	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrian Ulises Mercado Martínez

Asignatura: Fundamentos de programación

Grupo: 07

No de Práctica(s): 04

Integrante(s): Arredondo Granados Gerardo
Casillas Herrera Leonardo Didier
Diaz Gonzalez Rivas Angel Iñaqui
Galván Castro Martha Selene

No. de Equipo de cómputo empleado: 04

No. de Lista o Brigada: 04

Semestre: 2022-1

Fecha de entrega: 17 de octubre del 2021

Observaciones:

CALIFICACIÓN: _____

Práctica 4
Diagramas de flujo

Índice **2**
Objetivo **3**
Introducción **3**
Desarrollo **6**
Conclusiones **15**
Conclusiones individuales 15
Referencias **15**

Objetivo:

El alumno elaborará diagramas de flujo que representen soluciones algorítmicas vistas como una serie de acciones que comprendan un proceso.

Introducción:

En la computación los diagramas de flujo son herramientas usadas muy comúnmente, en sí, es un proceso en representación gráfica, es decir, un diagrama de flujo muestra gráficamente el flujo de las acciones a cumplir, se puede entender el por qué el uso de los diagramas de flujo en la programación. Siendo aún más específicos, dentro de las ciencias de la computación, el diagrama de flujo es la representación gráfica del algoritmo, y estos son usados en la etapa posterior a la creación del código, ya que con un diagrama de flujo realizado correctamente, es mucho más sencillo tener un código correcto.

Formas de los diagramas de flujo:

Los diagramas de flujo poseen símbolos los cuales permiten estructurar la solución gráficamente, a continuación, se muestran los símbolos:

1. Se debe tener un inicio y un fin en un diagrama de flujo y estos son representados con un con la palabra inicio y Fin dentro de óvalos

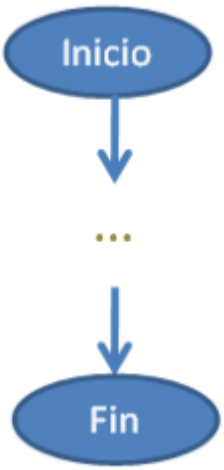


Figura 1. Representación gráfica de Inicio y Fin

2. Se usan flechas para indicar la dirección que se está siguiendo en el diagrama, estas flechas deben de ser horizontales o verticales

Figura 2. Representación gráfica de dirección de flujo

3. Todas estas flechas, sin excepción, deben de estar conectadas a otro símbolo

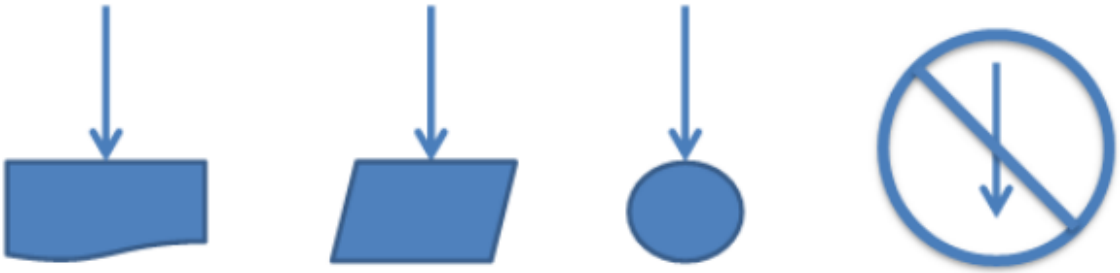


Figura 3. Conexión correcta de las líneas de dirección de flujo

4. El diagrama de flujo debe de estar construido con una dirección de arriba a abajo, y de izquierda a derecha
5. No se debe de utilizar notación especial a algún lenguaje de programación, al diagrama de flujo le debe ser posible programarse en cualquier lenguaje
6. Cuando un diagrama es muy largo, tanto que es necesario el uso de más de una página, se deben de usar una numeración en los símbolos adecuadamente

Conexión entre diferentes páginas. 

7. No es posible que a un símbolo, le llegue más de una flecha

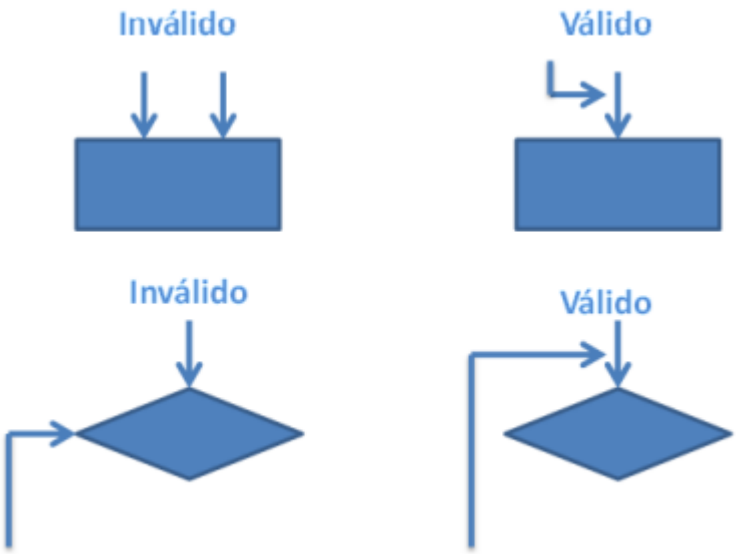


Figura 4. Uso de líneas de dirección

Estructuras de control de flujo

Las estructuras de control de flujo son las que permiten al diagrama la ejecución condicional como la repetición de un conjunto de instrucciones. En los diagramas de flujo existen 3 estructuras de control, las cuales son: secuencial, condicional y repetitivas

1. Estructuras de control secuencial
- Estas estructuras son las sentencias, o también conocidas como declaraciones, que se realizan una a continuación de otra, en el orden en que están escritas
2. Estructuras de control condicionales
- Son las encargadas de evaluar una expresión lógica y dependiendo de ese resultado, se realizará un flujo u otro de instrucciones, se aquí su nombre condicionales
3. Estructuras de control iterativas
- Estas estructuras de control también son llamadas cíclicas, permiten realizar una serie de instrucciones mientras se cumpla la expresión lógica. Dentro de los lenguajes de programación existen dos expresiones cíclicas, **Mientras** y **Hacer-mientras**

La estructura Mientras, primero valida la condición y en el caso de ser verdadera, procesa a realizar las instrucciones de la estructura y regresa a validar la condición; cuando la condición es falsa, se rompe el ciclo y el flujo del diagrama continúa con las estructuras que sigan

La otra expresión Hacer-Mientras, es la que primero realiza las instrucciones y después de hacerlas, válida la expresión lógica, si esta expresión resulta ser verdadera, se realizan de nuevo las acciones; pero cuando resulta ser falsa, se rompe el ciclo y el flujo continúa como vimos en Mientras.

Desarrollo:

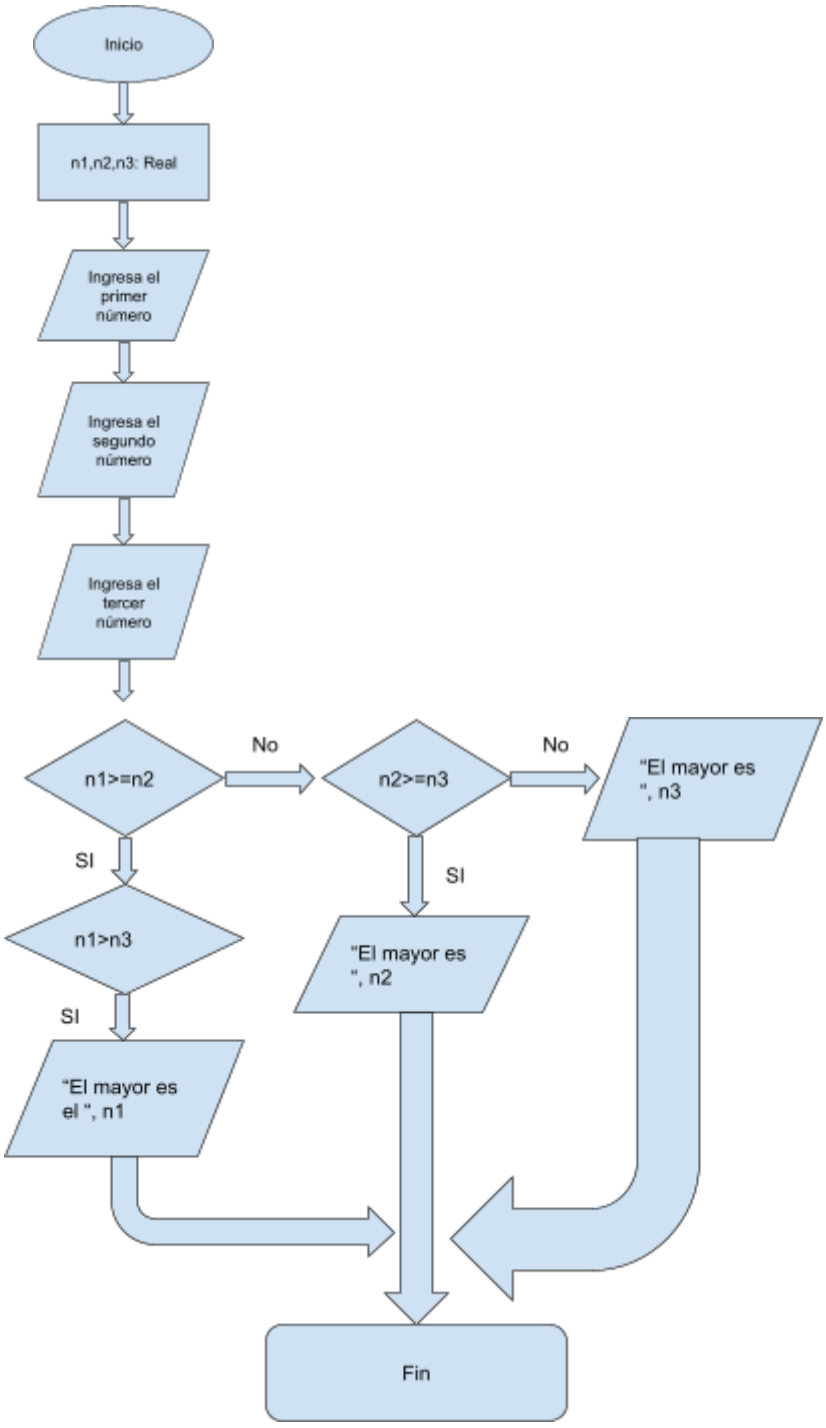
Ejercicios:

- Construye un algoritmo que reciba tres números reales, identifique cuál es el mayor. Considere que los números pueden ser iguales.

Restricción: que el número sea real y entero
Datos de entrada: 3 números reales
Datos de salida: El número mayor
Dominio:Todos los números enteros

1. Solicitar un número real y almacenarlo en una variable.
2. Solicitar otro número real y almacenarlo en otra variable.
3. Solicitar otro número real y almacenarlo en otra variable.
4. Hacer una condición if, si el primer número es mayor al segundo y tercero, imprimir que el número 1 es mayor.
5. Si no se cumple esto, abrir otra condición if en el else del primer if.
6. En este if se validará si el número 2 es mayor al número 3.
7. Si esto es verdad, se imprimirá el número 2.
8. Si no se cumple, llegamos al segundo else.
9. Se imprimirá el número 3.

Diagrama de flujo

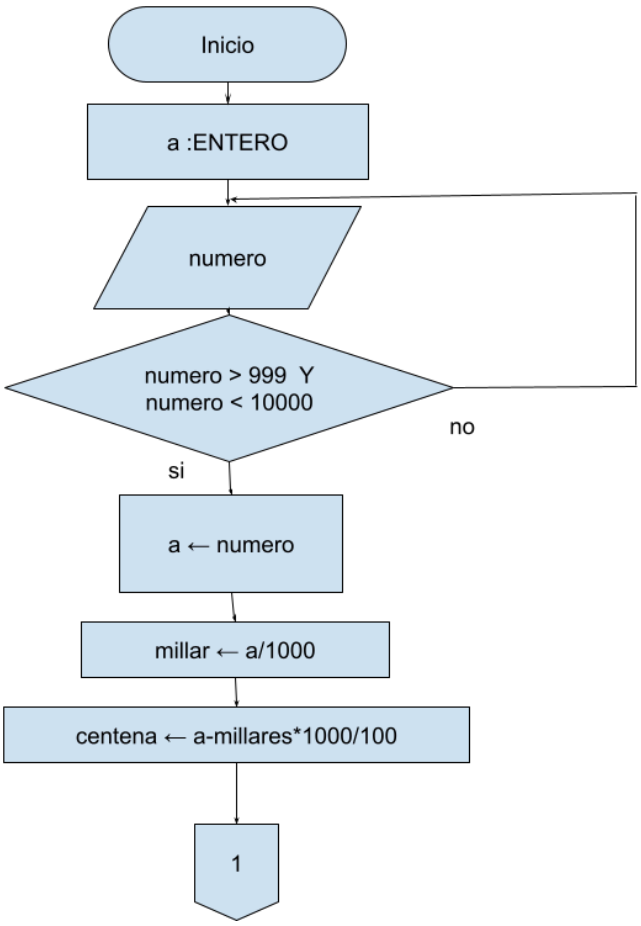


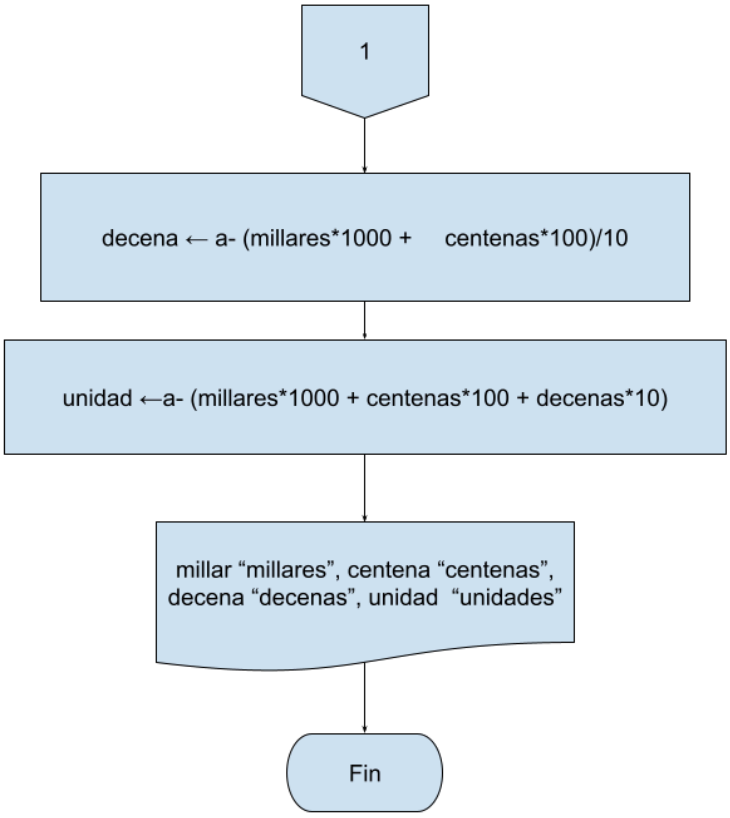
- Elabore un algoritmo que permita imprimir de forma separada los cuatro dígitos que forman un número. **Sugerencia:** Utilice la división entera y el operador de módulo.

Datos de entrada:
Número entero de 4 dígitos
Datos de salida:
4 dígitos que formen un número

Dominio:
Números enteros mayores a 999 y menores a 10 000

1. Solicitar un número de 4 dígitos y almacenarlo en una variable.
2. Si el número entero no tiene 4 cifras, vuelve al paso 1.
3. Separar el número en millares, centenas, decenas, unidades.
4. Imprimir los dígitos formados con separación en el orden millar, centena, decena y unidad.

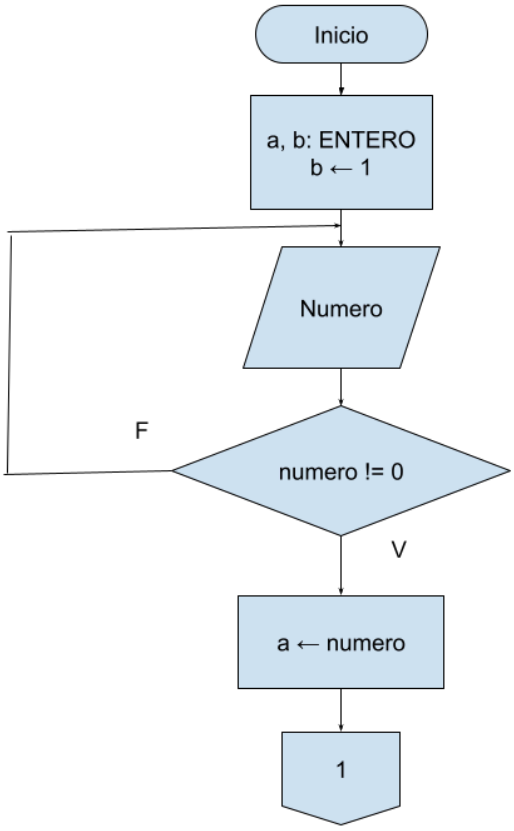


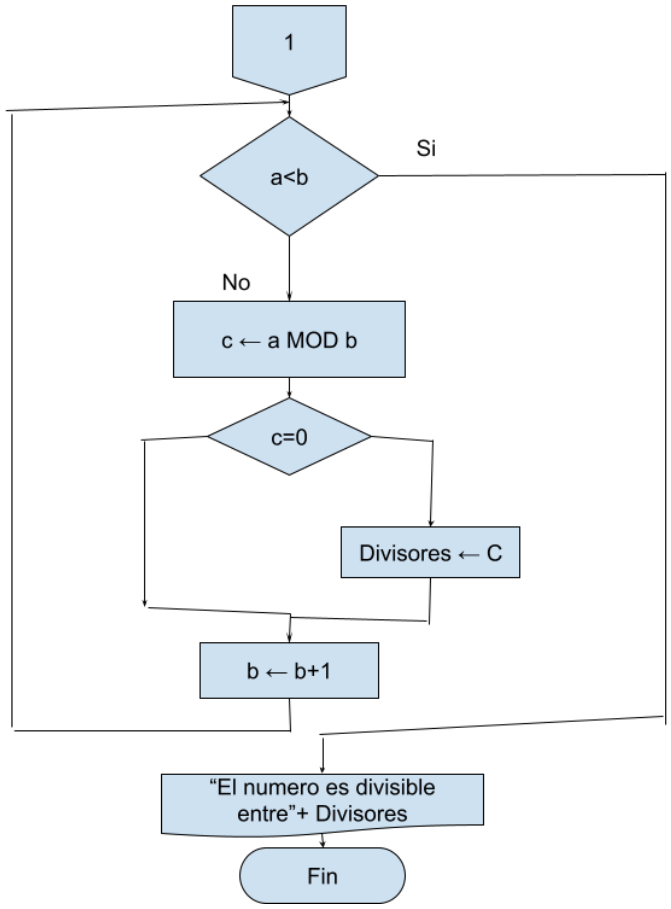


- Elabore un algoritmo para determinar los divisores de un número.

Restricción: que el número no sea negativo o igual a cero
Datos de entrada:
Un número natural que no sea 0
Datos de salida:
Divisores del número entero
Dominio:
Todos los números naturales excepto el 0

1. Solicitar al usuario un número natural que no sea 0 y guardarlo como una variable..
   1.1 Si el número entero es menor a cero, vuelve al paso 1.
2. Dividir el número entre valores menores o iguales al mismo, tales que su residuo sea igual al número cero
3. Todos los números con los que el residuo NO fue igual a cero, descartarlos
4. Imprimir los números para los cuales el residuo de la división es igual a cero


Inicio
a, b: ENTERO
b ← 1
Numero
F
numero != 0
V
a ← numero
1

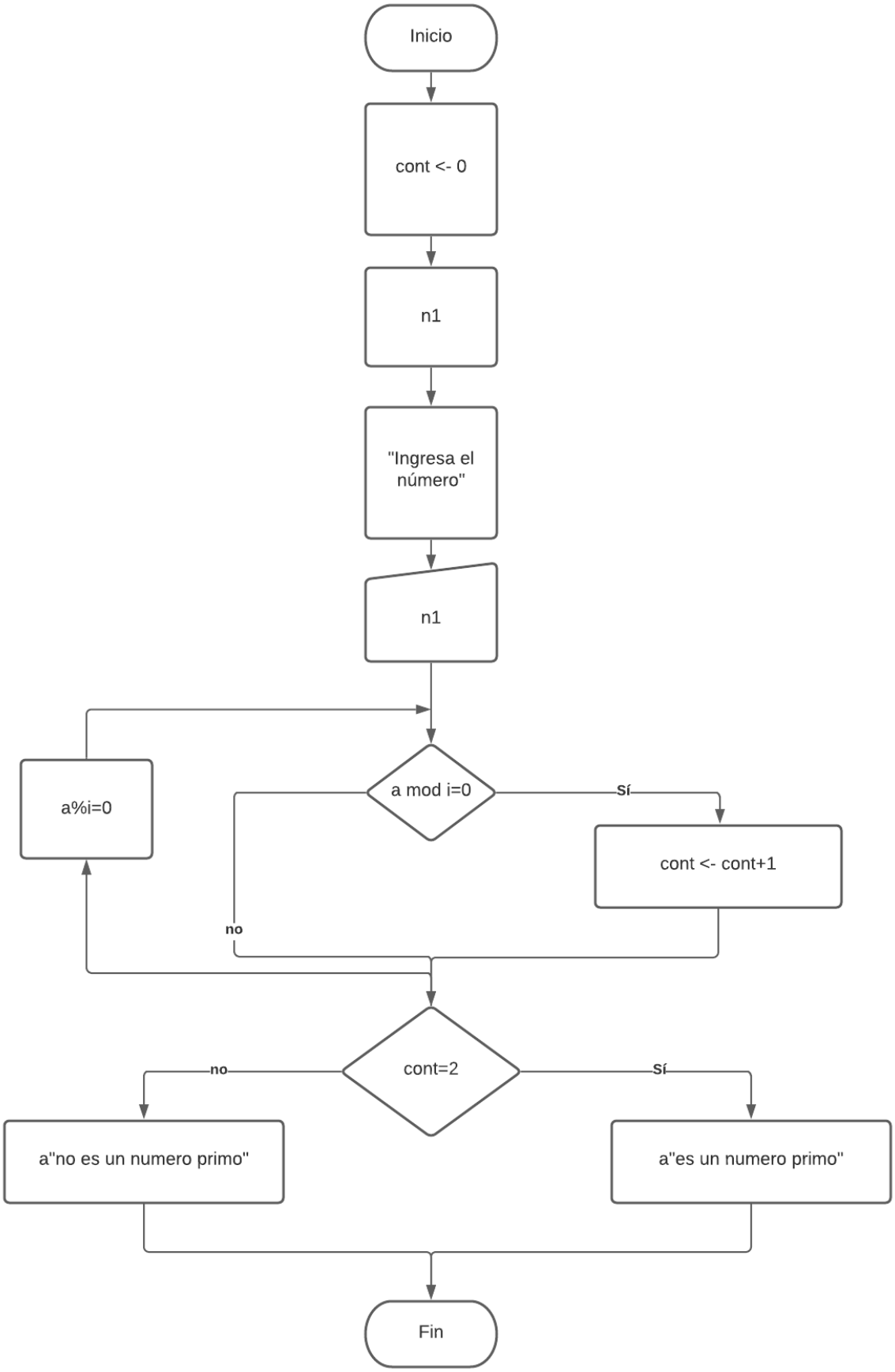


Ejercicios:

- Se dice que un número N es primo si los únicos enteros positivos que lo dividen son exactamente 1 y N. Elabore el algoritmo que de solución al problema.

Restricción: en número de entrada debe de ser entero y positivo
Datos de entrada :
Un número entero
Datos de salida :
El número es primo o no lo es
Dominio:
Los números entero de (0, infinito)

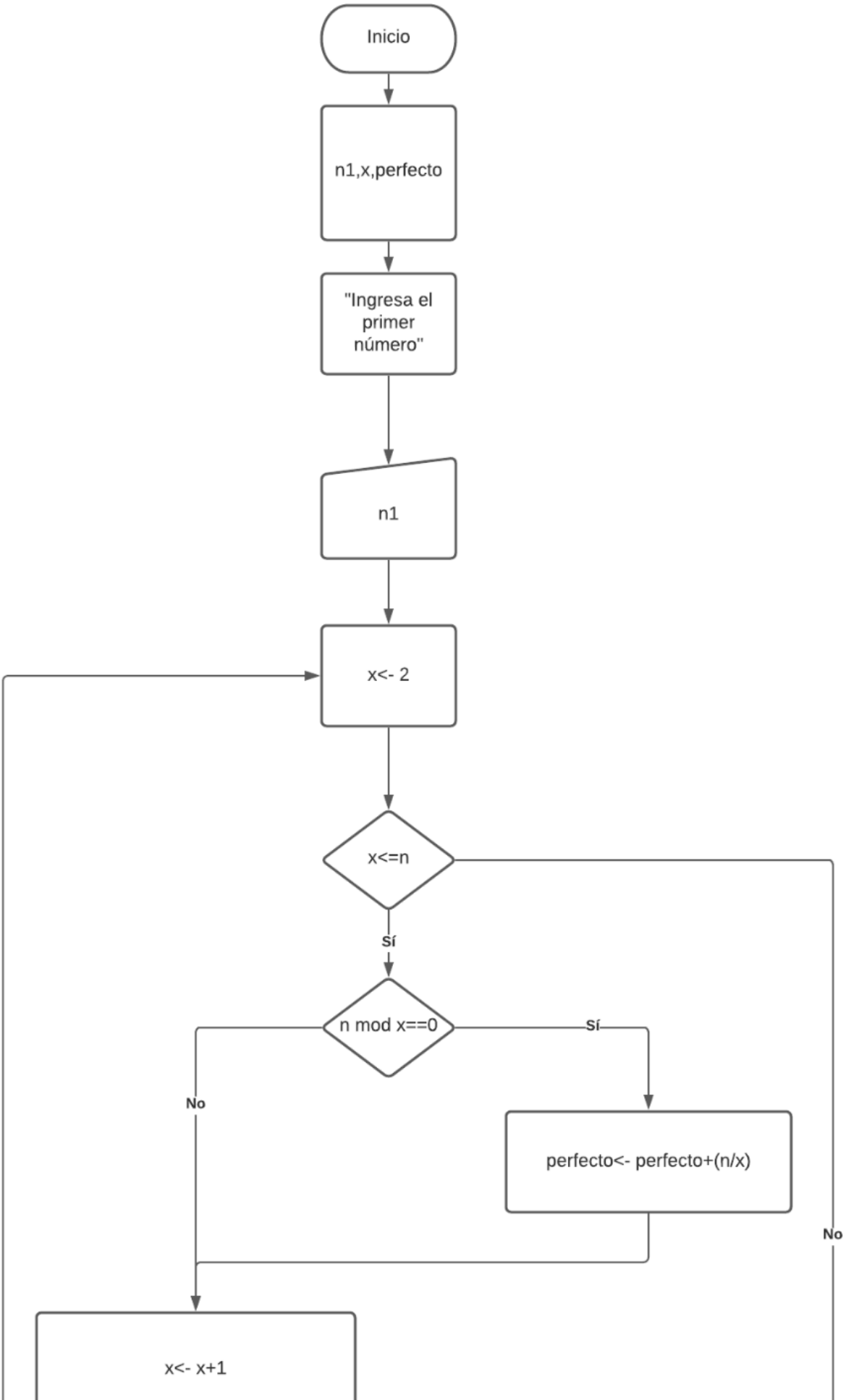
1. Solicitar al usuario un número entero positivo
   1.1 Si el número entero es menor a cero o igual a este, volver al paso 1
2. Almacenar ese número en una variable de tipo entero
3. Empezar a dividir el número empezando con 1 número uno, siguiendo del número 2, siguiendo del número 3, y así sucesivamente hasta alcanzar el mismo número que ha sido ingresado
4. El conjunto de números que arroje como residuo 0, deberán ser el número 1, y el mismo número ingresado
5. Si es que se cumple el paso 4, mostrar en pantalla “El número es primo”
   5.1 De lo contrario, mostrar en pantalla “El número no es primo”


Inicio
cont <- 0
n1
"Ingresa el número"
n1
a mod i=0
Sí
cont <- cont+1
no
a%i=0
cont=2
no
Sí
a"no es un numero primo"
a"es un numero primo"
Fin

- Elaborar un algoritmo que reciba un número entero y calcule los números perfectos que existen entre 1 y el número seleccionado. Un número se considera perfecto si la suma de todos sus divisores son igual a dicho número.

Restricción:
El número debe ser entero y mayor a 1
Datos de entrada:
Un número entero
Datos de salida :
Los números perfectos entre el 1 y el número seleccionado
Dominio:
Todos los números enteros empezando por el 1

1. Solicitar al usuario un número entero positivo
   1.1 Si el número entero es menor a cero, vuelve al paso 1.
2. Almacenar el número en una variable de tipo entero.
3. Dividir el número entre todos los números enteros positivos menores a él.
4. Sumar todos los divisores encontrados.
5. Comparar el resultado de la suma con el número dado, si el resultado fue igual al número dado entonces es un número perfecto, si el resultado fue diferente al número dado entonces no es un número perfecto.


Inicio
n1,x,perfecto
"Ingresa el primer número"
n1
x<- 2
x<=n
Sí
n mod x==0
Sí
No
perfecto<- perfecto+(n/x)
x<- x+1
No

Conclusiones

Conclusión Grupal

Como conclusión se puede decir que a lo largo de la práctica a partir de los algoritmos se crearon los diagramas de flujo utilizando elementos como variables de tipo entero y carácter, los datos de entrada, operadores lógicos como el ←, !=, <, >. También aprendimos cómo distintos símbolos propios de los diagramas de flujo, las estructuras secuenciales, las estructuras selectivas y las estructuras iterativas ya que los distintos problemas ocupan el uso de distintas estructuras.Por lo que se puede decir que cumplimos con el objetivo de practica.

Conclusiones individuales

Arredondo Granados Gerardo: Pienso que fue una buena práctica ya que se cumplieron los objetivos que se habían planteado, el cual era la elaboración de diagramas de flujo, en esta práctica se realizaron varios de estos diagramas. Más aparte de esto, para nosotros, los diagramas de flujo son indispensables para nuestra carrera, así que es bueno tener una buenas bases de lo que es un diagrama de flujo y cómo realizarlo.

Casillas Herrera Leonardo Didier: En la práctica pudimos realizar los diagramas de flujo de los algoritmos que hicimos utilizando la simbología que se debe utilizar al crear los diagramas y logramos verlo como una serie de pasos que necesitamos para un proceso o para resolver un problema, por lo que se lograron todos los objetivos planteados en esta práctica.

Diaz Gonzalez Rivas Angel Iñaqui: En está práctica pudimos observar de una manera más clara los algoritmos al representarlos gráficamente, así como la forma de utilizar los símbolos y formas que tienen su representación dentro del algoritmo. También se utilizaron diferentes tipos de estructuras de control de flujo, es decir, se utilizó la estructura secuencial, iterativa y condicional. Todas estas estructuras nos ayudaron a representar las acciones que se deberían realizar para resolver el problema que se busca solucionar.

Galvan Castro Martha Selene Para concluir se podría decir que se cumplió con el objetivo ya que a lo largo de esta práctica utilizamos nuestros conocimientos en Diagramas de flujo, al aplicarlo ya que con unos algoritmos previamente realizamos hicimos sus Diagramas de flujo lo cual nos hizo encontrarle la utilidad a sus distintos elementos como son sus estructuras, sus variables y sus operadores lógicos. Además que para estos algoritmos se requerían elementos muy variados lo cual favoreció para ampliar nuestro conocimiento sobre Diagramas de flujo.

Referencias

- Metodología de la programación. Osvaldo Cairó, tercera edición, México D.F., Alfaomega 2005.
- Metodología de la programación a través de pseudocódigo. Miguel Ángel Rodríguez Almeida, primera edición, McGraw Hill