Guía 3

Introducción a la programación – Estructuras Iterativas

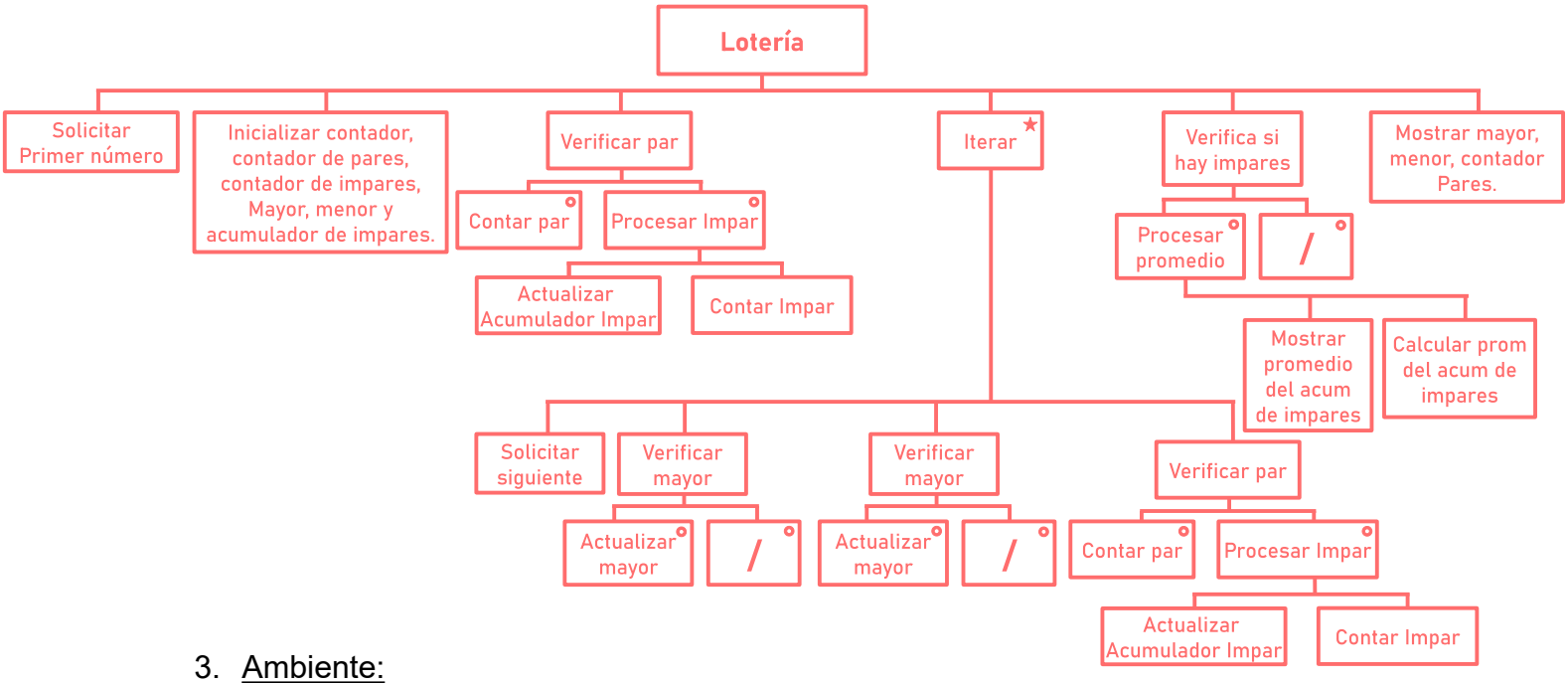
Ejercicio 1) Lotería:

1. Análisis:
   a. Entradas:
      - 20 números de lotería.

   b. Salidas:
      - Promedio de los números impares ingresados.
      - Mayor número de los 20 ingresados.
      - Menor número de los 20 ingresados.
      - Cantidad de números impares sorteados.

   c. Relación:
      - Iterar 20 veces: Pedir número al usuario.
        - Comprobar si es Par o Impar y acumularlos individualmente.
        - Definir número mayor.
        - Definir número menor.

2. Estrategia:



3. Ambiente:

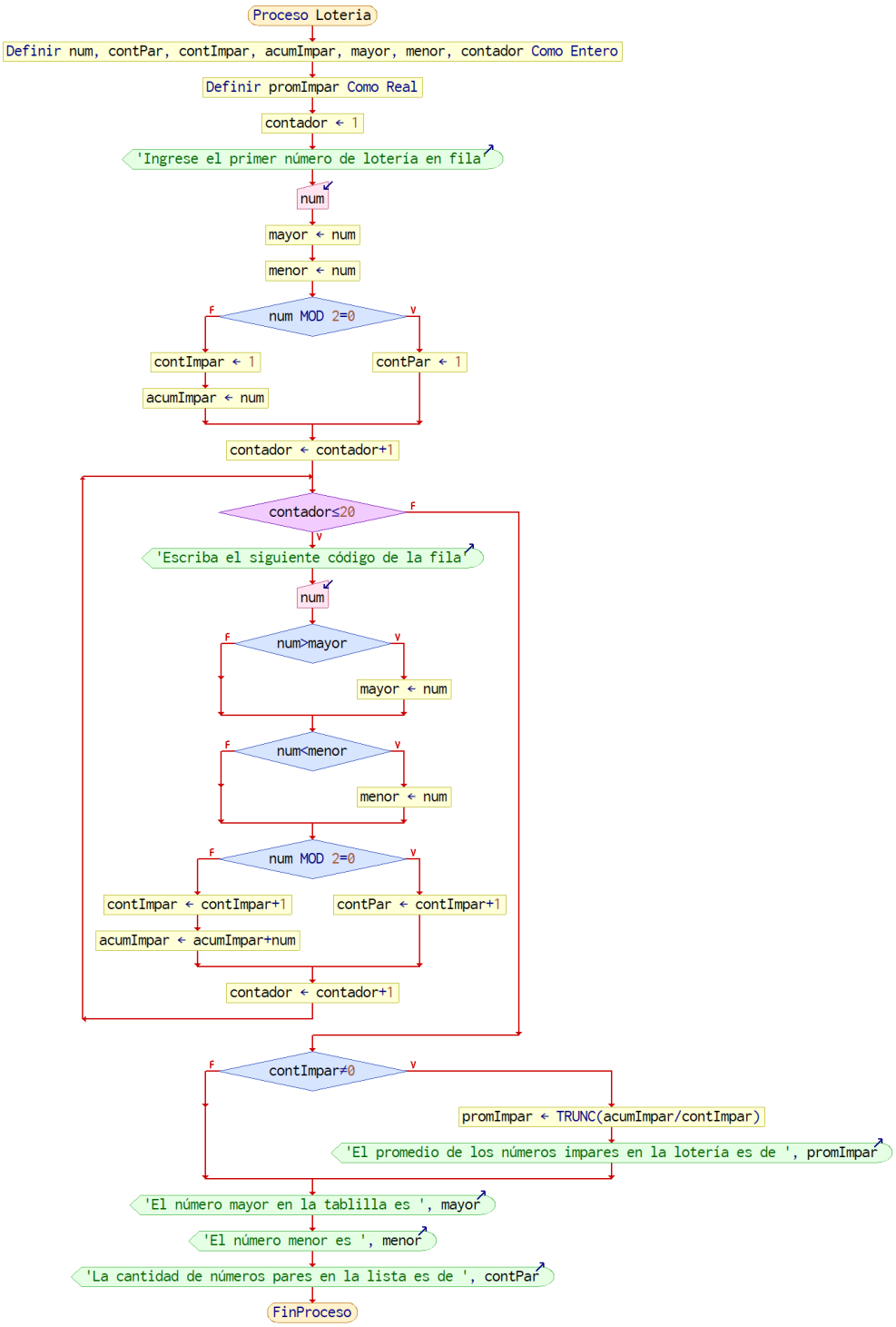
Variable	Tipo de Datos	Descripción
num	Entero	Número sorteado a ingresar.
contPar	Entero	Cantidad de números pares en la tablilla.
contImpar	Entero	Cantidad de números impares en la tablilla.
acumImpar	Entero	Acumulador de la suma de todos números impares.
promImpar	Real	Promedio de todos los números impares.
mayor	Entero	Número mayor ingresado.
menor	Entero	Número menor ingresado.
contador	Entero	Contador de iteración de números (del 1 al 20)

4. Algoritmo:

a. Seudocódigo:

```
Proceso Loteria
        Definir num, contPar, contImpar, acumImpar, mayor, menor, contador Como Entero;
        Definir promImpar Como Real;
        contador ← 1;

        Escribir "Ingrese el primer número de lotería en fila";
        Leer num;

        mayor ← num;
        menor ← num;

        Si num % 2 = 0 Entonces
                contPar ← 1;
        SiNo
                contImpar ← 1;
                acumImpar ← num;
        FinSi
        contador ← contador + 1;

        Mientras contador <= 20 Hacer
                Escribir "Escriba el siguiente código de la fila";
                Leer num;

                Si num > mayor Entonces
                        mayor ← num;
                FinSi

                Si num < menor Entonces
                        menor ← num;
                FinSi

                Si num % 2 = 0 Entonces
                        contPar ← contImpar + 1;
                SiNo
                        contImpar ← contImpar + 1;
                        acumImpar ← acumImpar + num;
                FinSi

                contador ← contador + 1;
        FinMientras

        Si contImpar ≠ 0 Entonces
                promImpar ← TRUNC(acumImpar / contImpar);
                Escribir "El promedio de los números impares en la lotería es de ", promImpar;
        FinSi

        Escribir "El número mayor en la tablilla es ", mayor;
        Escribir "El número menor es ", menor;
        Escribir "La cantidad de números pares en la lista es de ", contPar;
FinProceso
```

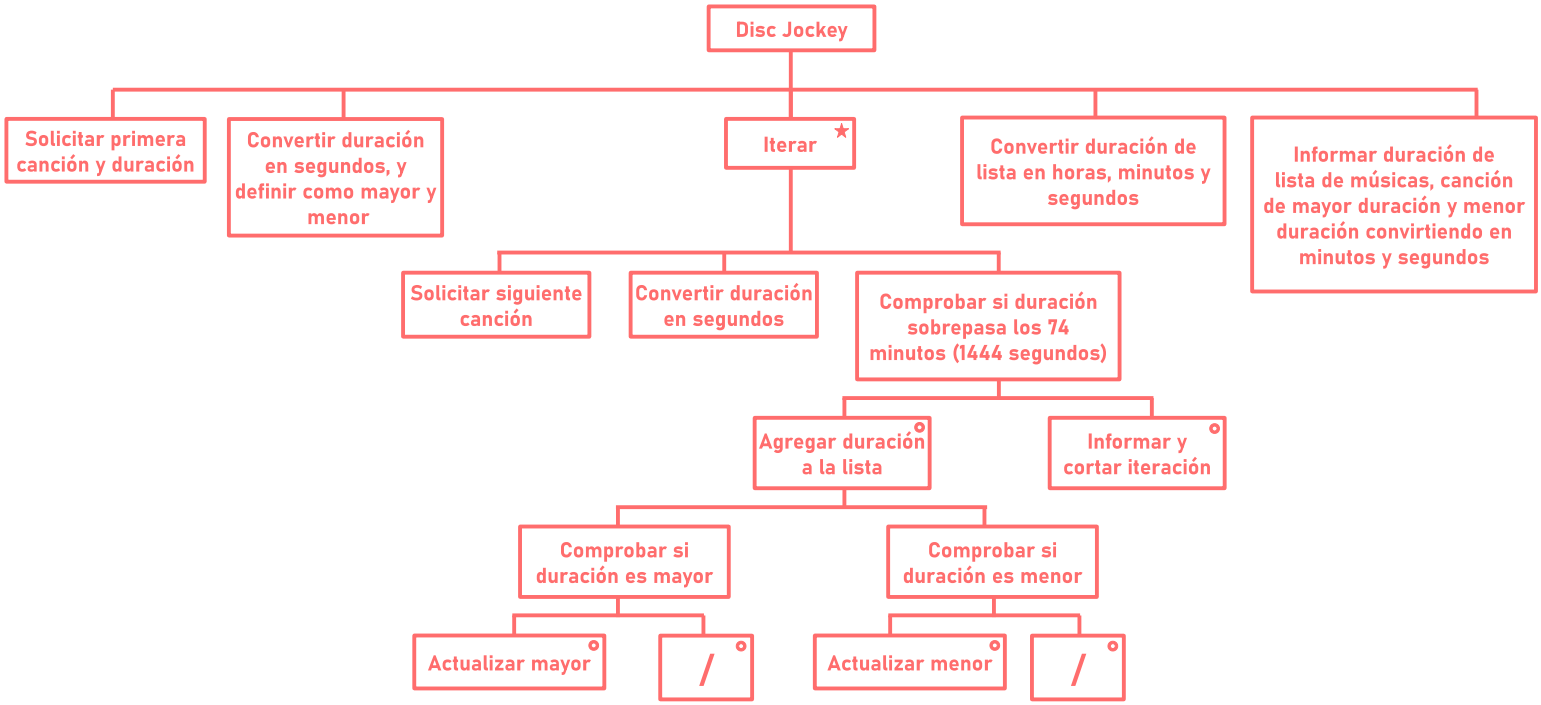
b. Diagrama de flujos:


Proceso Loteria
Definir num, contPar, contImpar, acumImpar, mayor, menor, contador Como Entero
Definir promImpar Como Real
contador ← 1
'Ingrese el primer número de lotería en fila
num
mayor ← num
menor ← num
num MOD 2=0
F
V
contImpar ← 1
contPar ← 1
acumImpar ← num
contador ← contador+1
contador≤20
'Escriba el siguiente código de la fila
num
num>mayor
mayor ← num
num<menor
menor ← num
num MOD 2=0
contImpar ← contImpar+1
contPar ← contImpar+1
acumImpar ← acumImpar+num
contador ← contador+1
contImpar≠0
promImpar ← TRUNC(acumImpar/contImpar)
'El promedio de los números impares en la lotería es de ', promImpar
'El número mayor en la tablilla es ', mayor
'El número menor es ', menor
'La cantidad de números pares en la lista es de ', contPar
FinProceso

Ejercicio 2) Disc Jockey:

1. Análisis:
   a. Entradas:
      - Nombre y duración de canciones ingresadas
   b. Salidas:
      - Total de horas, minutos y segundos de la lista de músicas
      - Música de más duración
      - Música de menos duración
   c. Relación:
      - Iterar hasta que el usuario ingrese -1 o la lista supere los 74 minutos
        - Informar si se continua (1 para sí, -1 para no)
        - Cargar nombre y duración (minutos y segundos) de la música
        - Actualizar música de mayor duración
        - Actualizar música de menor duración
      - Definir tiempo total de lista de canciones

2. Estrategia:



3. Ambiente:

Variable	Tipo de Datos	Descripción
nombre	Texto	Nombre de canción ingresada
musicaMins	Entero	Duración de minutos de canción ingresada
musicaSegs	Entero	Duración de segundos de canción ingresada
duracion	Entero	Conversión del tiempo de música en segundos
totalSegundos	Entero	Total de duración de lista de músicas en segundos
horas	Entero	Duración total de la lista en horas
mins	Entero	Duración total de la lista en minutos
segs	Entero	Duración total de la lista en segundos
nombreMayor	Entero	Nombre de la canción con mayor duración
mayorDuracion	Entero	Tiempo de la canción con mayor duración
nombreMenor	Entero	Nombre de la canción con menor duración
menorDuracion	Entero	Tiempo de la canción con menor duración
continuar	Booleano	Bandera que define si la iteración continúa o corta al llegar a 74 minutos límite

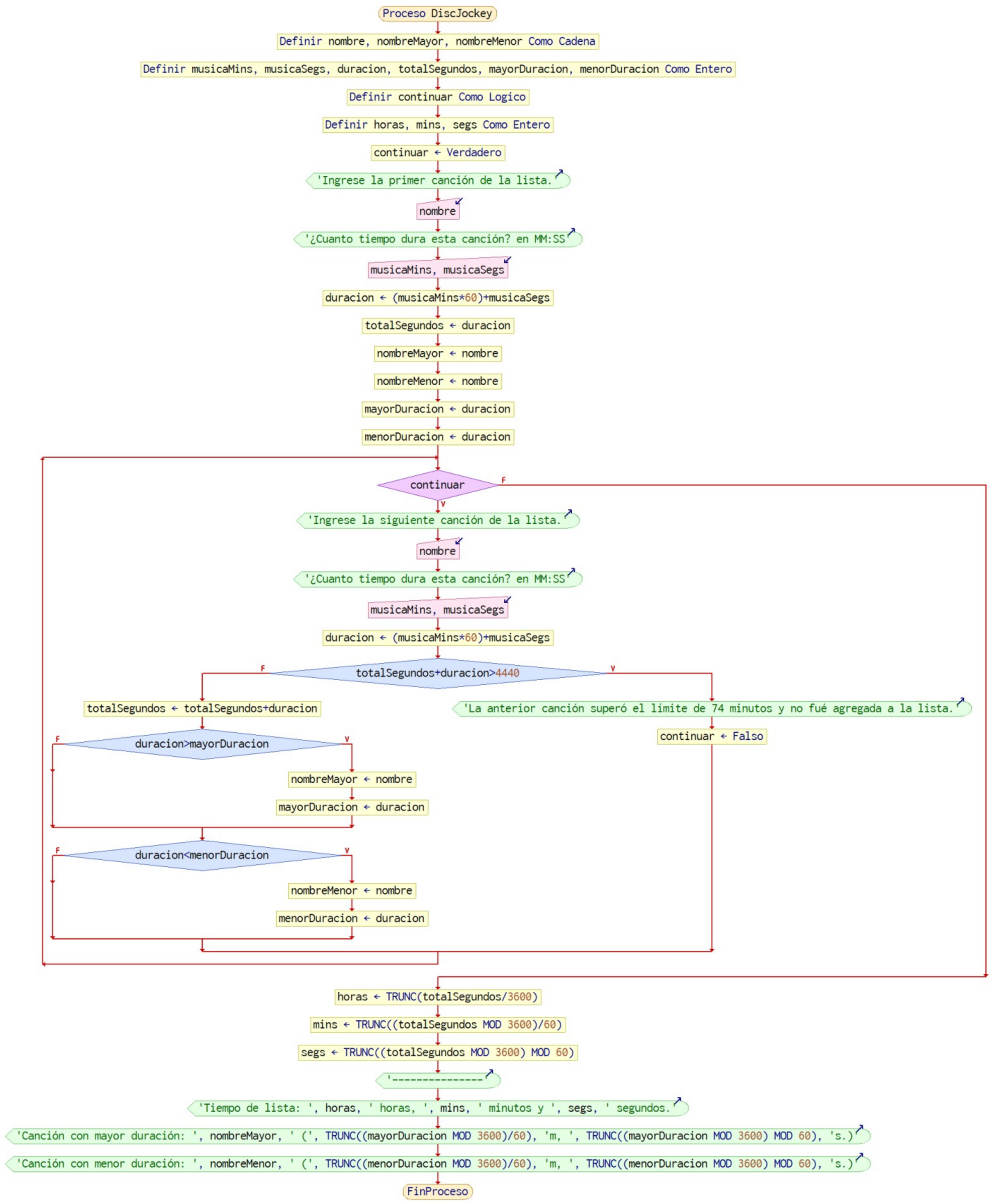
4. Algoritmo:

a. Seudocódigo:

```
Proceso DiscJockey
	Definir nombre, nombreMayor, nombreMenor Como Texto;
	Definir musicaMins, musicaSegs, duracion, totalSegundos, mayorDuracion, menorDuracion Como Entero;
	Definir continuar Como Logico;
	Definir horas, mins, segs Como Entero;

	continuar <- Verdadero;

	Escribir "Ingrese la primer canción de la lista.";
	Leer nombre;
	Escribir "¿Cuanto tiempo dura esta canción? en MM:SS";
	Leer musicaMins, musicaSegs;

	duracion <- (musicaMins * 60) + musicaSegs;

	totalSegundos <- duracion;
	nombreMayor <- nombre;
	nombreMenor <- nombre;
	mayorDuracion <- duracion;
	menorDuracion <- duracion;

	Mientras continuar Hacer
		Escribir "Ingrese la siguiente canción de la lista.";
		Leer nombre;
		Escribir "¿Cuanto tiempo dura esta canción? en MM:SS";
		Leer musicaMins, musicaSegs;

		duracion <- (musicaMins * 60) + musicaSegs;

		Si totalSegundos + duracion > 4440 Entonces
			Escribir "La anterior canción superó el límite de 74 minutos y no fué agregada a la lista.";
			continuar <- Falso;
		SiNo
			totalSegundos <- totalSegundos + duracion;

			Si duracion > mayorDuracion Entonces
				nombreMayor <- nombre;
				mayorDuracion <- duracion;
			FinSi

			Si duracion < menorDuracion Entonces
				nombreMenor <- nombre;
				menorDuracion <- duracion;
			FinSi
		FinSi
	FinMientras

	horas <- TRUNC(totalSegundos / 3600);
	mins <- TRUNC((totalSegundos mod 3600) / 60);
```

```
50              segs <- TRUNC((totalSegundos mod 3600) mod 60);
51
52              Escribir "----------";
53              Escribir "Tiempo de lista: ", horas, " horas, ", mins, " minutos y ", segs, " segundos.";
54              Escribir "Canción con mayor duración: ", nombreMayor, " (", TRUNC((mayorDuracion mod 3600) / 60), "m, ",
    TRUNC((mayorDuracion mod 3600) mod 60), "s.)";
55              Escribir "Canción con menor duración: ", nombreMenor, " (", TRUNC((menorDuracion mod 3600) / 60), "m, ",
    TRUNC((menorDuracion mod 3600) mod 60), "s.)";
56  FinProceso
```

b. Diagrama de flujos:



Ejercicio 3: Tornillos

1. Análisis:
   a. Entradas:

- Número de código del producto (0 para cortar iteración)
- Medida esperada en el lote
- Medidas de los 10 tornillos en el lote

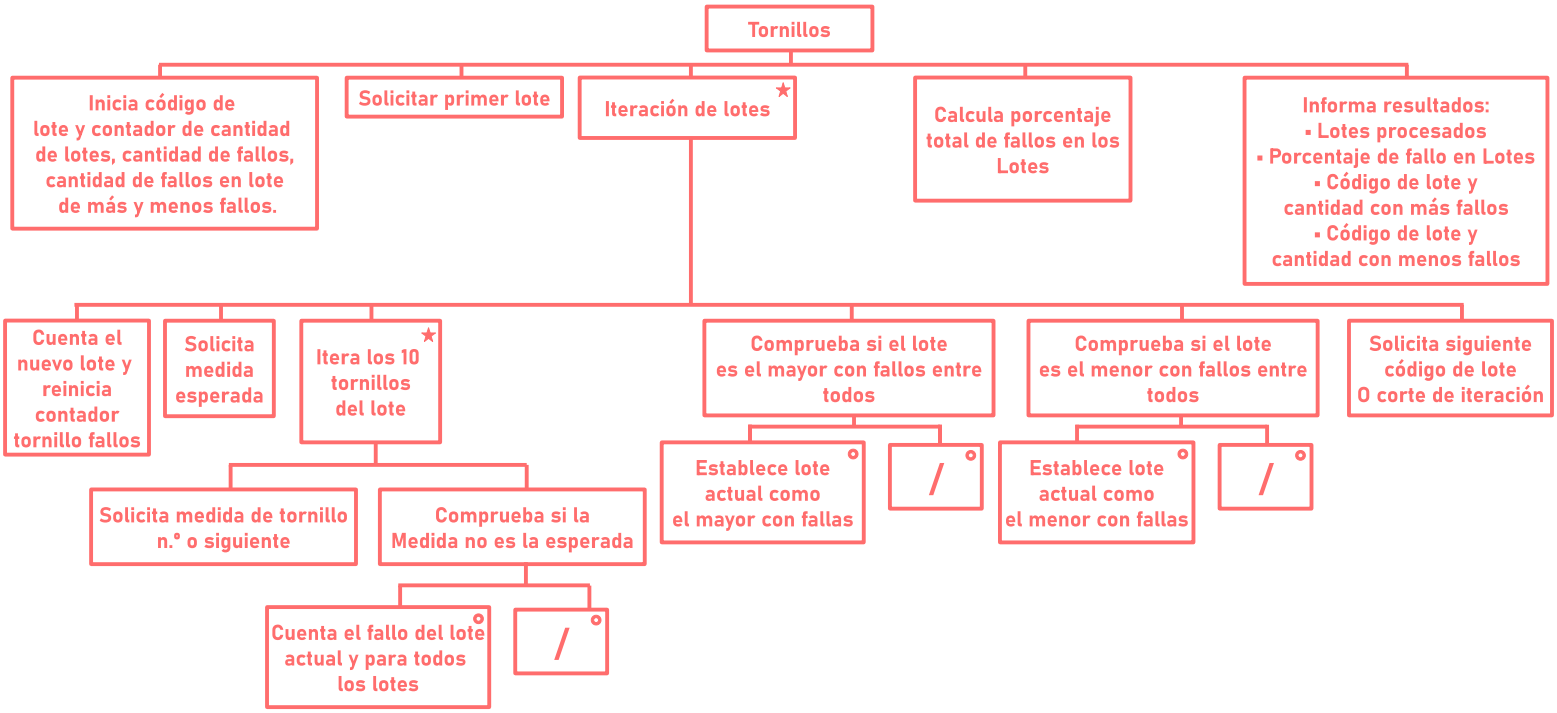
b. Salidas:

- Cantidad de lotes
- Lote con mayor cantidad de fallas
- Lote con menor cantidad de fallas
- Porcentaje total de fallos

c. Relación:

- Iterar hasta que se reciba “0”
  - Verificar si la medición del tornillo es igual a la medida esperada
  - Contar tornillo con medida esperada o con falla
  - Contar cantidad de lotes
  - Contar y verificar menor y mayor cantidad de lotes con fallas
- Calcular porcentaje de productos con fallas

2. Estrategia:



3. Ambiente:

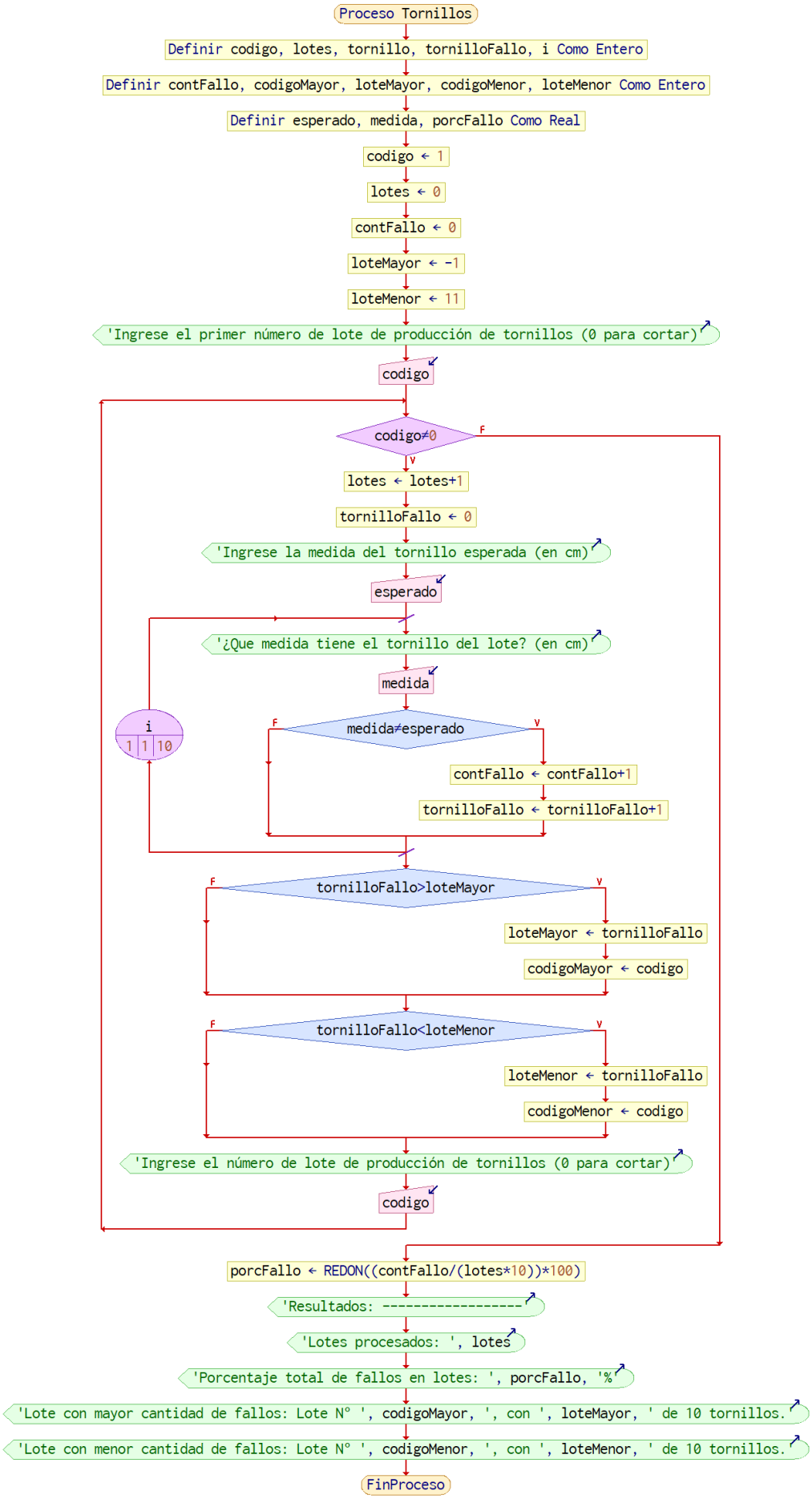
Variable	Tipo de Datos	Descripción
codigo	Entero	Número de código de lote (0 para corte)
lotes	Entero	Cantidad de lotes procesados en la iteración
tornillo	Entero	Contador de tornillo actual
tornilloFallo	Entero	Cuenta los tornillos con la medida falla en el lote
esperado	Real	Medida que deben tener los tornillo esperada en el lote
medida	Real	Medida de tornillo individual comparada con medida esperada
contFallo	Entero	Contador de tornillos totales con fallos en la medida
codigoMayor	Entero	Número de código con más fallos
loteMayor	Entero	Lote con mayores fallos en medición
codigoMenor	Entero	Número de código con menos fallos
loteMenor	Entero	Lote con menores fallos en medición
porcFallo	Real	Porcentaje total de productos con medidas falladas
i	Entero	Itera la cantidad de tornillos en el lote

4. Algoritmo:
   a. Pseudocódigo:

```
1  Proceso Tornillos
2      Definir codigo, lotes, tornillo, tornilloFallo, i Como Entero;
3      Definir contFallo, codigoMayor, loteMayor, codigoMenor, loteMenor Como Entero;
4      Definir esperado, medida, porcFallo Como Real;
5
6      codigo <- 1;
7      lotes <- 0;
8      contFallo <- 0;
9      loteMayor <- -1;
10     loteMenor <- 11;
11
12
13     Escribir "Ingrese el primer número de lote de producción de tornillos (0 para cortar)";
14     Leer codigo;
15
16     Mientras codigo <> 0 Hacer
17         lotes <- lotes + 1;
18         tornilloFallo <- 0;
19
20         Escribir "Ingrese la medida del tornillo esperada (en cm)";
21         Leer esperado;
22
23         Para i <- 1 Hasta 10 Con Paso 1 Hacer
24             Escribir "¿Que medida tiene el tornillo del lote? (en cm)";
25             Leer medida;
26
27             Si medida <> esperado Entonces
28                 contFallo <- contFallo + 1;
29                 tornilloFallo <- tornilloFallo + 1;
30             FinSi
31
32         FinPara
33
34         Si tornilloFallo > loteMayor Entonces
35             loteMayor <- tornilloFallo;
36             codigoMayor <- codigo;
37         FinSi
38
39         Si tornilloFallo < loteMenor Entonces
40             loteMenor <- tornilloFallo;
41             codigoMenor <- codigo;
42         FinSi
43
44         Escribir "Ingrese el número de lote de producción de tornillos (0 para cortar)";
45         Leer codigo;
46     FinMientras
47
48
49     porcFallo <- REDON((contFallo / (lotes * 10)) * 100);
```

```
50
51              Escribir "Resultados: ----------";
52              Escribir "Lotes procesados: ", lotes;
53              Escribir "Porcentaje total de fallos en lotes: ", porcFallo, "%";
54              Escribir "Lote con mayor cantidad de fallos: Lote N° ", codigoMayor, ", con ", loteMayor, " de 10 tornillos.";
55              Escribir "Lote con menor cantidad de fallos: Lote N° ", codigoMenor, ", con ", loteMenor, " de 10 tornillos.";
56
57  FinProceso
```

b. Diagrama de Flujos:


Proceso Tornillos
Definir codigo, lotes, tornillo, tornilloFallo, i Como Entero
Definir contFallo, codigoMayor, loteMayor, codigoMenor, loteMenor Como Entero
Definir esperado, medida, porcFallo Como Real
codigo ← 1
lotes ← 0
contFallo ← 0
loteMayor ← -1
loteMenor ← 11
'Ingrese el primer número de lote de producción de tornillos (0 para cortar)'
codigo
codigo≠0
F
V
lotes ← lotes+1
tornilloFallo ← 0
'Ingrese la medida del tornillo esperada (en cm)'
esperado
'¿Que medida tiene el tornillo del lote? (en cm)'
medida
i
1 1 10
F
medida≠esperado
V
contFallo ← contFallo+1
tornilloFallo ← tornilloFallo+1
F
tornilloFallo>loteMayor
V
loteMayor ← tornilloFallo
codigoMayor ← codigo
F
tornilloFallo<loteMenor
V
loteMenor ← tornilloFallo
codigoMenor ← codigo
'Ingrese el número de lote de producción de tornillos (0 para cortar)'
codigo
porcFallo ← REDON((contFallo/(lotes*10))*100)
'Resultados: ----------'
'Lotes procesados: ', lotes
'Porcentaje total de fallos en lotes: ', porcFallo, '%'
'Lote con mayor cantidad de fallos: Lote N° ', codigoMayor, ', con ', loteMayor, ' de 10 tornillos.'
'Lote con menor cantidad de fallos: Lote N° ', codigoMenor, ', con ', loteMenor, ' de 10 tornillos.'
FinProceso

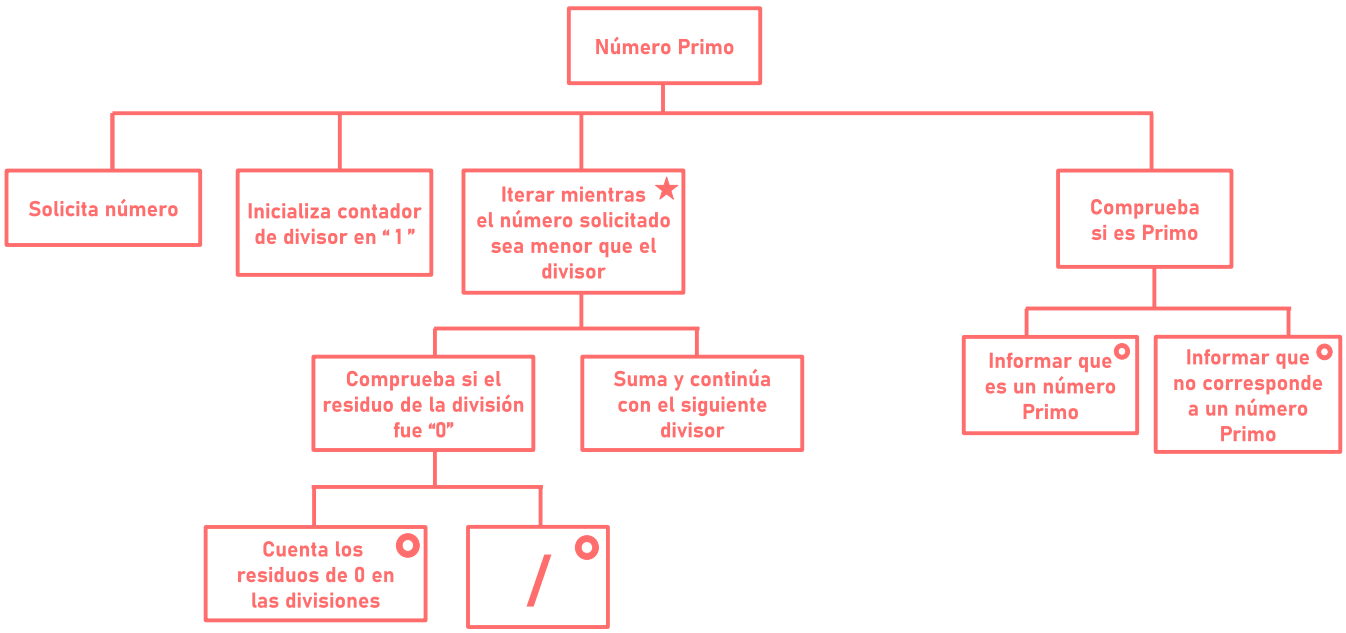
Ejercicio 4: Si es Primo

1. Análisis:
   a. Entradas:
      - Número cualquiera

   b. Salidas:
      - Informar si es un número Primo o no

   c. Relación:
      - Iterar número si es mayor o igual al divisor
        - Comprobar si resto = 0
        - Contar restos de 0
        - Sumar al divisor y continuar con la siguiente comprobación
      - Si el contador de resto terminó en 2 (se dividió por sí mismo y por 1)
        - Mostrar si es primo o no

2. Estrategia:



3. Ambiente:

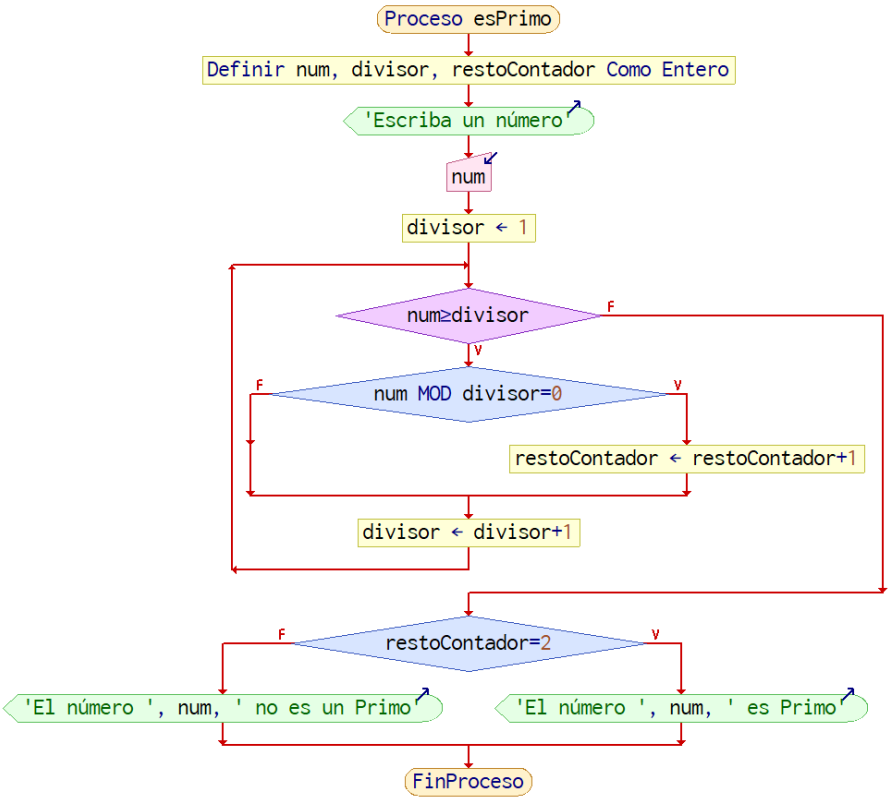
Variable	Tipo de Datos	Descripción
num	Entero	Número ingresado para comprobar.
divisor	Entero	Divide el número ingresado "num" veces
restoContador	Entero	Contador de residuo "0" en las divisiones

4. Algoritmo:
   a. Pseudocódigo:

```
1  Proceso esPrimo
2
3          Definir num, divisor, restoContador Como Entero;
4
5          Escribir "Escriba un número";
6          Leer num;
7
8          divisor <- 1;
```

```
9
10              Mientras num >= divisor Hacer
11                      Si num % divisor = 0 Entonces
12                              restoContador <- restoContador + 1;
13                      FinSi
14                      divisor <- divisor + 1;
15              FinMientras
16
17              Si restoContador = 2 Entonces
18                      Escribir "El número ", num, " es Primo";
19              SiNo
20                      Escribir "El número ", num, " no es un Primo";
21              FinSi
22      FinProceso
```

b. Diagrama de Flujos:



Ejercicio 5: Generar números primos

1. Análisis:
   a. Entradas:
      - Rango de inicio
      - Rango de final
   b. Salidas:
      - Cantidad de números primos presentes entre “Rango de Inicio” y “Fin”
   c. Relación:
      -