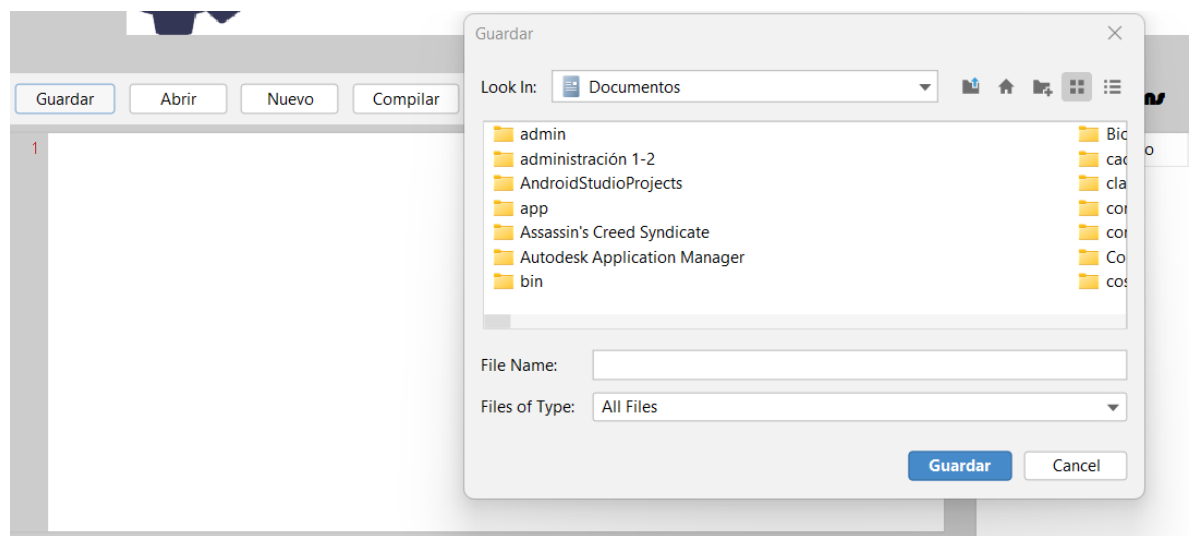
Area de errores y resultado de compilación:



Se encuentra en la parte inferior del compilador, y es del largo del área de programación. En esta sección, son mostrados los errores que pudiese presentar el código escrito en el área programable. Dichos errores describirán el tipo, la línea donde se encuentra y una descripción de por qué ha sido detectado como un error. De no haber errores, solo se mostrará el resultado de la compilación.

Vistas adicionales

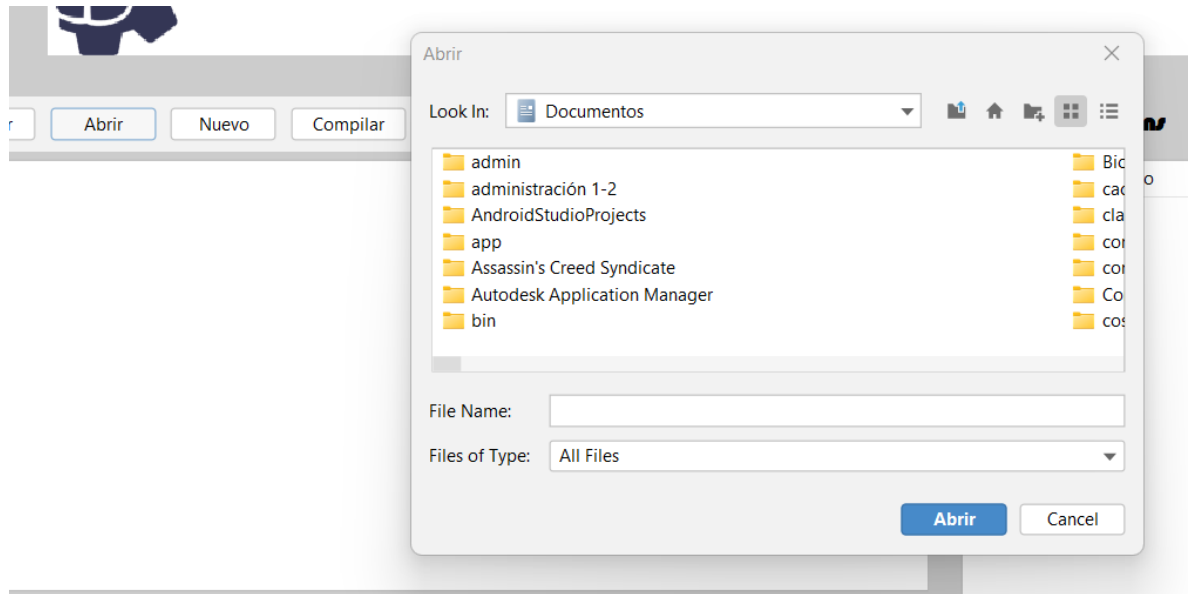
Guardar



La ventana emergente de guardar, nos permitirá elegir mediante el explorador de archivos, el nombre y ubicación del archivo en el que estamos trabajando.

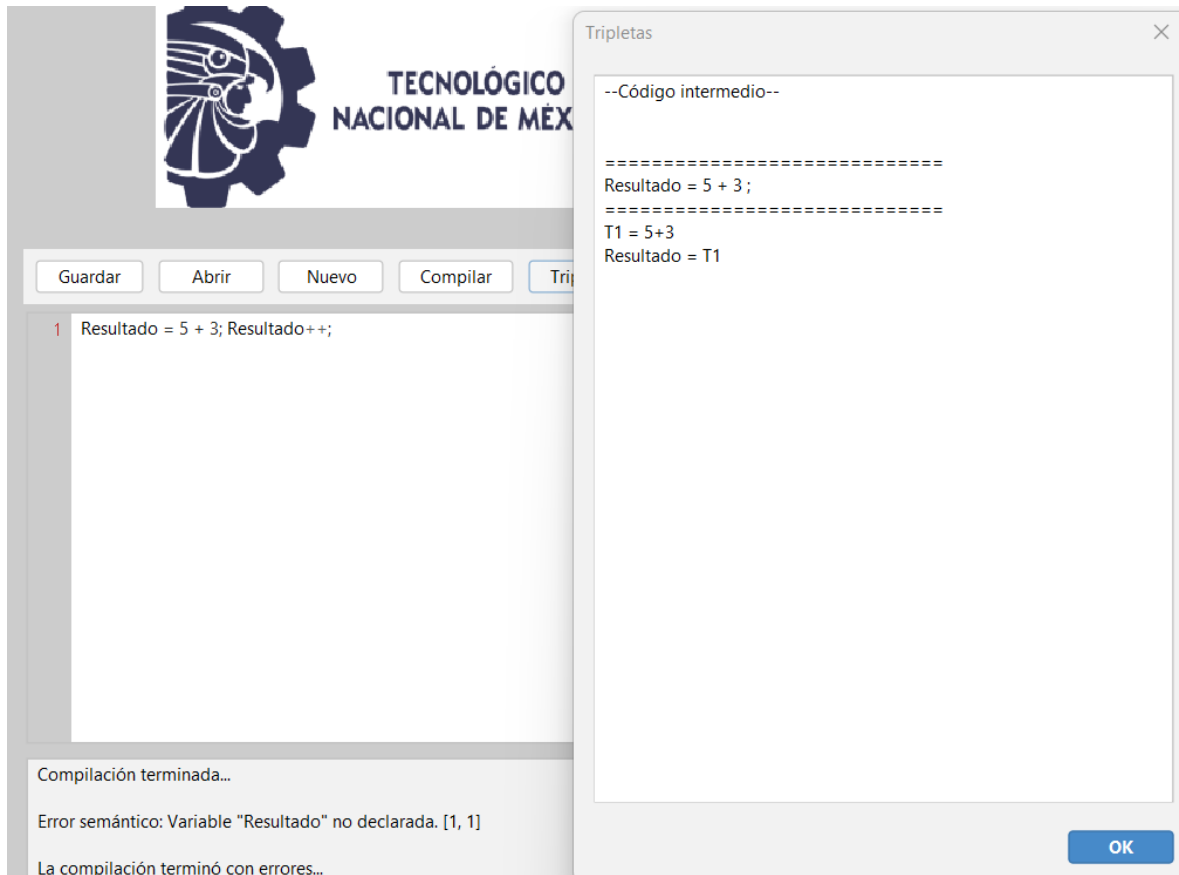
COMPILADOR CHAIRCODE

Abrir



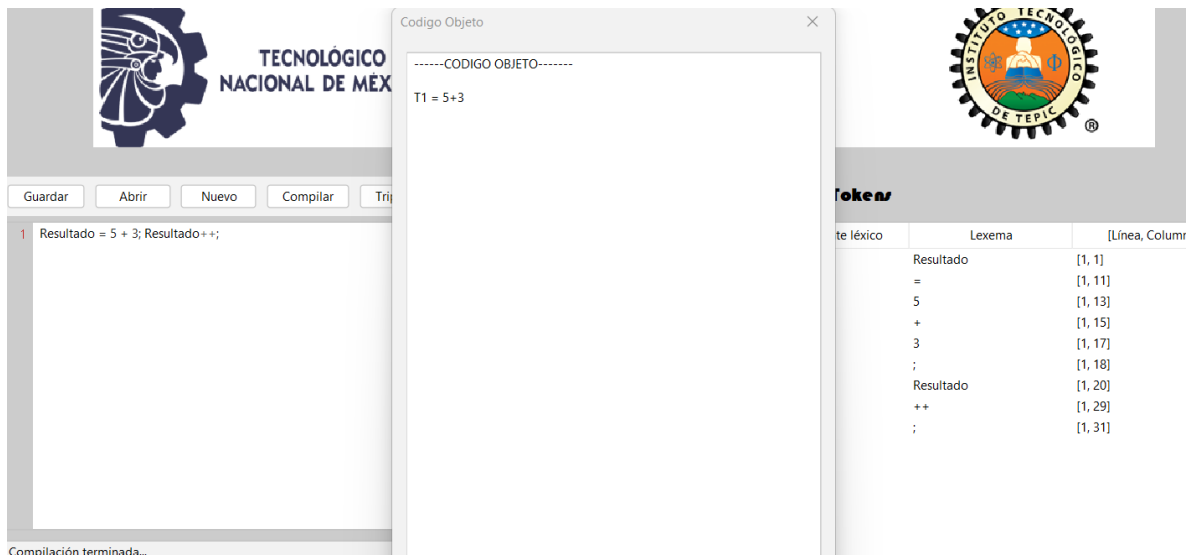
La ventana emergente del botón abrir nos permite seleccionar un archivo previamente creado el cual queramos abrir en el compilador.

Tripletas



La ventana emergente nos muestra el código intermedio de lo que se encuentre en el área de programación.

Código Objeto



The screenshot displays the CHAIRCODE compiler interface. On the left, a window titled 'Código Objeto' shows the source code: 'Resultado = 5 + 3; Resultado++;'. Below the code editor, a status bar indicates 'Compilación terminada...'. In the center, a modal window titled 'Codigo Objeto' displays the generated object code: 'T1 = 5+3'. On the right, a window titled 'Tokens' shows a table of tokens with columns for 'Token', 'Lexema', and '[Línea, Columna]'. The table lists the tokens for the source code: 'Resultado', '=', '5', '+', '3', ';', 'Resultado', '++', and ';', along with their corresponding line and column positions.

Token	Lexema	[Línea, Columna]
Resultado	Resultado	[1, 1]
=	=	[1, 11]
5	5	[1, 13]
+	+	[1, 15]
3	3	[1, 17]
;	;	[1, 18]
Resultado	Resultado	[1, 20]
++	++	[1, 29]
;	;	[1, 31]

La ventana emergente nos muestra el código objeto de lo que se encuentre en el área de programación.

Ensamblador

```
B dw 1,0
EDAD dw 1,0
H dw 1,0
x dw 1,0
.code
INICIO: MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX

;=====
FLOAT B = 1 + 1 * 3 ;
;=====
MOV AX,1
MOV BX,3
MUL BX
MOV BX,1
ADD BX,AX

;=====
INT EDAD = 34 * 8 + 5 * 0.125 ;
;=====
MOV AX,34
MOV BX,8
MUL BX
MOV BX,5
MOV CX,0.125
MUL CX
ADD AX,BX

;=====
```

OK

Ventana emergente con código convertido a ensamblador.

CAPÍTULO 4: CRUD

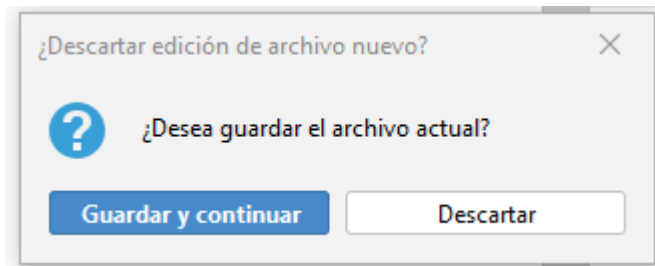
En el mundo de la informática, CRUD es el acrónimo, o siglas, de Create, Read, Update y Delete, que, en español, se traducen como Crear, Leer, Actualizar y Eliminar. Este acrónimo, generalmente, se refiere a las bases de datos.

En un compilador, cualquiera del que se hable, el CRUD se refiere a la creación de un programa, su lectura y ejecución, las modificaciones necesarias al encontrar errores o mejorar el código y la eliminación de dichos errores o código innecesarios.

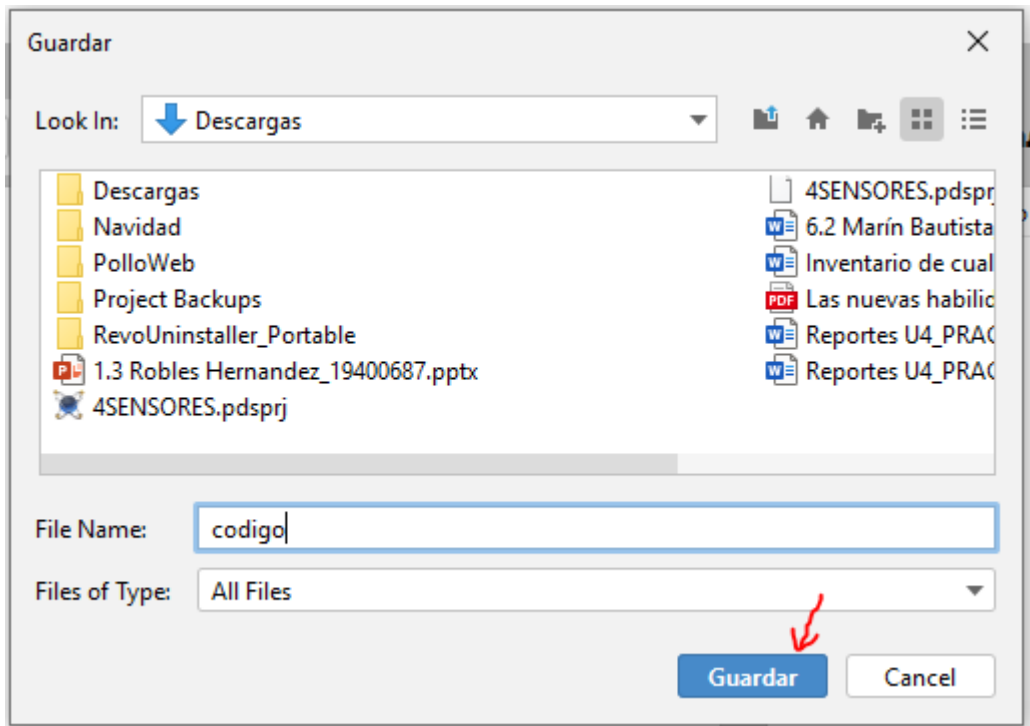
Crear un nuevo proyecto:

Crear un nuevo código es, en pocas palabras, generar un nuevo archivo, codificar (es decir, programar) y guardarlo para abrirlo posteriormente para esto es necesario ingresar el código y luego dar click en nuevo.

Enseguida nos pedirá que si queremos guardar el código y continuar, le daremos en guardar y continuar.

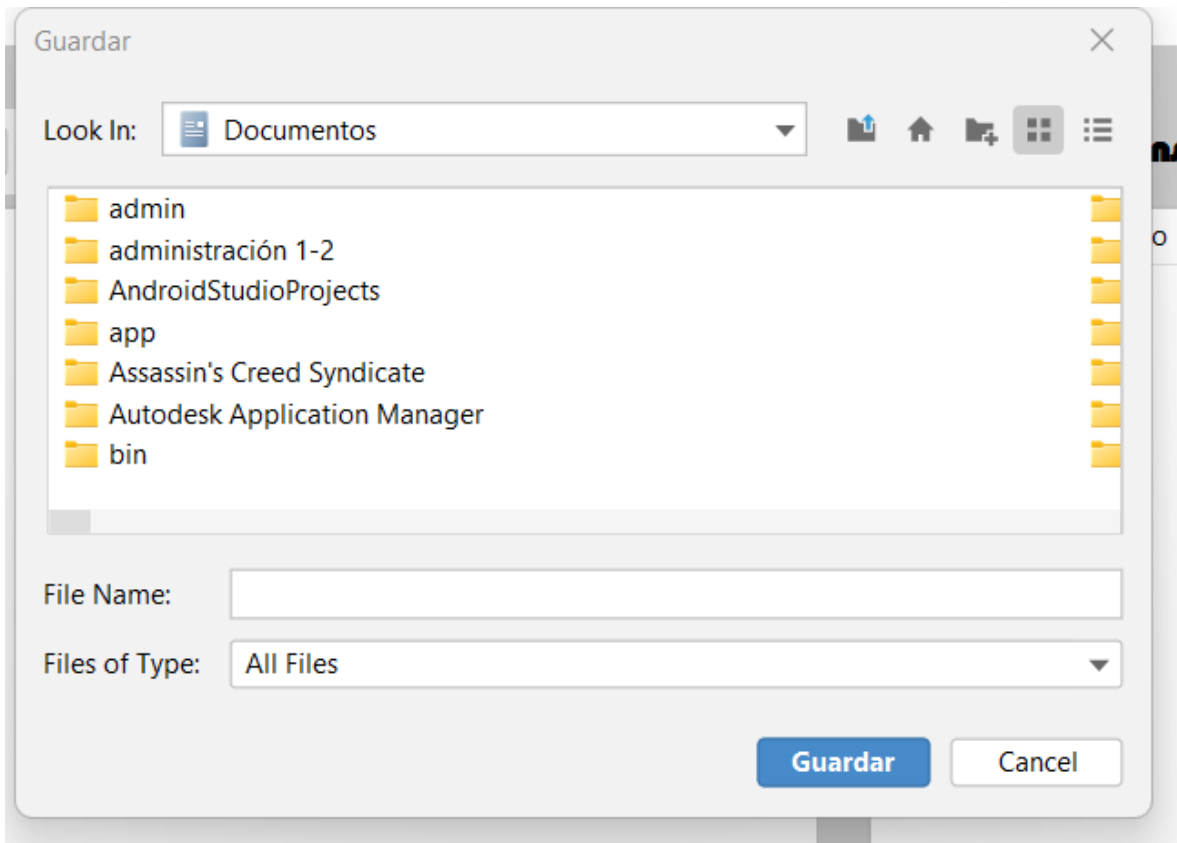


Después nos aparecerá una ventana donde asignaremos un carpeta donde se guarde y un nombre.



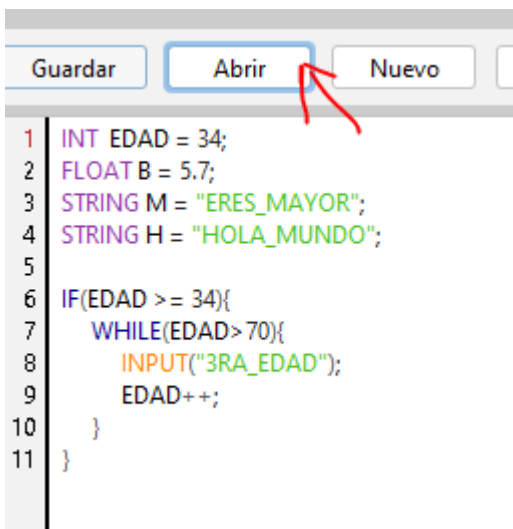
Guardar un proyecto:

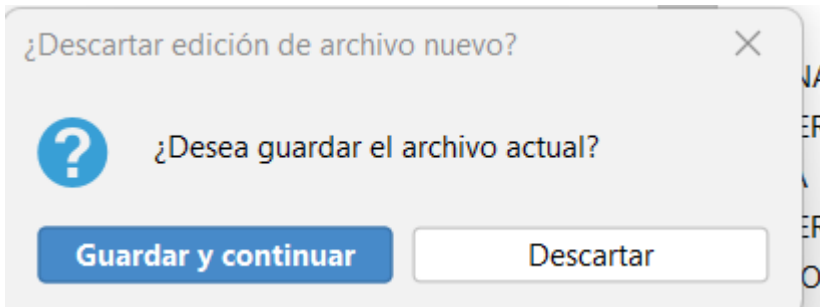
Para guardar un proyecto únicamente se debe presionar el botón "Guardar" que se encuentra en la barra de herramientas, posteriormente se abrirá la ventana emergente correspondiente donde se decidirá la ubicación y el nombre que queremos darle al archivo para guardarlo.



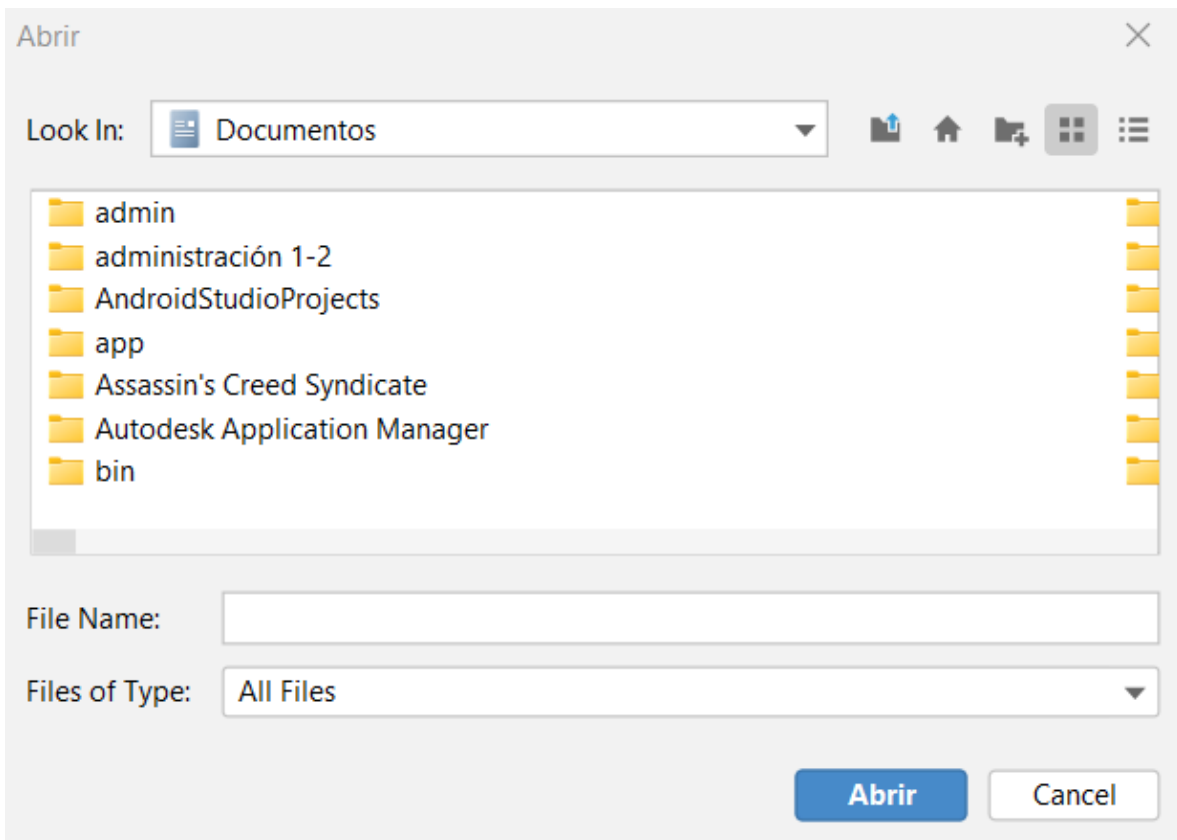
Abrir un proyecto existente:

Para abrir un proyecto ya existente se requiere presionar el botón Abrir que se encuentra en la barra de herramientas, en caso de estar trabajando en otro proyecto en ese momento se abrirá una ventana emergente dando la opción de guardar el progreso que se tenga hasta el momento.





Independientemente de la elección se abrirá otra ventana donde usted señalará el archivo que quiere abrir y una vez seleccionado se abrirá.



Ingresar Código:

Primeramente ingresamos el que queremos compilar.

```
1 INT EDAD = 34;
2 FLOAT B = 5.7;
3 STRING M = "ERES_MAYOR";
4 STRING H = "HOLA_MUNDO";
5
6 IF(EDAD >= 34){
7     WHILE(EDAD>70){
8         INPUT("3RA_EDAD");
9         EDAD++;
10    }
11 }
```

luego lo guardamos de la manera en la que se muestra anteriormente.

Compilación:

Después de escribir nuestro código, para realizar la compilación presionaremos el botón

Compilar

“compilar” en la barra de herramientas.

En caso de que el compilador detecte errores serán señalados en la zona de Errores y Resultados indicando la ubicación del error así como la causa de que sea señalado como tal.

Compilación terminada...

Ejemplo con error:

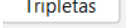
Compilación terminada...

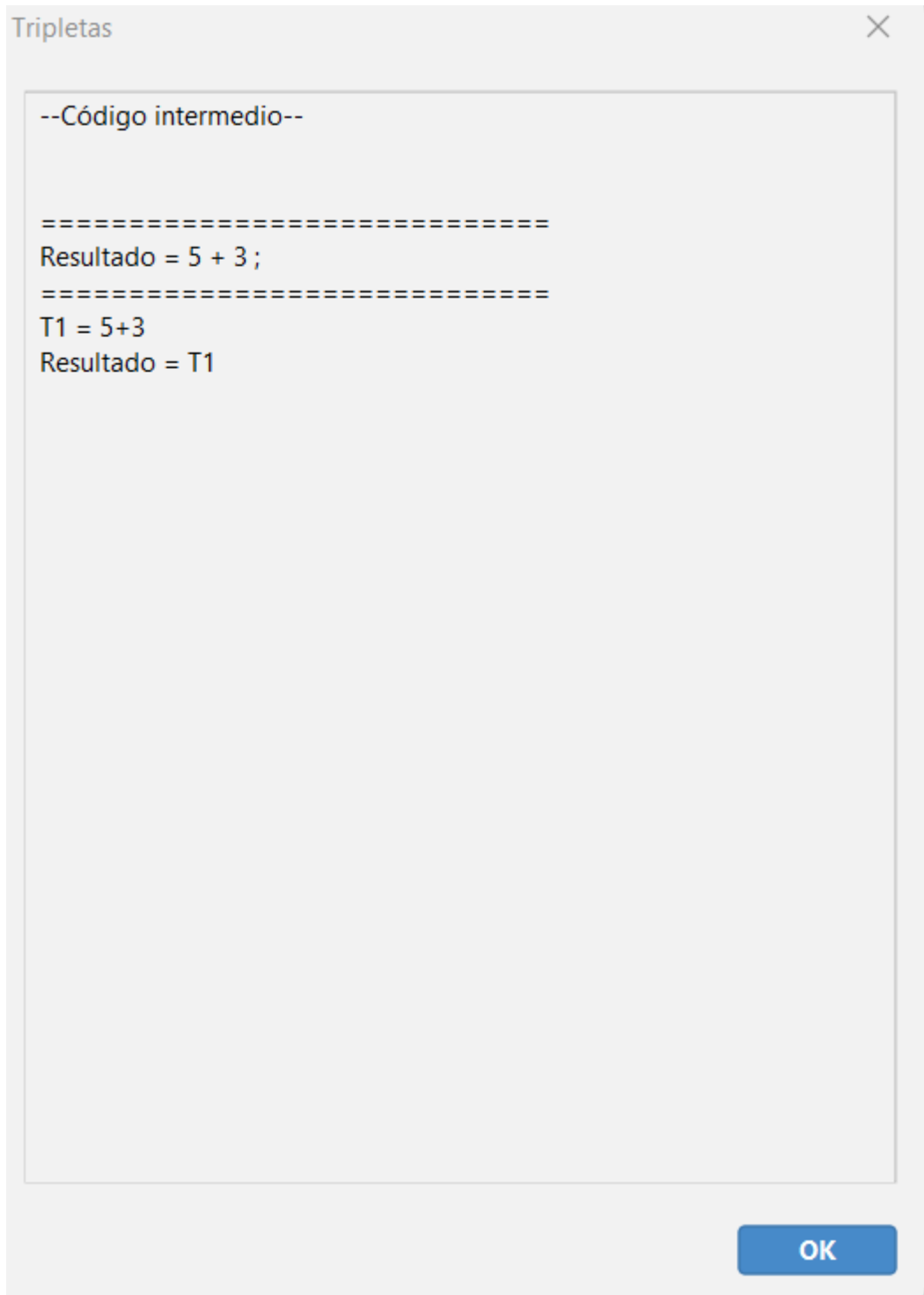
ERROR_SINTACTICO: PUNTOCOMA(;) NO AGREGADO EN LA DECLARACION [2, 1]

La compilación terminó con errores...

```
1 INT EDAD = 34;
2 FLOAT B = 5.7;
3 STRING M = "ERES_MAYOR";
```

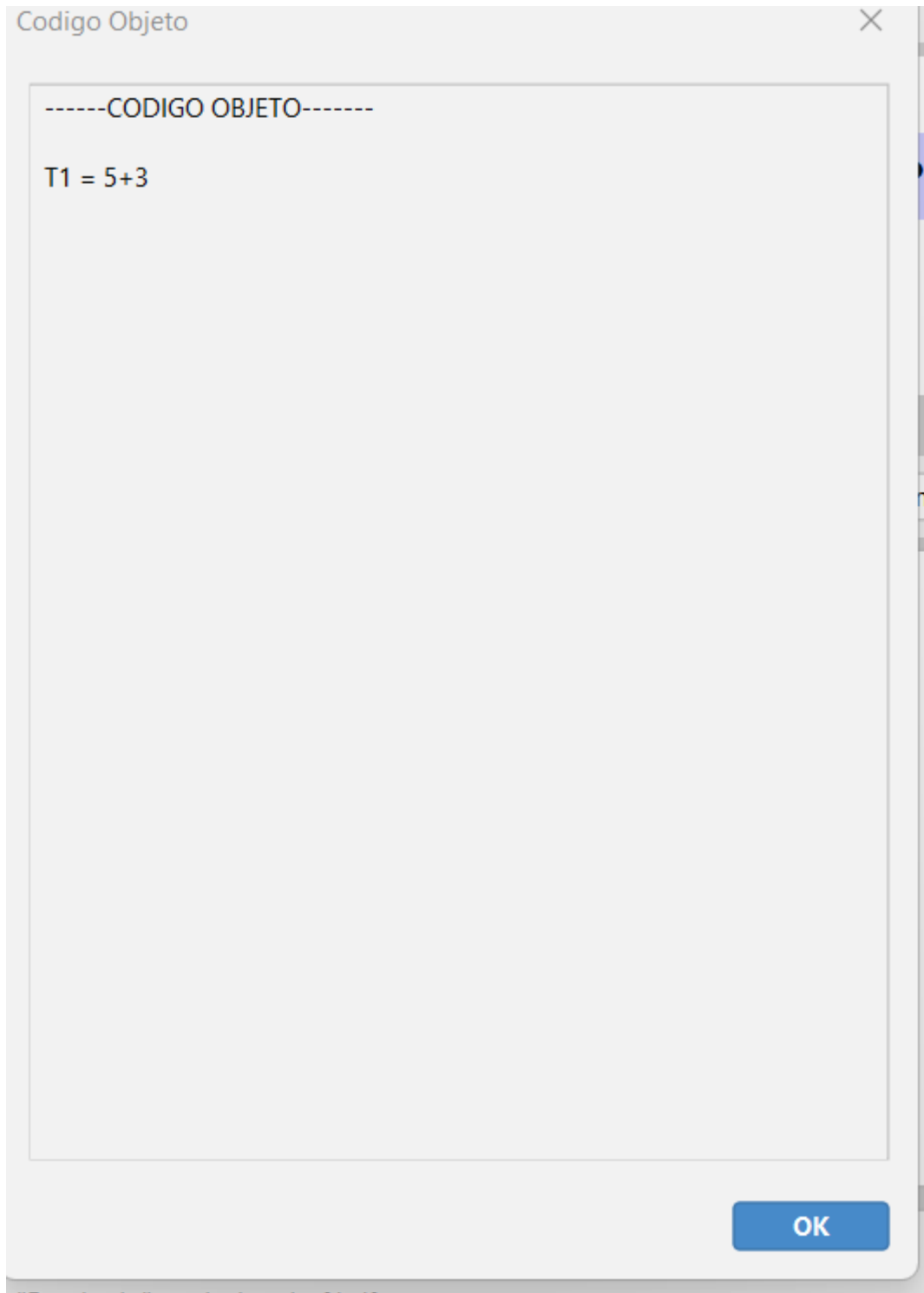

Generación de Código intermedio

Para generar el código intermedio únicamente se debe presionar el botón “tripletas” . Cuando nuestro código esté listo para ser generado, abrirá la ventana emergente correspondiente a la sección Tripletas del capítulo 3.



Generacion deCodigo Objeto

Para la generación de código objeto únicamente se requiere el código escrito correctamente y presionar el botón “CodObj” para generar el mismo, eso abrirá la ventana emergente en la sección de código objeto.



Optimización

Para generar la optimización del código presentado en el área de programación únicamente se requiere que el código se encuentre escrito correctamente y presionar el botón compilar en el

COMPILADOR CHAIRCODE

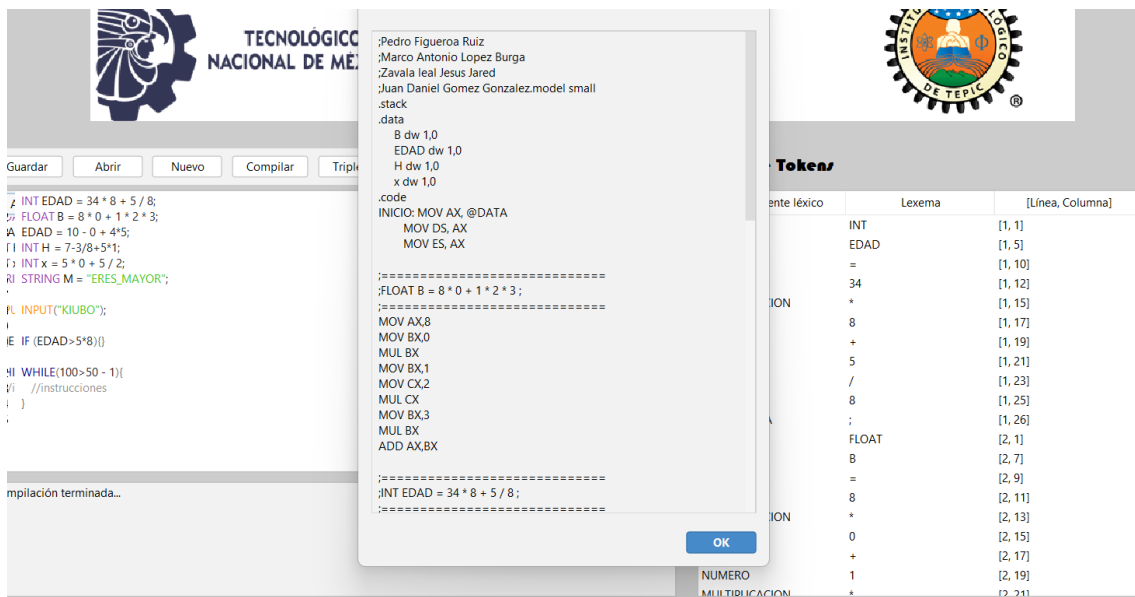
menú del programa para asegurarse, posteriormente aparecerá una ventana emergente dando la opción de optimizar el código analizado.



Al seleccionar la opción "Yes" el código será optimizado como lo considere mejor el compilador.

Generación de código ensamblador

Para generar el código en lenguaje ensamblador se requiere haber compilado previamente el código y posteriormente hacer click en el botón ensamblador que se encuentra en el menú del compilador.



Apéndices:

Apéndice A:

A.1: Palabras reservadas:

Palabra reservada	Descripción	Ejemplo
IMPORT	Cadena mayus y min I,M,P,O,R,T	IMPORT ó import

DEF	Cadena mayus y min D, E, F	DEF ó def
CLASS	Cadena mayus y min C, L, A, S, S	CLASS ó class
IF	Cadena mayus y min I, F	IF ó if
ELSE	Cadena mayus y min E, L, S, E	ELSE ó else
FOR	Cadena mayus y min F, O, R	FOR ó for
IN	Cadena mayus y min I, N	IN ó in
RANGE	Cadena mayus y min R, A, N, G, E	RANGE ó range
SELF	Cadena mayus y min S, E, L, F	SELF ó self
WHILE	Cadena mayus y min W, H, I, L, E	WHILE ó while
TRY	Cadena mayus y min T, R, Y	TRY ó try
EXCEPT	Cadena mayus y min E, X, C, E, P, T	EXCEPT ó except

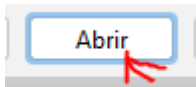
COMPILADOR CHAIRCODE

RETURN	Cadena mayus y min R, E, T, U, R, N	RETURN ó return
BREAK	Cadena mayus y min B, R, E, A, K	BREAK ó break
NEXT	Cadena mayus y min N, E, X, T	NEXT ó next
INPUT	Cadena mayus y min I, N, P, U, T	INPUT ó input
PRINT	Cadena mayus y min P, R, I, N, T	PRINT ó print
INT	Cadena mayus y min I, N, T	INT ó int
FLOAT	Cadena mayus y min F, L, O, A, T	FLOAT ó float
STRING	Cadena mayus y min S, T, R, I, N, G	STRING ó string
POWER	Cadena mayus y min P, O, W, E, R	POWER ó power
SQRT	Cadena mayus y min S, Q, R, T	SQRT ó sqrt
AND	Cadena mayus y min A, N, D	AND ó and
OR	Cadena mayus y min O, R	OR ó or
NOT	Cadena mayus y min N, O, T	NOT ó not

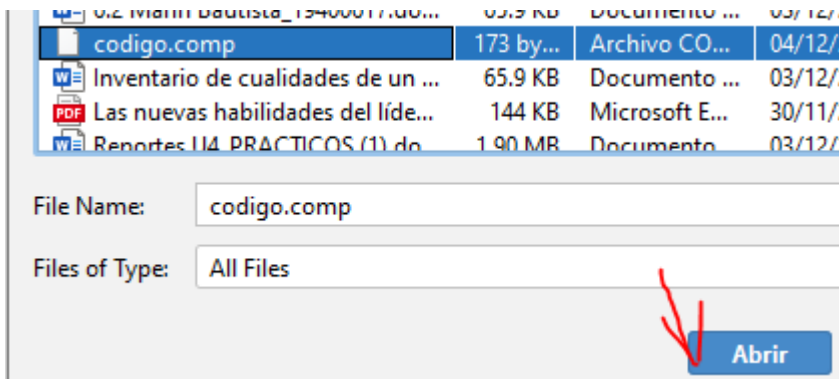
Apéndice B:

B.1: Ejemplo Funcional:

- Primero abriremos o crearemos un nuevo Archivo.



- Seleccionaremos el archivo que queremos abrir y lo abrimos.



- Se mostrará el código en el área de texto.

```

1  INT EDAD = 34;
2  FLOAT B = 5.7
3  STRING M = "ERES_MAYOR";
4  STRING H = "HOLA_MUNDO";
5
6  IF(EDAD >= 34){
7      WHILE(EDAD > 70){
8          INPUT("3RA_EDAD");
9          EDAD++;
10     }
11 }
12
13 WHILE(100 > 56){
14     FOR(i IN RANGE(1,400)){
15         B = 50*8;
16     }
17 }
    
```

- Ahora compilamos para detectar los tokens, errores léxicos, errores semánticos y errores sintácticos.
- En caso de aparecer algún error nos dirá cual es y donde se encuentra para que podamos solucionarlo.

COMPILADOR CHAIRCODE

Compilación terminada...

ERROR_SINTACTICO: PUNTOCOMA(;) NO AGREGADO EN LA DECLARACION [2, 1]
Error semántico: Variable "B" no declarada. [15, 11]

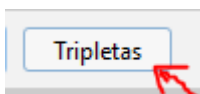
La compilación terminó con errores...

```
1 | INT EDAD = 34;  
2 | FLOAT B = 5.7  
3 | STRING M = "ERES_MAYOR";  
  |
```

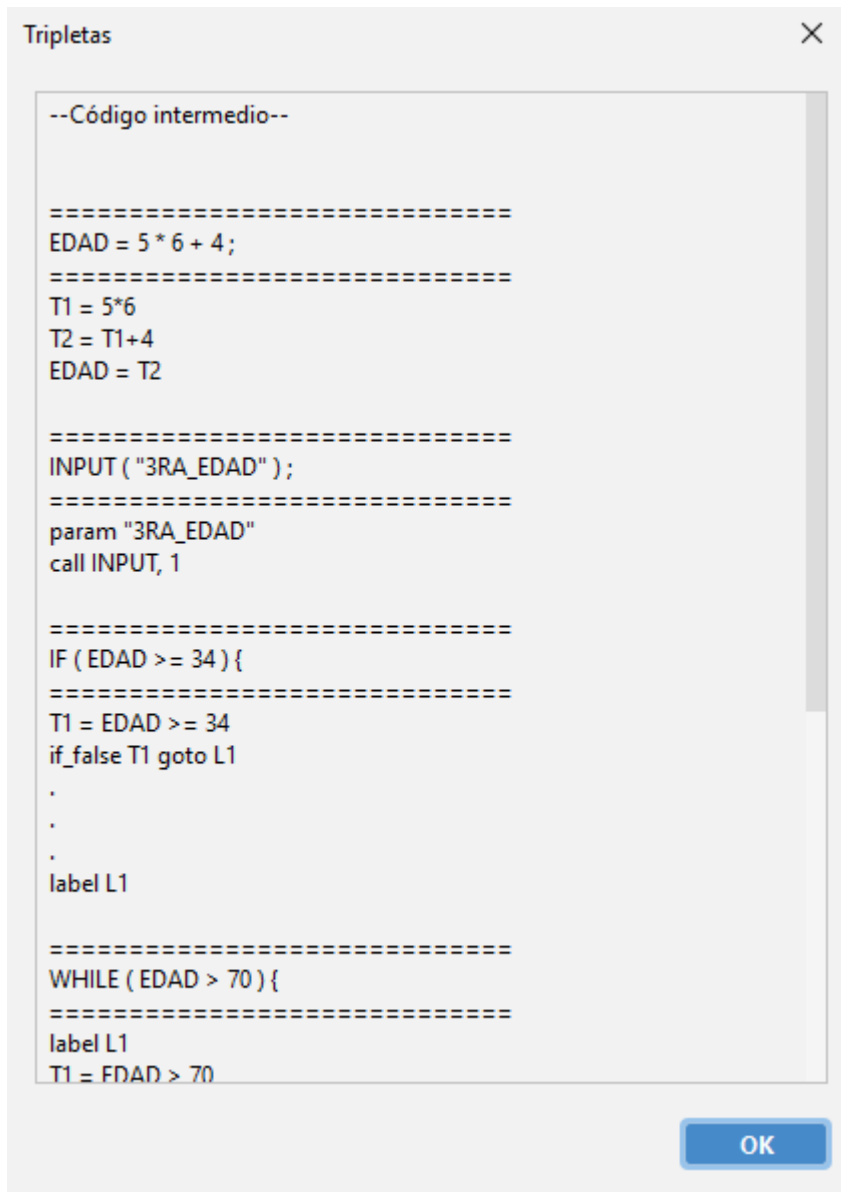
- La tabla de tokens nos mostrará todos los elementos del código.

Componente léxico	Lexema	[Línea, Columna]
INT	INT	[1, 1]
ID	EDAD	[1, 6]
ASIGNACION	=	[1, 11]
NUMERO	34	[1, 13]
PUNTOCOMA	;	[1, 15]
FLOAT	FLOAT	[2, 1]
ID	B	[2, 7]
ASIGNACION	=	[2, 9]
REAL	1.7	[2, 11]
PUNTOCOMA	;	[2, 14]
STRING	STRING	[3, 1]
ID	M	[3, 8]
ASIGNACION	=	[3, 10]
CADENA	"ERES_MAYOR"	[3, 12]
PUNTOCOMA	;	[3, 24]
STRING	STRING	[4, 1]
ID	H	[4, 8]
ASIGNACION	=	[4, 10]
CADENA	"HOLA_MUNDO"	[4, 12]

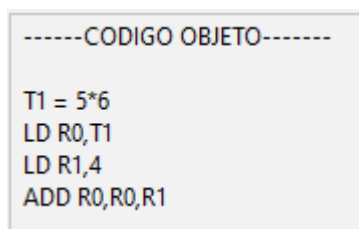
- Ya compilado sin errores podemos generar el código intermedio.



- En este podremos ver las líneas de código y su equivalente en 3 direcciones.



- También podemos generar el código objeto y que lo muestre en el emulador EMU 8086.




COMPILADOR CHAIRCODE

```
.model small
.stack
.data
    EDAD DB 0,0
.code
INICIO: MOV AX, @DATA
        MOV DS, AX
        MOV ES, AX






-----CODIGO OBJETO-----

T1 = 5*6
MOV AX,T1
MOV BX,4
ADD AX,BX

FIN: MOV AX,4C00H
     INT 21H
     END
```

 edit: C:\Users\CUATE\Desktop\TEC\Semestre 7\Lenguajes y

file edit bookmarks assembler emulator math ascii

 new  open  examples  save  compile

```
01 |.model small
02 |.stack
03 |.data
04 |.code
05 |INICIO: MOV AX, @DATA
06 |        MOV DS, AX
07 |        MOV ES, AX
08 |
09 |-----CODIGO OBJETO-----
10 |
11 |FIN: MOV AX,4C00H
12 |    INT 21H
13 |    END
14 |
```