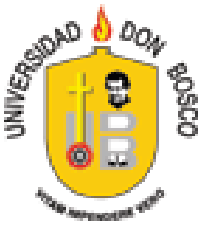
UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA



TRABAJO DE GRADUACION PARA OPTAR AL GRADO DE INGENIERO EN CIENCIAS DE LA COMPUTACION

DESARROLLO DE UN EDITOR E INTERPRETE DE DIAGRAMAS DE FLUJO DE DATOS CON GENERACION DE CODIGO EN C++ Y JAVA

PRESENTADO POR:

OSCAR DOUGLAS BONILLA CASTRO

FERNANDO ENRIQUE CASTRO MORALES

ASESOR:

ING. JAIME ANAYA

SEPTIEMBRE DE 2007

SOYAPANGO – SAN SALVADOR - EL SALVADOR

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

AUTORIDADES

ING. MIGUEL FEDERICO HUGUET RIVERA
RECTOR

PBRO. VICTOR BERMUDEZ YANEZ
VICERRECTOR ACADEMICO

LIC. MARIO OLMOS ARGUETA
SECRETARIO GENERAL

ING. ERNESTO GODOFREDO GIRON
DECANO FACULTAD DE INGENIERIA

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

TRABAJO DE GRADUACION PARA OPTAR AL GRADO DE
INGENIERO EN CIENCIAS DE LA COMPUTACION

DESARROLLO DE UN EDITOR E INTERPRETE DE DIAGRAMAS DE FLUJO DE DATOS CON GENERACION DE CODIGO EN C++ Y JAVA

TRIBUNAL EXAMINADOR

ING. RENE TEJADA	ING. MELVYN GOMEZ	DR. JORGE LEMUS
JURADO EVALUADOR	**JURADO EVALUADOR**	**JURADO EVALUADOR**

ING. JAIME ANAYA	ING. CARLOS TEJADA
ASESOR	**TUTOR**

AGRADECIMIENTOS (Oscar Bonilla)

Es inevitable dedicar esta tesis a Dios y a todos los seres queridos que me rodean. Primero a Dios porque fue él quien puso todas esas herramientas en mi camino para que yo lograra todas las metas que me he propuesto a lo largo de mi vida cotidiana, y de mi vida Universitaria.

A mis padres Nelson Bonilla y Gloria Castro de Bonilla por haberme inculcado el valor de no desanimarme jamás por los obstáculos que encontrase en mi vida, fueron ellos los que primero me enseñaron a encontrar una solución a todos los problemas que se presentaran ante mí. Me enseñaron a mirar mas allá de los problemas de solo lo que se muestra ligeramente sino a analizar toda la situación como una sola sobre los sucesos que podrían pasar al tomar una decisión.

A mis hermanos Nelson y Salvador por haber estado ahí cada vez que necesite un consuelo, alguien con quien hablar, con quien reír o con quien apoyarme cuando no encontraba la solución de algo. A mi novia Korina por estar ahí para apoyarme, y no dejar que me desanimara nunca, a tener entusiasmo y coraje para poder salir adelante.

A los Scouts por mostrarme el camino de una disciplina sana y vocación a ayudar a los demás sin esperar nada a cambio. A mis profesores a lo largo de mi vida como estudiante que me han ido mostrando un camino lleno de cosas nuevas que aprender.

A mi compañero de tesis por no desanimarse y trabajar a mi lado para que los objetivos se lograran; a nuestro asesor el Ing. Jaime Anaya y nuestro tutor el Ing. Carlos Tejada por guiarnos sobre este camino culminante del trabajo de graduación. Agradezco también a mis amigos y compañeros de universidad que estuvieron ahí para apoyarme y hacer que se sintiera menos la presión de estudiar a Walter, Rigoberto, Josué, José Luís, Angel, Ricardo a todos ellos gracias.

AGRADECIMIENTOS (Fernando Castro)

A Dios por haberme dado la oportunidad de existir en este tiempo y en este espacio.

A mi madre María, que siempre ha sido un modelo a seguir, siempre estuvo ahí cuando necesite apoyo y sin su esfuerzo no hubiera sido posible la culminación de mi carrera.

A mi abuela Maria Brígida, que ha sido el segundo pilar en mi vida y siempre me ha apoyado en todo.

A mi asesor Ing. Anaya y a mi tutor Ing. Tejada, sin su apoyo no habría sido posible la realización de este trabajo de graduación.

A mi compañero de tesis Oscar, sin su apoyo no habría sido posible la culminación de este proyecto.

A mis amigos, que siempre a lo largo de toda mi carrera me brindaron sus conocimientos y su apoyo incondicional.

INDICE

Índice de Figuras.

Índice de tablas.

INTRODUCCIÓN.

Actualmente el uso de la informática no se limita a crear aplicaciones comerciales enfocadas a realizar tareas complejas que faciliten el desarrollo de una tarea en empresas. Sino que se está volviendo una necesidad la creación de software educativo y de alto nivel que pueda facilitar la capacitación de los nuevos programadores y así puedan posteriormente enfocarse en desarrollar aplicaciones más complejas.

Por lo que para analizar y resolver cualquier tipo de problema es necesario seguir una serie de pasos, es decir, es necesaria la implementación de un algoritmo por medio del cual se pueda resolver dicho problema. Uno de los conceptos más aceptados de algoritmo dice: "es un procedimiento para la resolución de problemas de cualquier tipo por medio de una determinada secuencia de pasos simples y no ambiguos". La manera más fácil de representar un algoritmo es por medio de un diagrama de flujo de datos.

El presente proyecto consiste en el desarrollo de un editor e interprete de diagramas de flujo de datos con generación de código en C++ y Java. Con el desarrollo de esta aplicación se pretende beneficiar a los nuevos programadores brindándoles una herramienta que les permita asimilar los conocimientos básicos de la informática de una manera práctica y además se vayan familiarizando con los lenguajes de alto nivel.

En el presente documento se muestran los antecedentes y la importancia por la cual este proyecto se ha realizado, también se da un breve marco teórico que sirva como introducción a la aplicación. Así como también todo lo relacionado a la planificación para el desarrollo de este proyecto como lo son alcances y objetivos que sirvan para dimensionar la aplicación.

CAPITULO I

ANTECEDENTES E IMPORTANCIA

1.1 ANTECEDENTES.

La introducción de la informática como un apoyo al proceso de aprendizaje y asimilación de conceptos básicos en entidades educativas ha resultado en la demanda creciente de software educativo. Por lo que es necesario brindar una mejor capacitación a los nuevos programadores por medio de herramientas que les permitan asimilar de una manera más practicas los conceptos básicos de la informática.

La investigación se enfoca en desarrollar una aplicación que permita al usuario generar un diagrama de flujo de datos de manera fácil y además brinde herramientas que permitan interactuar con dicho diagrama para comprender de una manera práctica el funcionamiento del mismo. También se podrá generar el respectivo código en C++ y Java del diagrama creado, este es un aspecto que hace que esta aplicación sea única en el mercado, ya que no existe ninguna herramienta que permita la generación de código en lenguaje de alto nivel a partir de un diagrama de flujo de datos.

La única aplicación enfocada al desarrollo de diagramas de flujos de datos es el Dfd® que es un editor e intérprete de diagramas de flujo. Con la cual la Universidad del Magdalena obtuvo el primer puesto con el trabajo de investigación y desarrollo "Editor e Intérprete de Algoritmos Representados en Diagramas de Flujo" en el Segundo Evento de Investigación y Divulgación Tecnológica en el Área de Sistemas y Computación, realizado a nivel nacional por la Universidad Cooperativa de Colombia (Santa Marta, octubre de 1997).

También obtuvo el primer puesto con el trabajo de investigación y desarrollo "Editor e Intérprete de Algoritmos Representados en Diagramas de Flujo" en el IV Premio Colombiano de Informática Educativa, realizado en Manizales por RIBIE-COL y el Ministerio de Educación Nacional en abril de 1998.

El proyecto fue publicado en la revista RIE (Revista de Informática Educativa) de la Universidad de los Andes en el volumen 11 Nº 1.

El proyecto se presentó en el IV congreso Iberoamericano de informática educativa RIBIE '98, con una ponencia ante la comunidad de habla hispana y portuguesa en el marco del congreso del 20 al 23 de octubre de 1998 en BRASILIA - BRASIL.

Aunque está herramienta tiene sus facilidades también presenta ciertos errores de diseño y de consumo de hardware, además no permite la generación de código en lenguaje de alto nivel.

Lo que se pretende con la aplicación es potenciar la capacidad de los nuevos programadores brindándoles una herramienta que les permita interactuar con los algoritmos de manera práctica y que además puedan desde un principio generar software.

1.2 IMPORTANCIA DE LA INVESTIGACIÓN.

Actualmente no es posible encontrar una aplicación que permita a los nuevos programadores diseñar algoritmos por medio de diagramas de flujo y generar su respectivo código funcional. En las instituciones educativas se enseña de manera teórica el manejo de algoritmos y se enseña posteriormente la generación de código en un lenguaje de alto nivel. No existen laboratorios de algoritmos, ni mucho menos aplicaciones que permitan poner en práctica los nuevos conocimientos sobre algoritmos.

El diseño de software utilizando diagramas de flujo es una de las herramientas básicas de la informática ya que con esto se incrementa la capacidad de análisis y se va forjando en los nuevos programadores una disciplina de ingeniería de software ya que actualmente cuando es necesario diseñar una aplicación lo primero que se hace es comenzar a programar sin haber hecho antes una diseño completo de la aplicación deseada lo que deriva en gasto innecesario de tiempo y recursos.

1.3 PLANTEAMIENTO DEL PROBLEMA.

¿Existe actualmente en el mercado o en las instituciones educativas una herramienta informática que facilite la adquisición de conocimientos sobre algoritmos de manera práctica y que además tenga la facilidad de generar código en lenguajes de alto nivel como lo son C++ y Java?

Por el momento no ha sido posible encontrar una herramienta de este tipo que sea fácil de utilizar y le permita al alumno experimentar de una manera práctica el manejo de algoritmos y a la vez permita familiarizarse con los lenguajes de alto nivel.

1.3.1 DEFINICIÓN DEL TEMA.

Desarrollo de un Editor e Intérprete de Diagrama de Flujo de Datos con generación de código en C++ y Java para la enseñanza y Aprendizaje de la Programación Estructurada.

El objetivo es desarrollar una herramienta que permita al alumno crear diagramas de flujo de datos y estos puedan ser analizados por medio de herramientas de ejecución como “Ejecución Paso a Paso” y una ventana de depuración que muestre como cambian los valores de las expresiones a medida que un algoritmo se ejecuta. Se tendrán también las opciones de generar el código en Java o C++ del diagrama de flujo que se ha creado.

1.4 JUSTIFICACIÓN.

La base de la programación en computadoras esta en el manejo de algoritmos. Por lo que brindar una herramienta que facilite al instructor la enseñanza de una manera más interactiva y práctica beneficiará la asimilación de nuevos conocimientos por parte del alumno.

Se pretende el desarrollo de la creatividad y el aumento de interés en estudiantes que se encuentran en niveles básicos de aprendizaje, brindándoles una herramienta que les permita diseñar algoritmos en diagramas de flujo de

datos y les permita manipularlo a su antojo para posteriormente generar la codificación en lenguaje de alto nivel.

La aplicación está orientada a los nuevos programadores pero su interfaz permitirá que personas de cualquier otra disciplina de ingeniería puedan generar diagramas de flujo de datos en el que se detallen procesos como de producción, manufactura, de seguridad, etc.

1.5 OBJETIVOS.

1.5.1 OBJETIVO GENERAL.

- Desarrollar un Editor e Intérprete de Diagrama de Flujo de Datos con generación de código en C++ y Java.

1.5.2 OBJETIVOS ESPECÍFICOS.

- Desarrollo de un ambiente gráfico que facilite la interacción del usuario con la aplicación.
- Minimizar los requerimientos de hardware para el funcionamiento de la aplicación.
- Facilitar la creación, edición y almacenamiento de los diagramas de flujo de datos del usuario.
- Generar herramientas de ejecución y depuración de fácil utilización y comprensión.
- Generar codificación a los lenguajes Java y C++ del diagrama de flujo de manera sencilla y efectiva.
- Generar diagramas de flujo ejemplos para que el usuario puede tener una biblioteca de algoritmos en el cual apoyarse.
- Elaboración de un manual de usuario y archivos de ayuda para la comprensión completa de la herramienta.

1.6 ALCANCES.

La aplicación se ha desarrollado utilizando Visual C++ 6.0 con Qt Designer 4.2.2, diseñado para correr en la plataforma Windows.

La aplicación cuenta con una interfaz gráfica que facilita la creación y edición de diagramas de flujo de datos.

El usuario puede guardar su diagrama de flujo de datos en el disco duro o en cualquier sistema de almacenamiento. Puede cargar un diagrama que haya creado anteriormente para modificarlo o examinarlo en profundidad.

El usuario puede utilizar herramientas de ejecución como “Ejecución Paso a Paso” y una ventana de depuración que muestre como cambian los valores de las expresiones a medida que un algoritmo se ejecuta.

Los errores que estén presentes en la sintaxis o en la evaluación de las expresiones en cualquier lugar del diagrama se mostraran al usuario en mensajes indicándole donde se encuentran. Antes de la ejecución del algoritmo se realizará una revisión completa del diagrama para garantizar que no haya problemas posteriores para la generación de código.

El usuario podrá optar por generar el código al lenguaje C++ o Java del diagrama de flujo que se este utilizando. Se creará un archivo .cpp en el caso de C++ y .java para el caso de Java. Dicho archivo conteniendo el código podrá ser compilado y ejecutado en cualquier herramienta comercial o no comercial sin errores y deberá ejecutar el algoritmo que se ha representado en el diagrama de flujo.

1.7 DELIMITACIÓN.

La aplicación generará la codificación hacia los lenguajes C++ y Java del diagrama de flujo con el cual se este trabajando, pero en ningún momento la aplicación será capaz de compilar ni ejecutar dicho código.

Se garantiza que el código generado sea efectivo y pueda ser procesado por cualquier herramienta comercial o no comercial como Microsoft Visual C++, Dev C++, etc. para el caso de C++ y NetBeans, JDeveloper, etc. para el caso de java. La aplicación no podrá utilizar funciones predefinidas por el usuario.

1.8 PROYECCIÓN SOCIAL.

La base de la programación en computadoras esta en el manejo de algoritmos. Por lo que se pretende crear una aplicación gratuita y de libre distribución que facilite la comprensión y asimilación de conocimientos informáticos en el área de algoritmos para alumnos que se encuentran en sus primeros años de formación en las ciencias de la computación.

Se quiere crear una herramienta que permita al instructor enseñar de una manera más dinámica conocimientos en el área de algoritmos que son la base para la comprensión de temas posteriores y que permitan al alumno pueda llevar a la practica los conocimientos que se van adquiriendo desde un principio para así potenciar su creatividad e interés por la computación.

La facilidad de uso de la aplicación permitirá que usuarios de otras carreras de ingeniería puedan utilizar la aplicación para generar sus diagramas de flujo de datos en el cual estén plasmados procesos de producción, de seguridad, etc.

1.9 METODOLOGÍA DE LA INVESTIGACIÓN.

La investigación realizada es directa como también de tipo experimental. En la primera fase se realizó la recopilación de la información, enfocada en la teoría de diagramas de flujo y en técnicas de programación que nos faciliten la interpretación y la generación de código.

Como segunda fase se tuvo el diseño de la aplicación que permita generar diagramas de flujo de datos y brinde herramientas de depuración y ejecución. Así como también permita la generación de código en C++ y Java.

1.9.1 INVESTIGACIÓN BIBLIOGRÁFICA

Se Investigó acerca de distintos componentes y herramientas de programación necesarias para la creación de un editor e intérprete de diagramas de flujo de datos. En el material utilizado se incluyen manuales, libros, e información existente en Internet.

1.9.2 ASESORÍA PROFESIONAL

Se crearon sesiones con profesionales en la materia que tengan experiencia en programación y en aspectos didácticos, que nos permitieron recopilar mayor información acerca de los aspectos a vigilar en el desarrollo de la aplicación.

1.9.3 INFORMACIÓN DISPONIBLE EN INTERNET

Una herramienta que permite la búsqueda de amplios conocimientos sobre programación y adquisición de herramientas de desarrollo de software. Para la investigación de el proyecto, Internet fue la herramienta de mayor potencial que se utilizó, encontrando documentación validada a nivel mundial, por reconocidas empresas dedicadas al rubro de software, así como también prestigiosas universidades a nivel mundial, como documentación de profesionales dedicados a compartir sus investigaciones por medio de Internet.

1.9.4 ESTUDIO TÉCNICO DE APLICACIONES EXISTENTES

1.9.4.1 DFD.

Dfd® es un editor e intérprete de diagramas de flujo creado en 1997. Su interfaz gráfica facilita en gran medida la creación de diagramas de flujo para la representación de algoritmos que solucionan problemas por computadora.

El programa Dfd® en su totalidad, fue realizado en lenguaje C++ utilizando programación orientada a objetos y corre bajo Windows a 32 bits, ver figura 1. El software Dfd® incluye un archivo de ayuda para Windows, que provee un fácil acceso a información necesaria para trabajar con Dfd®, documentación sobre los símbolos u objetos que conforman los diagramas, las funciones, los operadores, los mensajes de error y demás aspectos relacionados con Dfd®.

El software Dfd® se encuentra inscrito en el Registro Nacional de Derechos de Autor de Colombia, correspondiente al libro 13, tomo 3, partida 166.

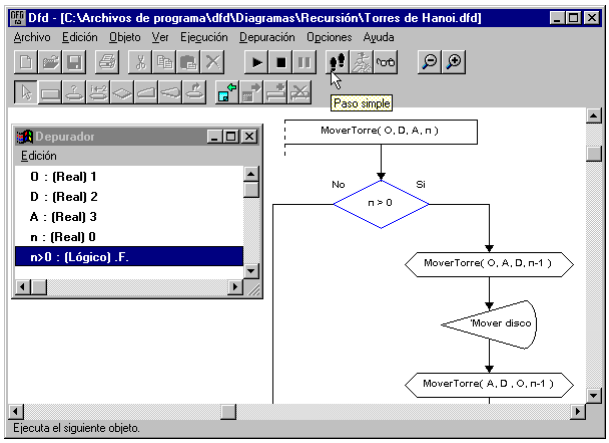


Figura 1. Pantalla principal del Dfd®.

El Dfd® no posee la generación de código en lenguaje de alto nivel, además su interfaz pese a ser amigable tiene errores de diseño; además posee un error en cuanto a consumo de procesador ya que al utilizar la pantalla de diseño de diagrama el uso de procesador llega al 100% y se mantiene hasta que no se utilice cualquier otra función de la aplicación.

1.9.4.2 AUTOFLOWCHART 1.0

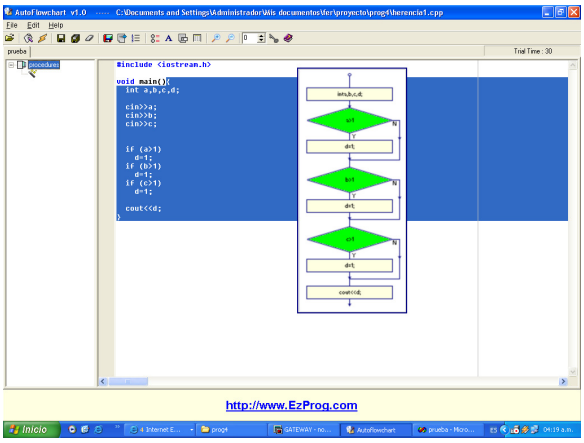


Figura 2. Pantalla principal de Auto FlowChart 1.0

AutoFlowchart es un programa que permita la esquematización a partir de un código fuente. Dicho esquema se puede expandir o encoger y el usuario puede predefinir las medidas del esquema. Permite mover y agrandar el esquema, se puede exportar dicho esquema al tipo de archivo Microsoft Word o de Mapa de Bits. Sirve a los programadores a visualizar la idea general del código fuente.

Diseñado para correr en cualquier versión de Windows, con un tamaño de 1.37 MB de archivo de instalación. Es libre para probar con una licencia de 30 días, el precio para adquirirlo es de $79. Soporta C, C++, VC++ (Visual C++ .NET) y Delphi.

Esta aplicación hace lo contrario a la aplicación propuesta en este proyecto, ya que a partir del código fuente genera el DFD pero no permite la manipulación del mismo ni mucho menos utilizar herramientas de ejecución como ejecución paso a paso. Además su precio de adquisición es elevado, la única forma es adquirirlo a través de Internet.

1.9.4.3 SMARTDRAW.

Este es un software que permite la creación detalladas de diagramas de una manera rápida y efectiva. Incluye cientos de plantillas profesionales.

Soporta los siguientes tipos de diagramas:

- Diagramas UML
- Diagramas ERD (Entidad - Relacion)
- Diagramas de flujo de datos (DFD)
- SSADM
- Mapa de sitios Web

El precio de la licencia de este software es:

- 1 usuario: $197
- 5 usuario: $885
- 10 usuario: $1495

Este software esta diseñado solo para crear Diagramas, no permite la manipulación a nivel de ejecución de los DFD y su precio de adquisición esta demasiado elevado.

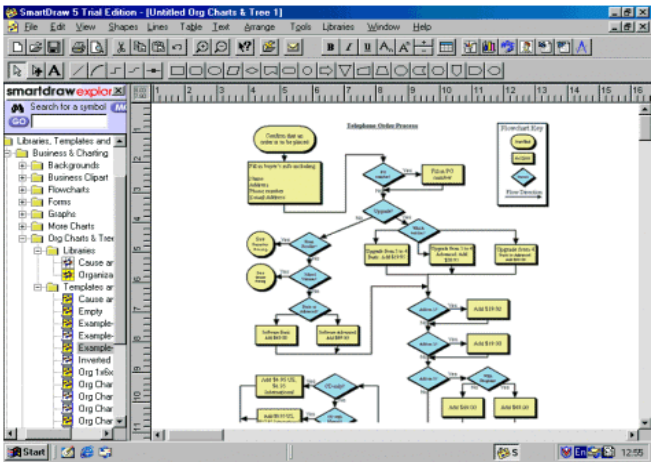


Figura 3. Pantalla principal de SmartDraw

CAPITULO II
MARCO TEORICO

2.1 REFERENCIAS HISTÓRICAS.

La palabra *algoritmo* proviene del nombre del matemático llamado Muhammad ibn Musa al-Jwarizmi que vivió entre los siglos VIII y IX. Su trabajo consistió en preservar y difundir el conocimiento de la antigua Grecia y de la India. Sus libros eran de fácil comprensión, de ahí que su principal logro no fuera el de crear nuevos teoremas o corrientes de pensamiento, sino el de simplificar la matemática a punto tal que pudieran ser comprendidas y aplicadas por un mayor número de personas. Cabe destacar cómo señaló las virtudes del sistema decimal indio (en contra de los sistemas tradicionales árabes) y cómo explicó que, mediante una especificación clara y concisa de cómo calcular sistemáticamente, se podrían definir algoritmos que fueran usados en dispositivos mecánicos en vez de las manos (por ejemplo, ábacos). También estudió la manera de reducir las operaciones que formaban el cálculo. Es por esto que aun no siendo el creador del primer algoritmo, el concepto lleva aunque no su nombre, sí su pseudónimo.

Así, de la palabra *algorismo*, que originalmente hacía referencia a las reglas de uso de la aritmética utilizando dígitos árabes, se evolucionó a la palabra latina, derivación de al-Khwarizmi, algobarismus, que más tarde mutaría a *algoritmo* en el siglo XVIII. La palabra ha cambiado de forma que en su definición se incluye a todos los procedimientos finitos para resolver problemas. Ya en el siglo XIX, se produjo el primer algoritmo escrito para un computador. La autora fue Ada Byron, en cuyos escritos se detallaban la máquina analítica en 1842. Por ello que es considerada por muchos como la primera programadora aunque, desde Charles Babbage, nadie completó su máquina, por lo que el algoritmo nunca se implementó.

La falta de rigor matemático en la definición de "procedimiento bien definido" para los algoritmos trajo algunas dificultades a los matemáticos y lógicos del siglo XIX y comienzos de XX. Este problema fue en gran parte resuelto con la descripción de la máquina de Turing, un modelo abstracto de computadora formulado por Alan Turing, y la demostración de que cualquier método anticipado por otros matemáticos que pueda encontrarse para describir "procedimientos bien definidos" puede ser emulado en una máquina de Turing (una afirmación conocida como "tesis de Church-Turing").

En la actualidad, el criterio formal para definir un algoritmo es que se trata de un proceso que puede implementarse en una máquina de Turing completamente especificada, o en alguno de los formalismos equivalentes

2.2 MARCO CONCEPTUAL.

2.2.1 COMPUTADORAS Y SOFTWARE.

El software es un elemento totalmente intangible, pero sin este elemento una computadora no podría trabajar. Muchas veces se tiene una computadora en la que se quiere realizar un trabajo pero muchas de esas veces no es de conocimiento del usuario que para poder lograr el trabajo final o el resultado de este trabajo sea exitoso, la computadora debe pasar por una serie de trabajos intermedios; para que esto suceda de alguna forma se le debe dar la información a la computadora de cómo realizar los procesos intermedios para que esta pueda realizar su objetivo; dicha información se transmite a la computadora siguiendo una terminología determinada y esto es lo que conforma el software.

Se han desarrollado diversas técnicas para analizar los trabajos que realiza la computadora, así como un conjunto de símbolos y palabras producto de los análisis efectuados para ordenar de modo racional los pasos que debe seguir la computadora para realizar el trabajo. Este conjunto de órdenes constituye lo que comúnmente se llama programa. A cada una de las órdenes que constituyen este programa se les llama instrucciones o sentencias.

Dentro del software existe otro nivel complejo y especializado que se encarga de efectuar el enlace entre los programas y los elementos de hardware. Por esta razón el software se divide en dos apartados:

- *Software de Base.*
- *Software de Aplicación.*

2.2.2 SOFTWARE DE BASE Y DE APLICACIÓN.

El software de base esta formado por los programas que sirven de enlace entres los programas escritos por un programador, con el fin de realizar un determinado trabajo y los elementos hardware de la computadora.

El software base esta formado fundamentalmente por los siguientes elementos:

- *Traductores: intérpretes y compiladores.*
- *El ensamblador.*
- *Los programas de utilidad.*
- *Sistema operativo.*

Siendo el sistema operativo el elemento principal del software de base. Por otra parte, el software de aplicación es todo el conjunto de programas escritos para resolver problemas específicos planteados por el usuario.

2.2.3 TRADUCTORES: INTÉRPRETES Y COMPILADORES.

La computadora solo puede ejecutar instrucciones escritas en un lenguaje formado por secuencias de ceros y unos (código binario), al que normalmente se denomina lenguaje maquina. Para ello cualquier lenguaje de programación que no sea lenguaje maquina necesitara un proceso de traducción. Este proceso se lleva a cabo en los intérpretes y los compiladores.

Cualquier lenguaje de programación se puede traducir mediante u interprete construido expresamente para este lenguaje, así que no existe un interprete único para todos los lenguajes. El trabajo de un interprete es de traducir las instrucciones del lenguaje de programación al lenguaje maquina, con el fin de que la computadora pueda ejecutar esas instrucciones.

Los compiladores traducen las sentencias o instrucciones del lenguaje de programación y las convierten en un conjunto de instrucciones en lenguaje maquina directamente ejecutables por la computadora. El proceso de traducción del compilador no se realiza simultáneamente, como lo hace el intérprete, sino que se hace un proceso aparte.

2.3 FUNDAMENTOS DE INTÉRPRETES.

2.3.1 INTERPRETE.

Un intérprete es un programa que analiza y ejecuta simultáneamente un programa escrito en un lenguaje fuente.

En la Figura 4 se presenta el esquema general de un intérprete visto como una *caja negra.* Cualquier intérprete tiene dos entradas: un programa P escrito en un lenguaje fuente LF junto con los datos de entrada; a partir de dichas entradas, mediante un proceso de interpretación va produciendo unos resultados.

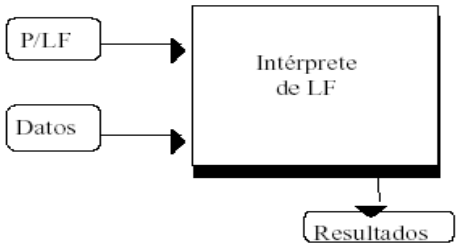


Figura 4. Esquema general de un intérprete.

Los compiladores a diferencia de los interpretes, transforman el programa a un programa equivalente en un código objeto a esto s ele conoce como fase de compilación, y en un segundo paso generan los resultados a partir de los datos de entrada a esto se le conoce como fase de ejecución tal como se muestra en la Figura 5.

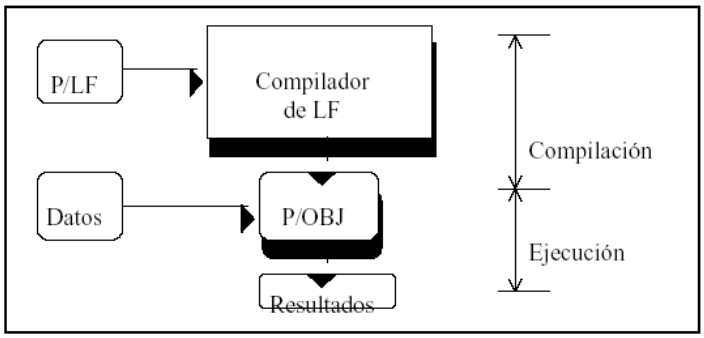


Figura 5. Esquema general de un compilador

2.3.2 ESTRUCTURA DE UN INTÉRPRETE.

A la hora de construir un intérprete es conveniente utilizar una Representación Interna (RI) del lenguaje fuente a analizar. De esta forma, la organización interna de la mayoría de los intérpretes se descompone en los módulos, ver figura 6:

1 Traductor a Representación Interna: Toma como entrada el código del programa *P* en Lenguaje Fuente, lo analiza y lo transforma a la representación interna correspondiente a dicho programa *P*.

2 Representación Interna (P/RI): La representación interna debe ser consistente con el programa original. Entre los tipos de representación interna, los árboles sintácticos son los más utilizados y, si las características del lenguaje lo permiten, pueden utilizarse estructuras de pila para una mayor eficiencia.

3 Tabla de símbolos**:** Durante el proceso de traducción, es conveniente ir creando una tabla con información relativa a los símbolos que aparecen. La información a almacenar en dicha tabla de símbolos depende de la complejidad del lenguaje fuente.

Se pueden almacenar etiquetas para instrucciones de salto, información sobre identificadores (nombre, tipo, línea en la que aparecen, etc.) o cualquier otro tipo de información que se necesite en la etapa de evaluación.

4 Evaluador de Representación Interna: A partir de la Representación Interna anterior y de los datos de entrada, se llevan a cabo las acciones indicadas para obtener los resultados. Durante el proceso de evaluación es necesario contemplar la aparición de errores.

5 Tratamiento de errores**:** Durante el proceso de evaluación pueden aparecer diversos errores como desbordamiento de la pila, divisiones por cero, etc. que el intérprete debe contemplar.

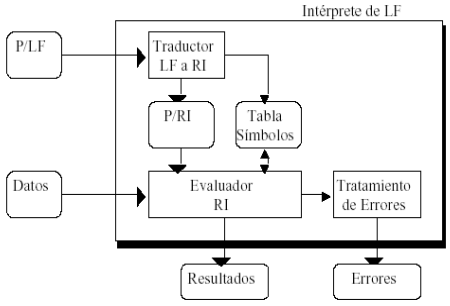


Figura 6. Organización interna de un intérprete.

Dependiendo de la complejidad del código a analizar, el intérprete puede contener módulos similares a los de un compilador tradicional: Análisis léxico, Sintáctico y Semántico. Durante la evaluación, el intérprete interactúa con los recursos del sistema como la memoria, discos, etc. Muchos sistemas interpretados liberan al programador del manejo explícito de memoria mediante técnicas de recolección de basura.

A la hora de evaluar la representación interna, existen dos métodos fundamentales: la interpretación iterativa y la interpretación recursiva.

a) Interpretación Iterativa

La interpretación iterativa es apropiada para lenguajes sencillos, donde se analiza y ejecuta cada expresión de forma directa, como podrían ser los códigos de máquinas abstractas o lenguajes de sentencias simples. La interpretación consiste en un ciclo básico de búsqueda, análisis y ejecución de instrucciones.

El esquema sería:

Inicializar
REPETIR
    Buscar siguiente Instrucción *i*
    SI encontrada ENTONCES
        Analizar *i*
        Ejecutar *i*
HASTA (que no haya más instrucciones)

Cada instrucción se busca en el almacenamiento (memoria o disco) o, en algunos casos, es introducida directamente por el usuario.

Luego la instrucción es analizada en sus componentes y ejecutada. Normalmente, el lenguaje fuente contiene varios tipos de instrucciones, de forma que la ejecución se descompone en varios casos, uno por cada tipo de instrucción.

b) Interpretación Recursiva

Comúnmente, el diseño de nuevos lenguajes de programación se realiza en dos fases:

Una primera fase de especificación semántica mediante la construcción de un intérprete prototipo que actúa como una especificación ejecutable y una segunda fase de implementación del compilador de dicho lenguaje.

Para la construcción de prototipos suele utilizarse un modelo de interpretación recursiva donde las sentencias pueden estar compuestas de otras sentencias y la ejecución de una sentencia puede lanzar la ejecución de otras sentencias de forma recursiva.

Los intérpretes recursivos no son apropiados para aplicaciones prácticas debido a su ineficiencia y se utilizan únicamente como prototipo ejecutable del lenguaje.

El problema de especificar un lenguaje mediante un intérprete prototipo es decidir en qué lenguaje se implementa dicho intérprete. Dicho lenguaje debe ser suficientemente expresivo y no ambiguo para definir claramente cómo funcionan las diferentes construcciones. En muchos casos se opta por utilizar lenguajes ya implementados pero que carecen de una especificación semántica clara. La tendencia actual es investigar técnicas de especificación semántica formal que permitan generar automáticamente este tipo de intérpretes.

2.3.3 VENTAJAS DE LA UTILIZACIÓN DE INTÉRPRETES.

En general, la utilización de compiladores permite construir programas más eficientes que los correspondientes interpretados. Esto es debido a que durante la ejecución de código compilado no es necesario realizar complejos análisis; un buen compilador es capaz de detectar errores y optimizar el código generado.

Los interpretes, por definición realizan la fase de análisis y ejecución a la vez, lo cual imposibilita las optimizaciones. Por lo tanto los sistemas interpretados suelen ser menos eficientes que los compilados. No obstante, con los nuevos avances informáticos aumentan la velocidad de procesamiento y capacidad de memoria de los computadores. Actualmente, la eficiencia es un problema menos grave y muchas veces se prefieren sistemas que permitan un desarrollo rápido de aplicaciones que cumplan fielmente la tarea encomendada.

2.3.4 VENTAJAS DE INTERPRETAR FRENTE A COMPILAR.

A continuación se enumeran algunas ventajas de los sistemas interpretados:

- Los intérpretes son más sencillos de implementar. Esto facilita el estudio de la corrección del intérprete.
- Proporcionan una mayor flexibilidad. Esto permite modificar y ampliar características del lenguaje fuente.
- No es necesario contener en memoria todo el código fuente. Esto permite la utilización en sistemas de poca memoria o en entornos de red.
- Facilitan la meta-propagación. Un programa puede manipular su propio código fuente a medida que se ejecuta.
- Aumentan la portabilidad del lenguaje. Para que el lenguaje interpretado funcione en otra maquina solo es necesario que su interprete funcione en dicha maquina.
- Facilitan el desarrollo rápido de prototipos. Ya que no existen etapas intermedias de compilación.
- Potencian la utilización de sistemas interactivos y facilitan las tareas de depuración.

2.3.5 APLICACIONES DE SISTEMAS BASADOS EN INTÉRPRETES.

Los sistemas interpretados han tenido una gran importancia desde la aparición de los primeros computadores. En la actualidad la evolución del hardware abre nuevas posibilidades a los sistemas interpretados. La preocupación ya no es tanto la eficiencia como la capacidad de desarrollo rápido de nuevas aplicaciones. Las que se podrían resumir como principales aplicaciones son las siguientes:

- Intérpretes de Comandos: los sistemas operativos cuentan con los intérpretes de comandos como el Korn-shell, C-Shell, JCL, etc. Estos intérpretes toman un lenguaje fuente que puede incluir sentencias de control y ejecutan los diferentes comandos a medida que aparecen en el lenguaje.

- Lenguajes basados en Escritos: diseñados como herramientas que sirvan de enlace entre diferentes sistemas o aplicaciones. Suelen ser interpretados con el fin de admitir una mayor flexibilidad a la hora de afrontar cada sistema.
- Entornos de Programación: existen ciertos lenguajes que contienen características que impiden su compilación o no es efectiva. Estos lenguajes suelen tener disponer de un complejo entorno de desarrollo interactivo con facilidades para la depuración de programas.
- Lenguajes de Propósito Específico: ciertos lenguajes incluyen sentencias que realizan tareas complejas en contextos específicos.
- Sistemas de Tiempo Real: entornos que permiten modificar el código de una aplicación en tiempo de ejecución de forma interactiva.
- Intérprete de código Intermedio: una tendencia tradicional de compiladores es la generación de código intermedio para una maquina abstracta; el siguiente paso puede ser la generación de código objeto a partir del código intermedio para una maquina concreta, finalizando el proceso de compilación o interpretar dicho código intermedio en una maquina concreta.

2.3.6 TIPOS DE INTÉRPRETES.

Existen diferentes métodos de interpretación según la estructura interna del intérprete. Es conveniente observar que algunos métodos podrían considerarse híbridos ya que mezclan los procesos de compilación e interpretación.

Intérpretes Puros: Son los que analizan y ejecutan sentencia por sentencia todo el programa fuente.

Siguen el modelo de interpretación iterativa y por tanto se utilizan principalmente para lenguajes sencillos. Los intérpretes puros se han utilizado desde la primera generación de computadoras al permitir la ejecución de largos programas en computadores de memoria reducida, ya que solo debían contener en memoria el intérprete y la sentencia a analizar y ejecutar en cada momento. El principal problema de este intérprete es que si a mitad del programa fuente se producen errores, se debe volver a comenzar el proceso. La Figura 7 muestra el esquema de un intérprete puro.

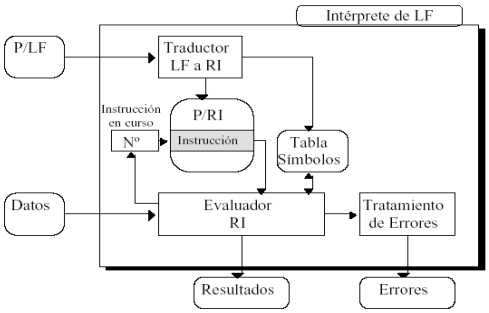


Figura 7. Esquema de un intérprete puro

Intérpretes Avanzados o Normales: Incorporan un paso previo de análisis de todo el programa fuente. Generando posteriormente un lenguaje intermedio que es ejecutado por ellos mismos. De esta forma en caso de errores sintácticos no pasan de la fase de análisis; se utilizan para lenguajes más avanzados que los intérpretes puros, ya que permiten realizar un análisis mas detallado del programa fuente.

Intérpretes Incrementales: Existen ciertos tipos de lenguajes que por sus características, no se pueden compilar directamente. La razón es que pueden manejar objetos funciones que no son conocidos en tiempo de compilación, ya que se crean dinámicamente en tiempo de ejecución. Con el propósito de obtener mayor eficiencia que la interpretación simple, se diseñan los compiladores incrementales. La idea es compilar aquellas partes estáticas del programa en lenguaje fuente, marcando como dinámicas las que no puedan compilarse.

Posteriormente en tiempo de ejecución, el sistema podrá compilar algunas partes dinámicas o recompilar partes dinámicas que hayan sido modificadas. Estos sistemas no producen un código objeto independiente, sino que acompañan al sistema que permite compilar módulos en tiempo de ejecución. Generalmente los compiladores incrementales se utilizan en sistemas interactivos donde conviven módulos compilados con módulos modificables.

Evaluadores Parciales: La utilización de evaluadores parciales o especializadores surge al considerar que muchos programas contienen dos tipos de datos de entrada. Existen una serie de datos de entrada que son diferentes en cada ejecución mientras que otros datos no varían de una ejecución a otra.

El primer conjunto se conoce como datos de entrada dinámicos, mientras que el segundo conjunto serian los datos de entrada estáticos. La principal ventaja de la evaluación parcial es la eficiencia.

Compiladores "Just in Time": Con la aparición de Internet surge la necesidad de distribuir programas de una forma independiente de la maquina permitiendo su ejecución en una amplia variedad de plataformas. Las ventajas de la interpretación Just in Time son:

- Los Programas grandes contienen porciones de código que no son ejecutadas en una ejecución típica del programa.
- Los sistemas tradicionales realizan la compilación de todo el código antes de la ejecución, lo que para el usuario puede presentar un lapso de tiempo substancial entre el momento en que todas las unidades de compilación han sido transmitidas y el momento en que la ejecución puede comenzar.

Compilación Continua: surge como un intento de mejorar la compilación "Just in time". El sistema mezcla el proceso de compilación a código nativo con el proceso de interpretación. Para lograr esto el sistema dispone de dos módulos: un modulo de interpretación de los códigos de bytes y otro modulo de compilación de códigos de bytes a código nativo. La idea consiste en que ambos módulos actúen a la vez a modo que el sistema no se detenga a compilar un modulo, sino que vaya interpretando hasta que le compilador haya generado el código nativo.

Estos son los módulos que intervienen en este tipo de compilación:

- CODIGO: Contiene una mezcla de código fuente y código nativo del programa. Inicialmente todo el código esta sin compilar, pero a medida que el programa es ejecutado, el compilador genera traducciones a código nativo de las unidades de compilación.

- COMPILADOR: El modulo compilador traduce las unidades de compilación a código nativo. A medida que se finaliza la traducción de una unidad, la versión en código nativa se deja disponible al intérprete.

- INTERPRETE: El modulo interprete se responsabiliza de la ejecución actual del programa. comienza interpretando del código fuente, haciendo saltos a las versiones en código nativo a medida que estas están disponibles.

- MONITOR: Se encarga de coordinar la comunicación entre los dos módulos anteriores.

La principal ventaja de la compilación continua respecto a la compilación Just in time radica en no tener que esperar a compilar una unidad para comenzar su ejecución.

2.4 COMPILADORES.

Un compilador es un programa que, a su vez, traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente. Usualmente el segundo lenguaje es código maquina, pero también puede ser simplemente texto. Este proceso de traducción se conoce como compilación la Figura 8 muestra el proceso de compilación.

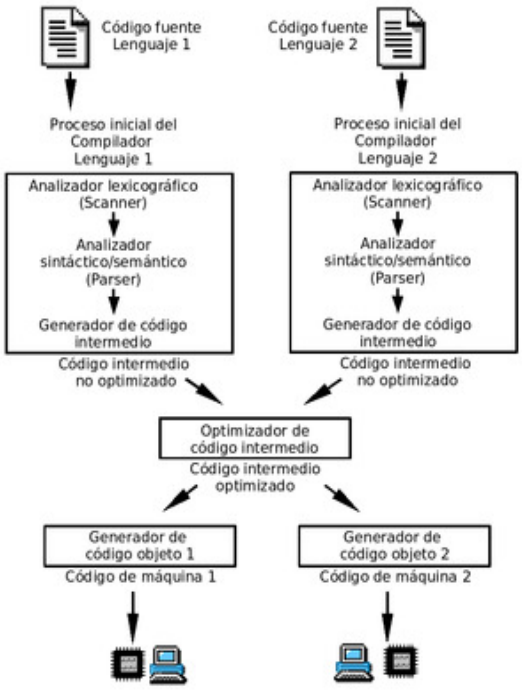


Figura 8. Esquema del proceso de compilación.

La razón principal para querer usar un compilador es querer traducir un programa de un lenguaje de alto nivel, a otro lenguaje de nivel inferior. De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a como piensa el humano, para luego compilarlo a un programa más manejable por una computadora.

En pocas palabras un compilador es cualquier programa que toma como entrada un texto escrito en un lenguaje, llamado fuente y da como salida otro texto en un lenguaje denominado objeto tal como lo muestra la Figura 9.

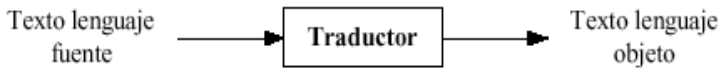


Figura 9. Esquema de la función de un compilador.

2.4.1 HISTORIA DE LOS COMPILADORES.

La palabra compilador se atribuye a Grace Murray Hopper quien visualizo la implementación de un lenguaje de alto nivel, de forma bastante precisa para su época. Los primeros compiladores fueron escritos a fines de los años 50. Se dice que FORTRAN fue el primer lenguaje compilado que tomo 18 años desarrollarlo, esto sucedió ya que se desarrollaba el lenguaje al mismo tiempo que se desarrollaba; el compilador fue diseñado para obtener el mejor código posible.

2.4.2 PARTES DE UN COMPILADOR.

Normalmente los compiladores están divididos en dos partes:

- *FRONT END:* Es la parte analiza el código fuente, comprueba su validez, genera el árbol de derivación y rellena los valores de las tabla de símbolos. Esta parte suele ser independiente de la plataforma o sistema para el cual se vaya a compilar.

- *BACK END:* Es la parte que genera el código maquina, especifico de una plataforma, a partir de los resultados de la fase de análisis, realizada por el Front End.

Esta división permite que el mismo Back End se utilice para generar el código maquina de varios lenguajes de programación distintos y que el mismo Front End que sirve para analizar el código fuente de un lenguaje de programación concreto sirva para la generación de código maquina en varias plataformas distintas. El código que genera el Back End normalmente no se puede ejecutar directamente, sino que necesita ser enlazado por un programa enlazador.

2.4.3 TIPOS DE COMPILADORES.

Compiladores Optimizadores: Realizan cambios en el código para mejorar su eficiencia, pero la funcionalidad del programa original.

Compiladores de una Sola Pasada: Generan el código maquina a partir de una única lectura del código fuente.

Compiladores de Varias Pasadas: Necesitan leer el código fuente varias veces antes de poder producir el código maquina.

Compiladores Just In Time: Forman parte de un intérprete y compilan partes del código según se necesitan.

Compiladores Incrementales: Generan un código objeto instrucción por instrucción, cuando el usuario teclea cada orden individual. Los demás tipos de compiladores requieren que todos los enunciados o instrucciones se compilen conjuntamente.

Ensamblador: El lenguaje fuente es lenguaje ensamblador y posee una estructura sencilla.

Compilador Cruzado: se genera código en lenguaje objeto para una maquina diferente de la que se esta utilizando para compilar.

Compilador con Montador: Compila distintos módulos de forma independiente y después es capaz de enlazarlos.

Autocompilador: Esta escrito en el mismo lenguaje que va a compilar. Evidentemente nos e puede ejecutar la primera vez, sirve para hacer aplicaciones al lenguaje, mejorar el código generado.

Metacompilador: es sinónimo de compilador de compiladores y se refiere a un programa que recibe como entrada las especificaciones del lenguaje para el que se desea obtener un compilador y genera como salida el compilador para ese lenguaje. El desarrollo de los metacompiladores se encuentra con la dificultad de unir la generación de código con la parte de análisis.

Descompilador: Es un programa que acepta como entrada código maquina y lo traduce a un lenguaje de alto nivel, realizando el proceso inverso a la compilación.

2.4.4 FASES DE LA COMPILACIÓN.

Es el proceso de traducción de programas fuentes a programas objeto. El programa obtenido de la compilación ha sido traducido normalmente a lenguaje maquina.

Para conseguir el programa maquina real de debe utilizar un programa llamado montador o enlazador (linker). El proceso de montaje conduce a un programa en lenguaje maquina directamente ejecutable. El proceso de ejecución de un programa escrito en un lenguaje de programación y mediante un compilador tiene los siguientes pasos:

- Escritura del programa fuente con un editor y guardarlo en un dispositivo de almacenamiento.
- Introducir el programa fuente en memoria.
- Compilar el programa con el compilador.
- Verificar y corregir errores de compilación.
- Obtención del programa objeto.
- El enlazador obtiene el programa ejecutable.
- Se ejecuta el programa y si no existen errores, se tendrá la salida del programa.

2.4.5 FORMA DE EXAMINAR DE UN COMPILADOR.

En la compilación hay dos partes, análisis y síntesis. La parte del análisis divide al programa fuente en sus elementos componentes y crea una representación intermedia. De las dos partes, la síntesis es la que requiere la técnica mas especializada.

Durante el análisis se determina las operaciones que implica el programa fuente y se registra en una estructura jerárquica llamada árbol. Se usa una clase especial de árbol llamado árbol sintáctico, donde cada nodo representa una operación y los hijos de un nodo son los argumentos de la operación.

2.4.6 ANÁLISIS DE UN CÓDIGO FUENTE.

Para que el compilador pueda analizar el código fuente es necesario que realice tres procesos de análisis los cuales son: análisis Léxico, análisis Léxico y análisis Semántico.

- análisis Léxico: también es conocido como scanner, lee los caracteres uno a uno desde la entrada y va tomando grupos de caracteres con alguna relación entre si a los cuales se les denomina Tokens, que constituirán la entrada para la siguiente etapa del compilador. Cada Token es una secuencia de caracteres que son tratados como una única entidad.

- análisis sintáctico: también es llamado como Parser, recibe como entrada los tokens que le pasa el Analizador Léxico y comprueba si esos tokens van llegando en el orden correcto. La salida teórica de la fase de análisis seria un árbol sintáctico.

- análisis Semántico: Es posterior al sintáctico y mucho mas difícil de formalizar que este. Se trata de determinar el tipo de resultados intermedios, comprobar que los argumentos que tiene un operador pertenecen al conjunto de los operadores posibles, y si son compatibles entre si. Es decir se comprobara que el significado de lo que se va leyendo es valido.

2.4.7 VENTAJAS DE COMPILAR FRENTE A INTERPRETAR.

- Se compila una vez, y se puede ejecutar múltiples veces.
- En bucles, la compilación genera código equivalente al bucle, pero interpretándolo se traduce tantas veces una línea como veces se repite el bucle.
- El compilador tiene una visión global del programa, por lo que la información de mensajes de error es más detallada.

2.5 COMPILADORES E INTÉRPRETES EN GENERAL.

Un compilador traduce un programa escrito en lenguaje fuente a un lenguaje objeto de bajo nivel, el cual puede ser lenguaje ensamblador o lenguaje de maquina de una computadora particular. El programa que se escribe en el lenguaje fuente es llamado programa fuente, el cual se edita en uno o más archivos fuentes. El compilador traduce cada archivo fuente a un archivo objeto, si el archivo fuente contiene lenguaje ensamblador se debe correr un ensamblador para convertirlos a lenguaje maquina, luego se corre un lenguaje de utilidad llamado linker (vinculador) para combinar los archivos objeto a un programa objeto. Una vez creado, se puede cargar en la memoria de una computadora y ejecutarse. Un compilador puede también traducir primero el programa fuente a un código intermedio y luego traducir el código intermedio en un lenguaje objeto.

Por otro lado, un intérprete no produce un programa objeto. Puede traducir un programa fuente a un código intermedio interno que se puede ejecutar más eficientemente, o podría simplemente ejecutar las sentencias del programa directamente. Esto quiere decir que el intérprete traduce un programa a las acciones especificadas por el programa.

2.5.1 DIFERENCIAS ENTRE COMPILADORES E INTÉRPRETES.

Básicamente lo que el compilador hace para ejecutar el programa es revisar completamente el código del programa para determinar si no hay ningún error de sintaxis, si el compilador no encontró ningún error entonces ejecuta el programa, en caso contrario lo que hará es finalizar la compilación e indicar que ha encontrado errores en el código y no ejecutara el programa.

Por otro lado el interprete verifica una línea del código y ejecuta esa línea si se llegara a encontrar con un error el programa se termina en ese instante y se debe comenzar a ejecutarlo desde el principio una vez mas.

2.5.2 VENTAJAS Y DESVENTAJAS DE LOS COMPILADORES E INTÉRPRETES.

- Al ejecutar un programa fuente con un intérprete, simplemente se carga el programa fuente al intérprete, y este revisa y ejecuta el programa. Un compilador revisa el programa fuente y produce un programa objeto. Después de correr el compilador, podría ser necesario correr un linker para luego cargar el programa objeto en la memoria para poder ejecutarlo; si el compilador generara un programa objeto en lenguaje ensamblador, también se debe correr un ensamblador. Así que un intérprete definitivamente tiene ventajas ante un compilador cunado se trata sobre el esfuerzo requerido para ejecutar el programa fuente.

- Los intérpretes pueden ser más versátiles que los compiladores. Teniendo en mente que los intérpretes son programas por si mismos, y como cualquier otro programa pueden ser creados para ejecutarse en diferentes computadoras. Un compilador sin embargo genera programas objetos para una computadora en particular.

- Uno de los casos que podría pasar es que el programa fuente contenga un error lógico que no aparece hasta el tiempo de ejecución; como un interprete tiene el control cuando esta ejecutando el programa fuente, puede parar e indicar el numero de línea de la sentencia que contiene el error y el nombre de la variable; incluso puede dar alguna ayuda de cómo corregir el error antes de reanudar con la ejecución.

  El programa objeto generado por un compilador, por un lado, usualmente se ejecuta el mismo. Información del programa fuente tales como números de línea y nombres de variables, podrían no ser presentes en el programa objeto.

Cuando ocurre un error en tiempo de ejecución, el programa simplemente termina y tal vez imprima un mensaje conteniendo la dirección de la instrucción con error, depende del programador averiguar que sentencia pertenece a la dirección de el error y que variable esta errónea.

- Cuando se trata de depurar, un intérprete es generalmente la mejor opción. Sin embargo ahora muchos ambientes de desarrollo de programas modernos tienen compiladores con la capacidad de depurar casi tan bien como los interpretes. Se compila un programa y luego se ejecuta bajo el control del ambiente; si un error ocurre en tiempo de ejecución, es mostrada la información y control necesarios para corregir el error, luego se puede reanudar la ejecución del programa, o compilar y correr luego. Tales compiladores no obstante generan información extra o instrucciones en el programa objeto para mantener el ambiente informado de la situación actual de la ejecución; esto usualmente hace que el programa objeto se menos eficiente de lo que podría ser.

- Después de haber depurado el programa la interrogante será la velocidad con que el programa correrá. Teniendo en cuenta que el interprete ejecuta las instrucciones del código fuente línea por línea, cada vez que ejecute una línea de instrucción la revisa para determinar que operación se le ordena realizar. Con un compilador, la computadora ejecuta un programa de lenguaje maquina, generado ya sea directamente por un compilador o directamente con un ensamblador. Desde que una computadora ejecuta el lenguaje maquina a una alta velocidad así como un programa puede correr desde 10 a 100 veces más rápido que un programa fuente interpretado. De este modo un programa compilado es mas rápido que un programa interpretado, esto es verdad en el caso de un compilador que genere especialmente código eficiente.

2.6 ALGORITMOS.

El programador no es más que una persona que resuelve problemas, pero para que este programador llegue a ser eficaz debe resolver los problemas de una forma rigurosa y sistemática.

Un algoritmo es un método para resolver un problema, la resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto tal como lo muestra la siguiente figura que muestra la resolución de un problema.

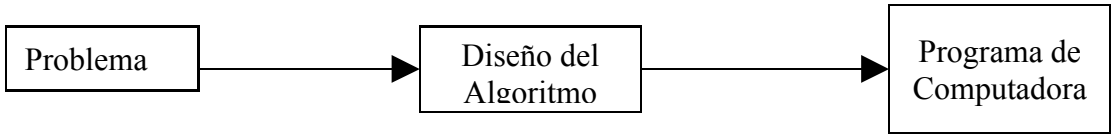


Figura 10. Pasos para la resolución de un problema

Pasos Para la resolución de un Problema:

1. Diseño del algoritmo, que describe la secuencia ordenada de los pasos que conducen a la solución de un problema dado. A esto se le denomina Análisis del problema y Desarrollo del algoritmo.
2. Expresar el algoritmo como un programa en un lenguaje de programación adecuado.
3. Ejecución y validación del programa por la computadora.

Para llegar a la realización de un programa es necesaria la realización previa de un algoritmo. Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora en que se ejecuten.

En cada problema el algoritmo se puede expresar en un lenguaje de programación diferente y ejecutarse en una computadora distinta. No obstante el algoritmo será el mismo.

En la computación los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan solo un medio para expresar el algoritmo y una computadora es solo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin. La definición de un algoritmo debe describir tres partes: Entrada, Proceso y Salida.

2.6.1 CARACTERÍSTICAS DE LOS ALGORITMOS.

- Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.

- Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces se debe obtener el mismo resultado las dos veces.
- Un algoritmo debe ser finito.

Las técnicas de desarrollo y diseño de programas que se utilizan en la programación convencional tienen inconvenientes, sobre todo a la hora de verificar y modificar dicho programa. Hoy en día están adquiriendo gran importancia las técnicas de programación, cuyo objetivo principal es de facilitar la comprensión del programa y además permiten de forma rápida, las ampliaciones y modificaciones que surjan en la fase del ciclo de vida de un sistema.

En la programación convencional de suele hacer uso indiscriminado sin control de las instrucciones de salto condicional e incondicional, esto produce cierta complejidad en el entendimiento y modificación de los programas. Eliminar esta complejidad es una característica de la programación estructurada y por eso se ha definido como la técnica de programación sin saltos condicionales e incondicionales.

De una forma general los Diagramas de flujos de datos son graficas dirigidas en donde los nodos especifican las actividades del proceso y los arcos la transferencia de datos entre los nodos. Un diagrama de flujos de datos representa el flujo de datos entre estatutos individuales o entre bloques de estatutos dentro de una rutina; flujo de datos entre rutinas secuenciales, flujos de datos entre procesos concurrentes o flujos de datos entre sistemas de computo distribuidos, donde cada nodo representa una unidad de proceso geográficamente separada. Distinto a otros diagramas de flujo, las burbujas no indican la lógica de decisión o las condiciones bajos las cuales varios nodos de proceso se activen.

Los diagramas de datos pueden expresarse utilizando una notación informal o por medio de símbolos especiales para denotar a los nodos de proceso a los nodos de entrada. Los diagramas generales especifican los procesos de un sistema en forma funcional, cada diagrama describe la entrada, los pasos del proceso y la salida para la función en cuestión, un diagrama general puede indicar la localización de los diagramas de detalles subordinados necesarios.

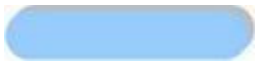
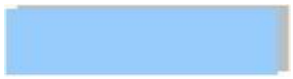
2.7 DIAGRAMAS DE FLUJO DE DATOS (DFD).

Los diagramas de flujo representan la forma más tradicional para especificar y documentar los detalles algorítmicos de un producto de programación.

Los diagramas de flujo son una representación grafica de los pasos de un proceso, útil para determinar como funciona realmente el proceso para producir un resultado; representan la forma más tradicional para especificar los detalles algorítmicos de un proceso. El resultado puede ser un producto, un servicio, información o una combinación de los tres. Se utilizan principalmente en programación, economía y procesos industriales.

Estos diagramas utilizan una serie de símbolos que representan las funciones de los programas, y de líneas de flujo, que denotan la secuencia en la que se deberán realizar estas funciones. Las instrucciones se escriben dentro de los símbolos (ver figura 11). Son modelos tecnológicos utilizados para comprender los rudimentos de la programación lineal.

En los diagramas de flujos hay literalmente docenas de símbolos orientados a utilizarse para la representación de los pasos del los diagramas de flujo en los cuales los más comunes se encuentran:

Símbolo	Descripción
	**Inicio / Terminación.** Este símbolo se utiliza para señalar el comienzo así como el final de un diagrama. Tradicionalmente se colocan las palabras “INICIO” ó “FIN” dentro de la figura para hacerlo más explícito.
	**Proceso de datos.** Este símbolo se utiliza para señalar operaciones matemáticas, aritméticas o procesos específicos que se realicen con nuestros datos.
	**Decisión.** Este símbolo representa una disyuntiva lógica o decisión. En su interior se anota una instrucción o pregunta que pueda ser evaluada como cierta o falsa y que determine el flujo del programa.
	**Desplegado de información.** Este símbolo se utiliza para mostrar un resultado, el cual puede representar la solución al problema que se pretende resolver y que fue conseguida a través del resto del diagrama.

	Entrada Manual de Datos. Sirve también para la introducción de datos desde el teclado.
	Entrada/Salida. Entrada de datos introducidos por el teclado muchas veces este símbolo es utilizado también como salida de datos

Tabla 1.Simbología de diagrama de flujo de datos

En la diagramación, también contamos con una serie de símbolos auxi[illegible] no intervienen en el proceso del algoritmo, pero que pueden ser útiles para ayudarnos a dar claridad a nuestros diagramas, algunos de ellos son los siguientes:

Símbolo	Descripción
	Conector. Este símbolo se utiliza para indicar un salto dentro del diagrama. Se utiliza con el propósito de facilitar la disposición plana de un diagrama y evitar el cruce excesivo de líneas a través del mismo.
B	**Conector de página.** Este conector es idéntico en funcionamiento que el anterior, pero su forma pentagonal lo distingue e indica que debemos buscar el "gemelo" en una página distinta de la actual. Este conector lleva asociado una especie de salto entre páginas.

Tabla 2.Conectores de diagramas de flujo de datos

Cabe mencionar que no se debe abusar del uso de conectores pues de lo contrario comenzaría a perderse la claridad que se pretende alcanzar con el diagrama.

Ejemplo de un diagrama de flujo simple:

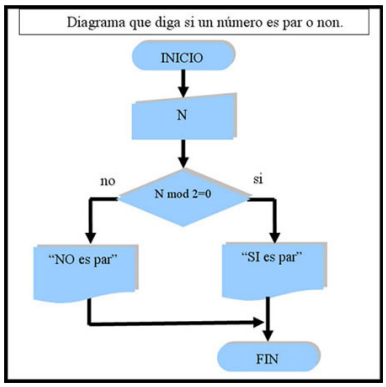


Figura 11. Ejemplo de un Diagrama de Flujo de Datos.

Los diagramas de flujo se leen de arriba hacia abajo, como lo indican las líneas de flujo. La única variación que se hace a esta lectura vertical es la efectuada es al instante en el que se encuentra con una decisión, lo que podría hacer un cambio en la secuencia.

Para funciones de entrada/salida, por lo general un símbolo representa una sola instrucción. Sin embargo, para funciones de procesamiento podemos combinar una o más instrucciones de procesamiento estrechamente relacionadas dentro de un solo símbolo de procesamiento. Este método de representación permite que los diagramas de flujo sean más sencillos y más cortos, así como también facilita la lectura del diagrama de flujo para quien quiera comprender la función que se va a realizar.

El enunciado dentro de cada símbolo no tiene que ser único, se pueden utilizar muchos otros, siempre y cuando el significado preciso sea claro.

Otra definición del diagrama de flujo es la siguiente: “Es un esquema para representar gráficamente un algoritmo. Se basan en la utilización de diversos símbolos para representar operaciones específicas. Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación. Para hacer comprensibles los diagramas a todas las personas, los símbolos se someten a una normalización; es decir, se hicieron símbolos casi universales, ya que, en un principio cada usuario podría tener sus propios símbolos para representar sus procesos en forma de Diagrama de Flujo.

Esto trajo como consecuencia que sólo aquel que conocía sus símbolos, los podía interpretar. La simbología utilizada para la elaboración de diagramas de flujo es variable y debe ajustarse a un patrón definido previamente.”

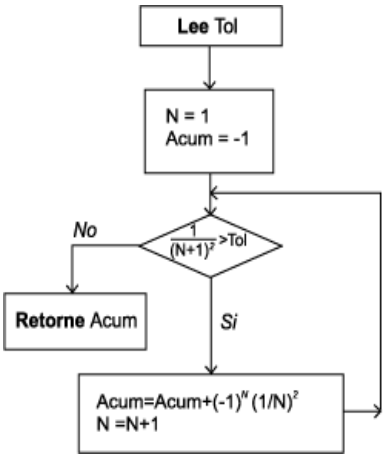


Figura 12. Ejemplo de diagrama de flujo.

2.7.1 REGLAS PARA LA CREACIÓN DE DIAGRAMAS.

1. Los Diagramas de flujo deben escribirse de arriba hacia abajo, y/o de izquierda a derecha.
2. Los símbolos se unen con líneas, las cuales tienen en la punta una flecha que indica la dirección que fluye la información procesos, se deben de utilizar solamente líneas de flujo horizontal o verticales (nunca diagonales).
3. Se debe evitar el cruce de líneas, para lo cual se quisiera separar el flujo del diagrama a un sitio distinto, se pudiera realizar utilizando los conectores. Se debe tener en cuenta que solo se vana utilizar conectores cuando sea estrictamente necesario.
4. No deben quedar líneas de flujo sin conectar
5. Todo texto escrito dentro de un símbolo debe ser legible, preciso, evitando el uso de muchas palabras.
6. Todos los símbolos pueden tener más de una línea de entrada, a excepción del símbolo final.
7. Solo los símbolos de decisión pueden y deben tener más de una línea de flujo de salida.

Los diagramas de flujo describen los procesos que cambian o transforman los datos en un sistema, las entidades externas que son fuentes o destinos de los datos y los almacenamientos o depósitos de datos a los cuales tiene acceso el sistema.

En síntesis, el diagrama de flujos de datos describe:

- Los lugares de origen y destino de los datos (los limites del sistema).
- Las transformaciones a las que son sometidos los datos (los procesos internos).
- Los lugares en los que se almacenan los datos dentro del sistema.
- Los canales por donde circulan los datos.

Con lo anteriormente mencionado se puede sostener que antes de codificar un programa, se debe dibujar un diagrama de flujo que exhiba la secuencia de las etapas de programación que se deberán codificar. La elaboración de diagramas de flujo consiste en comprender el flujo lógico de los procesamientos que se pueden realizar en la computadora. La razón de que el programador elabore diagramas de flujo es para asegurarse de que las etapas que se codificaran en el programa se integraran adecuadamente.

2.7.2 NACIMIENTO DE LOS DIAGRAMAS DE FLUJOS.

En un principio la programación estructurada fue desarrollada en sus principios por Edgar w, Dijkstra en sus *Notes and Structure Programming* y se basa en el denominado *Teorema de la Estructura* desarrollado en 1966 por Bomh y Jacopini, que se ratifico con los trabajos de Charlan D. Mills. En la actualidad existen diversas definiciones de estos diagramas pero todas ellas giran alrededor del Teorema de Estructura, técnica que se basa a través de módulos o bloques.

Para entender el Teorema de Estructura es necesario primero verificar algunos conceptos que trata este teorema.

1. *Diagrama Propio:* es aquel que posee un solo punto de entrada y uno de salida.
2. *Programa Propio:* es aquel programa que cumple con las siguientes condiciones:
    - Posee un solo inicio y un solo fin.
    - Todo elemento del programa es accesible, es decir, existe al menos

un camino desde el inicio al fin que pasa a través de el.

- No posee bucles infinitos.

3. *Teorema de la Estructura:* todo programa propio realice el programa que realice, tiene al menos un programa propio equivalente que solo utiliza las estructuras básicas de la programación que son:
    - La secuencia
    - La selección
    - La repetición

2.7.3 CARACTERÍSTICAS DE LOS DIAGRAMAS ESTRUCTURADOS.

Generalmente en las empresas se dispone de distintas personas para darle mantenimiento a un sistema y estas personas muchas veces son constantemente trasladas a otros puesto, es por eso que la importancia de realizar Diagramas de Flujo del sistema lo mas detallado y explicado posible es muy vital, así que no es necesario que el programador del sistema este asesorando todo el tiempo a cada una de las personas que tendrán a cargo dicho sistema.

El diseñador de la aplicación debe tener en cuenta todo el tiempo que no solo el tendrá acceso a los diagramas de flujo de dicha aplicación es por eso que los debe desarrollar los mas fácil posible en comprensión, para que a la otra persona le sea fácil la modificación y mantenimiento de la aplicación. Por tal cuestión el Diagrama Estructurado ofrece muchas ventajas para lograr estos objetivos.

Dicho esto se puede decir que los Diagramas Estructurados son:

- fácil de leer y comprender.
- fácil de codificar en una amplia gama de lenguajes y en diferentes sistemas.
- fácil de mantener.
- Eficiente, aprovechando al máximo los recursos de la computadora.

2.7.4 PRINCIPALES ESTRUCTURAS DE LOS DIAGRAMAS DE FLUJO ESTRUCTURADOS.

1. Estructura Secuencial: Es una estructura con una entrada y una salida en la cual figuran una serie de acciones cuya ejecución es lineal y en el orden en que aparecen, a la vez todas las acciones tienen una única entrada y salida.

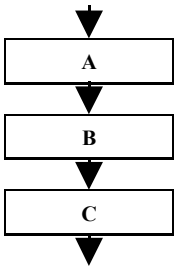


Figura 13. Estructura Secuencial

2. Estructura Alternativa: Es una estructura con una sola entrada y una sola salida en la cual se realiza una acción de entre varias, según una condición o se realiza un acción conforme a un cumplimiento o de una determinada condición. Esta condición puede ser simple o compuesta, existen dos tipos: De dos salidas en la que una de ellas puede ser la decisión nula; de tres o más salidas que también se llama múltiple.

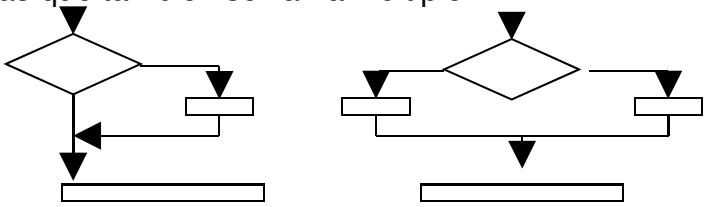
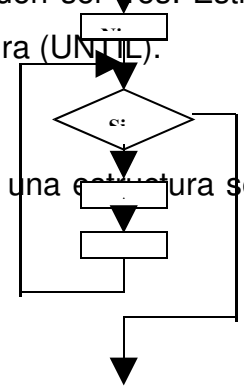


Figura 14. Estructura Alternativa

3. Estructura Repetitiva: Es una estructura con una entrada y una salida en la cual se repite una acción un numero determinado o indeterminado de veces, dependiendo en este caso del cumplimiento de una condición las estructuras repetitivas pueden ser tres: Estructura Para (FOR), Estructura Mientras (WHILE), Estructura (UNTIL).



4. Estructura Para (FOR): En una estructura se repite una acción un número

fijo de veces representado normalmente por N.

Figura 15. Estructura FOR

5. Estructura Mientras (WHILE): En esta estructura se repite una acción mientras se cumpla la condición del bucle. La característica principal de esta estructura es la de que la condición es evaluada siempre antes de cada repetición. El número de repeticiones oscila entre cero e infinito, dependiendo de la evaluación de la condición, cuyos argumentos en los casos de repetición, al menos una vez, deberán modificarse dentro del bucle, de no ser así el número de repeticiones será infinito, y se formara un bucle sin salida.

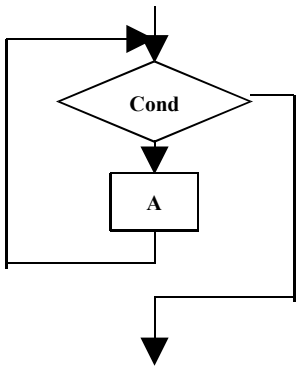


Figura 16. Estructura While.

6. Estructura Hasta (UNTIL): En esta estructura se repita una acción hasta que se cumpla la condición que controla el bucle, la cual se evalúa después de cada ejecución del mismo. El número de repeticiones oscila entre uno e infinito, dependiendo de la evaluación de la condición, cuyos argumentos en los casos de repetición, al menos dos veces, deberán modificarse dentro del bucle, de no ser así el bucle tendrá repeticiones infinitas y se formara un bucle sin salida.

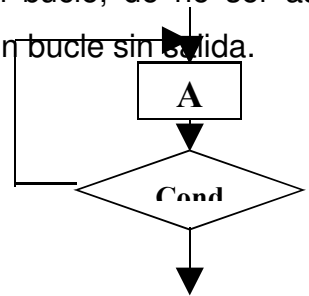


Figura 17. Estructura Until

CAPITULO III

HERRAMIENTAS A UTILIZAR

3.1 VISUAL C++ 6.0 PROFESIONAL

Visual C++ esta contenido en el paquete de Visual Studio 6.0 Profesional de Microsoft y proporciona a los desarrolladores la licencia para distribuir programas desarrollados bajo esta herramienta, y tiene la ventaja sobre la Edición básica de añadir servicios y controles para plataformas Win32, incluyendo Windows 95/98 y Windows NT. El entorno de trabajo ha sido diseñado para brindar la mayor comodidad al usuario a la hora de generar sus aplicaciones, ver Figura 18.

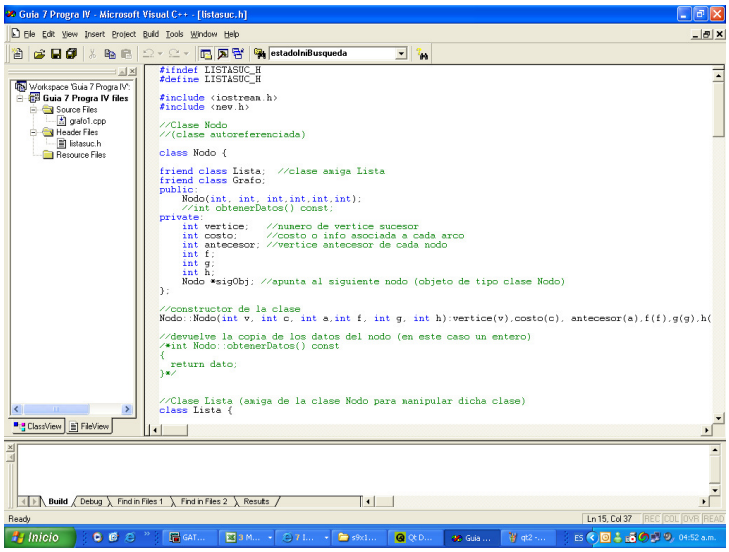


Figura 18. Entorno de trabajo en Visual C++ 6.0 Profesional.

El compilador C++ incluye todos los archivos de cabecera, bibliotecas y editores de diálogo y de recursos necesarios para crear una aplicación en Windows verdaderamente consistente, ver Figura 19. Microsoft también ha incorporado los editores de recursos para mapas de BIT, iconos, cursores, menús y cuadros de diálogo, directamente integrados en el entorno. Y hablando de integración, la nueva clase de asistentes (ClassWizards) ayuda a construir aplicaciones OLE utilizando las bibliotecas MicrosoftFoundation CLass (MFÑ)en tiempo récord.

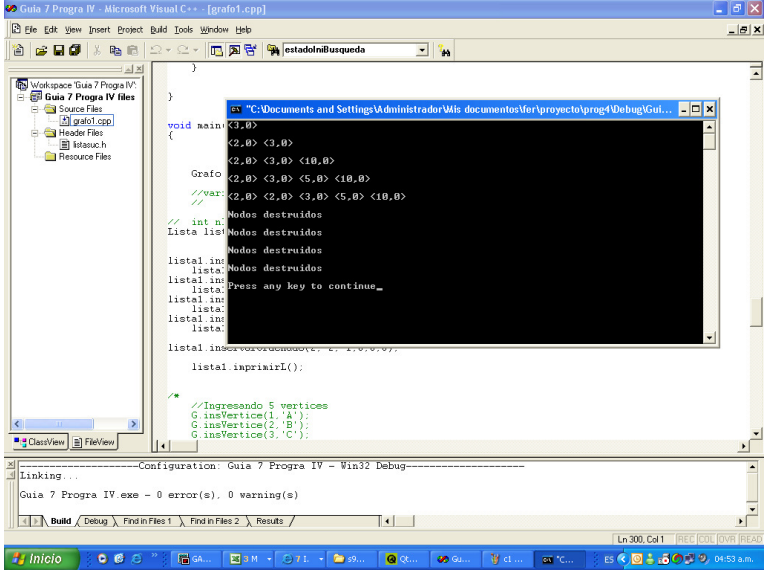


Figura 19. Ejecución de código fuente en C++ 6.0 Profesional

3.1.1 REQUERIMIENTOS DE RECURSOS.

En cuanto a requerimientos de software y de hardware se recomienda el siguiente perfil de sistema para optimizar el ciclo de desarrollo de programas en C y C++.

- PC con procesador Pentium, a velocidad de 200MHz (o superior).
- 32 MB de RAM.
- 1GB de espacio de disco fijo.
- Monitor Super VGA.
- Unidad de disco de alta densidad (3,5 pulgadas).
- Unidad de CD-ROM (para documentación en línea).
- Ratón IntelliPoint de Microsoft.

Como podemos ver los requerimientos que esta herramienta necesita para utilizar su potencial son mínimos comparados con otras herramientas de desarrollo como el JDeveloper para Java.

Otro aspecto que se valoro a la hora de elegir la herramienta de desarrollo es que visual C++ 6.0 es compatible con Qt, insertando una barra de herramientas que permite combinar ambas herramientas de manera fácil, ver Figura 20.

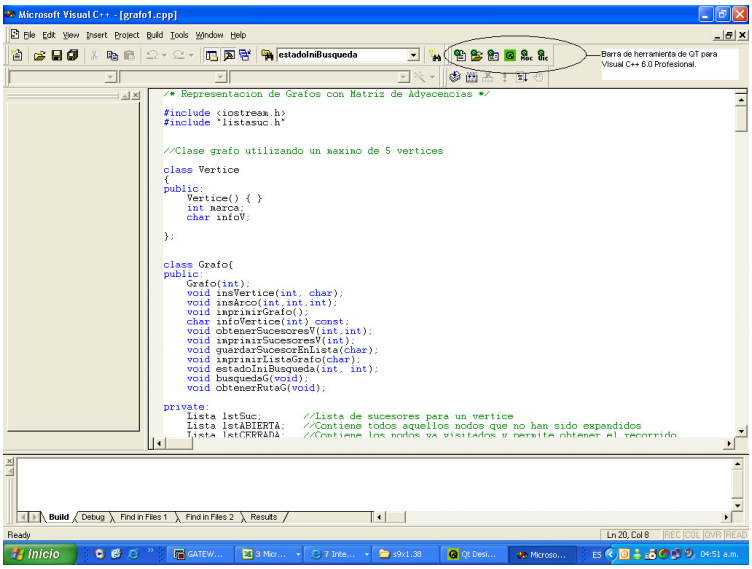


Figura 20. Barra de herramienta de integración con QT.

La desventaja que presenta esta herramienta es el alto precio que se debe pagar por obtener todo el paquete de Visual Studio 6.0 Profesional, pero luego de ver lo que adquiere por dicho precio y las facilidades que representa a la hora de desarrollar una aplicación no se ve tanto como una desventaja sino como una inversión.

3.2 QT DESIGNER.

Qt es una amplia plataforma de desarrollo que incluye clases, librerías y herramientas para la producción de aplicaciones de interfaz gráfica en C++ que pueden operar en varias plataformas. Con Qt se pueden desarrollar ricas aplicaciones gráficas, incluye soporte de nuevas tecnologías como OpenGL, XML, Bases de Datos, programación para redes, internacionalización y mucho más.

Qt dispone de una amplia gama de herramientas que facilitan la creación de formas, botones y ventanas de dialogo con el uso del ratón. Las aplicaciones creadas con Qt son muy elegantes, se ven y se operan mejor que las aplicaciones nativas. Qt dispone de tres grandes ventajas ante las librerías de ventanas rivales:

1. Qt es completamente gratuito para aplicaciones de código abierto.
2. Las herramientas, librerías y clases están disponibles para casi todas las plataformas Unix y sus derivados (como Linux, MacOS X, Solaris, etc) como también para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código y la aplicación se verá y actuará como mejor que una aplicación nativa.
3. Qt tiene una extensa librería con clases y herramientas para la creación de ricas aplicaciones. Estas librerías y clases están bien documentadas, son muy fáciles de usar y tienen una gran herencia de programación orientada a objetos lo cual hace de la programación de interfaces gráficas verdaderamente cómoda.

La nueva versión 4.2.2 permite la creación más fácil de ventanas a través de una interfaz grafica que permite dar formato de una manera más rápida que las versiones anteriores, ver Figura 21.

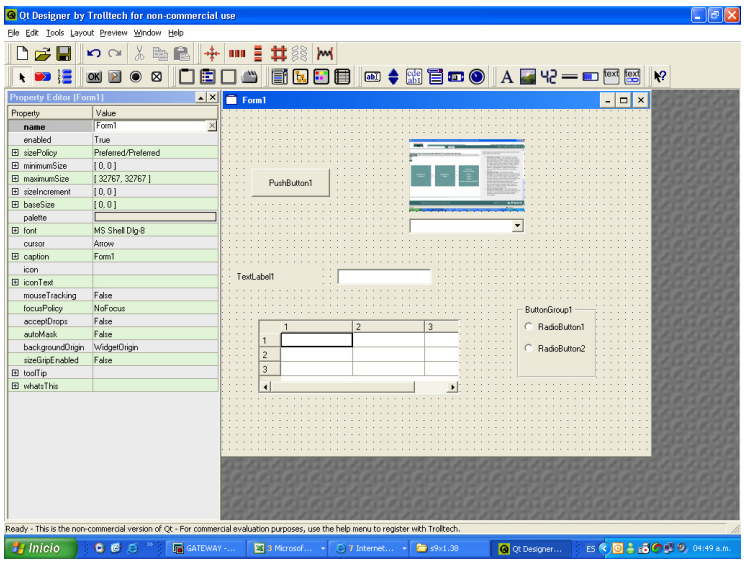


Figura 21. Entorno de trabajo del QT Designer 4.2.2

El QT Designer permite generar el código C++ de toda la interfaz visual que se ha creado para manipular los eventos desde visual C++ 6.0 lo que facilita la integración de las aplicaciones.

3.3 MODULO QCANVAS.

Para la interfaz grafica se utilizó el modulo QCanvas el cual es una clase que provee un área de gráficos en 2D sobre los cuales se pueden crear objetos llamados QCanvasItem.

Un canvas conteniendo muchos objetos es muy diferente a los widgets conteniendo muchos subwidgets por las siguientes razones:

- Los objetos son dibujados más rápidos que los widgets, en especial cuando no son rectangulares
- Los objetos utilizan menos memoria que los widgets.
- Encontrar los objetos dentro del área de dibuja es eficiente.
- Se pueden crear muchas vistas para el mismo canvas.

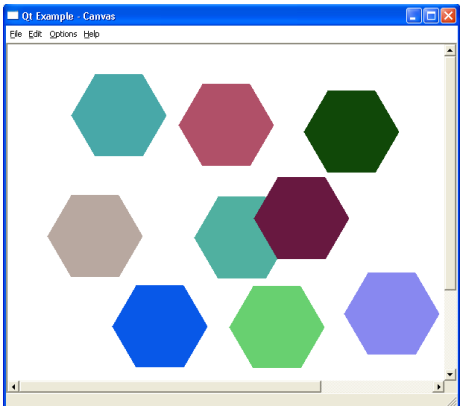


Figura 22. Ejemplos de figuras con canvas

Los objetos que se crearon para este proyecto son los siguientes:

- **Para procesos:**

  Clase principal: QCanvasRectangle.
  Clase heredada creada: c_proceso.
  Dimensiones: 150 x 50 píxeles.

- **Para entradas desde el teclado (input):**

Clase principal: QCanvasPolygon

Clase heredada creada: c_input.

Dimensiones: 150 x 50 píxeles.

- **Para muestra de datos en pantalla (ouput):**

Clase principal: QCanvasPolygon

Clase heredada creada: c_input.

Dimensiones: 190 x 50 píxeles.

- **Para bucles o lazos (ciclo For y ciclo While)**

Para ciclo for:

Clase principal: QCanvasPolygon

Clase heredada creada: c_for.

Dimensiones: 150 x 50 píxeles.

Para ciclo while:

Clase principal: QCanvasPolygon

Clase heredada creada: c_while.

Dimensiones: 150 x 50 píxeles.

- **Para decisiones (If)**

Clase principal: QCanvasPolygon

Clase heredada creada: c_if.

Dimensiones: 200 x 50 píxeles.

CAPITULO IV
LA APLICACIÓN

4.1 REQUERIMIENTOS DEL SISTEMA

El propósito de determinar los requerimientos es para tener una delimitación de la aplicación. Es la primera etapa en el desarrollo de un sistema, ya que en esta se especifican los datos que deben ser obtenidos del usuario y cuales deben ser procesados por la aplicación, así como determinar algunos requisitos que el usuario necesita para ejecutar la aplicación.

4.1.1 REQUERIMIENTOS PARA LA EJECUCION DEL SISTEMA

La aplicación necesita ejecutarse sobre una estación que posea las siguientes aplicaciones:

SOFTWARE

- **Sistema Operativo Microsoft Windows XP Professional**

Windows XP proporciona un nivel de estabilidad que permite la administración de manera eficiente los recursos del sistema, por lo que mantiene el sistema funcionando tan rápido como sea posible aún si se ejecutan múltiples aplicaciones simultáneamente.

Para responder a fallos inesperados se cuenta con herramientas que permiten regresar el sistema a fechas previas en las que este se ejecutaba eficientemente. La aplicación puede correr en sistemas operativos mas recientes que Windows XP Professional.

- **qt-mt230nc.dll**

Es un archivo .dll que permite que Windows pueda correr aplicaciones elaboradas con la herramienta Qt-designer.

HARDWARE

Aquí se detallan los requerimientos mínimos para que la aplicación pueda correr en la estación de trabajo:

- 64 MB de memoria RAM.
- 700 Mhz de procesador ya sea Intel Celeron o superiores.
- 20 MB de espacio libre en disco duro.

4.1.2 REQUERIMIENTOS DEL USUARIO

Para el funcionamiento adecuado del sistema se requiere que el usuario especifique la información para la creación de los objetos dentro del diagrama a medida que este los va creando. A continuación se detalla el requerimiento por cada objeto:

- Proceso: El usuario debe especificar las variables a utilizar ya sean de tipo Numérico, Carácter o Lógico. También se debe especificar las sentencias a realizar. Estas pueden ser utilizando funciones matemáticas o lógicas.

- Petición de datos: El usuario debe seleccionar de las variables ya creadas cuales solicitara un valor por medio del teclado.

- Muestra de datos: El usuario especificara el texto a mostrar. Dicho texto podrá contener el nombre de variables para mostrar su respectivo valor.

- Ciclo While: El usuario debe introducir una condición teniendo en cuenta las reglas que rigen dicha sentencia y las variables creadas.

- Ciclo For: El usuario debe introducir una condición teniendo en cuenta las reglas que rigen dicha sentencia y las variables creadas.

- Decisión If: El usuario debe introducir una condición teniendo en cuenta las reglas que rigen dicha sentencia y las variables creadas.

Luego de especificados los detalles de cada objeto el usuario deberá especificar la localización de los mismos dentro del diagrama hasta obtener el diseño deseado.

4.2 ANÁLISIS

En esta etapa se procesan los requerimientos de la fase anterior, se depuran y se reestructuran para crear el cuerpo de aplicación y así poder pasar a la etapa de diseño de la aplicación.

4.2.1 TIPO DE INTÉRPRETE

El sistema utiliza un Interprete Puro, recordando la definición de un interprete puro expone que un interprete puro es el que analiza y ejecuta sentencia por sentencia de todo el programa fuente. Para el caso del sistema, cuando se solicita ejecutar un diagrama, el intérprete hace un análisis de cada sentencia, si no se encuentra algún error entonces la sentencia se ejecuta.

Para determinar si hubo algún error en la sentencia previo a la ejecución se hace un análisis léxico, sintáctico y semántico.

4.2.2 PROCESO DE INTERPRETACION.

4.2.2.1 DESCRIPCION GENERAL

El proceso de interpretación se realiza de la siguiente forma:

- Se toma el contenido del objeto según el orden de ejecución del diagrama. Dicho contenido se toma línea por línea para su respectivo análisis.

- Se toma la línea y se le aplica el análisis léxico para identificar cada token que conforma dicha sentencia. Si no se encontró un error se prosigue con el análisis sintáctico, si se encontró se detiene el proceso de análisis y se muestra el respectivo mensaje de error.

- Se realiza el análisis sintáctico para determinar si el orden de los tokens es el adecuado dentro de la sentencia. En caso de un error se detiene el análisis y se muestra el respectivo mensaje de error.

- Se realiza análisis semántico en el cual se toma en si los operadores que se han encontrado se encuentran dentro de la tabla de símbolos y si los argumentos de las funciones son correctos de acuerdo al tipo de función y de acuerdo al tipo de dato.

- Si no se encontró ningún error, entonces se ejecuta la sentencia y se retoma la siguiente línea del objeto y se repite la fase de análisis y ejecución hasta que ya no hay más objetos que interpretar.

4.2.2.2 TABLA DE SIMBOLOS

La tabla de símbolos es la que contiene los identificadores permitidos; cualquier palabra o carácter que no este definido dentro de la tabla y que no se haya especificado como constante será motivo de error. A continuación se presenta la tabla de símbolos utilizada para el proceso de interpretación.

+	>	not	atan	carácter
-	<	++	exp	lógico
*	<=	--	log	for
/	>=	cos	ln	while
^	=	sin	sqrt	if
%	!=	tan	ceil	
(	And	acos	abs	
)	Or	asin	numérico	

Tabla 3. Tabla de símbolos

4.2.2.3 ANALISIS LEXICO

En esta etapa se descompone la línea o sentencia en palabras o tokens. Luego cada token es comparado con los existentes en la tabla de símbolos para establecer si este token es permitido o no.

Para realizar el escaneo de tokens se utiliza las funciones de manejo de caracteres que brinda la herramienta Qt, las cuales están contenidas en la librería QString.

4.2.2.4 ANALISIS SINTACTICO

En esta etapa se toman los tokens que se han identificado en el análisis léxico y se verifican si cada token tomado esta en el orden correcto dentro de la línea o sentencia.

4.2.2.5 ANALISIS SEMANTICO

En esta etapa se verifican que los argumentos de las funciones sean validos y que las operaciones entre variables sean del mismo tipo.

4.2.2.6 MANEJO DE ERRORES

Para el manejo de errores se ha creado una bandera lógica con un valor predefinido de falso. Al entrar en cada etapa del análisis esta bandera puede cambiar su valor lógico si se encuentra un error, luego se genera un mensaje de error el cual es especificado por cada etapa de análisis.

El modulo principal esta verificando el valor de dicha bandera al terminar cada etapa de análisis, si el valor lógico encontrado es falso se procede a la siguiente etapa, si es verdadero se detiene la ejecución y se genera un cuadro de dialogo en el cual se muestra el tipo error y la línea en donde se encontró el error.

4.2.3 FUNCIONES Y OPERADORES SOPORTADOS POR LA APLICACIÓN

OPERADORES DE ASOCIATIVIDAD:

- *Paréntesis ():* Los paréntesis modifican la secuencia de evaluación de una expresión.

OPERADORES MATEMATICOS:

- *Suma +:* Operador que indicara que se realizara una suma entre variables
- *Resta -:* Operador que indica que se realizara la operación de resta entre variables.
- *Exponenciación ^:* Operador que indicara que el valor de una variable será elevado a un exponente.
- *Multiplicación *:* Operador que indica una multiplicación entre los valores de variables.

- *División /:* Operador que indica la operación aritmética de división entre los valores de las variables.
- *Modulo %:* Operador que indica el residuo después de la división entre valores.

OPERADORES LOGICOS RELACIONALES:

- *Mayor que >:* Operador que permite evaluar el mayor valor entre dos expresiones.
- *Menor que <:* Operador que permite evaluar el menor valor entre dos expresiones.
- *Menor o igual a <=:* Operador que permite evaluar si el valor de una expresión es menor o igual a otra expresión.
- *Mayor o igual a >=:* Operador que permite evaluar si el valor de una expresión es mayor o igual a otra expresión.
- *Igual a =:* Operador que permite evaluar si dos expresiones poseen el mismo valor.
- *Diferente a !=:* Operador que permite evaluara si el valor de una expresión es distinto al valor de otra expresión.

OPERADORES LOGICOS BOOLEANOS:

- *AND (Y lógico):* Función que devuelve resultado verdadero si ambas expresiones son verdaderas.
- *OR (Ó lógico):* Función que devuelve resultado verdadero si una de las expresiones es verdadera.
- *NOT (Negación Lógica):* Función que invierte el valor de una expresión.

OPERADORES DE INCREMENTO Y DECREMENTO:

- *Incremento (expresión ++):* Función que incrementa una expresión en valor de uno en uno.
- *Decremento (expresión --):* Función que decrementa una expresión en un valor de uno en uno.

FUNCIONES TRIGONOMETRICAS:

- *COS(expresión):* Función que devuelve el coseno de una expresión
- *SIN(expresión):* Función que devuelve el seno de una expresión
- *TAN(expresión):* Función que devuelve la tangente de una expresión
- *ACOS(expresión):* Función que devuelve el coseno inverso de una expresión
- *ASIN(expresión):* Función que devuelve el seno inverso de una expresión
- *ATAN(expresión):* Función que devuelve la tangente inversa de una expresión.

FUNCIONES EXPONENCIALES Y LOGARITMICAS:

- *EXP(expresión):* Función que calcula el exponencial de una expresión.
- *LOG(expresión):* Función que calcula el logaritmo base 10 de una expresión.
- *LN(expresión):* Función que calcula el logaritmo natural de una expresión.

FUNCIONES DE RAICES:

- *SQRT(expresión):* Función que permite evaluar la raíz cuadrada de una expresión.

FUNCIONES DE ABSOLUTO Y REDONDEO:

- *CEIL(expresión):* Función que permite redondear el valor de una expresión
- *ABS(expresión):* Función que permite evaluar el valor absoluto de una expresión.

4.3 DISEÑO DE LA APLICACION

En esta etapa se modela el sistema y se le da forma para que soporte los requisitos. Se generan los módulos a utilizar para que la aplicación sea capaz de satisfacer las expectativas y requerimientos determinados anteriormente.

4.3.1 MÓDULOS A UTILIZAR

4.3.1.1 MÓDULO DE MANEJO GRAFICO

En este modulo se ha creado todo el motor grafico el cual consiste en un modulo CanvasView que es el que maneja todos los eventos del Mouse que el usuario realiza y el modulo Canvas el cual permite todo el manejo de los objetos dentro del área de trabajo.

4.3.1.2 MÓDULO DE ANALISIS

Este modulo comprende todo el análisis previo a la ejecución del diagrama, el cual esta conformado por el análisis léxico, sintáctico y semántico de cada sentencia. En este modulo se da el manejo de errores previo a la ejecución.

4.3.1.3 MÓDULO DE EJECUCION

Este modulo es llamado luego que el análisis de la sentencia se ha completado sin errores, en este se tiene el manejo en memoria de las distintas variables creadas por el usuario, así como su respectivo valor a medida que se avanza en la ejecución del diagrama. Se tiene un manejo de los errores en tiempo de ejecución como son los desbordamientos de memoria provocados por operaciones matemáticas.

4.3.1.4 MÓDULO DE GENERACION DE CODIGO

Es modulo comprende el proceso de creación de un fichero conteniendo todos los elementos del diagrama en lenguaje de alto nivel C++ o Java. Este modulo solo puede ser utilizado si y solo si la ejecución de un diagrama se ha terminado sin problemas.

4.3.1.5 MÓDULO DE SUBPROGRAMA

Este modulo contiene toda la metodología para crear y guardar todo el contenido de los subprogramas y de la manera en que se pasan los valores del diagrama principal al subprograma.

4.3.2 Clases a utilizar

4.3.2.1 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.

4.3.2.2 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	Oculta 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.

4.3.2.3 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 outuput
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 una 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 esta 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.

4.3.2.4 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.

4.3.2.5 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.

4.3.2.6 Clase análisis.

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

Variables:

Tipo	Nombre	Descripcion
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.

4.3.2.7 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.

4.3.3 ESTRUCTURA DE DATOS A UTILIZAR

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.

4.3.3.1 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.

4.3.3.2 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.

4.4 IMPLEMENTACIÓN.

Esta es la etapa en la cual los módulos creados en el diseño se vuelven código fuente y se genera un ejecutable.

4.4.1 DESCRIPCION DE LA APLICACION

4.4.1.1 CREACION DEL DIAGRAMA

El sistema permite la inserción de las siguientes instrucciones:

- Procesos
- Muestra de Datos
- Petición de Datos

- Ciclo For
- Ciclo While
- Decisión If

Cada una de estas instrucciones se representa con sus respectivos símbolos dentro del diagrama. La aplicación permite que el usuario pueda mover o arrastrar los símbolos a la posición que le sea más conveniente. Además la aplicación presenta la posibilidad de eliminar algún objeto dentro del diagrama.

Una vez creado el diagrama a las necesidades de usuario este tiene la opción de guardar el diagrama o ejecutarlo.

4.4.1.2 EJECUCION DEL DIAGRAMA

Para ejecutar un diagrama se puede abrir un diagrama anteriormente creado, o crear un nuevo diagrama y ejecutarlo para permitir al intérprete analizarlo para su ejecución exitosa.

En caso de que se encuentre algún error la ejecución se detiene y se le muestra al usuario el mensaje de error respectivo.

4.4.1.3 GENERACION DE CODIGO

Una vez creado y ejecutado exitosamente el diagrama se tiene la opción de generar el código de dicho diagrama ya sea en lenguaje de alto nivel C++ o Java según la necesidad del usuario.

4.4.2 FUNCIONAMIENTO DEL SISTEMA

A continuación se explica el funcionamiento del sistema utilizando diagramas de flujos de datos.

4.4.2.1 DIAGRAMA DE INSERCION DE UN NUEVO OBJETO

A continuación se muestra el diagrama de flujo de datos para que el usuario inserte un nuevo objeto ya sea un proceso, una muestra de datos, petición de datos, ciclo while, ciclo for o decisión If.

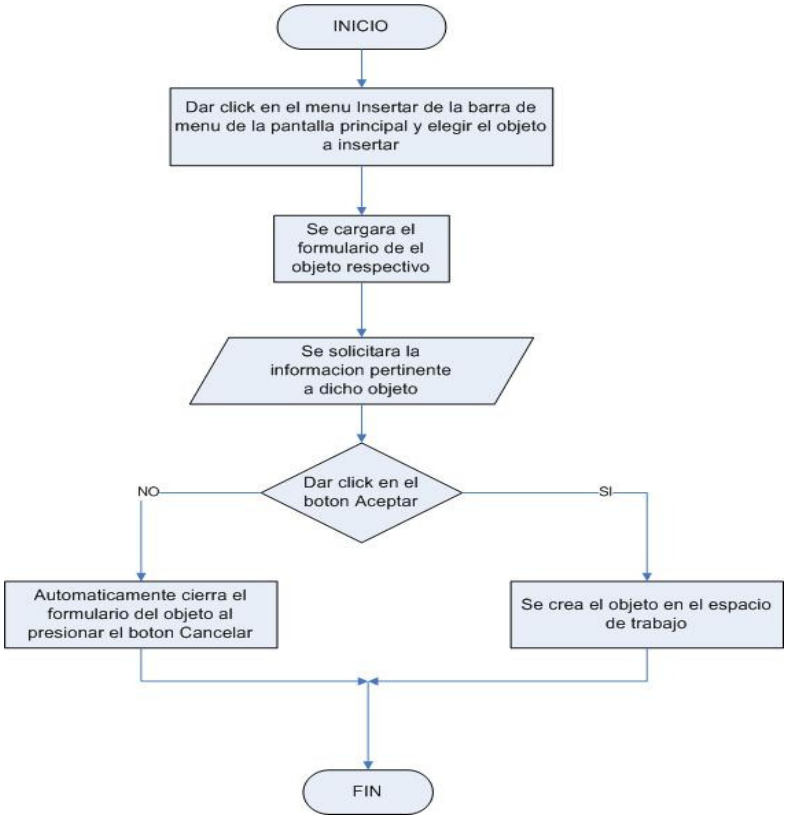


Figura 23. Diagrama de Inserción de Objetos.

4.4.2.2 DIAGRAMA DE MODIFICACION DE UN OBJETO

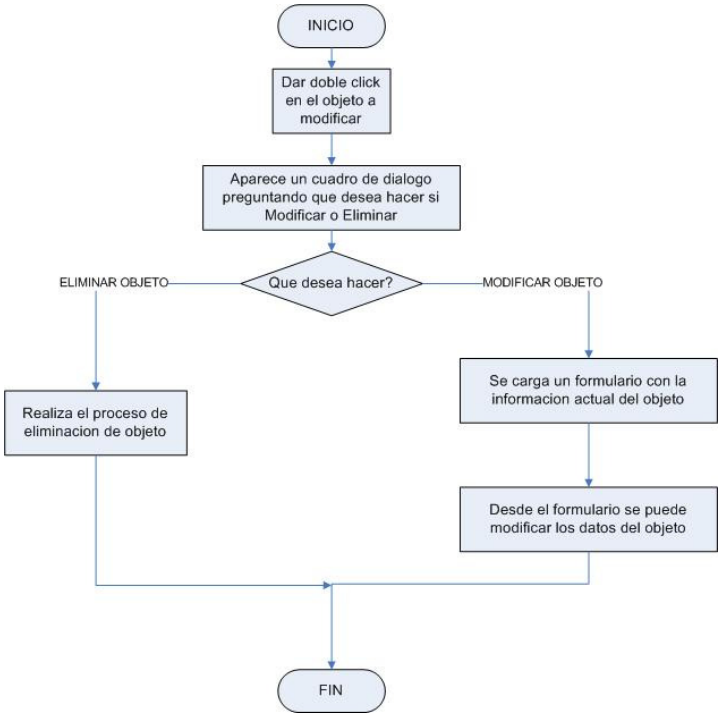


Figura 25. Diagrama de Modificación de Objetos

4.4.2.3 DIAGRAMA DE ELIMINACION DE OBJETOS

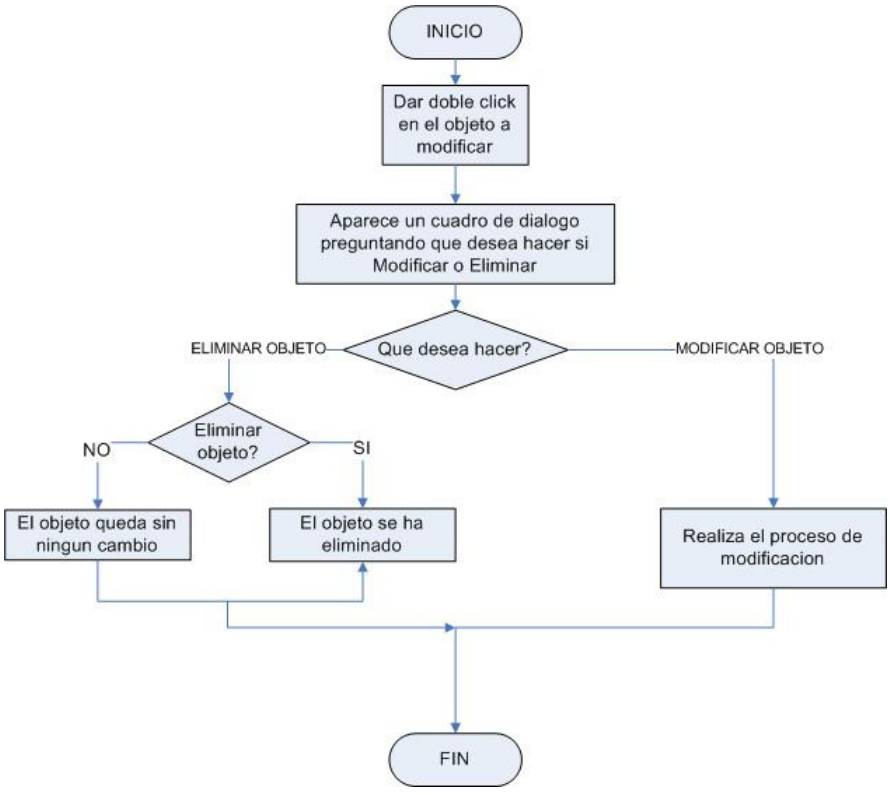


Figura 26. Diagrama de Eliminación de Objetos

4.4.2.4 DIAGRAMA DE EJECUCION DE UN DIAGRAMA

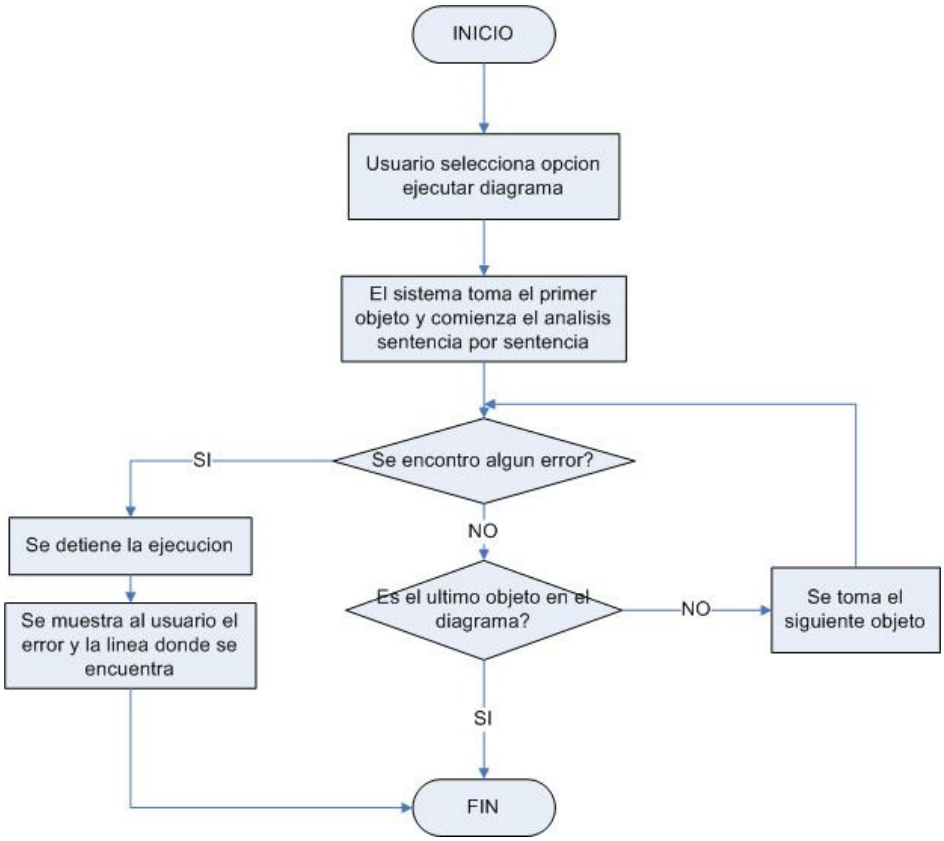


Figura 27. Diagrama de Ejecución de Diagramas

4.4.3 FORMULARIOS

4.4.3.1 PANTALLA PRINCIPAL DE LA APLICACIÓN

La pantalla principal de la aplicación consta de un área de trabajo en la que el usuario puede crear su diagrama de flujo de datos; al mismo tiempo la pantalla principal contiene una barra de menús ubicada en la parte superior de ella. Cuando se crea un diagrama y éste crece a lo ancho o hacia abajo, el espacio de trabajo esta configurado para que este crezca conforme el diagrama se va expandiendo.

En el menú Archivo se encuentran las opciones:

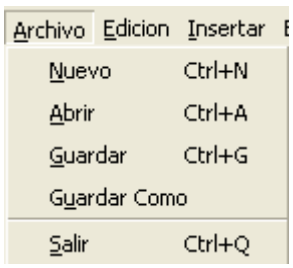


Figura 28. Menú Archivo

- ***Nuevo:*** que sirve para crear un nuevo espacio de trabajo y crear otro diagrama; también se puede solicitar este comando utilizando las teclas Ctrl.+N

- ***Abrir:*** esta opción esta diseñada para abrir cualquier diagrama que se haya creado con anterioridad, para acceder a este comando desde teclado se utiliza la combinación de teclas Ctrl + A.

- ***Guardar:*** esta opción se utiliza para guardar los datos de los diagramas creados en el espacio de trabajo, estos datos son guardados en un archivo de extensión .tcb, en donde se almacena el tipo de objeto, coordenadas en donde se encuentra en el diagrama y el contenido de el mismo, al decir contenido del mismo se refiere a el tipo de variables utilizadas, los valores de las variables, para esta opción se puede acceder desde teclado usando las teclas Ctrl + G.

- ***Salir:*** por si se desea terminar con la aplicación. Se puede acceder a este comando utilizando las teclas Ctrl + Q.

En el menú Edición se encuentra la opción:

Figura 29. Menú Edición

- ***Actualizar:*** que se utiliza para hacer un redibujamiento del diagrama con los últimos movimientos realizados. Para acceder a este comando desde este teclado se utiliza la combinación de teclas F5.

En el menú Insertar se encuentran las opciones:

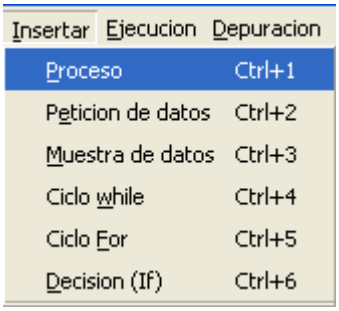


Figura 30. Menú Insertar

- ***Proceso:*** se encarga de llamar en pantalla el formulario de Procesos el cual consiste en solicitar al usuario las variables, valores iniciales y sentencia que dicho proceso realizara. Para acceder al comando también se utiliza Ctrl + 1.

- ***Petición de Datos:*** presenta en pantalla el formulario de Petición de Datos el cual consiste en solicitar al usuario por medio de su teclado los valores que dicha variable contendrá. Para acceder a este comando desde teclado se utiliza Ctrl + 2.

- ***Muestra de Datos:*** carga en la pantalla el formulario de Muestra de datos el cual consiste en solicitar al usuario algún mensaje que este quiera presentar en pantalla a la hora de ejecución. Para acceder desde el teclado se usa la combinación Ctrl + 3.

- ***Ciclo While:*** presenta en pantalla el Formulario de Ciclo While el cual consiste en la creación de un enlace que sea valido mientras se cumpla la condición determinada que el usuario tendrá que ingresar. Se utilizan las combinaciones de teclas Ctrl + 4.

- ***Ciclo For:*** carga el formulario de Ciclo For que al igual que el ciclo While representa un bucle con la diferencia de que este bucle se cumple mediante un contador determinado por el usuario. Desde teclado se accede con las teclas Ctrl + 5.

- ***Decisión IF:*** presenta el formulario de Decisiones IF, el cual permite al usuario crear una condición determinada por el usuario sea esta verdadera o falsa. Se utiliza Ctrl + 6 para acceder desde la pantalla principal.

En el menú Ejecución se encuentran las opciones:

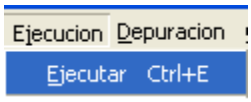


Figura 31. Menú Ejecución

- ***Ejecutar:*** ya creado el diagrama en el espacio de trabajo el usuario tiene la opción de ejecutar el diagrama esta opción es la encargada de realizar esa tarea. Para ejecutar este comando desde el teclado se hace con la combinación de teclas Ctrl + E.

En el menú de Depuración se encuentran las opciones:

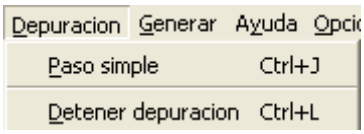


Figura 32. Menú Depuración

- ***Paso Simple:*** se utiliza para depurar el diagrama, es decir verificar que no se encuentre ningún error en tiempo de ejecución, con este comando la depuración surge paso a paso. Se puede activar por medio del teclado con la combinación de teclas Ctrl + J.

- ***Detener Depuración:*** se utiliza cuando no se desea continuar con la depuración se tiene la opción de finalizar el proceso de depuración por medio de este comando. Para utilizarlo mediante le teclado se utilizan las letras Ctrl + L.

  En el menú de Generar se encuentran las opciones:

Figura 33. Menú Generar

- ***C++:*** una vez ejecutado con éxito el diagrama se tiene la posibilidad de generar su respectivo codigo en C++ escogiendo esta opción.

- ***Java:*** Se tiene la posibilidad de generar el codigo respectivo del diagrama creado hacia el lenguaje Java.

  En el menú de Ayuda se encuentran las opciones:

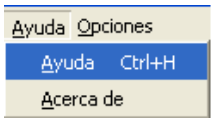


Figura 34. Menú Ayuda

- ***Ayuda:*** En este menú se puede encontrar todos los temas referidos a esta aplicación con el fin de que se aprenda a utilizar en su totalidad y de una manera sencilla. También se puede acceder a este comando utilizando las teclas Ctl + H.

- ***Acerca de:*** En esta sección se encuentra la información de la aplicación referente a los derechos de autor, versión, los nombres de los autores y una breve descripción de la aplicación.

  En el menú de Opciones se encuentra la siguiente opción:

Figura 35. Menú Opciones

- ***Doble Buffer:*** esta opción sirve para las computadoras con poca capacidad de memoria y velocidad; si la opción se encuentra señalada significa que se ha activado la opción de movimiento de los objetos de una forma rápido; es decir para computadoras de mayor capacidad de memoria y velocidad.

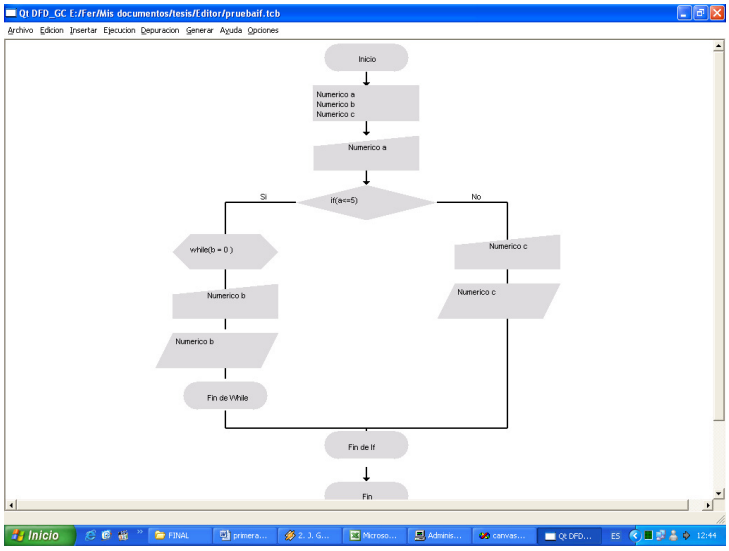


Figura 36. Pantalla principal de la aplicación.

4.4.3.2 FORMULARIO DE INSERCION DE PROCESOS

Este formulario presenta dos pestañas para la inserción de los procesos; la primera pestaña mostrada en la Figura 37 titulada “Inicialización de variables”.

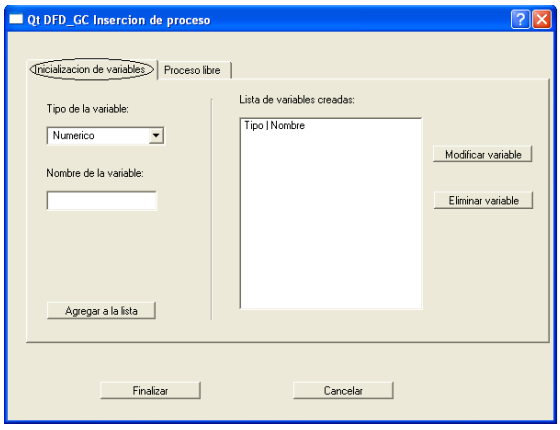


Figura 37. Formulario de inserción de Procesos (Introducción de Variables)

Sirve para crear los nombres de las variables y sus valores, los cuales serán utilizados en dicho proceso como lo muestra la siguiente imagen.

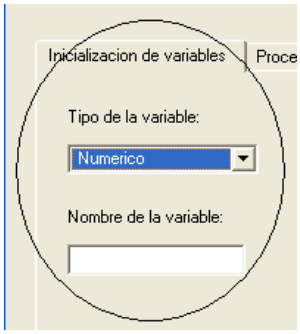


Figura 38. Tipo y nombre de variable

Para la creación de las variables se tiene la opción de elegir que tipo de variable se utilizara en las cuales se tiene de tipo carácter, numérico y lógico, así como lo indica la imagen a continuación.

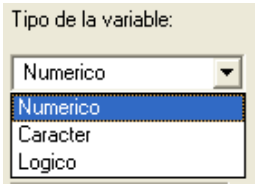


Figura 39. Tipo de variable

Luego de creada la variable se guarda en una lista de variables creadas, haciendo clic en el botón de agregar a lista; que sirve para registro por si el usuario necesita usar una de esas variables.

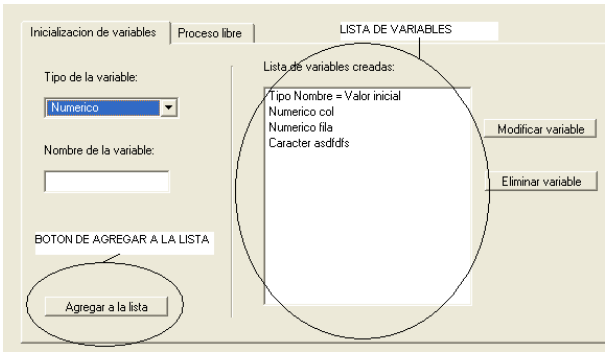


Figura 40. Lista de variables

Después de crear las variables y almacenarlas en las lista de variables creadas se tiene la opción de modificar dicha variable utilizando el botón titulado “Modificar Variable”, primero se debe seleccionar la variable a modificar de la lista luego presionar el botón de modificar para poder cambiar ya sea su nombre o tipo de variable. También se cuenta con la opción de eliminar la variable para esto basta con solo seleccionar la variable de la lista y dar clic en el botón titulado “Eliminar variable”, la siguiente imagen ilustra los dos botones en el formulario.

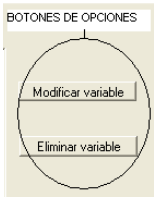


Figura 41.Botones de opciones

La segunda pestaña titulada “Proceso libre” como lo muestra la Figura 42, sirve para crear la sentencia que se realizara en el proceso.

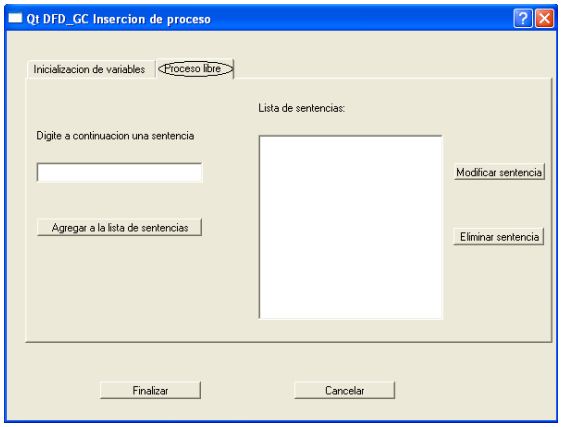


Figura 42. Formulario de Inserción de Procesos (Proceso Libre)

Una vez creada la sentencia se guarda en una lista de sentencias creadas por medio del botono titulado “Agregar a la lista de sentencias”, esto sirve para mantener un registro de las sentencias por si el usuario desea usar las mismas sentencias o modificarla. Deberán ir colocadas las variables antes creadas, al mismo tiempo la aplicación tiene la opción de modificar o eliminar la sentencia de un proceso. En la siguiente imagen se muestra las partes antes mencionadas.

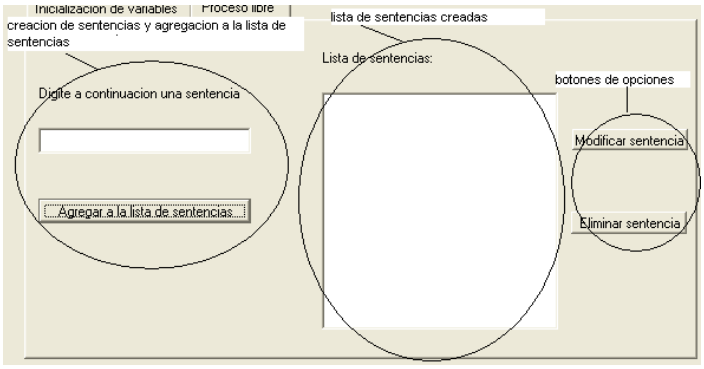


Figura 43. Creación de sentencias

Luego de haber creado las variables y la sentencias que se quiere que el proceso realice se tienen los botones de comando, cuyas acciones es de “Finalizar”, si se ha terminado de crear las variables que se deseen o “Cancelar” sino se quiere hacer ningún cambio a las variables del proceso o la creación de dicho proceso, dichos botones son mostrados en la siguiente imagen.

Figura 44. Botones de Comando Formulario Proceso

Este mismo formulario sirve para crear todas las variables que se utilizaran a lo largo del diagrama de flujo.

4.4.3.3 FORMULARIO DE PETICION DE DATOS

Este formulario se utiliza para pedir al usuario que ingresa el valor de una variable desde su teclado. En la Figura 45 se muestra el formulario de petición de datos.

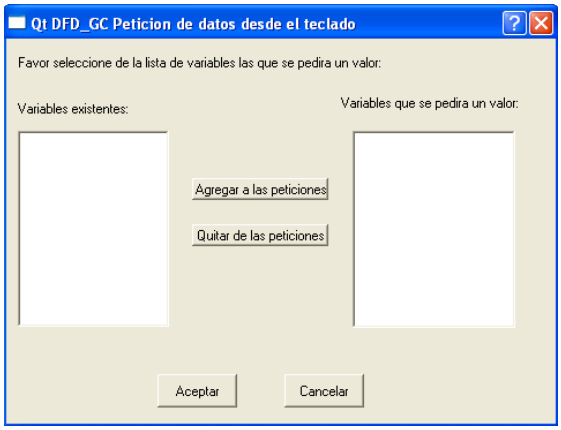


Figura 45. Formulario de Petición de Datos

Al lado izquierdo del formulario se encuentran las variables anteriormente creadas en los procesos en este caso el usuario puede optar por seleccionar una de ellas para que esta sea agregada a la lista de petición de valores, la siguiente imagen muestra la lista de variables a solicitar su valor.

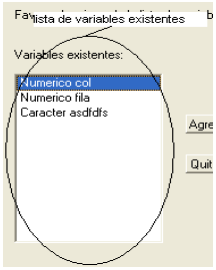


Figura 46. Variables existentes formulario petición de datos

Al lado derecho del formulario se encuentra la lista de variables que se pedirán sus valores tal y como lo muestra la siguiente imagen.

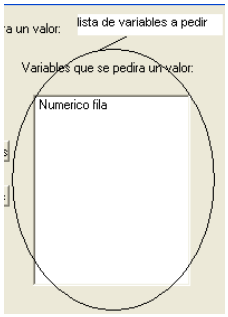


Figura 47. Lista de variables a pedir al usuario formulario petición de datos

En la parte central del formulario se encuentran los botones de opciones que permiten agregar las peticiones de variables a una lista o quitar dichas peticiones de variables, las cuales el usuario creo previamente; así como lo muestra la imagen.

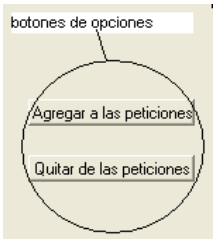


Figura 48. Botones de opciones formulario petición de datos

Luego de haber seleccionado las variables que serán solicitadas al usuario se tienen los botones de comando, cuyas acciones es de “Aceptar”, si se ha terminado de seleccionar las variables que se deseen, o “Cancelar” sino se quiere hacer ninguna petición de variables, dichos botones son mostrados en la siguiente imagen.

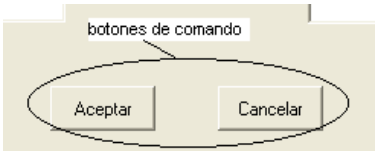


Figura 49. Botones de comando formulario petición de datos

4.4.3.4 FORMULARIO DE MUESTRA DE DATOS

El formulario para la muestra de datos se utiliza para desplegar en pantalla los valores que contengan las variables que el usuario quiera que se visualice cuando el diagrama se este ejecutando. Como se ve en la Figura 48.

El formulario al igual que el de petición de datos posee una lista de variables existentes, de esta lista se debe seleccionar la variable que se desea mostrar, dando click en el boton agregar a la muestra, para que después pasen a ser almacenadas en la lista de variables a mostrar, para luego aceptar o cancelar la instrucción.

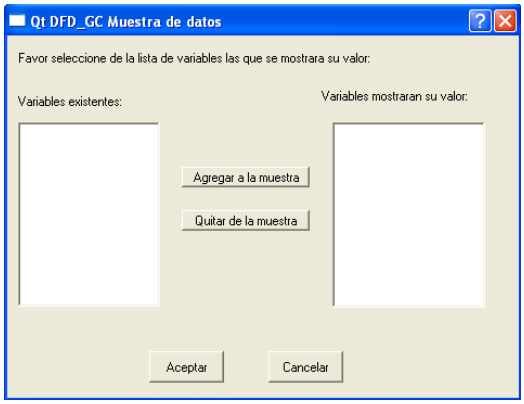


Figura 50. Formulario de muestra de datos

4.4.3.5 FORMULARIO PARA CICLO WHILE

Este formulario presenta las opciones para formar un ciclo while en el diagrama de flujos, como lo muestra la Figura 51.

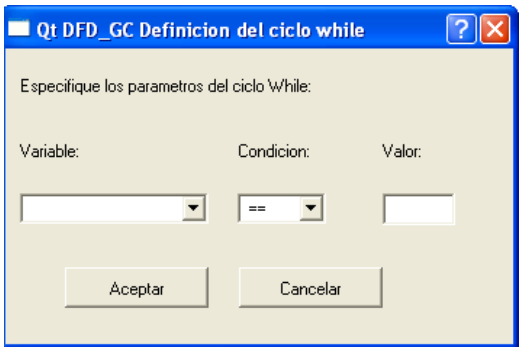


Figura 51. Formulario para ciclo While.

En este formulario se puede escoger la variable que se necesita para la condición desde una lista que contiene las variables ya creadas como lo muestra la siguiente imagen.

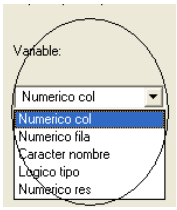


Figura 52. Selección de variable ciclo While

Luego de haber escogido la variable para la condición se debe elegir el operador para la condición; los cuales permite el operador igual (=), distinto a (!=), mayor que (>), mayor o igual que (>=), menor que (<), menor o igual que (<=) como se ilustra en la siguiente imagen.

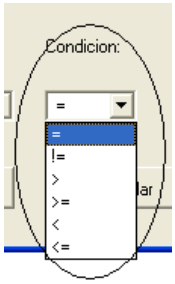


Figura 53. Condición ciclo While

Para finalizar la condición se debe determinar un valor de paro que será comparado con la variable escogida, así como en el siguiente dibujo.

Figura 54. Valor de paro ciclo While

Luego de haber creado la condición se tienen los botones de comando "Aceptar" y "Cancelar", que servirán para determinar si se quiere insertar el ciclo while con la condición determinada o cancelar la operación. El siguiente dibujo muestra los botones anteriormente mencionados.

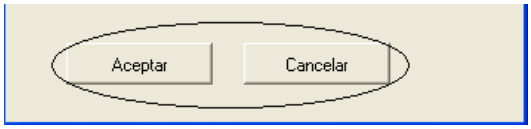


Figura 55. Botones de comando formulario ciclo While.

4.4.3.6 FORMULARIO PARA CICLO FOR

Para la sentencia For el formulario presenta tres secciones en las que el usuario debe determinar los valores y operadores que serán usados en la sentencia, La Figura 56 ilustra toda la composición de este formulario.

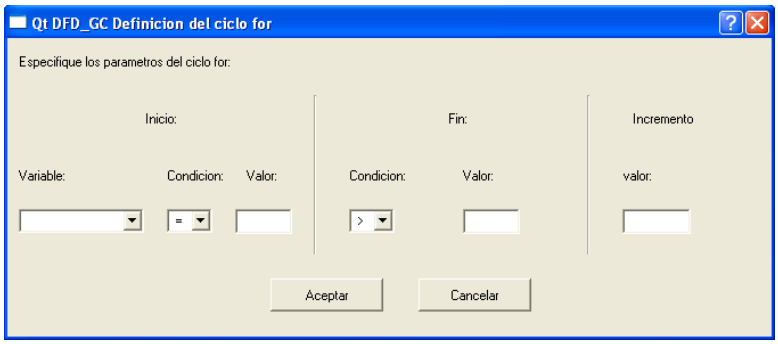


Figura 56. Formulario para ciclo For

La primera parte de la sentencia es la inicialización de la variable de control del bucle la cual se puede acceder desde una lista de variables ya creadas, se debe elegir el operador para la inicialización de la variable de control y el valor con que esta inicializara. A continuación se ilustra este proceso.

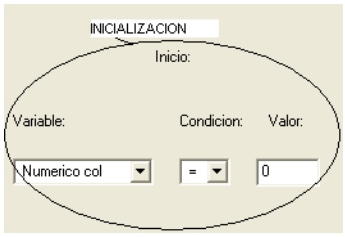


Figura 57. Inicialización ciclo For

La segunda parte de la sentencia debe ser la condición de iteración, esta expresión lógico determina si las sentencias se han de ejecutar mientras sea verdadera, esto se hace eligiendo la variable de control de la lista de variables, el operador para la condición y el valor que esta variable tendrá, así como se muestra en la siguiente imagen.

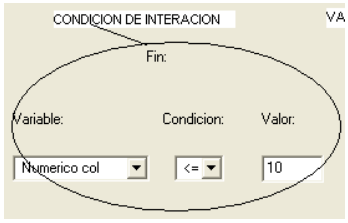


Figura 58. Condición de Iteración ciclo For

En la parte final de la sentencia For se debe colocar el valor del incremento o decremento que la variable de control tendrá, la inserción de este valor en el formulario se muestra en la siguiente imagen.

Figura 59. Valor incremento ciclo For

Luego de haberse elegido los valores que tomara la sentencia se tiene la opción de Aceptar la sentencia o Cancelar tal y como se ve en la siguiente ilustración.

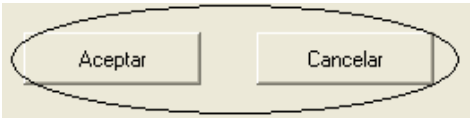


Figura 60. Botones de Comando formulario ciclo For

4.4.3.7 FORMULARIO PARA DECISION IF

Para la sentencia de condición If, el formulario presenta la opción de creación de la condición que se utilizara para evaluar; tal y como aparece en la Figura 61.

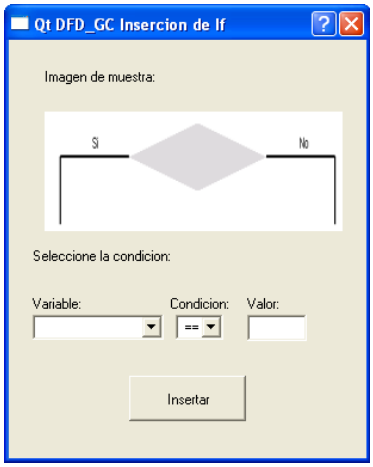


Figura 61. Formulario para decisiones IF

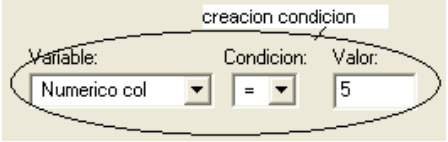


Figura 62. Creación de condición If

Al terminar de elegir las opciones y de crear la sentencia se cuenta con los botones de comando “Insertar” si se desea agregar la condición If al diagrama o “Cancelar” si se desea abortar la acción, estos botones se ilustran en la imagen siguiente.

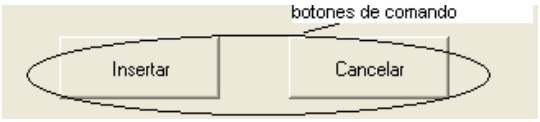


Figura 63. Botones de comando formulario sentencia If

CONCLUSIONES

Al término del desarrollo de esta herramienta informática, se pueden enumerar las siguientes conclusiones:

- La arquitectura del sistema esta compuesta por el Lenguaje de Programación C++ en ambiente Windows y la interfaz grafica mediante la librería Qt Designer, esto facilito la creación de una interfaz amigable y fácil de utilizar para cualquier tipo de usuario.

- Se efectuaron pruebas para determinar el funcionamiento de la herramienta, observando que tiene un uso de recursos de procesador más bajo que el de otras herramientas similares.

- Gracias al manejo gráfico creado para esta aplicación, facilita en gran medida la creación, edición y almacenamiento de diagramas de flujo por parte del usuario.

- Se generaron herramientas de depuración que permitan al usuario la ejecución paso a paso del diagrama creado y permitan comprender de una manera más dinámica y didáctica el funcionamiento del algoritmo.

- La aplicación permite generar la codificación para los lenguajes de alto nivel C++ y Java para que permita al usuario irse familiarizando con las sentencias y las reglar propias de cada lenguaje de programación.

- Junto con la aplicación se brinda una biblioteca de archivos ejemplos que contienen diagramas de algoritmos para ilustrar el fácil manejo de la herramienta y al mismo tiempo le permita al usuario familiarizarse con la aplicación.

- Se genero un manual de usuario para que esta herramienta pueda ser comprendida en su totalidad. Asi mismo se cuenta con una aplicación adjunta conteniendo toda la ayuda con interfaz Html para facilitar la utilización de parte del usuario.

RECOMENDACIONES

Debido a que esta aplicación se creo en una plataforma de Microsoft, se puede tomar como prototipo para que estudiantes o personas interesadas en la mejora de un sistema como este, que pueda ayudar a otros a comprender mejor los procesos de los algoritmos y pueda servir como herramienta para que otros creen diagramas de flujo.

Se recomienda la investigación para poder migrar esta aplicación a otra plataforma como Linux. También se alienta a migrar esta aplicación a una plataforma no comercial como lo es Dev C++, con el fin de que se pueda distribuir libremente y asi poder obtener la licencia de libre distribución de la GNU.

La aplicación no consta con funciones predefinidas por el usuario, por lo tanto se sugiere el seguimiento de esta aplicación para la investigación de este objetivo.

Al tratarse la primera versión de esta aplicación se hace énfasis en que puede ser mejorada, como por ejemplo el incrementar el número de funciones soportadas por la aplicación. Así como el mejoramiento del manejo gráfico y las herramientas de depuración. También se incita a implementar el modulo de subprogramas, el cual no fue posible crearlo en este prototipo. Con esto se conseguiría hacer los diagramas más compactos y familiarizarse con el uso de funciones a la hora de crear la codificación de dichos subprogramas en forma de funciones

BIBLIOGRAFÍA.

1. **González Robledo, Hugo F. "Programación de Sistemas I" Itlsp.edu.mx**

2. **Deitel, H.M. y P.J. "Cómo programar en C y C++" Editorial Prentice Hall**

3. **Joyanes Aguilar, Luis. "Programación en Algoritmos, estructuras de datos y objetos" Editorial McGraw Hill**

4. **Farina, Mario V. "Diagramas de Flujo" Editorial Diana México**

5. **Stern, Nancy B. "Diagramas de Flujo: Manual de Lógica para Computadoras" Editorial Limusa México**

6. **Mak, Ronald. "Writing Compilers and Interpreters: An Applied Approach Using C++" Editorial Wiley**

1. http://es.wikipedia.org/wiki/Diagrama_de_flujo
2. http://mx.geocities.com/ikky_fenix/ind_proy.html
3. http://es.wikipedia.org/wiki/Diagrama_de_flujo
4. http://es.wikipedia.org/wiki/Algoritmo
5. http://www.di.uniovi.es/~labra/FTP/Interpretes.pdf
6. http://www.network-press.org/?diagramas_flujo
7. http://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel
8. http://entrenate.dgsca.unam.mx/introduccion/leng_alton.html
9. http://usuarios.lycos.es/tutoinformatica/lenguajes.html
10. http://html.rincondelvago.com/lenguajes-de-bajo-nivel.html#
11. http://www.mis-algoritmos.com/aprenda-a-crear-diagramas-de-flujo
12. http://www.bsiamericas.com/Mex+Calidad/Resumen/Que+es+el+ISO9000.xalter
13. http://www.consultec.es/consultoria/implantacion_ISO9000.asp
14. http://deming.eng.clemson.edu/pub/tutorials/qctools/flowm.htm
15. "Ley de Propiedad intelectual" www.cnr.gob.sv

GLOSARIO.

A

Algoritmo: es una secuencia finita de operaciones realizables, no ambiguas, cuya ejecución da una solución de un problema en un tiempo finito.

C

Compilador: Es un programa que traduce un lenguaje de alto nivel al lenguaje máquina. Un programa compilado indica que ha sido traducido y está listo para ser ejecutado. Una vez compilado el programa, el resultado en forma de programa objeto será directamente ejecutable.

D

Diagrama de Flujo de datos (DFD): Es un esquema para representar gráficamente un algoritmo.

H

Hardware: se refiere a todos los componentes físicos (que se pueden tocar) de la computadora: discos, unidades de disco, monitor, teclado, ratón (Mouse), impresora, placas, chips y demás periféricos.

Hipervínculo: Un hipervínculo es un elemento de un documento electrónico que hace referencia a otro recurso.

Html: El HTML, acrónimo inglés de **H**yper**T**ext **M**arkup **L**anguage (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato de las paginas Web.

I

Interprete: Un intérprete traduce cada instrucción o sentencia del programa escrito a un lenguaje máquina e inmediatamente se ejecuta. Encuentran su mayor ventaja en la interacción con el usuario, al facilitar el desarrollo y puesta a punto de programas, ya que los errores son fáciles de detectar y sobre todo de corregir.

L

Lenguaje fuente: Programa en su forma original, tal y como fue escrito por el programador, el código fuente no es ejecutable directamente por el computador, debe convertirse en lenguaje de maquina mediante compiladores, ensambladores o interpretes.

Lenguaje de programación: Es un lenguaje que puede ser utilizado para controlar el comportamiento de una maquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos. Permite a un programador especificar de manera precisa sobre que datos una computadora debe operar, como deben ser almacenados y transmitidos y que acciones debe tomar bajo ciertas circunstancias.

Lenguaje maquina: Es el sistema de códigos directamente interpretable por un circuito micro programable, como el microprocesador de una computadora. Este lenguaje esta compuesto por un conjunto de instrucciones que determinan acciones a ser tomadas por la maquina.

Lenguaje de marcación: Un lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

Lenguaje de alto nivel: Se caracterizan por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en vez de a la capacidad ejecutiva de las maquinas.

Lenguaje de bajo nivel: Es el que proporciona poca o ninguna abstracción del microprocesador de una computadora.

Lenguaje ensamblador: Es un tipo de lenguaje de bajo nivel utilizado para escribir programas informáticos, y constituye la representación mas directa del código maquina.

M

Microprocesador: Es un chip que sirve como cerebro de la computadora, el cual esta compuesto de millones de transistores.

N

Navegador: Un navegador o browser es una aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores Web de todo el mundo a través de Internet.

P

Protocolo: conjunto de leyes o reglas que rigen la forma en como los dispositivos de una red se comunican entre ellos, el protocolo establece el formato en que se enviaran los datos, la sincronización, secuenciación y el manejo de errores en una red.

S

Software: Se denomina software (también programática, equipamiento lógico o soporte lógico) a todos los componentes intangibles de un ordenador o computadora, es decir, al conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica.

Sintaxis: Conjunto de reglas gramaticales que rige la escritura de programas computacionales.

W

Widget: Componente con el cual el usuario interactúa, ya sea gráfico o de control. Son ejemplo de widget las ventanas, cajas de texto, checkboxs, listbox, entre otros. Son utilizados por los programadores para hacer interfaces gráficas de usuario (GUI)

ANEXOS.

Normas ISO 9000.

La familia de normas ISO 9000 es un conjunto de normas de calidad establecidas por la *Organización Internacional para la Estandarización* (ISO), que se pueden aplicar en cualquier tipo de organización. Su implementación en las organizaciones, supone una gran cantidad de ventajas para sus empresas. Los principales beneficios son:

- Reducción de rechazos e incidencias en la producción.
- Aumento de la productividad.
- Mayor compromiso con los requisitos del cliente.
- Mejora continua.

La familia de normas apareció por primera vez en 1987 teniendo como base una norma estándar británica (BS), y se extendió principalmente a partir de su versión de 1994, estando actualmente en su versión 2000. Las normas ISO de 1994 estaban principalmente pensadas para organizaciones que realizaban proceso productivo y por tanto su implantación en empresas de servicios era muy dura y por lo tanto sigue la creencia que es un sistema burocrático.

Pasos a seguir para el registro y aprobación de patentes y derechos de autor.

Aprobación de patentes.

1. Presentar solicitud original al centro nacional de registro, a la dirección de propiedad intelectual específicamente al departamento de patentes.
2. Anexar el recibo de pago por un valor de $ 57.14 dólares.
3. Adjuntar documento técnico o llamado también memoria descriptiva, en original y dos copias, este debe contener:

Descripción de la invención es la relación o descripción de la invención de manera suficientemente clara y completa, para evaluarla y para que una persona versada en la materia técnica correspondiente pueda ejecutarla.

La descripción indicará el nombre de la invención o modelo de utilidad e incluirá la siguiente información:

i. El sector tecnológico al que se refiere o al cual se aplica;

ii. La tecnología anterior conocida por el solicitante que pueda considerarse Útil para la comprensión y el examen de la invención o modelo de utilidad Referencias a los documentos y publicaciones anteriores relativas a Dicha tecnología;

- Descripción de la invención o modelo de utilidad en términos que Permitan la comprensión del problema técnico y de la solución aportada y Exponer las ventajas de éstas con respecto a la tecnología anterior;

iii. Descripción de la mejor manera conocida por el solicitante para ejecutar o Llevar a la práctica la invención o modelo de utilidad, utilizando ejemplos y Referencias a dibujos;

La manera en que la invención o modelo de utilidad puede ser producida O utilizada en alguna actividad, salvo cuando ello resulte evidente de la

Descripción o de su naturaleza. (art. 138 de la Ley de Propiedad Intelectual)

4. Presentar Reivindicaciones,(Las reivindicaciones definirán la materia para la cual se desea protección mediante la patente. Las reivindicaciones deberán ser claras y concisas y estar totalmente sustentada por la descripción. (art.140 de la ley de propiedad intelectual.)

5. presentación de Dibujos, En el caso de los dibujos, será indispensable la presentación de éstos cuando fuere necesario para comprender, evaluar y ejecutar la invención o modelo de utilidades.

6. Resumen de la invención. dicho resumen comprenderá una síntesis de lo divulgado en la descripción y una reseña de las reivindicaciones y los dibujos que hubieran, y en su caso incluirá la fórmula química o el dibujo que mejor caracterice la invención.

7. El resumen permitirá comprender lo esencial del problema técnico y la solución aportada por la invención, así como el uso principal de las

mismas. El resumen servirá exclusivamente para fines de información técnica y no será utilizado para interpretar el alcance de la protección.

8. presentación de Poder (en caso comparezca por medio de apoderado).

En caso de ser una patente de **Diseño industrial se presentara:**

- Representación gráfica del diseño industrial(En ella se identificará el solicitante y al creador del diseño, y se indicará el tipo o género de productos a los cuales se aplicará el diseño y la clase o clases a las cuales pertenecen dichos productos de acuerdo con la clasificación, así como los demás datos indicados en las disposiciones reglamentarias correspondientes. La solicitud será acompañada de representaciones gráficas del diseño, conforme a lo dispuesto en las disposiciones reglamentarias correspondientes, y del comprobante de pago de los derechos establecidos.

(Regulado y establecido en el art. 142 de la Ley de Propiedad Intelectual)

A continuación se presenta el formulario legal para poder registrar una patente en el país.

CENTRO NACIONAL DE REGISTROS
DIRECCION DE PROPIEDAD INTELECTUAL
DEPARTAMENTO DE PATENTES DE INVENCION

SOLICITUD DE PATENTES

DE INVENCION (20 años) ☐ DE MODELO DE UTILIDAD (10 años) ☐

RESERVADO PARA OFICINA

Comprobante de pago	☐	Descripción	☐	Poder	☐
Reivindicaciones	☐	Dibujos	☐	NIT	☐
Resumen	☐	Cesión	☐		

LA SOLICITUD DEBERA DE IR ACOMPAÑADA DE LA SIGUIENTE DOCUMENTACION: DESCRIPCION (Art. 138), REIVINDICACIONES (Art.140), DIBUJOS (Art.139), Y RESUMEN (Art.141) LOS DOCUMENTOS DEBERAN PRESENTARSE EN ORIGINAL Y DOS COPIAS, MAS UNA COPIA DEL PODER O FIANZA (Art.136)

(54) Título de la Invención
o Modelo de Utilidad:

País de Origen de la Invención
o Modelo de Utilidad

(30) Reivindica Prioridad (Art. 144) SI ☐ NO ☐

(31) No. de Prioridad_____(32)Fecha de Prioridad_____(33)País de Prioridad_____
Si posee mas de una prioridad por favor especifique:

Fecha de Presentanción de otra solicitud u otro Título de Protección (Art. 136)

Oficina:_____

(71)DATOS DEL SOLICITANTE O PROPIETARIO(Art. 136)

Nombre:_____

Domicilio: _____

Profesión:_____Telefono:_____

Fax:_____

(73) DATOS DE (LOS) INVENTORES (ES) (Art.136)

Nombre:_____ Nacionalidad:_____

Domicilio:_____

Nombre:_____ Nacionalidad:_____

Domicilio:_____

(74) DATOS DEL APODERADO O MANDATARIO O REPRESENTANTE LEGAL (SI LO HUBIERA) (Art.136)

Nombre:_____ Profesión:_____

Domicilio:_____ Fax:_____

Teléfono:_____

Poder:_____ Inscrito al No:_____ Tomo:_____ Del Libro:_____

SEÑALO PARA OIR NOTIFICACIONES (Art. 137)

Dirección:_____

Telefono:_____ Fax:_____

Autorizo a_____ Para recibir documentos y notificaciones.

En consecuencia pido se me tenga por parte en el carácter en que comparezco se admita a trámite de ley mi solicitud y oportunamente se me conceda el Registro Correspondiente.

San Salvador,_____ de_____ del año_____

Firma:_____

Nota: si la persona interesada no se presenta personalmente la firma deberá ser autenticada por un notario

Aprobación de derechos de autor.

En sus requisitos se encuentra que los únicos documentos que se deben mostrar son los siguientes:

1. Solicitud en Limpio y Original
2. Dos ejemplares de la obra (de preferencia en CD)
3. Recibo de pago por $11.43
4. Otros derechos: Constancias $3.42; Certificaciones literales $5.71; mas $0.22 por cada pagina adicional después de la primera hoja.
5. si es persona jurídica presentar nombramiento vigente del representante legal, contrato de trabajo o escritura de cesión de derechos de la persona que hizo físicamente la obra. Además manifestar que su representada es la dueña de los derechos patrimoniales de la obra (en lugar de mencionar que soy el autor de la obra).

Se presenta el siguiente formulario de registro de derechos de autor.

CENTRO NACIONAL DE REGISTROS
REGISTRO DE LA PROPIEDAD INTELECTUAL
MODELO DE SOLICITUD PARA EL DEPOSITO DE OBRAS

SEÑOR REGISTRADOR DE LA PROPIEDAD INTELECTUAL:

Yo,_____, de_____años de edad,_____(profesión u oficio), del domicilio de_____,con Documento Unico de Identidad Número_____, de nacionalidad salvadoreña, a usted EXPONGO: Que soy el autor de la obra denominada_____, la cual consiste en_____(breve síntesis de cuatro renglones).

Dicha obra la divulgué en ____(lugar, fecha y editorial). Si no ha sido publicada manifestar que es inédita.

Por lo anteriormente expuesto a usted respetuosamente PIDO:

Me admita la presente solicitud, me tenga por parte en el carácter en que comparezco y oportunamente me extienda el certificado de depósito respectivo.

Señalo para oír notificaciones_____(Dirección) y comisiono para recibirlas a_____.

San Salvador,_____de_____de_____.

Firma.

REQUISITOS: La solicitud se presenta en limpio, original , dos ejemplares de la obra (de preferencia en CD) y el recibo de pago por $ 11.43.

Otros derechos: Constancias: $3.42, Certificaciones literales: $ 5.71, más $ 0.22 por cada página adicional después de la primera hoja.

En caso de obras musicales se presenta la letra original y copia (original y copia de la partitura si la tiene) ó 2 grabaciones)

Si es **PERSONA JURIDICA**, presentar nombramiento vigente del representante legal, y contrato de trabajo o escritura de cesión de derechos del (las) persona (s) que hizo físicamente la obra. Además manifestar que su representada es la dueña de los derechos patrimoniales de la obra_____(en lugar de mencionar que soy el autor de la obra_____)

Si es por medio de apoderado, el poder tiene que ser delegado en persona que sea abogado.

REGISTRO DE LA APLICACIÓN PARA DERECHOS DE AUTOR

Acorde a la licencia proporcionada por la herramienta Qt Designer, toda aplicación de tercera parte que se realice con ella puede ser distribuida libremente sin costo alguno; por otro lado, con respecto a la licencia de Microsoft Visual Studio 6.0, al ser comprada esta a libre decisión del programador si solo cobrar los derechos de desarrollo del software o poder cobrar también por su distribución, no obstante el usuario que va adquirir esta herramienta debe tener instalado y con licencia legal para utilizar los productos Microsoft.

Dependerá de los desarrolladores si se hará con código abierto, ya que al realizar el registro de derecho de autor se esta haciendo notificar que esa aplicación ha sido desarrollada por un grupo de personas y que no se puede utilizar en nombre de otras.

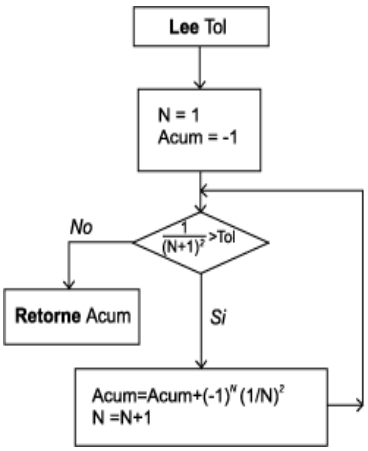
Para que un software sea libre se tiene que publicar con una licencia de software libre generalmente se utiliza la licencia GNU GPL, y para que se pueda redistribuir y mejorar por medio de una documentación libre tiene que tener una licencia de documentación libre GNU FDL.

MANUAL DE USUARIO

DFD_GC

1. Que es un Diagrama de Flujo de Datos?

Los diagramas de flujo son una representación grafica de los pasos de un proceso, útil para determinar como funciona realmente el proceso para producir un resultado; representan la forma más tradicional para especificar los detalles algorítmicos de un proceso. El resultado puede ser un producto, un servicio, información o una combinación de los tres. Se utilizan principalmente en programación, economía y procesos industriales.



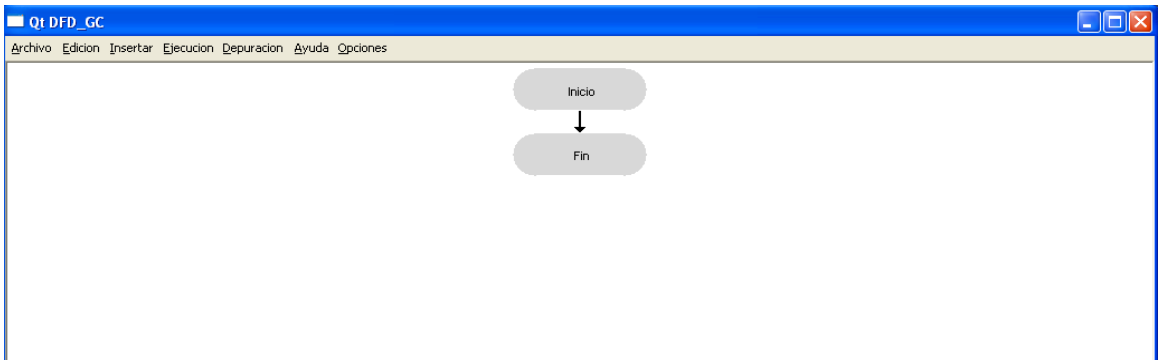
2. Que es el DFD_GC?

Es un editor e interprete de diagramas de Flujo de Datos con generación de código en C++ y Java. Con esta aplicación se puede crear diagramas útiles para varias áreas tales como programación, economía y procesos industriales; además de tener la facilidad de crear, editar y eliminar los diagramas de flujo, la aplicación puede ejecutar dicho diagrama para verificar su total funcionalidad, además posee la opción para el área de programación de que se pueda generar el código de alto nivel ya sea C++ o Java del diagrama de flujos creado y para esto debe ser ejecutado previamente.

3. Como utilizar la Aplicación

3.1 Crear un Nuevo Diagrama

La aplicación cuenta con un área de trabajo simple de utilizar, con un sistema de menú, los cuales pueden ser accesados desde la pantalla principal o utilizando combinaciones de teclas.

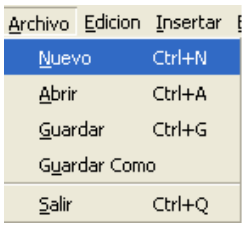


3.1.1 Insertar un Objeto

La aplicación permite la inserción de las siguientes instrucciones:

- Procesos
- Muestra de Datos
- Petición de Datos
- Ciclo For
- Ciclo While
- Decisión If

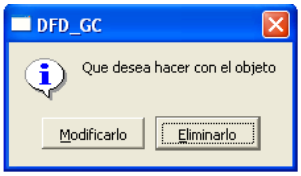
Cada una de estas instrucciones se representa con sus respectivos símbolos dentro del diagrama. Para poder realizar la inserción de una de estas instrucciones sobre el área de trabajo, se puede hacer desde el menú Insertar y eligiendo una de las instrucciones que se desea utilizar o por medio de la combinación de teclas de cada una de las instrucciones.



Luego de haber elegido una instrucción aparecerán en cada flecha del diagrama unos cuadros verdes, esto indica que en uno de esos cuadros se puede insertar la instrucción; para elegir la posición en la que se quiere insertar el objeto basta con dar click sobre el cuadro en la posición que se requiere luego se debe actualizar el diagrama por medio del menú Edicion|Actualizar o utilizando la tecla F5.

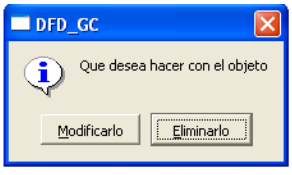
3.1.2 Modificar un Objeto

Para modificar un objeto se debe dar doble click sobre el objeto a modificar y se mostrara un cuadro de dialogo como el que se muestra en la siguiente imagen, en este cuadro se presentan dos opciones si se desea modificar o eliminar, para este caso se debe elegir modificar. A continuación se presentara nuevamente el cuadro correspondiente de la instrucción que se ha seleccionado, para que se puedan realizar los cambios pertinentes; luego de haber hecho los cambios se debe actualizar el diagrama por medio de menú Edición y eligiendo Actualizar o por medio de la tecla F5.



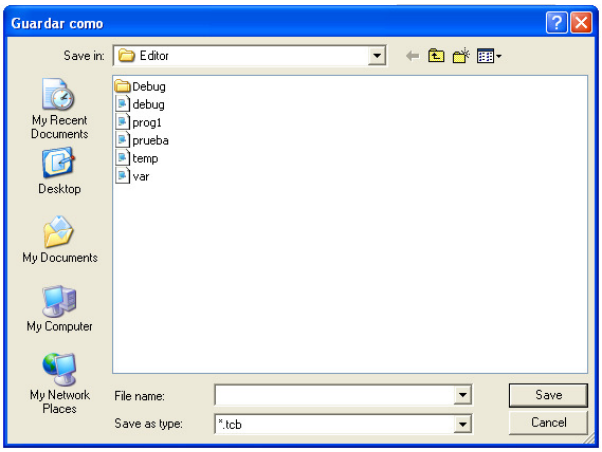
3.1.3 Eliminar un Objeto

Se debe seleccionar el objeto dando doble click sobre el, un cuadro de dialogo aparecerá con dos opciones Modificarlo o Eliminarlo, para este caso se debe elegir Eliminarlo, luego de haber elegido la opción eliminarlo se debe actualizar el diagrama por medio de menú Edición y eligiendo Actualizar o por medio de la tecla F5. La siguiente imagen muestra el cuadro de dialogo que aparecerá con las dos opciones.



3.2 Guardar un Diagrama

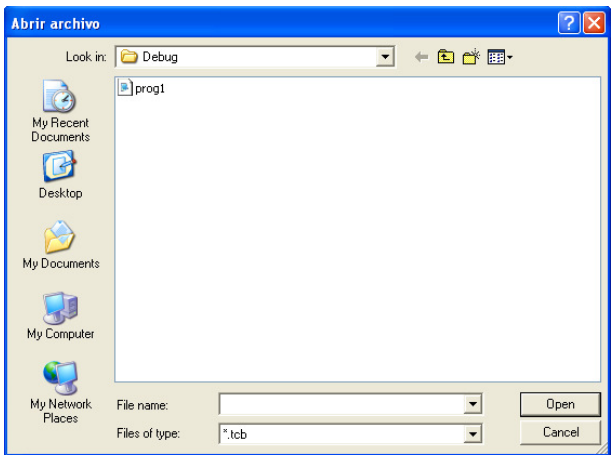
Una vez creado el diagrama y a veces hasta ejecutado se tiene la opción de guardar dicho diagrama con los últimos detalles que se realizaron, dicho diagrama es guardado en cualquier directorio que se elija y su extensión es .tcb; para guardarlo solamente es necesario hacer click en el menú Archivo de la barra de menú, luego seleccionar la opción Guardar o Guardar como, o se puede hacer por medio de la selección de teclas Ctrl + G para el comando Guardar. La siguiente imagen muestra la ventana que se mostrara cuando se elija cualquiera de las dos opciones, en dicha ventana se muestra el explorador de la carpeta donde se desea guardar, el nombre que se le asignara y la extensión del diagrama.



3.3 Abrir un Diagrama

Para esta opción se debe seleccionar el menú Archivo opción Abrir de la barra de menú o por medio de la combinación de teclas Ctrl + A, seguido de esto se cargara la ventana del explorador de las carpetas para poder buscar el diagrama que se desea abrir, luego de haber encontrado y haber seleccionado el diagrama que se quiere abrir, se da click en la opción Abrir de la ventana del explorador; la pantalla principal aparentemente no presentara ningún cambio es decir no se visualizara el diagrama, para poder cargarlo en su totalidad al área de

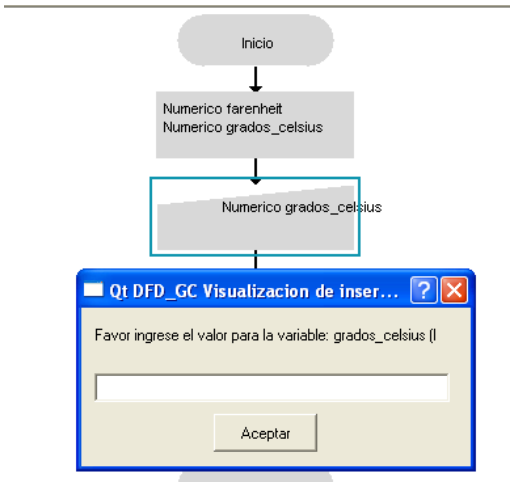
trabajo se debe Actualizar el área de trabajo por medio de el menú Edición opción Actualizar o con la tecla F5, seguido a esto se visualizara el diagrama solicitado.



3.4 Depuración de un Diagrama

Para el proceso de depuración, cada objeto que se esta analizando es marcado por un rectángulo en su contorno; se tienen tres opciones desde el menú Depuración de la barra de menú, una de estas opciones es la de Paso Simple la cual puede ser activada también con la combinación de teclas Ctrl + J, esta opción se utiliza para verificar paso a paso el funcionamiento del diagrama cada vez que se analiza un objeto es necesario activar nuevamente la opción paso simple para que vaya recorriendo cada objeto.

Se cuenta también con otra opción denominada Detener Depuración este comando es útil para cuando se quiere terminar la depuración sin necesidad de esperar a que la aplicación la termine, se puede acceder por medio de las teclas Ctrl + L, la siguiente imagen muestra como se detalla el proceso cuando se esta depurando.



3.5 Ejecución de un Diagrama

Después de haber abierto o creado un diagrama se puede ejecutar utilizando el menú Ejecución seleccionando Ejecutar o utilizando las teclas Ctrl + E.

4. Funciones y Operadores soportados por la Aplicación

OPERADORES

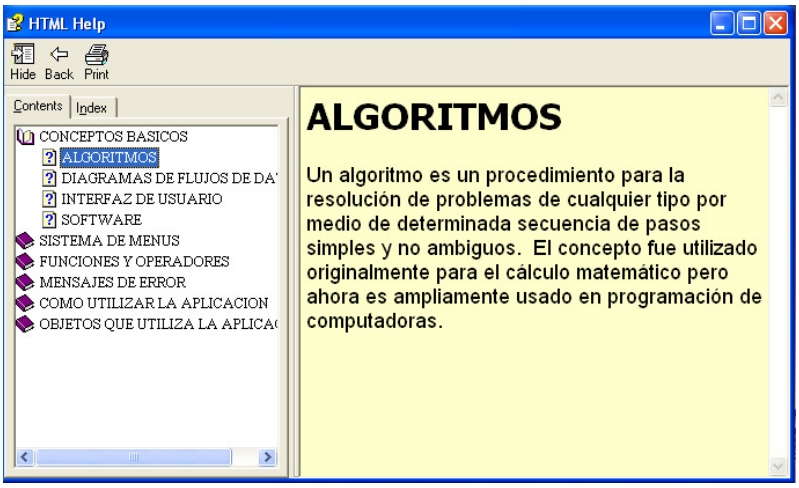
OPERADORES DE ASOCIATIVIDAD	**OPERADORES MATEMATICOS**	**OPERADORES RELACIONALES**	**OPERADORES BOOLEANOS**
PARENTESIS ( )	SUMA +	MAYOR QUE >	AND
	RESTA -	MENOR QUE <	OR
	MULTIPLICACION *	IGUAL A =	NOT
	DIVISION /	DIFERENTE DE ! =	
	EXPONENCIACION ^	MAYOR O IGUAL QUE > =	
	MODULO %	MENOR O IGUAL QUE < =	

FUNCIONES

FUNCIONES MATEMATICAS	
VALOR ABSOLUTO	**ABS (X)**
SENO	**SIN (X)**
COSENO	**COS (X)**
TANGENTE	**TAN (X)**
ARCOSENO	**ASIN (X)**
ARCOCOSENO	**ACOS (X)**
ARCOTANGENTE	**ATAN (X)**
LOGARITMO BASE 10	**LOG (X)**
LOGARITMO NATURAL	**LN (X)**
EXPONENCIAL	**EXP (X)**
RAIZ CUDRADA	**SQRT (X)**
REDONDEO	**CEIL (X)**

5. Como utilizar la Ayuda de la Aplicación

Solo basta con dar click en el menú Ayuda de la barra de menú principal o usando la combinación de letras Ctrl + H en donde se presentara la ventana principal de la ayuda tal y como se muestra en la siguiente imagen.



En dicha ventana puede visualizarse las secciones de la ayuda, solo es necesario hacer doble click sobre cada libro o titulo para poder desplegar los temas contenidos en el.

6. Como abrir los Archivos Ejemplos

Para esta opción solamente se debe dar click en el menú Archivo y seleccionar la opción Abrir o usando las teclas Ctrl + A, seguido a esto se mostrara la ventana del explorador de Windows en la que se debe buscar la carpeta Ejemplos, dicha carpeta contiene todos los archivos ejemplos necesarios para la comprensión de la aplicación asi como también ejemplos de funciones utilizadas usualmente para la creación de programas, es importante recordar que después de elegir el archivo ejemplo que se necesite se debe actualizar el entorno, por medio de el menú Edición y la opción Actualizar o por medio de la tecla F5.

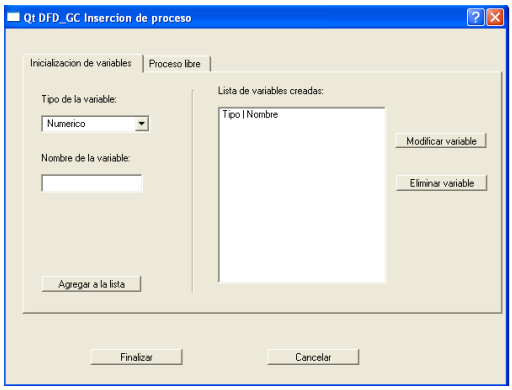
7. Objetos que utiliza la Aplicación

- *OBJETO PROCESO*

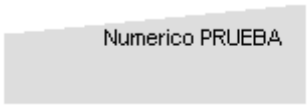
```
Numerico PRUEBA
Caracter HOLA
Logico BANDERA
```

En este objeto se deben crear la variables que se utilizaran en el diagrama, al mismo tiempo es en este objeto donde se deben asignar los valores si se desea una asignación inicial a las variables, se pueden crear también sentencias que se necesite que el proceso haga.

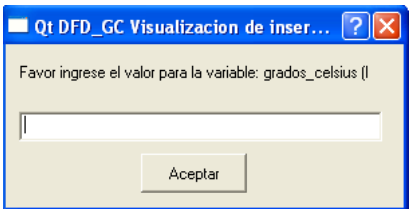
La siguiente imagen muestra la ventana principal que se carga antes de insertar dicho objeto o al momento que se quiera hacer una modificación. Para acceder a esta ventana se puede hacer con las teclas Ctrl + 1.



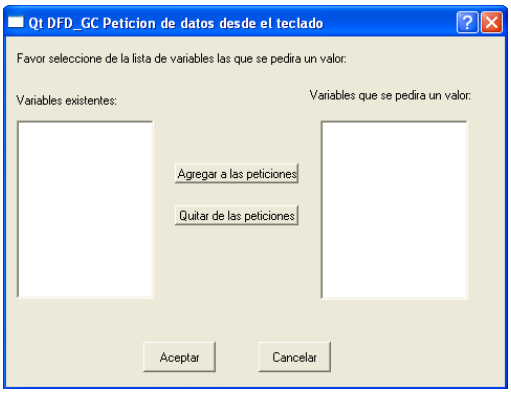
- *OBJETO PETICION DE DATOS (INPUT)*



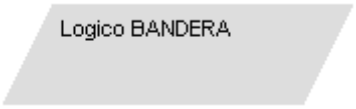
Este objeto permite la inserción de datos desde el teclado introducidos por el usuario, y cada dato introducido es asignado a cada una de las variables que esta siendo requerida. Al ejecutarse el objeto generara una ventana como la siguiente:



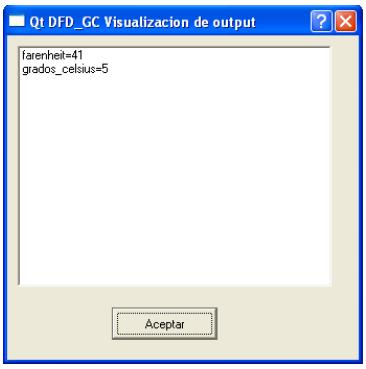
Para insertar este objeto se deben elegir las variables que se desean pedir en el tiempo de Ejecución, la ventana que se presentara para la inserción de este objeto es la siguiente. Para acceder a esta ventana se puede hacer con las teclas Ctrl + 2.



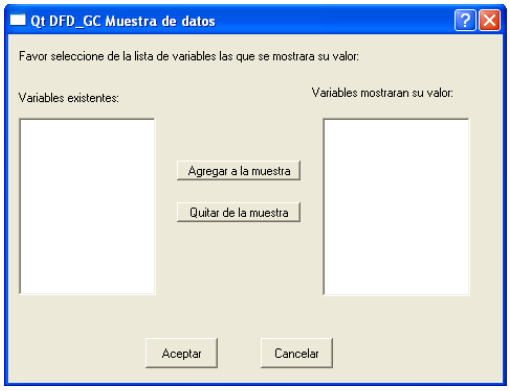
- *OBJETO MUESTRA DE DATOS (OUTPUT)*



Este objeto permite presentar en pantalla a la hora de Ejecución el valor de alguna variable, la siguiente imagen muestra la ventana que se presentara cuando se este ejecutando dicha instrucción.



Para poder insertar este objeto es necesario elegir la variable que se quiere cuyo valor se quiere visualizar, se puede hacer una lista de valores los cuales se quieren ver a la hora de ejecutarse la siguiente imagen muestra la ventana principal para la inserción del objeto. Para acceder a esta ventana se puede hacer con las teclas Ctrl + 3.



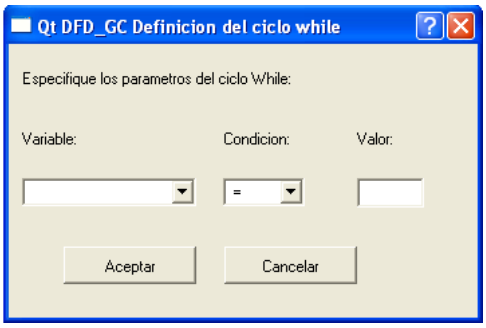
- *OBJETO PARA CICLOS*

Este objeto es utilizado para dos tipos de ciclos el ciclo While y el ciclo For.

CICLO WHILE

Para poder insertar este objeto con respecto al ciclo While la ventana principal que se mostrara es como se ve en la siguiente imagen. Para acceder a esta ventana se puede hacer con las teclas Ctrl + 4.

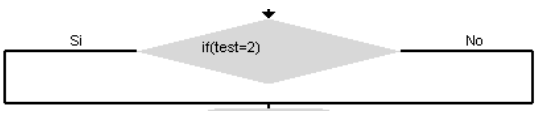
.



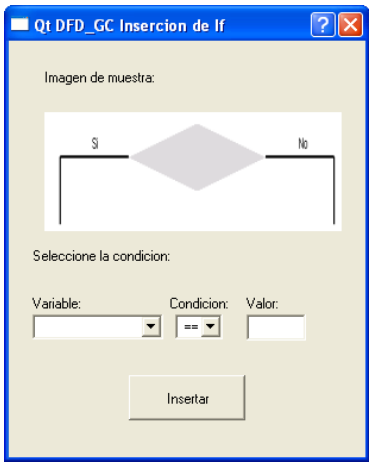
CICLO FOR

Para insertar este objeto solamente se debe acceder a su menú desde Insertar y elegir dicha instrucción la cual mostrara la siguiente ventana, siendo esta la ventana principal para la inserción de la instrucción For. Para acceder a esta ventana se puede hacer con las teclas Ctrl + 5.

- *OBJETO PARA DECISION IF*



Este objeto es insertado cuando se crea la instrucción de Decisión. Para poder insertar esta instrucción al diagrama se debe hacer desde su respectivo comando desde la barra de menú o usando su combinación de teclas, al hacer esto presentara la pantalla principal de la Decisión If tal y como se muestra en la siguiente imagen. Para acceder a esta ventana se puede hacer con las teclas Ctrl + 6.

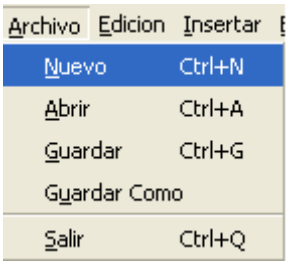


- *OBJETOS PARA TERMINACION DE CICLOS*

Este tipo de objeto se genera automáticamente al crear un ciclo While, un ciclo For o una Decisión If, lo cual significa que se esta finalizando dicha instrucción.

8. Sistema de Menú de la Aplicación

En el menú Archivo se encuentran las opciones:



- ***Nuevo:*** que sirve para crear un nuevo espacio de trabajo y crear otro diagrama; también se puede solicitar este comando utilizando las teclas Ctrl.+N

- ***Abrir:*** esta opción esta diseñada para abrir cualquier diagrama que se haya creado con anterioridad, para acceder a este comando desde teclado se utiliza la combinación de teclas Ctrl + A.

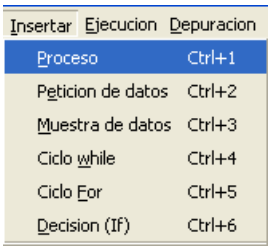
- ***Guardar:*** esta opción se utiliza para guardar los datos de los diagramas creados en el espacio de trabajo, estos datos son guardados en un archivo de extensión .tcb, en donde se almacena el tipo de objeto, coordenadas en donde se encuentra en el diagrama y el contenido de el mismo, al decir contenido del mismo se refiere a el tipo de variables utilizadas, los valores de las variables, para esta opción se puede acceder desde teclado usando las teclas Ctrl + G.

- ***Salir:*** por si se desea terminar con la aplicación. Se puede acceder a este comando utilizando las teclas Ctrl + Q.

En el menú Edición se encuentra la opción:

- ***Actualizar:*** que se utiliza para hacer un redibujamiento del diagrama con los últimos movimientos realizados. Para acceder a este comando desde este teclado se utiliza la combinación de teclas F5.

En el menú Insertar se encuentran las opciones:



- ***Proceso:*** se encarga de llamar en pantalla el formulario de Procesos el cual consiste en solicitar al usuario las variables, valores iniciales y sentencia que dicho proceso realizara. Para acceder al comando también se utiliza Ctrl + 1.

- ***Petición de Datos:*** presenta en pantalla el formulario de Petición de Datos el cual consiste en solicitar al usuario por medio de su teclado los valores que dicha variable contendrá. Para acceder a este comando desde teclado se utiliza Ctrl + 2.

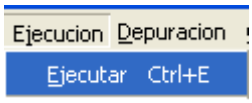
- ***Muestra de Datos:*** carga en la pantalla el formulario de Muestra de datos el cual consiste en solicitar al usuario algún mensaje que este quiera presentar en pantalla a la hora de ejecución. Para acceder desde el teclado se usa la combinación Ctrl + 3.

- ***Ciclo While:*** presenta en pantalla el Formulario de Ciclo While el cual consiste en la creación de un enlace que sea valido mientras se cumpla la condición determinada que el usuario tendrá que ingresar. Se utilizan las combinaciones de teclas Ctrl + 4.

- ***Ciclo For:*** carga el formulario de Ciclo For que al igual que el ciclo While representa un bucle con la diferencia de que este bucle se cumple mediante un contador determinado por el usuario. Desde teclado se accede con las teclas Ctrl + 5.

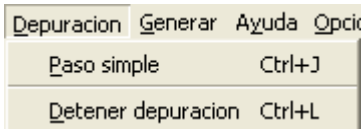
- ***Decisión IF:*** presenta el formulario de Decisiones IF, el cual permite al usuario crear una condición determinada por el usuario sea esta verdadera o falsa. Se utiliza Ctrl + 6 para acceder desde la pantalla principal.

En el menú Ejecución se encuentran las opciones:



- ***Ejecutar:*** ya creado el diagrama en el espacio de trabajo el usuario tiene la opción de ejecutar el diagrama esta opción es la encargada de realizar esa tarea. Para ejecutar este comando desde el teclado se hace con la combinación de teclas Ctrl + E.

En el menú de Depuración se encuentran las opciones:



- ***Paso Simple:*** se utiliza para depurar el diagrama, es decir verificar que no se encuentre ningún error en tiempo de ejecución, con este comando la depuración surge paso a paso. Se puede activar por medio del teclado con la combinación de teclas Ctrl + J.

- ***Detener Depuración:*** se utiliza cuando no se desea continuar con la depuración se tiene la opción de finalizar el proceso de depuración por medio de este comando. Para utilizarlo mediante le teclado se utilizan las letras Ctrl + L.

  En el menú de Generar se encuentran las opciones:

- ***C++:*** una vez ejecutado con éxito el diagrama se tiene la posibilidad de generar su respectivo codigo en C++ escogiendo esta opción.

- ***Java:*** Se tiene la posibilidad de generar el codigo respectivo del diagrama creado hacia el lenguaje Java.

  En el menú de Opciones se encuentra la siguiente opción:

- ***Doble Buffer:*** esta opción sirve para las computadoras con poca capacidad de memoria y velocidad; si la opción se encuentra señalada significa que se ha activado la opción de movimiento de los objetos de una forma rápido; es decir para computadoras de mayor capacidad de memoria y velocidad.

9. Generación de Código

Esta opción puede ser utilizada solamente si el diagrama que se ha creado y ha sido ejecutado exitosamente, una ejecución exitosa consiste en que al realizar la depuración y ejecución no se ha encontrado ningún error en el diagrama; luego de haber pasado la ejecución se tiene el menú Generar en donde presenta las opciones C++ y Java, que son los dos lenguajes de alto nivel al que la aplicación puede traducir el diagrama a su respectivo código.

Luego de haber elegido cualquiera de las dos opciones, una ventana se mostrara con el código equivalente al diagrama.

10. Mensajes de Error

- Error en valor introducido de la variable

  Este mensaje de error aparecerá cada vez que se introduzca un valor erróneo a alguna variable como por ejemplo se tiene una variable y se pide que se introduzca su valor, si se asigna un valor que no sea numérico a dicha variable se producirá este tipo de mensaje.

- Error en nombre de variable

  Este mensaje de error surgirá cada vez que se este intentando declarar una variable y el nombre de dicha variable sea compuesto y este separada por espacios como por ejemplo: agenda mensual, este nombre debe declararse agenda_mensual, agenda-mensual o agendamensual.

- Nombre de variable es palabra reservada.

  Este tipo de error se presenta al intentar introducir el nombre de una variable, cuyo nombre sea el mismo a una palabra reservada por ejemplo: numérico, carácter, lógico o similar al de algún tipo de instrucción como: For, while, etc.

- La variable ya esta definida

Este mensaje representa que se ha declarado una variable anteriormente y se esta tratando de definirla una vez mas.

- Verificar manejo de paréntesis en la expresión

Se muestra este mensaje cada vez que se este utilizando los paréntesis, debe ser verificado que cada paréntesis que se abra debe estar cerrado.

- Hay varios signos =

Este tipo de mensajes se aparecen cada vez que se este haciendo una asignación de un valor a una variable o el uso de alguna formula matemática, hace referencia a que la asignación o igualdad se esta haciendo mas de una vez es decir el signo esta siendo utilizado mas de una vez.

- No se reconoce la palabra

Este mensaje de error aparecerá cada vez que se utilice una variable en alguna parte del diagrama y no se haya declarado anteriormente.

- División entre cero

Este mensaje advierte que el valor de la parte divisora de esta operación es igual a cero, cosa que por reglas matemáticas es imposible.

- La asignación de valores solo se realiza sobre variables

Este error se presenta en tiempo de revisión cuando se intenta hacer una asignación a un campo numérico.

- Se ha producido un bucle infinito

Este tipo de error se produce en tiempo de Ejecución, cuando el valor de las iteraciones dentro de un bucle llega al valor de 9,999. lo cual produce que se detenga el programa.

- Los tipos de datos no coinciden

Este error se presenta en tiempo de Ejecución cuando se intenta hacer una operación con tipos de datos diferentes.

- Desbordamiento en la operacion

Se presenta en tiempo de Ejecución cuando en la evaluación de una expresión se obtiene un valor de tipo de dato Real superior a 1*10200 ó inferior a -1*10200.

Ejemplo:
10^2000 * 10^2000

- El incremento debe ser un valor numérico

Este error se presenta en tiempo de revisión cuando en un Objeto Ciclo FOR se encuentra que el incremento no es un valor numérico.

- El valor limite para el contador debe ser numérico

Este error se presenta en tiempo de Ejecución cuando en un Ciclo FOR se intenta establecer como valor límite para el contador un valor que no es de tipo numérico.

TABLA COMPARATIVA ENTRE DFD_GC Y DFD.

	DFD_GC	DFD
Porcentaje de uso del procesador	0% - 4%	100%
Permite la utilización de subprogramas	No	Si
Permite la creación de código a lenguajes de alto nivel	Si	No
Permite el almacenamiento en disco de los diagramas creados	Si	Si
Permite la portabilidad de la aplicación	Si	Si
Nivel facilidad en la creación del diagrama	Fácil	Difícil
Nivel facilidad en la utilización de herramientas de depuración y ejecución	Fácil	Fácil