

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1
2do Semestre 2021
Ing. Mario Bautista
Aux. Emely García
Aux. José Morán

MANUAL DE USUARIO SYSCOMPILER

Marvin Eduardo Catalán Véliz
201905554
Guatemala, 6 de nov. de 21

Índice

1. Descripción SYSCOMPILER	4
2. Entorno de trabajo	4
2.1 Editor	4
2.2 Funcionalidades	4
2.3 Características	6
2.4 Herramientas	6
2.5 Reportes	7
2.6 Área de consola	7
3. Descripción del lenguaje	7
3.1 Case insensitive	7
3.2 Comentarios	7
3.2.1 Comentarios de una línea	7
3.2.2 Comentarios multilínea	7
3.3 Tipos de datos	8
3.4 Secuencia de escape	8
3.5 Operadores aritméticos:	9
3.5.1 Suma	9
3.5.2 Resta	9
3.5.3 Multiplicación	9
3.5.4 División	10
3.5.5 Potencia	10
3.5.6 Módulo	10
3.5.7 Negación Unaria	10
3.6 Operadores racionales	11
3.7 Operador Ternario	11
3.8 Operadores lógicos	11
3.9 Signos de agrupación	12
3.10 Precedencia de operaciones	12
3.11 Caracteres de finalización y encapsulamiento de sentencias ...	12
3.12 Declaración y asignación de variables	12
3.13 Casteos	13
3.14 Incremento y decremento	13
3.15 Sentencia de control	14
3.15.1 If	14

3.15.2 Switch case	15
3.16 Sentencias Cíclicas	16
3.16.1 While	16
3.16.2 For	16
3.16.3 Do-While	17
3.17 Sentencias de transferencia	17
3.17.1 Break	17
3.17.2 Continue	18
3.17.3 Return	18
3.18 Métodos	18
3.19 Llamadas	19
3.20 Funciones WriteLine	19
3.21 Función ToLower	19
3.22 Función ToUpper	20
3.23 Funciones Nativas	20
Length	20
Truncate	20
Round	20
Typeof	20
To String	20
To CharArray	20
3.24 Start with	21
4. Reportes	21
4.1 Tabla de Símbolos	22
4.2 Tabla de Errores	22
4.3 AST	22
4.4 Salidas en consola	23

1. Descripción SYSCOMPILER

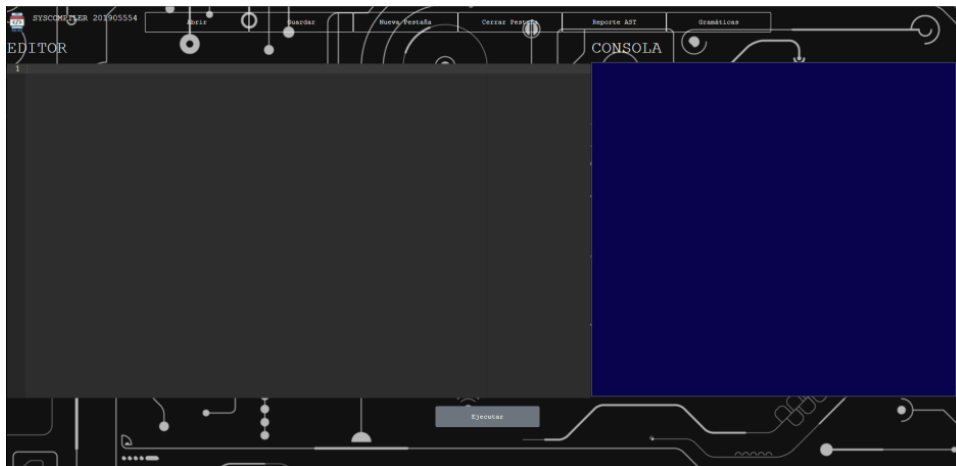
Como estudiante de Organización de Lenguajes y Compiladores 1, se ha solicitado un nuevo proyecto dado los altos conocimientos en temas de análisis léxico, sintáctico y semántico en el que se debe de crear un lenguaje de programación para que los estudiantes del curso de Introducción a la Programación y Computación 1, aprendan a programar y tener conocimiento de todas las generalidades de un lenguaje de programación. El cual servirá para las primeras prácticas de laboratorio del curso antes mencionado.

2. Entorno de trabajo

2.1 Editor

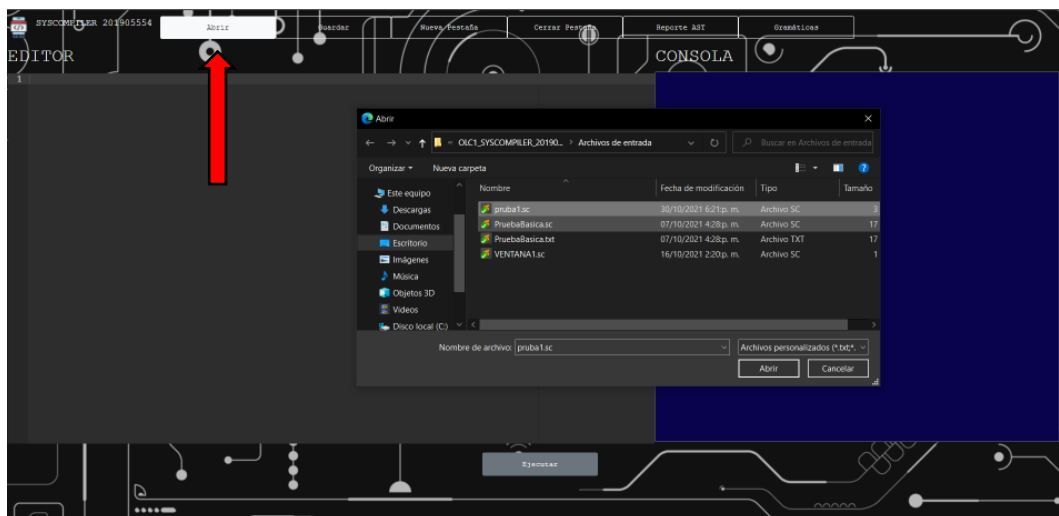
La finalidad es proporcionar ciertas funcionalidades, características y herramientas que serán de utilidad al usuario. En este se podrán abrir diferentes archivos al mismo tiempo y deberá mostrar la línea actual.

Se adjunta la interfaz gráfica de esta web app:

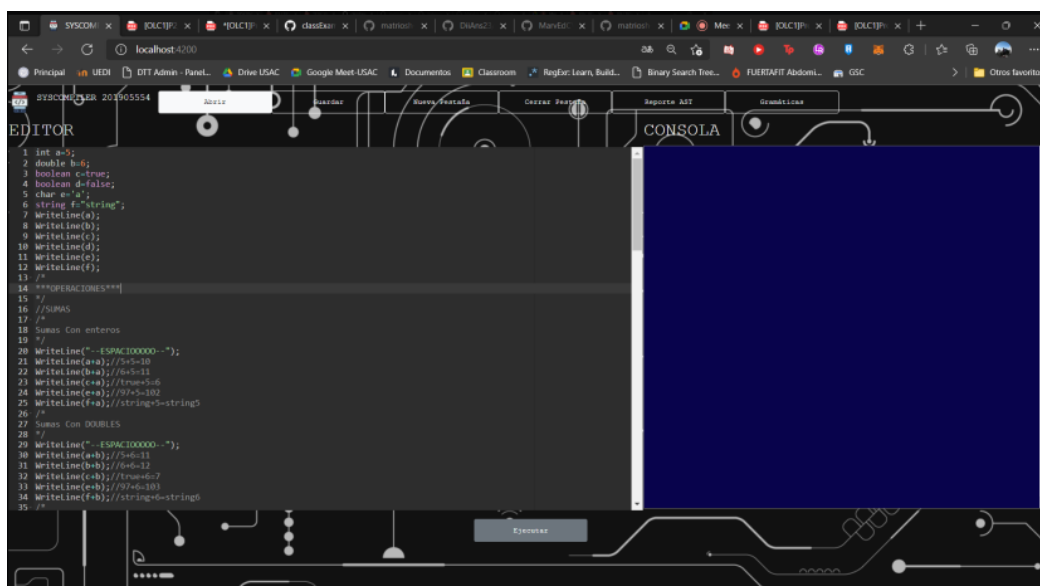


2.2 Funcionalidades

- **Crear archivos:** El editor será capaz de crear archivos en blanco.
- **Abrir archivo:** En el editor se debe de dar clic en "abrir"



Luego puede seleccionar el archivo que desea cargar, y finalmente el archivo será cargado con éxito.



- **Guardar el archivo:** Se descarga automáticamente el archivo que esta en el editor de texto.



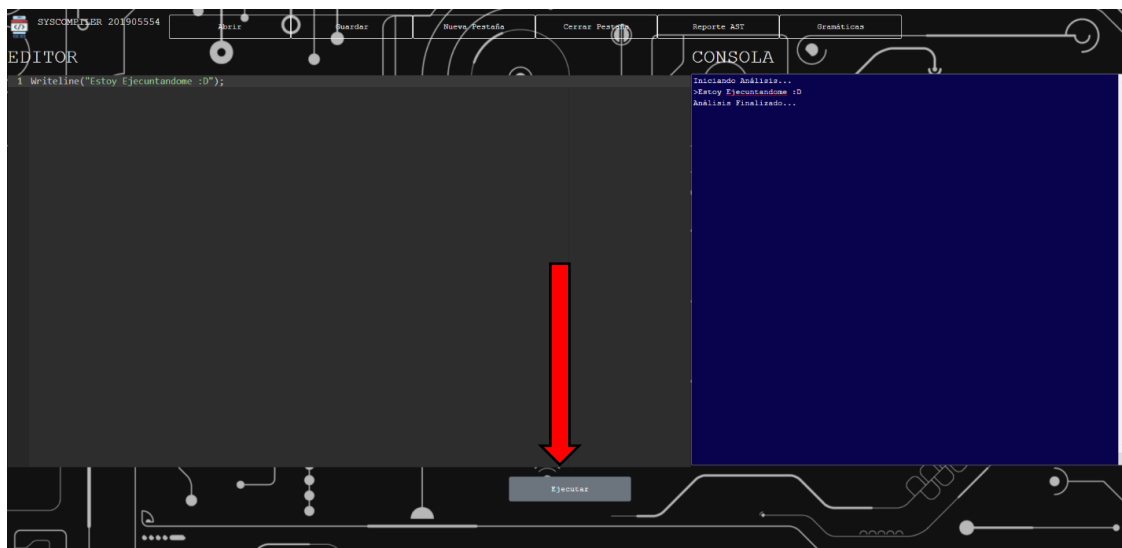
- **Eliminar pestaña:** permitirá cerrar la pestaña actual.

2.3 Características

- **Múltiples Pestañas:** se podrán crear nuevas pestañas con la finalidad de ver y abrir los archivos de prueba en la aplicación.

2.4 Herramientas

- **Ejecutar:** realizará los análisis léxico, sintáctico y semántico, además de ejecutar todas las sentencias.



2.5 Reportes

- **Reporte de Errores:** Se mostrarán todos los errores encontrados al realizar el análisis léxico, sintáctico y semántico.
- **Generar Árbol AST (Árbol de Análisis Sintáctico):** se genera una imagen del árbol de análisis sintáctico al realizar los análisis.
- **Reporte de Tabla de Símbolos:** Se mostrarán todas las variables, métodos y funciones que han sido declarados dentro del flujo del programa.

2.6 Área de consola

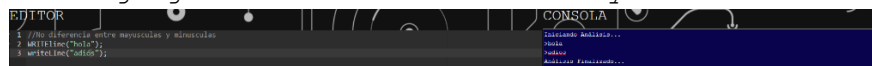
En esta área se mostrarán los resultados, mensajes y todo lo que sea indicado dentro del lenguaje.



3. Descripción del lenguaje

3.1 Case insensitive

El lenguaje no diferencia entre mayúsculas o minúsculas.



3.2 Comentarios

El lenguaje soporta dos tipos de comentarios que son los siguientes:

3.2.1 Comentarios de una línea

Deben comenzar con `//` y terminar con un salto de línea.

3.2.2 Comentarios multilínea

Deben comenzar con `/*` y terminar con `*/`.

Se adjunta ejemplos de los tipos de comentarios mencionados:

```

EDITOR
1 //Esto es un comentario unilínea
2
3 /*
4 Esto es un
5 comentario
6 multilinea
7 */

```

3.3 Tipos de datos

Los tipos de dato que soportará el lenguaje en concepto de un tipo de variable se definen a continuación:

TIPO	DEFINICION	DESCRIPCION	EJEMPLO	OBSERVACIONES	DEFAULT
Entero	Int	Este tipo de datos aceptará solamente números enteros.	1, 50, 100, 25552, etc.	Del -2147483648 al 2147483647	0
Doble	Double	Admite valores numéricos con decimales.	1.2, 50.23, 00.34, etc.	Se manejará cualquier cantidad de decimales	0.0
Booleano	Boolean	Admite valores que indican verdadero o falso.	True, false	Si se asigna un valor booleano a un entero se tomará como 1 o 0 respectivamente.	True
Caracter	Char	Tipo de dato que únicamente aceptará un único carácter, y estará delimitado por comillas simples. ''	'a', 'b', 'c', 'E', 'Z', '1', '2', '^', '%', ' ', '=', '!', '&', '/', '\\', '\n', etc.	En el caso de querer escribir comilla simple escribir se escribirá \ y después comilla simple \, si se quiere escribir \ se escribirá dos veces \\, existirá también \n, \t, \r, \".	'\u0000' (carácter 0)
Cadena	String	Es un grupo o conjunto de caracteres que pueden tener cualquier carácter, y este se encontrará delimitado por comillas dobles. ""	"cadena1", "-- ** cadena 1"	Se permitirá cualquier carácter entre las comillas dobles, incluyendo las secuencias de escape: \" comilla doble \\ barra invertida \\n salto de línea \\r retorno de carro \\t tabulación	"" (string vacío)

3.4 Secuencia de escape

Las secuencias de escape se utilizan para definir ciertos caracteres especiales dentro de cadenas de texto. Las secuencias de escape disponibles son las siguientes:

SECUENCIA	DESCRIPCION	EJEMPLO
\\n	Salto de línea	"Hola\\nMundo"
\\	Barra invertida	"C:\\miCarpeta\\Personal"
\"	Comilla doble	"\\\"Esto es una cadena\\\""
\\t	Tabulación	"\\tEsto es una tabulación"
'	Comilla Simple	"\\'Estas son comillas simples\\'"

Se adjunta ejemplo de las secuencias de escape:

EDITOR

```

1 //Ejemplo de secuencias de escape
2 WriteLine("Aquí un salto de línea\n Escribe una dirección C:\\miCarpeta\\Persona \\n ");
3 WriteLine("\\t Es es una tabulación");
4 WriteLine("\\ Estas son comillas simples");

```

CONSOLA

```

Iniciando Análisis...
>Pase un salto de línea
Escriba una dirección "C:\\miCarpeta\\Persona
>
Es es una tabulación
>"Estas son comillas simples"
Análisis Finalizado...

```

3.5 Operadores aritméticos:

3.5.1 Suma

Es la operación aritmética que consiste en realizar la suma entre dos o más valores. El símbolo a utilizar es el signo más +.

Especificaciones de la operación suma

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

+	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble	Entero	Entero	Cadena
Doble	Doble	Doble	Doble	Double	Cadena
Boolean	Entero	Doble			Cadena
Caracter	Entero	Doble		Cadena	Cadena
Cadena	Cadena	Cadena	Cadena	Cadena	Cadena

A continuación, se muestra un ejemplo de error de suma utilizando boolean + boolean:

EDITOR

```

1 WriteLine(true+true);

```

CONSOLA

```

Iniciando Análisis...
Error:Semántico
Mensaje:El op_BooleanOperati.2 COM 2
Línea:1
Columna:19
Análisis Finalizado...

```

3.5.2 Resta

Consiste en realizar la resta entre dos o más valores. El símbolo por utilizar es el signo menos -.

Especificaciones de la operación resta

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

-	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble	Entero	Entero	
Doble	Doble	Doble	Doble	Doble	
Boolean	Entero	Doble			
Caracter	Entero	Doble			
Cadena					

3.5.3 Multiplicación

Consiste en sumar un número (multiplicando) tantas veces como indica otro número (multiplicador). El signo para representar la operación es el asterisco*.

Especificaciones de la operación multiplicación

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

*	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble		Entero	
Doble	Doble	Doble		Doble	
Boolean					
Caracter	Entero	Doble			
Cadena					

3.5.4 División

Consiste en separar un todo en varias partes, al todo se le conoce como dividendo, al total de partes se le llama divisor y el resultado recibe el nombre de cociente. El operador de la división es la diagonal /

Especificaciones de la operación división

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

/	Entero	Doble	Boolean	Caracter	Cadena
Entero	Doble	Doble		Doble	
Doble	Doble	Doble		Doble	
Boolean					
Caracter	Doble	Doble			
Cadena					

3.5.5 Potencia

Es una operación aritmética de la forma a^b donde **a** es el valor de la base y **b** es el valor del exponente que nos indicará cuantas veces queremos multiplicar el mismo número. Para realizar la operación se utilizará el signo ^

Especificaciones de la operación potencia

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

^	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble			
Doble	Doble	Doble			
Boolean					
Caracter					
Cadena					

3.5.6 Módulo

Es una operación aritmética que obtiene el resto de la división de un numero entre otro. El signo a utilizar es el porcentaje %

Especificaciones de la operación potencia

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

%	Entero	Doble	Boolean	Caracter	Cadena
Entero	Doble	Doble			
Doble	Doble	Doble			
Boolean					
Caracter					
Cadena					

3.5.7 Negación Unaria

Es una operación que niega el valor de un número, es decir que devuelve el contrario del valor original

Especificaciones de la operación negación

A continuación, se especifica en una tabla los resultados que se deberán obtener con esta operación.

-num	Resultado
Entero	Entero
Doble	Doble
Boolean	
Caracter	
Cadena	

3.6 Operadores racionales

Son los símbolos que tienen como finalidad comparar expresiones, dando como resultado valores booleanos. A continuación, se definen los símbolos que serán aceptados dentro del lenguaje:

OPERADOR	DESCRIPCIÓN	EJEMPLO
==	Igualación: Compara ambos valores y verifica si son iguales. - Iguales= True - No iguales= False	1 == 1 "hola" == "hola" 25.5933 == 90.8883 25.5 == 20
!=	Diferenciación: Compara ambos lados y verifica si son distintos. - Iguales= False - No iguales= True	1 != 2, var1 != var2 25.5 != 20 50 != 'F' "hola" != "hola"
<	Menor que: Compara ambos lados y verifica si el derecho es mayor que el izquierdo. - Derecho mayor= True - Izquierdo mayor= False	(5/(5+5))<(8*8) 25.5 < 20 25.5 < 20 50 < 'F'
<=	Menor o igual que: Compara ambos lados y verifica si el derecho es mayor o igual que el izquierdo. - Derecho mayor o igual= True - Izquierdo mayor= False	55+66<=44 25.5 <= 20 25.5 <= 20 50 <= 'F'
>	Mayor que: Compara ambos lados y verifica si el izquierdo es mayor que el derecho. - Derecho mayor= False - Izquierdo mayor= True	(5+5.5)>8.98 25.5 > 20 25.5 > 20 50 > 'F'
>=	Mayor o igual que: Compara ambos lados y verifica si el izquierdo es mayor o igual que el derecho. - Derecho menor o igual= True - Izquierdo menor= False	5-6>=4+6 25.5 >= 20 25.5 >= 20 50 >= 'F'

3.7 Operador Ternario

Este operador hace uso de 3 operandos para simplificar la instrucción 'if' por lo que a menudo este operador se le considera como un atajo para la instrucción 'if'. El primer operando corresponde a la condición que debe de cumplir una expresión para que el operador retorne como valor el resultado de la expresión del segundo operando, en caso de no cumplir con la expresión el operador debe de retornar el valor de la expresión del tercer operando.

```

EDITOR
1 //Ejemplo del uso del operador ternario
2 int edad = 18;
3 boolean banderaedad = false;
4 banderaedad = edad > 17 ? true : false;
5

```

3.8 Operadores lógicos

Son los símbolos que tienen como finalidad comparar expresiones a nivel lógico (verdadero o falso). A

continuación, se definen los símbolos que serán aceptados dentro del lenguaje:

OPERADOR	DESCRIPCIÓN	EJEMPLO	OBSERVACIONES
	OR: Compara expresiones lógicas y si al menos una es verdadera entonces devuelve verdadero en otro caso retorna falso	(55.5) bandera==true Devuelve true	bandera es true
&&	AND: Compara expresiones lógicas y si son ambas verdaderas entonces devuelve verdadero en otro caso retorna falso	(flag1) && ("hola" == "hola") Devuelve true	flag1 es true
!	NOT: Devuelve el valor inverso de una expresión lógica si esta es verdadera entonces devolverá falso, de lo contrario retorna verdadero.	!var1 Devuelve falso	var1 es true

3.9 Signos de agrupación

Los signos de agrupación serán utilizados para agrupar operaciones aritméticas, lógicas o relacionales. Los símbolos de agrupación están dados por ().

3.10 Precedencia de operaciones

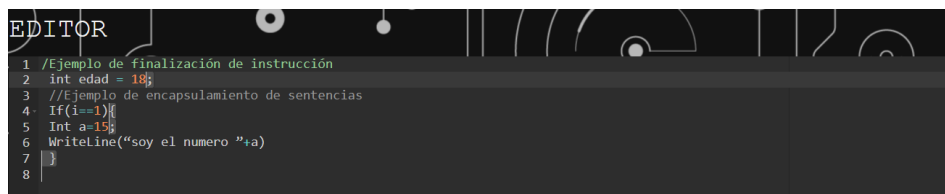
Indica la importancia en que una operación debe realizarse por encima del resto. A continuación, se define la misma.

NIVEL	OPERADOR	ASOCIATIVIDAD
0	-	Derecha
1	^	No asociativa
2	/, *	Izquierda
3	+, -	Izquierda
4	==, !=, <, <=, >, >=	Izquierda
5	!	Derecha
6	&&	Izquierda
7		Izquierda

3.11 Caracteres de finalización y encapsulamiento de sentencias

El lenguaje se verá restringido por dos reglas que ayudan a finalizar una instrucción y encapsular sentencias:

- **Finalización de instrucciones:** para finalizar una instrucción se utilizará el signo ;
- **Encapsular sentencias:** para encapsular sentencias dadas por los ciclos, métodos, funciones, etc., se utilizará los signos { }



```

EDITOR
1 //Ejemplo de finalización de instrucción
2 int edad = 18;
3 //Ejemplo de encapsulamiento de sentencias
4 if(i==1){
5     int a=15;
6     WriteLine("soy el numero "+a)
7 }
8
  
```

3.12 Declaración y asignación de variables

Una variable deberá de ser declarada antes de poder ser utilizada. Todas las variables tendrán un tipo de dato y un

nombre de identificador. Las variables pueden ser declaradas global y localmente.

También se tendrá la opción de poder crear múltiples variables al mismo tiempo, al crear múltiples variables al mismo tiempo se tendrá la opción de crear todas las variables con un mismo valor.

```
EDITOR
1 //Ejemplos
2 int numero;
3 int var1, var2, var3;
4 string cadena = "hola";
5 char var_1 = 'a';
6 boolean verdadero;
7 boolean flag1, flag2, flag3 = true;
8 char ch1, ch2, ch3 = 'R';
9
```

3.13 Casteos

Son una forma de indicar al lenguaje que convierta un tipo de dato en otro, por lo que, si queremos cambiar un valor a otro tipo, es la forma adecuada de hacerlo. Para hacer esto, se colocará la palabra reservada del tipo de dato destino entre paréntesis seguido de una expresión. **Ejemplo:** `'(<TIPO> ')<EXPRESION>`

El lenguaje aceptará los siguientes casteos:

- Int a double
- Double a Int
- Int a String
- Int a Char
- Double a String
- Char a int
- Char a double

```
EDITOR
1 //EJEMPLOS CASTEOS
2 int edad = (int) 18.6; //toma el valor entero de 18
3 char letra = (char) 70; //tomar el valor 'F' ya que el 70 en ascii es F
4 double numero = (double) 16; //toma el valor 16.0
```

3.14 Incremento y decremento

Ayudan a realizar la suma o resta continua de un valor de uno en uno, es decir si incrementamos una variable, se incrementará de uno en uno, mientras que, si realizamos un decremento, hará la operación contraria.

```

EDITOR
1 //Ejemplos INCREMENTO DECREMENTO
2 int edad = 18;
3 edad++; //tiene el valor de 19
4 edad--; //tiene el valor 18
5
6 int anio=2020;
7 anio = 1 + anio++; //obtiene el valor de 2022
8 anio = anio--; //obtiene el valor de 2021

```

3.15 Sentencia de control

Estas sentencias modifican el flujo del programa introduciendo condicionales. Las sentencias de control para el programa son IF y SWITCH.

3.15.1 If

Ejecuta las instrucciones sólo si se cumple una condición. Si la condición es falsa, se omiten las sentencias dentro de la sentencia

3.15.1.1 If

```

EDITOR
1 //EJEMPLO IF
2 if (x < 50)
3 {
4     WriteLine("Menor que 50");
5     //Más sentencias
6 }

```

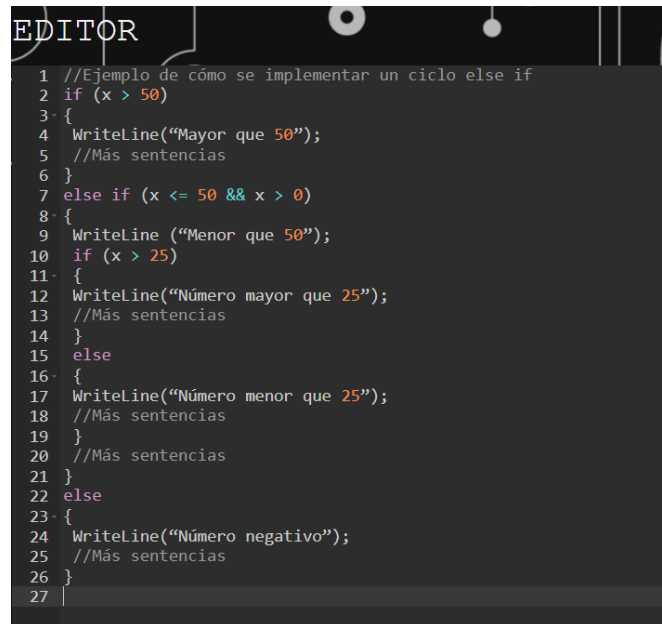
3.15.1.2 If else

```

EDITOR
1 //Ejemplo de cómo se implementar un ciclo if-else
2 if (x < 50)
3 {
4     WriteLine("Menor que 50");
5     //Más sentencias
6 }
7 else
8 {
9     WriteLine("Mayor que 100");
10    //Más sentencias
11 }

```

3.15.1.3 Else

A screenshot of a code editor window titled "EDITOR". The editor contains a C# code snippet demonstrating an else if statement. The code is as follows:

```
1 //Ejemplo de cómo se implementar un ciclo else if
2 if (x > 50)
3 {
4     WriteLine("Mayor que 50");
5     //Más sentencias
6 }
7 else if (x <= 50 && x > 0)
8 {
9     WriteLine("Menor que 50");
10    if (x > 25)
11    {
12        WriteLine("Número mayor que 25");
13        //Más sentencias
14    }
15    else
16    {
17        WriteLine("Número menor que 25");
18        //Más sentencias
19    }
20    //Más sentencias
21 }
22 else
23 {
24     WriteLine("Número negativo");
25     //Más sentencias
26 }
27
```

3.15.2 Switch case

Es una estructura utilizada para agilizar la toma de decisiones múltiples, trabaja de la misma manera que lo harían sucesivos if.

3.15.2.1 Switch

Estructura principal del switch, donde se indica la expresión a evaluar.

3.15.2.2 Case

Estructura que contiene las diversas opciones a evaluar con la expresión establecida en el switch.

3.15.2.3 Default

Estructura que contiene las sentencias si en dado caso no haya salido del switch por medio de una sentencia **break**.

Se adjunta las distintas maneras de switch case anteriormente mencionados:

```

EDITOR
1 // EJEMPLO DE SWITCH
2 int edad = 18;
3 switch( edad ) {
4     Case 10:
5     WriteLine("Tengo 10 años.");
6     // mas sentencias
7     Break;
8     Case 18:
9     WriteLine("Tengo 18 años.");
10    // mas sentencias
11    Case 25:
12    WriteLine("Tengo 25 años.");
13    // mas sentencias
14    Break;
15    Default:
16    WriteLine("No se que edad tengo. :(");
17    // mas sentencias
18    Break;
19 }
20 /* Salida esperada
21 Tengo 18 años.
22 No se que edad tengo. :(
23 */
24

```

3.16 Sentencias Cíclicas

Son una secuencia de instrucciones de código que se ejecutan una vez tras otra mientras la condición que se ha asignado para que pueda ejecutarse sea verdadera. En el lenguaje actual, se podrán realizar 3 sentencias cíclicas que se describen a continuación.

3.16.1 While

Es una sentencia que ejecuta una secuencia de instrucciones mientras la condición de ejecución se mantenga verdadera.

```

EDITOR
1 //Ejemplo de cómo se implementar un ciclo while
2 while (x<100){
3     if (x > 50)
4     {
5         WriteLine("Mayor que 50");
6         //Más sentencias
7     }
8     else
9     {
10        WriteLine("Menor que 100");
11        //Más sentencias
12    }
13    X++;
14    //Más sentencias
15 }

```

3.16.2 For

Es una sentencia que nos permite ejecutar N cantidad de veces la secuencia de instrucciones que se encuentra dentro de ella.


```

EDITOR
1 //Ejemplo 1: declaración dentro del for con incremento
2 for ( int i=0; i<3;i++){
3   Writeline("i="+i)
4   //más sentencias
5 }
6 /*RESULTADO
7 i=0
8 i=1
9 i=2
10 */
11 //Ejemplo 2: asignación de variable previamente declarada y decremento por asignación
12 for ( i=5; i>2;i=i-1 ){
13   Writeline("i="+i)
14   //más sentencias
15 }
16 /*RESULTADO
17 i=5
18 i=4
19 i=3
20 */

```

3.16.3 Do-While

Es una sentencia que ejecuta al menos una vez el conjunto de instrucciones que se encuentran dentro de ella y que se sigue ejecutando mientras la condición sea verdadera.

```

EDITOR
1 Int a=5;
2 Do{
3   If (a>1 && a <3){
4     Writeline(true)
5   }
6   Else{
7     Writeline(false)
8   }
9   a--;
10 } while (a>0);
11 /*RESULTADO
12 false
13 false
14 false
15 true
16 true*/

```

3.17 Sentencias de transferencia

Permiten manipular el comportamiento de los bucles, ya sea para detenerlo o para saltarse algunas iteraciones. El lenguaje soporta las siguientes sentencias:

3.17.1 Break

La sentencia break hace que se salga del ciclo inmediatamente, es decir que el código que se encuentre después del break en la misma iteración no se ejecutara y este se saldrá del ciclo.

```

EDITOR
1 //Ejemplo en un ciclo for
2 for(int i = 0; i < 9; i++){
3     if(i==5){
4         WriteLine("Me salgo del ciclo en el numero " + i);
5         break;
6     }
7     WriteLine(i);
8 }
9

```

3.17.2 Continue

Puede detener la ejecución de la iteración actual y saltar a la siguiente.

```

EDITOR
1 //Ejemplo en un ciclo for
2 for(int i = 0; i < 9; i++){
3     if(i==5){
4         WriteLine("Me salte el numero " + i);
5         continue;
6     }
7     WriteLine(i);
8 }
9

```

3.17.3 Return

Finaliza la ejecución de un método o función y puede especificar un valor para ser devuelto a quien llama a la función.

3.18 Métodos

También es una subrutina de código que se identifica con un nombre, tipo y un conjunto de parámetros, aunque a diferencia de las funciones estas subrutinas no deben de retornar un valor. Para este lenguaje los métodos serán declarados haciendo uso de la palabra reservada 'void' al inicio, luego se indicará el identificador del método, seguido de una lista de parámetros dentro de paréntesis (esta lista de parámetros puede estar vacía en el caso de que la función no utilice parámetros).

Cada parámetro debe estar compuesto por su tipo seguido de un identificador, para el caso de que sean varios parámetros se debe utilizar comas para separar cada parámetro y en el caso de que no se usen parámetros no se deberá incluir nada dentro de los paréntesis. Luego de definir el método y sus parámetros se declara el cuerpo del método, el cual es un conjunto de instrucciones delimitadas por llaves {}.

```

EDITOR
1 void holamundo(){
2   WriteLine("Hola mundo");
3 }
4

```

3.19 Llamadas

Función que especifica la relación entre los parámetros reales y los formales y ejecuta la función. Se asocian normalmente por posición, aunque opcionalmente también se pueden asociar por nombre. Si la función tiene parámetros formales por omisión, no es necesario asociarles un parámetro real. Devuelve un resultado que ha de ser recogido, bien asignándolo a una variable del tipo adecuado, integrándolo en una expresión.

La sintaxis de las llamadas de los métodos y funciones serán la misma.

```

EDITOR
1 //Ejemplo de llamada de un método
2
3   WriteLine("Ejemplo de llamada a método");
4   holamundo();
5- /* Salida esperada
6   Ejemplo de llamada a método
7   Hola Mundo
8   */
9 //Ejemplo de llamada de una función
10  WriteLine("Ejemplo de llamada a función");
11  int num = suma(6,5); // a = 11
12  WriteLine("El valor de a es: " + a);
13- /* Salida esperada
14   Ejemplo de llamada a función
15   Aquí puede venir cualquier sentencia :D
16   El valor de a es: 11
17   */
18  int suma(int num1, int num2)
19- {
20    WriteLine("Aquí puede venir cualquier sentencia :D");
21    return num1 + num2;
22-   WriteLine("Aquí pueden venir más sentencias, pero no se ejecutarán
23   por la sentencia RETURN D:"); //WriteLine en una línea
24 }

```

3.20 Funciones WriteLine

Esta función nos permite imprimir expresiones con valores únicamente de tipo entero, doble, booleano, cadena y carácter.

```

1
2 //Ejemplo
3 WriteLine("Hola mundo!!");
4 WriteLine("Sale compi \n" + valor);
5 WriteLine(suma(2,2));
6

```

3.21 Función ToLower

Esta función recibe como parámetro una expresión de tipo cadena y retorna una nueva cadena con todas las letras minúsculas.

```

2
3 //Ejemplo
4 string cad_1 = toLower("hola MunDo"); // cad_1 = "hola mundo"
5 string cad_2 = toLower("RESULTADO = " + 100); // cad_2 = "resultado = 100"
6

```

3.22 Función ToUpper

Esta función recibe como parámetro una expresión de tipo cadena retorna una nueva cadena con todas las letras mayúsculas.

```

1
2 //Ejemplo
3 string cad_1 = toUpper("hola MunDo"); // cad_1 = "HOLA MUNDO"
4 string cad_2 = toUpper("resultado = " + 100); // cad_2 = "RESULTADO = 100"
5

```

3.23 Funciones Nativas

Length

Esta función recibe como parámetro un vector, una lista o una cadena y devuelve el tamaño de este.

Truncate

Esta función recibe como parámetro un valor numérico. Permite eliminar los decimales de un número, retornando un entero.

Round

Esta función recibe como parámetro un valor numérico. Permite redondear los números decimales según las siguientes reglas:
 Si el decimal es mayor o igual que 0.5, se aproxima al número superior
 Si el decimal es menor que 0.5, se aproxima al número inferior

Typeof

Esta función retorna una cadena con el nombre del tipo de dato evaluado.

To String

Esta función permite convertir un valor de tipo numérico o booleano en texto.

To CharArray

Esta función permite convertir una cadena en un arreglo de caracteres.

A continuación, se adjunta todas las funciones nativas mencionadas anteriormente

```

2 DynamicList<int> lista2 =new DynamicList <int>;
3 string[ ] vector2 = {"hola", "Mundo"};
4 int tam_lista = length(lista2); // tam_lista = 0
5 int tam_vector = length(vector2); // tam_vector = 2
6 int tam_hola = length(tam_vector[0]); // tam_hola = 4
7
8 // Ejemplo
9 int nuevoValor = truncate(3.53); // nuevoValor = 3
10 int otroValor = truncate(10); // otroValor = 10
11 double decimal = 15.4654849;
12 int entero = truncate(decimal); // entero = 15
13
14 // Ejemplo
15 Double valor = round(5.8); //valor = 6
16 Double valor2 = round(5.4); //valor2 = 5
17
18 //Ejemplo
19 DynamicList<int> lista2 =new DynamicList <int>;
20 String tipo = typeof(15); // tipo = "int"
21 String tipo2 = typeof(15.25); // tipo = "double"
22 String tipo3 = typeof(lista2); // tipo3 = "lista"
23
24 //Ejemplo
25 String valor = toString(14); // valor = "14"
26 String valor2 = toString(true); // valor = "true"
27
28 //Ejemplo
29 DynamicList<char> caracteres = toCharArray("Hola");
30 /*
31 caracteres [[0]] = "H"
32 caracteres [[1]] = "o"
33 caracteres [[2]] = "l"
34 caracteres [[3]] = "a"
35 */

```

3.24 Start with

Para poder ejecutar todo el código generado dentro del lenguaje, se utilizará la sentencia START WITH para indicar que método o función es la que iniciará con la lógica del programa.

```

EDITOR
1 //Ejemplo 1
2 void funcion1(){
3 Writeline("hola");
4 }
5 start with funcion1();
6 /*RESULTADO
7 hola
8 */
9 //Ejemplo 2
10 void funcion2(string mensaje){
11 Writeline(mensaje);
12 }
13 start with funcion2("hola soy un mensaje");
14 /*RESULTADO
15 Hola soy un mensaje
16 */
17
18

```

4. Reportes

Los reportes son una parte fundamental de SysCompiler, ya que muestra de forma visual las herramientas utilizadas para realizar la ejecución del código.

A continuación, se muestran ejemplos de estos reportes.

4.1 Tabla de Símbolos

Se muestran todas las variables, funciones y métodos declarados.

TABLA DE SÍMBOLOS						
#	Identificador	Tipo símbolo	Tipo dato	Entorno	Línea	Columna
1	r_toradians	Variable	1	Entorno	1	0
2	r_sine	Variable	1	Entorno	2	0
3	x1	Variable	1	Entorno	41	4
4	y1	Variable	1	Entorno	41	4
5	angle	Variable	1	Entorno	41	4
6	depth	Variable	0	Entorno	41	4
7	x	Variable	1	Entorno	27	8
8	sin	Variable	1	Entorno	8	4
9	fact	Variable	0	Entorno	9	4
10	i	Variable	0	Entorno	10	4
11	j	Variable	0	Entorno	13	8
12	x2	Variable	1	Entorno	28	8
13	y2	Variable	1	Entorno	31	8
1	toradians	Método	Void	Entorno	3	0
2	sine	Método	Void	Entorno	7	0
3	drawtree	Método	Void	Entorno	24	0
4	principal	Método	Void	Entorno	39	0

Se adjunta otro ejemplo de tabla de símbolos

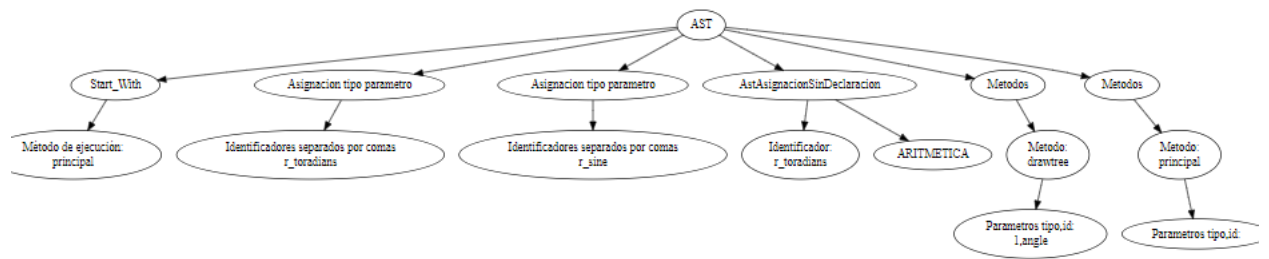
TABLA DE SÍMBOLOS						
#	Identificador	Tipo símbolo	Tipo dato	Entorno	Línea	Columna
1	n	Variable	0	Entorno	3	4
2	cadenafigura	Variable	4	Entorno	6	8
3	i	Variable	1	Entorno	7	8
4	j	Variable	1	Entorno	12	12
5	absolutoi	Variable	1	Entorno	15	16
6	absolutoj	Variable	1	Entorno	17	16
1	metodo1	Método	Void	Entorno	1	0
2	figura1	Método	Void	Entorno	5	0

4.2 Tabla de Errores

#	Tipo de Error	Descripción	Línea	Columna
1	Léxico	El carácter "\$" no pertenece al lenguaje.	7	32
2	Sintáctico	Encontrado Identificador "Ejemplo", se esperaba Palabra Reservada "Valor"	150	12

4.3 AST

Se muestra el árbol de sintaxis producido al analizar los archivos de entrada



4.4 Salidas en consola

La consola es el área de salida del intérprete. Por medio de esta herramienta se podrán visualizar las salidas generadas por la función nativa "WriteLine", así como los errores léxicos, sintácticos y semánticos.

EDITOR	CONSOLA
<pre> 1 void function(){ 2 int a=1; 3 writeline("hola :c"); 4 a=2.34; 5 } 6 start with function(); </pre>	<pre> Iniciando Analisis... >hola :c -->Error:Semantico --Mensaje:No se puede asignar el valor a la variable ya que es de un tipo diferente ya que la variables de tipo: 0 -->Linea:1 -->Columna:4 Analisis Finalizado... </pre>