Adrian Sanchez Chacon
Sergio Lechuga Márquez
Javier Rodríguez López

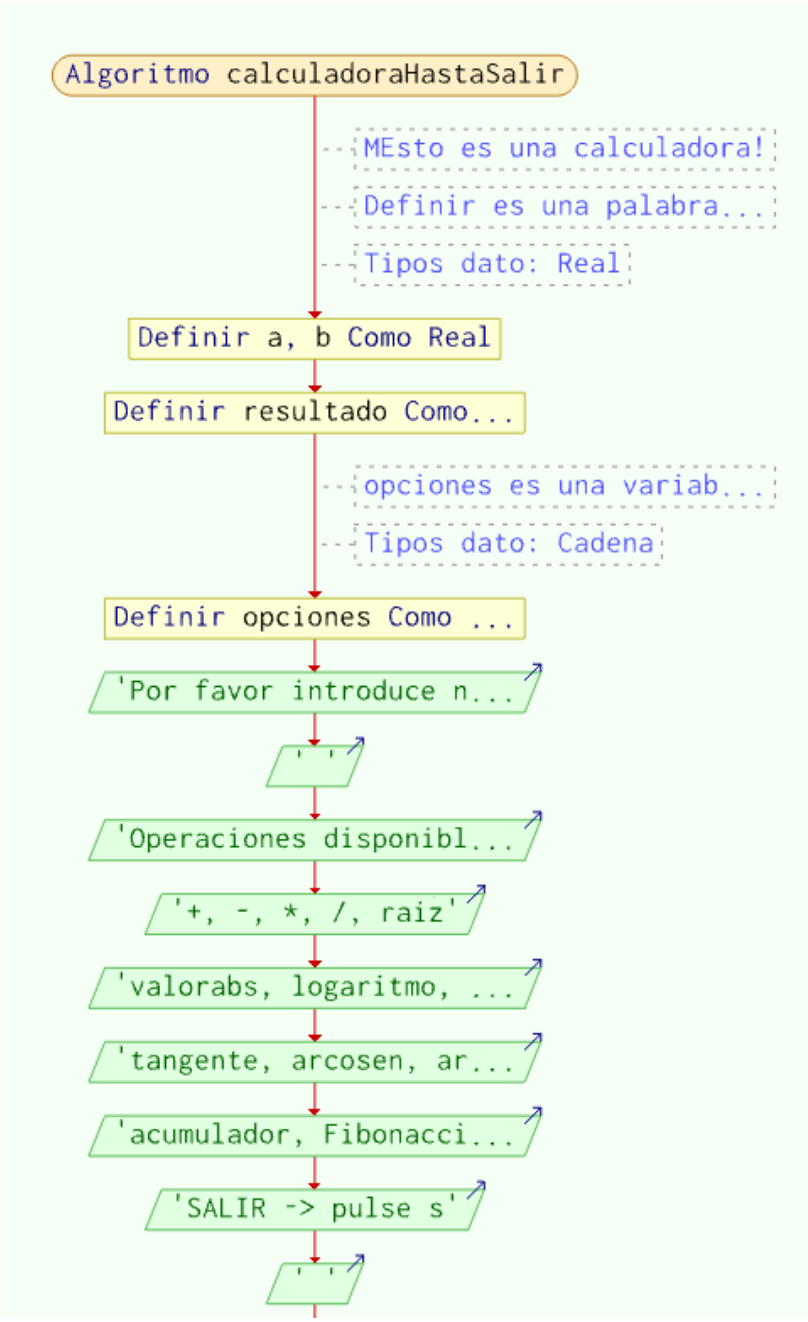
PROYECTO PROGRAMACIÓN UD1

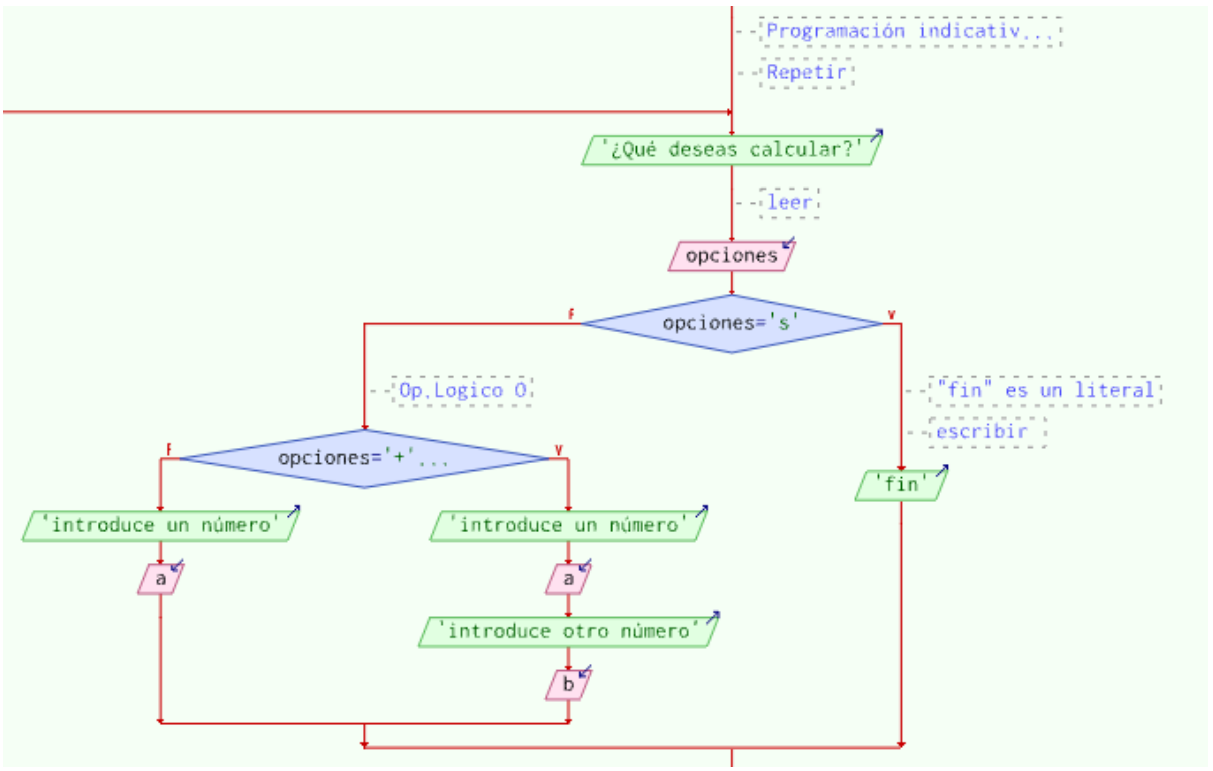
Índice

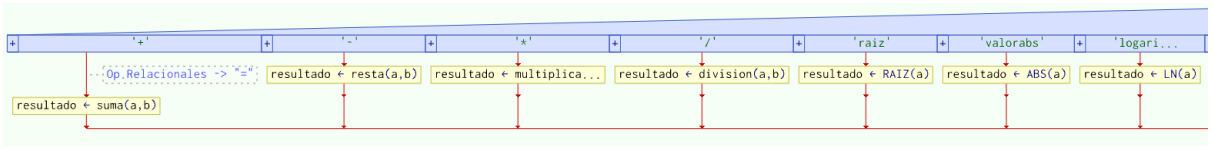
1. Diagrama de flujo
   1.1 Logaritmo general
   1.2 Funciones
      1.2.1 Suma
      1.2.2 Resta
      1.2.3 Multiplicación
      1.2.4 División
      1.2.5 Acumulador
      1.2.6 Sucesión Fibonacci
      1.2.7 Número mayor
2. Código fuente
3. Cohesión del código
4. Acoplamiento del código
5. Elementos del código

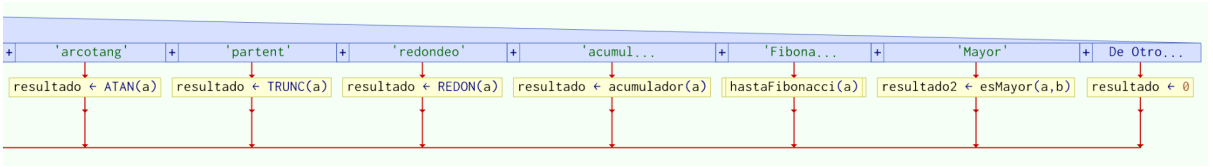
1. Diagrama de flujo:

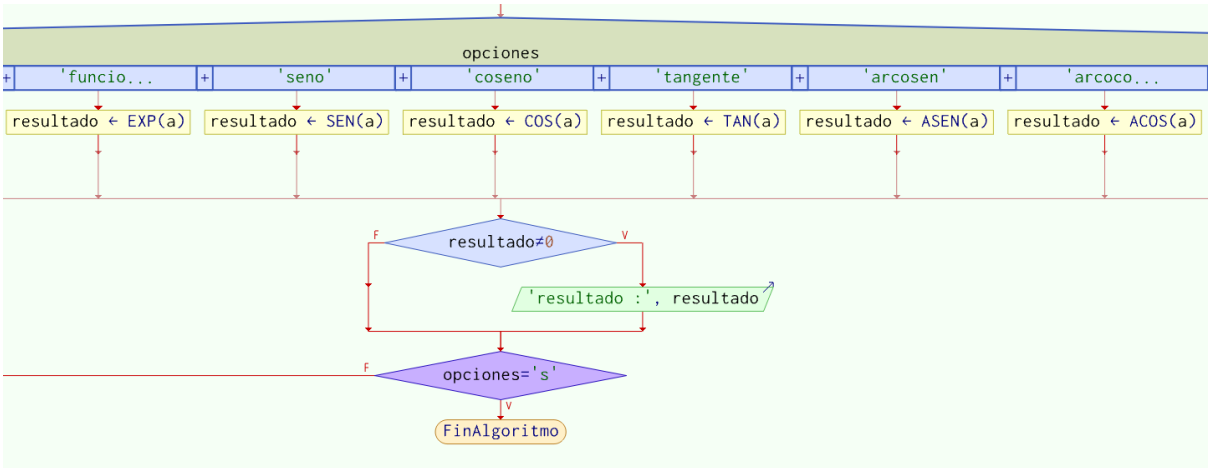
1.1 Logaritmo general




Programación indicativ...
Repetir
'¿Qué deseas calcular?'
leer
opciones
opciones='s'
F
V
Op.Logico O
"fin" es un literal
escribir
opciones='+'...
'fin'
'introduce un número'
'introduce un número'
a
a
'introduce otro número'
b

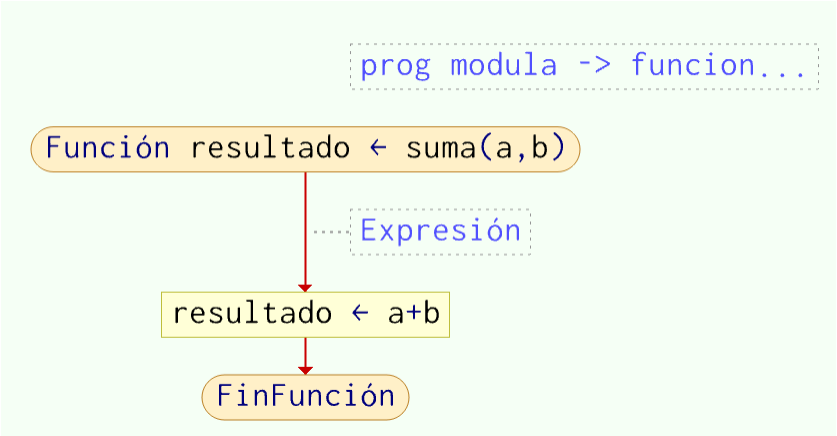

'+'
'-'
'*'
'/'
'raiz'
'valorabs'
'logari...
Op.Relacionales -> "="
resultado ← resta(a,b)
resultado ← multiplica...
resultado ← division(a,b)
resultado ← RAIZ(a)
resultado ← ABS(a)
resultado ← LN(a)
resultado ← suma(a,b)


'arcotang'
'partent'
'redondeo'
'acumul...
'Fibona...
'Mayor'
De Otro...
resultado ← ATAN(a)
resultado ← TRUNC(a)
resultado ← REDON(a)
resultado ← acumulador(a)
hastaFibonacci(a)
resultado2 ← esMayor(a,b)
resultado ← 0

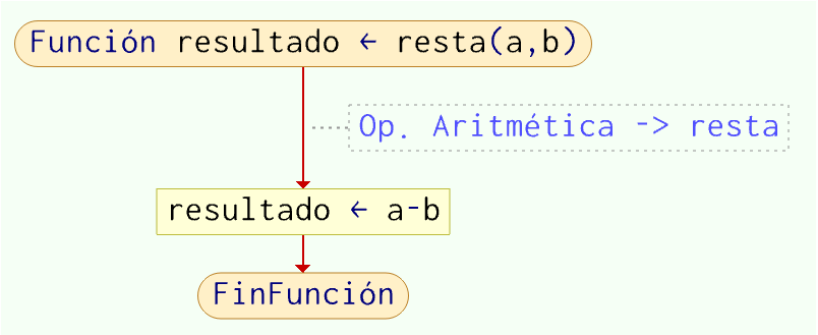

opciones
'funcio...
'seno'
'coseno'
'tangente'
'arcosen'
'arcoco...
resultado ← EXP(a)
resultado ← SEN(a)
resultado ← COS(a)
resultado ← TAN(a)
resultado ← ASEN(a)
resultado ← ACOS(a)
resultado≠0
F
V
'resultado :', resultado
opciones='s'
FinAlgoritmo

1.2 Funciones:

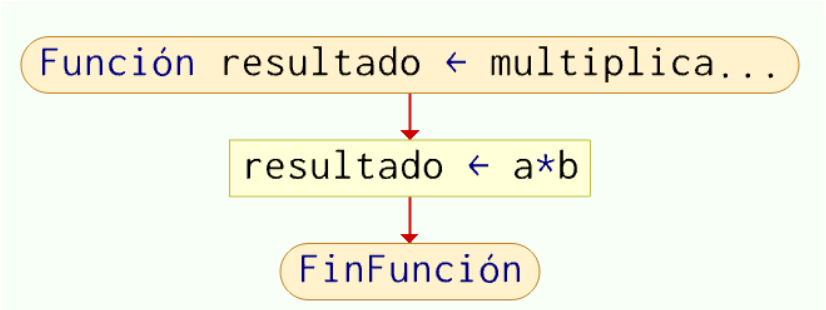
1.2.1 Suma:



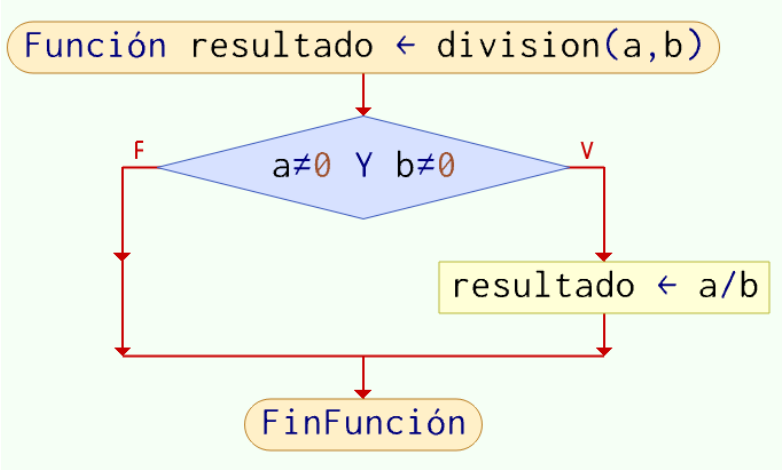
1.2.2 Resta:



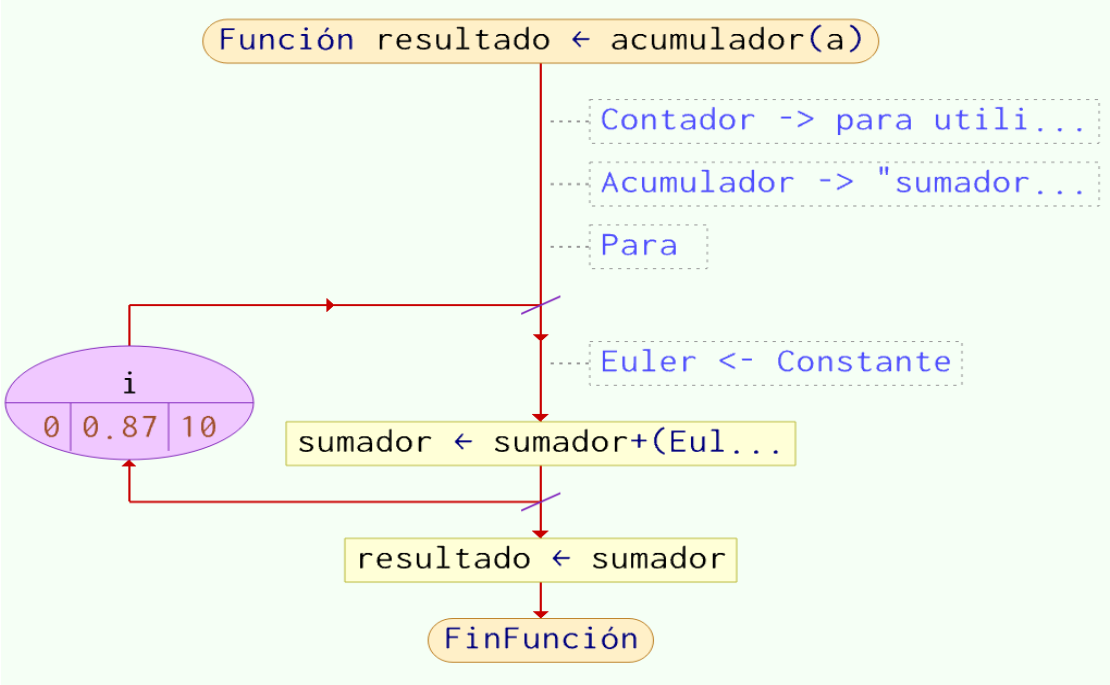
1.2.3 Multiplicación:



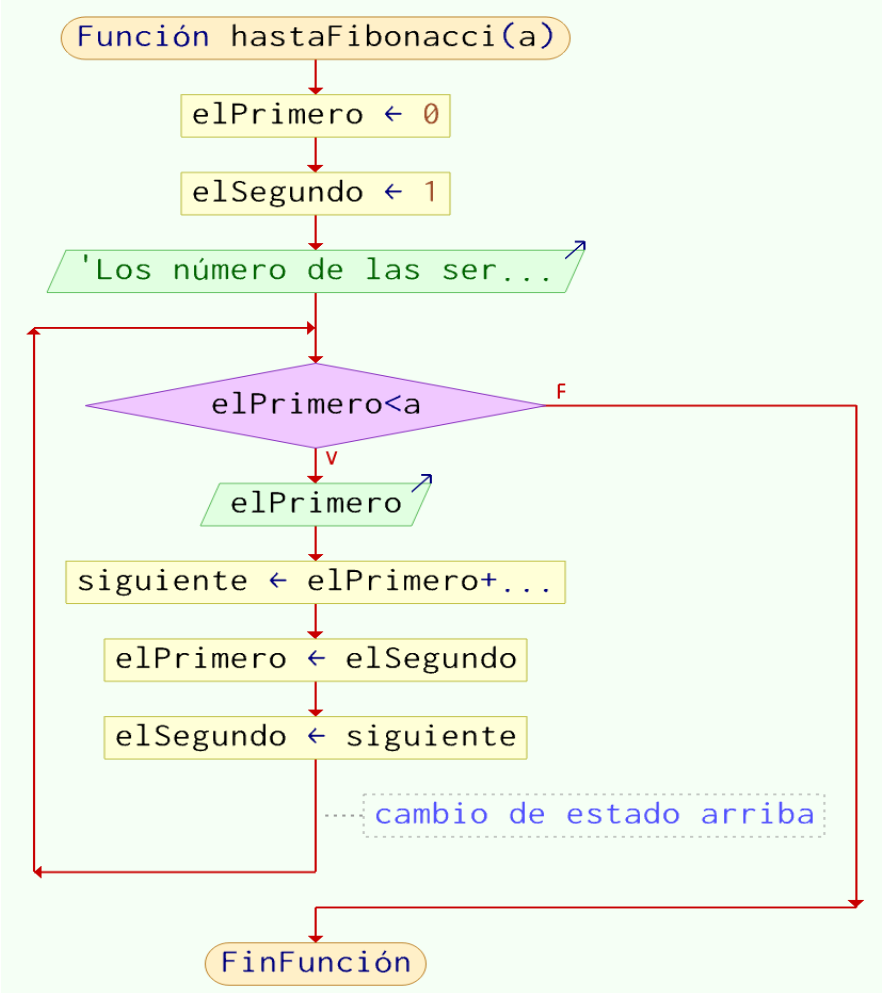
1.2.4 División:



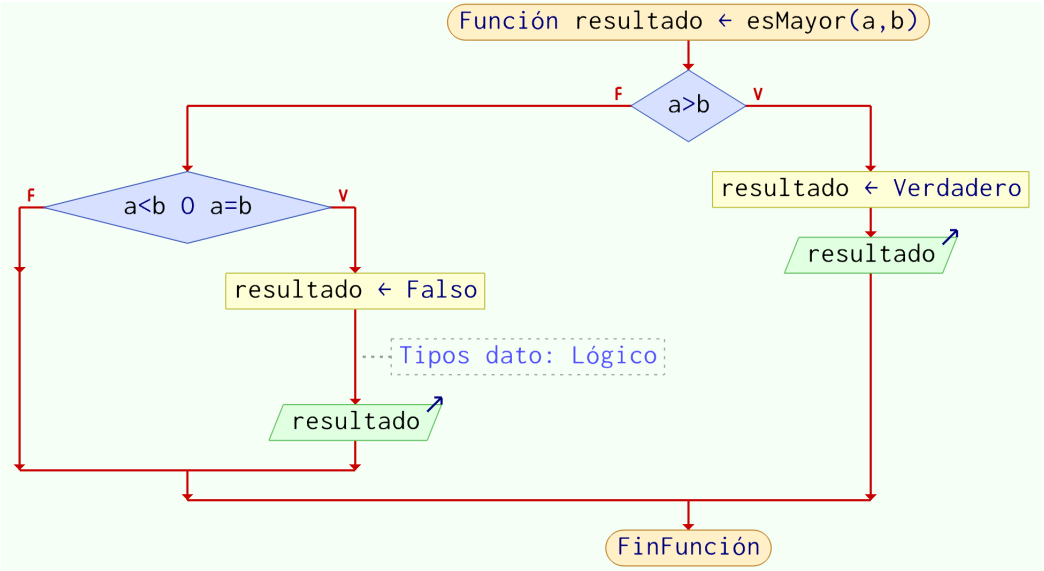
1.2.5 Acumulador:



1.2.6 Contar hasta un número n en la sucesión de Fibonacci:



1.2.7 Qué número es mayor:



2. Código fuente:

```
Algoritmo calculadoraHastaSalir
    //¡Esto es una calculadora!

    //Definir es una palabra reservada
    //Tipos dato: Real
    Definir a, b Como Real
    Definir resultado Como Real
    //opciones es una variable, aquí se define su tipo
    //Tipos dato: Cadena
    Definir opciones como cadena

    Escribir "Por favor introduce números positivos, que no sean 0"
    Escribir " "
    Escribir "Operaciones disponibles: "
    Escribir "+, -, *, /, raiz"
    Escribir  "valorabs, logaritmo, funcionexp, seno, coseno"
    Escribir  "tangente, arcosen, arcotang, redondeo"
    Escribir "acumulador, Fibonacci, Mayor"
    Escribir "SALIR -> pulse s"
    Escribir " "
```

```
    //Programación indicativa -> iteración -> Repetir
    // Repetir
    Repetir
        Escribir "¿Qué deseas calcular?"
        //leer
        leer opciones
        si opciones = "s"
            // "fin" es un literal
            //escribir
            escribir "fin"
        sino
            // Op.Logico O
            si opciones="+" o opciones="-" o opciones="*" o opciones="/" o opciones="Mayor" entonces
                Escribir "introduce un número"
                leer a
                Escribir "introduce otro número"
                leer b
            sino
                Escribir "introduce un número"
                leer a
            FinSi
        FinSi
```

```
43          segun opciones hacer
44              caso "+":
45                  //Op.Relacionales -> "="
46                  resultado = suma(a,b)
47              caso "-":
48                  resultado = resta(a,b)
49              caso "*":
50                  resultado = multiplica(a,b)
51              caso "/":
52                  resultado = division(a,b)
53              caso "raiz":
54                  resultado = RAIZ(a)
55              caso "valorabs":
56                  resultado = ABS (a)
57              caso "logaritmo":
58                  resultado = LN(a)
59              caso "funcionexp":
60                  resultado = EXP (a)
61              caso "seno":
62                  resultado = SEN(a)
63              caso "coseno":
64                  resultado = COS(a)
65              caso "tangente":
66                  resultado = TAN(a)
67              caso "arcosen":
68                  resultado = ASEN(a)
69              caso "arcocosen":
70                  resultado = ACOS(a)
71              caso "arcotang":
72                  resultado = ATAN(a)
73              caso "partent":
74                  resultado = TRUNC(a)
75              caso "redondeo":
76                  resultado = REDON(a)
77              caso "acumulador":
78                  resultado = acumulador(a)
79              caso "Fibonacci":
80                  hastaFibonacci(a)
81              caso "Mayor":
82                  resultado2=esMayor(a,b)
83              De Otro Modo:
84                  resultado = 0
85          FinSegun
```

```
86
87          si resultado ≠ 0
88              Escribir "resultado :", resultado
89          FinSi
90
91      Hasta Que opciones = "s"
92
93  FinAlgoritmo
```

```
//prog modula -> funciones y procedimientos
Funcion resultado=suma(a,b)
    //Expresión
    resultado = a + b
FinFuncion

Funcion resultado=resta(a,b)
    //Op. Aritmética -> resta
    resultado = a - b
FinFuncion

Función resultado=multiplica(a,b)
    resultado= a * b
FinFuncion

Funcion resultado=division(a,b)
    si a≠0 Y b≠0 Entonces
        resultado = a/b
    FinSi
FinFuncion

Funcion resultado=acumulador(a)
    //Contador -> para utiliza el contador i hasta i =10
    //Acumulador -> "sumador" ejerce de acumulador
    //Para
    Para i = 0 hasta 10 Con Paso 0.87 Hacer
        //Euler <- Constante
        sumador = sumador + (Euler*a)
    FinPara
    resultado = sumador
FinFuncion

Funcion hastaFibonacci(a)
    elPrimero = 0
    elSegundo = 1
    Escribir "Los número de las serie Fibonacci hasta ", a, " son: "

    Mientras elPrimero < a Hacer
        Escribir elPrimero
        siguiente = elPrimero + elSegundo
        elPrimero = elSegundo
        elSegundo = siguiente
        //cambio de estado arriba

    Fin Mientras
FinFuncion

Funcion resultado=esMayor(a,b)
    si a > b Entonces
        resultado = Verdadero
        Escribir resultado
    SiNo
        si a < b O a=b Entonces
            resultado = Falso
            //Tipos dato: Lógico
            Escribir resultado
        FinSi
    FinSi
FinFuncion
```

3. Explica si tu código está o no cohesionado y por qué. Señala alguna parte a modo de ejemplo.

Cohesión general: La estructura general del código tiene sentido, y las funciones están bastante centradas en realizar operaciones matemáticas concretas. Sin embargo, algunos elementos podrían mejorar la cohesión, como la organización de la entrada de datos o el manejo de errores.

Puntos a revisar:

Entrada de datos repetida:

La solicitud de los números para las operaciones matemáticas se repite dos veces dependiendo de la operación, lo que genera cierta duplicación. Si las operaciones (+, -, *, /) requieren dos números y las demás solo uno, esto podría haberse organizado de manera más eficiente.

Ejemplo:

```
si opciones="+" o opciones="-" o opciones="*" o opciones="/" entonces
    Escribir "introduce un número"
    leer a
    Escribir "introduce otro número"
    leer b
```

4. Explica si tu código está o no acoplado y por qué. Señala alguna parte a modo de ejemplo.

El **acoplamiento** es relativamente **alto** en ciertos puntos, debido a la fuerte dependencia entre las distintas partes del código.

1. Dependencia entre la interfaz de usuario y las funciones matemáticas:

El código solicita los valores de entrada (a y b) en la misma función donde se selecciona la operación. Las funciones matemáticas (suma, resta, multiplica, etc.) están fuertemente acopladas con el flujo principal, ya que el cálculo sólo ocurre después de que la elección de la operación y los números se introducen directamente en el mismo bloque. Esto crea una fuerte dependencia entre la entrada de datos y la lógica de operaciones.

```
si opciones="+" o opciones="-" o opciones="*" o opciones="/" entonces
    Escribir "introduce un número"
    leer a
    Escribir "introduce otro número"
    leer b
```

Esta parte del código está muy acoplada con las operaciones matemáticas. Si quisieras cambiar la forma de leer las entradas o separar la interacción con el usuario de la lógica de cálculo, tendrías que modificar el bloque completo. Esto hace que las partes del código no sean fácilmente reutilizables ni modificables sin afectar a otras áreas.

2. Falta de separación entre operaciones y control del flujo:

Dentro del bloque según opciones, cada operación se maneja de manera independiente, pero cada operación está directamente acoplada al flujo principal. Por ejemplo, el caso + directamente invoca la función suma(a, b).

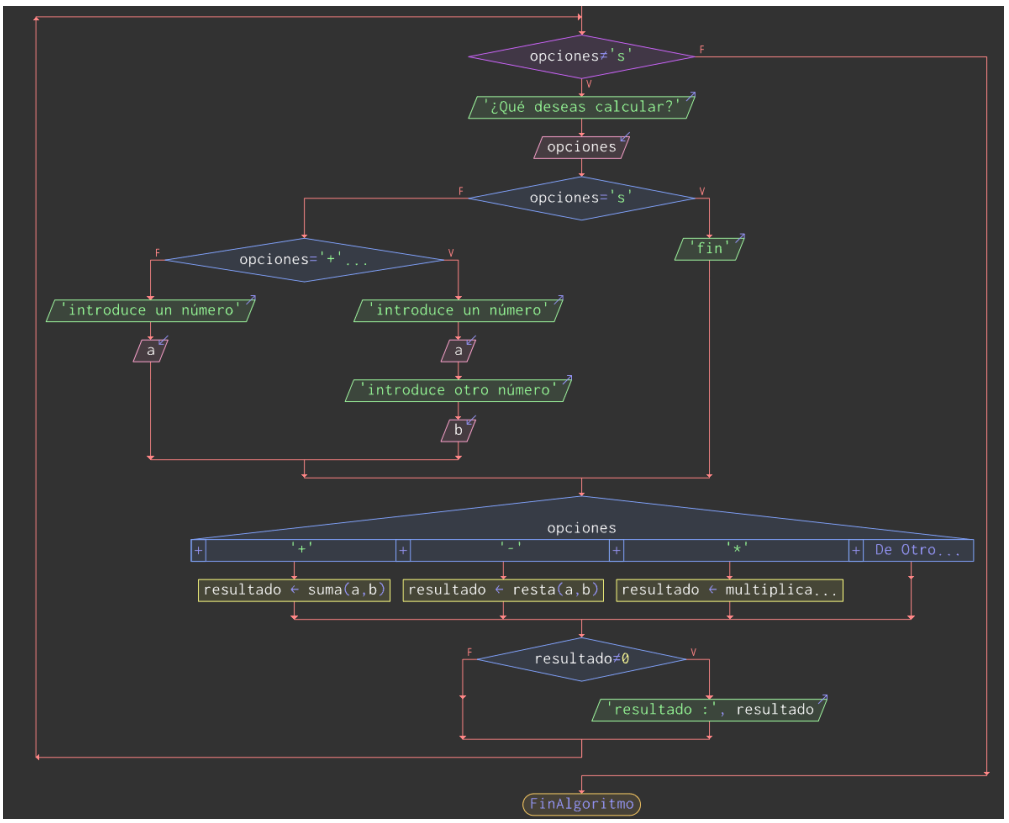
```
caso "+":
    resultado = suma(a,b)
```

Si quisieras cambiar el nombre de la función o su comportamiento, tendrías que modificar el código principal cada vez que utilices una operación. Esto es un acoplamiento fuerte entre las operaciones y el resto del código.

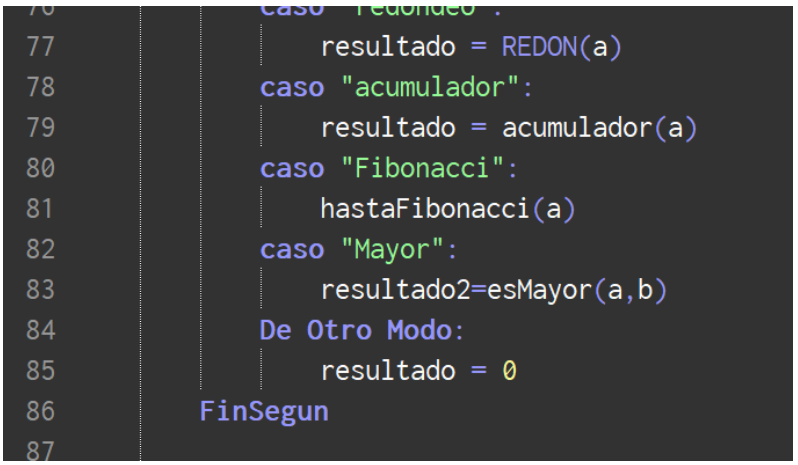
5. Elementos del código:

- **Bucle principal:**

  Continúa solicitando operaciones hasta que el usuario escriba 's' para salir.
  Pide al usuario que ingrese números dependiendo de la operación elegida.
  Realiza la operación correspondiente utilizando un bloque Según.
  Muestra el resultado si es distinto de cero.




```
//Programación indicativa -> iteración -> Mientras
// Mientras
Mientras  opciones ≠ "s"
    Escribir "¿Qué deseas calcular?"
    //leer
    leer opciones
    si opciones = "s"
        // "fin" es un literal
        //escribir
        escribir "fin"
    sino
        // Op.Logico O
        si opciones="+" o opciones="-" o opciones="*" o opciones="/" o opciones="Mayor" entonces
            Escribir "introduce un número"
            leer a
            Escribir "introduce otro número"
            leer b
        sino
            Escribir "introduce un número"
            leer a
        FinSi
    FinSi
    segun opciones hacer
        caso "+":
            //Op.Relacionales -> "="
            resultado = suma(a,b)
        caso "-":
            resultado = resta(a,b)
        caso "*":
```



Funciones

suma(a, b): Devuelve la suma de a y b.

resta(a, b): Devuelve la resta de a menos b.

multiplica(a, b): Devuelve el producto de a y b.

división(a, b): Devuelve la división de a entre b si ambos son diferentes de cero.

acumulador(a): Acumula el producto de a con la constante de Euler, sumando en un bucle.

hastaFibonacci(a): Imprime los números de Fibonacci menores que a.

esMayor(a, b): Compara a y b, y devuelve un booleano indicando si a es mayor.

```
127 Funcion hastaFibonacci(a)
128     elPrimero = 0
129     elSegundo = 1
130     Escribir "Los número de las serie Fibonacci hasta ", a, " son: "
131 
132     Mientras elPrimero < a Hacer
133         Escribir elPrimero
134         siguiente = elPrimero + elSegundo
135         elPrimero = elSegundo
136         elSegundo = siguiente
137         //cambio de estado arriba
138 
```

Final de trabajo de programación.