SISTEMA HÍBRIDO

Universidad de Guadalajara, Centro Universitario de Ciencias Exactas e Ingenierías.

Autores:

Juan José Salazar Villegas. Código: 215661291

Paola Vanessa Del Rio Gómez. Código: 215480181

Mally Samira Hernández Martínez. Código: 220286113

Juan Emmanuel Fernández de Lara Hernández. Código: 220286571

30/04/2023

Carrera: Ingeniería En Computación (INCO)

Asignatura: Traductores De Lenguajes II

Maestro: Ramos Barajas Armando

Origen del informe: Guadalajara Jalisco México

Ciclo: 2023 A

Sección: D03

Actividad: 4

Índice

Introducción

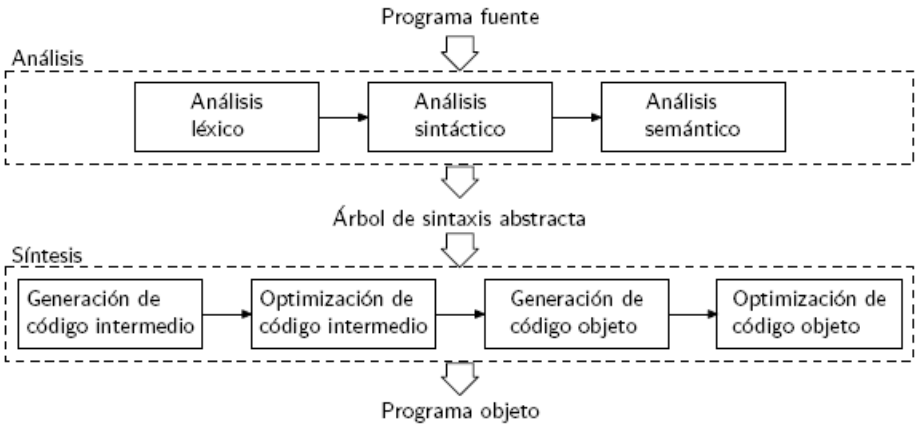
Ahora que comprendemos cada una de las fases por las cuales nuestro compilador: identifica, analiza y evalúa tocará analizar desde un punto de vista distinto siendo este en ensamblador y como lo conocemos recordemos que este trabajara a un nivel bajo donde podremos operar más fácilmente con la memoria de nuestra computadora y facilitando programas que parecen complejos a un lenguaje de alto nivel por ello en esta práctica realizaremos un híbrido de lo que utilizamos con el lenguaje “C” y Ensamblador esto para eficiente los procesos así como el análisis necesario para llevar a cabo esta tarea híbrida.

Como hemos aprendido durante lo largo del semestre la programación en ensamblador nos da la posibilidad de manejar los datos directamente al tener un contacto directo de los registros de memoria, así como sus operaciones, con esto y la combinación del lenguaje C tomados de la mano nos da la posibilidad de programación de sistemas complejos con manejo más directo de la memoria dando mayor eficiencia y velocidad.

Accediendo a un nivel más bajo para obtener mayor operatividad y flexibilidad con el fin de economizar memoria, aplicar algoritmos críticos entre otras lo más óptimo.

Como anteriormente establecimos y justificamos la programación en C + + es uno de los lenguajes más adecuados para la programación en sistemas o aquellos que buscan trabajar a un nivel bajo interactuando directamente con los registros hasta pudiendo sustituir al lenguaje ensamblador en algunos casos.

Ahora volviendo a interactuar con un nivel bajo según sea la necesidad ya sean mayor operaciones como depuración y optimización de nuestro programa economizando recursos, memoria uso de algoritmos con mayor rapidez que manejan constantemente
operaciones críticas de nuestro sistema. Así pues, con estas pautas y recordando aquellas que habíamos establecido podremos ver que una unificación entre ensamblador y C + + podrían dar mucho según nuestros objetivos.



Objetivo General:

La palabra clave **__asm__ __volatile__** invoca el ensamblador alineado y puede aparecer siempre que una instrucción de C sea válida. Se puede considerar como la instrucción más importante para la implementación del lenguaje ensamblador en C. No puede aparecer por sí sola. Debe ir seguida de una instrucción de ensamblado, un grupo de instrucciones entre llaves o, como mínimo, un par de llaves vacías. El término "bloque __asm__ __volatile__" aquí hace referencia a cualquier instrucción o grupo de instrucciones, incluido o no entre llaves.

Si bien de primera instancia la palabra clave **"__asm__ __volatile__"** puede utilizarse en un entorno de desarrollo determinado, por cuestiones de pruebas, se podrían implementar otros términos ya que dicha palabra tiene una funcionalidad temporal. Asm Volatile también evita que el compilador elimine la expresión si decide que los valores de salida no se utilizan. Sin embargo, esto solo puede suceder si hay valores de salida. Además, existen diversos comandos dentro de esta sintaxis que permiten ingresar comandos de ensamblaje. Los siguientes son los principales elementos para la representación de los comandos necesarios:

MODIFICADORES	
=	Indica que el operando es de sólo escritura para esta instrucción. El valor anterior se descarta y se sustituye por los datos de salida.
+	Indica que el operando es tanto leído como escrito por la instrucción.

&	Indica que el operando puede modificarse antes de que la instrucción termine de utilizar los operandos de entrada; un registro que se utiliza como entrada no debe reutilizarse aquí.
CONSTANTES	
a	Utiliza un registro de dirección (registro de propósito general excepto r0).
d	Utiliza un registro de datos que sea un registro de propósito general arbitrario. Esta restricción es la misma que la restricción r.
g	Utiliza un registro general, una memoria o un operando inmediato.
i	Utilizar un operando literal entero o de cadena inmediato.
m	Utilizar un operando de memoria soportado por la máquina.
n	Utiliza un entero inmediato.
o	Utilizar un operando de memoria que sea desplazable.
r	Utilice un registro general.
s	Utiliza un operando literal de cadena.
0,1…8,9	Una restricción de concordancia. Asigna el mismo registro en la salida que en la entrada correspondiente.

Objetivo General:

Se busca implementar un Código combinado siendo C + + y Ensamblador esto para construir nuestra calculadora gracias a lo visto durante el semestre, repasando y analizando cómo aplicar dichos aprendizajes.

Objetivo Particular:

La realización de un programa que permita la generación de código en base a todos los trabajos realizados durante el semestre, en donde se analice este programa con metodologías típicas de la ingeniería de software. Además de la realización de diagramas que permitan la representación gráfica del sistema.

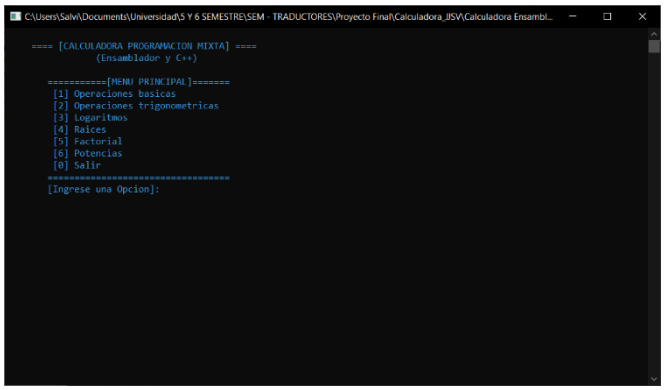
Desarrollo (Pantallazos)

Desarrollo del programa:

Dentro de la elaboración de este programa mixto y con los conocimientos antes expuestos para poder poner en práctica una colaboración entre ambos lenguajes de alto y bajo nivel se dará una pantalla de cada una de las operaciones posibles de nuestro producto final.

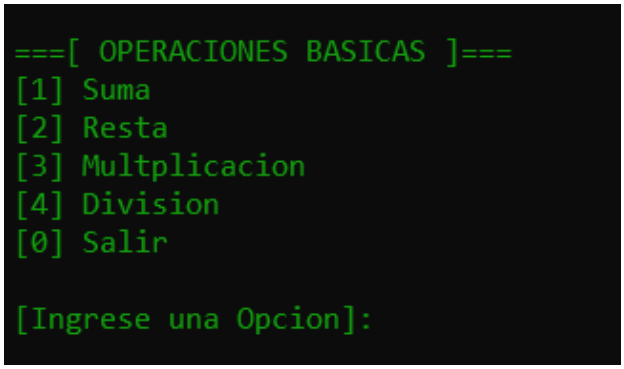
Menú Principal:

```
system("cls");
system ("color 3");
cout << endl << "     ==== [CALCULADORA PROGRAMACION MIXTA] ==== "<<endl;
cout << "            (Ensamblador y C++) "<<endl;
cout << endl << "   ==========[MENU PRINCIPAL]======="<<endl;
cout << "    [1] Operaciones basicas          "<<endl;
cout << "    [2] Operaciones trigonometricas  "<<endl;
cout << "    [3] Logaritmos                   "<<endl;
cout << "    [4] Raices                       "<<endl;
cout << "    [5] Factorial                    "<<endl;
cout << "    [6] Potencias                    "<<endl;
cout << "    [0] Salir                        "<<endl;
cout << "   ================================="<<endl;
cout << "   [Ingrese una Opcion]: ";
cin >> option;
```

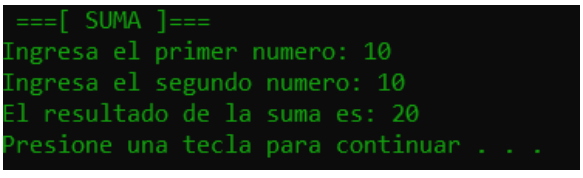


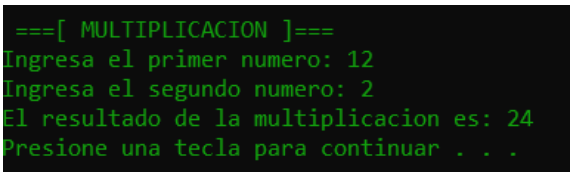
Despliegue de Operaciones Básicas:

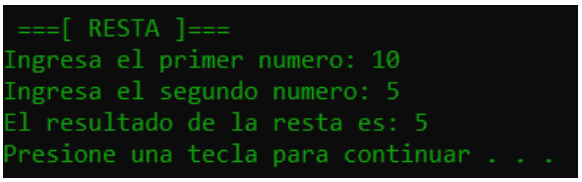
```
system("cls");
system ("color 2");
cout << endl << " ===[ OPERACIONES BASICAS ]==="<<endl;
cout << " [1] Suma"<<endl;
cout << " [2] Resta"<<endl;
cout << " [3] Multplicacion"<<endl;
cout << " [4] Division"<<endl;
cout << " [0] Salir"<<endl;
cout << endl << " [Ingrese una Opcion]: ";
```

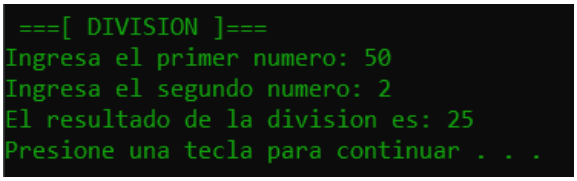


Desempeño









Operaciones trigonométricas (En radianes):

```
system("cls");
system ("color 2");
cout << endl << " ===[ OPERACIONES TRIGONOMETRICAS ]==="<<endl;
cout << " [1] Seno x"<<endl;
cout << " [2] Coseno x"<<endl;
cout << " [3] Tangente x"<<endl;
cout << " [4] Secante x"<<endl;
cout << " [5] CoSecante x"<<endl;
cout << " [6] CoTangente x"<<endl;
cout << " [0] Salir"<<endl;
cout << endl << " [Ingrese una Opcion]: ";
cin >> option;
```

```
===[ OPERACIONES TRIGONOMETRICAS ]===
[1] Seno x
[2] Coseno x
[3] Tangente x
[4] Secante x
[5] CoSecante x
[6] CoTangente x
[0] Salir

[Ingrese una Opcion]:
```

Desempeño:

```
===[ TANGENTE X ]===
Ingresa el valor de x: 54
Tan x: 0.6738
Presione una tecla para continuar . . .
```

```
===[ COSENO X ]===
Ingresa el valor de x: 55
Cos x: 0.0221268
Presione una tecla para continuar . . .
```

```
===[ SENO X ]===
Ingresa el valor de x: 22
Sin x: -0.00885131
Presione una tecla para continuar . . .
```

```
===[ SECANTE X ]===
Ingresa el valor de x: 65
Sec x: -1.77792
Presione una tecla para continuar . . .
```

```
===[ CO-SECANTE X ]===
Ingresa el valor de x: 89
CoSec x: 1.1627
Presione una tecla para continuar . . .
```

```
===[ CO-TANGENTE X ]===
Ingresa el valor de x: 64
CoTan x: 0.42592
Presione una tecla para continuar . . .
```

Logaritmos:

```
system("cls");
system ("color 2");
cout << endl << "  ===[ POTENCIAS ]==="<<endl;
cout << " [1] Potencia cuadratica"<<endl;
cout << " [2] Potencia a la n"<<endl;
cout << " [0] Salir"<<endl;
cout << endl << " [Ingrese una Opcion]: ";
cin >> option;
```

```
===[ LOGARITMOS ]===
[1] Logaritmo Natural
[2] Logaritmo base 2
[3] Logaritmo base 10
[0] Salir

[Ingrese una Opcion]:
```

Desempeño:

```
===[ LOGARITMO NATURAL ]===
Ingresa el valor de x: 55
El logaritmo natural de x es: 4.00733
Presione una tecla para continuar . . .
```

```
Ingresa el valor de x: 33
El logaritmo base 2 de x es: 5.04439
Presione una tecla para continuar . . .
```

```
===[ LOGARITMO BASE 10 ]===
Ingresa el valor de x: 100
El logaritmo base 10 de x es: 2
Presione una tecla para continuar . . .
```

Potencias:

```
system("cls");
system ("color 2");
cout << endl << "  ===[ POTENCIAS ]==="<<endl;
cout << " [1] Potencia cuadratica"<<endl;
cout << " [2] Potencia a la n"<<endl;
cout << " [0] Salir"<<endl;
cout << endl << " [Ingrese una Opcion]: ";
cin >> option;
```

```
 ===[ POTENCIAS ]===
[1] Potencia cuadratica
[2] Potencia a la n
[0] Salir

[Ingrese una Opcion]:
```

Desempeño

```
 ===[ POTENCIA CUADRATICA ]===
Ingresa el numero: 55
El valor cuadratico del numero es: 3025
Presione una tecla para continuar . . .
```

```
 ===[ POTENCIA A LA N ]===
Ingresa el numero a calcular su potencia: 61
Ingresa la potencia: 4
El valor a la potencia de n, del numero es: 1.38458e+007
Presione una tecla para continuar . . .
```

Raíces:

```
system("cls");
system ("color 2");
cout << endl << " ===[ RAICES ]==="<<endl;
cout << " [1] Raiz cuadratica"<<endl;
cout << " [2] Raiz a la n"<<endl;
cout << " [0] Salir "<<endl;
cout << endl << " [Ingrese una Opcion]: ";
cin >> option;
```

```
===[ RAICES ]===
[1] Raiz cuadratica
[2] Raiz a la n
[0] Salir

[Ingrese una Opcion]:
```

Desempeño

```
 ===[ RAIZ CUADRADA ]===
Ingresa el numero: 22
La raiz cuadrada del numero es: 4.69042
Presione una tecla para continuar . . .
```

```
 ===[ RAIZ A LA N ]===
Ingresa el numero a calcular su raiz: 33
Ingresa el valor de n: 5
El valor a la potencia de n, del numero es: 2.01235
Presione una tecla para continuar . . .
```

Factorial:

```
system("cls");
system ("color 2");
cout << endl << " ===[ FACTORIAL ]=== "<<endl;
cout << " [1] Ingresar el valor del factorial"<<endl;
cout << " [0] Salir "<<endl;
cout << endl << " [Ingrese una Opcion]: ";
cin >> option;
```

```
===[ FACTORIAL ]===
[1] Ingresar el valor del factorial
[0] Salir

[Ingrese una Opcion]:
```

Desempeño

```
 ===[ FACTORIAL ]===
Ingresa el numero: 5
El factorial del numero es: 120
Presione una tecla para continuar . . .
```

Conclusiones

Juan José Salazar Villegas:

Dentro de la codificación puede considerar que al principio fue un reto ya que combinar ambos es recordar su sintaxis y al depurar se intercala entre uno y otro confundiendo un poco, fuera de esto es satisfactorio el ver en total funcionamiento la calculadora siendo algo fácil de implementar para ambos lenguajes. Ahora bien, como se planteó se modificó el producto final y con ello el flujograma planteado siendo las operaciones básicas se añadió operaciones más complejas como lo son trigonométricas, así como potencias, logaritmos, raíces y factoriales. Con esto e implementando lo visto durante el semestre damos muestra de todo con este producto final.

Paola Vanessa Del Rio Gómez:

A mí parecer, este programa me pareció algo difícil en tanto la implementación de ya todo, aunque no debería generar problemas porque ya sabíamos el proceso los hacía al momento de que queríamos juntar todo pero nos aparecía alguno que otro bug, así que tuvimos que ponernos a investigar sobre cómo hacerlo de manera correcta para una buena implementación, una vez hecho esto fue fácil de hacer, de igual forma me gustó mucho el resultado final, ya que ahí se vio reflejado el esfuerzo de todo el semestre.

Hernandez Martinez Mally Samira:

En conclusión, este programa de igual forma que mis compañeros se me hizo algo complicado ya que comprender cómo funciona el sistema híbrido es algo puede ser una solución efectiva para problemas específicos, pero es importante evaluar cuidadosamente las necesidades y desafíos específicos antes de decidir si es buena opción; en este caso si fue buena opción al plantear las operaciones básicas y añadiendo las más complejas como potencias, trigonometricas , raíces, etc. Además de que tuvimos una buena comunicación en el equipo a la hora de crear y ejecutar dicho programa.

Juan Emmanuel Fernández de Lara Hernández:

El programa de calculadora híbrida desarrollado por el equipo de trabajo presentó algunos retos iniciales en cuanto a la combinación de sintaxis de ambos lenguajes, así como en la depuración del código. Sin embargo, una vez superados estos obstáculos, el resultado final fue satisfactorio y reflejó el esfuerzo y aprendizaje obtenido durante el semestre.

El trabajo en equipo y la buena comunicación fueron elementos clave para lograr una implementación exitosa del programa. En resumen, el proyecto permitió a los integrantes del equipo aplicar los conocimientos adquiridos en el semestre y enfrentarse a un reto real de programación.

Bibliografia

- *Ensamblador alineado (C)*. (2022b, septiembre 26). Microsoft Learn. https://learn.microsoft.com/es-es/cpp/c-language/inline-assembler-c?view=msvc-170

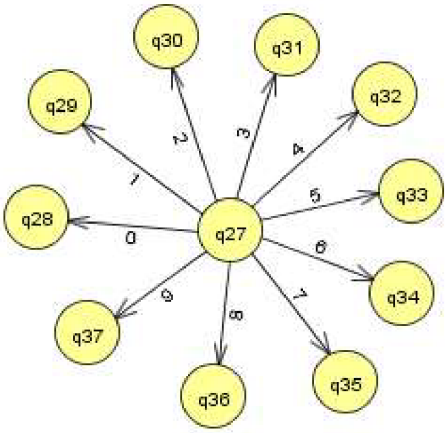
Apéndices

No se cuenta con apéndices para este reporte.

Acrónimos

No se cuenta con apéndices para este reporte.

Diagramas



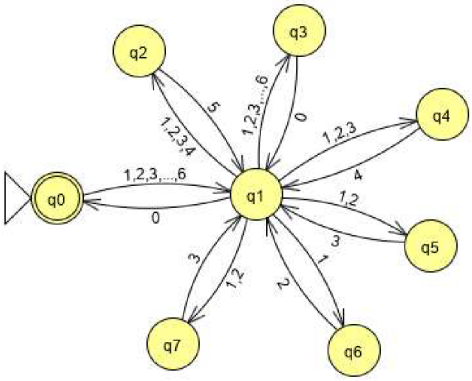


Diagrama De Casos De Estudio:

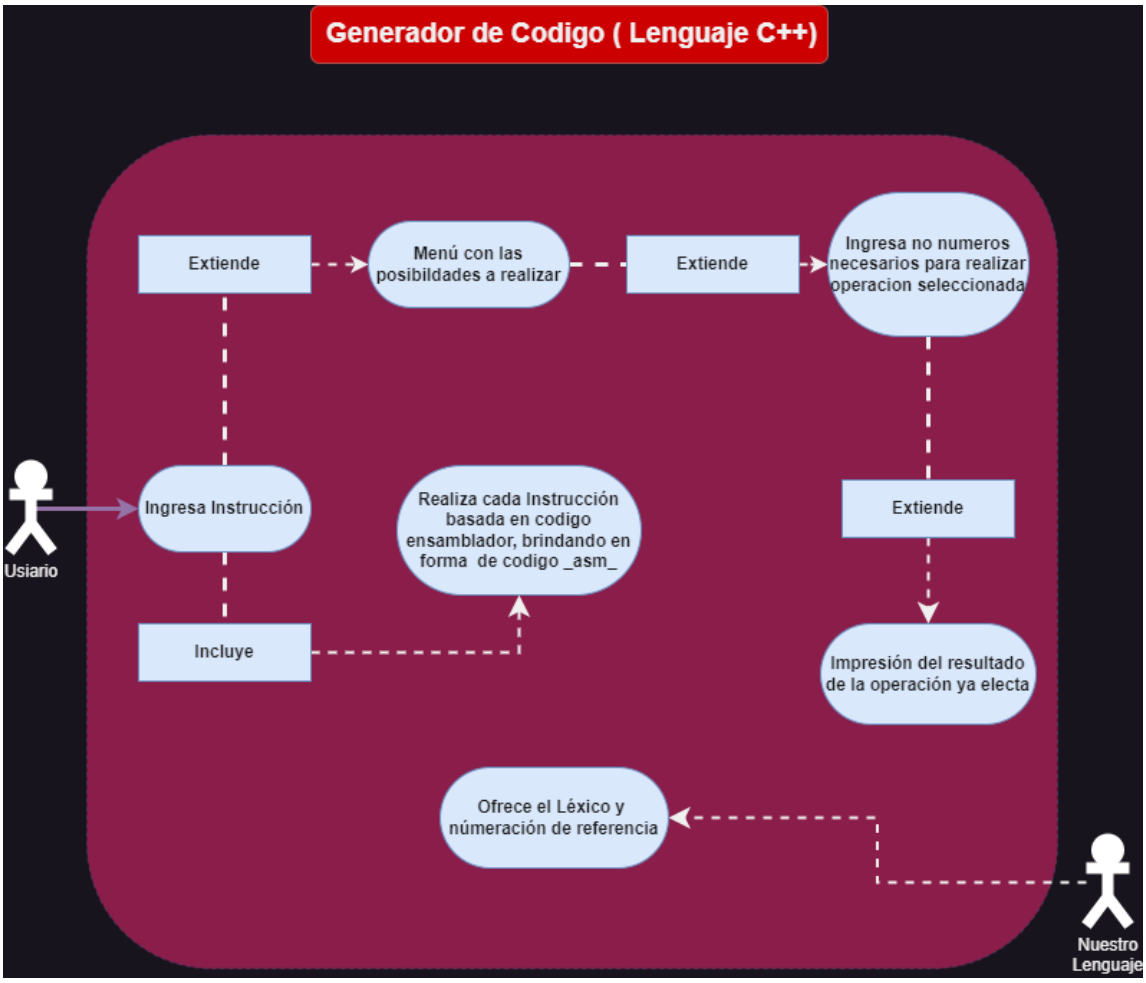


Tabla de transiciones

EDO	0	1	2	3	4	5	6
q0	0	{q1}	{q1}	{q1}	{q1}	{q1}	{q1}
q1	{q0}	{q2},{q3},{q4},{q5},{q6},{q7}	{q2},{q3},{q4},{q5},{q7}	{q2},{q3},{q4}	{q2},{q3}	{q3}	{q3}
q2	0	0	0	0	0	{q1}	0
q3	{q1}	0	0	0	0	0	0
q4	0	0	0	0	{q1}	0	0
q5	0	0	0	{q1}	0	0	0
q6	0	0	{q1}	0	0	0	0
q7	0	0	0	{q1}	0	0	0

EDO	0	1	2	3	4	5	6	7	8	9
q27	{q28}	{q29}	{q30}	{q31}	{q32}	{q33}	{q34}	{q35}	{q36}	{q37}
q28	0	0	0	0	0	0	0	0	0	0
q29	0	0	0	0	0	0	0	0	0	0
q30	0	0	0	0	0	0	0	0	0	0
q31	0	0	0	0	0	0	0	0	0	0
q32	0	0	0	0	0	0	0	0	0	0
q33	0	0	0	0	0	0	0	0	0	0
q34	0	0	0	0	0	0	0	0	0	0
q35	0	0	0	0	0	0	0	0	0	0
q36	0	0	0	0	0	0	0	0	0	0
q37	0	0	0	0	0	0	0	0	0	0

Requisitos Funcionales:

Numero de requisito: RF01

Nombre de requisito: Actualización de tokens

Tipo: Requisito

Fuente del requisito: Función básica de compilador

Prioridad del requisito: Alto/Esencial

Descripción:

Al iniciar el programa este actualizará los tokens establecidos junto a las funciones que clasifican los tipos de datos.

Número de requisito: RF02

Nombre de requisito: Procesamiento de entradas.

Tipo: Requisito

Fuente del requisito: Función básica de compilador

Prioridad del requisito: Alto/Esencial

Descripción:

El usuario ingresa algún código, cadena o palabra la cual se procesa y detectará en qué categoría pertenece y asignarla para posteriormente mostrar en pantalla y dar el tipo de dato relacionado a su entrada. De no ser así se mostrará un mensaje de error.

Número de requisito: RF03

Nombre de requisito: Reacción a errores.

Tipo: Requisito

Fuente del requisito: Función básica de compilador

Prioridad del requisito: Alto/Esencial

Descripción:

Cuando se presenta algún tipo de error desde la perspectiva del usuario no generará conflictos ya que no interrumpirá el curso del programa ni la validación de códigos consecuentes.

Requisitos No Funcionales:

Número de requisito: **RNF 01**

Nombre de requisito: GUI / Interfaz Visual

Tipo: Requisito

Fuente del requisito: Interfaz de usuario.

Prioridad del requisito: Media/Deseado

Descripción:

Al pensar en Interfaz hablamos de un menú simple donde se pueda desplazar fácilmente y tenga un orden claro y directa esto para que el usuario pueda ingresar datos fácilmente sin tener que comprender un complejo sistema de interfaz

Número de requisito: **RNF 02**

Nombre de requisito: Eficacia y calidad del software.

Tipo: Requisito

Fuente del requisito: Funcionalidad óptima del programa brindando una buena experiencia.

Prioridad del requisito: Alta/Esencial

Descripción:

El programa debe ser capaz de analizar y catalogar correctamente la mayoría de los caracteres de entrada por el usuario, dando una experiencia óptima.

Número de requisito: **RNF 03**

Nombre de requisito: Accesibilidad y funcionalidad óptima

Tipo: Requisito

Fuente del requisito: Facilidad de uso.

Prioridad del requisito: Alta/Esencial

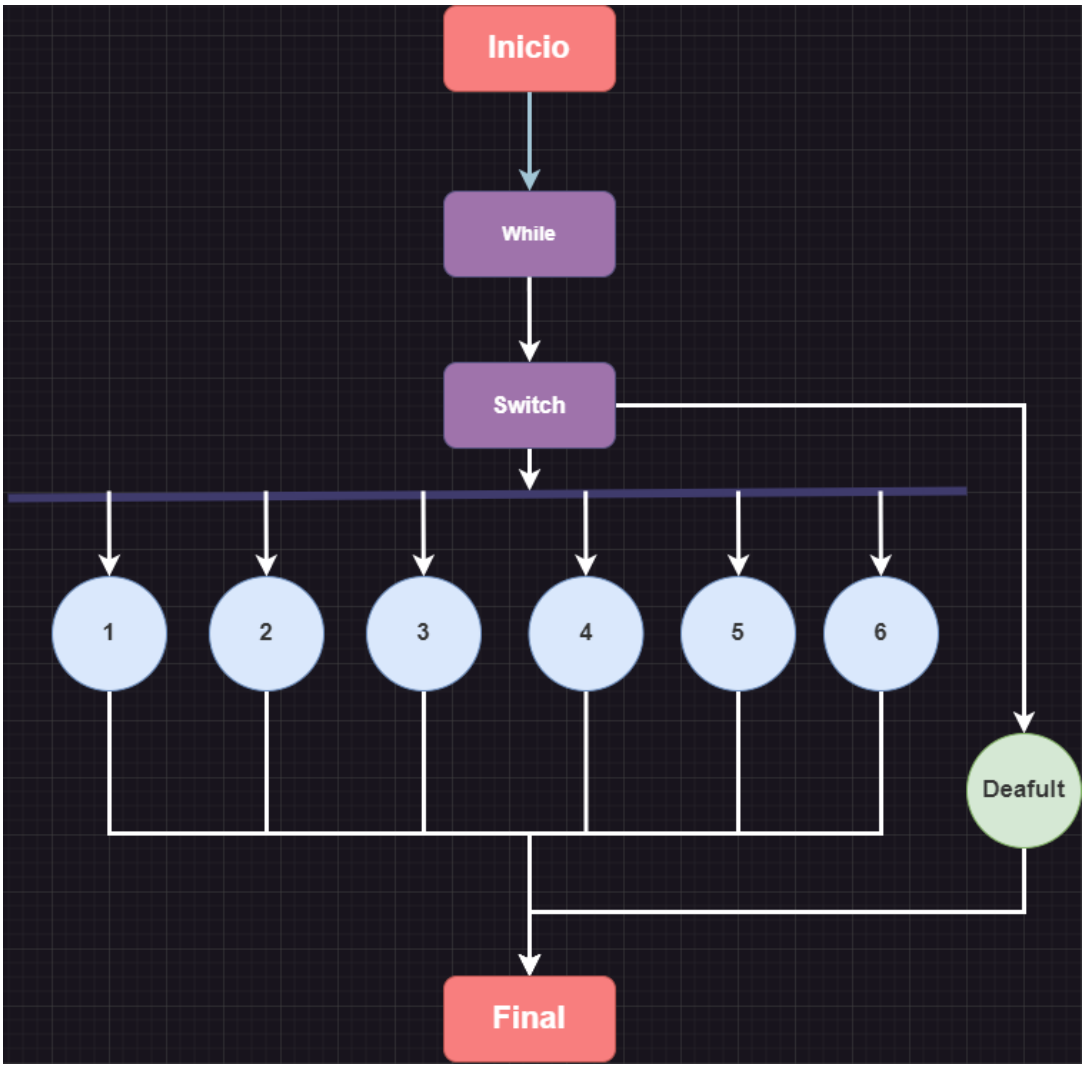
Descripción:

El acceso y desempeño de las funciones del software deberán ser rápidos y fáciles de usar siendo intuitivo y limitando trabas de uso. De ser necesario se buscará optimizar ante el número de errores y el poco desempeño.

Complejidad Ciclomática

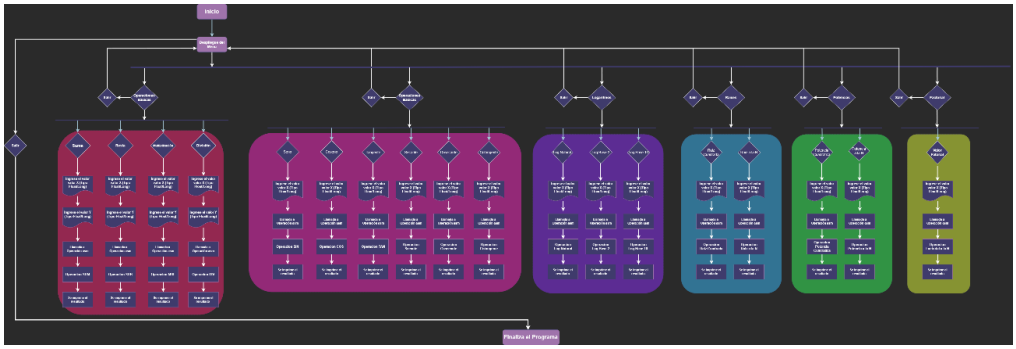
```
10  //MENU PRINCIPAL
11  void MenuProgram::menu()
12  {
13      char option;
14      do
15      {
16          system("cls");
17          system ("color 3");
18          cout << endl << "     ==== [CALCULADORA PROGRAMACION MIXTA] ==== "<<endl;
19          cout << "            (Ensamblador y C++) "<<endl;
20          cout << "        By: Juan Jose Salazar Villegas "<<endl;
21          cout << endl << "   ===========[MENU PRINCIPAL]=======:"<<endl;
22          cout << "    [1] Operaciones basicas          "<<endl;
23          cout << "    [2] Operaciones trigonometricas  "<<endl;
24          cout << "    [3] Logaritmos                   "<<endl;
25          cout << "    [4] Raices                       "<<endl;
26          cout << "    [5] Factorial                    "<<endl;
27          cout << "    [6] Potencias                    "<<endl;
28          cout << "    [0] Salir                        "<<endl;
29          cout << "   ================================="<<endl;
30          cout << "   [Ingrese una Opcion]: ";
31
32
33          cin >> option;
34          switch (option)
35          {
36          case '1':
37              selectOperationBasics();
38              break;
39          case '2':
40              selectTrigonometircFunction();
41              break;
42          case '3':
43              selectLogaritm();
44              break;
45          case '4':
46              enterDataRoots();
47              break;
48          case '5':
49              enterDataFactorial();
50              break;
51          case '6':
52              enterDataPow();
53              break;
54          default:
55              break;
56          }
57      }
58      while(option != '0');
59  }
60  //MENU OPERACIONES BASICAS
```

Diagrama



Fórmula Complejidad Ciclomática:

C(V) = Aristas – Nodos + 2
C(V) = 17 – 11 + 2
C(V) = 8



COCOMO

tipo orgánico:

Tamaño	501
Tipo	Organic

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

PM	Organic	Semidetached	Embedded
A_{PM}	2.40	3.00	3.60
B_{PM}	1.05	1.12	1.20

$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

TDEV	Organic	Semidetached	Embedded
A_{TDEV}	2.50	2.50	2.50
B_{TDEV}	0.38	0.35	0.32

$$PM = PM_{nominal} \cdot \Pi_{i=i}^{15} EM_i$$

Parametros elegidos			
A_{PM}	2.40	A_{TDEV}	2.50
B_{PM}	1.05	B_{TDEV}	0.38

Esfuerzo	1.16	MM
Duracion	2.65	Meses
Team	0.44	Por persona

	Esfuerzo	Schedule
Planes y requisitos	0.1	0.3
Diseño	0.1	0.2
Desarrollo	0.7	1.7
Integracion y pruebas	0.4	0.7
Total	1.2	3.0

	Esfuerzo	Schedule
Planes y requisitos	6%	12%
Diseño	7%	8%
Desarrollo	62%	65%
Integracion y pruebas	31%	27%
Total	100%	100%

Tipo semi-acoplado:

Tamaño	501
Tipo	Semidetached

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

PM	Organic	Semidetached	Embedded
A_{PM}	2.40	3.00	3.60
B_{PM}	1.05	1.12	1.20

$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

TDEV	Organic	Semidetached	Embedded
A_{TDEV}	2.50	2.50	2.50
B_{TDEV}	0.38	0.35	0.32

$$PM = PM_{nominal} \cdot \Pi_{i=i}^{15} EM_i$$

Parametros elegidos			
A_{PM}	3.00	A_{TDEV}	2.50
B_{PM}	1.12	B_{TDEV}	0.35

Esfuerzo	1.38	MM
Duracion	2.80	Meses
Team	0.49	Por persona

	Esfuerzo	Schedule
Planes y requisitos	0.1	0.3
Diseño	0.1	0.2
Desarrollo	0.9	1.8
Integracion y pruebas	0.4	0.8
Total	**1.5**	**3.1**

	Esfuerzo	Schedule
Planes y requisitos	6%	12%
Diseño	7%	8%
Desarrollo	62%	65%
Integracion y pruebas	31%	27%
Total	**100%**	**100%**

Tipo empotrado:

Tamaño	501
Tipo	Embedded

$$PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}$$

PM	Organic	Semidetached	Embedded
A_{PM}	2.40	3.00	3.60
B_{PM}	1.05	1.12	1.20

$$TDEV = A_{TDEV}(PM)^{B_{TDEV}}$$

TDEV	Organic	Semidetached	Embedded
A_{TDEV}	2.50	2.50	2.50
B_{TDEV}	0.38	0.35	0.32

$$PM = PM_{nominal} \cdot \Pi_{i=i}^{15} EM_i$$

Parametros elegidos			
A_{PM}	**3.60**	A_{TDEV}	**2.50**
B_{PM}	**1.20**	B_{TDEV}	**0.32**

Esfuerzo	1.57	MM
Duracion	2.89	Meses
Team	0.54	Por persona

	Esfuerzo	Schedule
Planes y requisitos	0.1	0.3
Diseño	0.1	0.2
Desarrollo	1.0	1.9
Integracion y pruebas	0.5	0.8
Total	**1.7**	**3.2**

	Esfuerzo	Schedule
Planes y requisitos	6%	12%
Diseño	7%	8%
Desarrollo	62%	65%
Integracion y pruebas	31%	27%
Total	**100%**	**100%**

Pruebas Caja Negra y Caja Blanca

Caja Negra:

```
===[ OPERACIONES TRIGONOMETRICAS ]===
[1] Seno x
[2] Coseno x
[3] Tangente x
[4] Secante x
[5] CoSecante x
[6] CoTangente x
[0] Salir

[Ingrese una Opcion]:
```

Caja Blanca (Logaritmo Natural):

```
===[ LOGARITMOS ]===
[1] Logaritmo Natural
[2] Logaritmo base 2
[3] Logaritmo base 10
[0] Salir

[Ingrese una Opcion]:
```

```
 ===[ LOGARITMO NATURAL ]===
Ingresa el valor de x: 55
El logaritmo natural de x es: 4.00733
Presione una tecla para continuar . . .
```