

1. Estructura de Archivos.

Dentro del proyecto del DFD_GC se encuentran los siguientes tipos de archivos:

1.1 Archivos con extensión .h o archivos de cabecera.

Los archivos de cabecera son requeridos para la programación estructurada y en ellos se definen las funciones que se utilizarán en los archivos fuente (.cpp). Para este proyecto se crearon los siguientes archivos:

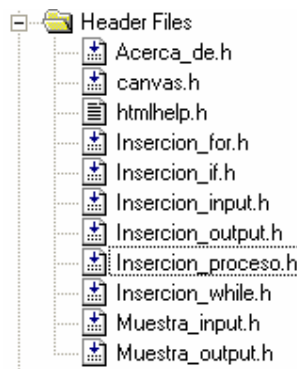


Figura 1. Archivos de cabecera creados para el proyecto.

1.2 Archivos moc con extensión .cpp.

Los archivos moc son creados con la herramienta Meta Object Compiler y son necesarios para el funcionamiento de los formularios creados desde el Qt Designer.

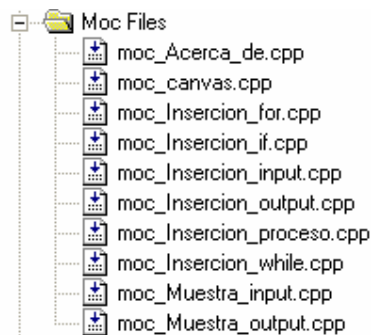


Figura 2. Archivos moc creados para el proyecto.

1.3 Archivos fuente de extensión .cpp.

Los archivos fuentes contienen la implementación de las funciones creadas en los archivos de cabecera.

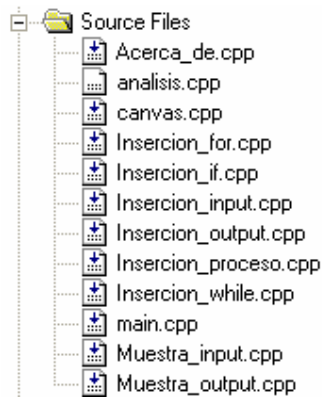


Figura 3. Archivos fuentes creados para el proyecto.

2. Formularios existentes.

Para el desarrollo de la aplicación se crearon los siguientes formularios:

2.1 Formulario de inserción de proceso.

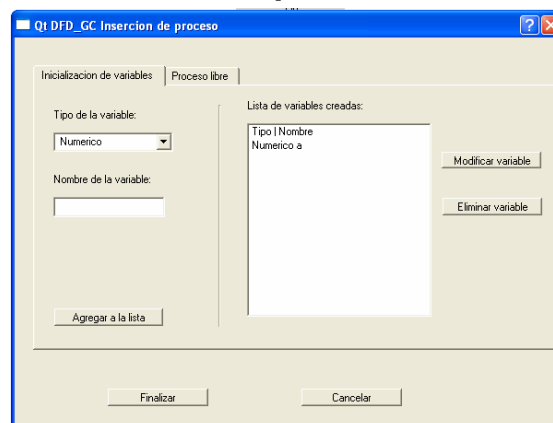


Figura 4. Formulario de inserción de proceso.

Este formulario posee los elementos necesarios para la creación de variables y las sentencias para poder dimensionar un objeto de tipo proceso

dentro del diagrama. Dichos parámetros son pasados a una instancia de la clase Lista que guarda la información de todos los objetos creados hasta el momento. Para poder crear este formulario es necesario de los siguientes archivos:

- Inserción_proceso.h: El cual contiene la clase frm_proceso y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc_Insercion_proceso.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Inserción_proceso.cpp: Archivo fuente que contiene toda la implementación de la clase frm_proceso y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.2 Formulario de inserción de input.

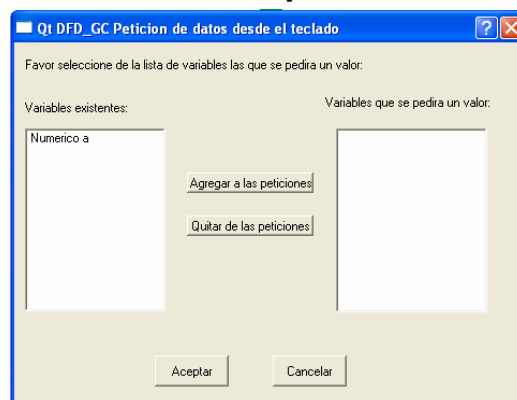


Figura 5. Formulario de inserción de input.

Este formulario posee los elementos necesarios para la petición de datos desde el teclado y para poder dimensionar un objeto de tipo input dentro del diagrama. Dichos parámetros son pasados a una instancia de la clase Lista que guarda la información de todos los objetos creados hasta el momento. Para poder crear este formulario es necesario de los siguientes archivos:

- Inserción_input.h: El cual contiene la clase Inserción_input y todas las funciones que deben realizar los elementos creados en el formulario.

- Moc Insercion input.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Inserción input.cpp: Archivo fuente que contiene toda la implementación de la clase Inserción_input y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.3 Formulario de inserción de output.

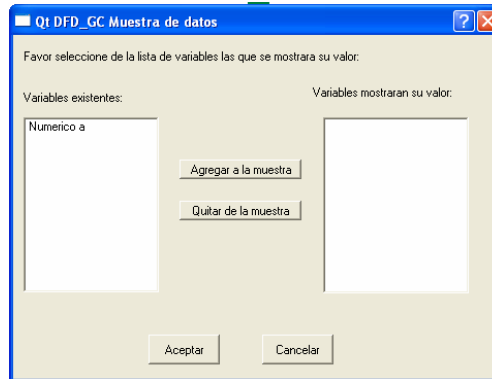


Figura 6. Formulario de inserción de output.

Este formulario posee los elementos necesarios para la muestra del valor de las variables en pantalla y para poder dimensionar un objeto de tipo output dentro del diagrama. Dichos parámetros son pasados a una instancia de la clase Lista que guarda la información de todos los objetos creados hasta el momento. Para poder crear este formulario es necesario de los siguientes archivos:

- Inserción output.h: El cual contiene la clase Inserción_output y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc Insercion output.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Inserción output.cpp: Archivo fuente que contiene toda la implementación de la clase Inserción_output y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.4 Formulario de inserción de while.

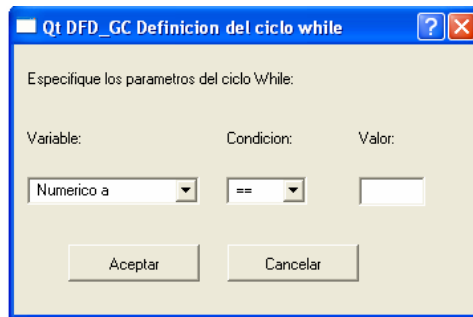


Figura 7. Formulario de inserción de while.

Este formulario posee los elementos necesarios para la creación de un ciclo while y para poder dimensionar un objeto de tipo while dentro del diagrama. Dichos parámetros son pasados a una instancia de la clase Lista que guarda la información de todos los objetos creados hasta el momento. Para poder crear este formulario es necesario de los siguientes archivos:

- Inserción_while.h: El cual contiene la clase Inserción_while y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc_Insercion_while.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Inserción_while.cpp: Archivo fuente que contiene toda la implementación de la clase Inserción_while y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.5 Formulario de inserción de for.

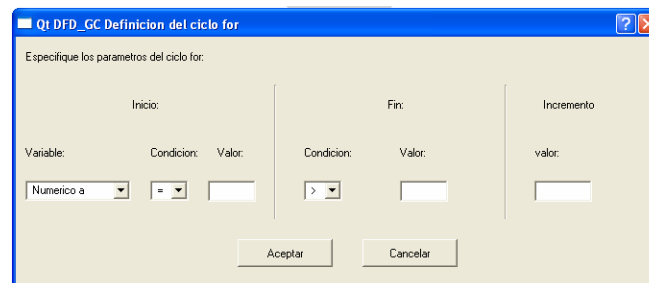


Figura 8. Formulario de inserción de for.

Este formulario posee los elementos necesarios para la creación de un ciclo for y para poder dimensionar un objeto de tipo for dentro del diagrama. Dichos parámetros son pasados a una instancia de la clase Lista que guarda la

información de todos los objetos creados hasta el momento. Para poder crear este formulario es necesario de los siguientes archivos:

- Inserción_for.h: El cual contiene la clase Inserción_for y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc_Insercion_for.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Inserción_for.cpp: Archivo fuente que contiene toda la implementación de la clase Inserción_for y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.6 Formulario de inserción de if.

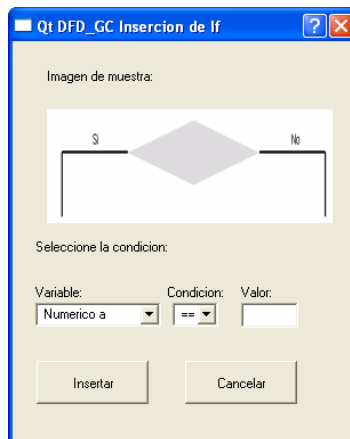


Figura 9. Formulario de inserción de if.

Este formulario posee los elementos necesarios para la creación de una decisión if y para poder dimensionar un objeto de tipo if dentro del diagrama. Dichos parámetros son pasados a una instancia de la clase Lista que guarda la información de todos los objetos creados hasta el momento. Para poder crear este formulario es necesario de los siguientes archivos:

- Inserción_if.h: El cual contiene la clase Inserción_if y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc_Insercion_if.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.

- Inserción_if.cpp: Archivo fuente que contiene toda la implementación de la clase Inserción_if y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.7 Formulario de muestra input.

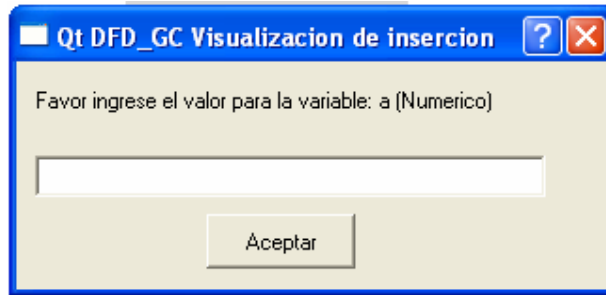


Figura 10. Formulario de inserción de muestra input.

Este formulario se muestra en tiempo de ejecución y por medio de este se le da el valor a la variable seleccionada. Para poder crear este formulario es necesario de los siguientes archivos:

- Muestra_input.h: El cual contiene la clase muestra_input y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc_Muestra_input.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Muestra_input.cpp: Archivo fuente que contiene toda la implementación de la clase muestra_input y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

2.8 Formulario de muestra output.

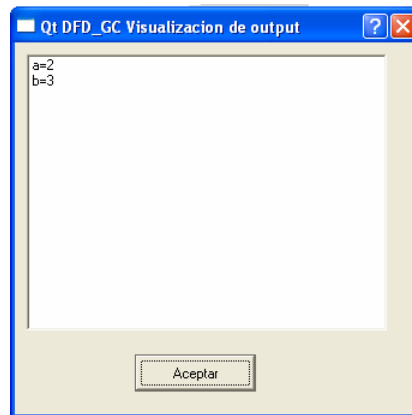


Figura 11. Formulario de muestra output.

Este formulario se muestra en tiempo de ejecución y por medio de este se muestra el valor que tiene las variables seleccionadas. Para poder crear este formulario es necesario de los siguientes archivos:

- Muestra_output.h: El cual contiene la clase muestra_output y todas las funciones que deben realizar los elementos creados en el formulario.
- Moc Muestra_output.cpp: El cual hace posible que los elementos creados en el formulario tengan funcionalidad.
- Muestra_output.cpp: Archivo fuente que contiene toda la implementación de la clase muestra_otput y maneja toda la información en tiempo de ejecución del formulario llamado desde la clase Main.

3.0 Clases implementadas.

3.1 Clase elemento.

Esta clase contiene la información de todos los objetos a crear dentro del diagrama.

Variables:

Tipo	Nombre	Descripción
entero	x	Valor de la posición en x dentro del canvas
entero	y	Valor de la posición en y dentro del canvas
entero	tipo	indica el tipo de objeto
entero	tif	indica el tipo de bucle o if que es el objeto

entero	cif	Contiene el numero de objetos creados de cada tif
entero	rama	Indica si se encuentra en una rama de if y si lo esta indica en que rama esta ya sea izquierda o derecha
lógico	nuevo	indica si el objeto se acaba de insertar dentro del canvas
carácter	contenido1	contiene todo el contenido del objeto
carácter	contenido2	variable que sirve como contenido auxiliar del objeto
carácter	texto	Almacena el texto a mostrar dentro del canvas
entero	ite	Variable que contiene el numero de iteraciones realizadas en un bucle
carácter	var_for	Almacena la variable de control dentro de un bucle for
carácter	inc_for	Almacena el incremento dentro de un bucle for
carácter	cond_for	Almacena la condición de paro de un bucle for

Tabla 4. Tabla de variables de la clase elemento.

Métodos:

Tipo	Nombre	Variables que necesitan un valor	Parámetro devuelto	Descripción
Qstring	t_ele		texto	Devuelve el texto del elemento
Qstring	c1_ele		contenido1	Devuelve el contenido1 del elemento
Qstring	c2_ele		contenido2	Devuelve el contenido2 del elemento
Qstring	cond_ele		cond_for	Devuelve la condición de paro de un ciclo for
Qstring	inc_ele		inc_for	Devuelve el incremento de un ciclo for
Qstring	var_ele		var_for	Devuelve la variable de control del ciclo for
int	x_ele		x	Devuelve el valor de x del objeto
int	y_ele		y	Devuelve el valor de y del objeto
int	tipo_ele		tipo	Devuelve el tipo de objeto
int	rama_ele		rama	Devuelve el valor de rama del objeto
int	cif_ele		cif	Devuelve el valor de cif del objeto
int	tif_ele		tif	Devuelve el valor de tif del objeto
int	ite_ele		ite	Devuelve el valor de ite del objeto
void	set_pos	x,y		Se utiliza para guardar la ubicación del objeto
void	set_cif	cif		Se le da valor al cif del objeto
void	set_texto	texto, contenido1, contenido2		Se guarda todo el contenido + el texto a mostrar
void	set_rama	rama		Se almacena el valor de rama
void	set_var	var_for		Se guarda el valor de la variable de control
void	set_cond	cond_for		Se guarda la condición de paro
void	set_inc	inc_for		Se guarda la expresión de

				incremento
void	clr_texto			texto, contenido 1 y 2 son limpiados
void	clr_rama			Se limpia el valor de rama
void	clr_nuevo			Se limpia el valor de nuevo
void	clr_ite			Se limpia valor de ite
void	inc_ite			Se aumenta en 1 el valor de ite
bool	nuevo_ele			Retorna el valor de la variable nuevo

Tabla 5. Tabla de métodos de la clase elemento.

3.2 Clase EditorFigura.

Clase heredada de QCanvasView necesaria para el manejo de eventos del Mouse dentro del canvas.

Métodos:

Tipo	Nombre	Descripción
void	limpiar	Elimina todos los objetos creados en el canvas
void	eliminar_objeto	Elimina el objeto seleccionado
void	posicionar_objeto	Posiciona dentro del canvas el nuevo objeto insertado
void	ordenar_lista	Se ordena la lista de objetos por medio del método burbuja
void	ocultar_vacio	Oculto los cuadros que hacen posible la inserción de un nuevo objeto
void	mostrar_vacio	Muestra los cuadros que hacen posible la inserción de un nuevo objeto
void	borrar_seleccion	Borra el rectángulo que se crea alrededor de los objetos al momento de la ejecución
void	modificar_objeto	Modifica el contenido de un objeto seleccionado
void	i_proceso	Se determina la información del nuevo objeto y se agrega a la lista de objetos creados
void	i_input	Se determina la información del nuevo objeto y se agrega a la lista de objetos creados
void	i_output	Se determina la información del nuevo objeto y se agrega a la lista de objetos creados
void	i_while	Se determina la información del nuevo objeto y se agrega a la lista de objetos creados
void	i_for	Se determina la información del nuevo objeto y se agrega a la lista de objetos creados
void	i_if	Se determina la información del nuevo objeto y se agrega a la lista de objetos creados
void	buscar_if	Se identifica si el objeto se encuentra dentro una rama de un if

Tabla 6. Tabla de métodos de la clase EditorFigura.

3.3 Clase Main.

Clase heredada de QMainWindow, esta es la ventana principal que contiene el canvas.

Variables:

Tipo	Nombre	Descripción
EditorFigura	editor	Objeto que permite acceder a los métodos de la clase EditorFigura.
QCanvas	canvas	Objeto de tipo canvas que se convierte en la zona de trabajo del editor de diagrama
QPopupMenu	Opciones	Objeto de tipo popupmenu que se utiliza para la interacción del menú herramienta doble buffer
Entero	Dbf_id	Valor que guarda el estado del doble buffer

Tabla 7. Tabla de variables de la clase Main.

Métodos:

Tipo	Nombre	Descripción
void	crear_proceso	Crea un nuevo objeto de tipo proceso
void	crear_input	Crea un nuevo objeto de tipo input
void	crear_output	Crea un nuevo objeto de tipo output
void	crear_while	Crea un nuevo objeto de tipo while
void	crear_for	Crea un nuevo objeto de tipo for
void	crear_if	Crea un nuevo objeto de tipo if
void	crear_finwhile	Crea un nuevo objeto de tipo terminal de un while
void	crear_finfor	Crea un nuevo objeto de tipo terminal de un for
void	crear_finif	Crea un nuevo objeto de tipo terminal de un if
void	crear_fin	Crea un nuevo objeto terminal del diagrama principal
void	crear_seleccion	Crea un nuevo objeto de tipo terminal de un while
void	abrir_archivo	Se realiza el procedimiento para la carga de un nuevo diagrama a partir de un archivo guardado
void	ins_flecha	Se dibuja una flecha hacia abajo en el canvas
void	ins_vacio	Se crea un nuevo objeto de tipo c_vacio
void	ins_vacioizq	Se crea un nuevo objeto de tipo c_vacio_izq
void	ins_vacioder	Se crea un nuevo objeto de tipo c_vacio_der
void	linea_rama	Se dibuja una línea vertical para las ramas de un if
void	linea_b	Se dibuja una línea blanca en el canvas
void	if_hor_izq	Se crean las líneas para completar un if horizontal con condición verdadera del lado izquierdo
void	if_hor_der	Se crean las líneas para completar un if horizontal con condición verdadera del lado derecho
void	info_bucle	Permite saber la variable y el incremento del bucle que se está ejecutando
int	ejecucion	Contiene todo el proceso para ejecutar los distintos objetos

int	fin_bucle	devuelve el índice del fin de un bucle cuando se esta ejecutando
-----	-----------	--

Tabla 8. Tabla de métodos de la clase Main.

Public slots:

Los public slots son funciones que pueden ser accedidas a partir de un evento como un clic del Mouse, el presionar una tecla o una selección del menú de la aplicación.

Tipo	Nombre	Descripción
void	limpiar	Se utiliza para eliminar todos los objetos del canvas
void	nuevo	Se crea un nuevo diagrama, función accedida desde el menú
vod	abrir	Despliega el dialogo para cargar una nuevo diagrama desde un archivo, función accedida desde el menú
void	guardar	Guarda en un archivo el diagrama actual, función accedida desde el menú
void	guardar_como	Guarda en un nuevo archivo especificado el diagrama actual, función accedida desde el menú
void	actualizar	Actualiza el canvas cuando se ha realizado algún cambio, función accedida desde el menú
void	ins_proceso	prepara el canvas para permitir la inserción de un nuevo objeto proceso
void	ins_input	prepara el canvas para permitir la inserción de un nuevo objeto input
void	ins_output	prepara el canvas para permitir la inserción de un nuevo objeto output
void	ins_while	prepara el canvas para permitir la inserción de un nuevo objeto while
void	ins_for	prepara el canvas para permitir la inserción de un nuevo objeto for
void	ins_if	prepara el canvas para permitir la inserción de un nuevo objeto if
void	salir	Termina la ejecución de la aplicación, función accedida desde el menú
void	ejecutar_hasta	Indica hasta que objeto se debe realizar la ejecución dentro de un diagrama, función accedida desde el menú
void	detener_ejecucion	Detiene la ejecución actual del diagrama, función accedida desde el menú
void	generar_c	Genera la codificación del diagrama en lenguaje c++, función accedida desde el menú
void	generar_java	Genera la codificación del diagrama en lenguaje Java, función accedida desde el menú
void	ayuda	Abre el archivo de ayuda de la aplicación, función accedida desde el menú
void	acerca_de	Abre el formulario donde se muestra la información sobre la aplicación

Tabla 9. Tabla de public slots de la clase Main.

3.4 Clase variable.

Contiene toda la información de las variables creadas dentro del diagrama.

Variables:

Tipo	Nombre	Descripción
Entero	tipo	Indica si el tipo es numérico, lógico o carácter
Entero	valor_l	Contiene el valor lógico de la variable
Decimal	valor_n	Contiene el valor numérico de la variable
Carácter	nombre	Contiene el nombre de la variable creada
Carácter	valor_c	Contiene el valor tipo carácter

Tabla 10. Tabla de variables de la clase variable.

Metodos:

Tipo	Nombre	Descripción
void	svalor_n	Sirve para establecer el valor numérico de la variable
void	svalor_l	Sirve para establecer el valor lógico de la variable
void	svalor_c	Sirve para establecer el valor de tipo carácter de la variable
float	rvalor_n	Retorna el valor numérico de la variable
int	rvalor_l	Retorna el valor lógico de la variable
int	vtipo	Retorna el tipo de la variable
Qstring	vnombre	Retorna el nombre de la variable
Qstring	rvalor_c	Retorna el valor de tipo carácter de la variable

Tabla 11. Tabla de métodos de la clase variable.

3.5 Clase c_token.

Se utiliza esta clase para el análisis del contenido de un objeto en específico. En esta se guarda la información de cada token generado de una sentencia.

Variables:

Tipo	Nombre	Descripción
Entero	tipo	Indica si el tipo de token es numérico, lógico o carácter

Decimal	val_n	Contiene el valor numérico de la variable
Caracter	car	Contiene el nombre del token
Caracter	val_c	Contiene el valor de tipo carácter del token

Tabla 12. Tabla de variables de la clase *c_token*.

Métodos:

Tipo	Nombre	Descripción
void	set_valn	Sirve para establecer el valor numérico del token
void	set_valc	Sirve para establecer el valor de tipo carácter del token
void	set_tipo	Sirve para establecer el tipo de token
float	ret_valn	Retorna el valor numérico del token
int	Tipo_ele	Retorna el tipo de token
Qstring	car_ele	Retorna el token
Qstring	ret_valc	Retorna el valor de tipo carácter de la variable

Tabla 13. Tabla de métodos de la clase *c_token*.

3.6 Clase análisis.

Se utiliza para el análisis de una línea que se extrae del contenido de un objeto en específico.

Variables:

Tipo	Nombre	Descripción
Entero	tipo_linea	Indica el tipo de línea a analizar
Decimal	valor_n	Contiene el valor numerico de la variable
Caracter	t_error	Contiene el nombre de la variable creada
Caracter	l_error	Contiene el valor tipo carácter
Logico	h_error	Bandera que sirve para indicar si ha habido un error o no

Tabla 14. Tabla de variables de la clase *análisis*.

Métodos:

Tipo	Nombre	Descripción
bool	hubo_error	Devuelve si hubo algún error al analizar la línea.
Qstring	texto_error	Devuelve el mensaje de error que se dio
Qstring	linea_error	Devuelve la línea donde ocurrió el error
void	set_error	Sirve para establecer que hubo un error
void	inicio	Prepara las variables y estructuras para iniciar el análisis de un diagrama

void	avariable	Analiza la creación de las variables
void	asentencia	Analiza las sentencias o asignaciones
void	apeticion	Analiza las peticiones de valores para las variables del diagrama
void	atexto	Devuelve el valor almacenado en las variables
void	esentencia	Contiene todo el proceso para ejecutar una sentencia
void	econdicion	Contiene todo el proceso para evaluar una sentencia lógica
void	ver_espacios	Se verifica el numero de espacios de una línea
void	ver_nombre	Se hace la verificación del nombre de una variable
void	ver_parentesis	Se hace un chequeo de los paréntesis de la sentencia
void	ver_operadores	Se hace la segmentación de la línea en tokens y se analiza cada uno
int	acondicion	Se analiza si una condición cumple con los elementos para ser evaluada
int	econdicion	Se evalúa la condición y se devuelve el resultado de la misma
int	es_funcion	Se verifica si un token es una función y se devuelve de que tipo es
int	es_variable	Se verifica si un token es una variable y se devuelve de que tipo es
int	es_constante	Se verifica si es una constante y se devuelve de que tipo es

Tabla 15. Tabla de métodos de la clase análisis.

3.7 Clases de objetos.

A continuación se presenta una tabla conteniendo las clases derivadas de las clases objetos que tiene Qt para permitir la creación de un nuevo objeto dentro del canvas. El valor rtti es un número entero único que se utiliza para identificar cada objeto dentro del canvas.

Clase heredada	Clase Padre definida en Qt	Valor rtti devuelto	Descripción de objeto
c_vacio	QCanvasRectangle	9111	Rectángulo verde que permite especificar la posición de un nuevo elemento
c_linea	QCanvasLine	9222	Línea negra, sus dimensiones son especificadas al crear el objeto
c_lineab	QCanvasLine	9232	Línea blanca, sus dimensiones son especificadas al crear el objeto
c_texto	QCanvasText	9333	texto libre, se utiliza en la rama de los if
c_terminal	QCanvasRectangle	9444	Objeto terminal que indica el fin del diagrama
c_seleccion	QCanvasRectangle	9555	Rectángulo color aqua que se coloca sobre el objeto que se esta ejecutando
c_vacioizq	QCanvasRectangle	9666	Rectángulo verde que especifica que un objeto esta en la rama izquierda de un if
c_vacioder	QCanvasRectangle	9777	Rectángulo verde que especifica que un objeto esta en la rama derecha de un if
c_proceso	QCanvasRectangle	1111	Figura gris que indica un proceso dentro del diagrama
c_tproceso	QCanvasText	1110	Texto que se muestra sobre la figura de proceso

c_input	QCanvasPolygon	2222	Figura gris que indica un input o petición de datos desde el teclado dentro del diagrama
c_tinput	QCanvasText	2220	Texto que se muestra sobre la figura de input
c_output	QCanvasPolygon	3333	Figura gris que indica un output o muestra de datos en pantalla dentro del diagrama
c_toutput	QCanvasText	3330	Texto que se muestra sobre la figura de output
c_while	QCanvasPolygon	4444	Figura gris que indica un proceso dentro del diagrama
c_twhile	QCanvasText	4440	Texto que se muestra sobre la figura de proceso
c_for	QCanvasPolygon	5555	Figura gris que indica un proceso dentro del diagrama
c_tfor	QCanvasText	5550	Texto que se muestra sobre la figura de proceso
c_if	QCanvasPolygon	6666	Figura gris que indica un proceso dentro del diagrama
c_tif	QCanvasText	6660	Texto que se muestra sobre la figura de proceso
c_finwhile	QCanvasRectangle	7111	Objeto terminal que indica el fin de un bucle while
c_finfor	QCanvasRectangle	7222	Objeto terminal que indica el fin de un bucle for
c_finif	QCanvasRectangle	7333	Objeto terminal que indica el fin de un if

Tabla 16. Tabla de clases heredadas para la creación de objetos.

4. ESTRUCTURA DE DATOS UTILIZADAS

A continuación se presentan las estructuras de datos a utilizar, para esto se usaron las funciones que brinda el Qt para manejo de listas enlazadas y pilas.

4.1 LISTAS ENLAZADAS

- Lista

Se creo esta lista utilizando instancias de la clase elemento. Esta lista contiene todos los objetos de tipo proceso, input, output, while, for e if creados.

- Variables

Se creo esta lista utilizando instancias de la clase variable. Esta lista contiene todas las variables creadas a lo largo del diagrama que se esta ejecutando.

- Tokens

Se creo esta lista utilizando instancias de la clase `c_token`. Esta lista contiene todos los tokens que se generan del análisis de una línea al momento de ejecutar un objeto.

- Postfija

Se creo esta lista utilizando instancias de la clase `c_token`. Esta lista contiene la expresión en notación postfija de una sentencia.

4.2 PILAS

- Pila

Se creo esta pila utilizando instancias de la clase `c_token`. Se utiliza esta pila de forma auxiliar para la conversión de notacion infija a postfija.

- Pila_if

Se creo esta pila utilizando instancias de la clase `elemento`. Se utiliza esta pila de forma auxiliar para la ejecución de los if anidados.