Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

**Alumno**

Córdova Fernández Karla Lilia

**Unidad de Aprendizaje:** Programación Avanzada

**Profesor**

M. en C. Niels Henrik Navarrete Manzanilla

**Práctica 1**

Uso de “scanf”

**Ciudad de México; a 24 de octubre de 2020.**

**Índice**

Contenido

[INTRODUCCIÓN 4](#_Toc54465412)

[DESARROLLO 5](#_Toc54465413)

[PROGRAMA 1. Secuencia ordenada 5](#_Toc54465414)

[Descripción 5](#_Toc54465415)

[Análisis 5](#_Toc54465416)

[Requerimientos Funcionales 6](#_Toc54465417)

[Requerimientos no Funcionales 6](#_Toc54465418)

[Diagrama de flujo 7](#_Toc54465419)

[Código en C 8](#_Toc54465420)

[Resultados (compilación) 9](#_Toc54465421)

[PROGRAMA 2. Tabla de multiplicar 10](#_Toc54465422)

[Descripción 10](#_Toc54465423)

[Análisis 10](#_Toc54465424)

[Requerimientos Funcionales 11](#_Toc54465425)

[Requerimientos no Funcionales 11](#_Toc54465426)

[Diagrama de flujo 12](#_Toc54465427)

[Código en C 13](#_Toc54465428)

[Resultados (compilación) 14](#_Toc54465429)

[PROGRAMA 3. Mayor y menor 15](#_Toc54465430)

[Descripción 15](#_Toc54465431)

[Análisis 15](#_Toc54465432)

[Requerimientos Funcionales 16](#_Toc54465433)

[Requerimientos no Funcionales 16](#_Toc54465434)

[Diagrama de flujo 17](#_Toc54465435)

[Código en C 18](#_Toc54465436)

[Resultados (compilación) 19](#_Toc54465437)

[PROGRAMA 4. Tres números comparados 20](#_Toc54465438)

[Descripción 20](#_Toc54465439)

[Análisis 20](#_Toc54465440)

[Requerimientos Funcionales 21](#_Toc54465441)

[Requerimientos no Funcionales 21](#_Toc54465442)

[Diagrama de flujo 22](#_Toc54465443)

[Código en C 23](#_Toc54465444)

[Resultados (compilación) 24](#_Toc54465445)

[PROGRAMA 5 División con restas 25](#_Toc54465446)

[Descripción 25](#_Toc54465447)

[Análisis 25](#_Toc54465448)

[Requerimientos Funcionales 26](#_Toc54465449)

[Requerimientos no Funcionales 26](#_Toc54465450)

[Diagrama de flujo 27](#_Toc54465451)

[Código en C 28](#_Toc54465452)

[Resultados (compilación) 29](#_Toc54465453)

[CONCLUSIONES 30](#_Toc54465454)

|  |
| --- |
| INTRODUCCIÓN |

La función scanf en C sirve para leer los datos de entrada y almacenarlos en los argumentos que se pasan de referencia. La estructura de ésta función es scanf( “% tipo de dato ”, &variable );

Para el llenado de arreglos, también se puede utilizar dentro de un for o for anidados dependiendo si se trata de un array unidimensional o bidimensional.

En ésta práctica se utilizará para el traspaso de datos numéricos desde teclado, combinándolo con funciones de ciclos, condiciones, ternas, método burbuja y otros métodos vistos en clase.

|  |
| --- |
| DESARROLLO |

# PROGRAMA 1. Secuencia ordenada

### Descripción

Crear un programa que reciba una secuencia de números (1,2,3....n), se debe obtener como salida los números ingresados por el usuario de acuerdo al orden (posición de cada uno de los números) y después mostrarlos en orden descendiente (n....3,2,1).

### Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

int longitud; variable que indica la cantidad de números en la secuencia.

float grupo[ longitud ]; arreglo donde se almacenará la secuencia de números.

Salidas:

float grupo [ longitud ];

1. **¿Qué es lo que hará el programa?**

Se solicitarán primero la cantidad de números de la secuencia y los elementos, que se almacenarán en un entero y un arreglo de flotantes.

Se imprimirán todos los elementos conforme fueron guardados.

Usando el método burbuja, los elementos del arreglo se acomodarán en orden descendente (dos for’s anidados, donde si un número en una posición es menor que el siguiente, se intercambian de posición).

Se imprimirá el arreglo en el nuevo orden.

1. **¿Qué espero de salida?**

Imprimir -> float grupo [ longitud ]; en orden de introducción y ordenados descendentemente.

### Requerimientos Funcionales

El programa permite introducir una secuencia de números enteros y flotantes (positivos o negativos) de longitud finita preestablecida por el usuario.

El programa muestra en pantalla la cadena en el orden de introducción y acomodados en orden descendente.

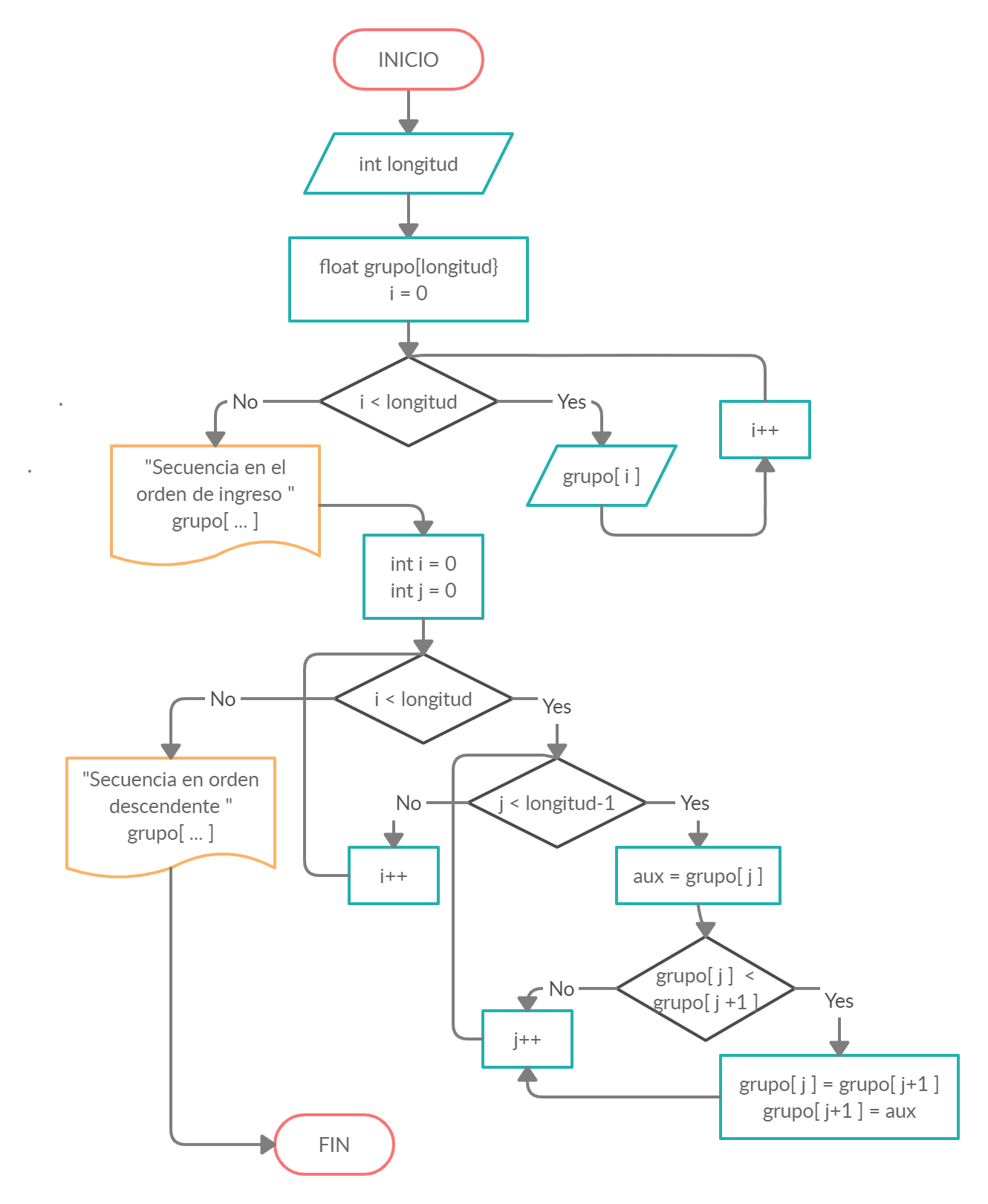
El programa se repite hasta que el usuario de en la opción de salir.

### Requerimientos no Funcionales

El programa no permite introducir una secuencia de números que supere o sea menor a la longitud establecida inicialmente por el usuario.

El programa sólo imprime los números con dos decimales.

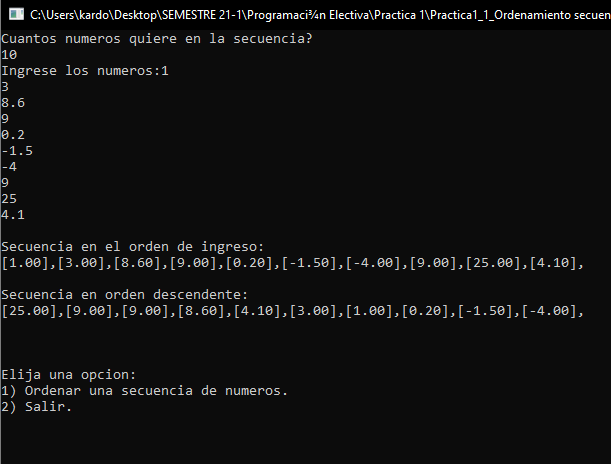
### Diagrama de flujo



### Código en C



### Resultados (compilación)



longitud = 10

Secuencia inicial

grupo[] = { 1 , 3 , 8.6 , 9 , 0.2 , -1.5 , -4 , 9 , 25 , 4.1 }

Secuencia ordenada

grupo[] = { 25 , 9 , 9 , 8.6 , 4.1 , 3 , 1 , 0.2 , -1.5 , -4 }

# PROGRAMA 2. Tabla de multiplicar

### Descripción

Realizar la tabla de multiplicar dado x's número.

### Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

int numero; variable del número que se va a multiplicar.

int maximo; variable del factor que será el máximo en la tabla.

Salidas:

int mult; todas las multiplicaciones del número.

1. **¿Qué es lo que hará el programa?**

Utilizando un ciclo for, el número del que se busca obtener la tabla se multiplicará por todos los factores enteros que van desde el uno hasta el valor introducido como máximo.

1. **¿Qué espero de salida?**

Imprimir -> int mult; el número de entrada multiplicado por todos los valores enteros de 1 hasta el número máximo.

### Requerimientos Funcionales

El programa permite obtener la tabla de multiplicar de cualquier número entero positivo o negativo.

El programa permite decidir hasta qué número desea el usuario que llegue la tabla de multiplicar (la tabla siempre iniciará desde uno).

El programa se repite hasta que el usuario de en la opción de salir.

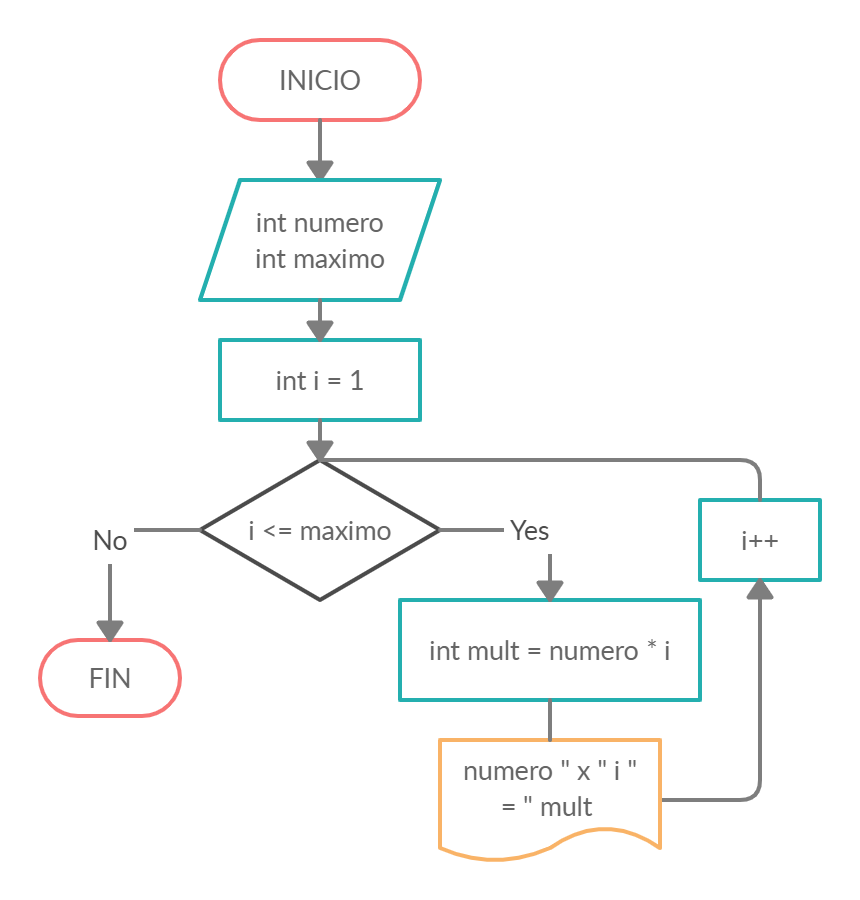
### Requerimientos no Funcionales

El programa no acepta números decimales o flotantes tanto en el número como en el factor máximo.

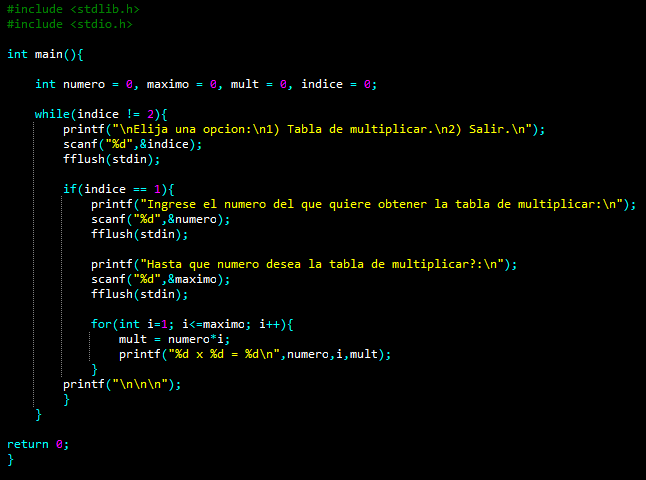
El programa no permite que el factor máximo sea menor a uno.

El programa no pregunta el factor mínimo, siempre empieza desde uno.

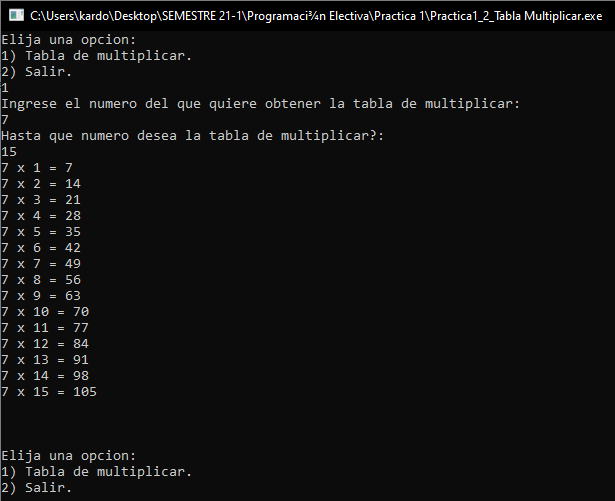
### Diagrama de flujo



### Código en C



### Resultados (compilación)



numero = 7

maximo = 15

mult = 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105

# PROGRAMA 3. Mayor y menor

### Descripción

Crear un programa que obtenga el mayor y el menor de una secuencia de números.

### Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

int numero; variable de la cantidad de elementos de la secuencia.

float secuencia[numero]; arreglo de longitud establecida para almacenar la secuencia.

Salidas:

float mayor; número mayor de la secuencia.

float menor; número menor de la secuencia.

1. **¿Qué es lo que hará el programa?**

Con dos ciclos for anidados, para el número mayor se comparará el dato en esta variable con todas las posiciones del arreglo, si un número en el arreglo es mayor que la variable mayor, adquirirá su valor.

Con el número menor será un proceso similar, a diferencia que cada vez que encuentre un número más pequeño, adquirirá este valor.

1. **¿Qué espero de salida?**

Imprimir -> float mayor, menor; Se imprimirán ambos números en pantalla.

### Requerimientos Funcionales

El programa permite introducir una secuencia de números de longitud finita decidida por el usuario.

El programa acepta números enteros y flotantes, positivos y negativos.

El programa compara y obtiene qué número es mayor y menor.

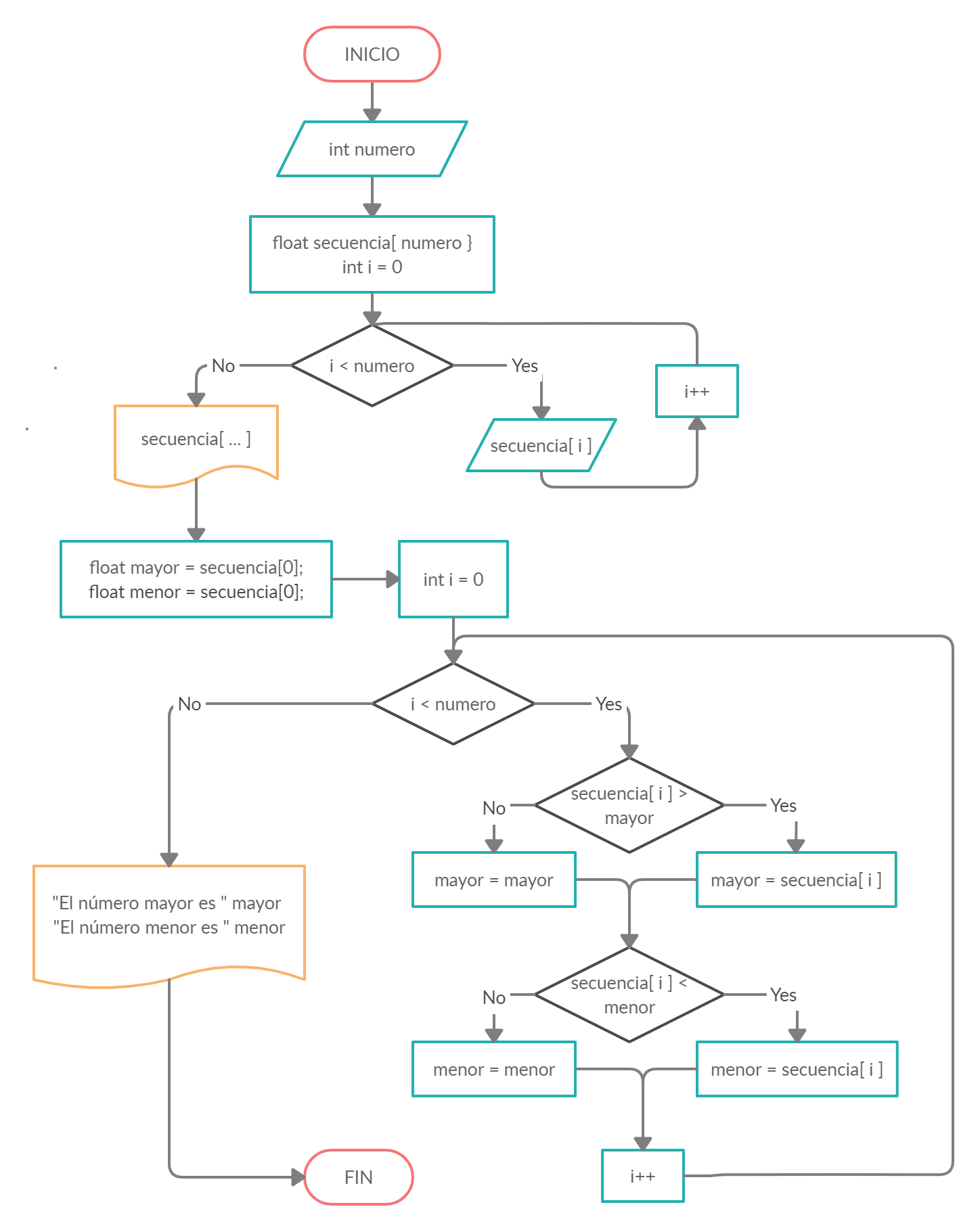
El programa se repite hasta que el usuario de en la opción de salir.

### Requerimientos no Funcionales

El programa no admite una cantidad de números mayor o menor a la establecida con anticipación.

El programa no admite introducir una longitud menor o igual a cero.

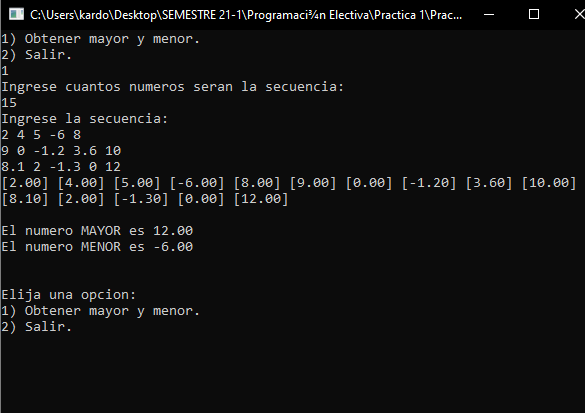
### Diagrama de flujo



### Código en C



### Resultados (compilación)



numero = 15

secuencia[] = {2, 4, 5, -6, 8, 9, 0, -1.2, 3.6, 10, 8.1, 2, -1.3, 0, 12}

mayor = 12

menor = -6

# PROGRAMA 4. Tres números comparados

### Descripción

Diseñar un algoritmo que lea 3 números y encuentre si uno de ellos es la suma de los otros dos.

### Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

float numA;

float numB;

float numC; números a comparar.

1. **¿Qué es lo que hará el programa?**

Con uso de las ternas, se comparará si numA + numB suman numC, si no, se comparará si numA y numC suman numB, en caso de que tampoco, comparará si numB y numC suman numA.

Si ninguno de los casos se cumple, significará que ninguna pareja de números suman al otro.

1. **¿Qué espero de salida?**

Imprimir -> Imprimir qué números suman al otro; Se imprimirán ambos números en pantalla.

### Requerimientos Funcionales

El programa permite introducir tres números enteros y flotantes positivos y negativos.

El programa verifica si la suma de dos números de los introducidos suman al elemento restante.

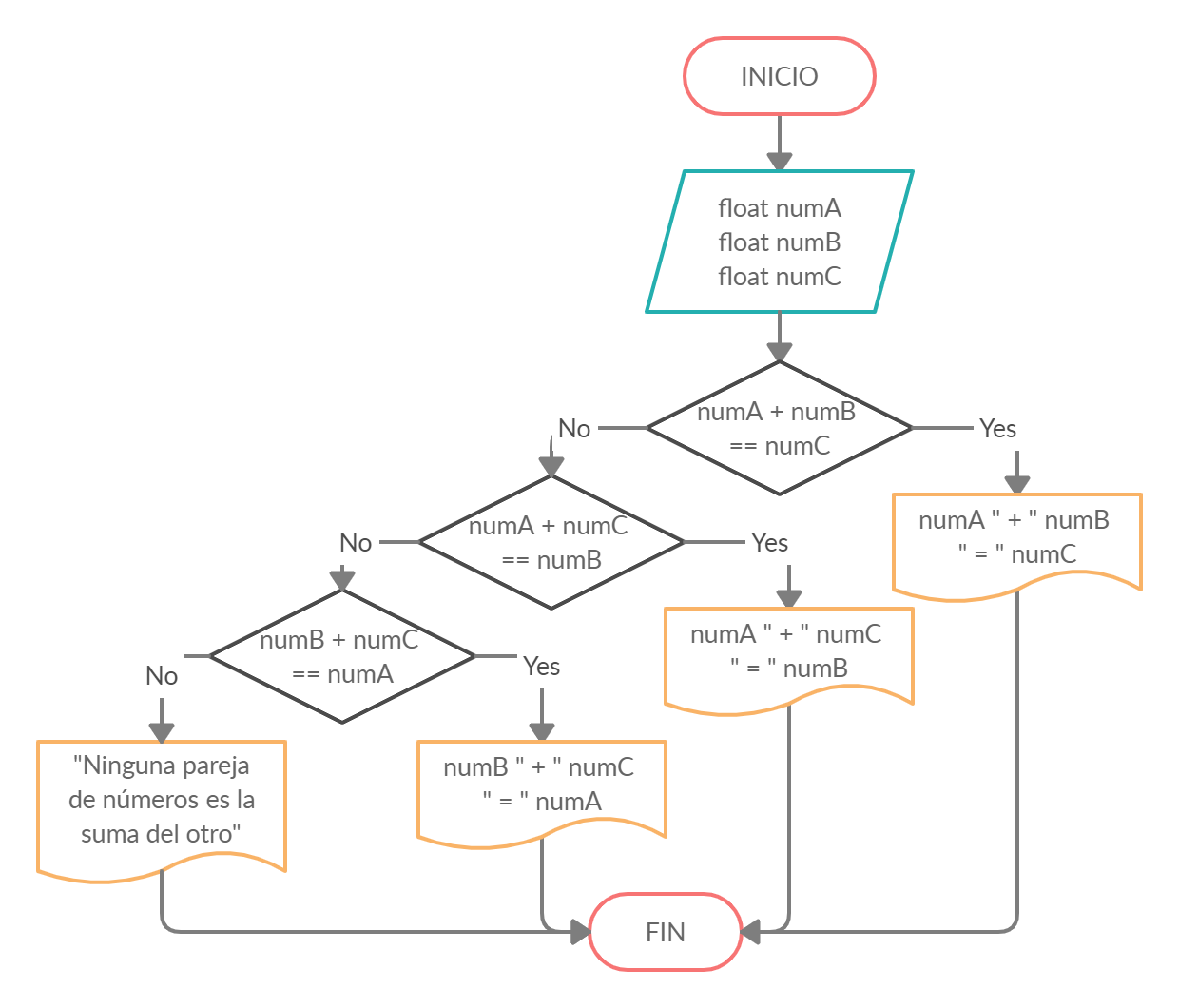
El programa se detiene hasta que el usuario de la opción de salir.

### Requerimientos no Funcionales

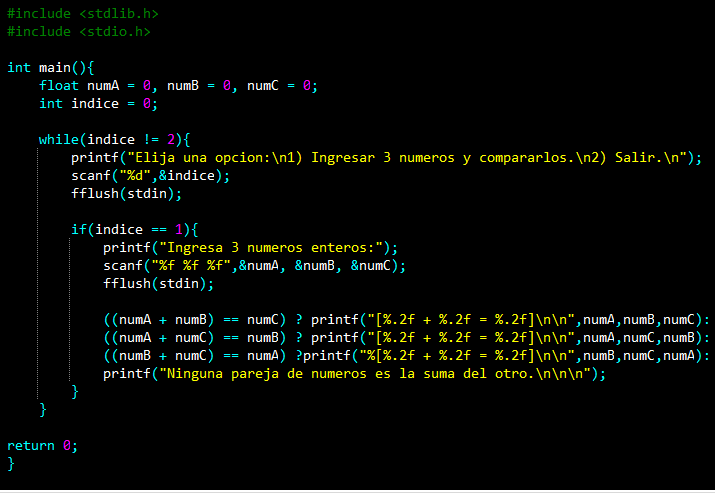
El programa no admite más o menos de tres números.

El programa sólo imprime números con dos decimales.

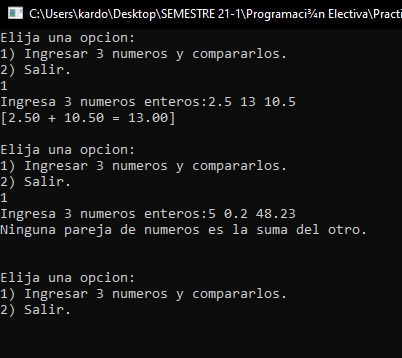
### Diagrama de flujo



### Código en C



### Resultados (compilación)



Caso 1:

numA = 2.5

numB = 13

numC = 10.5

Salida-> “[ 2.5 + 10.5 = 13 ]”

Caso 2:

numA = 5

numB = 0.2

numC = 48.23

Salida-> “Ninguna pareja de números es la suma del otro”

# PROGRAMA 5 División con restas

### Descripción

Realizar un programa que divida dos números dados, resolverlo mediante el método de restas sucesivas.

### Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

int dividendo;

int divisor;

Salidas:

int resultado;

int residuo;

1. **¿Qué es lo que hará el programa?**

Con el método de las sumas, al dividendo se le restará el divisor. La división terminará cuando al residuo de la resta ya no se le puede restar el divisor sin crear valores negativos.

1. **¿Qué espero de salida?**

Imprimir -> resultado, residuo

### Requerimientos Funcionales

El programa acepta números enteros sólo positivos.

El programa realiza la división por el método de las sumas decisivas y el resultado es un entero, con el residuo como otro entero.

### Requerimientos no Funcionales

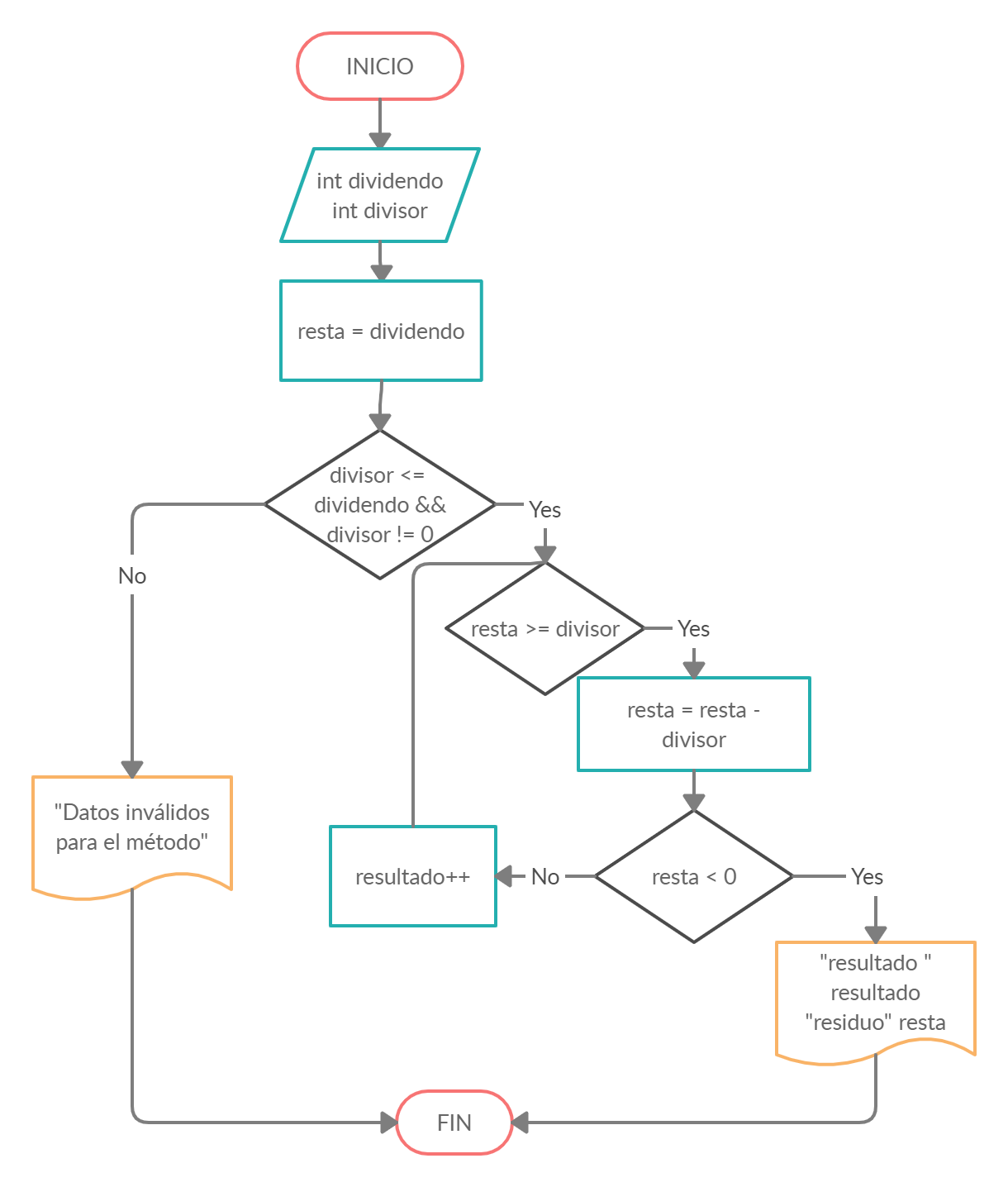
El programa recibe únicamente dos valores, el divisor y el dividendo.

El programa no acepta números negativos, ni decimales o flotantes.

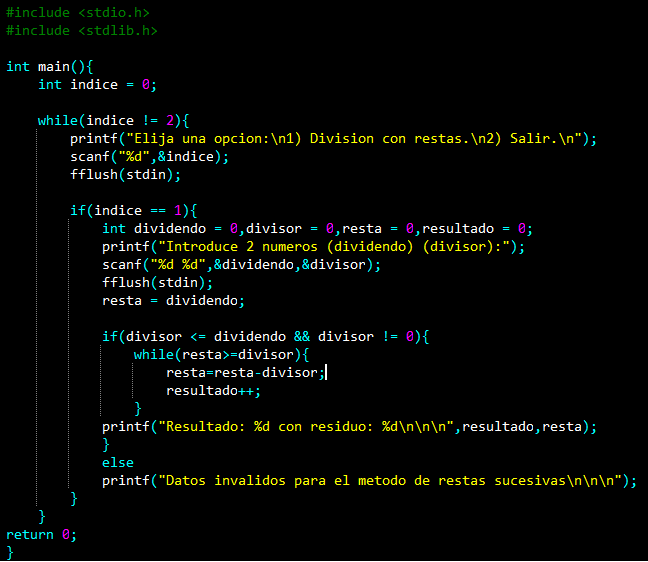
El programa no realiza la división si el divisor es mayor que el dividendo.

El programa no devuelve valores decimales ni flotantes.

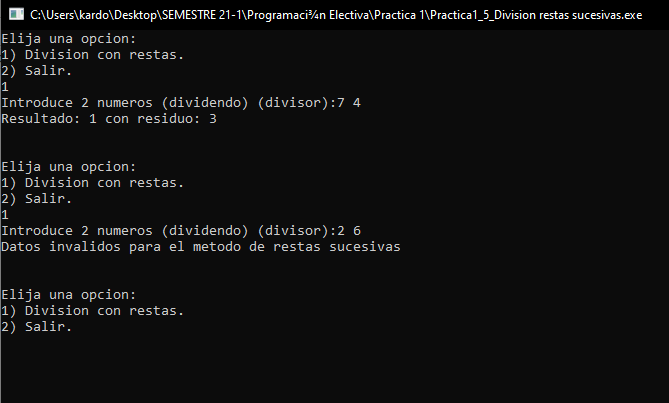
### Diagrama de flujo



### Código en C



### Resultados (compilación)



Caso 1:

dividendo = 7

divisor = 4

resultado = 1

residuo = 3

Caso 2:

dividendo = 2

divisor = 6

No se puede realizar la división

|  |
| --- |
| CONCLUSIONES |

El uso del scanf en esta práctica fue fácil, sólo que, si la introducción del tipo de dato era incorrecta, los programas fallaban o entraban en un ciclo infinito. La función fflush( stdin ) solucionó esos problemas ya que después de un scanf limpia el buffer de entrada.