

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES



TRADUCTORES DE LENGUAJE II

ARMANDO RAMOS BARAJAS

ACT 2. ANÁLISIS SINTÁCTICO

FECHA DE ENTREGA: 09 de octubre del 2022

SECCIÓN: D04

ALUMNOS:

- Robles Mora Alejandra
- Paniagua Rosales Lysander Josué
- Monreal Zambrano Edgar Omar

Introducción

En la actividad anterior aprendimos hacer los tokens de un compilador, para esta actividad vamos a validar la sintaxis que usan los compiladores para ensamblar el programa, para saber si se está haciendo una asignación correctamente, una operación aritmética o hacer un bloque de código correctamente (que tenga su operador de apertura y cierre correspondiente) por ejemplo para las clases tiene ese carácter de apertura "{" y su correspondiente cierre "}"; las funciones, las condicionales, los bucles "while", "for" y "do while" tiene ambos caracteres, los paréntesis para los parámetros o hacer la condición y las llaves para el resto del bloque. Para esto se debe de tener la sintaxis sino no se entendería el código.

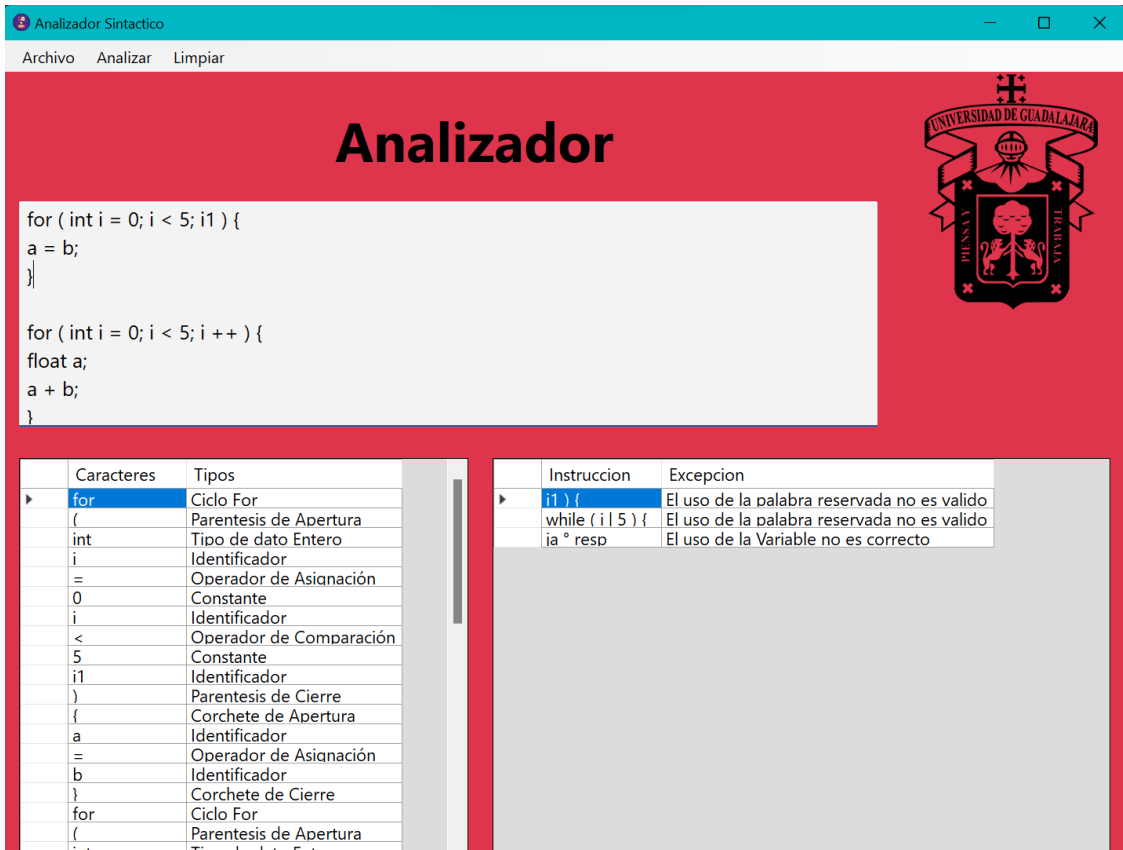
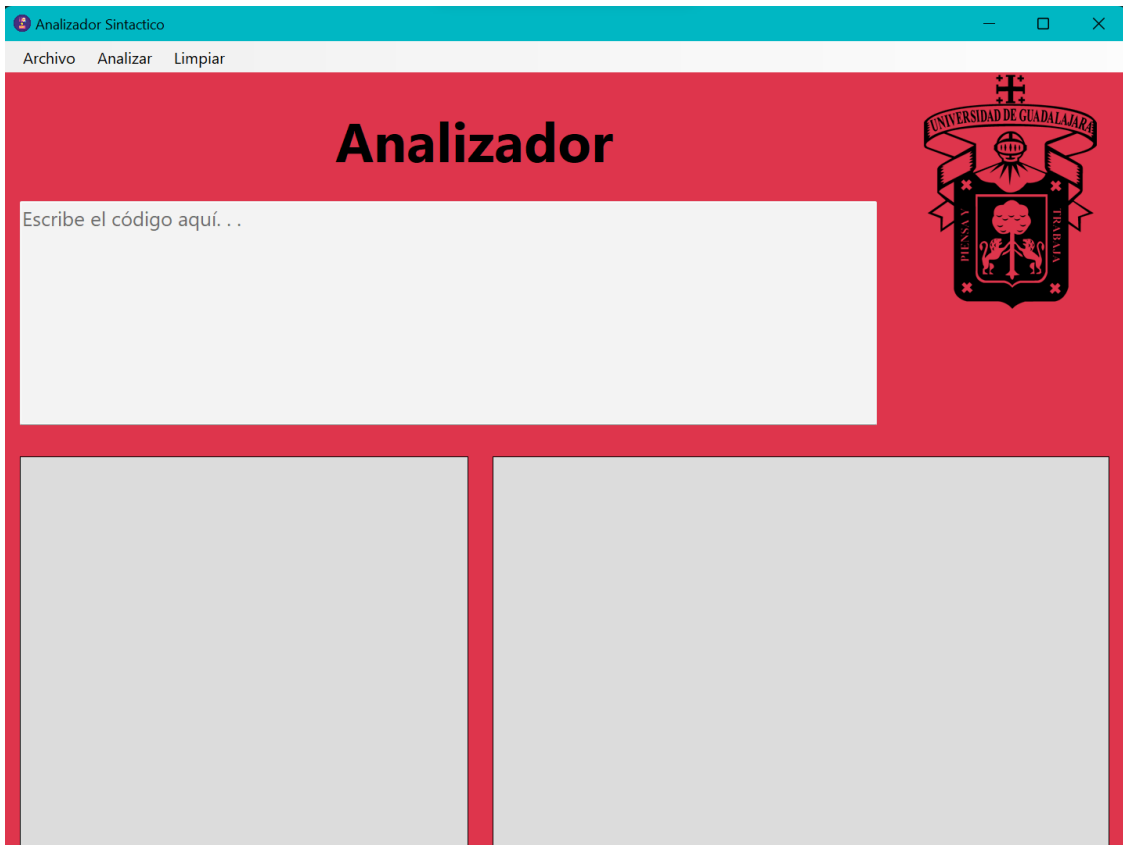
Objetivo general

Establecer una sintaxis para la lectura del código y entender los bloques que tiene un programa estándar. Validar si una declaración de una variable es correcta o si una función está bien definida.

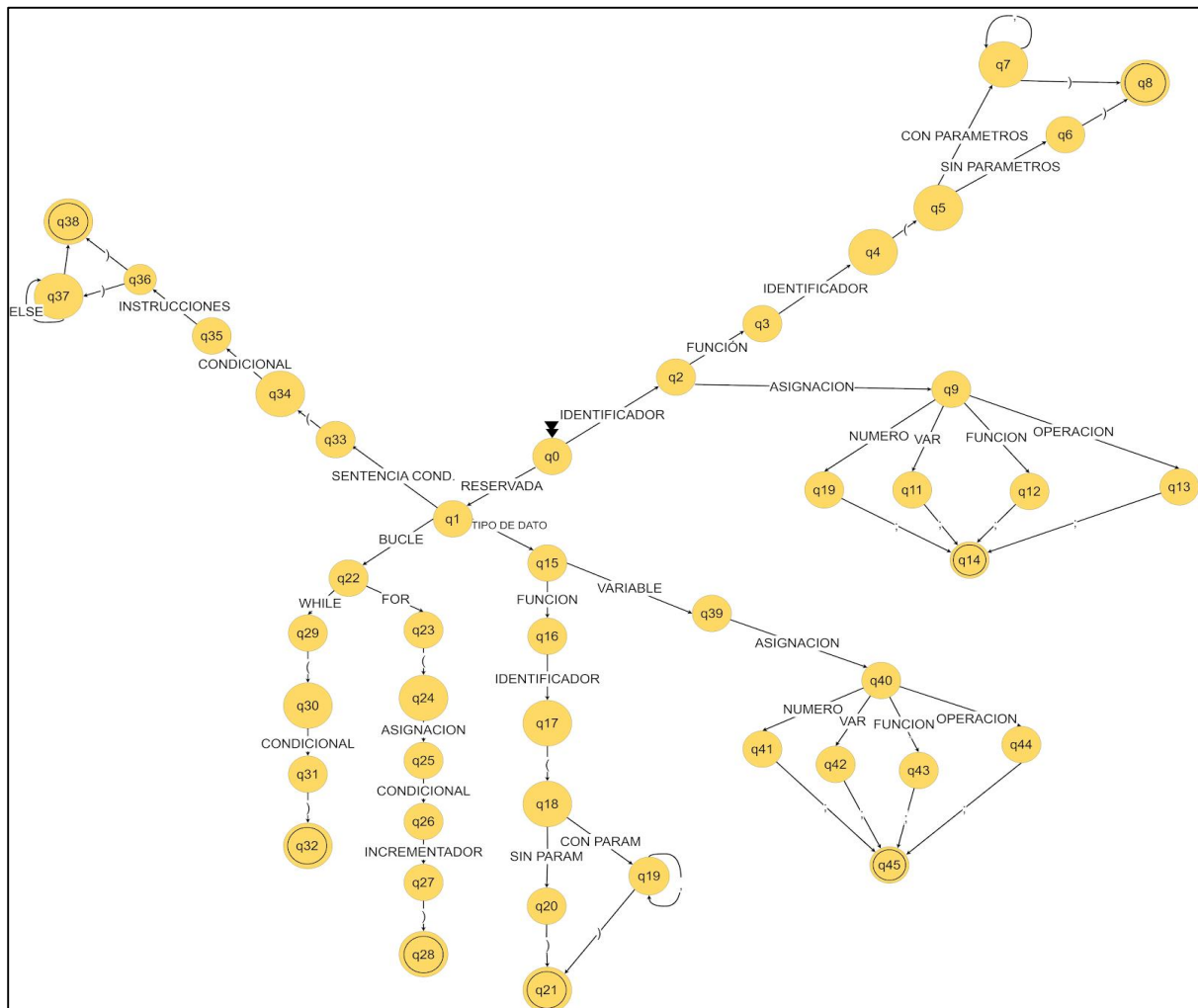
Objetivo particular

Ingresar una instrucción en el teclado y analizar qué tipo de instrucción se trata identificando la primera palabra con el analizador léxico, una vez identificada que tipo de instrucción es validar su respectiva sintaxis. Si no es una palabra reservada solo valida si se trata de una asignación o llamada a una función.

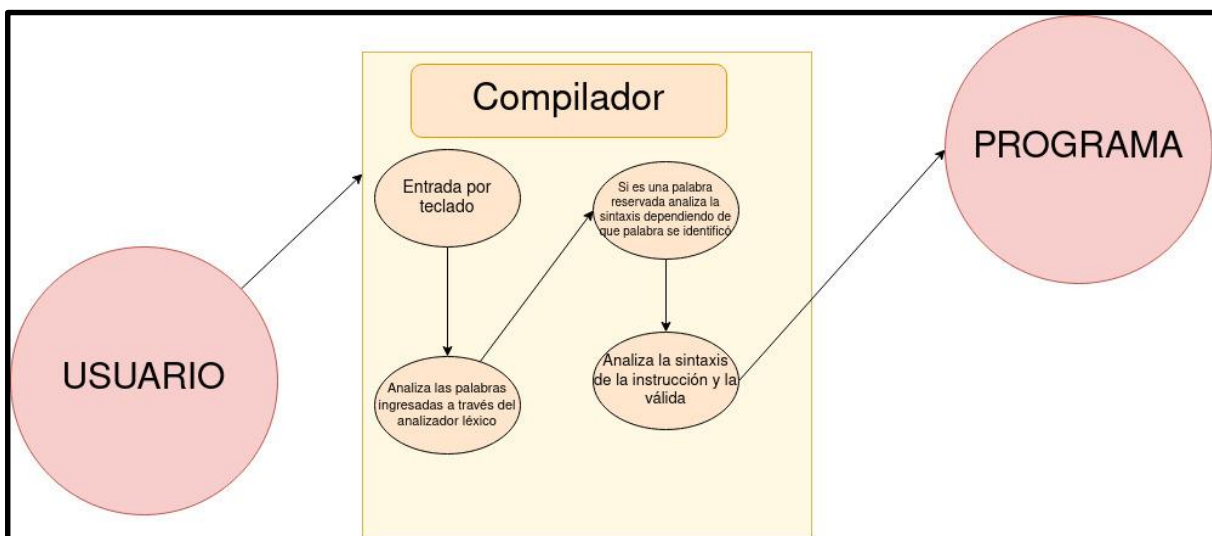
Desarrollo (pantallazos)



Apéndices (grafos)



Diagramas de casos de estudio



Conclusiones

En el análisis sintáctico es complicado validar el orden de una instrucción y más cuando hay saltos de línea entre un parámetro; por ejemplo hay veces en que en páginas web con javascript en peticiones AJAX se usan `.then()` y ésta en sus parámetros tiene una función dentro que por supuesto contiene saltos de línea y en ese punto fue un poco complicado ya que teníamos que validar varios caracteres de agrupación y conocer cuál le corresponde a qué operador de apertura.

Un punto que consideramos era cómo iba a ser nuestra sintaxis, si como Python o C++/JS ya que en C++/JS la variable es declarada con un tipo de dato (`int`, `double`, `float`, `char`, `string`, etc.) Aquí no estamos analizando si una variable fue declarada anteriormente o no ya que eso es parte del análisis semántico y eso lo decidiremos en su debido tiempo. Pero decidimos usar un poco de ambos, la sintaxis de Python para las variables. En las funciones usar la sintaxis de otros lenguajes como C++, JavaScript, Java, C#, etc. Es decir la sintaxis de Python en funciones no tiene llaves, sino solamente tabuladores para saber que parte del código es parte del bloque de la función, pero en los demás lenguajes mencionados se usa llaves para conocer que parte del código forma del bloque de la función.

El pensar como va hacer nuestro compilador con lo que en realidad podíamos realizar por nuestros conocimientos en un lenguaje nuevo hizo cambiar algunos puntos para adaptarnos a nuestros límites y realizar la entrega. Y nos imaginamos que las compañías así manejan su trabajo en la programación, teniendo en cuenta el objetivo pero también las limitantes del equipo o de los recursos, acercándonos más a la realidad donde no todo se puede hacer por cuestiones de tiempo o costos.

Requisitos funcionales

Número de requisito	1
Nombre de requisito	Bucles
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	Si es <code>while</code> , <code>do while</code> revisa que solo tenga un comparador, no se acepta asignaciones. Si es <code>for</code> revisa que antes del primer <code>;</code> sea una asignación o declaración, el segundo que sea la condicional y el ultimo parametro que solo diga una variable o tenga una operación de asignación, para todos los casos verifica si se tiene un agrupador de apertura y cierre de paréntesis
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Ingresar una palabra reservada para identificar de que bucle se trata

Número de requisito	2
Nombre de requisito	Condicionales
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	Si se ingresa una palabra verifica que solo se tenga una comparativa o y sus 2 paréntesis de apertura y cierre
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Ingresar condicional (if, else if, else, switch)

Número de requisito	3
Nombre de requisito	Funciones
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	Verifica que la primera palabra no sea una palabra reservada para identificarlo como el nombre de la función, verifica que tenga sus agrupadores de y que si tiene comas debe de tener un valor más de 1 por la cantidad de comas, es decir variables = comas + 1
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Ingresar un identificador para la función y sus agrupadores; si existen parámetros ingresar igual o más de uno con sus respectivas comas

Número de requisito	4
Nombre de requisito	Asignaciones y/o declaraciones de variables
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	La primera palabra de la línea de código analiza si corresponde a un tipo de dato y si no es así entonces (y no es ninguna otra palabra reservada) es un identificador, analiza si después del identificador hay un punto y coma. Si hay un carácter = identifica si es un valor numérico, si tiene paréntesis valida que sea una función o si son letras solamente válida que tenga su punto y coma ya que se trata de otra variable.
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Ingresar una palabra reservada para identificar de que bucle se trata

Requisitos no funcionales

Número de requisito	1
Nombre de requisito	Interfaz gráfica
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	A través de una GUI el usuario podrá ingresar texto y a partir de ahí identificar qué palabra reservada existe.
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial <input checked="" type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

El usuario podrá ingresar líneas de código (con saltos de línea si lo desea) y se hará un proceso para verificar cuales palabras que se ingresan son de bucles, condicionales, tipos de datos o identificadores a una variable. También verificará si en caso de que existan agrupadores tenga el de apertura y el de cierre, si son asignaciones que tengan un valor de una variable, operación, numérico o llamada a una función.

Caja negra y caja blanca

Analizador Sintactico

Archivo Analizar Limpiar


Analizador

```
int Hola = 1;
cout << Hola <<;

while ( i | 9 ) {
cout << i <<;
}
int i = 0;
```

Caracteres	Tipos
int	Tipo de dato Entero
Hola	Identificador
=	Operador de Asignación
1	Constante
cout	Identificador
<<	Flujo de Salida
Hola	Identificador
<<	Flujo de Salida
while	Ciclo While
(Parentesis de Apertura
i	Identificador
	Caracter No Valido
9	Constante
)	Parentesis de Cierre
{	Corchete de Apertura
cout	Flujo de Salida
<<	Flujo de Salida
i	Identificador
<<	Flujo de Salida

Instruccion	Excepcion
cout << Hola <<	El uso de la Variable no es correcto
while (i 9) {	El uso de la palabra reservada no es valido
while (i ? ? 5) {	El uso de la palabra reservada no es valido
i)	El uso de la palabra reservada no es valido
int QA respondi	La declaracion de un nuevo Tipo de Dato no es correcta



Analizador Sintactico

Archivo Analizar Limpiar


Analizador

```
int Hola = 1;
cout << Hola <<;

int i = 0;
while ( i || 9 ) {
cout << i <<;
}
```

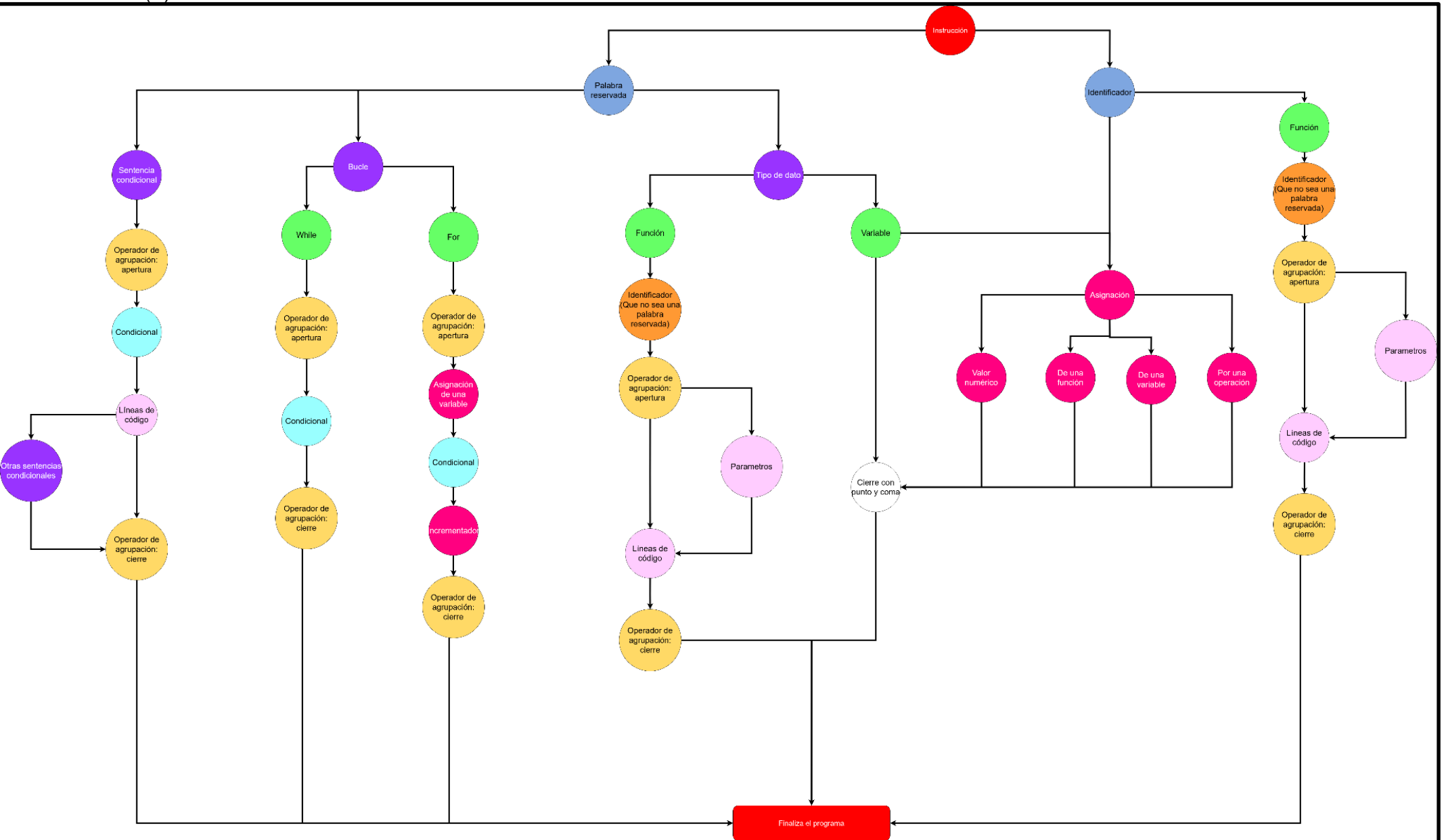
Caracteres	Tipos
int	Tipo de dato Entero
Hola	Identificador
=	Operador de Asignación
1	Constante
cout	Flujo de Salida
<<	Flujo de Salida
Hola	Identificador
<<	Flujo de Salida
int	Tipo de dato Entero
i	Identificador
=	Operador de Asignación
0	Constante
while	Ciclo While
(Parentesis de Apertura
i	Identificador
	Operador OR
9	Constante
)	Parentesis de Cierre
{	Corchete de Apertura

Instruccion	Excepcion
-------------	-----------



Complejidad ciclomática

$$C(V) = \text{Aristas} - \text{Nodos} + 2$$



COCOMO

Estimación de líneas de código y puntos de función										
Puntos de Función No Alineados										
Complejidad										
Descripción	Baja			Media			Alta			
Entradas	1	x	3		x	4		x	6	3
Salidas	1	x	4		x	5		x	7	4
Procesos	5	x	3	4	x	4		x	6	31
Archivos	2	x	7	1	x	10		x	15	24
Interfaces		x	5	1	x	7		x	10	7
									Suma Total	69

Lenguaje	f	LCD/PF	PF=	47.96
JS	4.5	42	LCD=	353

KLDC	353
------	-----

El Modelo Básico de COCOMO estima el esfuerzo y el tiempo empleado en el desarrollo de un proyecto de software usando dos variables predictivas denominadas factores de costo (cost drivers): el tamaño del software y el modo de desarrollo.

Las ecuaciones básicas son:

Tipo de proyecto	Ecuaciones		
	Esfuerzo (en meses-personas)	Duración (en meses)	Número de personas
Modo Orgánico	$2.4(KLDC)^{1.05}$	$2.5(Esfuerzo)^{0.38}$	$Esfuerzo/Duración$
Modo Semiacoplado	$3.0(KLDC)^{1.12}$	$2.5(Esfuerzo)^{0.35}$	$Esfuerzo/Duración$
Modo Empotrado	$3.6(KLDC)^{1.20}$	$2.5(Esfuerzo)^{0.32}$	$Esfuerzo/Duración$

Modo	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semi-Acoplado	3	1.12	2.5	0.35
Empotrado	3.6	1.2	2.5	0.32

MODO	Orgánico
ESFUERZO	1135.990455
TIEMPO	36.22283086
PERSONAS	31.36117272

KLDC	353
------	-----

Un mes-persona equivale a 152 horas de trabajo y corresponde a la cantidad de tiempo que una persona dedica durante un mes a trabajar en un proyecto de desarrollo de software. Este valor tiene en cuenta los fines de semana pero excluye feriados y vacaciones.

Costo Total del proyecto x número de horas:	\$271,671.23
Costo final con ganancia:	\$353,172.60