EDITORIAL MACRO

Ing. Ricardo Marcelo Villalobos

# Fundamentos de programación

# Visual Basic

## Más de 100 algoritmos codificados

Fundamentos de programación - Estructura secuencial
Estructura selectiva simple y doble - Estructura repetitiva «Mientras»
Cadenas de caracteres

2.ª edición

# Fundamentos de programación

# Visual Basic

Más de 100 algoritmos codificados

EDITORIAL MACRO®

España - México - Colombia - Chile - Ecuador - Perú - Bolivia - Uruguay - Guatemala - Costa Rica

**Fundamentos de programación Visual Basic**
Autor: Ricardo Walter Marcelo Villalobos



Coordinadora de edición:
Cynthia Arestegui Baca

Diseño de portada:
Alejandro Marcas León

Corrección de estilo:
Milton A. Gonzales M.

Diagramación:
Katia Valverde Espinoza

Edición a cargo de:
© Empresa Editora Macro EIRL
Av. Paseo de la República N.° 5613, Miraflores, Lima, Perú
Teléfono: (511) 748 0560
E-mail: proyecto@editorialmacro.com
Página web: www.editorialmacro.com

Primera edición digital: julio 2016

Disponible en: macro.bibliotecasenlinea.com

ISBN N.° 978-612-304-236-3
ISBN digital N.° 978-612-304-459-6

# AUTOR

## Ricardo Marcelo Villalobos

Profesional de sistemas y contabilidad, con más de diez años de experiencia en TI, ha participado como asesor y desarrollador en proyectos de *software* para diversas empresas privadas y públicas del país, como Minera del Hill, Aruntani, Verkaufen, MINSA, IPD; y transnacionales como Magna Rosseta Cerámica- MRC, en la cuales ha utilizando sus conocimientos de contabilidad y de ingeniería de *software* para realizar el análisis y diseños de *software* con RUP, UML y patrones de arquitectura. Asimismo, ha realizado diseños con lenguajes Java, .NET y PHP; también ha trabajado con base de datos Oracle, SQL Server, MySQL y PostgreSQL.

Asimismo, ha participado como expositor en diferentes universidades e institutos (Universidad Nacional de Ingeniería - CEPS-UNI, Universidad Nacional de Trujillo, Universidad César Vallejo de Trujillo, Universidad Nacional José Faustino Sánchez Carrión de Huacho, Instituto San Agustín, Instituto José Pardo, Instituto Manuel Seoane Corrales, Instituto La Reyna Mercedaria). Ha escrito libros, artículos y manuales de desarrollo de *software* (*Visual Basic Nivel III componentes*, *Oracle 10g*, manuales de VB.NET, ADO.NET, POO.NET, Access, Java POO, PHP Fundamentos, PHP POO).

En 2008 fue invitado por la Empresa Editora Macro para formar parte del *staff* de escritores, y salen a la luz cuatro obras relacionadas a los primeros pasos de la ingeniería de *software* (libros de fundamentos y mas de 100 Algoritmos con Visual Basic, Java, C++ y C#).

En la actualidad, difunde su experiencia como docente en la Universidad Nacional de Ingeniería (UNI-FIIS- CEPS-UNI) y el Instituto San Ignacio (ISIL); asimismo, realiza capacitaciones para empresas, como Telefónica del Perú, FAP, La Caja de Pensiones Militar Policial, ALPECO, Banco de Materiales, entre otros.

# Agradecimientos

Es difícil dejar de mencionar a las personas que día a día fortalecen el conocimiento y la sabiduría de los demás. Me faltarían líneas en este libro para mencionar a todos, pero quiero agradecer, en primer lugar, a Dios y a mis padres.

También a personas muy especiales que, con su sabiduría y experiencia, han ayudado y permitido plasmar muchas de sus ideas en esta obra: Peggy Sánchez, Sergio Matsukawa, Gustavo Coronel, Gino Henostroza, Julio Flores, Joel Carrasco, Luis Zúñiga, Jesús Echevarria y todos mis alumnos y amigos en general.

# Prólogo

Cómo no recordar las primeras clases de Algoritmo y la ilusión que todos tienen por aprender a programar. Esta obra plasma los primeros pasos que cualquier estudiante de la carrera de Ingeniería de Sistemas, *software* e informática debe conocer para empezar a analizar, diseñar y codificar sus primeros algoritmos; y así pasar la barrera que todo programador debe dominar, que son las estructuras de control de flujo tales como if, switch (C++, Java y C#), select case (vb), while y for.

Este libro contiene nueve capítulos con más de cien algoritmos resueltos y otros ochenta propuestos; estoy seguro de que al concluir la lectura, el usuario formará parte del mundo de los desarrolladores de *software*. En el primer capítulo se desarrollan los conceptos generales de arquitectura de la PC, *hardware*, *software*, lenguajes de programación, metodología de algoritmos, diagramas de flujo, pseudocódigo, variables, constantes, instrucciones, entre otros.

El segundo apartado contiene diez algoritmos básicos para entender y resolver en forma simple los problemas de entrada, proceso (secuencial) y salida de los cálculos realizados. El tercer capítulo presenta quince algoritmos con las estructuras más utilizadas en la solución de problemas, llamada if. En el cuarto capítulo se explica la forma más fácil de solucionar problemas sin el uso de if anidados y engorrosos. En el quinto capítulo se enseña a entender y dominar la estructura repetitiva, y a aplicar los conceptos de contador, acumulador, bucles, entre otros.

Debido a que muchas veces es más fácil resolver procesos repetitivos usando la estructura for, en el sexto apartado se encuentran quince problemas resueltos; aunque muchos de ellos pertenecen al capítulo anterior, esto servirá para analizar su simplicidad. En el sétimo apartado –tomando en cuenta que uno de los temas más utilizados en el manejo de colecciones de datos tiene que ver con los arreglos (*arrays*)– se explica el concepto y se resuelven problemas de arreglos, algoritmos de búsqueda y ordenación de datos. En el capítulo octavo, se explican y resuelven problemas con cadena de caracteres (texto). Finalmente, una de las mejores recomendaciones para resolver y reutilizar procesos es el concepto de divide y vencerás, por ello en el capítulo nueve se enseña cómo separar un problema en varias partes reutilizables.

# ÍNDICE

## Índice

## Capítulo 2
## Estructura secuencial ..... 33



## Capítulo 3
## Estructura selectiva simple y doble ..... 49

## Capítulo 4
## Estructura selectiva múltiple ..... 85



## Capítulo 5
## Estructura repetitiva «Mientras» ..... 125

## Capítulo 9
## SubAlgoritmos (procedimientos y funciones) ..... 249

# Capítulo 1

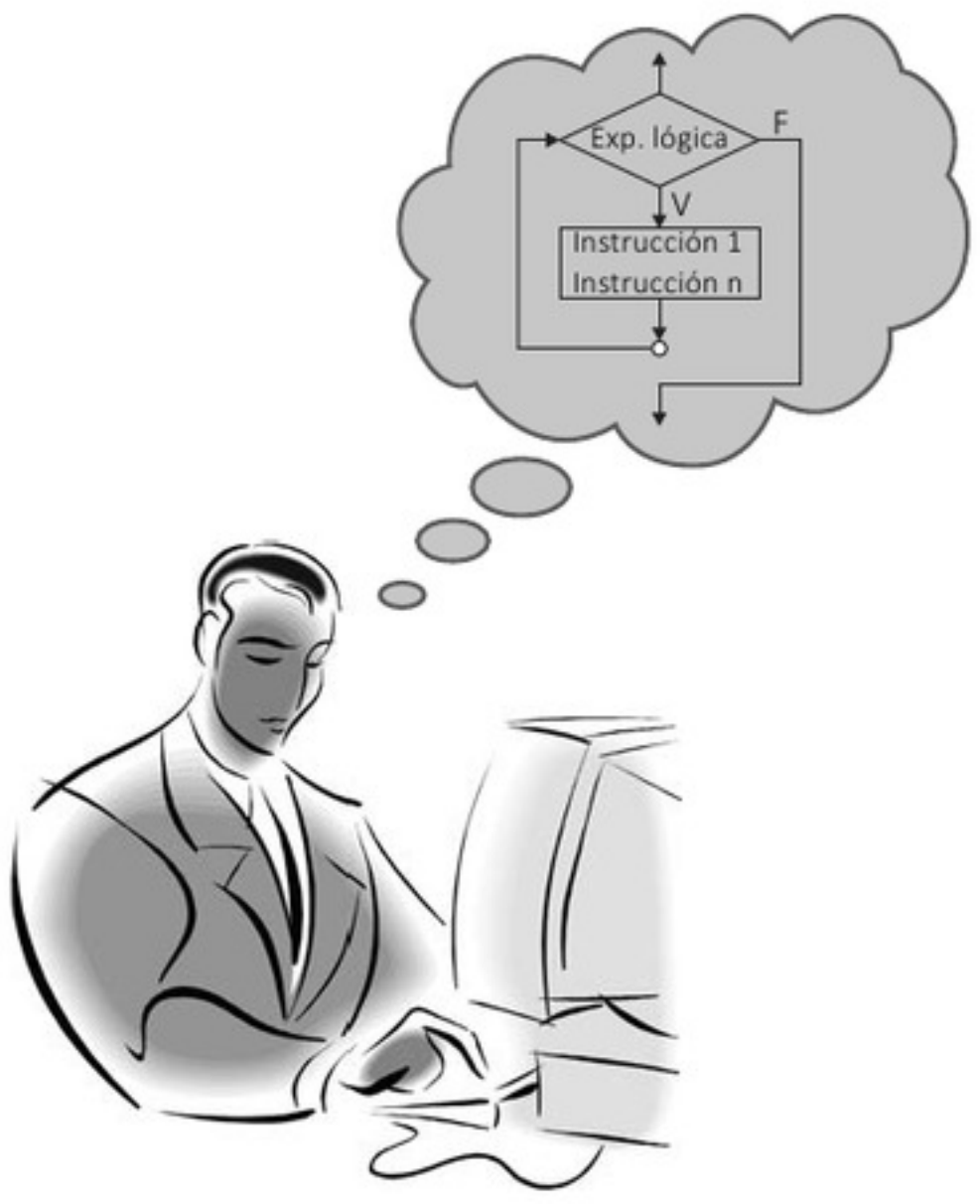
# Fundamentos de programación

## 1.1 Introducción

En los primeros ciclos de toda carrera profesional relacionada a la ingeniería de sistemas, los estudiantes requieren entender, aprender y dominar los fundamentos de programación para resolver problemas que permitirán automatizar procesos usando la computadora.

Saber programar es la base de toda su carrera y, para conseguir este objetivo, he plasmado mi experiencia de docencia de mas de diez años en el campo de la Ingeniería de Sistemas. Sé que este libro le ayudará a resolver todas sus dudas y dominar las principales estructuras de programación.

Este libro contiene más de 100 algoritmos resueltos y codificados en el lenguaje Visual Basic, que en la actualidad es uno de los lenguajes de programación propuesto por Microsoft para la tecnología .NET




A continuación, se describen los conceptos generales de los fundamentos de programación.

## 1.2 Computadora

Es un aparato electrónico que recibe datos (entrada), los procesa (instrucciones denominado programa) y devuelve información (salida), también conocido como ordenador o PC (Personal Computer). En la actualidad existe una variedad de computadoras para diferentes propósitos.

**Servidores**

**Computadora personal**

**Computadora portátil**

**PDA**

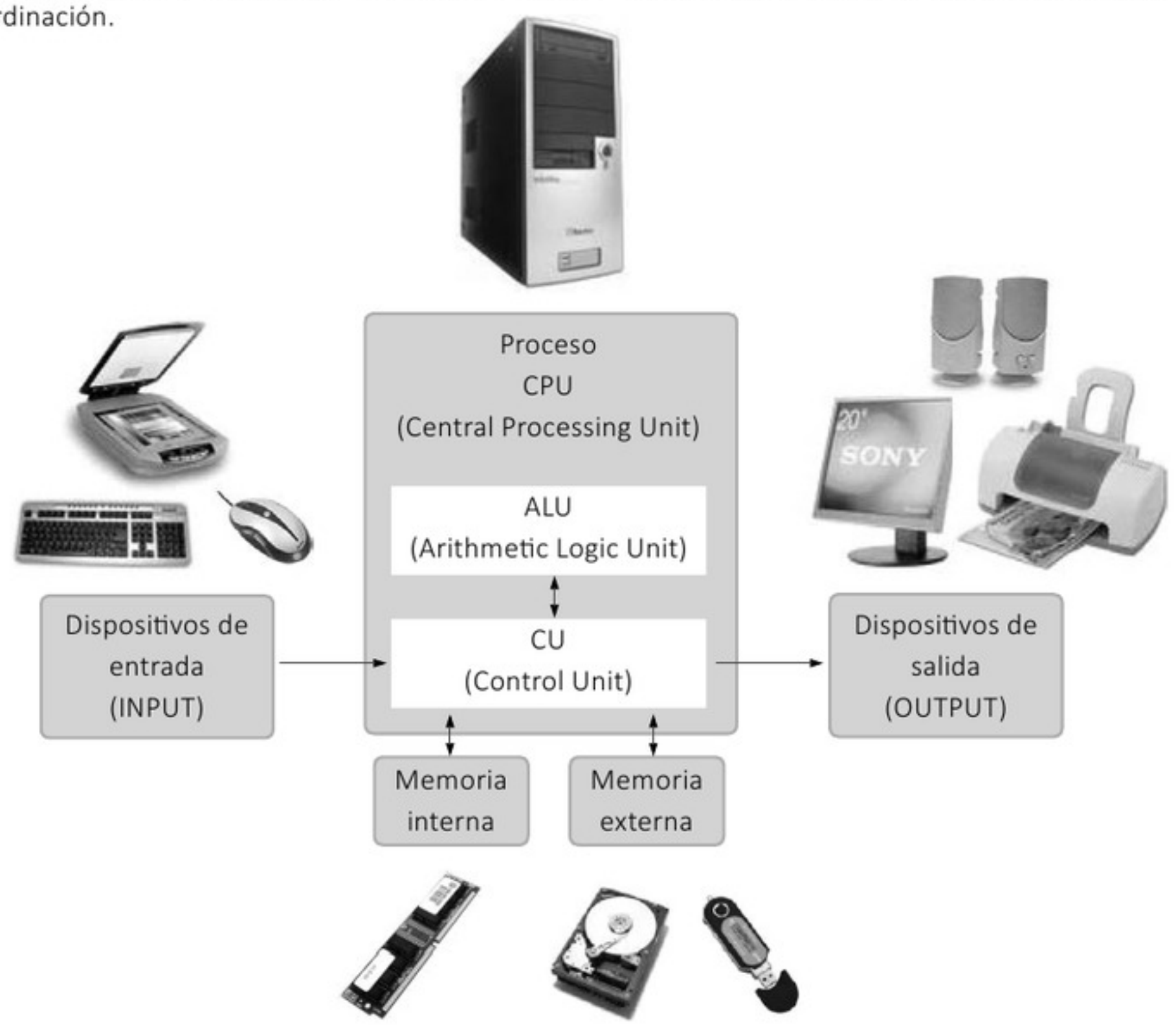
## 1.3 Arquitectura de una computadora

Las computadoras tienen dos componentes principales que son el ***hardware*** y el ***software***, que trabajan en coordinación para llevar a cabo sus objetivos.

**Hardware:** *Hard* (duro) – *ware* (componente), representa la parte física de la computadora.


**Software:** *Soft* (blando) – *ware* (componente), representa la parte lógica de la computadora (los programas); estos se encuentran almacenados en los componentes físicos de la computadora, tales como memorias RAM, ROM, Discos Duros (***Hard Disk***), entre otros.

La siguiente figura muestra la arquitectura de la computadora y sus principales componentes en coordinación.




## 1.4 Unidades de medida de almacenamiento

La memoria interna (RAM) y las memorias externas (disco duro) almacenan información. La información que se guarda y entiende la PC está en formato binario (0- 1).

**BIT (BInary DigiT):** El bit representa la unidad mínima de información que almacena una computadora.

**BYTE:** Está compuesto por 8 bit (01110011), entonces existe $2^8 = 256$ combinaciones diferentes (tabla de código ASCII).

Por lo general, la información se representa por caracteres y cada carácter (número, letra, símbolo, etc.) es un byte. Para medir la información se utilizan múltiplos de bytes.

| | | | |
|---|---|---|---|
| Byte | 1 B | | 8 bits |
| Kilobyte | 1 KB | $2^{10}$ bytes | 1024 bytes |
| Megabyte | 1 MB | $2^{20}$ bytes | 1024 KB |
| Gigabyte | 1 GB | $2^{30}$ bytes | 1024 MB |
| Terabyte | 1 TB | $2^{40}$ bytes | 1024 GB |

## 1.5 Sistemas de numeración

Todos los sistemas de numeración tienen una **base**, que es el número total de símbolos que utiliza el sistema. En el caso de la numeración decimal, la base es 10; en el sistema binario es 2.

El **Teorema Fundamental de la Numeración** permite saber el valor decimal que tiene cualquier número en cualquier base. Dicho teorema utiliza la siguiente fórmula:

$$\ldots + X_3 \cdot B^3 + X_2 \cdot B^2 + X_1 \cdot B^1 + X_0 \cdot B^0 + X_{-1} \cdot B^{-1} + X_{-2} \cdot B^{-2} + \ldots$$

**Donde**:

- **$X_i$:** Es el símbolo que se encuentra en la posición número **i** del número que se está convirtiendo. Teniendo en cuenta que la posición de las unidades es la posición 0 (la posición -1 sería la del primer decimal).
- **B:** Es la base del sistemas que se utiliza para representar al número.

Por ejemplo, si tenemos el número 153,6 utilizando el sistema octal (base ocho), el paso a decimal se sería:

$$1 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 8^0 + 6 \cdot 8^{-1} = 64 + 40 + 3 + 6 \div 8 = 107,75$$

## 1.6 Conversión binario a decimal

El **Teorema Fundamental de la Numeración** se puede aplicar para saber el número decimal representado por un número escrito en binario. Así, para el número binario 10011011011 la conversión sería (los ceros se han ignorado):

$$1 \cdot 2^{10} + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1243$$

## 1.7 Conversión decimal a binario

El método más utilizado consiste en ir haciendo divisiones sucesivas entre dos. Los restos son las cifras binarias. Por ejemplo, para pasar el 39:

```
39 ÷ 2 = 19 resto 1
19 ÷ 2 =  9 resto 1
 9 ÷ 2 =  4 resto 1
 4 ÷ 2 =  2 resto 0
 2 ÷ 2 =  1 resto 0
 1 ÷ 2 =  0 resto 1
```

Ahora las cifras binarias se toman al revés. Con lo cual, el número 100111 es el equivalente en binario de 39.

## 1.8 Representación de texto en el sistema binario

Puesto que una computadora no solo maneja números, habrán dígitos binarios que contengan información no traducible al sistema decimal. Todo depende de cómo se interprete esa traducción. Por ejemplo, en el caso del texto, lo que se hace es codificar cada carácter en una serie de números binarios. El código **ASCII** ha sido durante mucho tiempo el más utilizado; inicialmente era un código que utilizaba 7 bits para representar texto, lo que significaba que era capaz de codificar 127 caracteres. Por ejemplo, el número 65 (1000001 en binario) se utiliza para la **A** mayúscula.

Poco después apareció un problema: este código bastaba para los caracteres del inglés, pero no para otras lenguas. Entonces se añadió el octavo bit para representar otros 128 caracteres que son distintos, según idiomas (Europa Occidental usa unos códigos que no utiliza Europa Oriental).

Eso provoca que un código como el 190 signifique cosas diferentes si cambiamos de país. Por ello, cuando un ordenador necesita mostrar texto, tiene que saber qué juego de códigos debe de utilizar, lo cual supone un tremendo problema.

Una ampliación de este método de codificación es el código **UNICODE**, que puede utilizar hasta 4 bytes (32 bits), con lo que es capaz de codificar cualquier carácter en cualquier lengua del planeta, utilizando el mismo conjunto de códigos. Poco a poco se ha ido extendiendo cada vez más, pero la preponderancia histórica que ha tenido el código ASCII complica su popularidad.

## 1.9 Representación binaria de datos no numéricos ni de texto

En el caso de datos más complejos (imágenes, vídeo, audio) se necesita una codificación más compleja. Además, en estos datos no hay estándares, por lo que hay decenas de formas de codificar. En el caso, por ejemplo, de las imágenes, una forma básica de codificarlas en binario es la que graba cada **píxel** (cada punto distinguible en la imagen) mediante **tres bytes**: el primero graba el nivel de **rojo**; el segundo, el nivel de **azul**; y el tercero, el nivel de **verde**. Y así por cada píxel.

Por ejemplo, un punto en una imagen de color rojo puro:

**11111111 00000000 00000000**

Naturalmente, en una imagen no solo se graban los píxeles sino el tamaño de la imagen, el modelo de color, etc. De ahí que representar estos datos sea tan complejo para el ordenador (y tan complejo entenderlo para nosotros).

## 1.10 Los programas (*software*)

Los programas o ***software*** son un conjunto de instrucciones ordenadas para ejecutarse de forma rápida y precisa en una computadora. El ***software*** se divide en dos grupos: ***software*** de **sistema operativo** y ***software* de aplicaciones**.

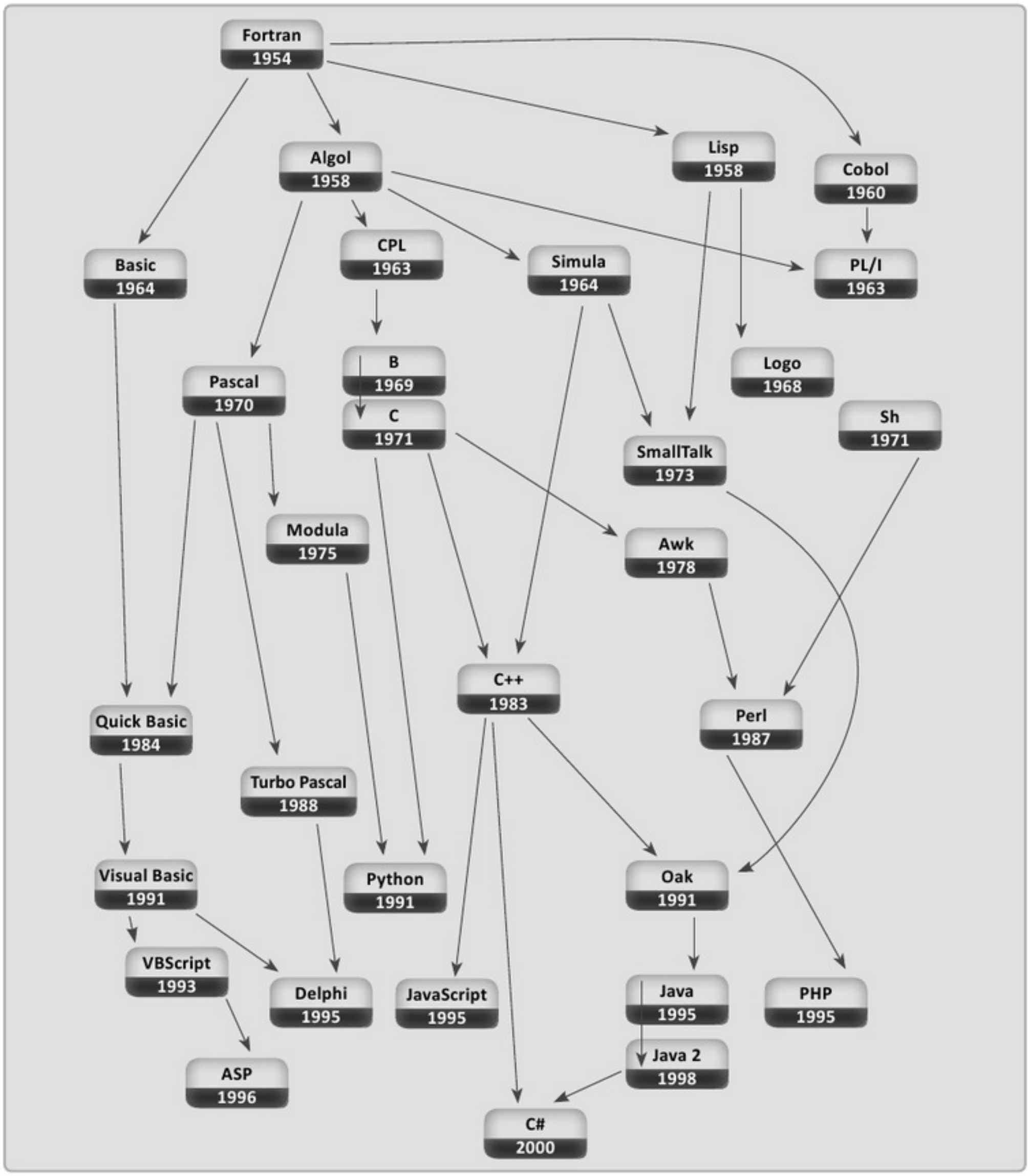
El proceso de escribir un programa se denomina **programación**, y el conjunto de instrucciones que se utilizan para escribir un programa se llama **lenguaje de programación**.

## 1.11 Lenguajes de programación

Sirve para escribir programas y permite la comunicación usuario (programador) versus máquina (PC).

Existen tres tipos de lenguajes de programación:

- **Lenguaje de máquina:** Programación binaria, difícil de programar y dependiente de la máquina.
- **Lenguaje de bajo nivel (ensamblador):** Usa símbolos nemotécnicos, necesita ser traducido al lenguaje de máquina y sigue siendo dependiente.
- **Lenguaje de alto nivel:** Cercano al lenguaje natural, tiempo de programación relativamente corto, es independiente de la máquina. A continuación se muestra un plano de la evolución de los lenguajes de programación de alto nivel.



## 1.12 Traductores del lenguaje de programación

Son programas que traducen los **códigos fuentes** (programas escritos en un lenguaje de alto nivel) a **código máquina**.

Los traductores se dividen en:

- **Intérpretes:** Traducción y ejecución secuencial (línea por línea), ejecución lenta.
- **Compiladores:** Traduce el código fuente a programa objeto (ejecutable código máquina). Ejecución rápida.

## 1.13 Ciclo de vida de un *software*

La construcción de un ***software,*** por más pequeño que sea, involucra las siguientes etapas:

- **Requerimiento:** Enunciado del problema a resolver.
- **Análisis:** ¿Qué? (Entender el problema – entrada – proceso – salida).
- **Diseño:** ¿Cómo? (Resolver el problema – algoritmo – diagrama de flujo – diseño de interfaz de usuario).
- **Implementación:** ¿Hacerlo? (Codificación / Programar).
- **Pruebas:** ¿Funciona? (Verificar / Comprobar).
- **Despliegue:** ¿Instalar? (Distribuir el programa).

## 1.14 Algoritmo

Método que describe la solución de un problema computacional mediante una serie de pasos precisos, definidos y finitos.

- **Preciso:** Indicar el orden de realización en cada paso.
- **Definido:** Al epetir los pasos n veces se obtiene el mismo resultado.
- **Finito:** Tiene un número determinado de pasos.

La solución de un algoritmo debe describir tres partes:

- **Entrada:** Datos que se necesitan para poder ejecutarse.
- **Proceso:** Acciones y cálculos a realizar.
- **Salida:** Resultado esperado.


La palabra algoritmo procede del matemático árabe **Mohamed Ibn Al Kow Rizmi**, quien escribió entre los años 800 y 825 su obra *Quitad Al Mugabala*, donde recogió el sistema de numeración hindú y el concepto del cero. **Fibonacci**, tradujo la obra al latín y la llamó *Algoritmi Dicit*.

El lenguaje algorítmico es aquel que implementa una solución teórica a un problema, indicando las operaciones a realizar y el orden en el que deben efectuarse. Por ejemplo, en el caso de que nos encontremos en casa con un foco malogrado de una lámpara, un posible algoritmo sería:

a. Comprobar si hay foco de repuesto.
b. En el caso de que haya, sustituir el foco anterior por el nuevo.
c. Si no hay foco de repuesto, bajar a comprar uno nuevo en la tienda y ponerlo en lugar del malogrado.

Los algoritmos son la base de la programación de ordenadores, ya que los **programas de ordenador** se pueden entender como algoritmos escritos en un código especial, entendible por un ordenador.

La desventaja del diseño de algoritmos radica en que no podemos escribir lo que deseemos; el lenguaje ha utilizar no debe dejar posibilidad de duda, debe recoger todas las posibilidades.

### 1.14.1 Características que deben de cumplir los algoritmos obligatoriamente

- **Un algoritmo debe resolver el problema para el que fue formulado**. Lógicamente, no sirve un algoritmo que no resuelve ese problema. En el caso de los programadores, a veces crean algoritmos que resuelven problemas diferentes al planteado.
- **Los algoritmos son independientes del lenguaje de programación**. Los algoritmos se escriben para poder ser utilizados en cualquier lenguaje de programación.
- **Los algoritmos deben ser precisos**. Los resultados de los cálculos deben ser exactos, de manera rigurosa. No es válido un algoritmo que sólo aproxime la solución.
- **Los algoritmos deben ser finitos**. Deben de finalizar en algún momento. No es un algoritmo válido aquel que produce situaciones en las que el algoritmo no termina.
- **Los algoritmos deben poder repetirse**. Deben permitir su ejecución las veces que haga falta. No son válidos los que tras ejecutarse una vez ya no pueden volver a hacerlo por la razón que sea.

### 1.14.2 Características aconsejables para los algoritmos

- **Validez:** Un algoritmo es válido si carece de errores. Un algoritmo puede resolver el problema para el que se planteó y, sin embargo, no ser válido debido a que posee errores.
- **Eficiencia:** Un algoritmo es eficiente si obtiene la solución al problema en poco tiempo. No lo es si tarda en obtener el resultado.
- **Óptimo:** Un algoritmo es óptimo si es el más eficiente posible y no contiene errores. La búsqueda de este algoritmo es el objetivo prioritario del programador. No siempre podemos garantizar que el algoritmo hallado sea el óptimo, a veces sí.

### 1.14.3 Fases en la creación de algoritmos

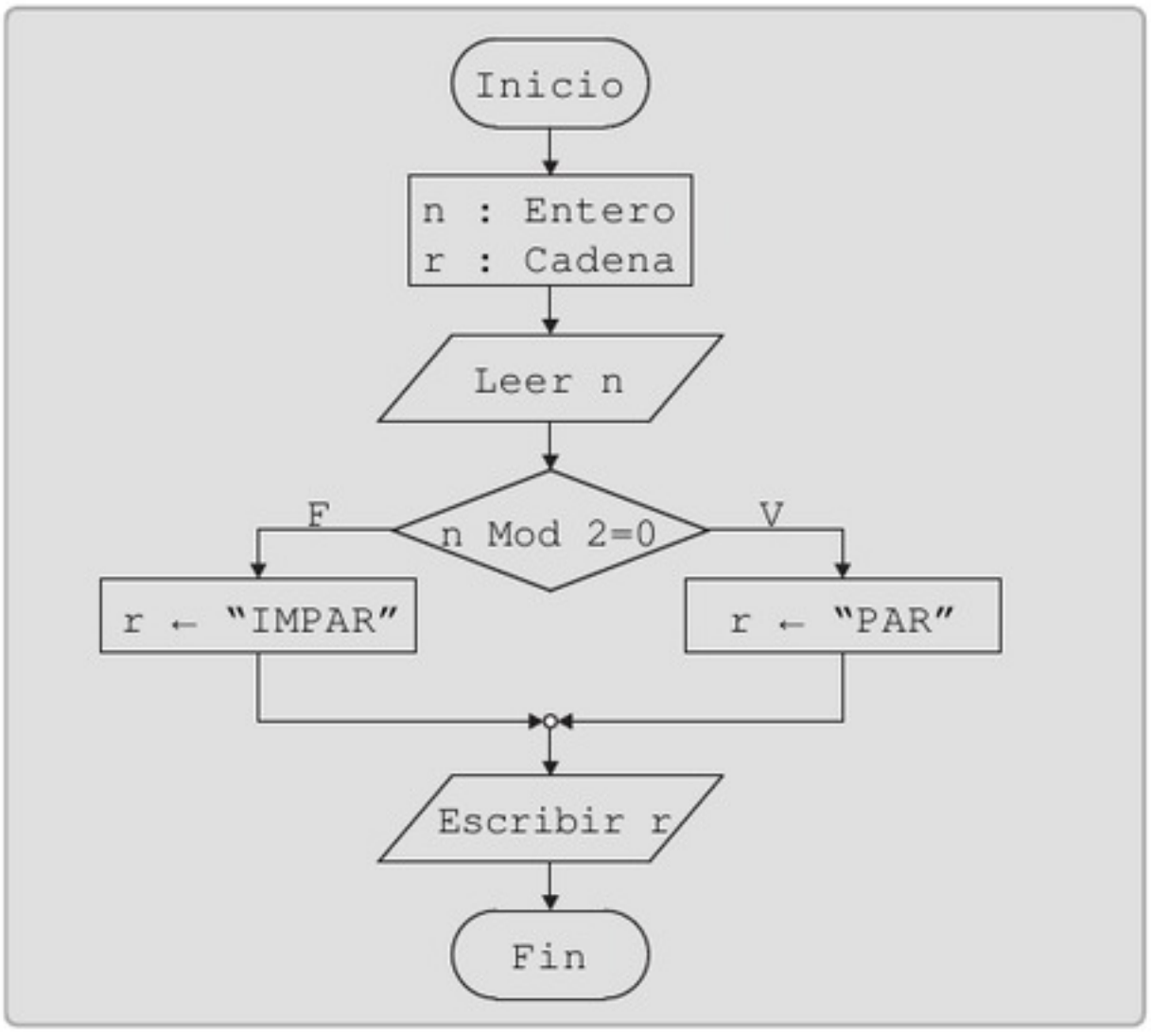
Hay tres fases en la elaboración de un algoritmo:

a. **Análisis:** En esta se determina cuál es exactamente el problema a resolver. Qué datos forman la entrada del algoritmo y cuáles deberán obtenerse como salida.
b. **Diseño:** Elaboración del algoritmo.
c. **Prueba:** Comprobación del resultado. Se observa si el algoritmo obtiene la salida esperada para todas las entradas.

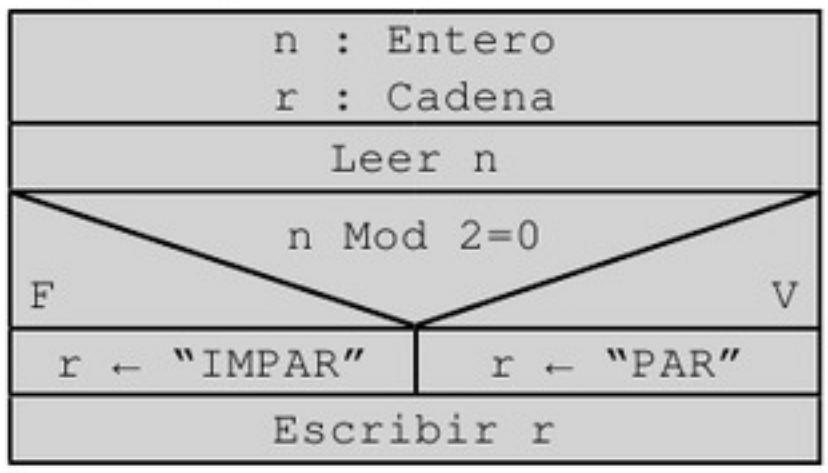
### 1.14.4 Herramientas de un algoritmo

Para expresar la solución de un problema se pueden usar diferentes herramientas de programación, tales como diagrama de flujo (*flow chart*), diagrama N-S (Nassi Schneiderman), pseudocódigo.

• **Diagrama de flujo:** Es una representación gráfica que utiliza símbolos normalizados por ANSI, y expresa las sucesivas instrucciones que se deben seguir para resolver el problema. Estas instrucciones no dependen de la sintaxis de ningún lenguaje de programación, sino que deben servir fácilmente para su transformación (codificación) en un lenguaje de programación.




• **Diagrama de Nassi Scheneiderman (N-S):** Conocido también como el diagrama de Chapin, es como un diagrama de flujo pero sin flechas y con cajas continuas.




• **Pseudocódigo:** Permite expresar las instrucciones en un lenguaje común (ingles, español, etc.) para facilitar tanto la escritura como la lectura de la solución de un programa. No existen reglas para escribir pseudocódigo.

```
Inicio

   //Variables
   n : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   Si n Mod 2 = 0 Entonces
       r ← "PAR"
   SiNo
       r ← "IMPAR"
   Fin Si

   //Salida
   Escribir r

Fin
```

### 1.14.5 Instrucciones

Son las acciones que debe realizar un algoritmo para resolver un problema. Las instrucciones más comunes son las siguientes:

**A. Instrucción de inicio/ fin:** Representa el inicio y fin de un algoritmo.

| Diagrama de flujo | Pseudocódigo |
|---|---|
| Inicio | Inicio |
| Fin | Fin |

**B. Instrucción de asignación:** Representa la asignación de un valor a una variable, se puede representar usando una flecha o el símbolo de igualdad, el cual es usado por muchos de los lenguajes de programación.

| Diagrama de flujo | Pseudocódigo |
|---|---|
| N ← 10 | N ← 10 |
| N = 10 | N = 10 |

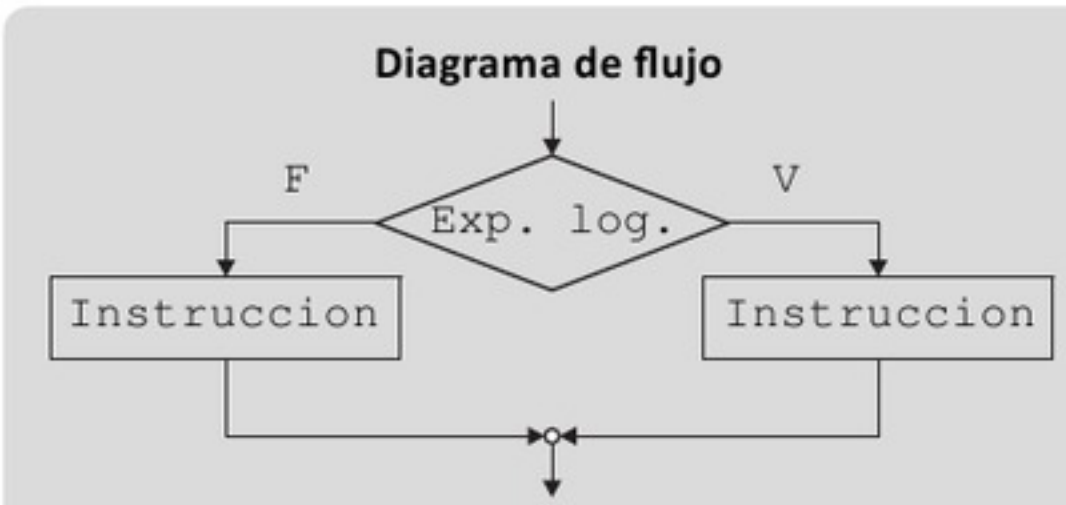
**C. Instrucción de lectura:** Representa el ingreso de datos mediante un dispositivo de entrada, que muchas veces es representado por un símbolo de teclado.

| Diagrama de flujo | Pseudocódigo |
|---|---|
| Leer N | Leer N |
| Leer N | Leer N |

**D. Instrucción de escritura:** Representa la salida de la información mediante un dispositivo de salida, puede ser representado por el símbolo de entrada/salida, por símbolo de pantalla o impresora.

| Diagrama de flujo | Pseudocódigo |
|---|---|
| Escribir R | Escribir R |
| Escribir R | Escribir R |
| Escribir R | Escribir R |

**E. Instrucción de bifurcación:** Cambian el flujo del programa según el resultado de una expresión lógica (condición).




**Pseudocódigo**

**Si** <Exp. Log> **Entonces**

<Instrucción>

**SiNo**

<Instrucción>

**Fin Si**

## 1.15 Comentarios

Permiten describir y explicar, además sirve como ayuda para recordar y entender las operaciones que se van a ejecutar. Los comentarios no son instrucciones, por lo tanto al ser traducido el código fuente a código binario (tiempo de compilación), los lenguajes de programación los ignoran. Dependiendo el lenguaje de programación los comentarios se escriben usando cierta simbología, en este libro usaremos el símbolo **//** en los pseudocódigos para colocar comentarios.

**Ejemplo pseudocódigo**

```
//Variables
N : Entero
```

**Visual Basic**

```
'Variables
Dim N As Integer
```

## 1.16 Palabras reservadas

Son palabras usadas por el lenguaje de programación que no deben ser utilizadas como identificadores de variables, funciones, entre otros.

**Algunas de las palabras reservadas de Visual Basic**

`Dim, Integer, Long, Single, Double, If, For, Select, Me ...`

## 1.17 Identificadores

Son los nombres que asignamos a las variables, constantes, funciones, objetos, entre otros; y no pueden coincidir con las palabras reservadas porque ocasionaría ambiguedad, y el compilador no lo entendería. Por lo general, los identificadores deben de cumplir las siguientes reglas:

- Deben comenzar por una letra. Evite usar ñ y tilde.
- No debe coincidir con palabras reservadas del lenguaje de programación que está utilizando.

**Error de Compilación Visual Basic**

```
// Identificador de Variable es Dim
// y esta es palabra reservada
Dim Dim As Integer
```

## 1.18 Variables

Representa un espacio de memoria RAM, el cual guarda un valor que servirá para algún proceso en particular; dicho valor puede ser modificado en cualquier momento. Las variables tienen, por lo general, un identificador (nombre) y, asignado, el tipo de dato que se esta utilizando; es decir, si almacena un número (entero), si es texto o alfanumérico (cadena), si es un valor verdadero o falso (lógico) llamado también booleano.

**Ejemplo pseudocódigo**

```
//Variables
N : Entero
```

**Visual Basic**

```
'Variables
Dim N As Integer
```

Para asignarle un valor, usamos el operador de asignación, para algoritmos usaremos (←) o ( = ); este último es el más usado por los lenguajes de programación.

**Ejemplo pseudocódigo**

```
//Asignar un valor
N ← 10
//Cambiar su valor
N ←50
```

**Visual Basic**

```
'Asignar un valor
N = 10
'Cambiar su valor
N = 50
```

## 1.19 Constantes

Representa un espacio de memoria RAM, el cual guarda un valor que servirá para algún proceso en particular; dicho valor permanece fijo, es decir, no puede cambiarse en la ejecución del programa. Las constantes tienen, al igual que las variables, un identificador (nombre) y un tipo de dato.

**Ejemplo pseudocódigo**

```
//Constantes
PI ← 3.14159 : Real
//Error ya no puede modificarlo
PI ← 3.14
```

**Visual Basic**

```
'Constantes
Const PI As Integer = 3.14159
'Error ya no puede modificarlo
PI = 3.14
```

## 1.20 Tipo de datos simples (primitivos)

Al declarar una variable, debemos indicar el tipo de dato que es permitido almacenar en dicha variable. Cada lenguaje de programación trabaja con una variedad de tipo de datos; por lo general, todos usan los llamados «primitivos», que son los siguientes:

**A. Entero**: Representan los números enteros (no almacena decimales).

**Ejemplo pseudocódigo**

```
//Crear la variable
//(identificador y tipo de dato)
N : Entero

//Asignar un valor
//(identificador, operador de asignación y valor)
N ← 15
```

En el lenguaje de Visual Basic el tipo entero se puede trabajar con **short**, **int** y **long**; la diferencia está en que uno almacena rangos de números diferentes, llamados también «entero corto» y «entero largo».

**Ejemplo Visual Basic**

```
'Entero corto
Dim N As Integer
'Asignar un valor (error de desbordamiento)
'Sobrepaso su limite (rango)
```

```
N = 45000
'Entero largo
Dim N As Long
'Asignar un valor
N = 4500099
```

**B. Real:** Representan los números reales (almacena decimales).

**Ejemplo pseudocódigo**

```
//Crear la variable
//(identificador y tipo de dato)
N : Real
//Asignar un valor
//(identificador, operador de asignación y valor)
N ← 15.75
```

En el lenguaje de Visual Basic, el tipo real se puede trabajar con **float** o **double**, la diferencia está en la cantidad de decimales que pueden almacenar, llamados también «precisión simple» y «precisión doble».

```
'Precisión simple
Dim N As Single
'Se redondea a 15.123457
N = 15.12345678
'Precisión doble
Dim N As Double
'Lo almacena sin redondear 15.12345678
N = 15.12345678
```

**C. Carácter:** Representa un carácter de cualquier tipo: texto, número, símbolo, etc. El valor se coloca entre comillas simples.

**Ejemplo pseudocódigo**

```
//Crear la variable
R : Caracter
//Asignar un valor
R ← 'A'
R ← '9'
R ← '*'
```

**Ejemplo Visual Basic**

```
'Crear la variable
Dim R As String
'Asignar un valor
R = "A"
R = "9"
R = "*"
```

**E. Lógico:** Representan los valores «verdadero» o «falso», conocidos también como boolean, no se colocan comillas simple ni dobles.

**Ejemplo pseudocódigo**

```
//Crear la variable
L : Logico
//Asignar un valor
L ← VERDADERO
L ← FALSO
```

En Visual Basic se utiliza el tipo de dato llamado «boolean», para almacenar valores lógicos.

**Ejemplo Visual Basic**

```
'Crear la variable
Dim L As Boolean
'Asignar un valor
L = True
L = False
```

## 1.21 Tipo de datos complejos (estructurados)

Son aquellos que están constituidos por tipos de datos simples y definen una estructura de datos, un ejemplo claro es el tipo cadena, que esta compuesta por un conjunto de caracteres (tipo de dato caracter). Existe una variedad de tipo de de datos complejos, el enfoque de este libro es Algoritmos y solo tocaremos dos tipos de datos complejos que son cadena y arreglos, los libros que profundizan el tema se llaman libros de Estructura de datos.

**A. Cadena:** Representa un conjunto de caracteres, internamente es un arreglo de caracteres; por lo general, se representa con comillas dobles.

**Ejemplo pseudocódigo**

```
//Crear la variable
R : Cadena
//Asignar un valor
R ← "ricardomarcelo@hotmail.com"
```

## 1.22 Operadores y expresiones

Son los que permiten realizar los cálculos entre valores fijos y variables. Los operadores se clasifican en: aritméticos, relacionales, lógicos y de cadena. Sobre estos expondremos a continuación.

**A. Operadores aritméticos:** Son aquellos operadores que permiten realizar las operaciones aritméticas, de la misma forma como se utilizan en las matemáticas.

| Operador | Descripción |
|---|---|
| + | Suma |
| - | Resta |
| * | Multiplicación |
| / | División |
| \ | División entera |
| ^ | Exponenciación |
| Mod | Módulo (resto de una división) |

Dependiendo el lenguaje de programación los operadores varían o no implementan uno u otro, en el caso de Visual Basic implementa todos los mencionados en la tabla.

**Expresiones aritméticas**

| | | |
|---|---|---|
| 8 x 3 | Equivale a | 8*3 = 24 |
| $8 \div 3$ o $\frac{8}{3}$ | Equivale a | 8/3 = 2.666666<br>8\3 = 2 |
| $8^2$ | Equivale a | 8^2 = 64 |
| $\sqrt{9}$ | Equivale a | 9^(1/2) = 3 |
| $9 \mid 4$<br>$\underline{\quad}$<br>$(1) \quad 2$ | Equivale a | 9 Mod 4 = 1 |

**B. Operadores relacionales:** Llamados también operadores de comparación, y permiten evaluar si dos valores guardan alguna relación entre sí.

| Operador | Descripción |
|---|---|
| = | Igualdad |
| > | Mayor que |
| >= | Menor o igual que |
| < | Menor que |
| <= | Menor o Igual que |
| <> | Diferente a |

Dependiendo el lenguaje de programación, los operadores varían o no implementan uno u otro; en el caso de Visual Basic implementa todos los mencionados en la tabla.

**Expresiones lógicas (condiciones)**

| | |
|---|---|
| 8 = 3 | Falso |
| 8 > 3 | Verdadero |
| 8 <= 3 | Verdadero |
| 8 <> 8 | Falso |

**C. Operadores lógicos:** Son aquellos operadores que se utilizan en combinación con los operadores de relación.

| Operador | Descripción |
|---|---|
| Y | Y lógico |
| O | O lógico |
| No | No lógico |

**«Y» lógico:** Si p y q son valores lógicos, ambos deben ser verdaderos para que **Y** devuelva verdadero.

**Expresiones lógicas (condiciones)**

| | | | |
|---|---|---|---|
| 8 > 4 | Y | 3 = 6 | Falso |
| 7 <> 5 | Y | 5>=4 | Verdadero |

**«O» lógico:** Si p y q son valores lógicos, uno de ellos debe ser verdadero para que **O** devuelva verdadero.

**Expresiones lógicas (condiciones)**

| | | | |
|---|---|---|---|
| 8 > 4 | O | 3 = 6 | Verdadero |
| 7 <> 5 | Y | 5>=4 | Verdadero |

**«No» lógico:** Si p es un valor lógico, el operador **No** invierte su valor.

**Expresiones lógicas (condiciones)**

| | |
|---|---|
| NO (8 > 4) | Falso |
| NO (7 <> 7) | Verdadero |

**D. Operadores de cadena:** Son aquellos operadores que permiten realizar operaciones con cadenas; por lo general, permiten unir en cadena, lo cual es llamado también «concatenar».

| Operador | Descripción |
|---|---|
| + | Unir cadenas |
| & | Unir cadenas |

**Expresiones de cadena**

| | |
|---|---|
| “Ricardo” + “ “ + “Marcelo” | Ricardo Marcelo |
| “ricardomarcelo” & “@” & “hotmail.com” | ricardomarcelo@hotmail.com |

## 1.23 Control de flujo

Todos los lenguajes de programación implementan estructuras para controlar la ejecución de un programa, estas son:

- Estructura secuencial
- Estructura selectiva simple y doble
- Estructura selectiva múltiple
- Estructura repetitiva mientras
- Estructura repetitiva para

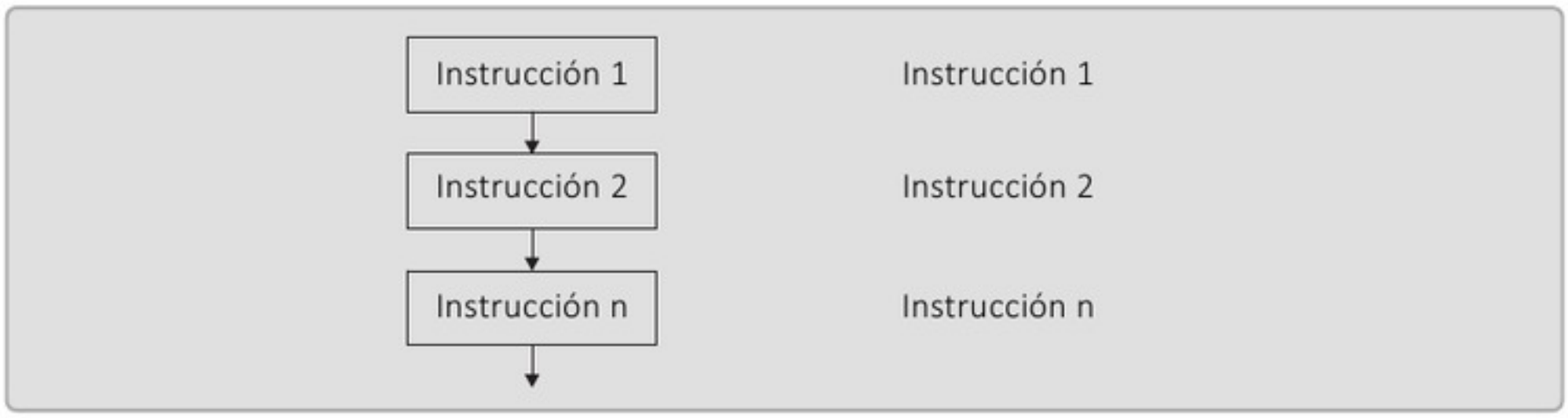
En los siguientes capítulos se explicarán cada una de las estructuras mencionadas.

# Capítulo 2

# Estructura secuencial

## 2.1 Estructura secuencial

Son aquellos algoritmos que ejecutan instrucciones en forma consecutiva, es decir, uno detrás de otro, hasta finalizar el proceso.




### Problema n.° 1

**Enunciado**: Dados dos números enteros, hallar la suma.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese dos números enteros; luego, que el sistema realice el cálculo respectivo para hallar la suma. Para ello usará la siguiente expresión.

**Expresión matemática**

s = n1 + n2

**Expresión en el lenguaje de programación**

s ← n1 + n2

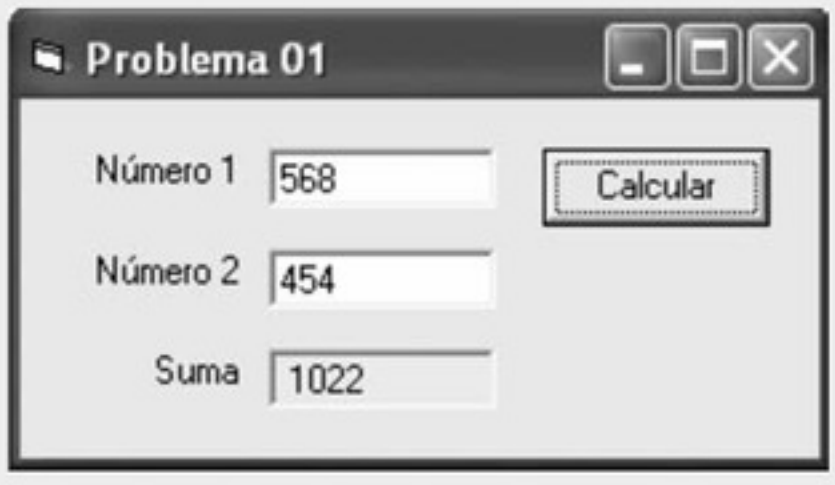
**Entrada**

- Dos números (n1 y n2)

**Salida**

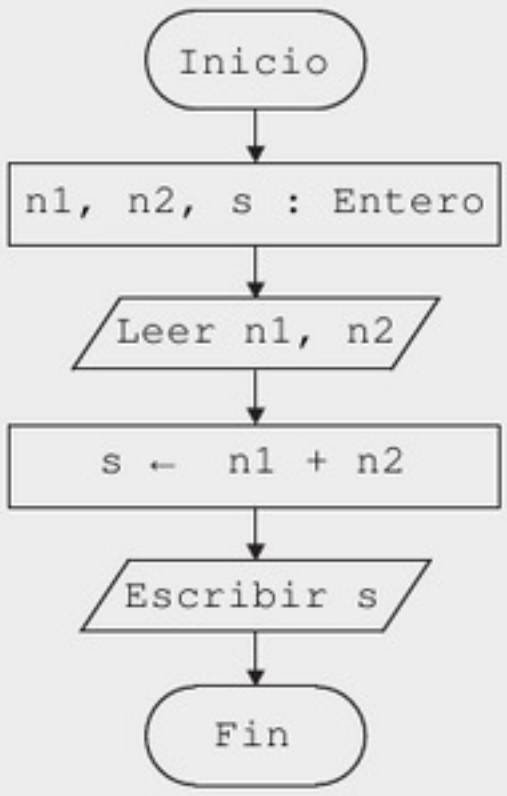
- La suma de los números ingresados (s)

**Diseño: Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n1, n2, s : Entero

   //Entrada
   Leer n1, n2

   //Proceso
   s ← n1 + n2

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim n1 As Integer
Dim n2 As Integer
Dim s As Integer

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
s = n1 + n2

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 2

**Enunciado**: Hallar el cociente y el residuo (resto) de dos números enteros.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese dos números enteros; luego, que el sistema realice el cálculo respectivo para hallar el cociente y residuo. Para esto use la siguiente expresión:

**Expresión en el lenguaje de programación**

c ← n1 / n2

r ← n1 Mod n2

**Entrada**

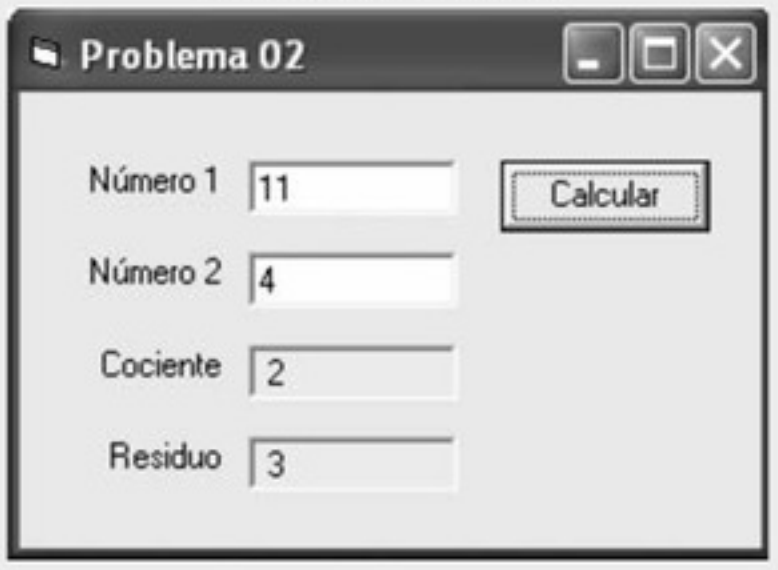
- Dos números (n1 y n2)

**Salida**

- El cociente (c)
- El Residuo (r)

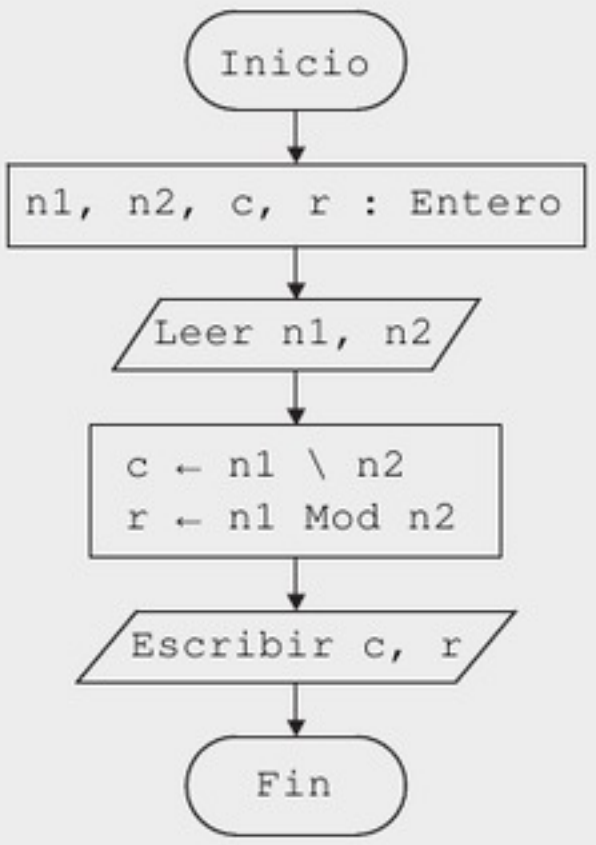
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n1, n2, c, r : Entero

   //Entrada
   Leer n1, n2

   //Proceso
   c ← n1 \ n2
   r ← n1 Mod n2

   //Salida
   Escribir c, r

Fin
```

**Codificación:**

```
'Variables
Dim n1 As Integer
Dim n2 As Integer
Dim c As Integer
Dim r As Integer

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
c = n1 \ n2
r = n1 Mod n2

'Salida
Me.txtc.Text = Str(c)
Me.txtr.Text = Str(r)
```

## Problema n.° 3

**Enunciado**: Dado el valor de venta de un producto, hallar el IGV (19 %) y el precio de venta.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el valor de venta del producto; luego, que el sistema realice el cálculo respectivo para hallar el IGV y el precio de venta. Para esto use la siguiente expresión:

**Expresión en el lenguaje de programación**

igv ← vv * 0.19

pv ← vv + igv

**Entrada**

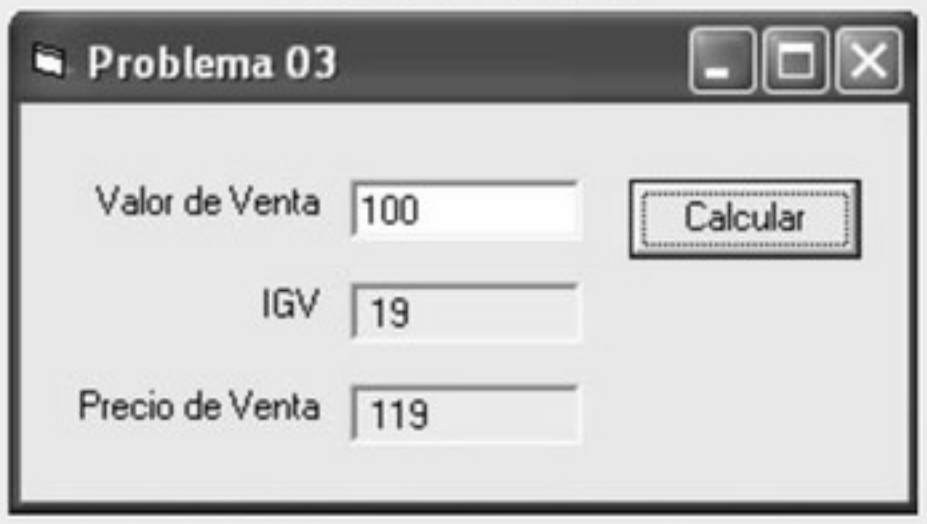
- Valor de venta (vv)

**Salida**

- El IGV (igv)
- El precio de venta (pv)

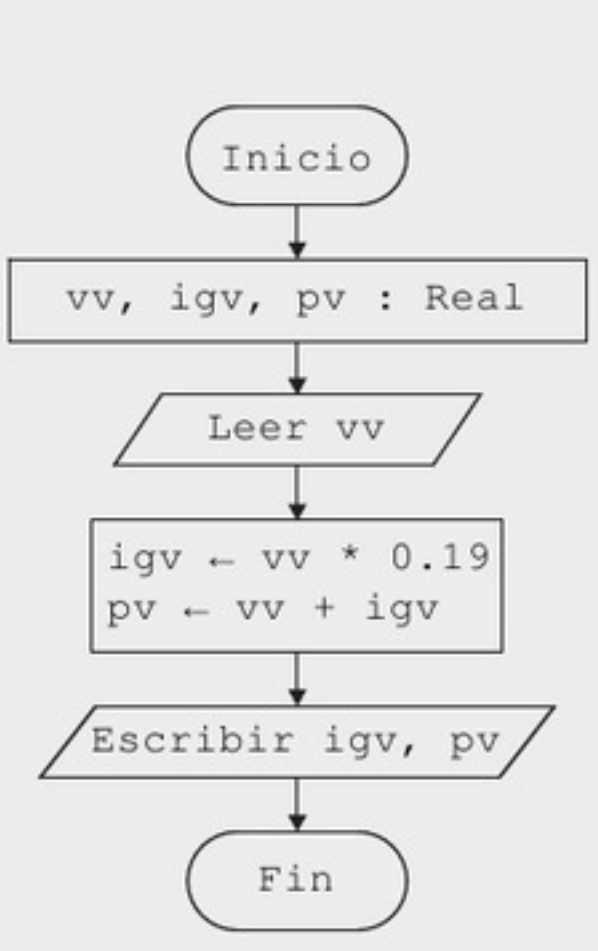
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   vv, igv, pv : Real

   //Entrada
   Leer vv

   //Proceso
   igv ← vv * 0.19
   pv ← vv + igv

   //Salida
   Escribir igv, pv

Fin
```

**Codificación:**

```
'Variables
Dim vv As Single
Dim igv As Single
Dim pv As Single

'Entrada
vv = Val(Me.txtvv.Text)

'Proceso
igv = vv * 0.19
pv = vv + igv

'Salida
Me.txtigv.Text = Str(igv)
Me.txtpv.Text = Str(pv)
```

## Problema n.° 4

**Enunciado**: Hallar la potencia de $a^n$, donde «a» y «n» pertenecen a $Z^+$ (números enteros positivos).

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese dos números enteros positivos «a» y «n»; luego, que el sistema procese y obtenga la potencia «p».

**Expresión matemática**

$$p = a^n = \underbrace{a \times a \times a \times \ldots \times a}_{\text{n factores}}$$

**Expresión en el lenguaje de programación**

p ← a ^ n

**Entrada**

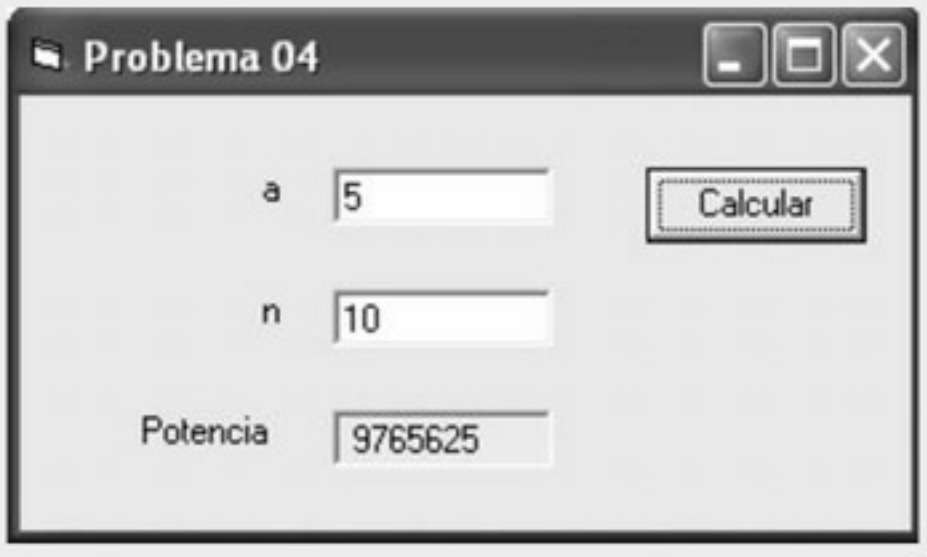
- Dos números enteros (a, n)

**Salida**

- La potencia (p)

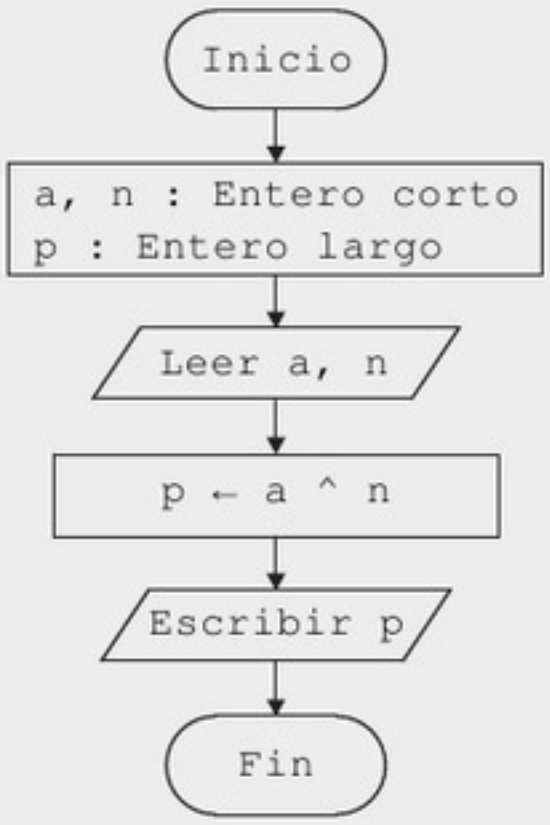
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   a, n : Entero corto
   p : Entero largo

   //Entrada
   Leer a, n

   //Proceso
   p ← a ^ n

   //Salida
   Escribir p

Fin
```

**Codificación:**

```
'Variables
Dim a As Integer
Dim n As Integer
Dim p As Long

'Entrada
a = Val(Me.txta.Text)
n = Val(Me.txtn.Text)

'Proceso
p = a ^ n

'Salida
Me.txtp.Text = Str(p)
```

## Problema n.° 5

**Enunciado**: Hallar la radicación de $\sqrt[n]{a}$ , donde «a» y «n» pertenecen a $Z^+$ (números enteros positivos).

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese dos números enteros positivos «a» y «n»; luego, que el sistema procese y obtienga la radicación «r».

**Expresión matemática**

$$r = \sqrt[n]{a} = a^{\frac{1}{n}}$$

**Expresión en el lenguaje de programación**

r ← a ^ (1/n)

**Entrada**

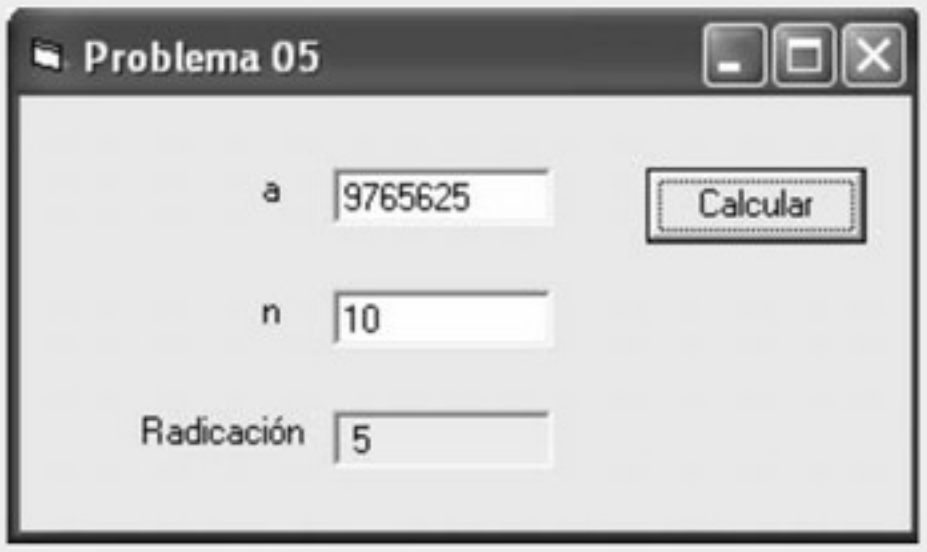
- Dos números enteros (a, n)

**Salida**

- La radicación (r)

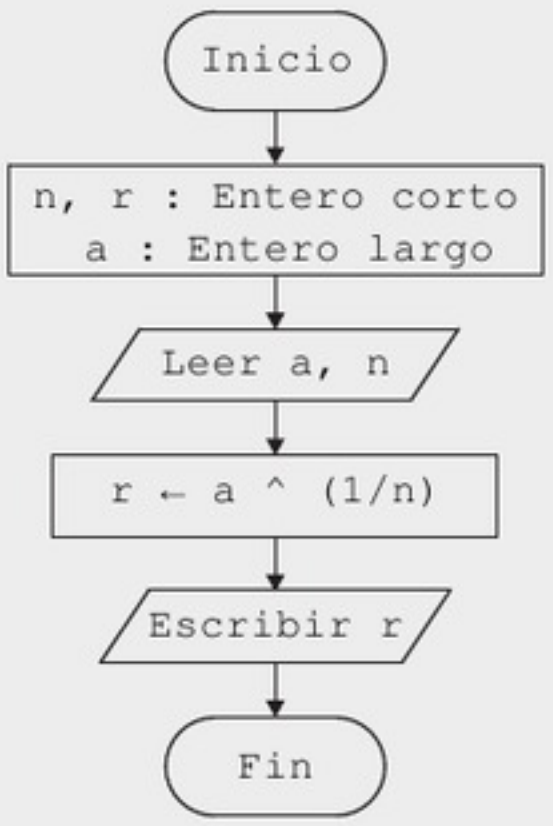
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, r : Entero corto
   a : Entero largo

   //Entrada
   Leer a, n

   //Proceso
   r ← a ^ (1/n)

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim r As Integer
Dim a As Long

'Entrada
a = Val(Me.txta.Text)
n = Val(Me.txtn.Text)

'Proceso
r = a ^ (1/n)

'Salida
Me.txtr.Text = Str(r)
```

## Problema n.° 6

**Enunciado**: Dado un número de 5 dígitos, devolver el número en orden inverso.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número «n»; luego, que el sistema procese y obtenga el número inverso «ni», realizando 4 divisiones sucesivas entre 10, para acumular el residuo y el último cociente.

```
12345 | 10
 ⑤ 1234 | 10
     ④ 123 | 10
         ③ 12 | 10
            ② ①
```

**Entrada**

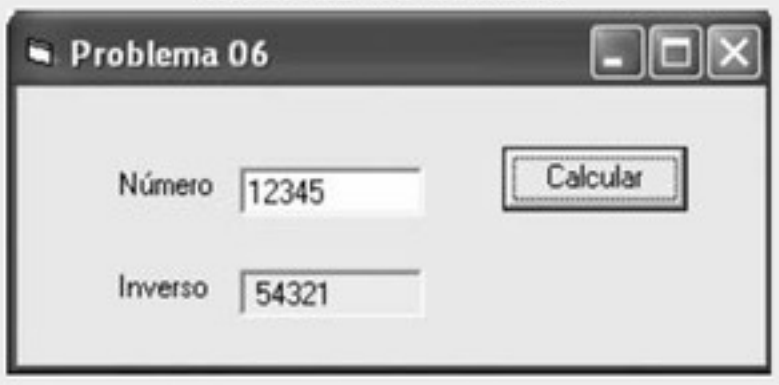
- Un número entero (n)

**Salida**

- El número inverso (ni)

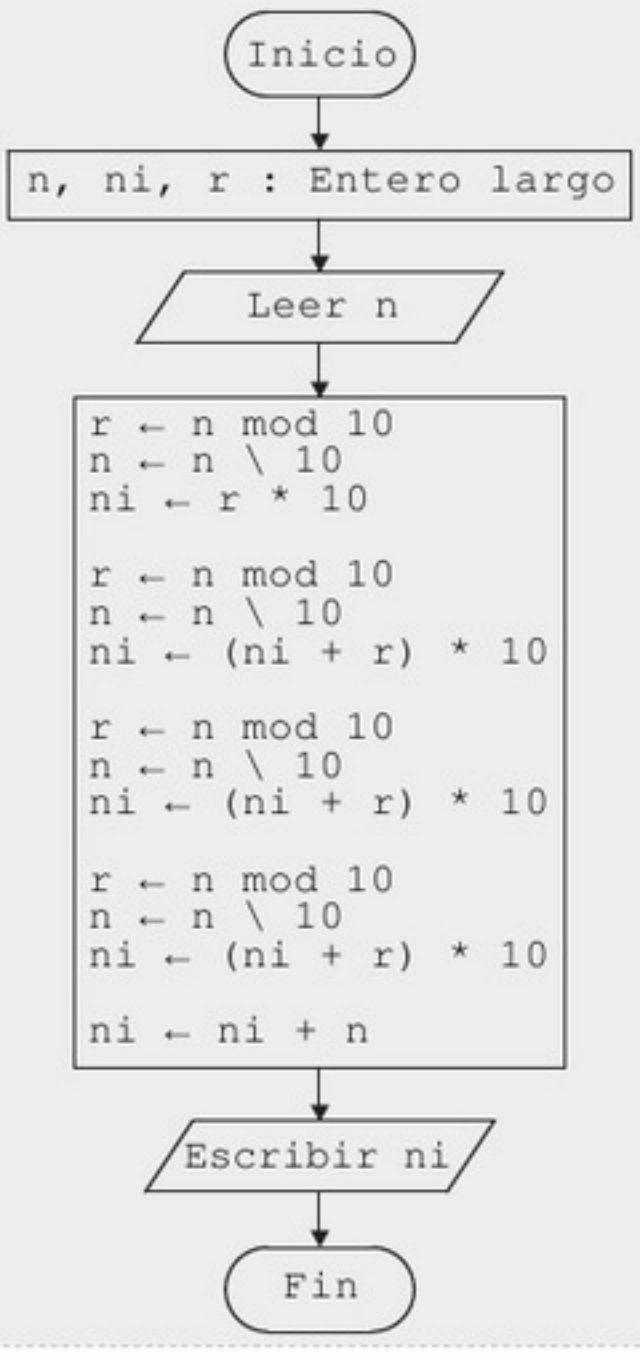
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio
   //Variables
   n, ni, r : Entero largo
   //Entrada
   Leer n
   //Proceso
   r ← n mod 10
   n ← n \ 10
   ni ← r * 10

   r ← n mod 10
   n ← n \ 10
   ni ← (ni + r) * 10

   r ← n mod 10
   n ← n \ 10
   ni ← (ni + r) * 10

   r ← n mod 10
   n ← n \ 10
   ni ← (ni + r) * 10

   r ← n mod 10
   n ← n \ 10
   ni ← (ni + r) * 10

   ni ← ni + n
   //Salida
   Escribir ni
Fin
```

**Codificación:**

```
'Variables
Dim n As Long
Dim ni As Long
Dim r As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
r = n mod 10
n = n \ 10
ni = r * 10

r = n mod 10
n = n \ 10
ni = (ni + r) * 10

r = n mod 10
n = n \ 10
ni = (ni + r) * 10

r = n mod 10
n = n \ 10
ni = (ni + r) * 10

ni = ni + n

'Salida
Me.txtni.Text = Str(ni)
```

## Problema n.° 7

**Enunciado**: Determinar la suma de los N primeros números enteros positivos ($Z^+$); use la siguiente fórmula:

$$S = \frac{N(N+1)}{2}$$

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero positivo «n»; luego, que el sistema procese y obtenga la suma de los primeros números enteros positivos hasta «n».

**Expresión matemática**

$$S = \frac{N(N+1)}{2}$$

**Expresión en el lenguaje de programación**

s ← (n * (n + 1)) / 2

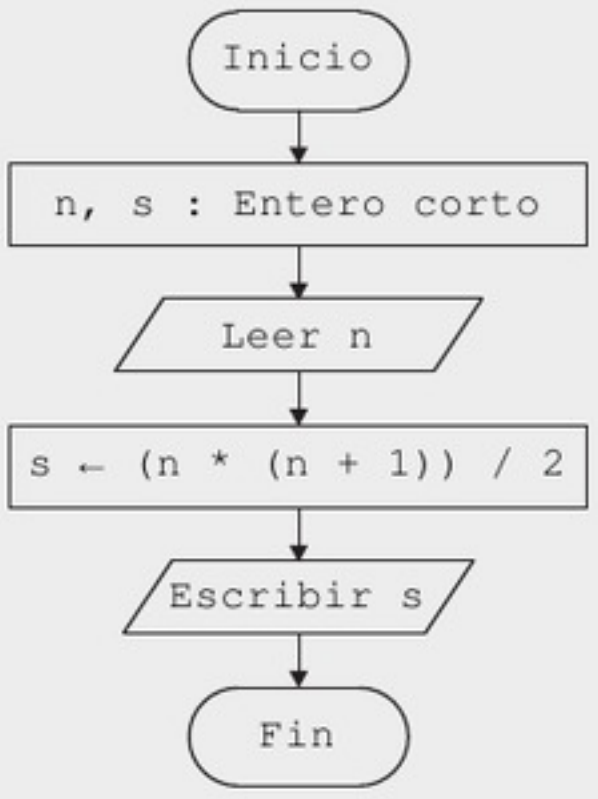
| Entrada | Salida |
|---|---|
| • Número entero (n) | • Suma (s) |

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, s : Entero corto

   //Entrada
   Leer n

   //Proceso
   s ← (n * (n + 1)) / 2

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim s As Integer

'Entrada
n = Val(Me.txtn.Text)

'Proceso
s = (n * (n + 1)) / 2

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 8

**Enunciado**: Calcular el interés compuesto generado por un capital depositado durante cierta cantidad de tiempo, a una tasa de interés determinada; aplique las siguientes fórmulas:

$$M = (1 + r\%)^t.C$$

$$I = M - C$$

**Monto (M):** Es la suma del capital más los intereses producidos en determinado tiempo.

**Tasa de interés (r%):** Es la ganancia que se obtiene por cada 100 unidades monetarias en cada periodo de tiempo.

**Capital (C):** Es todo aquello que se va a ceder o imponer durante algún tiempo para generar una ganancia.

**Interés (I):** Parte de la utilidad que obtiene el capitalista al prestar su dinero.

**Tiempo (t):** Es el periodo de tiempo durante el cual se cede el capital.

**Análisis**: Para la solución de este problema se requiere que el usuario ingrese el capital «c» y la tasa de interés «r»; luego, que el sistema procese y obtenga el interés ganado y el monto producido.

**Expresión matemática**

$$M = (1 + r\%)^t.C$$

**Expresión en el lenguaje de programación**

m ← ((1 + r / 100) ^ t) * c

**Entrada**

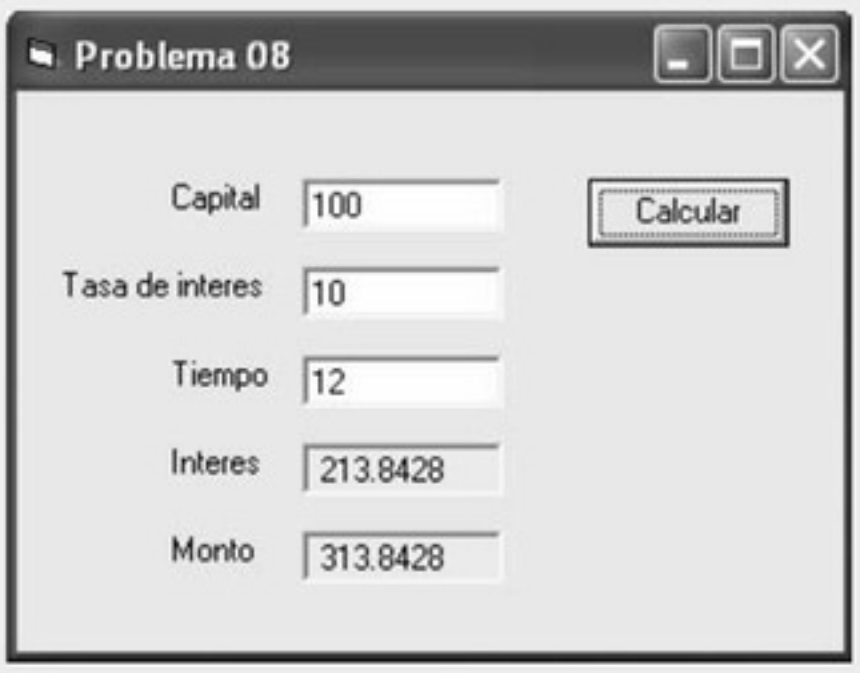
- Capital (c)
- Tasa de interés (r)
- Tiempo (t)

**Salida**

- Interés (i)
- Monto (m)

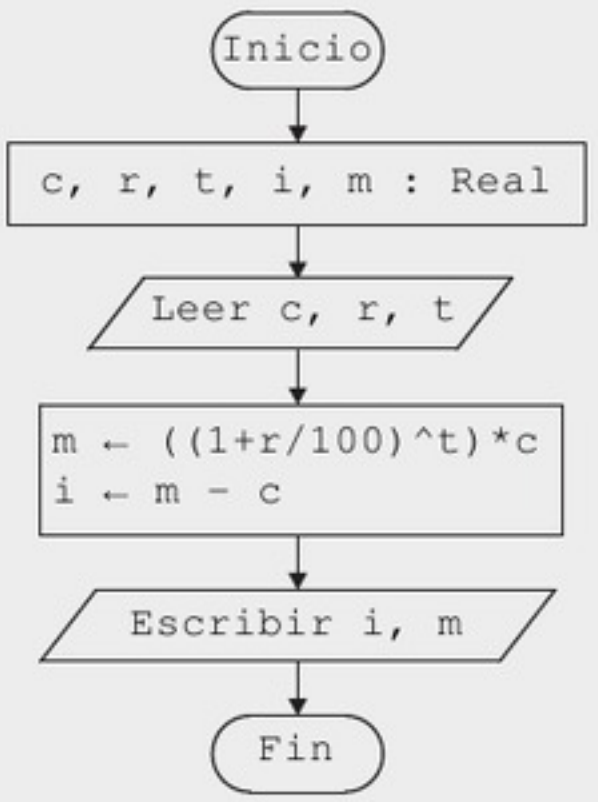
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c, r, t, i, m : Real

   //Entrada
   Leer c, r, t

   //Proceso
   m ← ((1+r/100)^t)*c
   i ← m - c

   //Salida
   Escribir i, m

Fin
```

**Codificación:**

```
'Variables
Dim c As Single
Dim r As Single
Dim t As Single
Dim i As Single
Dim m As Single

'Entrada
c = Val(Me.txtc.Text)
r = Val(Me.txtr.Text)
t = Val(Me.txtt.Text)

'Proceso
m = ((1 + r / 100) ^ t) * c
i = m - c

'Salida
Me.txti.Text = Str(i)
Me.txtm.Text = Str(m)
```

## Problema n.° 9

**Enunciado**: Crear un programa para encontrar el área de un círculo, usan la siguiente fórmula:

$$A = \pi.r^2$$

**Área (A):** Es el área del círculo.

**PI (π):** Representa el valor constante pi (3.14159).

**Radio (r):** Es el radio del círculo.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el radio del círculo; luego, que el sistema procese y obtenga el área del círculo.

**Expresión algebraica**

$$A = \pi.r^2$$

**Expresión en el lenguaje de programación**

A ← 3.14159 * r ^ 2

**Entrada**

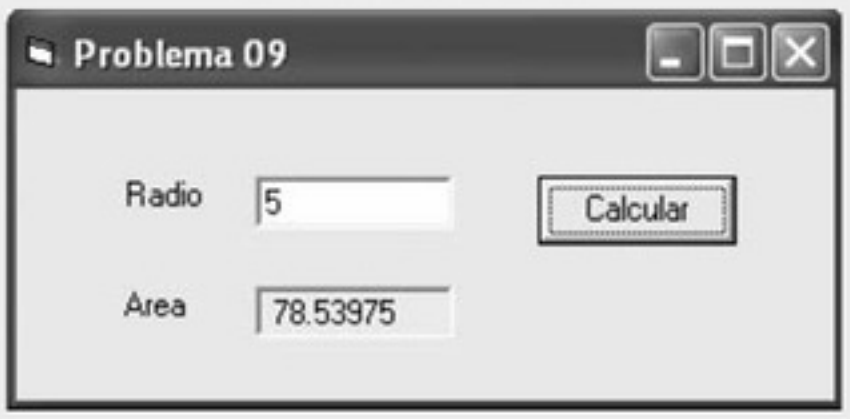
- Radio (r)

**Salida**

- Área (a)

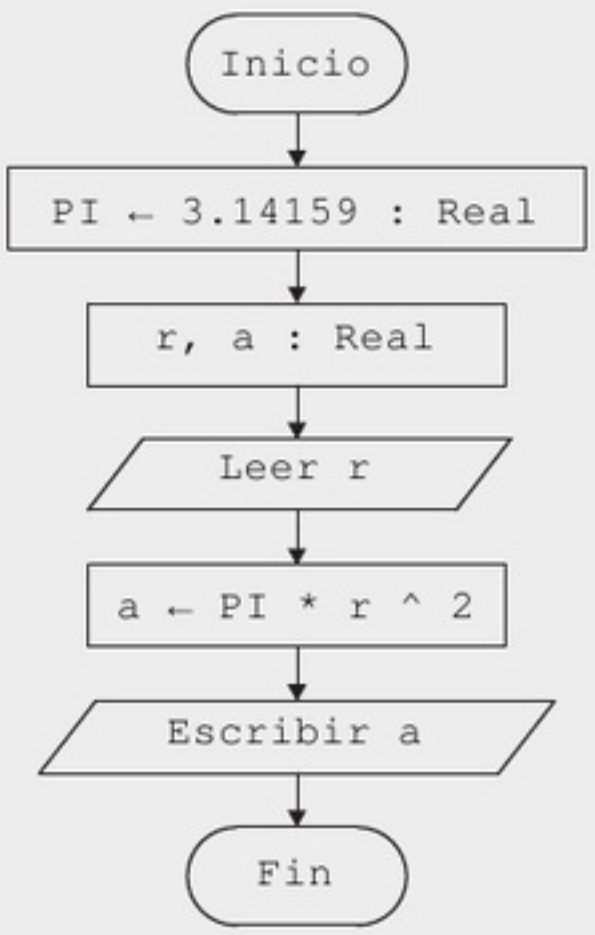
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Constantes
   PI = 3.14159 : Real

   //Variables
   r, a : Real

   //Entrada
   Leer r

   //Proceso
   a ← PI * r ^ 2

   //Salida
   Escribir a

Fin
```

**Codificación:**

```
'Constante
Const PI As Single = 3.14159

'Variables
Dim a As Single
Dim r As Single

'Entrada
r = Val(Me.txtr.Text)

'Proceso
a = PI * r ^ 2

'Salida
Me.txta.Text = Str(a)
```

## Problema n.° 10

**Enunciado**: Crear un programa que permita convertir una cantidad de segundos en horas, minutos y segundos.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un tiempo expresado en segundos; luego, que el sistema procese y obtenga el área del círculo.

**Entrada**

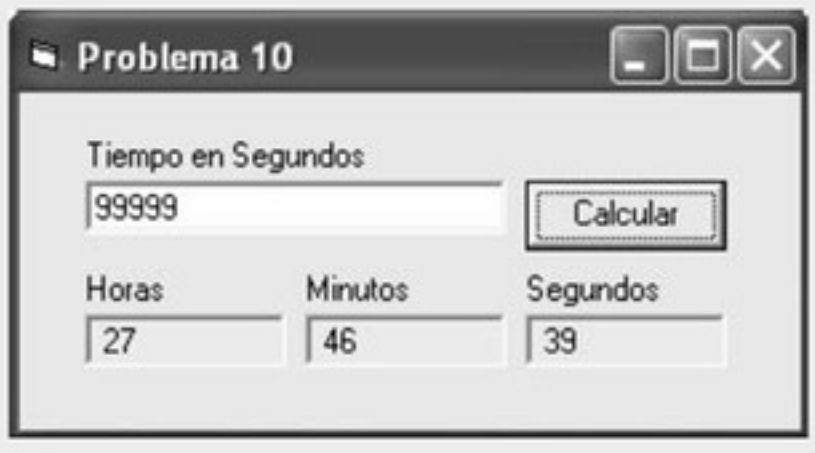
- Tiempo en segundos (t)

**Salida**

- Horas (h)
- Minutos (m)
- Segundos (s)

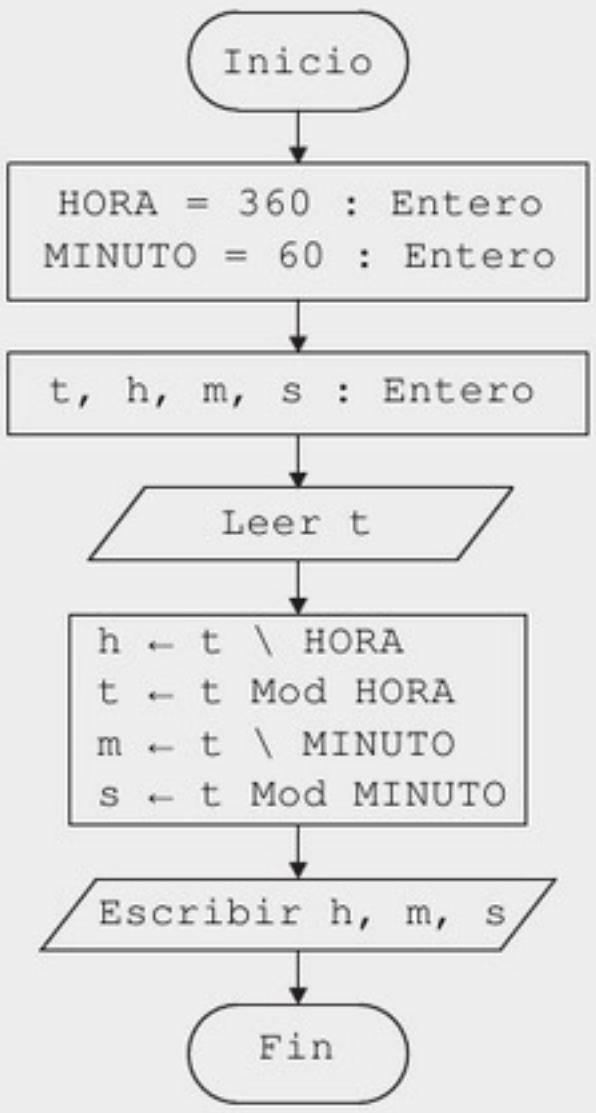
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Constantes
   HORA = 360 : Entero
   MINUTO = 60 : Entero

   //Variables
   t, h, m, s : Entero

   //Entrada
   Leer t

   //Proceso
   h ← t \ HORA
   t ← t Mod HORA
   m ← t \ MINUTO
   s ← t Mod MINUTO

   //Salida
   Escribir h, m, s

Fin
```

**Codificación:**

```
'Constantes
Const HORA As Long = 3600
Const MINUTO As Long = 60

'Variables
Dim t As Long
Dim h As Long
Dim m As Long
Dim s As Long

'Entrada
t = Val(Me.txtt.Text)

'Proceso
h = t \ HORA
t = t Mod HORA
m = t \ MINUTO
s = t Mod MINUTO

'Salida
Me.txth.Text = Str(h)
Me.txtm.Text = Str(m)
Me.txts.Text = Str(s)
```

## 2.2 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 1

**Enunciado**: Dados dos números enteros (Z), a y b, hallar a + b y a – b.

### Propuesto n.° 2

**Enunciado**: Dados dos números enteros, determinar cuántos números enteros están incluidos en ellos.

### Propuesto n.° 3

**Enunciado**: Dada una cantidad de milímetros, expresarlo en la máxima cantidad de metros, el resto en decímetros, centímetros y milímetros.

### Propuesto n.° 4

**Enunciado**: Obtener el valor de «c» y «d» de acuerdo a la siguiente fórmula:

$$c = \frac{(4a^4 + 3ba + b^2)}{a^2 - b^2} \qquad d = \frac{(3c^2 + a + b)}{4}$$

### Propuesto n.° 5

**Enunciado**: Dado 4 números enteros, obtener el porcentaje de cada uno en función a la suma de los 4 números ingresados.

### Propuesto n.° 6

**Enunciado**: Hallar el área y el perímetro de un cuadrado.

### Propuesto n.° 7

**Enunciado**: Dada una cantidad de horas, obtener su equivalente en minutos y segundos.

### Propuesto n.° 8

**Enunciado**: Convertir una cantidad de grados Fahrenheit a Celsius y Kelvin.

### Propuesto n.° 9

**Enunciado**: Hallar el área y el perímetro de un rectángulo.

### Propuesto n.° 10

**Enunciado**: Convertir grados sexagesimales a centesimales.

# Capítulo 3

# Estructura selectiva simple y doble

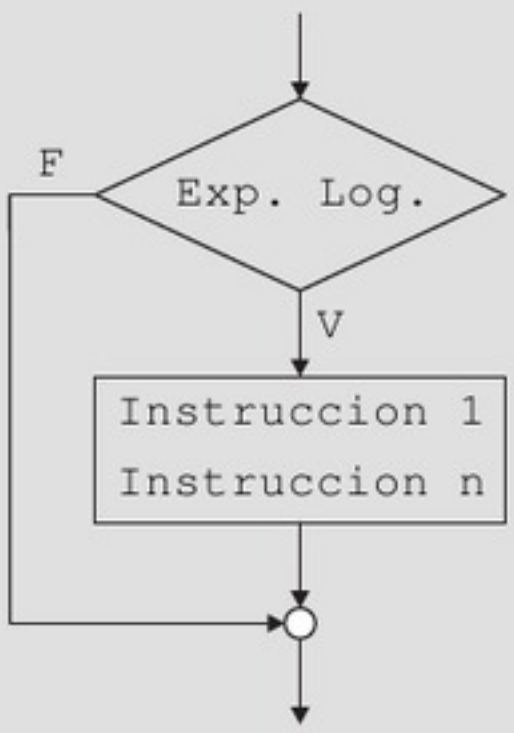
## 3.1 Introducción

Muchas veces tenemos que decidir si realizar una u otra tarea, dependiendo de una condición; en la programación existe una estructura que permite evaluar una condición (expresión lógica que devuelve «verdadero» o «falso») y determina qué instrucción o instrucciones se deben ejecutar si la condición es verdadera o si la condición es falsa.

En este capítulo usted aprenderá a resolver problemas que permitan evaluar condiciones lógicas; esta es una de las estructuras básicas y más utilizadas en todo lenguaje de programación, también se las conoce como estructuras condicionales, alternativas y de decisiones.

## 3.2 Estructura selectiva simple

Evalúa una expresión lógica (condición), si es verdadera ejecuta una determinada instrucción o instrucciones.




**Si** <Exp. Log.> **Entonces**

<Instruccion 1>

<Instruccion n>

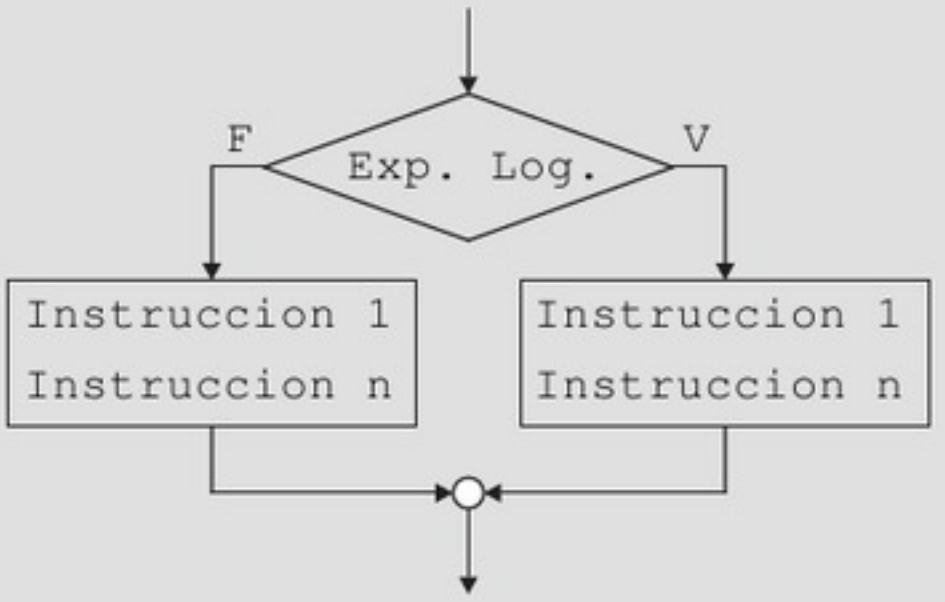
**Fin Si**

**Sintaxis Visual Basic**

```
If      <Exp. Log.> Then

        <Instruccion 1>
        <Instruccion n>
End If
```

## 3.3 Estructura selectiva doble

Evalúa una expresión lógica (condición), si es verdadera ejecuta una o varias instrucciones; si es falsa, ejecuta otro grupo de instrucciones.



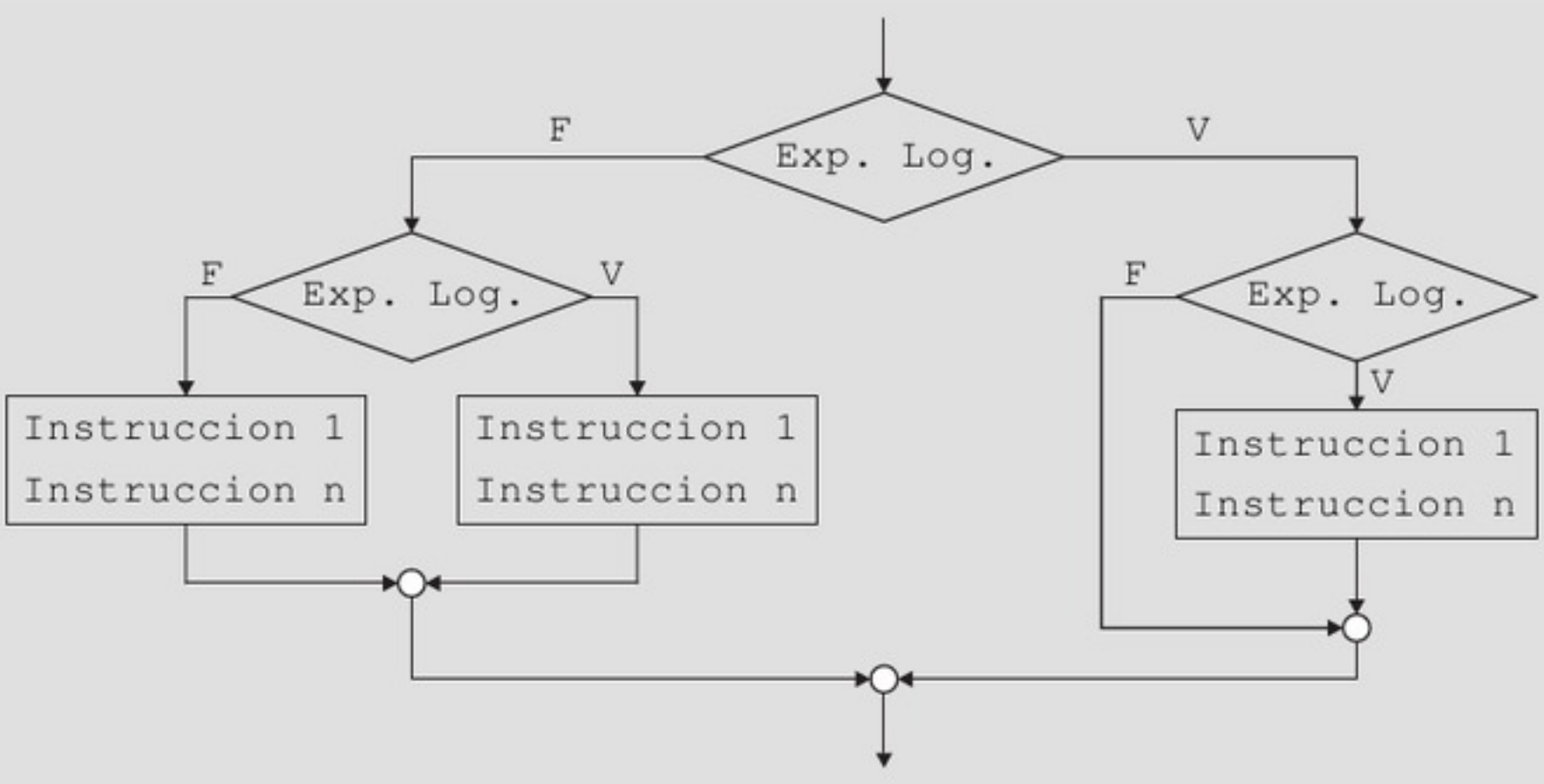

```
Si      <Exp. Log.>      Entonces
        <Instruccion 1>
        <Instruccion n>
SiNo
        <Instruccion 1>
        <Instruccion n>
Fin     Si
```

**Sintaxis Visual Basic**

```
If      <Exp. Log.>      Then
        <Instruccion 1>
        <Instruccion n>
Else
        <Instruccion 1>
        <Instruccion n>


End If
```

## 3.4 Estructuras anidadas

Son aquellas estructuras que contienen una o más estructuras; es decir, está permitido colocar dentro de una estructura, otra estructura.



```
Si          <Exp. Log.> Entonces
    Si      <Exp. Log.> Entonces
            <Instruccion 1>
            <Instruccion n>
    Fin Si
SiNo
    Si      <Exp. Log.> Entonces
            <Instruccion 1>
            <Instruccion n>
    SiNo
            <Instruccion 1>
            <Instruccion n>
    Fin Si
Fin Si
```

**Sintaxis Visual Basic**

```
If          <Exp. Log.> Then
    If      <Exp. Log.> Then
            <Instruccion 1>
            <Instruccion n>
    End If
Else
    If      <Exp. Log.> Then
            <Instruccion 1>
            <Instruccion n>
    Else
            <Instruccion 1>
            <Instruccion n>
    End If
End If
```

## Problema n.° 11

**Enunciado**: Dados dos números enteros diferentes, devolver el número mayor.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese dos números enteros diferentes; luego, que el sistema realice el proceso para devolver el número mayor.

**Expresión**

Si n1 > n2 ⇒ n1 es mayor

Si n2 > n1 ⇒ n2 es mayor

**Entrada**

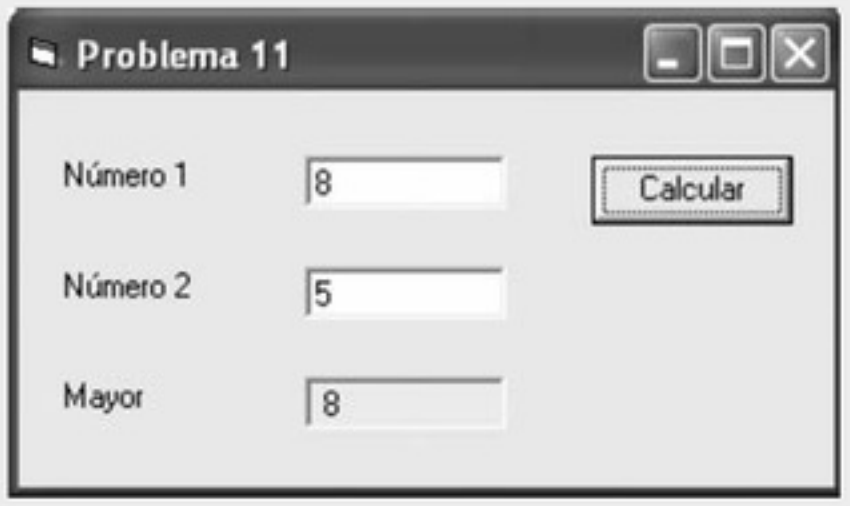
- Dos números (n1 y n2)

**Salida**

- Número mayor (m)

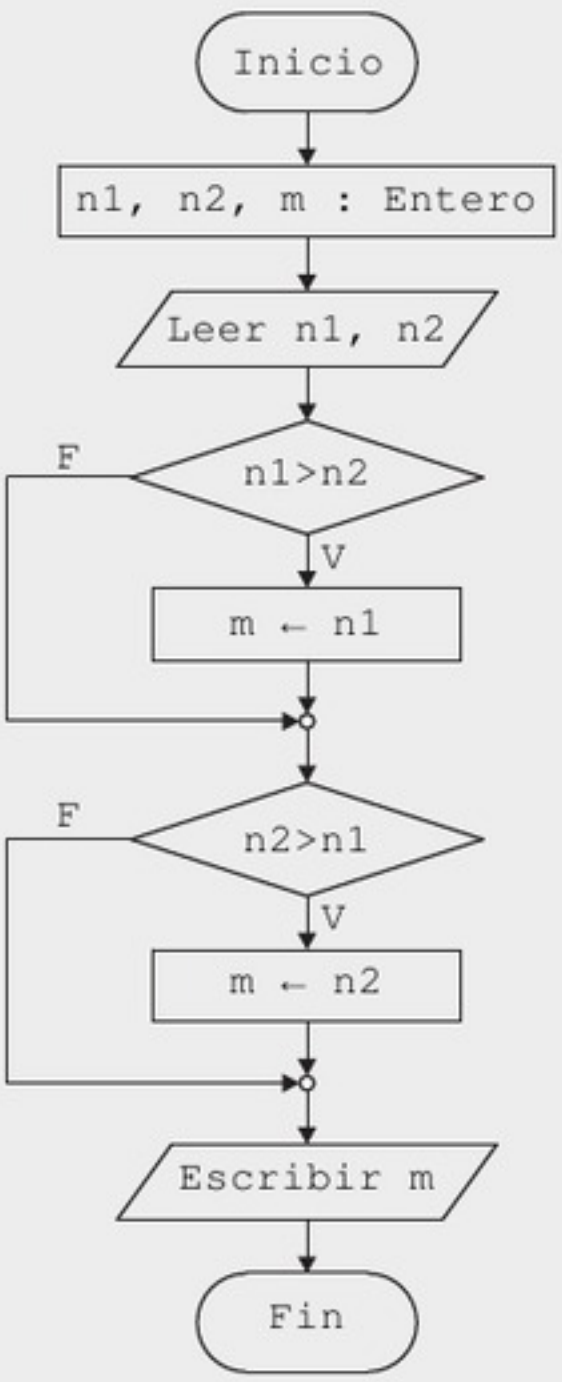
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n1, n2, m : Entero

   //Entrada
   Leer n1, n2

   //Proceso
   Si  n1 > n2 Entonces
       m ← n1
   Fin Si

   Si  n2 > n1 Entonces
       m ← n2
   Fin Si

   //Salida
   Escribir m

Fin
```

**Codificación:**

```
'Variables
Dim n1 As Integer
Dim n2 As Integer
Dim m As Integer

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
If n1 > n2 Then
    m = n1
End If

If n2 > n1 Then
    m = n2
End If

'Salida
Me.txtm.Text = Str(m)
```

## Problema n.° 12

**Enunciado**: Determinar si un número entero es positivo, negativo o neutro.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero; luego, que el sistema verifique si es «positivo», «negativo» o «neutro».

**Expresión**

Si n > 0 ⇒ POSITIVO

Si n < 0 ⇒ NEGATIVO

Si n = 0 ⇒ NEUTRO

**Entrada**

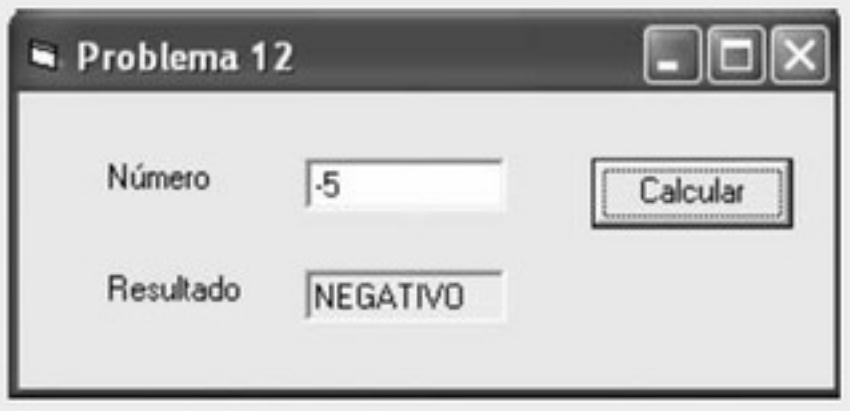
- Número (n)

**Salida**

- Resultado (r )
  - POSITIVO
  - NEGATIVO
  - NEUTRO

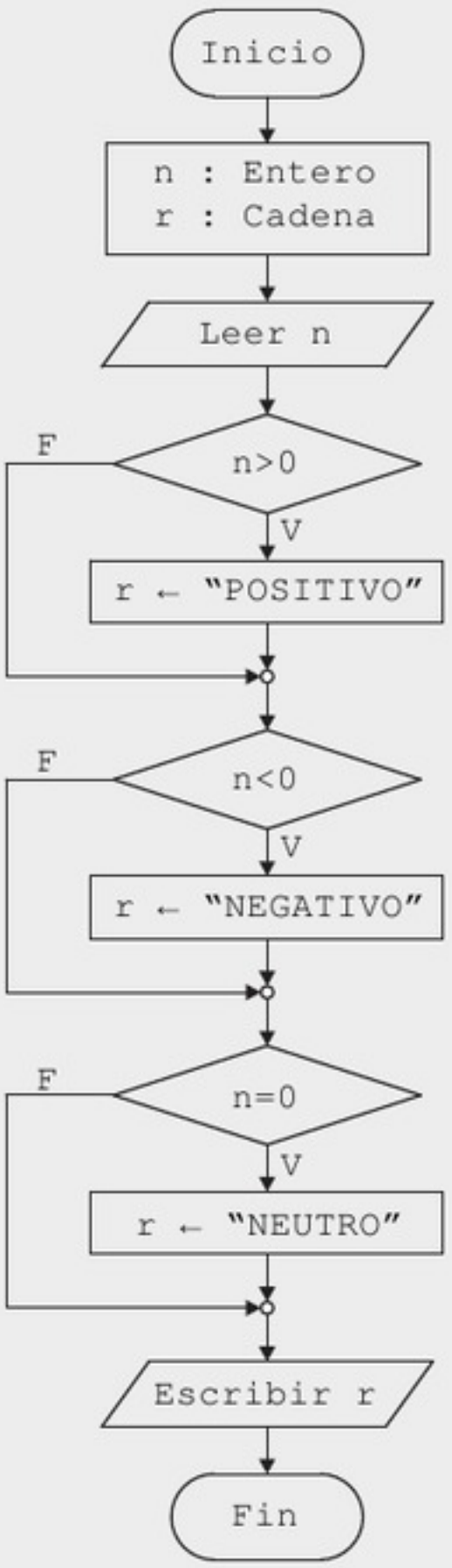
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   Si n > 0 Entonces
       r ← "POSITIVO"
   Fin Si

   Si n < 0 Entonces
       r ← "NEGATIVO"
   Fin Si

   Si n = 0 Entonces
       r ← "NEUTRO"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
If n > 0 Then
     r = "POSITIVO"
End If

If n < 0 Then
     r = "NEGATIVO"
End If

If n = 0 Then
     r = "NEUTRO"
End If
'Salida
Me.txtr.Text = r
```

## Problema n.° 13

**Enunciado**: Dado un carácter, determinar si es una vocal.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un carácter; luego, que el sistema verifique si es una vocal.

**Expresión**

Si c = 'a' v c = 'A' ⇒ VOCAL

Si c = 'e' v c = 'E' ⇒ VOCAL

Si c = 'i' v c = 'I' ⇒ VOCAL

Si c = 'o' v c = 'O' ⇒ VOCAL

Si c = 'u' v c = 'U' ⇒ VOCAL

**Entrada**

- Carácter (c)

**Salida**

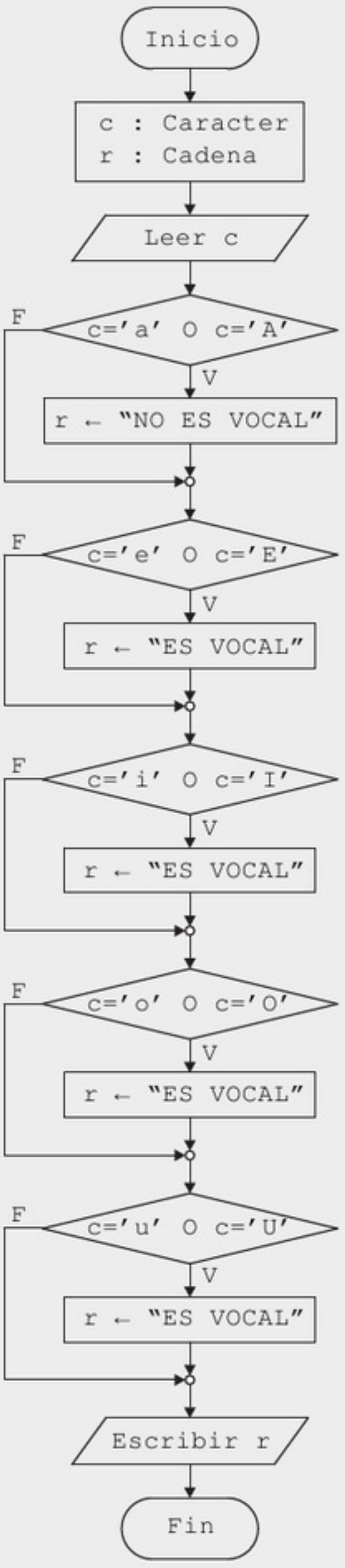
- Resultado (r)
  - ES VOCAL
  - NO ES VOCAL

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c : Caracter
   r : Cadena

   //Entrada
   Leer c

   //Proceso
   r ← "NO ES VOCAL"

   Si c='a' O c='A' Entonces
       r ← "ES VOCAL"
   Fin Si

   Si c='e' O c='E' Entonces
       r ← "ES VOCAL"
   Fin Si

   Si c='i' O c='I' Entonces
       r ← "ES VOCAL"
   Fin Si

   Si c='o' O c='O' Entonces
       r ← "ES VOCAL"
   Fin Si

   Si c='U' O c='U' Entonces
       r ← "ES VOCAL"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim c As String
Dim r As String

'Entrada
c = Me.txtc.Text

'Proceso
r = "NO ES VOCAL"
If c = "a" Or c = "A" Then
    r = "ES VOCAL"
End If
If c = "e" Or c = "E" Then
    r = "ES VOCAL"
End If
If c = "i" Or c = "I" Then
    r = "ES VOCAL"
End If
If c = "o" Or c = "O" Then
    r = "ES VOCAL"
End If
If c = "u" Or c = "U" Then
    r = "ES VOCAL"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 14

**Enunciado**: Determinar si un número es múltiplo de 3 y 5.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero n; luego, que el sistema analice y determine si el número es múltiplo de 3 y de 5.

**Expresión**

Si n Mod 3 = 0 ^ n Mod 5 = 0 ⇒

SI ES MULTIPLO DE 3 y 5

SiNo

NO ES MULTIPLO DE 3 y 5

**Entrada**

- Número (n)

**Salida**

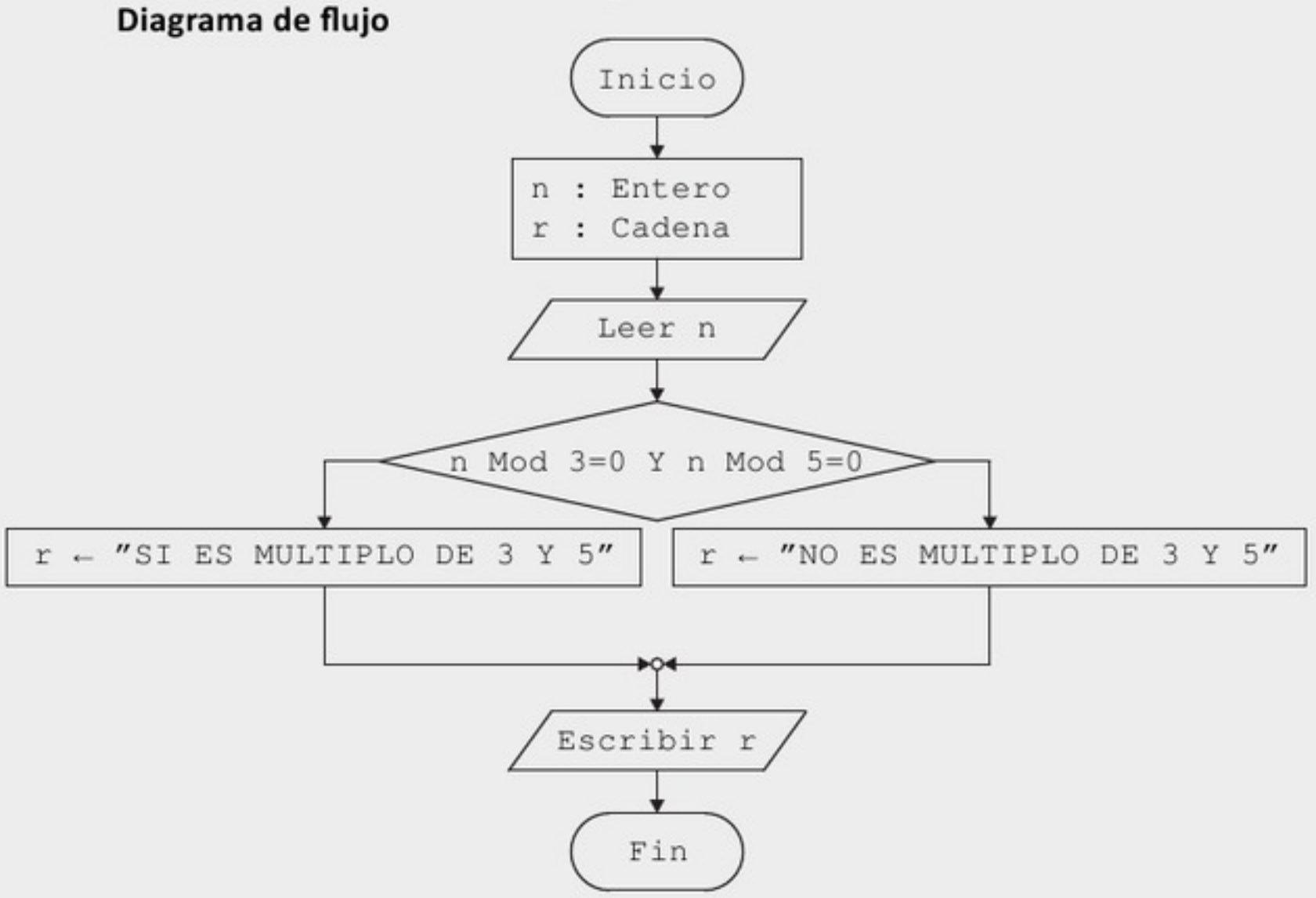
- Resultado (r)
  - ES MULTIPLO
  - NO ES MULTIPLO

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio
   //Variables
   n : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   Si n Mod 3 = 0 Y n Mod 5 = 0 Entonces
       r ← "SI ES MULTIPLO DE 3 y 5"
   SiNo
       r ← "NO ES MULTIPLO DE 3 y 5"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
If n Mod 3 = 0 And n Mod 5 = 0 Then
    r = "SI ES MULTIPLO DE 3 Y 5"
Else
    r = "NO ES MULTIPLO DE 3 Y 5"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 15

**Enunciado**: Determinar si un número entero es par o impar.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero n; luego, que el sistema verifique si el número es par o impar.

**Expresión**

Si n Mod 2 = 0 ⇒
    PAR
SiNo
    IMPAR

**Entrada**

- Número (n)

**Salida**

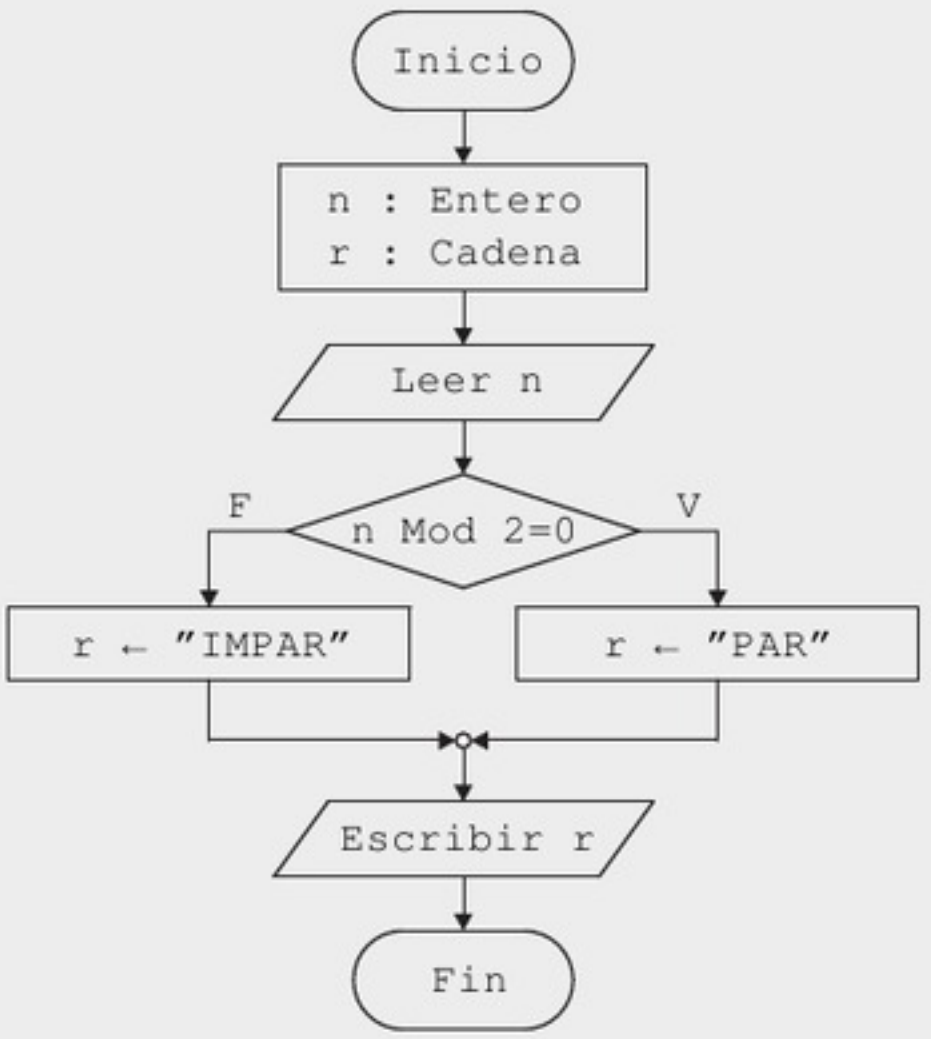
- Resultado (r)
  - PAR
  - IMPAR

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   Si n Mod 2 = 0 Entonces
       r ← "PAR"
   SiNo
       r ← "IMPAR"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
If n Mod 2 = 0 Then
     r = "PAR"
Else
     r = "IMPAR"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 16

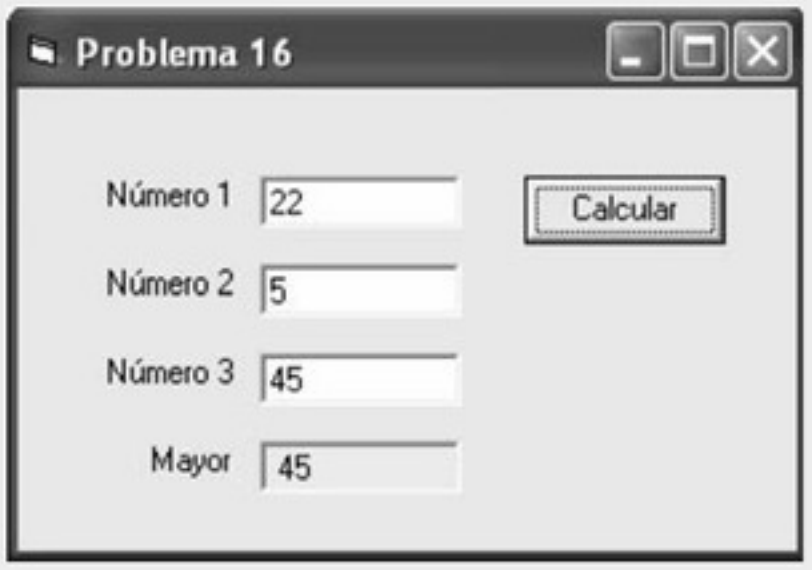
**Enunciado**: Dado tres números enteros, devolver el número mayor.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese tres números enteros n1, n2 y n3; luego, que el sistema verifique y devuelva el número mayor.

| Entrada | Salida |
|---|---|
| • Tres números (n1, n2, n3) | • Número mayor (m) |

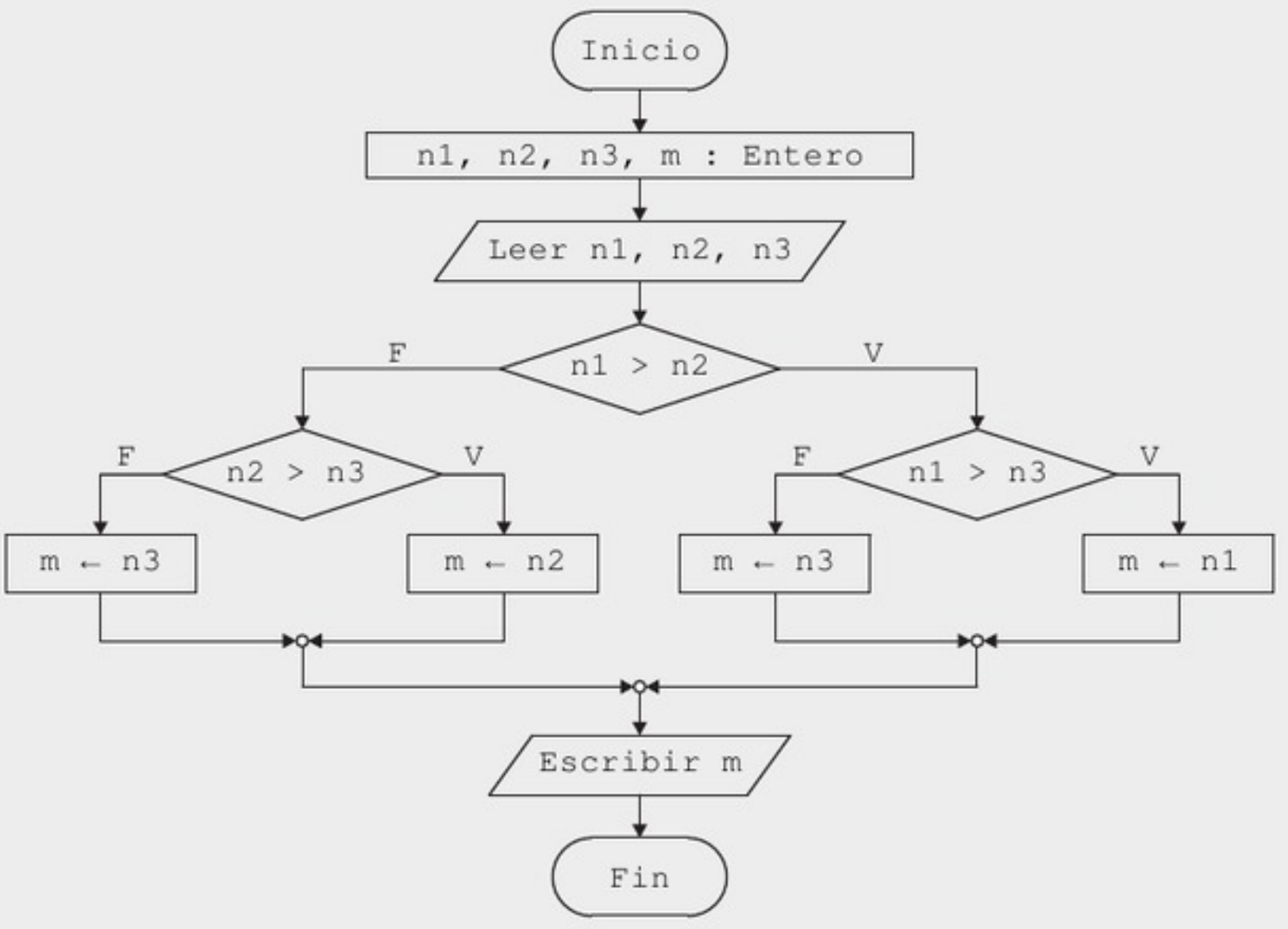
**Diseño**:

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**



**Pseudocódigo**

```
Inicio

   //Variables
   n1, n2, n3 : Entero

   //Entrada
   Leer n1, n2, n3

   //Proceso
   Si n1 > n2 Entonces
       Si n1 > n3 Entonces
             m ← n1
       SiNo
             m ← n3
       Fin Si
   SiNo
       Si n2 > n3 Entonces
             m ← n2
       SiNo
             m ← n3
       Fin Si
    Fin Si

   //Salida
   Escribir m

Fin
```

**Codificación:**

```
  'Variables
  Dim n1 As Integer
  Dim n2 As Integer
  Dim n3 As Integer
  Dim m As Integer

  'Entrada
  n1 = Val(Me.txtn1.Text)
  n2 = Val(Me.txtn2.Text)
  n3 = Val(Me.txtn3.Text)

  'Proceso
  If n1 > n2 Then
       If n1 > n3 Then
             m = n1
       Else
             m = n3
       End If
  Else
       If n2 > n3 Then
             m = n2
       Else
             m = n3
       End If
  End If

  'Salida
  Me.txtm.Text = Str(m)
```

## Problema n.° 17

**Enunciado**: Dado un número, devolver el doble si el número no es par; caso contrario, el triple.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero n; luego, que el sistema verifique y devuelva el doble o el triple del número.

**Expresión**

Si ~(n Mod 2 = 0) ⇒
    r = n * 2
SiNo
    r = n * 3

**Entrada**

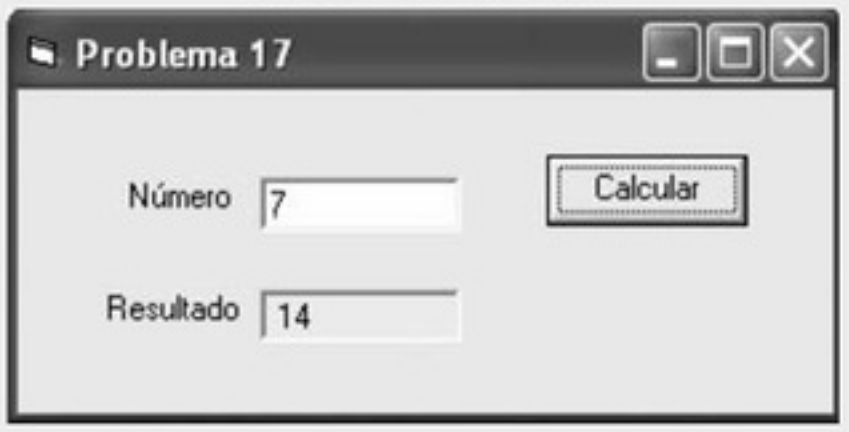
- Número entero (n)

**Salida**

- Resultado (r)

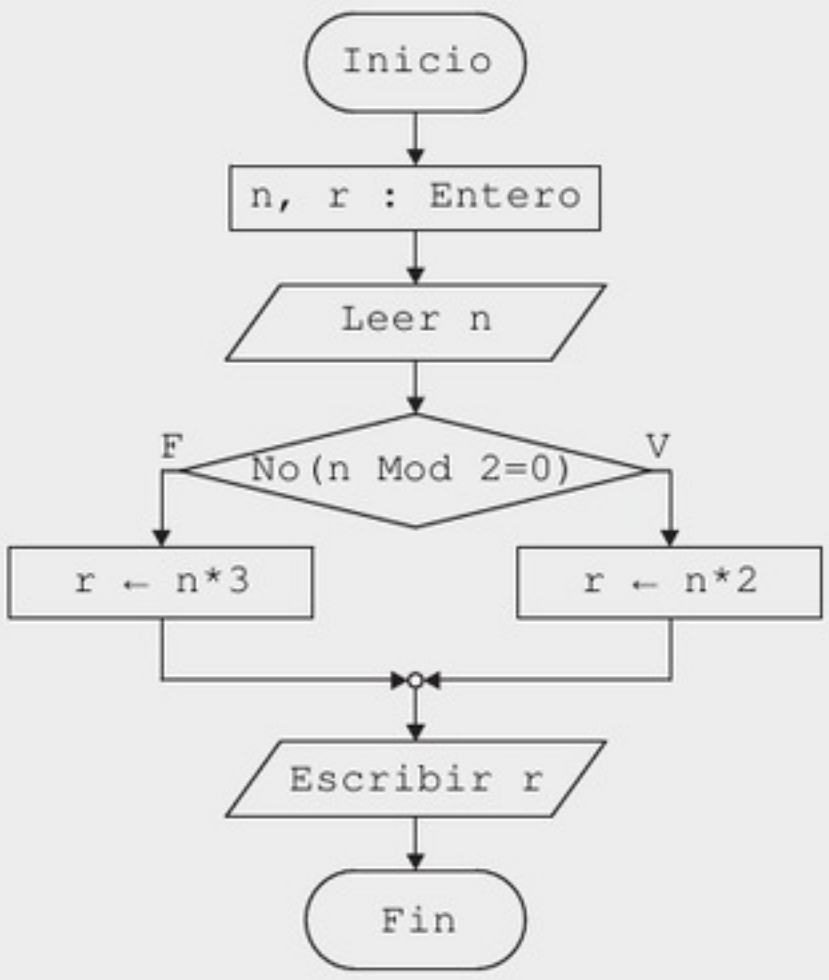
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, r : Entero

   //Entrada
   Leer n

   //Proceso
   Si No(n Mod 2=0) Entonces
       r ← n * 2
   SiNo
       r ← n * 3
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim r As Integer

'Entrada
n = Val(Me.txtn.Text)

'Proceso
If Not (n Mod 2 = 0) Then
    r = n * 2
Else
    r = n * 3
End If

'Salida
Me.txtr.Text = Str(r)
```

## Problema n.° 18

**Enunciado**: Dados 3 números, devolver los números en orden ascendente.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese tres números (n1, n2 y n3); luego, que el sistema verifique y devuelva los números ordenados en forma ascendente.

Primero se debe encontrar el número mayor, luego el número menor y, al final, el número intermedio; este resulta de sumar los tres números y luego restar el resultado de (mayor + menor).

**Entrada**

- Números (n1, n2, n3)

**Salida**

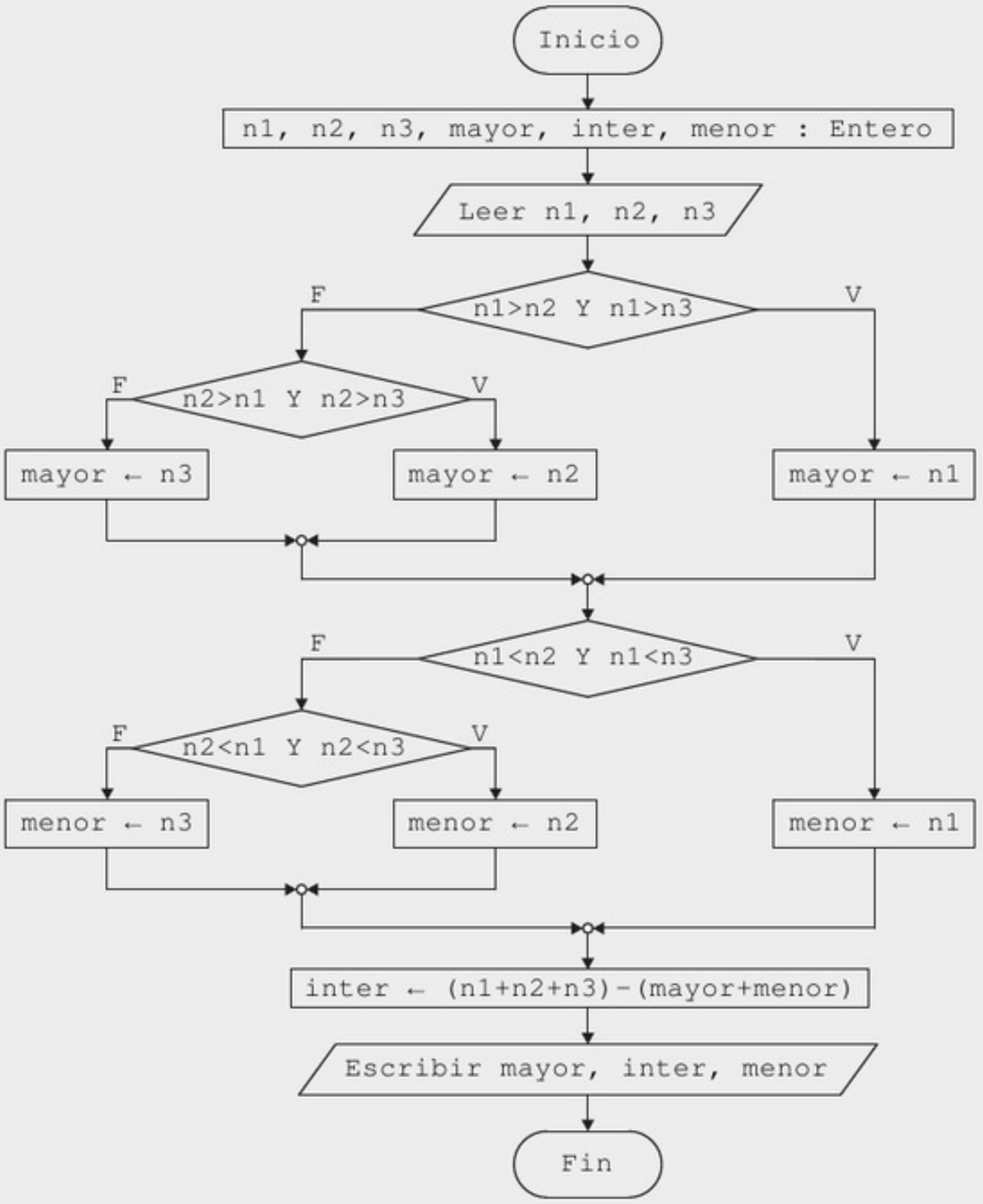
- Números ordenados (ma, int, me)

**Diseño:**

**Interfaz de usuario**

## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Variables
   n1, n2, n3, mayor, inter, menor : Entero

   //Entrada
   Leer n1, n2, n3

   //Proceso
   Si n1 > n2 Y n1 > n3 Entonces
      mayor ← n1
   SiNo
      Si n2 > n1 Y n2 > n3 Entonces
            mayor ← n2
      SiNo
            mayor ← n3
      Fin Si
   Fin Si
```

```
   Si n1 < n2 Y n1 < n3 Entonces
       menor ← n1
   SiNo
       Si n2 < n1 Y n2 < n3 Entonces
             menor ← n2
       SiNo
             menor ← n3
       Fin Si
   Fin Si

   inter ← (n1+n2+n3)-(mayor+menor)

   //Salida
   Escribir mayor, inter, menor

Fin
```

**Codificación:**

```
'Variables
Dim n1 As Integer
Dim n2 As Integer
Dim n3 As Integer
Dim mayor As Integer
Dim inter As Integer
Dim menor As Integer

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)
n3 = Val(Me.txtn3.Text)

'Proceso
If n1 > n2 And n1 > n3 Then
    mayor = n1
Else
    If n2 > n1 And n2 > n3 Then
          mayor = n2
    Else
          mayor = n3
    End If
End If

If n1 < n2 And n1 < n3 Then
    menor = n1
Else
          If n2 < n1 And n2 < n3 Then
          menor = n2
    Else
          menor = n3
    End If
End If

inter = (n1 + n2 + n3) - (mayor + menor)

'Salida
Me.txtMayor.Text = Str(mayor)
Me.txtInter.Text = Str(inter)
Me.txtMenor.Text = Str(menor)
```

## Problema n.° 19

**Enunciado**: Un restaurante ofrece un descuento del 10 % para consumos de hasta S/.100.00; y un descuento del 20 % para consumos mayores. En ambos casos se aplica un impuesto del 19 %. Determinar el monto del descuento, el impuesto y el importe a pagar.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el consumo; luego, que el sistema verifique y calcule el monto del descuento, el impuesto y el importe a pagar.

**Entrada**

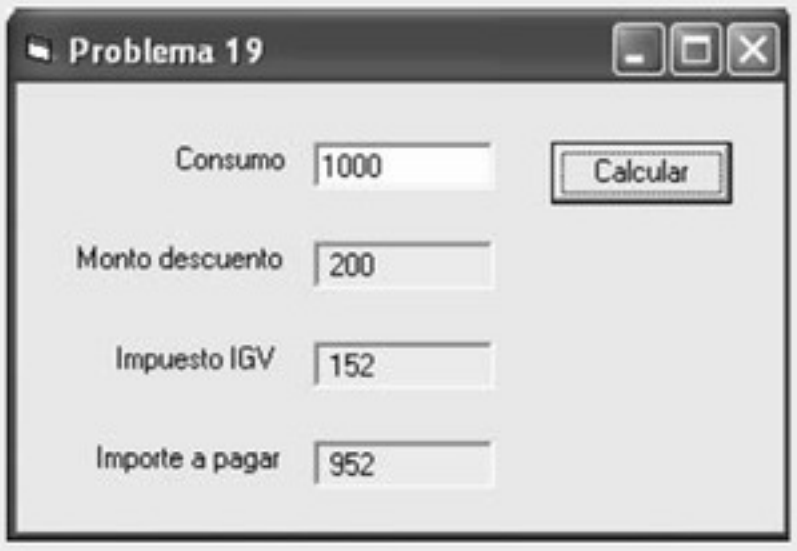
- Consumo (c)

**Salida**

- Monto del descuento (m_d)
- Impuesto (m_igv)
- Importe a pagar (p)

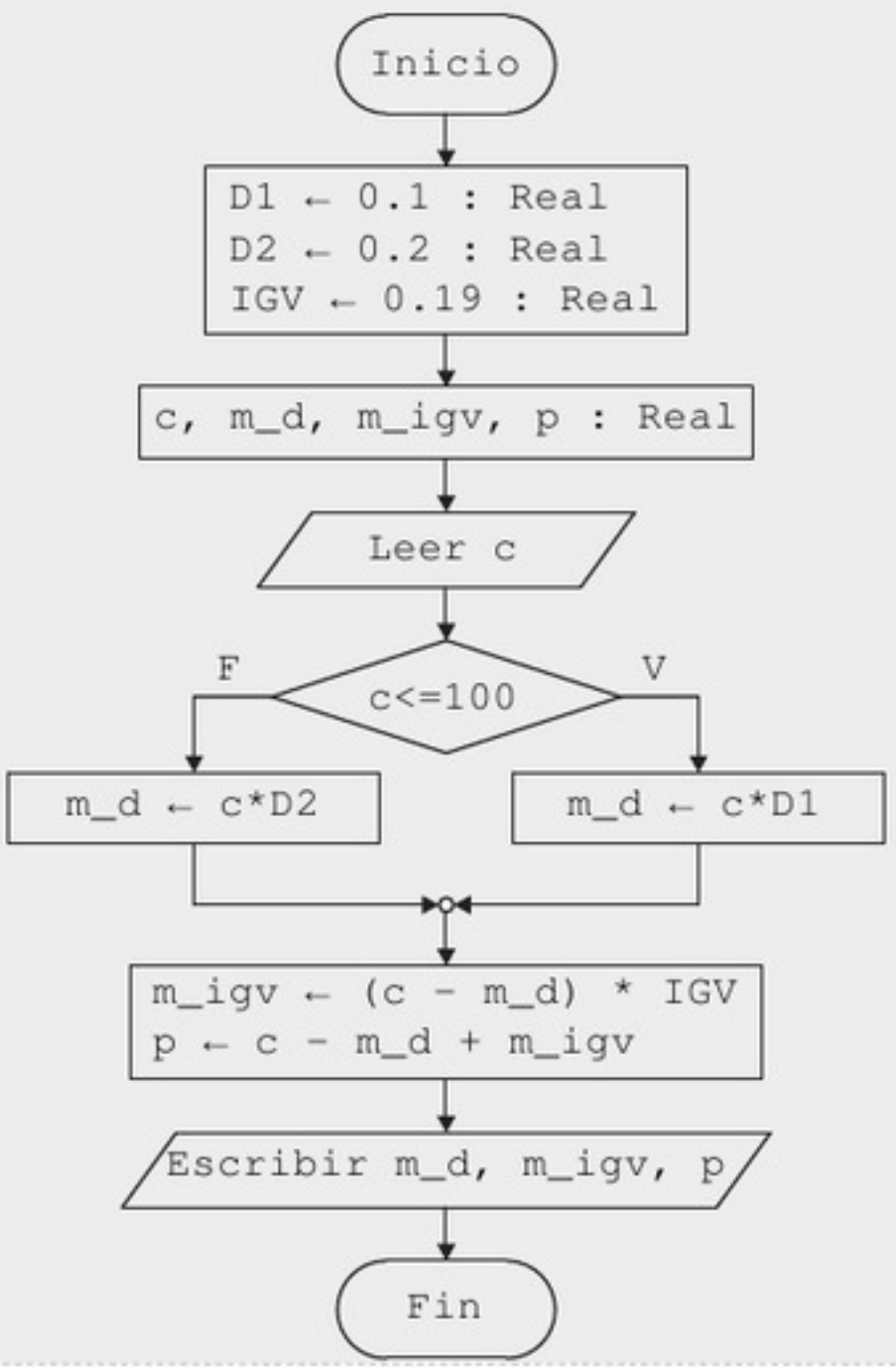
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Constantes
   D1 = 0.1 : Real
   D2 = 0.2 : Real
   IGV = 0.19 : Real

   //Variables
   c, m_d, m_igv, p : Real

   //Entrada
   Leer c

   //Proceso
   Si c <= 100 Entonces
       m_d ← c * D1
   SiNo
       m_d ← c * D2
   Fin Si

   m_igv ← (c - m_d) * IGV
   p ← c - m_d + m_igv

   //Salida
   Escribir m_d, m_igv, p

Fin
```

**Codificación:**

```
'Constante
Const D1 As Single = 0.1
Const D2 As Single = 0.2
Const IGV As Single = 0.19

'Variables
Dim c As Single
Dim m_d As Single
Dim m_igv As Single
Dim p As Single

'Entrada
c = Val(Me.txtc.Text)

'Proceso
If c <= 100 Then
    m_d = c * D1
Else
    m_d = c * D2
End If
m_igv = (c - m_d) * IGV
p = c - m_d + m_igv

'Salida
Me.txtm_d.Text = Str(m_d)
Me.txtm_igv.Text = Str(m_igv)
Me.txtp.Text = Str(p)
```

## Problema n.° 20

**Enunciado**: Debido a los excelentes resultados, el restaurante decide ampliar sus ofertas de acuerdo a la siguiente escala de consumo (ver tabla). Determinar el monto del descuento, el importe del impuesto y el importe a pagar.

| Consumo (S/.) | Descuento (%) |
|---|---|
| Hasta 100 | 10 |
| Mayor a 100 | 20 |
| Mayor a 200 | 30 |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el consumo; luego, el sistema verifique y calcule el monto del descuento, el impuesto y el importe a pagar.

**Entrada**

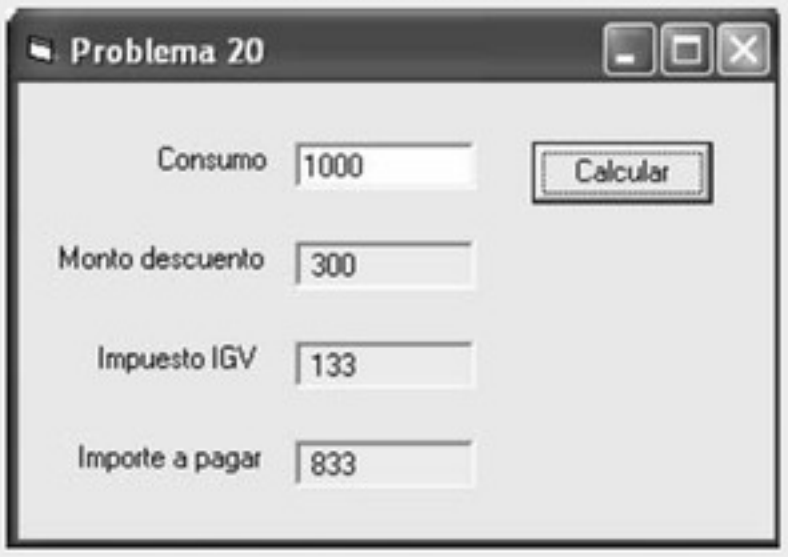
- Consumo (c)

**Salida**

- Monto del descuento (m_d)
- Impuesto (m_igv)
- Importe a pagar (p)

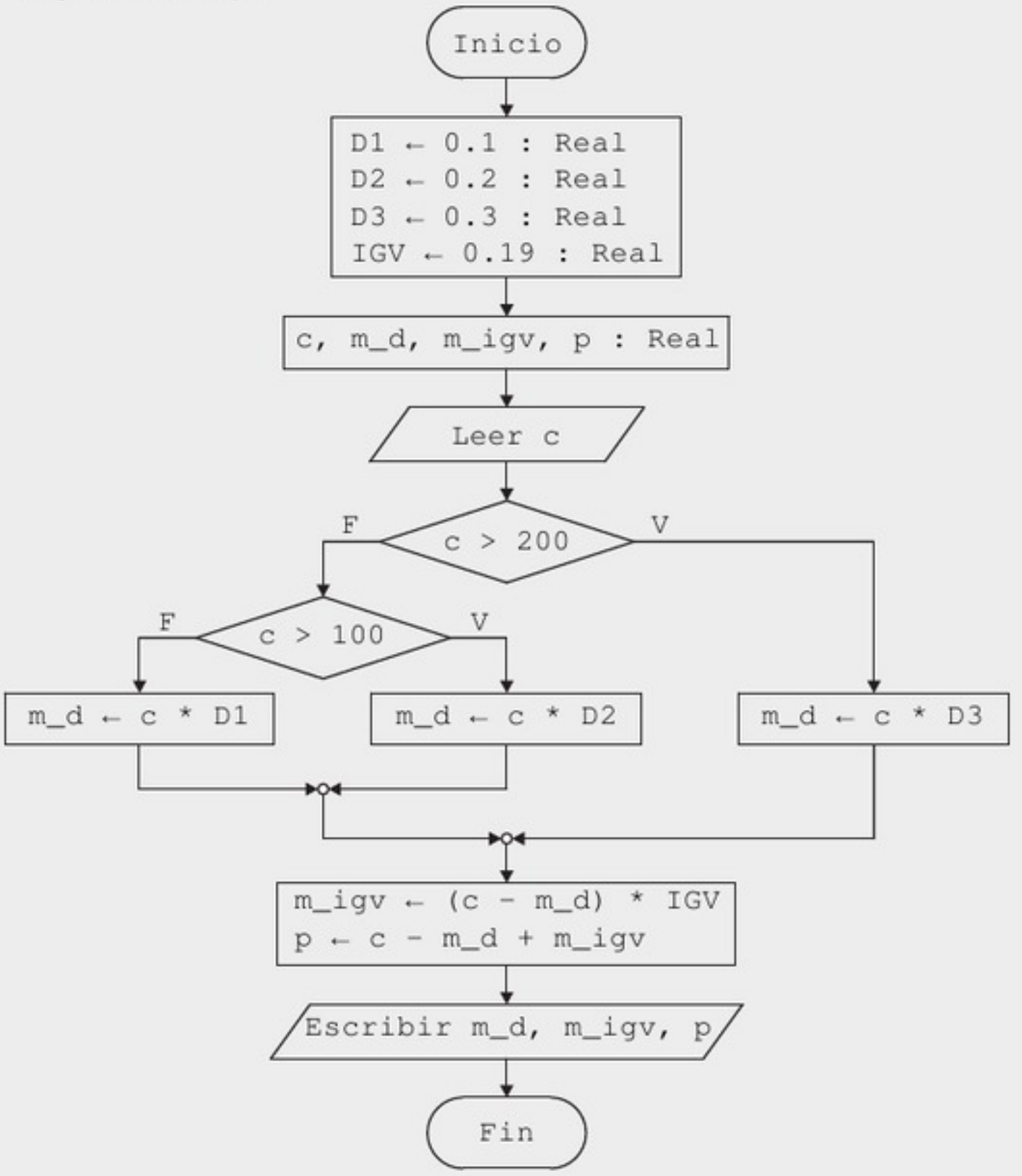
**Diseño:**

**Interfaz de usuario**




## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Constantes
   D1 = 0.1 : Real
   D2 = 0.2 : Real
   D3 = 0.3 : Real
   IGV = 0.19 : Real
```

```
//Variables
c, m_d, m_igv, p : Real

//Entrada
Leer c

//Proceso
Si c > 200 Entonces
    m_d ← c * D3
SiNo
    Si c > 100 Entonces
          m_d ← c * D2
    SiNo
          m_d ← c * D1
    Fin Si
Fin Si

m_igv ← (c - m_d) * IGV
p ← c - m_d + m_igv

//Salida
Escribir m_d, m_igv, p

Fin
```

**Codificación:**

```
'Constante
Const D1 As Single = 0.1
Const D2 As Single = 0.2
Const D3 As Single = 0.3
Const IGV As Single = 0.19

'Variables
Dim c As Single
Dim m_d As Single
Dim m_igv As Single
Dim p As Single

'Entrada
c = Val(Me.txtc.Text)

'Proceso
If c > 200 Then
     m_d = c * D3
Else
     If c > 100 Then
          m_d = c * D2
     Else
          m_d = c * D1
     End If
End If

m_igv = (c - m_d) * IGV
p = c - m_d + m_igv

'Salida
Me.txtm_d.Text = Str(m_d)
Me.txtm_igv.Text = Str(m_igv)
Me.txtp.Text = Str(p)
```

## Problema n.° 21

**Enunciado**: Al ingresar el valor de una temperatura, obtener el tipo de clima según la siguiente tabla.

| Temperatura | Tipo de clima |
|---|---|
| Temp. < 10 | Frío |
| Temp. entre 10 Y 20 | Nublado |
| Temp. entre 21 Y 30 | Calor |
| Temp. > 30 | Tropical |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese la temperatura; luego, que el sistema verifique y determine el clima.

**Entrada**

- Temperatura (t)

**Salida**

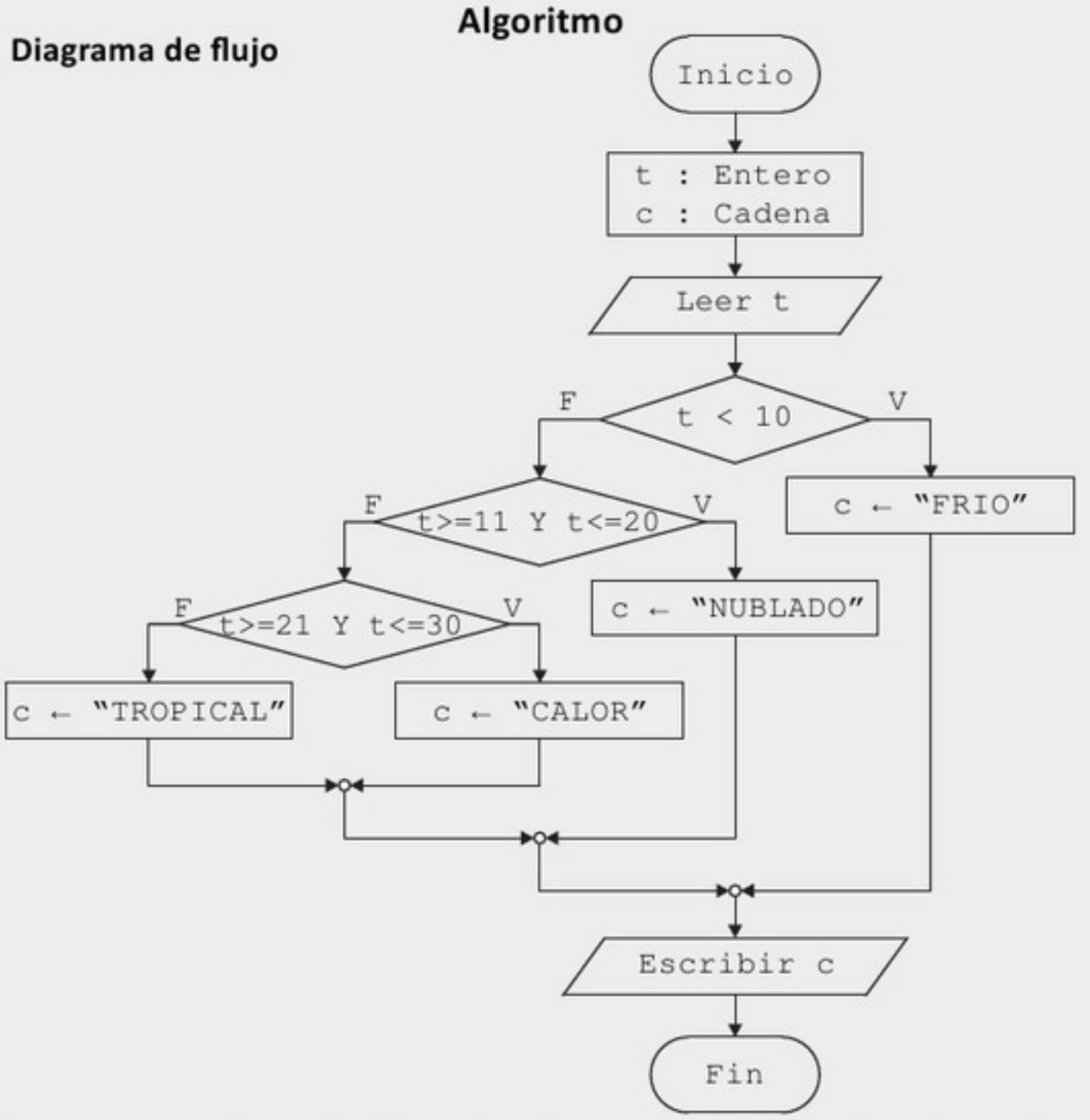
- Clima (c)

**Diseño:**

**Interfaz de usuario**


**Diagrama de flujo**

**Algoritmo**



**Pseudocódigo**

```
Inicio

      //Variables
      t : Entero
      c : Cadena

      //Entrada
      Leer t

      //Proceso
      Si t < 10 Entonces
            c ← "FRIO"
      SiNo
            Si t >= 11 Y t <=20 Entonces
                  c ← "NUBLADO"
            SiNo
                  Si t >= 21 Y t <=20 Entonces
                        c ← "CALOR"
                  SiNo
                        c ← "TROPICAL"
                  Fin Si
            Fin Si
      Fin Si

      //Salida
      Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim t As Integer
Dim c As String

'Entrada
t = Val(Me.txtt.Text)

'Proceso
If t < 10 Then
     c = "FRIO"
Else
     If t >= 10 And t <= 20 Then
          c = "NUBLADO"
     Else
          If t >= 21 And t <= 30 Then
               c = "CALOR"
          Else
               c = "TROPICAL"
          End If
     End If
End If

'Salida
Me.txtc.Text = c
```

## Problema n.° 22

**Enunciado**: Un negocio tiene dos tipos de cliente: cliente general (G) y cliente afiliado (A). También acepta dos formas de pago: al Contado (C) y en plazos (P). Nos piden crear un programa que al ingresar el monto de la compra se obtenga el **monto del descuento** o el **monto del recargo** y el **total a pagar,** según la siguiente tabla.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el monto de la compra, el tipo de cliente y la forma de pago; luego, que el sistema verifique y determine el monto de descuento o recargo y el total a pagar.

| Tipo | Contado (C) Descuento | Plazos (P) Recargo |
|---|---|---|
| Cliente general (G) | 15 % | 10 % |
| Cliente afiliado (A) | 20 % | 5 % |

**Entrada**

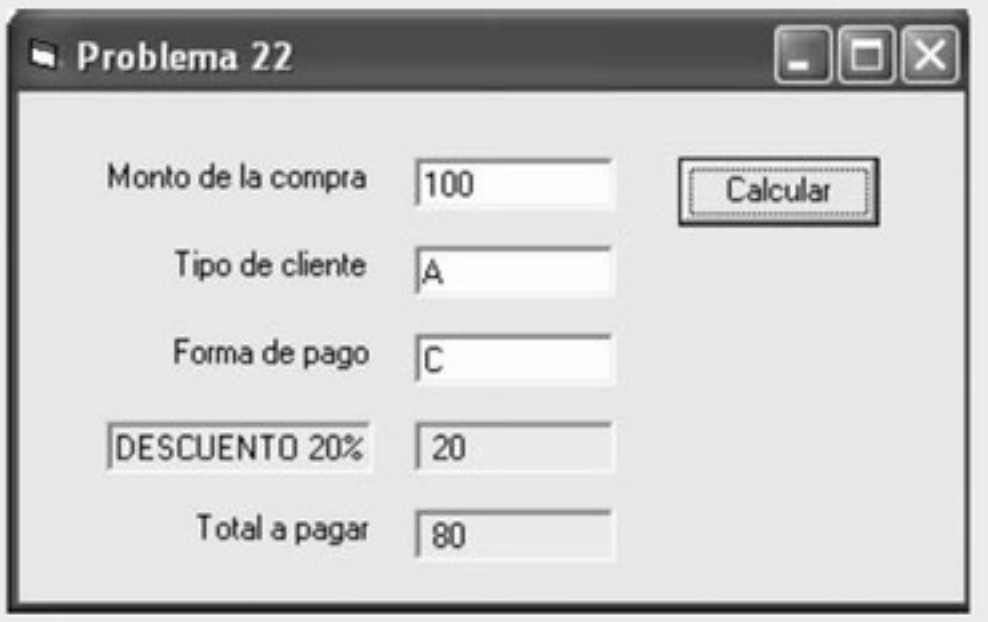
- Monto de la compra (mc)
- Tipo de cliente (tc)
- Forma de pago (fp)

**Salida**

- Monto de descuento o recargo (m)
- Total a pagar (tp)

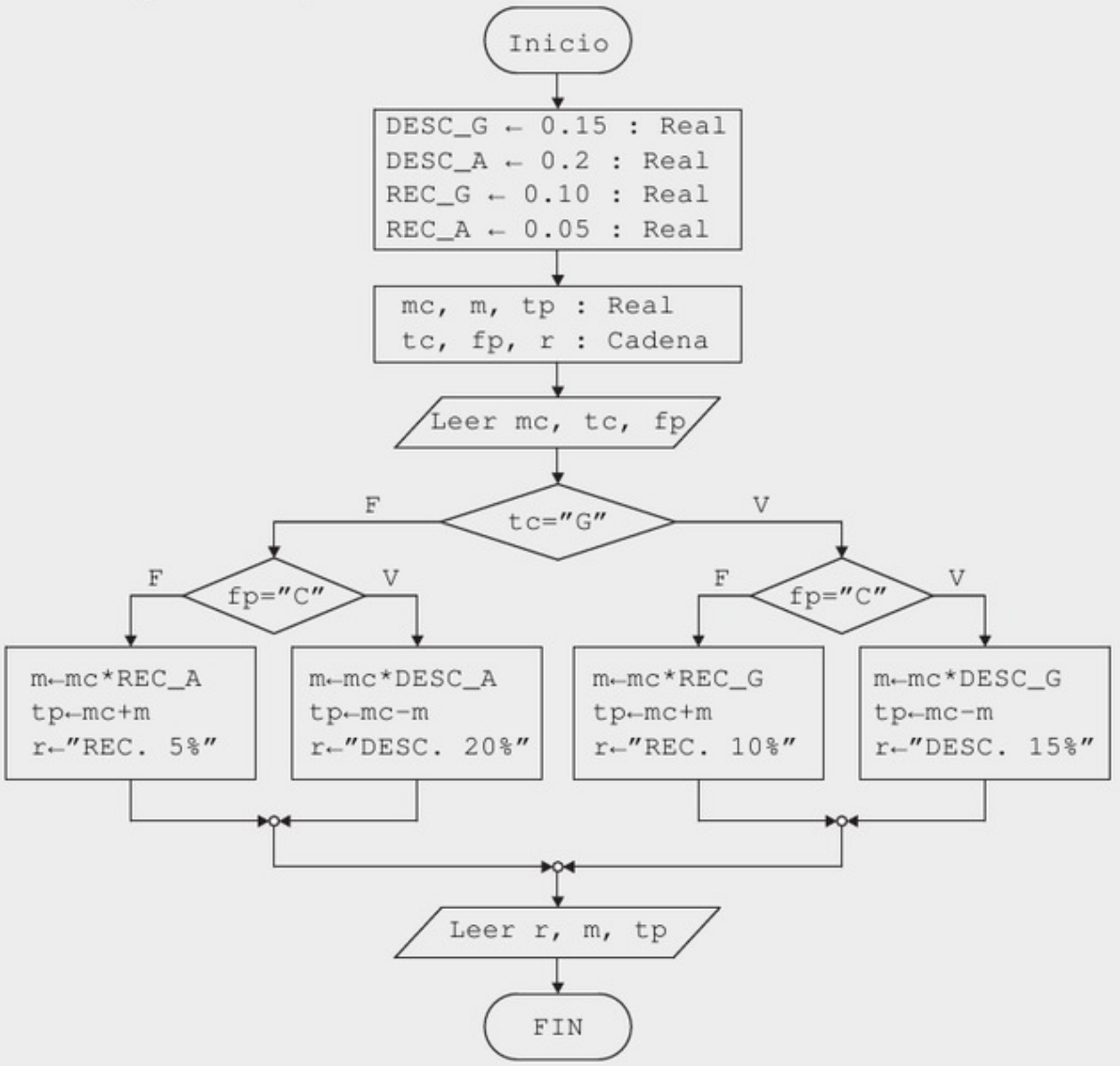
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

//Constantes
DESC_G = 0.15 : Real
DESC_A = 0.2 : Real
REC_G = 0.10 : Real
REC_A = 0.05 : Real

//Variables
mc, m, tp : Real
tc, fp, r : Cadena

//Entrada
Leer mc, tc, fp

//Proceso
Si tc = "G" Entonces
    Si fp = "C" Entonces
```

```
            m ← mc * DESC_G
            tp ← mc - m
            r ← "DESCUENTO 15%"
    SiNo
            m ← mc * REC_G
            tp ← mc + m
            r ← "RECARGA 10%"
    Fin Si
SiNo
    Si fp = "C" Entonces
            m ← mc * DESC_A
            tp ← mc - m
            r ← "DESCUENTO 20%"
    SiNo
            m ← mc * REC_A
            tp ← mc + m
            r ← "RECARGA 5%"
        Fin Si
   Fin Si

   //Salida
   Escribir r, m, tp

Fin
```

**Codificación:**

```
'Constantes
Const DESC_G As Single = 0.15
Const DESC_A As Single = 0.2
Const REC_G As Single = 0.1
Const REC_A As Single = 0.05

'Variables
Dim mc As Single
Dim tc As String
Dim fp As String
Dim r As String
Dim m As Single
Dim tp As Single

'Entrada
mc = Val(Me.txtmc.Text)
tc = Me.txttc.Text
fp = Me.txtfp.Text

'Proceso
If tc = "G" Then
    If fp = "C" Then
        m = mc * DESC_G
        tp = mc - m
```

```
            r = "DESCUENTO 15%"
      Else
            m = mc * REC_G
            tp = mc + m
            r = "RECARGO 10%"
      End If
Else
      If fp = "C" Then
            m = mc * DESC_A
            tp = mc - m
            r = "DESCUENTO 20%"
      Else
            m = mc * REC_A
            tp = mc + m
            r = "RECARGO 5%"
      End If
End If

'Salida
Me.txtr.Text = r
Me.txtm.Text = Str(m)
Me.txttp.Text = Str(tp)
```

## Problema n.° 23

**Enunciado**: Elabore un algoritmo que resuelva una ecuación de primer grado.

$$ax + b = 0 \qquad x = \frac{-b}{a}$$

Considerar si **a** es diferente a **0**, no es una ecuación de primer grado.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el valor de a y b; luego, que el sistema verifique y determine el valor de x.

**Entrada**

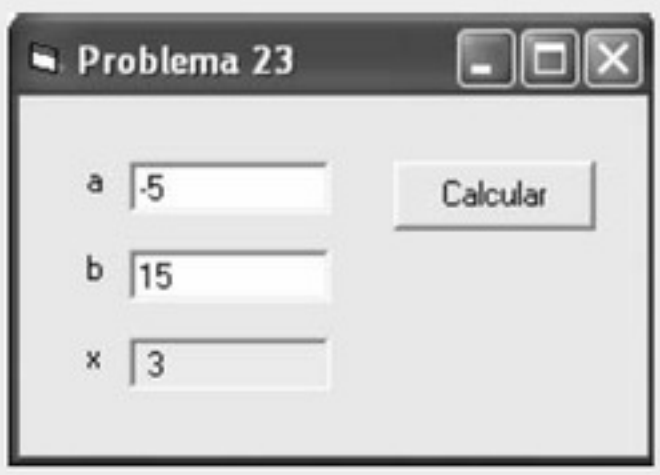
- Coeficiente a (a)
- Término independiente b (b)

**Salida**

- Raíz x (x)

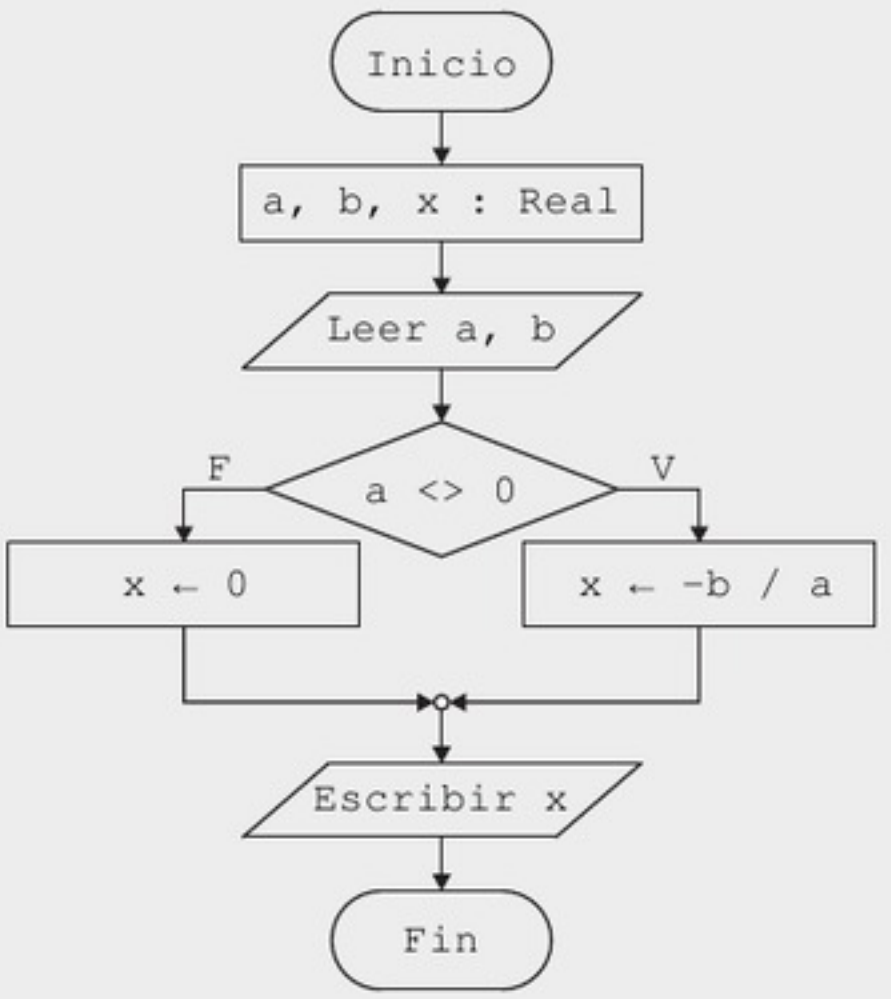
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   a, b, x : Real

   //Entrada
   Leer a, b

   //Proceso
   Si a <> 0 Entonces
       x ← -b / a
   SiNo
       x ← 0
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim a As Single
Dim b As Single
Dim x As Single

'Entrada
a = Val(Me.txta.Text)
b = Val(Me.txtb.Text)

'Proceso
If a <> 0 Then
     x = -b / a
Else
     x = 0
End If

'Salida
Me.txtx.Text = Str(x)
```

## Problema n.° 24

**Enunciado**: Elabore un algoritmo que obtenga las raíces reales de una ecuación de segundo grado.

$$ax^2 + bx + c = 0$$

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- Considerar que **a ≠ 0, para poder dividir.**
- Considerar **$b^2 - 4ac \neq 0$, para obtener la raíz cuadrada.**

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el valor de a, b y c; luego, que el sistema verifique y determine el valor de x1 y x2.

**Entrada**

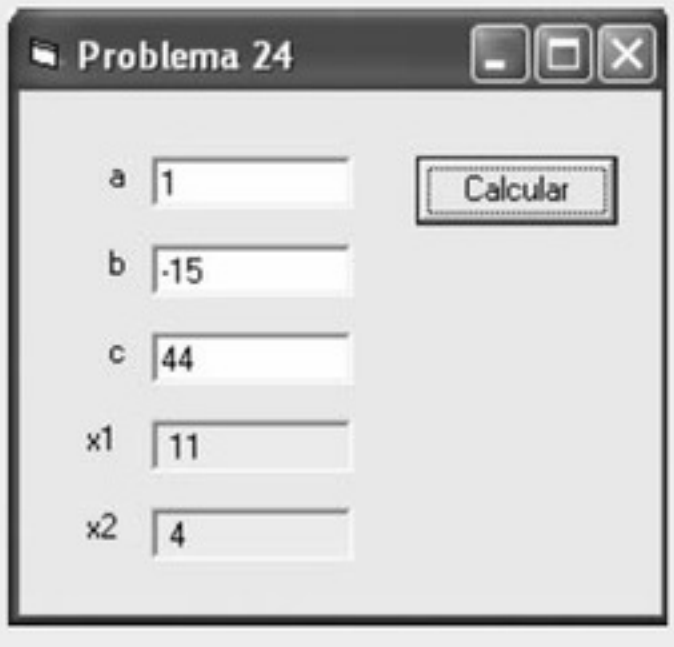
- Coeficiente a (a)
- Coeficiente b (b)
- Término independiente c (c)

**Salida**

- Primera raíz x (x1)
- Segunda raíz x (x2)

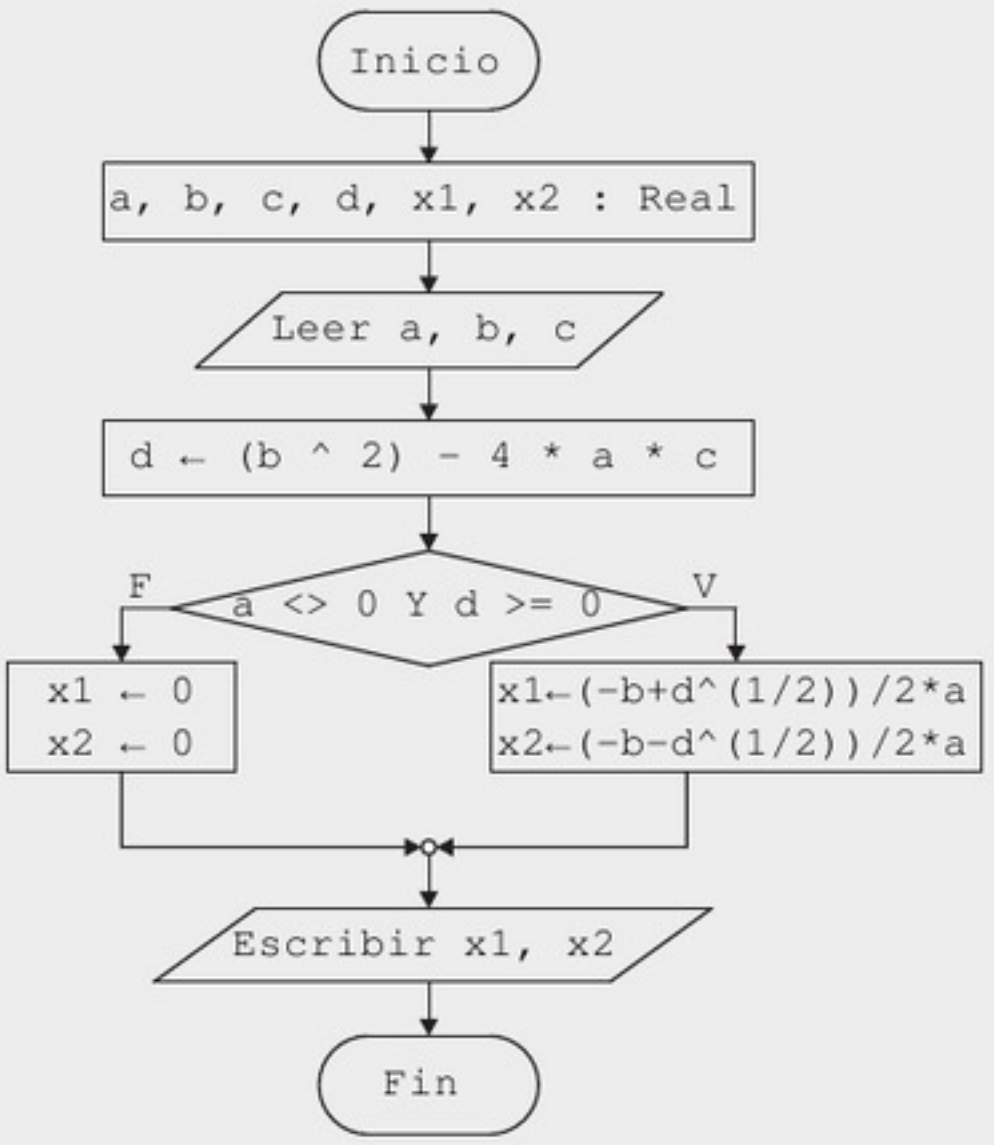
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   a, b, c, x1, x2 : Real

   //Entrada
   Leer a, b, c

   //Proceso
   d ← (b ^ 2) - 4 * a * c

   Si a <> 0 Y d >= 0  Entonces
     x1 ← (-b + d ^ (1 / 2)) / 2 * a
     x2 ← (-b - d ^ (1 / 2)) / 2 * a
   SiNo
     x1 ← 0
     x2 ← 0
   Fin Si

   //Salida
   Escribir x1, x2

Fin
```

**Codificación:**

```
'Variables
Dim a As Single
Dim b As Single
Dim c As Single
Dim d As Single
Dim x1 As Single
Dim x2 As Single

'Entrada
a = Val(Me.txta.Text)
b = Val(Me.txtb.Text)
c = Val(Me.txtc.Text)

'Proceso
d = (b ^ 2) - 4 * a * c
If a <> 0 And d >= 0 Then
     x1 = (-b + d ^ (1 / 2)) / 2 * a
     x2 = (-b - d ^ (1 / 2)) / 2 * a
Else
     x1 = 0
     x2 = 0
End If

'Salida
Me.txtx1.Text = Str(x1)
Me.txtx2.Text = Str(x2)
```

## Problema n.° 25

**Enunciado**: Dadas la hora, minuto y segundo, encuentre la hora del siguiente segundo.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese la hora, minuto y segundo; luego que el sistema verifique y determine la hora, minuto y segundo del siguiente segundo.

**Entrada**

- Hora (h)
- Minuto (m)
- Segundo (s)

**Salida**

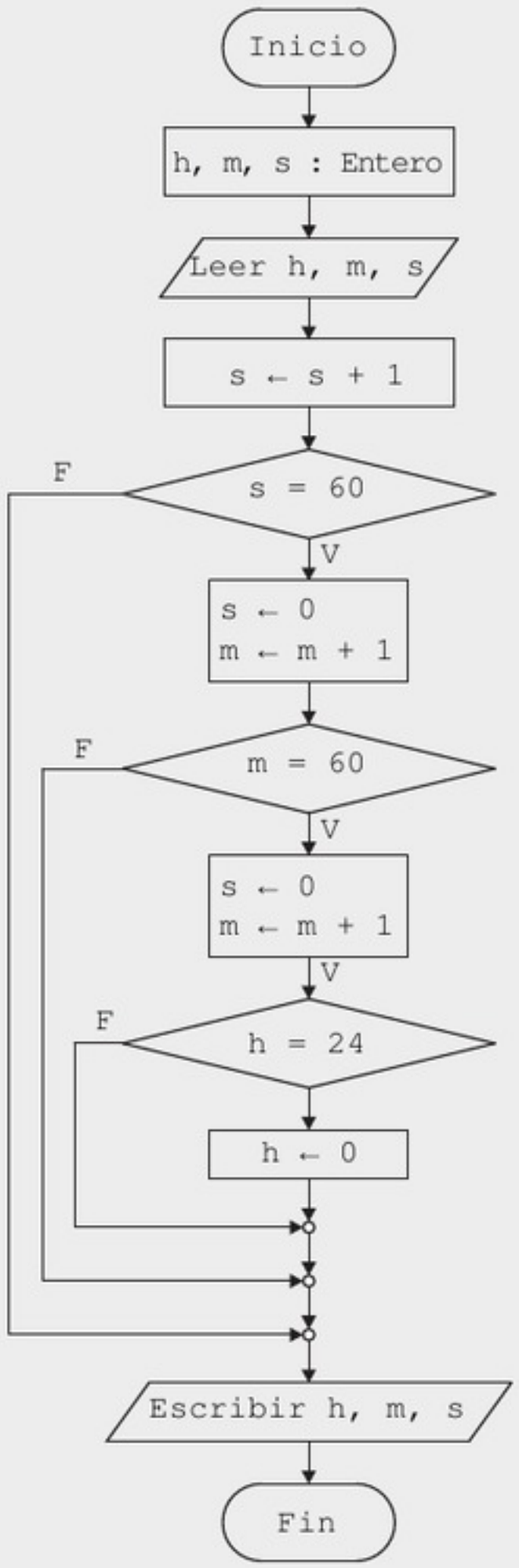
- Hora (h)
- Minuto (m)
- Segundo (s)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   h, m, s : Entero

   //Entrada
   Leer h, m, s

   //Proceso
   s ← s + 1
   Si  s = 60 Entonces
       s ← 0
       m ← m + 1
       Si     m = 60 Entonces
              m ← 0
              h ← h + 1
              Si     h = 60 Entonces
                     h ← 0
              Fin Si
       Fin Si
   Fin Si

   //Salida
   Escribir h, m, s

Fin
```

**Codificación:**

```
'Variables
Dim h As Integer
Dim m As Integer
Dim s As Integer

'Entrada
h = Val(Me.txth1.Text)
m = Val(Me.txtm1.Text)
s = Val(Me.txts1.Text)

'Proceso
s = s + 1
If s = 60 Then
     s = 0
     m = m + 1
     If m = 60 Then
          m = 0
          h = h + 1
          If h = 24 Then
               h = 0
          End If
     End If
End If

'Salida
Me.txth2.Text = Str(h)
Me.txtm2.Text = Str(m)
Me.txts2.Text = Str(s)
```

## 3.4 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 11

**Enunciado**: Dada la edad de una persona, determinar si es mayor o menor de edad; considere que son mayores de edad aquellas personas que tienes de 18 años a más.

### Propuesto n.° 12

**Enunciado**: Si tenemos dos números enteros, devolver el número menor.

### Propuesto n.° 13

**Enunciado**: Si tenemos dos números, determinar si son iguales o son diferentes.

### Propuesto n.° 14

**Enunciado**: Dado un número entero, devolver el doble del número si este es positivo; el triple, si es negativo; y cero si el número es neutro.

### Propuesto n.° 15

**Enunciado**: Crear un programa que al ingresar tres números enteros, los devuelva ordenados en forma ascendente y en forma descendente.

### Propuesto n.° 16

**Enunciado**: Después de ingresar 4 notas, obtener el promedio de la tres mejores y mostrar el mensaje «Aprobado», si el promedio es mayor o igual a 11; caso contrario, mostrar «Desaprobado».

### Propuesto n.° 17

**Enunciado**: Dados los siguientes datos de entrada: «saldo anterior», tipo de movimiento «R» (retiro) o «D» (depósito) y «monto de la transacción», obtener como dato de salida el saldo actual.

### Propuesto n.° 18

**Enunciado**: Dados 2 números enteros a y b, determinar cuál es mayor con respecto al otro.

**a** es mayor que **b**

**b** es mayor que **a**

**a** es igual a **b**

## Propuesto n.° 19

**Enunciado:** Dado que se tienen tres longitudes, diga si forman un triángulo.

**Teorema:** En todo triángulo, cada lado es menor que la suma de los otros dos, pero mayor que su diferencia.

## Propuesto n.° 20

**Enunciado:** Dado que tenemos tres longitudes; si forman un triángulo, devolver el tipo de triángulo según sus lados.

**T. equilátero:** Sus 3 lados son iguales.

**T. isósceles:** 2 lados iguales.

**T. escaleno:** 3 lados diferentes.

# Capítulo 4

# Estructura selectiva múltiple

## 4.1 Introducción

Sabemos que en la actualidad existen muchos sistemas financieros que ofrecen préstamos con condiciones diferentes; usted, al solicitar un préstamo, tiene que evaluar diversas alternativas y decidirse por una de ellas.

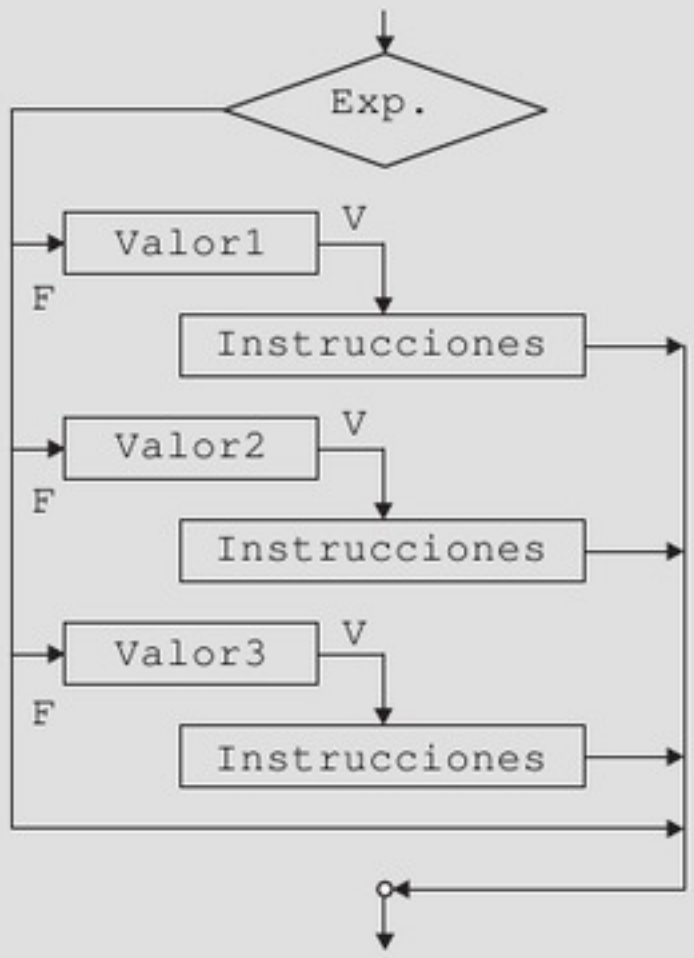
En los lenguajes de programación se cuenta con una implementación similar, la cual recibe el nombre de estructura selectiva múltiple, que permite evaluar varias alternativas y realizar el proceso para comprobar si cumple o no con la condición elegida.

Muchas veces, para solucionar este tipo de problemas, se utilizan estructuras selectivas dobles anidadas (en cascada), dando una solución muy complicada y confusa para analizar; es recomendable que cuando se tenga que evaluar varias alternativas se utilice estructuras selectiva múltiple porque es más legible, eficiente y fácil de interpretar.

## 4.2 Estructura selectiva múltiple

Permite comparar un valor con diversas alternativas; si la comparación tiene éxito, se ejecuta el grupo de instrucción que contenga la alternativa seleccionada y, luego, sale de la estructura.

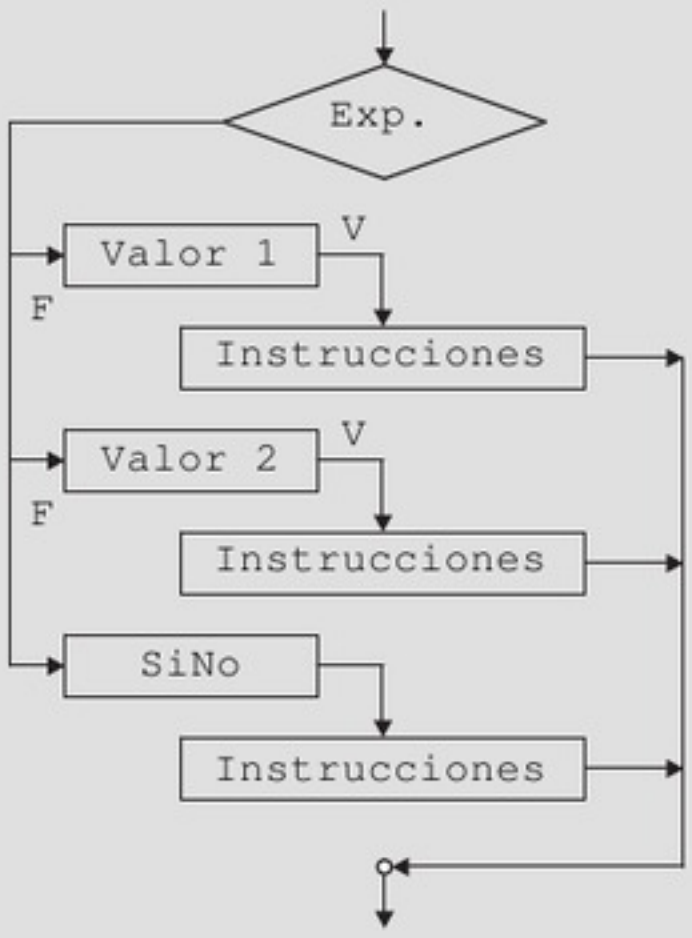
Muchas se pueden implementar, en forma opcional, una alternativa por defecto; es decir, si al comparar con todas las alternativas propuestas no se tiene éxito con ninguna, entonces se ejecuta la alternativa por defecto.




**En Caso que** <Exp.> **Sea**
**Caso** Valor1
<Instrucciones>
**Caso** Valor2
<Instrucciones>
**Caso** Valor3
<Instrucciones>
**Fin Caso**

**Sintaxis 1 Visual Basic**

```
Select Case <Exp. >
    Case Valor1
        <Instrucciones>
    Case Valor2
        <Instrucciones>
    Case Valor3
        <Instrucciones>
End Select
```



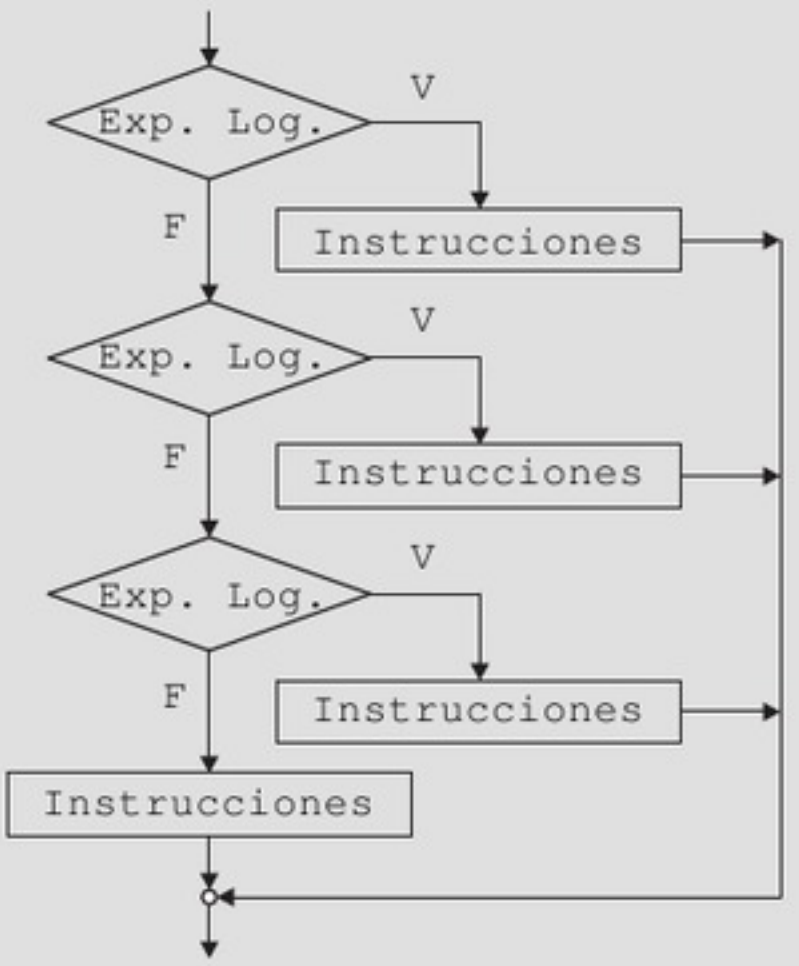

```
En Caso que <Exp.> Sea
    Caso Valor1
        <Instrucciones>
    Caso Valor2
        <Instrucciones>
    SiNo
        <Instrucciones>
Fin Caso
```

**Sintaxis 2 Visual Basic**

```
Select Case <Exp.>
    Case Valor1
        <Instrucciones>
    Case Valor2
        <Instrucciones>
    Case Else
        <Instrucciones>
End Select
```

### 4.2.1 Estructura selectiva múltiple usando rangos

La estructura selectiva múltiple permite comparar un valor (igualdad), pero cuando se requiere manejar rangos (>= Y <=) se puede usar una estructura selectiva múltiple similar a la estructura selectiva doble anidada.




**Si** <Exp.Log.> **Entonces**

<Instrucciones>

**SiNoSi** <Exp.Log.> **Entonces**

<Instrucciones>

**SiNoSi** <Exp.Log.> **Entonces**

<Instrucciones>

**SiNo**

<Instrucciones>

**Fin Si**

**Sintaxis Visual Basic**

```
If <Exp. Log.> Then
        <Instrucciones>
ElseIf <Exp. Log.> Then
        <Instrucciones>
ElseIf <Exp. Log.> Then
        <Instrucciones>
Else
        <Instrucciones>
End If
```

En el caso de Visual Basic, las sentencias múltiples usando **Select Case** permiten manejar rangos.

**Sintaxis 3 Visual Basic**

```
Select Case <Exp.>
    Case Valor1 To Valor2
        <Instrucciones>
    Case Valor3 To Valor4
        <Instrucciones>
    Case Else
        <Instrucciones>
End Select
```

## Problema n.° 26

**Enunciado**: Al ingresar un número entre 1 y 4 devolver la estación del año de acuerdo a la siguiente tabla:

| Número | Estación |
|---|---|
| 1 | Verano |
| 2 | Otoño |
| 3 | Invierno |
| 4 | Primavera |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero; luego, que el sistema realice el proceso para devolver la estación.

**Entrada**

- Número (n)

**Salida**

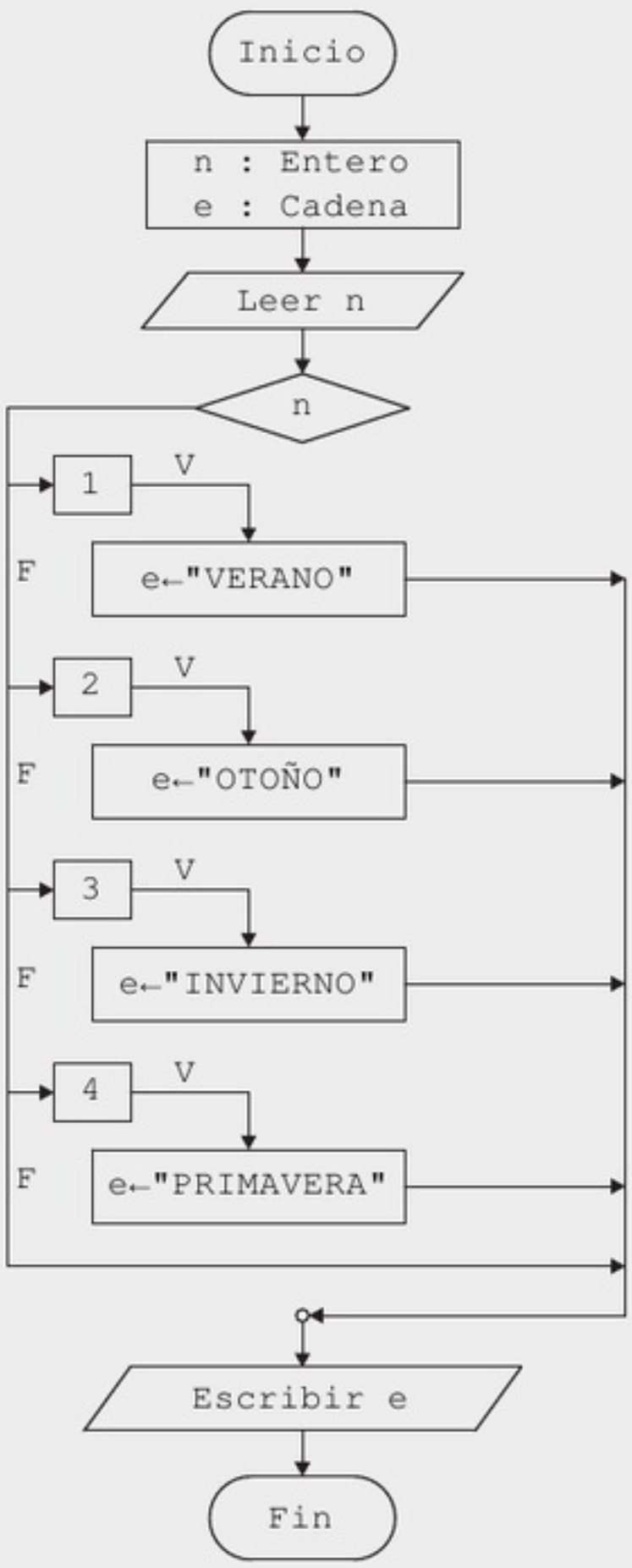
- Estación (e)

**Diseño:**

**Interfaz de usuario**


## Algoritmo

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n : Entero
   e : Cadena

   //Entrada
   Leer n

   //Proceso
   En Caso que n Sea
       Caso 1
          e ← "VERANO"
       Caso 2
          e ← "INVIERNO"
       Caso 3
          e ← "OTOÑO"
       Caso 4
          e ← "PRIMAVERA"
   Fin Caso

   //Salida
   Escribir e

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim e As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Select Case n
    Case 1
        e = "VERANO"
    Case 2
        e = "OTOÑO"
    Case 3
        e = "INVIERNO"
    Case 4
        e = "PRIMAVERA"
End Select

'Salida
Me.txte.Text = e
```

## Problema n.° 27

**Enunciado**: Dado un número entero de un dígito (0 al 9), devolver el número en letras.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero; luego, que el sistema verifique y devuelva el número en letras.

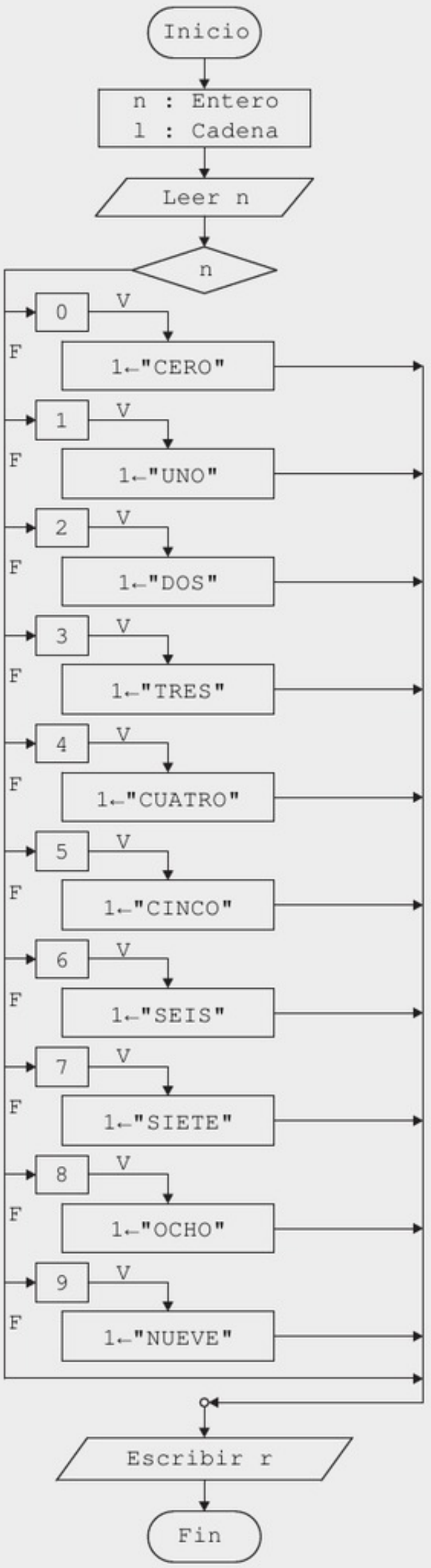
| Entrada | Salida |
|---|---|
| • Número (n) | • Resultado (r) |

**Diseño:**

**Interfaz de usuario**

## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Variables
   n : Entero
   l : Cadena

   //Entrada
   Leer n

   //Proceso
   En Caso que n Sea
       Caso 0
         l ← "CERO"
       Caso 1
         l ← "UNO"
       Caso 2
         l ← "DOS"
       Caso 3
         l ← "TRES"
       Caso 4
         l ← "CUATRO"
       Caso 5
         l ← "CINCO"
       Caso 6
         l ← "SEIS"
       Caso 7
         l ← "SIETE"
       Caso 8
         l ← "OCHO"
       Caso 7
         l ← "NUEVE"
   Fin Caso

   //Salida
   Escribir l

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim l As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Select Case n
     Case 0
          l = "CERO"
     Case 1
          l = "UNO"
     Case 2
          l = "DOS"
     Case 3
          l = "TRES"
     Case 4
          l = "CUATRO"
     Case 5
          l = "CINCO"
     Case 6
          l = "SEIS"
     Case 7
          l = "SIETE"
     Case 8
          l = "OCHO"
     Case 9
          l = "NUEVE"
End Select

'Salida
Me.txtl.Text = l
```

## Problema n.° 28

**Enunciado**: Dados dos números enteros y un operador (+, -, * y /), devolver la operación de los dos números según el operador ingresado. Considere que si el segundo número es cero y el operador es **/**, no es divisible con el primer número, entonces devolver como resultado 0.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un operador y dos números; luego, que el sistema verifique la operación y devuelva el resultado de la operación.

**Entrada**

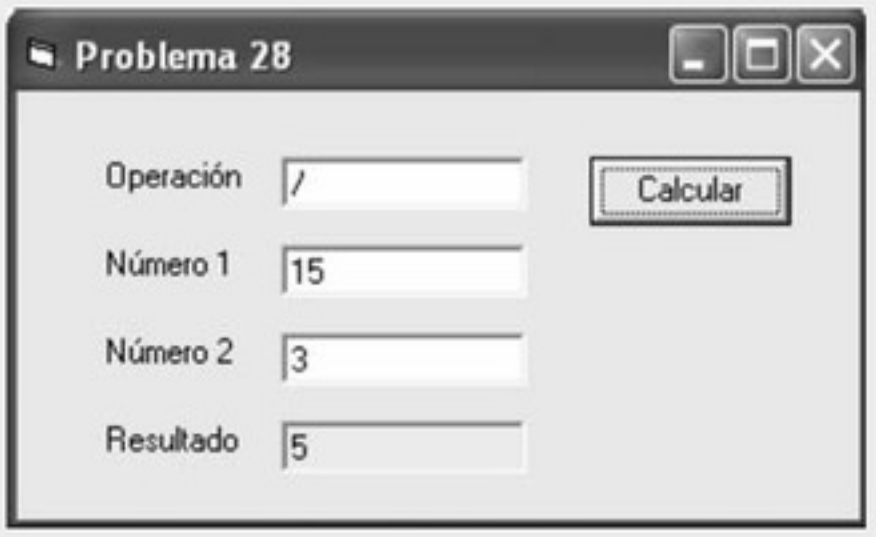
- Operador (op)
- Número (n1 y n2)

**Salida**

- Resultado (r)

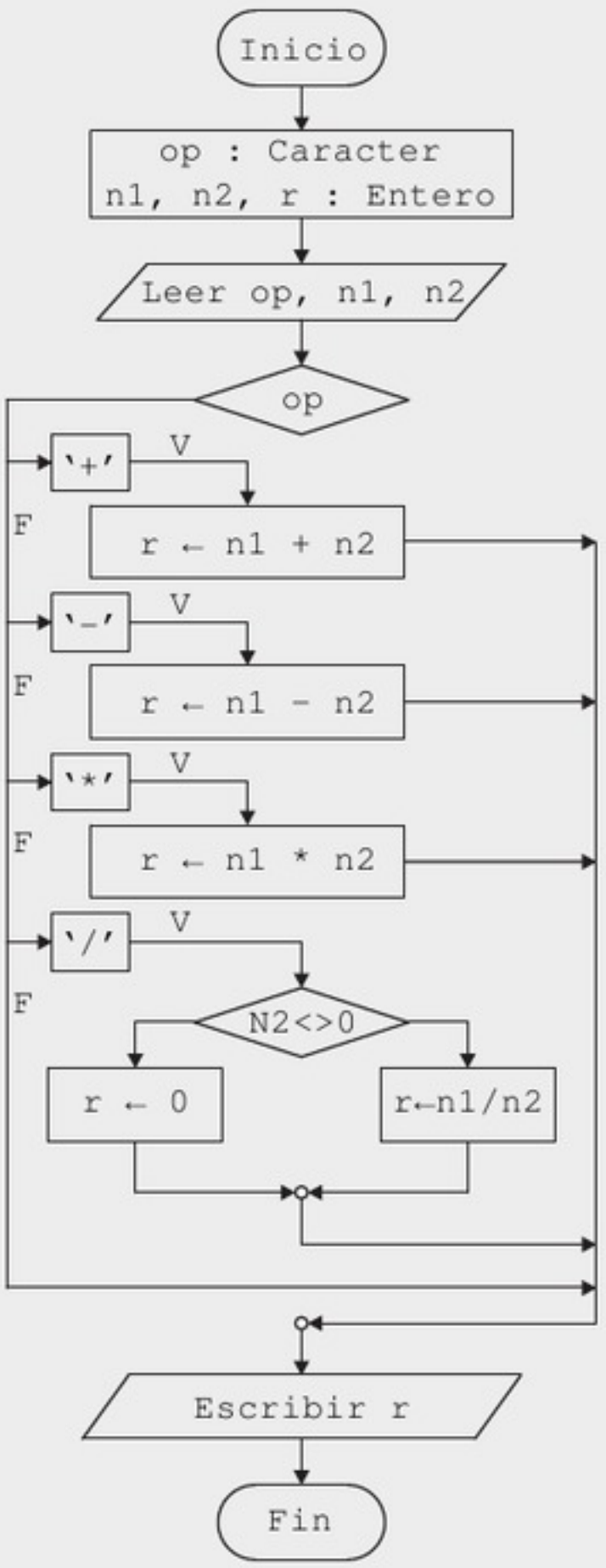
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   op : Caracter
   n1, n2, r : Entero

   //Entrada
   Leer op, n1, n2

   //Proceso
   En Caso que op Sea
       Caso '+'
          r ← n1 + n2
       Caso '-'
          r ← n1 - n2
       Caso '*'
          r ← n1 * n2
       Caso '/'
          Si n2 <> 0 Entonces
             r ← n1 + n2
          SiNo
             r ← 0
          Fin Sin
   Fin Caso

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim op As String
Dim n1 As Integer
Dim n2 As Integer
Dim r As Integer

'Entrada
op = Me.txtop.Text
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
Select Case op
     Case "+"
          r = n1 + n2
     Case "-"
          r = n1 - n2
     Case "*"
          r = n1 * n2
     Case "/"
          If n2 <> 0 Then
               r = n1 \ n2
          Else
               r = 0
          End If
End Select

'Salida
Me.txtr.Text = r
```

## Problema n.° 29

**Enunciado**: Dada una letra, determinar si es una vocal.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese una letra «l»; luego, que el sistema analice y determine si es una vocal.

**Entrada**

- Letra (l)

**Salida**

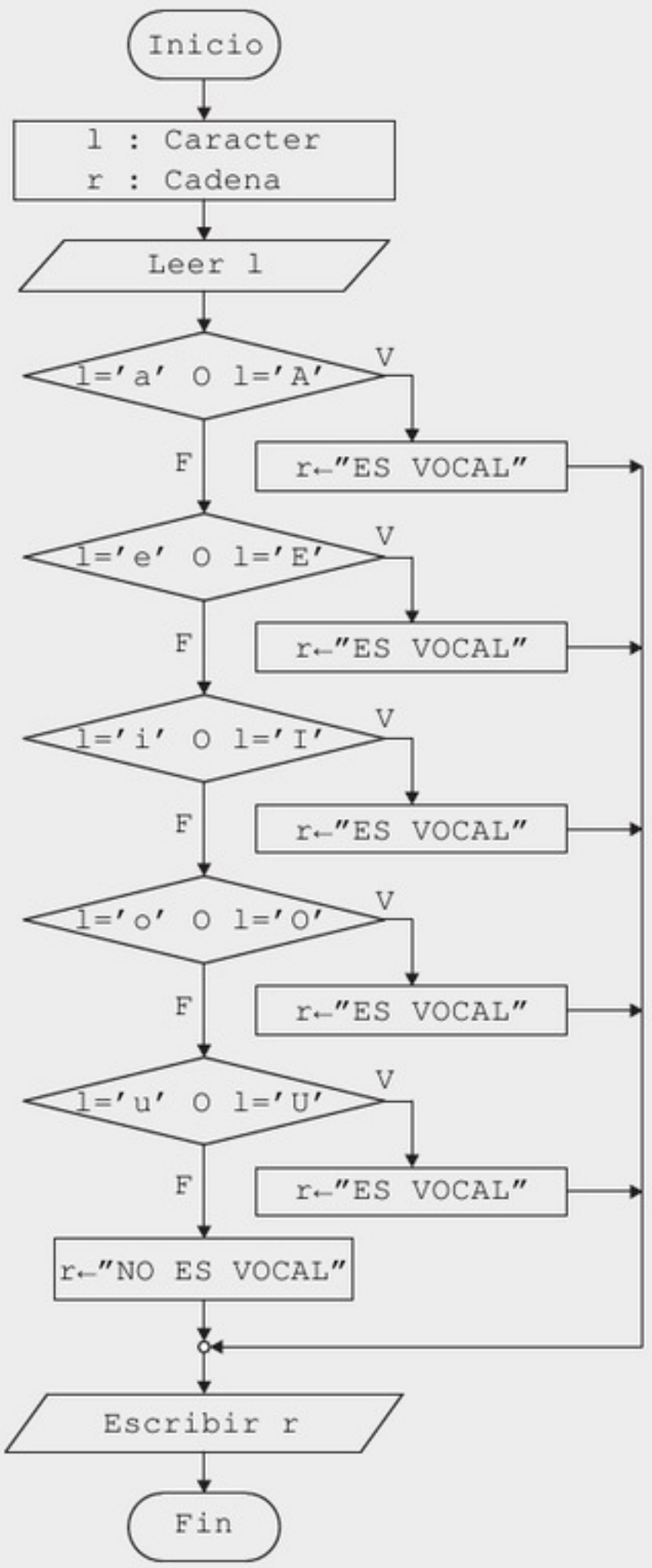
- Resultado (r)

**Diseño:**

**Interfaz de usuario**


## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Variables
   l : Caracter
   r : Cadena

   //Entrada
   Leer l

   //Proceso
   Si l = 'a' O l = 'A' Entonces
       r ← "ES VOCAL"
   SiNoSi l = 'e' O l = 'E' Entonces
       r ← "ES VOCAL"
   SiNoSi l = 'i' O l = 'I' Entonces
       r ← "ES VOCAL"
   SiNoSi l = 'o' O l = 'O' Entonces
       r ← "ES VOCAL"
   SiNoSi l = 'u' O l = 'U' Entonces
       r ← "ES VOCAL"
   SiNo
       r ← "NO ES VOCAL"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim l As String
Dim r As String

'Entrada
l = Me.txtl.Text

'Proceso
If l = "a" Or l = "A" Then
    r = "ES VOCAL"
ElseIf l = "e" Or l = "E" Then
    r = "ES VOCAL"
ElseIf l = "i" Or l = "I" Then
    r = "ES VOCAL"
ElseIf l = "o" Or l = "O" Then
    r = "ES VOCAL"
ElseIf l = "u" Or l = "U" Then
    r = "ES VOCAL"
Else
    r = "NO ES VOCAL"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 30

**Enunciado**: Al ingresar el número de un mes, devolver la estación del año de acuerdo a la siguiente tabla:

| Mes | Estación |
|---|---|
| 1, 2,3 | Verano |
| 4,5,6 | Otoño |
| 7,8,9 | Invierno |
| 10,11,12 | Primavera |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el número del mes; luego, que el sistema verifique y determine la estación.

**Entrada**

- Mes (m)

**Salida**

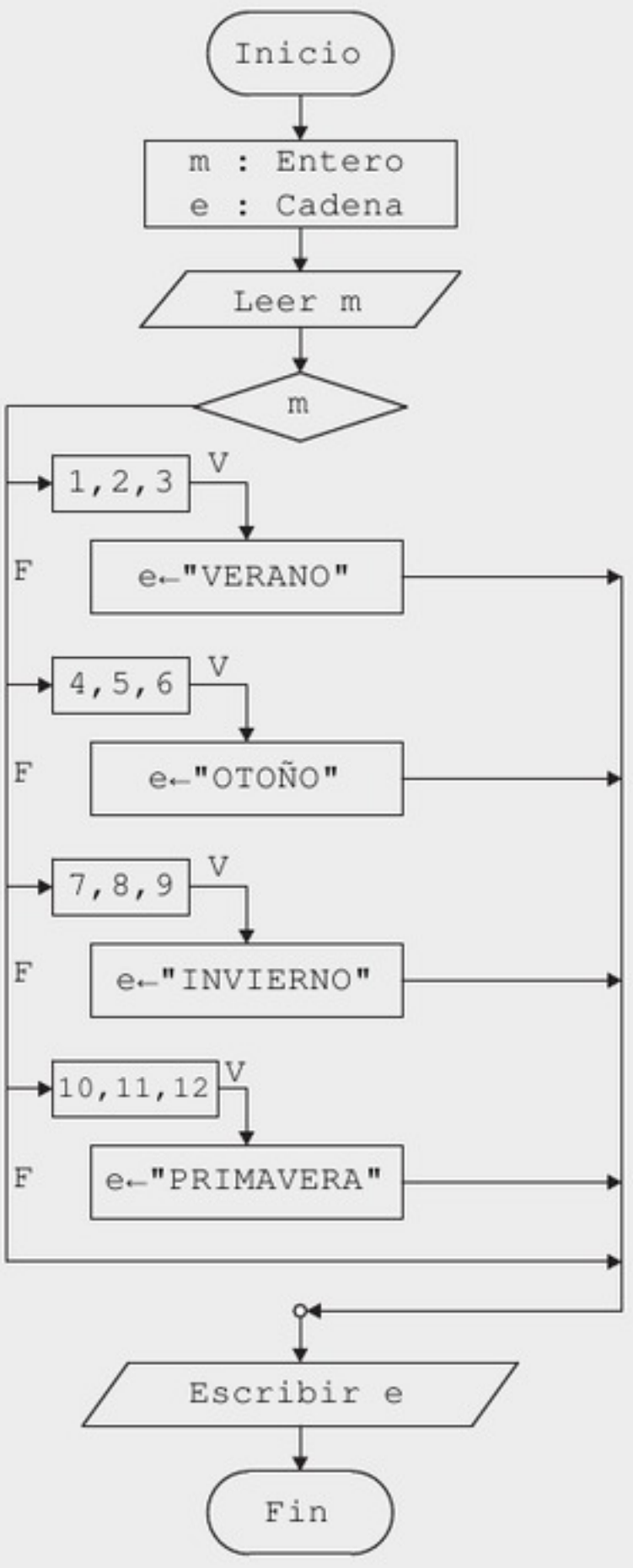
- Estación (e)

**Diseño:**

**Interfaz de usuario**


## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Variables
   m : Entero
   e : Cadena

   //Entrada
   Leer m

   //Proceso
   En Caso que m Sea
       Caso 1,2,3
          e ← "VERANO"
       Caso 4,5,6
          e ← "OTOÑO"
       Caso 7,8,9
          e ← "INVIERNO"
       Caso 10,11,12
          e ← "PRIMAVERA"
   Fin Caso

   //Salida
   Escribir e

Fin
```

**Codificación:**

```
'Variables
Dim m As Integer
Dim e As String

'Entrada
m = Val(Me.txtm.Text)

'Proceso
Select Case m
    Case 1, 2, 3
          e = "VERANO"
    Case 4, 5, 6
          e = "OTOÑO"
    Case 7, 8, 9
          e = "INVIERNO"
    Case 10, 11, 12
          e = "PRIMAVERA"
End Select

'Salida
Me.txte.Text = e
```

## Problema n.° 31

**Enunciado**: Dada la nota promedio de un alumno, obtener la categoría según la siguiente tabla:

| Promedio | Categoría |
|---|---|
| Entre 0 y 5 | Pésimo |
| Entre 6 y 10 | Malo |
| Entre 11 y 14 | Regular |
| Entre 15 y 17 | Bueno |
| Entre 18 y 20 | Excelente |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el promedio; luego, que el sistema verifique y devuelva la categoría.

**Entrada**

- Promedio (p)

**Salida**

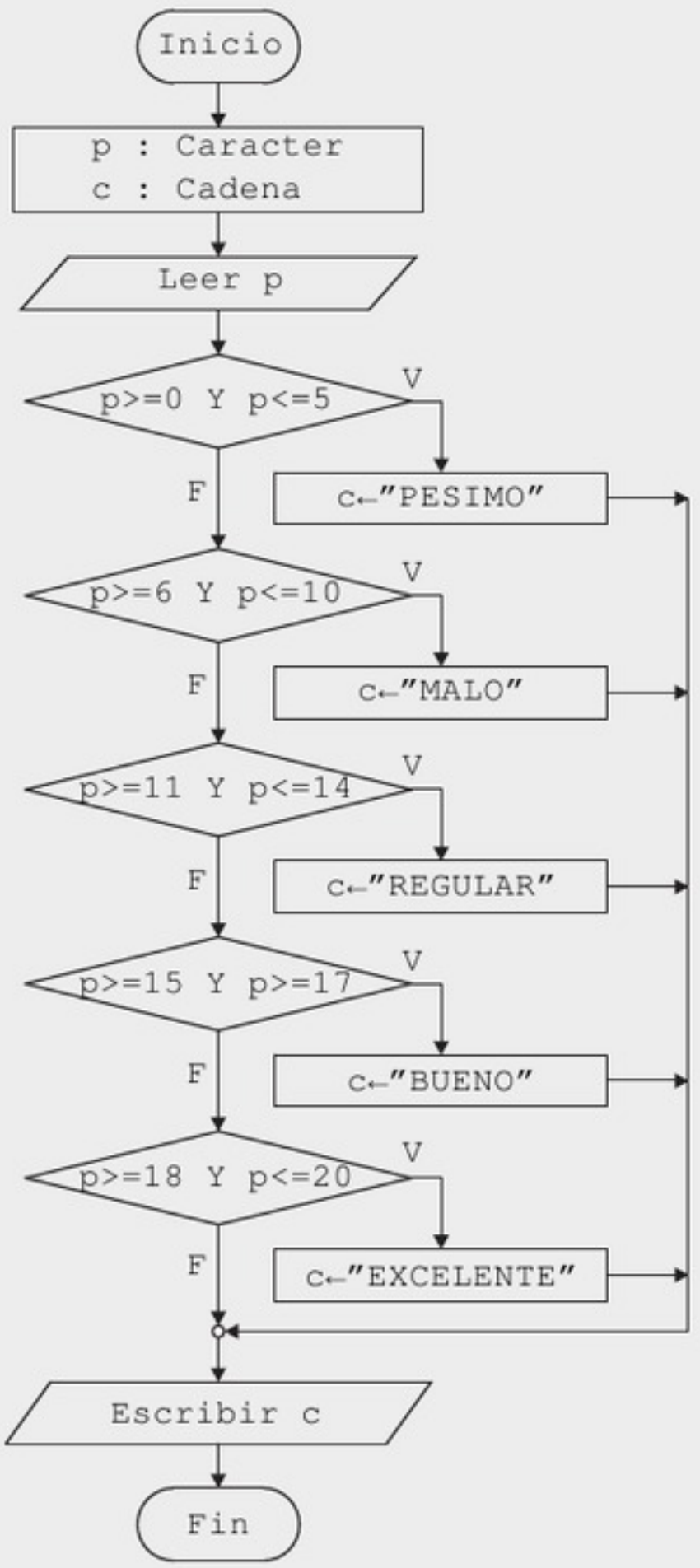
- Categoría (c)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   p : Entero
   c : Cadena

   //Entrada
   Leer p

   //Proceso
   Si p >= 0 Y p <= 5 Entonces
       c ← "PESIMO"
   SiNoSi p >= 6 Y p <= 10 Entonces
       c ← "MALO"
   SiNoSi p >= 11 Y p <= 14 Entonces
       c ← "REGULAR"
   SiNoSi p >= 15 Y p <= 17 Entonces
              c ← "BUENO"
   SiNoSi p >= 18 Y p <= 20 Entonces
       c ← "EXCELENTE"
   Fin Si

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim p As Integer
Dim c As String

'Entrada
p = Val(Me.txtp.Text)

'Proceso
If p >= 0 And p <= 5 Then
     c = "PESIMO"
ElseIf p >= 6 And p <= 10 Then
     c = "MALO"
ElseIf p >= 11 And p <= 14 Then
     c = "REGULAR"
ElseIf p >= 15 And p <= 17 Then
     c = "BUENO"
ElseIf p >= 18 And p <= 20 Then
     c = "EXCELENTE"
End If

'Salida
Me.txtc.Text = c
```

## Problema n.° 32

**Enunciado**: Al ingresar el día y el número de un mes, devolver la estación del año de acuerdo a la siguiente tabla:

| Estación | Tiempo |
|---|---|
| Verano | Del 21 de diciembre al 20 de marzo |
| Otoño | Del 21 de marzo al 21 de junio |
| Invierno | Del 22 de junio al 22 de septiembre |
| Primavera | Del 23 de septiembre al 20 de diciembre |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el día y el mes; luego, que el sistema verifique y devuelva la estación.

**Entrada**

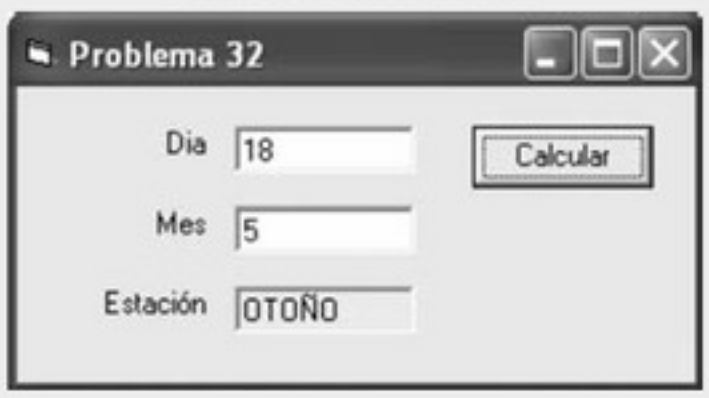
- Dia (d)
- Mes (m)

**Salida**

- Estación (e)

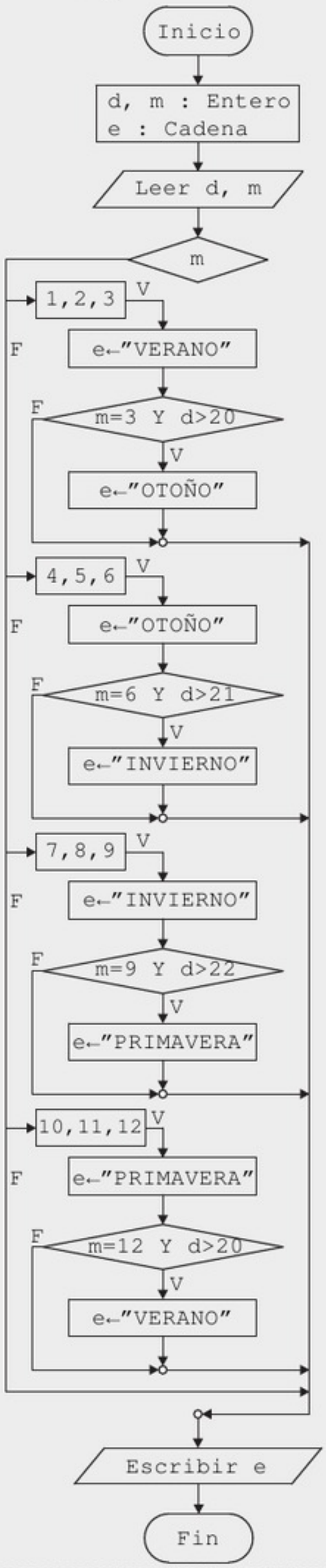
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   d, m : Entero
   e : Cadena

   //Entrada
   Leer d, m

   //Proceso
En Caso que m Sea
    Caso 1, 2, 3
      e ← "VERANO"
      Si m = 3 Y d > 20 Entonces
          e ← "OTOÑO"
      Fin Si
    Caso 4, 5, 6
      e ← "OTOÑO"
      Si m = 6 Y d > 21 Entonces
          e ← "INVIERNO"
      Fin Si
    Caso 7, 8, 9
      e ← "INVIERNO"
      Si m = 9 Y d > 22 Entonces
          e ← "PRIMAVERA"
      Fin Si
    Caso 10, 11, 12
      e ← "PRIMAVERA"
      Si m = 12 Y d > 20 Entonces
          e ← "VERANO"
      Fin Si
   Fin Caso

   //Salida
   Escribir e

Fin
```

**Codificación:**

```
'Variables
Dim d As Integer
Dim m As Integer
Dim e As String

'Entrada
d = Val(Me.txtd.Text)
m = Val(Me.txtm.Text)

'Proceso
Select Case m
    Case 1, 2, 3
          e = "VERANO"
          If m = 3 And d > 20 Then
                e = "OTOÑO"
          End If
    Case 4, 5, 6
          e = "OTOÑO"
          If m = 6 And d > 21 Then
                e = "INVIERNO"
          End If
    Case 7, 8, 9
          e = "INVIERNO"
          If m = 9 And d > 22 Then
                e = "PRIMAVERA"
          End If
    Case 10, 11, 12
          e = "PRIMAVERA"
          If m = 12 And d > 20 Then
                e = "VERANO"
          End If
End Select

'Salida
Me.txte.Text = e
```

## Problema n.° 33

**Enunciado**: En una universidad se ha establecido los siguientes puntajes de ingreso a sus respectivas facultades.

| Facultad | Puntaje mínimo |
|---|---|
| Sistemas | 100 |
| Electrónica | 90 |
| Industrial | 80 |
| Administración | 70 |

De acuerdo al puntaje obtenido por un postulante, determinar la facultad a la cual ingresó o dar un mensaje correspondiente en caso no ingrese.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el puntaje; luego, que el sistema verifique y devuelva la facultad a la que ingresó.

**Entrada**

- Puntaje (p)

**Salida**

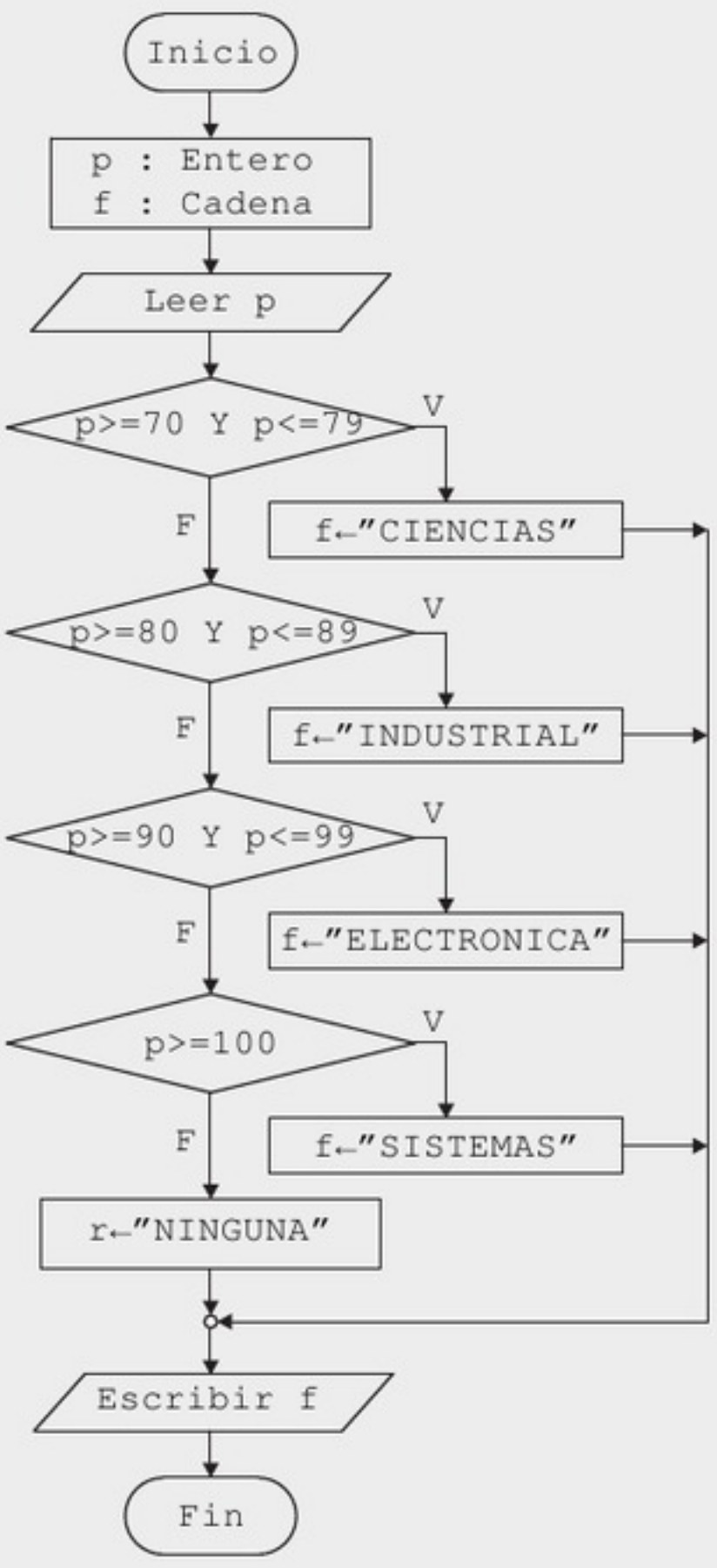
- Facultad (f)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   p : Entero
   f : Cadena

   //Entrada
   Leer p

   //Proceso
   Si p >= 70 Y p <= 79 Entonces
       f ← "CIENCIAS"
   SiNoSi p >= 80 Y p <= 89 Entonces
       f ← "INDUSTRIAL"
   SiNoSi p >= 90 Y p <= 99 Entonces
       f ← "ELECTRONICA"
   SiNoSi p >= 100 Entonces
       f ← "SISTEMAS"
   SiNo
       f ← "NINGUNO"
   Fin Si

   //Salida
   Escribir f

Fin
```

**Codificación:**

```
'Variables
Dim p As Integer
Dim f As String

'Entrada
p = Val(Me.txtp.Text)

'Proceso
If p >= 70 And p <= 79 Then
     f = "CIENCIAS"
ElseIf p >= 80 And p <= 89 Then
     f = "INDUSTRIAL"
ElseIf p >= 90 And p <= 99 Then
     f = "ELECTRONICA"
ElseIf p >= 100 Then
     f = "SISTEMAS"
Else
     f = "NINGUNA"
End If

'Salida
Me.txtf.Text = f
```

## Problema n.° 34

**Enunciado**: Determine el importe a pagar para el examen de admisión de una universidad, cuyo valor depende del nivel socioeconómico y el colegio de procedencia.

| | Nivel social | | |
|---|---|---|---|
| **Colegio** | **A** | **B** | **C** |
| **N**acional | 300 | 200 | 100 |
| **P**articular | 400 | 300 | 200 |

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el colegio y el nivel socioeconómico; luego, que el sistema verifique y determine el monto a pagar.

**Entrada**

- Colegio (c)
- Nivel (n)

**Salida**

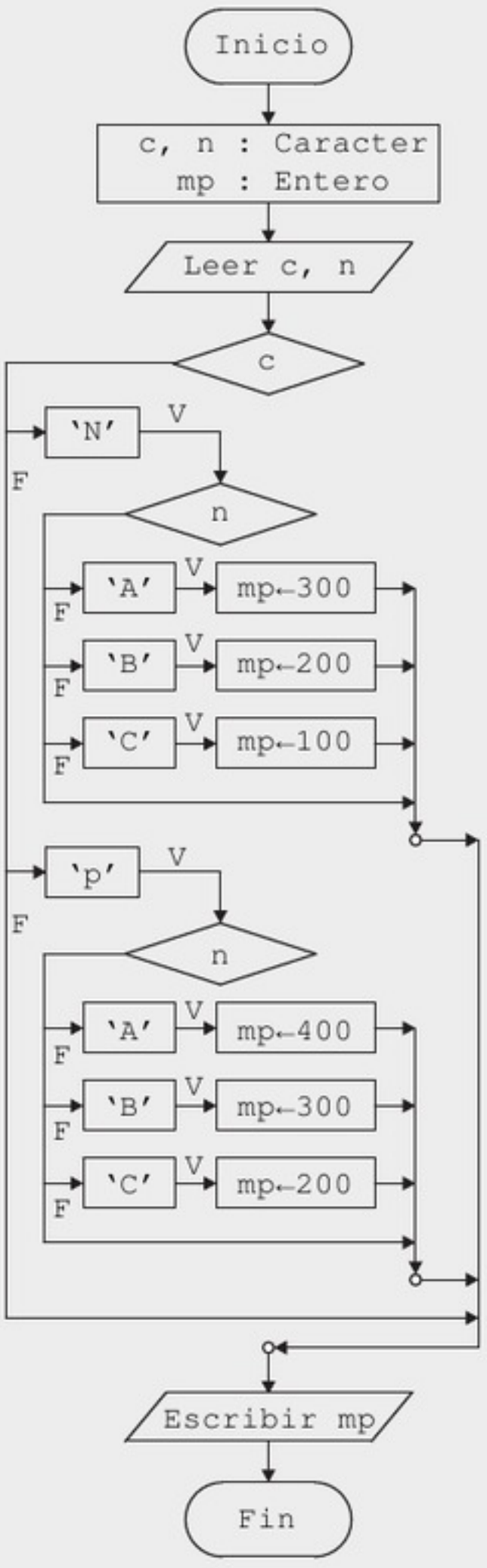
- Monto a pagar (mp)

**Diseño:**

**Interfaz de usuario**


## Algoritmo

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c, n : Carácter
   mp : Entero

   //Entrada
   Leer c, n

   //Proceso
   En Caso que c Sea
       Caso 'N'
              En Caso que n Sea
                     Caso 'A'
                            mp ← 300
                     Caso 'B'
                            mp ← 200
                     Caso 'C'
                            mp ← 100
              Fin Caso
       Caso 'P'
              En Caso que n Sea
                     Caso 'A'
                            mp ← 400
                     Caso 'B'
                            mp ← 300
                     Caso 'C'
                            mp ← 200
              Fin Caso
   Fin Caso

   //Salida
   Escribir mp

Fin
```

**Codificación:**

```
'Variables
Dim c As String
Dim n As String
Dim mp As Integer

'Entrada
c = Me.txtc.Text
n = Me.txtn.Text

'Proceso
Select Case c
     Case "N"
          Select Case n
               Case "A"
                    mp = 300
               Case "B"
                    mp = 200
               Case "C"
                    mp = 100
          End Select
     Case "P"
          Select Case n
               Case "A"
                    mp = 400
               Case "B"
                    mp = 300
               Case "C"
                    mp = 200
          End Select
End Select

'Salida
Me.txtmp.Text = Str(mp)
```

## Problema n.° 35

**Enunciado**: Dado el número del mes y el año (cuatro dígitos) de una fecha, determinar, en letras, el nombre del mes y cuantos días tiene. Considerar que febrero tiene 28 o 29 días si el año es bisiesto, un año es bisiesto si es múltiplo de 4, pero no de 100 y sí de 400.

**Análisis**: Para la solución de este problema se requiere que el usuario ingrese el número del mes y el año; luego, que el sistema verifique y determine cuántos días tiene y el nombre del mes.

**Entrada**

- Mes (m)
- Año (a)

**Salida**

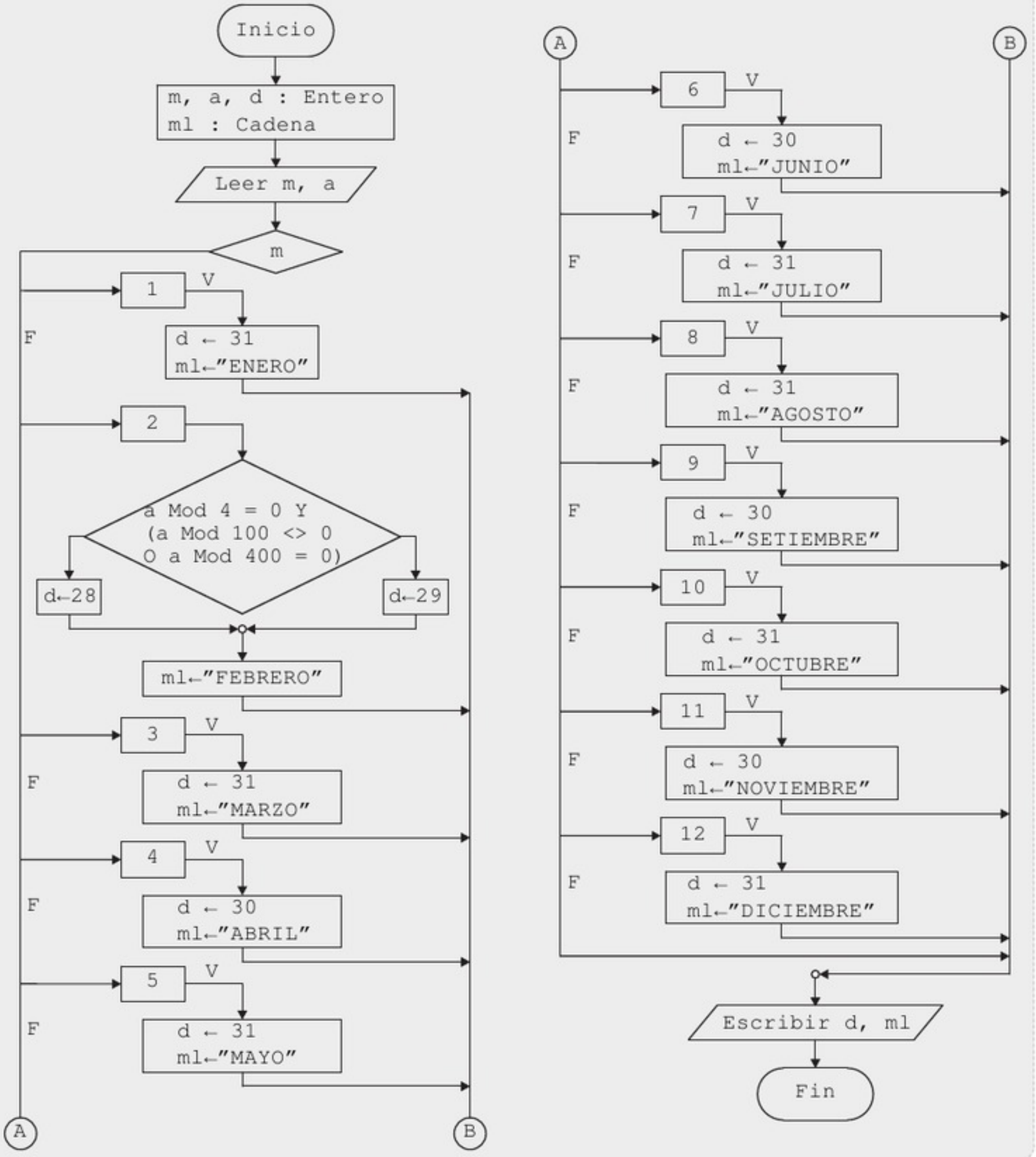
- Dias (d)
- Mes letras (ml)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**



**Pseudocódigo**

```
Inicio

   //Variables
   m, a, d : Entero
   ml : Cadena

   //Entrada
   Leer m, a

   //Proceso
   En Caso que m Sea
       Caso 1
          d ← 31
          ml ← "ENERO"
       Caso 2
          Si a Mod 4 = 0 Y (a Mod 100 <> 0 O
                            a Mod 400 = 0) Entonces
             d ← 29
          SiNo
             d ← 28
          Fin Si
          ml ← "FEBRERO"
       Caso 3
          d ← 31
          ml ← "MARZO"
       Caso 4
          d ← 30
          ml ← "ABRIL"
       Caso 5
          d ← 31
          ml ← "MAYO"
       Caso 6
          d ← 30
          ml ← "JUNIO"
       Caso 7
          d ← 31
          ml ← "JULIO"
       Caso 8
          d ← 31
          ml ← "AGOSTO"
       Caso 9
          d ← 30
          ml ← "SEPTIEMBRE"
       Caso 10
          d ← 31
          ml ← "OCTUBRE"
       Caso 11
          d ← 30
          ml ← "NOVIEMBRE"
       Caso 12
          d ← 31
          ml ← "DICIEMBRE"
   Fin Caso

   //Salida
   Escribir d, ml

Fin
```

**Codificación:**

```
'Variables
Dim m As Integer
Dim a As Integer
Dim d As Integer
Dim ml As String

'Entrada
m = Val(Me.txtm.Text)
a = Val(Me.txta.Text)

'Proceso
Select Case m
    Case 1
          d = 31
          ml = "ENERO"
  Case 2
  If a Mod 4 = 0 And (a Mod 100 <> 0 Or a Mod 400 = 0) Then
            d = 29
       Else
            d = 28
       End If
       ml = "FEBRERO"
  Case 3
       d = 31
       ml = "MARZO"
  Case 4
       d = 30
       ml = "ABRIL"
  Case 5
       d = 31
       ml = "MAYO"
  Case 6
       d = 30
       ml = "JUNIO"
  Case 7
       d = 31
       ml = "JULIO"
  Case 8
       d = 31
       ml = "AGOSTO"
    Case 9
          d = 30
          ml = "SEPTIEMBRE"
    Case 10
          d = 31
          ml = "OCTUBRE"
    Case 11
          d = 30
          ml = "NOVIEMBRE"
    Case 12
          d = 31
          ml = "DICIEMBRE"
End Select

'Salida
Me.txtd.Text = Str(d)
Me.txtml.Text = ml
```

## Problema n.° 36

**Enunciado**: Una empresa ha establecido diferentes precios a sus productos, según la calidad.

| Producto \ Calidad | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 5000 | 4500 | 4000 |
| 2 | 4500 | 4000 | 3500 |
| 3 | 4000 | 3500 | 3000 |

Cree un programa que devuelva el precio a pagar por un producto y una calidad dada.

**Análisis**: Para la solución de este problema se requiere que el usuario ingrese la calidad y el producto; luego, que el sistema verifique y determine el precio.

**Entrada**

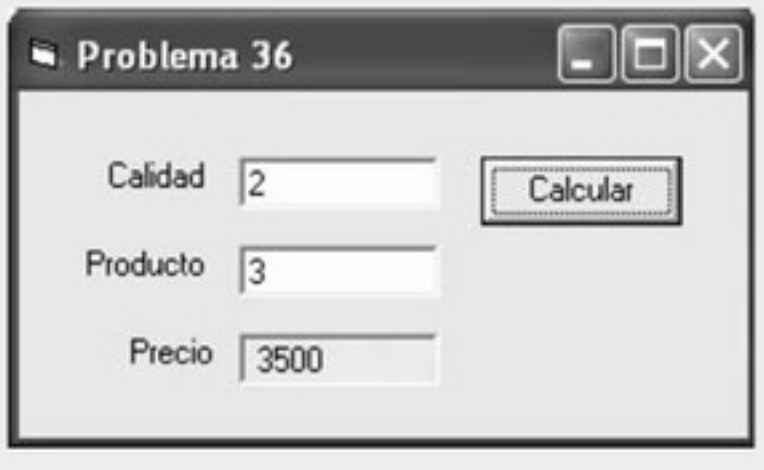
- Calidad (c)
- Producto (p)

**Salida**

- Precio (precio)

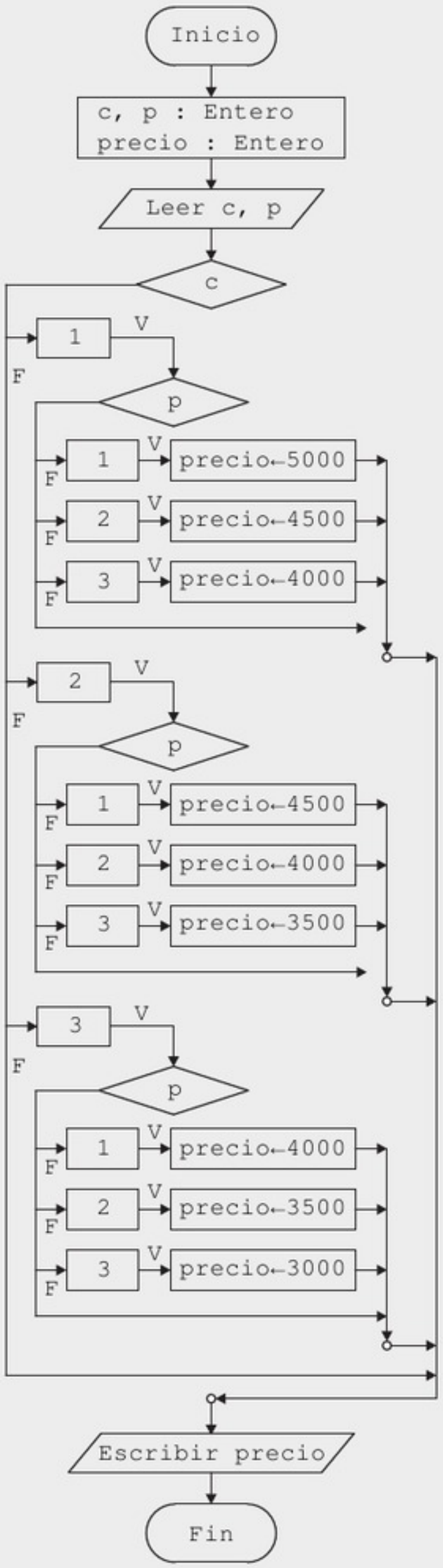
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c, p : Entero
   precio : Entero

   //Entrada
   Leer c, p

   //Proceso
En Caso que c Sea
    Caso 1
          En Caso que p Sea
             Caso 1
                precio ← 5000
             Caso 2
                precio ← 4500
             Caso 3
                precio ← 4000
          Fin Caso
    Caso 2
          En Caso que p Sea
             Caso 1
                precio ← 4500
             Caso 2
                precio ← 4000
             Caso 3
                precio ← 3500
          Fin Caso
    Caso 3
          En Caso que p Sea
             Caso 1
                precio ← 4000
             Caso 2
                precio ← 3500
             Caso 3
                precio ← 3000
          Fin Caso
   Fin Caso

   //Salida
   Escribir precio

Fin
```

**Codificación:**

```
'Variables
Dim c As Integer
Dim p As Integer
Dim precio As Integer

'Entrada
c = Val(Me.txtc.Text)
p = Val(Me.txtp.Text)

'Proceso
Select Case c
    Case 1
          Select Case p
                Case 1
                     precio = 5000
                Case 2
                     precio = 4500
           Case 3
                     precio = 4000
          End Select
    Case 2
          Select Case p
                Case 1
                     precio = 4500
                Case 2
                     precio = 4000
                Case 3
                     precio = 3500
          End Select
    Case 3
          Select Case p
                Case 1
                     precio = 4000
                Case 2
                     precio = 3500
                Case 3
                     precio = 3000
          End Select
End Select

'Salida
Me.txtprecio.Text = Str(precio)
```

## Problema n.° 37

**Enunciado**: Diseñe un algoritmo que califique el puntaje obtenido en el lanzamiento de tres dados en base a la cantidad de seis obtenidos, de acuerdo a lo siguiente:

**Tres seis:** Oro

**Dos seis:** Plata

**Un seis:** Bronce

**Ningún seis:** Perdió

**Análisis**: Para la solución de este problema se requiere que el usuario ingrese el puntaje de los dados; luego, que el sistema verifique y determine el premio.

**Entrada**

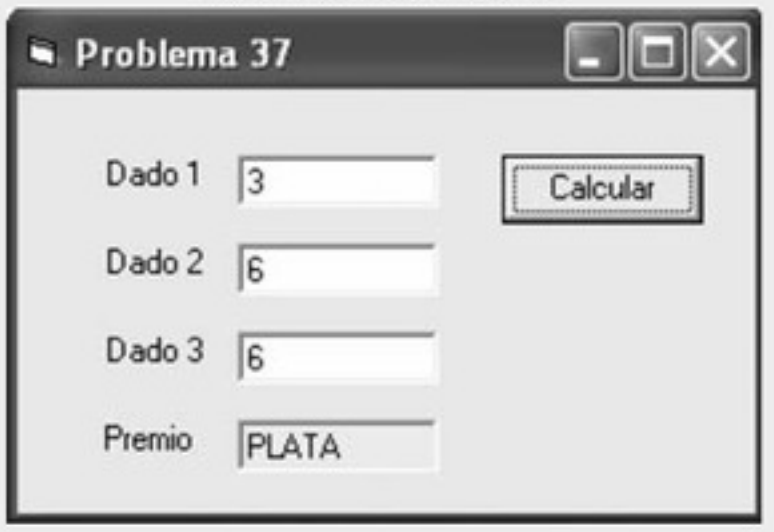
- Primer dado (d1)
- Segundo dado (d2)
- Tercer dado (d3)

**Salida**

- Premio (p)

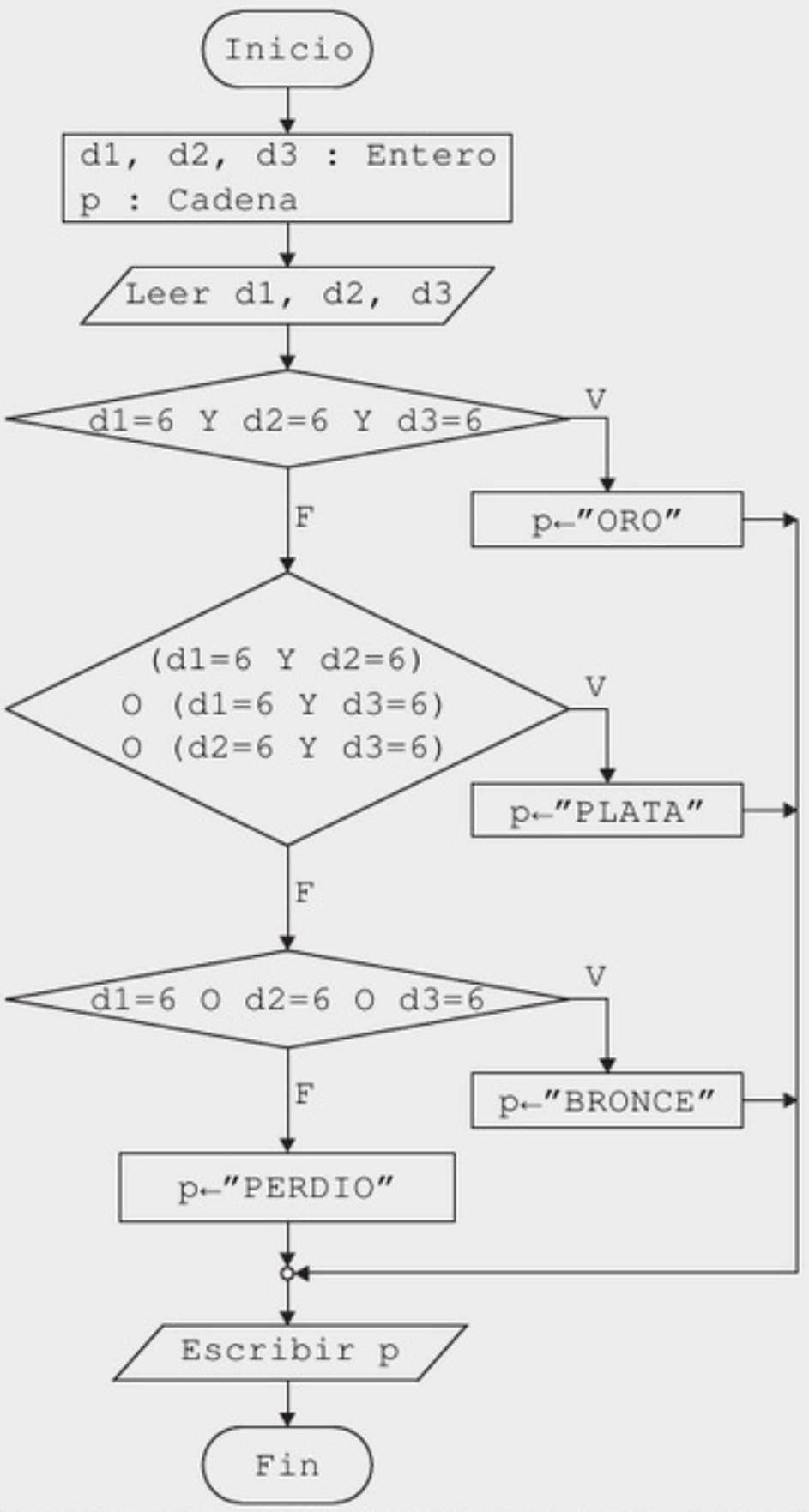
**Diseño:**

**Interfaz de usuario**




## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Variables
   d1, d2, d3 : Entero
   p : Cadena

   //Entrada
   Leer d1, d2, d3

   //Proceso
   Si d1=6 Y d2=6 Y d3=6 Entonces
       p ← "ORO"
   SiNoSi (d1=6 Y d2=6) O (d1=6 Y d3=6)
          O (d2=6 Y d3=6) Entonces
       p ← "PLATA"
   SiNoSi  d1=6 O d2=6 O d3=6 Entonces
       p ← "BRONCE"
   SiNo
       p ← "PERDIO"
   Fin Si

   //Salida
   Escribir p

Fin
```

**Codificación:**

```
'Variables
Dim d1 As Integer
Dim d2 As Integer
Dim d3 As Integer
Dim p As String

'Entrada
d1 = Val(Me.txtd1.Text)
d2 = Val(Me.txtd2.Text)
d3 = Val(Me.txtd3.Text)

'Proceso
If d1 = 6 And d2 = 6 And d3 = 6 Then
     p = "ORO"
ElseIf (d1 = 6 And d2 = 6) Or (d1 = 6 And d3 = 6) _
                  Or (d2 = 6 And d3 = 6) Then
     p = "PLATA"
ElseIf d1 = 6 Or d2 = 6 Or d3 = 6 Then
     p = "BRONCE"
Else
     p = "PERDIO"
End If

'Salida
Me.txtp.Text = p
```

## Problema n.° 38

**Enunciado**: Dado el día, mes y año, determine si es una fecha correcta, considere los años bisiestos.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el día, mes y año; luego, que el sistema verifique y determine si el resultado es o no una fecha correcta.

**Entrada**

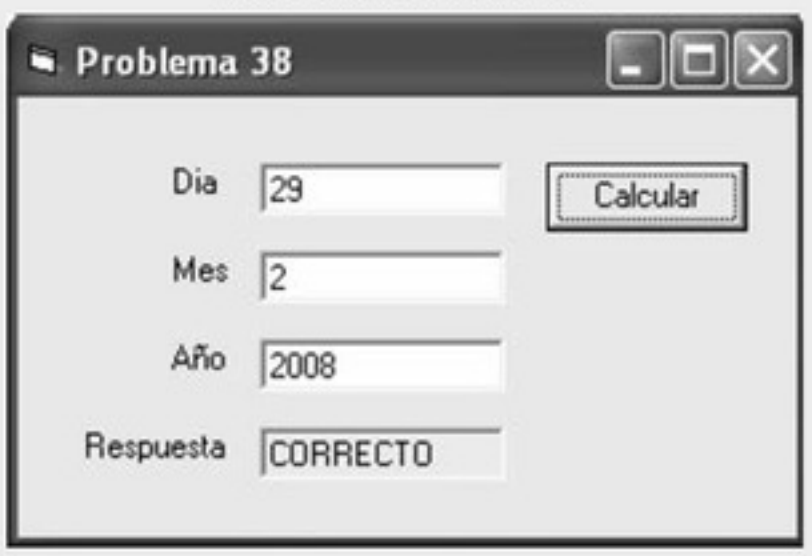
- Día (d)
- Mes (m)
- Año (a)

**Salida**

- Respuesta (r)

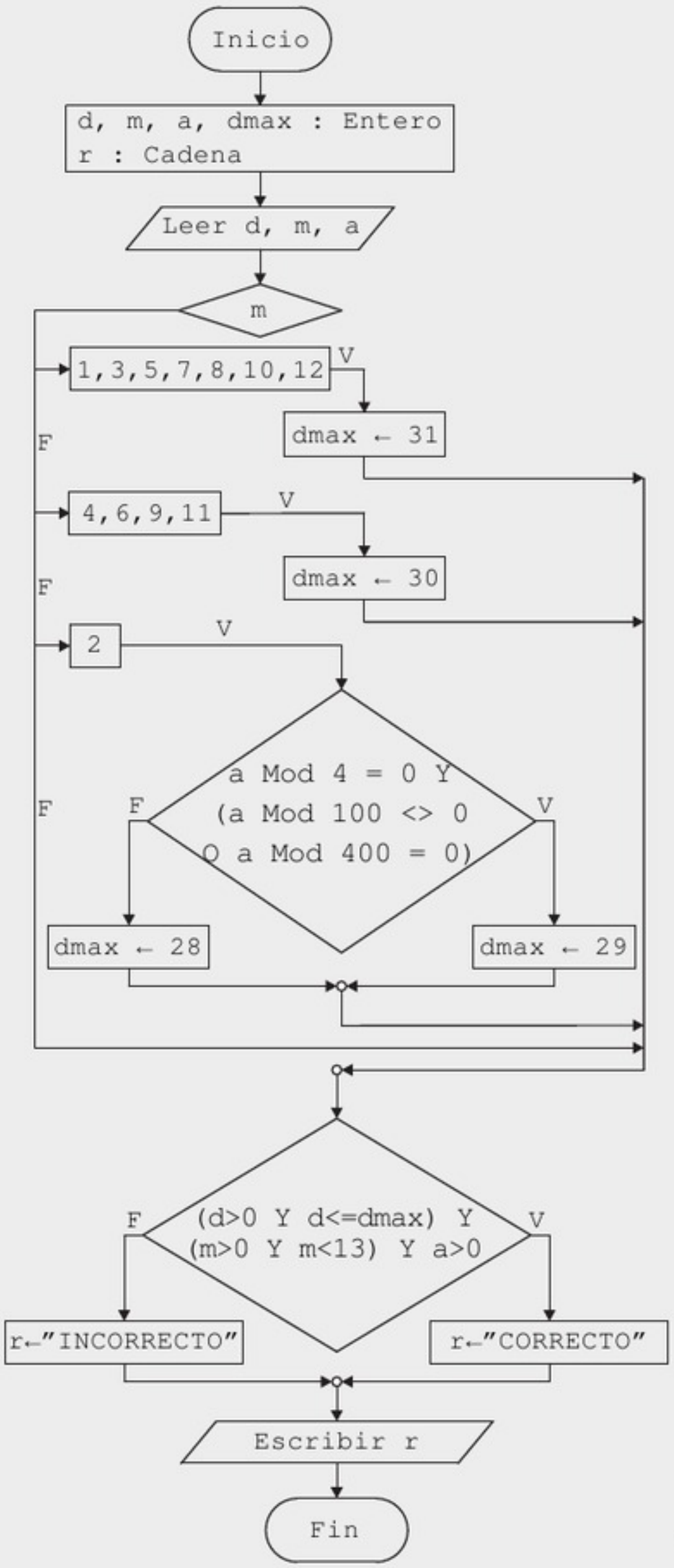
**Diseño:**

**Interfaz de usuario**



## Algoritmo

### Diagrama de flujo




### Pseudocódigo

```
Inicio

   //Variables
   d, m, a, dmax : Entero
   r : Cadena

   //Entrada
   Leer d, m, a

   //Proceso
En Caso que m Sea
    Caso 1,3,5,7,8,10,12
           dmax ← 31
    Caso 4, 6, 9, 11
           dmax ← 30
    Caso 2
    Si a Mod 4 = 0 Y (a Mod 100 <> 0
       O a Mod 400 = 0) Entonces
                  dmax ← 29
          SiNo
                  dmax ← 28
           Fin Si
Fin Caso
Si d>0 Y d<=dmax) Y (m>0 Y m<13)
       Y a>0  Entonces
    r ← "CORRECTO"
SiNo
    r ← "INCORRECTO"
Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim d As Integer
Dim m As Integer
Dim a As Integer
Dim dmax As Integer
Dim r As String
'Entrada
d = Val(Me.txtd.Text)
m = Val(Me.txtm.Text)
a = Val(Me.txta.Text)

'Proceso
Select Case m
     Case 1, 3, 5, 7, 8, 10, 12
           dmax = 31
     Case 4, 6, 9, 11
           dmax = 30
     Case 2
           If a Mod 4 = 0 And (Not (a Mod 100 = 0) _
                                  Or a Mod 400 = 0) Then
                 dmax = 29
           Else
                 dmax = 28
           End If
End Select

If (d > 0 And d <= dmax) And (m > 0 And m < 13) _
                        And a > 0 Then
     r = "CORRECTO"
Else
     r = "INCORRECTO"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 39

**Enunciado**: Dada una fecha válida, halle la fecha del siguiente día.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el día, mes y año; luego, que el sistema devuelve la fecha del siguiente día.

**Entrada**

- Día (d)
- Mes (m)
- Año (a)

**Salida**

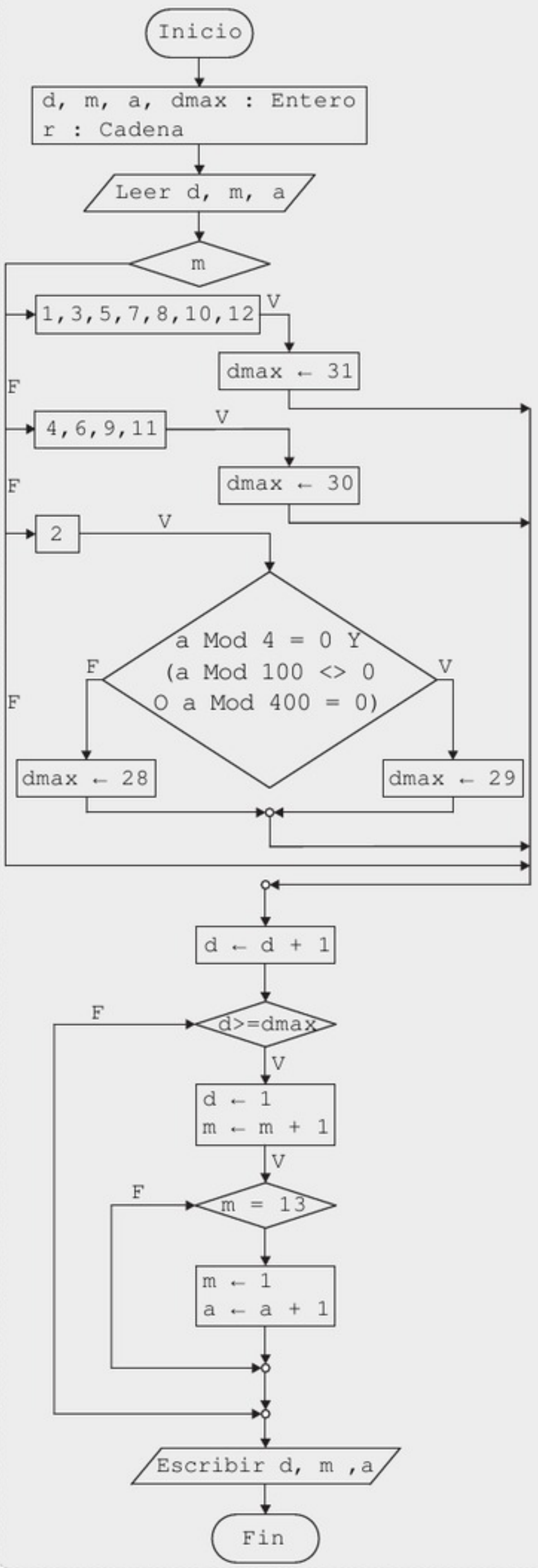
- Día (d)
- Mes (m)
- Año (a)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   d, m, a, dmax : Entero
   r : Cadena

   //Entrada
   Leer d, m, a

   //Proceso
   En Caso que m Sea
       Caso 1,3,5,7,8,10,12
             dmax ← 31
       Caso 4, 6, 9, 11
             dmax ← 30
       Caso 2
       Si a Mod 4 = 0 And (a Mod 100 <> 0
          Or a Mod 400 = 0) Entonces
                   dmax ← 29
             SiNo
                   dmax ← 28
             Fin Si
   Fin Caso
   d = d + 1
   Si d > dmax Entonces
       d ← 1
       m ← m + 1
       Si m = 13 Entonces
             m ← 1
             a ← a + 1
       Fin Si
   Fin Si

   //Salida
   Escribir d, m , a

Fin
```

**Codificación:**

```
'Variables
Dim d As Integer
Dim m As Integer
Dim a As Integer
Dim dmax As Integer

'Entrada
d = Val(Me.txtd1.Text)
m = Val(Me.txtm1.Text)
a = Val(Me.txta1.Text)

'Proceso
Select Case m
     Case 1, 3, 5, 7, 8, 10, 12
            dmax = 31
     Case 4, 6, 9, 11
            dmax = 30
     Case 2
            If a Mod 4 = 0 And (Not (a Mod 100 = 0) _
                              Or a Mod 400 = 0) Then
                  dmax = 29
            Else
                  dmax = 28           End If
End Select

d = d + 1

If d > dmax Then
     d = 1
     m = m + 1
     If m = 13 Then
           m = 1
           a = a + 1
     End If
End If

'Salida
Me.txtd2.Text = Str(d)
Me.txtm2.Text = Str(m)
Me.txta2.Text = Str(a)
```

## Problema n.° 40

**Enunciado**: Convierta a números romanos, números menores a 4000.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el número; luego, que el sistema convierta y devuelva el número en romano.

**Entrada**

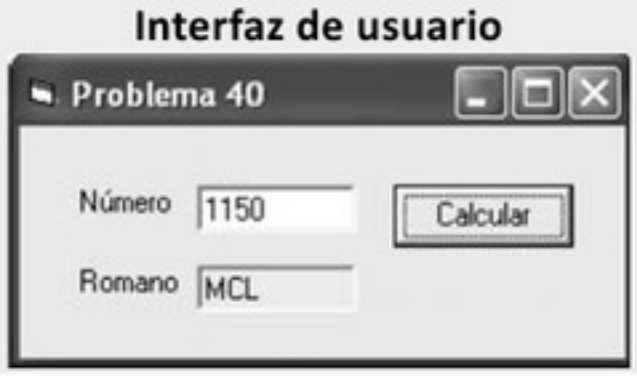
- Número decimal (n)

**Salida**

- Número romano (r)

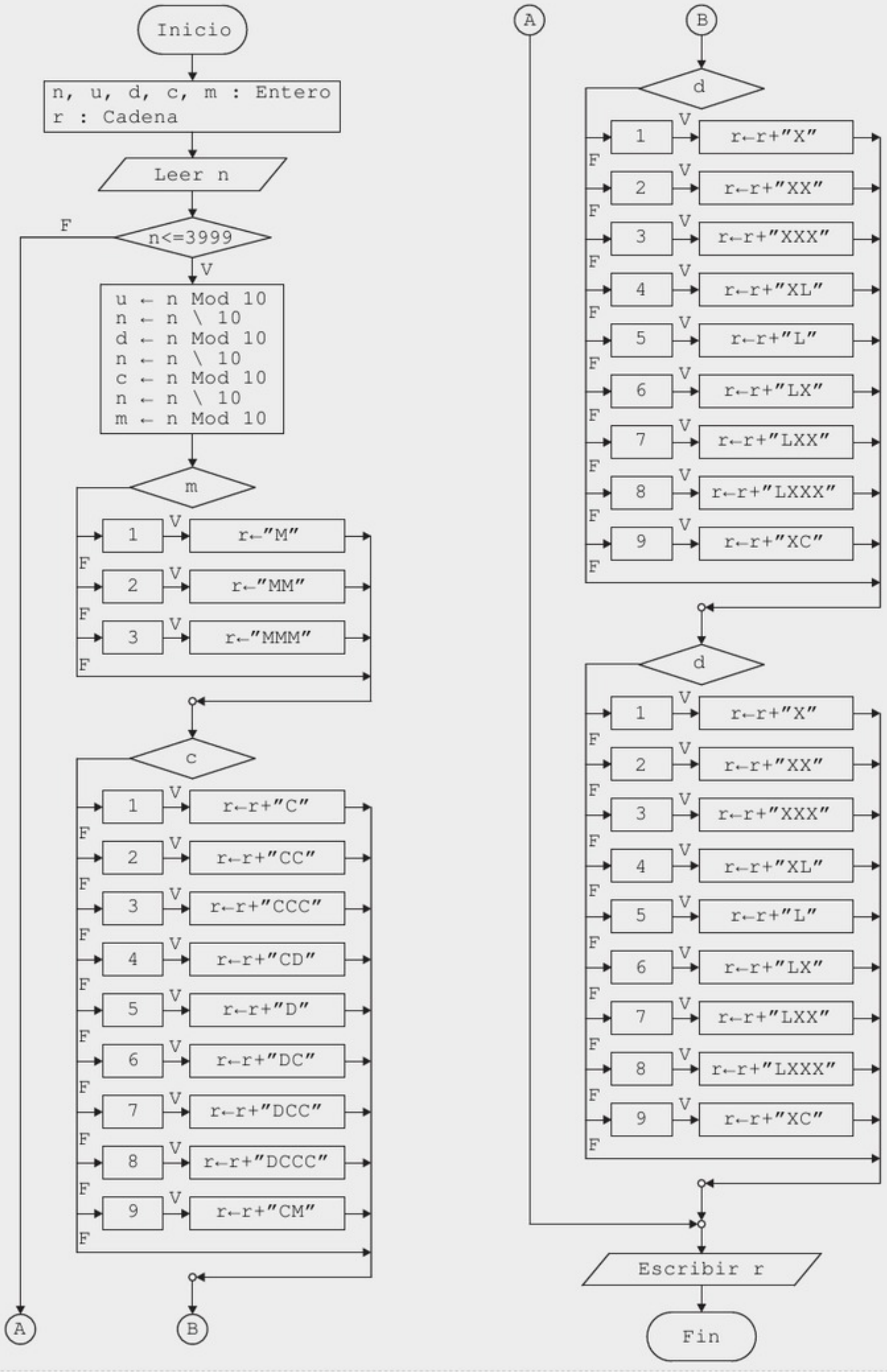
**Diseño:**

**Interfaz de usuario**



## Algoritmo

### Diagrama de flujo



**Pseudocódigo**

```
Inicio

   //Variables
   n, u, d, c, m : Entero
   r : Cadena


   //Entrada
   Leer n

   //Proceso
   Si n <= 3999 Entonces
        u ← n Mod 10
   n ← n \ 10
   d ← n Mod 10
   n ← n \ 10
   c ← n Mod 10
   n ← n \ 10
   m ← n Mod 10

        En Caso que m Sea
              Caso 1
                    r ← "M"
              Caso 2
                    r ← "MM"
              Caso 3
                    r ← "MMM"
        Fin Caso
        En Caso que c Sea
              Caso 1
                    r ← r + "C"
              Caso 2
                    r ← r + "CC"
              Caso 3
                    r ← r + "CCC"
              Caso 4
                    r ← r + "CD"
              Caso 5
                    r ← r + "D"
              Caso 6
                    r ← r + "DC"
              Caso 7
                    r ← r + "DCC"
              Caso 8
                    r ← r + "DCCC"
              Caso 9
                    r ← r + "CM"
        Fin Caso
        En Caso que d Sea
```

```
                Caso 1
                      r ← r + "X"
                Caso 2
                      r ← r + "XX"
                Caso 3
                      r ← r + "XXX"
                Caso 4
                      r ← r + "XL"
                Caso 5
                      r ← r + "L"
                Caso 6
                      r ← r + "LX"
                Caso 7
                      r ← r + "LXX"
                Caso 8
                      r ← r + "LXXX"
                Caso 9
                      r ← r + "XC"
          Fin Caso
          En Caso que u Sea
                Caso 1
                      r ← r + "I"
                Caso 2
                      r ← r + "II"
                Caso 3
                      r ← r + "III"
                Caso 4
                      r ← r + "IV"
                Caso 5
                      r ← r + "V"
                Caso 6
                      r ← r + "VI"
                Caso 7
                      r ← r + "VII"
                Caso 8
                      r ← r + "VIII"
                Caso 9
                      r ← r + "IX"
             Fin Caso
       Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim u As Integer
Dim d As Integer
Dim c As Integer
Dim m As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
If n <= 3999 Then
    u = n Mod 10
    n = n \ 10
    d = n Mod 10
    n = n \ 10
    c = n Mod 10
    n = n \ 10
    m = n Mod 10
    Select Case m
        Case 1
            r = "M"
        Case 2
            r = "MM"
        Case 3
            r = "MMM"
    End Select
    Select Case c
        Case 1
            r = r + "C"
        Case 2
            r = r + "CC"
        Case 3
            r = r + "CCC"
        Case 4
            r = r + "CD"
        Case 5
            r = r + "D"
        Case 6
            r = r + "DC"
        Case 7
            r = r + "DCC"
        Case 8
            r = r + "DCCC"
        Case 9
            r = r + "CM"
    End Select
    Select Case d
```

```
            Case 1
                  r = r + "X"
            Case 2
                  r = r + "XX"
            Case 3
                  r = r + "XXX"
            Case 4
                  r = r + "XL"
            Case 5
                  r = r + "L"
            Case 6
                  r = r + "LX"
            Case 7
                  r = r + "LXX"
            Case 8
                  r = r + "LXXX"
            Case 9
                  r = r + "XC"
      End Select
      Select Case u
            Case 1
                  r = r + "I"
            Case 2
                  r = r + "II"
            Case 3
                  r = r + "III"
            Case 4
                  r = r + "IV"
            Case 5
                  r = r + "V"
            Case 6
                  r = r + "VI"
            Case 7
                  r = r + "VII"
            Case 8
                  r = r + "VIII"
            Case 9
                  r = r + "IX"
      End Select
End If

'Salida
Me.txtr.Text = r
```

## 4.3 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 21

**Enunciado**: Dado el número de un mes, devolver el mes en letras.

### Propuesto n.° 22

**Enunciado**: Lea un número del 1 al 7 y devuelva el día de la semana, considere que 1 es domingo.

### Propuesto n.° 23

**Enunciado**: Dado los siguientes operadores aritméticos +,-, * y / , devuelva el nombre del operador.

### Propuesto n.° 24

**Enunciado**: Dado el número de un canal de televisión, determine cual es el nombre del canal.

### Propuesto n.° 25

**Enunciado**: En una empresa se ha determinado la siguiente política de descuento.

| Tarjeta \ Sexo | Hombres | Mujeres |
|---|---|---|
| Obrero | 15 % | 10 % |
| Empleado | 20 % | 15 % |

Determine mediante un programa cuál será el monto del descuento al sueldo ingresado de un trabajador.

### Propuesto n.° 26

**Enunciado**: Una frutería ofrece las manzanas con descuento, según la siguiente tabla:

| Kilos | % Descuento |
|---|---|
| 0- 2 | 0 % |
| 2.01- 5 | 10 % |
| 5.01- 10 | 20 % |
| Mayor a 10 | 30 % |

Determinar cuánto pagará una persona que quiera compra manzanas en esa frutería.

## Propuesto n.º 27

**Enunciado**: Obtenga el nombre del estado civil, según la siguiente tabla:

| Código | Estado civil |
|---|---|
| 0 | Soltero |
| 1 | Casado |
| 2 | Divorciado |
| 3 | Viudo |

## Propuesto n.° 28

**Enunciado**: Determinar el monto que recibirá un trabajador por utilidades, después de ingresar el tiempo de servicio y el cargo, según la siguiente tabla.

| Tiempo de Servicio \ Cargo | Administrador | Contador | Empleado |
|---|---|---|---|
| Entre 0 y 2 años | 2000 | 1500 | 1000 |
| Entre 3 y 5 años | 2500 | 2000 | 1500 |
| Entre 6 y 8 años | 3000 | 2500 | 2000 |
| Mayor a 8 años | 4000 | 3500 | 1500 |

## Propuesto n.° 29

**Enunciado**: Según la siguiente tabla, obtener la ciudad que visitará, después de ingresar su sexo y el puntaje obtenido en un examen.

| Puntaje \ Sexo | Masculino | Femenino |
|---|---|---|
| Entre 18 y 35 | Arequipa | Cuzco |
| Entre 36 y 75 | Cuzco | Iquitos |
| Mayor a 75 | Iquitos | Arequipa |

## Propuesto n.° 30

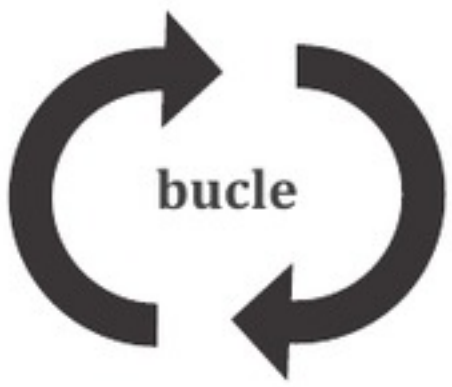
**Enunciado**: Dada una fecha, determine cuantos días faltan para que acabe el año.

# Capítulo 5

# Estructura repetitiva «Mientras»

## 5.1 Introducción

Muchas veces se requiere repetir una o varias instrucciones para llevar a cabo una tarea; en la programación se cuenta con estructuras que permiten realizar este proceso, llamadas también: bucles, iterativas, lazos, entre otros.




Dependiendo del lenguaje de programación, estas incorporan dos o más estructuras repetitivas, dentro de las cuales las infaltables son mientras (**while**) y para (**for**), con las cuales se puede resolver todo problema que involucre procesos repetitivos.

Cuando se trabaja con estas estructuras se utiliza términos como: contadores, acumuladores, forzar la salida del bucle y continuar al inicio del bucle.

## 5.2 Contador

Son variables enteras que se incrementan (+) o decrementan (-) con un valor constante, por ejemplo, una variable «c», cuyo valor se incrementa de 1 en 1; se conoce como variable «contador».

**Ejemplos pseudocódigo**

```
c ← c + 1
i ← i + 2
j ← j - 1
```

**Visual Basic**

```
c = c + 1
i = i + 2
j = j - 1
```

## 5.3 Acumulador

Son variables de cualquier tipo que almacenan valores variables; por ejemplo, la variable «c», cuyo valor se incrementa por el valor que va tomando otra variable llamada «x».

**Ejemplo pseudocódigo**

```
c ← c + x
i ← i + c
j ← j - i
```

**Visual Basic**

```
c = c + x
i = i + c
j = j - i
```

## 5.4 Salir del bucle

Es una instrucción que permite forzar la salida de un bucle, para esto los lenguajes de programación incorporan una instrucción que permite realizar dicha operación.

**Pseudocódigo**

```
Salir
```

**Visual Basic**

```
' Para Salir del Do While
Exit Do

' Para Salir del For
Exit For
```

## 5.5 Continuar al inicio del bucle

Es una instrucción que permite saltar al inicio del bucle para volver a ejecutarse; para esto, los lenguajes de programación incorporan una instrucción que permite realizar dicha operación.

**Pseudocódigo**

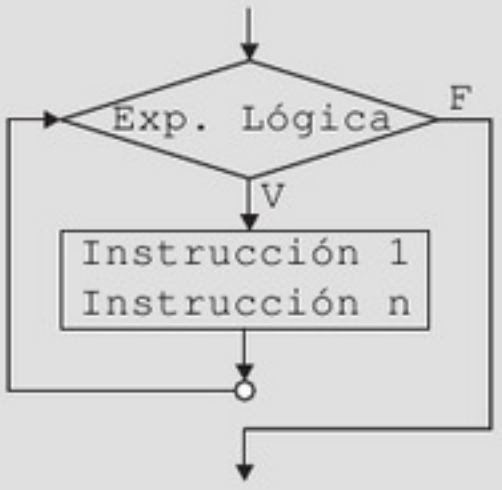
```
Continuar
```

**VB**

```
' Para Continuar al inicio del Do While y For
Continue
```

## 5.6 Estructura repetitiva «Mientras»

Permite repetir una o más instrucciones hasta que la condición (expresión lógica) sea verdadera; cuando la condición es falsa sale del bucle.



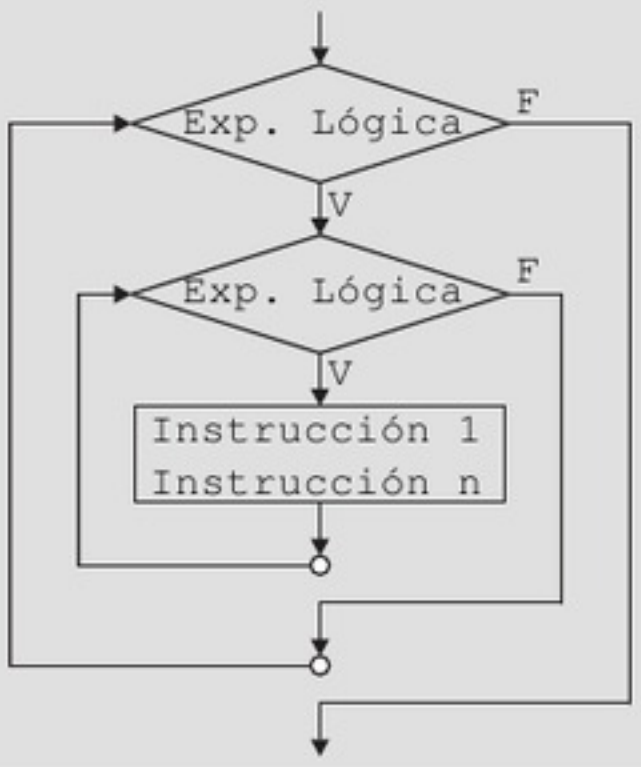

```
Mientras Exp. Lógica

   Instrucción 1
   Instrucción n

Fin Mientras
```

**Sintaxis Visual Basic**

```
Do While <Exp. Log.>

   <instrucción 1>
   <instrucción n>

Loop
```

## 5.7 Estructura repetitiva «Mientras» anidada

Dentro de la estructura repetitiva es posible colocar una o más estructuras repetitivas, así como otras estructuras.




```
Mientras Exp. Lógica

   Mientras Exp. Lógica

      Instrucción 1
      Instrucción n

   Fin Mientras

Fin Mientras
```

**Sintaxis Visual Basic**

```
Do While <Exp. Log.>
   Do While <Exp. Log.>

      <instruccion1>
      <instruccionn>

   Loop
Loop
```

## Problema n.° 41

**Enunciado**: Obtener la suma de los primeros N números naturales positivos.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema realice el proceso para devolver la suma de los N primeros números.

**Entrada**

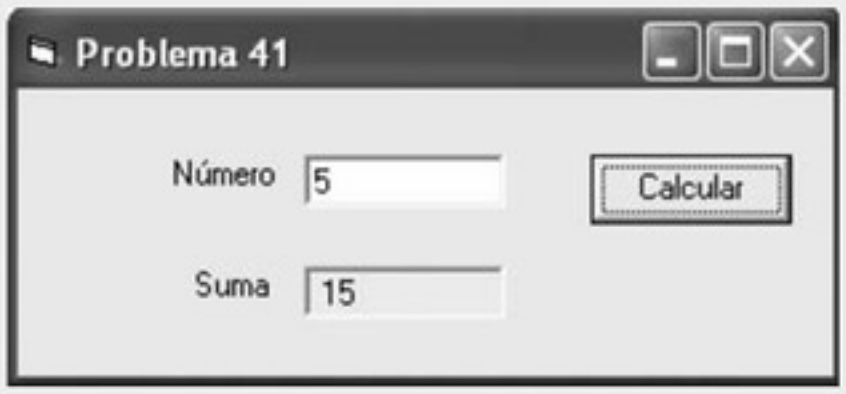
- Número (n)

**Salida**

- Suma (s)

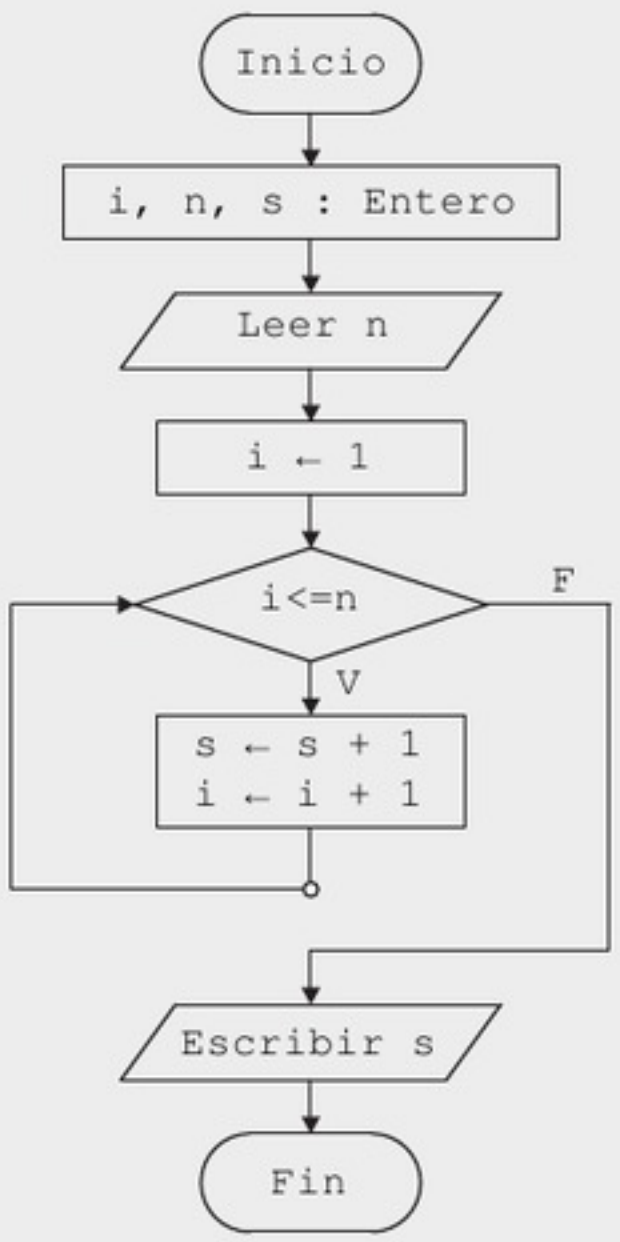
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, n, s : Entero

   //Entrada
   Leer n

   //Proceso
   i ← 1
   Mientras i<=n
       s ← s + 1
       i ← i + 1
   Fin Mientras

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim n As Integer
Dim s As Integer

'Entrada
n = Val(Me.txtn.Text)

'Proceso
i = 1
Do While i <= n
    s = s + i
    i = i + 1
Loop

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 42

**Enunciado**: Dado un rango de números enteros, obtener la cantidad de números enteros que contiene.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el número inicial y final; luego, que el sistema procese y devuelva la cantidad de números enteros que contiene el rango.

**Entrada**

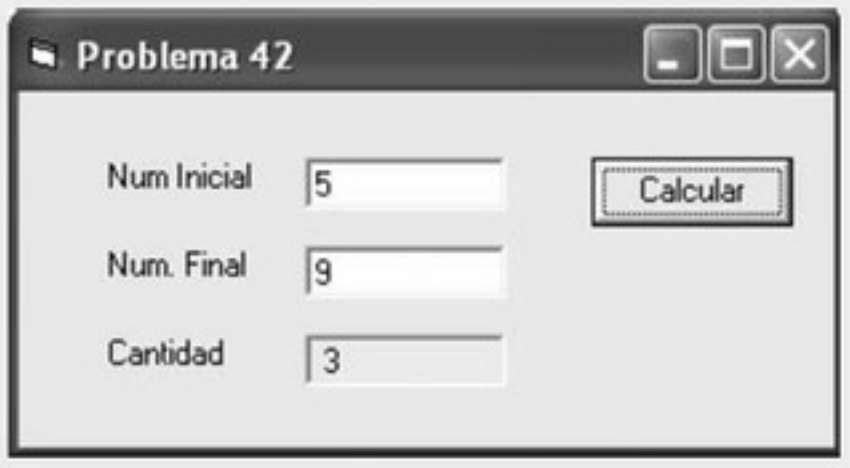
- Número inicial (ni)
- Número final (nf)

**Salida**

- Cantidad (c)

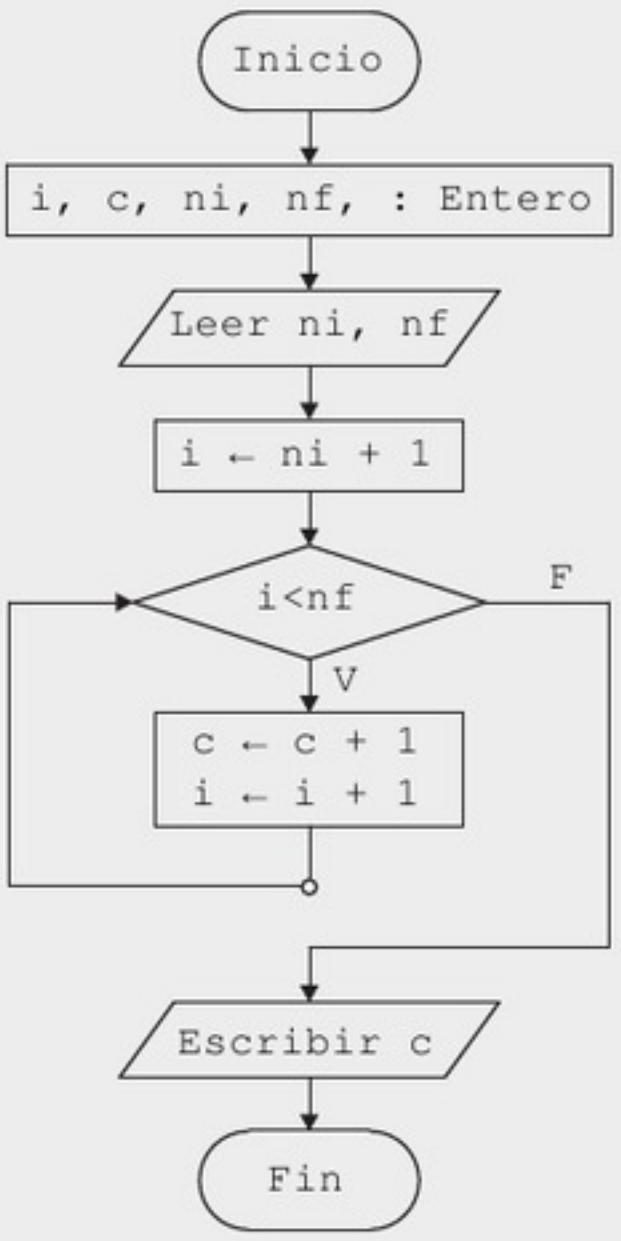
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, c, ni, nf : Entero

   //Entrada
   Leer ni, nf

   //Proceso
   i ← ni + 1
   Mientras i < nf
       c ← c + 1
       i ← i + 1
   Fin Mientras

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim ni As Integer
Dim nf As Integer
Dim c As Integer

'Entrada
ni = Val(Me.txtni.Text)
nf = Val(Me.txtnf.Text)

'Proceso
i = ni + 1
Do While i < nf
     c = c + 1
     i = i + 1
Loop

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 43

**Enunciado**: Dado un rango de números enteros, obtener la cantidad de números pares que contiene.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el número inicial y final; luego, que el sistema procese y devuelva la cantidad de números pares que contiene el rango.

**Entrada**

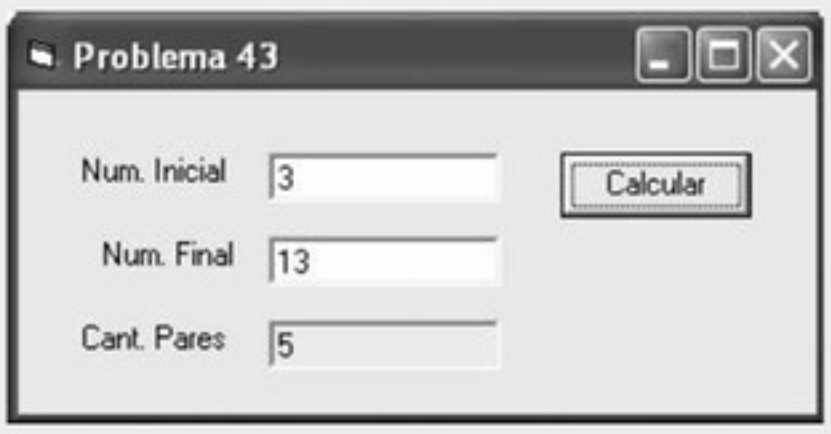
- Número inicial (ni)
- Número final (nf)

**Salida**

- Cantidad de pares (cp)

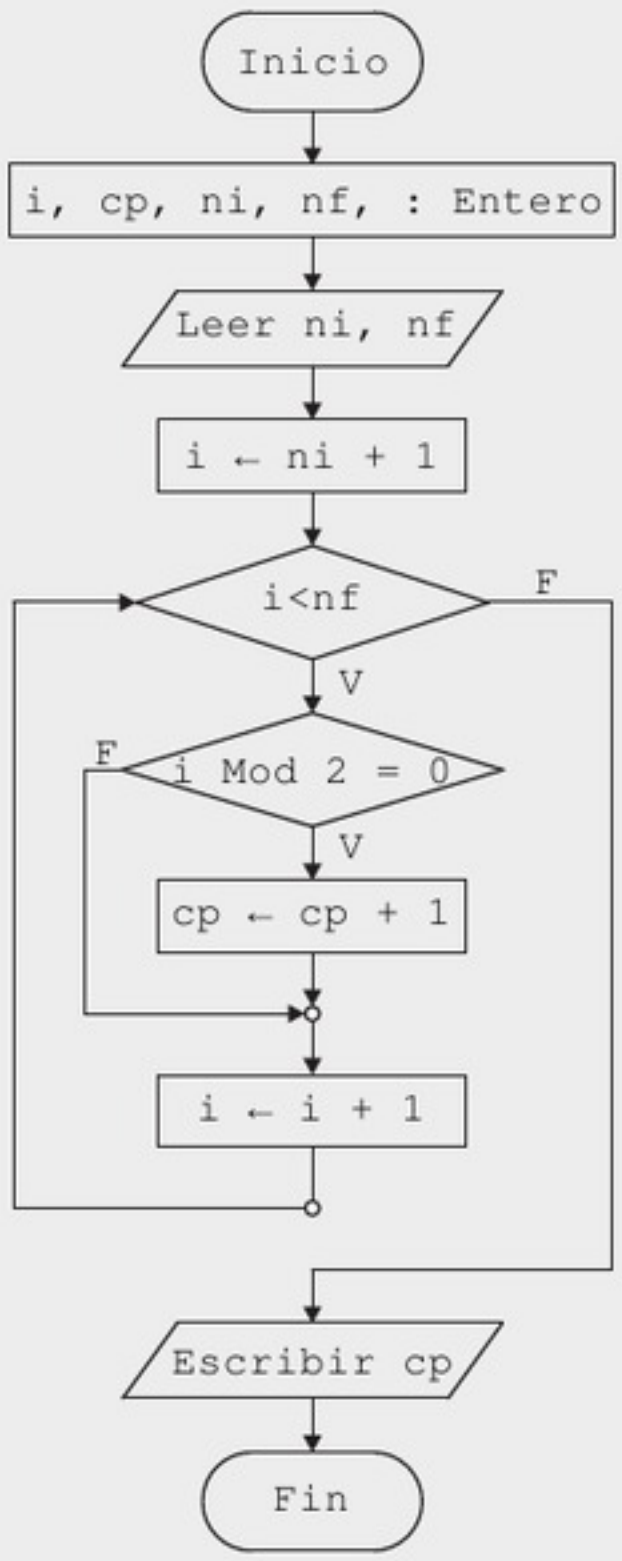
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio
   //Variables
   i, cp, ni, nf : Entero

   //Entrada
   Leer ni, nf

   //Proceso
   i ← ni + 1
   Mientras i < nf
       Si i Mod 2 = 0 Entonces
              cp ← cp + 1
       Fin Si
       i ← i + 1
   Fin Mientras

   //Salida
   Escribir cp

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim ni As Long
Dim nf As Long
Dim cp As Long

'Entrada
ni = Val(Me.txtni.Text)
nf = Val(Me.txtnf.Text)

'Proceso
i = ni + 1
Do While i < nf
     If i Mod 2 = 0 Then
          cp = cp + 1
     End If
     i = i + 1
Loop

'Salida
Me.txtcp.Text = cp
```

## Problema n.° 44

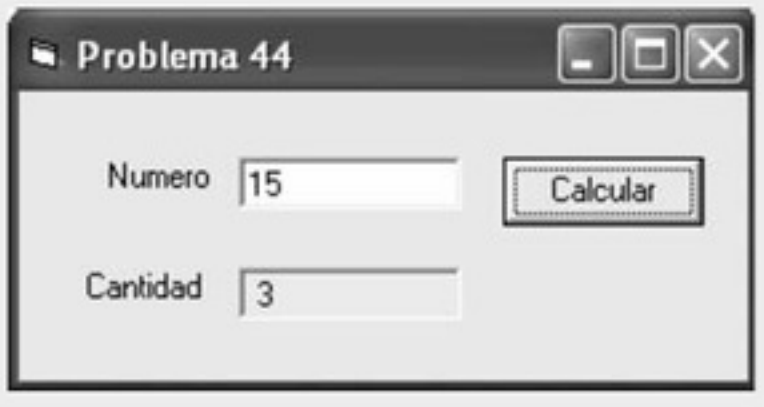
**Enunciado**: Obtener la cantidad de los primeros N números múltiplos de 5.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva la cantidad de números múltiplos de 5.

| Entrada | Salida |
|---|---|
| • Número (n) | • Cantidad (c) |

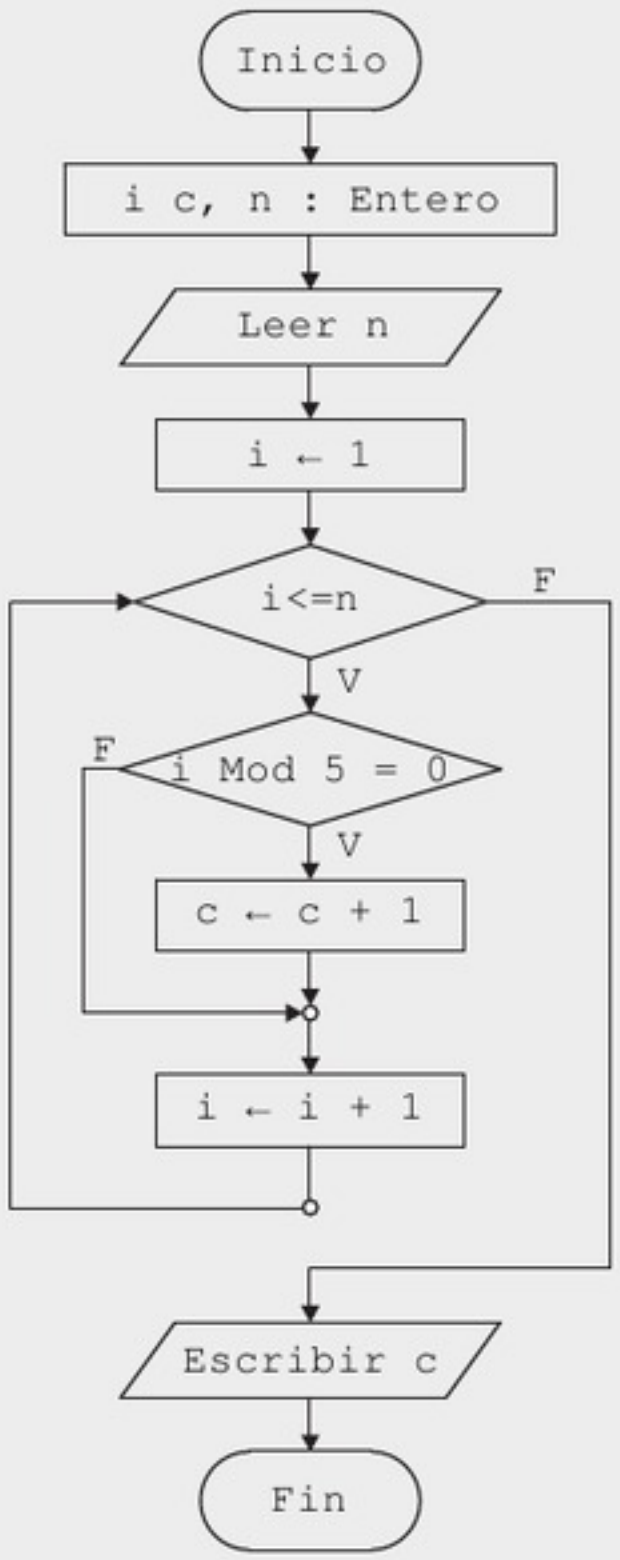
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, c, n : Entero

   //Entrada
   Leer n

   //Proceso
   i ← 1
   Mientras i <= n
       Si i Mod 5 = 0 Entonces
              c ← c + 1
       Fin Si
       i ← i + 1
   Fin Mientras

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim n As Long
Dim c As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
i = 1
Do While i <= n
     If i Mod 5 = 0 Then
            c = c + 1
     End If
     i = i + 1
Loop

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 45

**Enunciado**: Dado un número, determinar cuantos dígitos tiene.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero; luego, que el sistema verifique y determine la cantidad de dígitos que contiene.

**Entrada**

- Número (n)

**Salida**

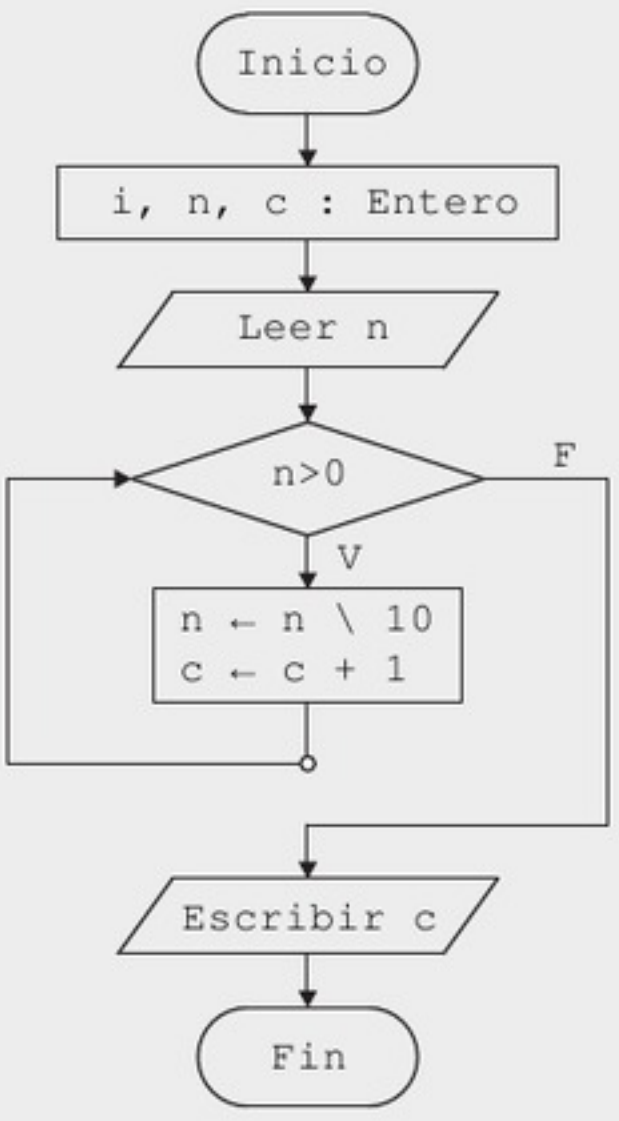
- Cantidad de dígitos (c)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, n, c : Entero

   //Entrada
   Leer n

   //Proceso
   Mientras n>0
       n ← n \ 10
       c ← c + 1
   Fin Mientras

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim n As Long
Dim c As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Do While n > 0
    n = n \ 10
    c = c + 1
Loop

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 46

**Enunciado**: Dado un número, determinar la cantidad de dígitos pares que contiene.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero; luego, que el sistema verifique y devuelva la cantidad de dígitos enteros que contiene el número.

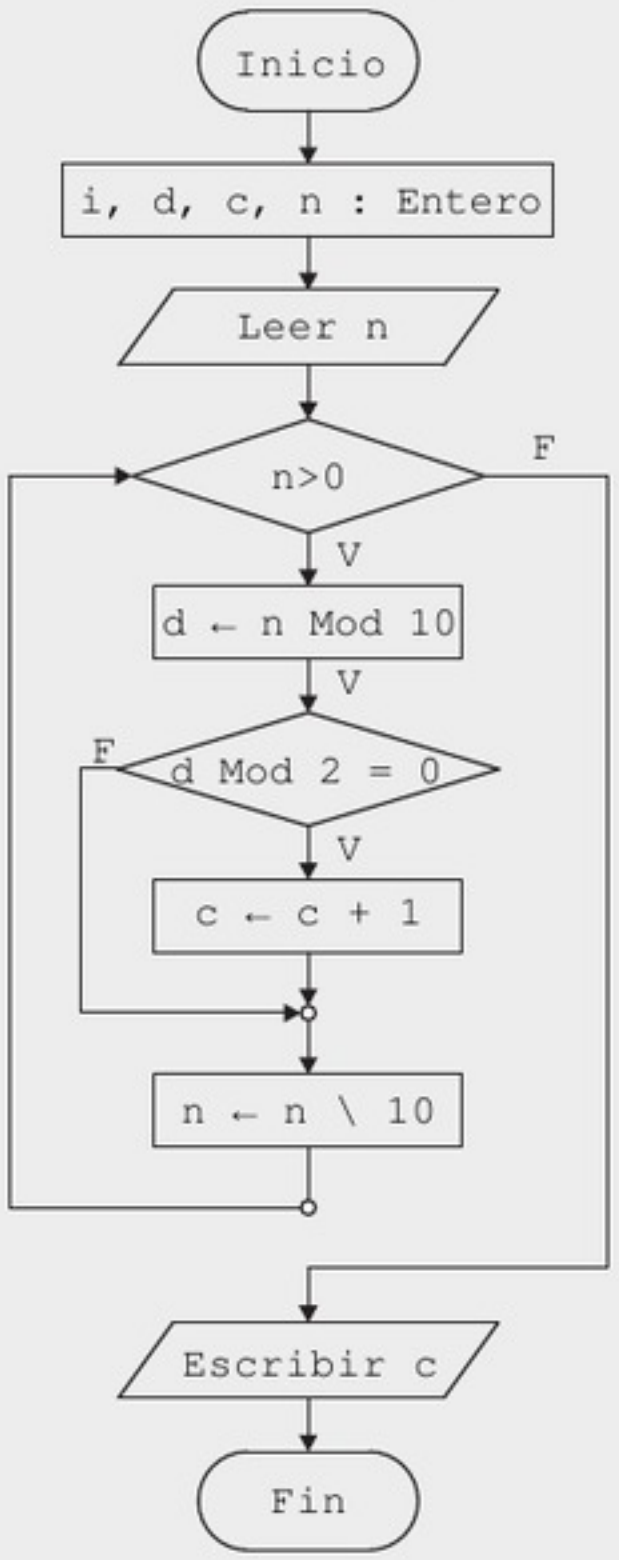
| Entrada | Salida |
|---|---|
| • Números (n) | • Cantidad de dígitos pares (c) |

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, d, c, n : Entero

   //Entrada
   Leer n

   //Proceso
   Mientras n > 0
       d ← n Mod 10
       Si d Mod 2 = 0
   Entonces
                c ← c + 1
       Fin Si
       n ← n \ 10
   Fin Mientras

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim d As Long
Dim c As Long
Dim n As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Do While n > 0
     d = n Mod 10
     If d Mod 2 = 0 Then
          c = c + 1
     End If
     n = n \ 10
Loop

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 47

**Enunciado**: Dado un número, devolver el dígito mayor.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número entero; luego, que el sistema verifique y devuelva el dígito mayor.

**Entrada**

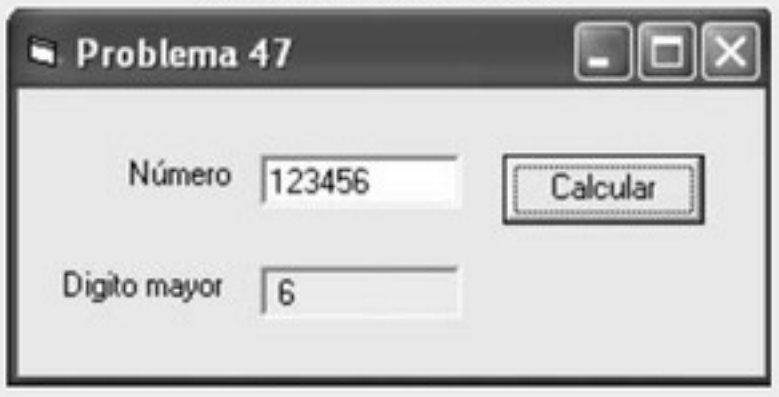
- Número entero (n)

**Salida**

- Dígito mayor (m)

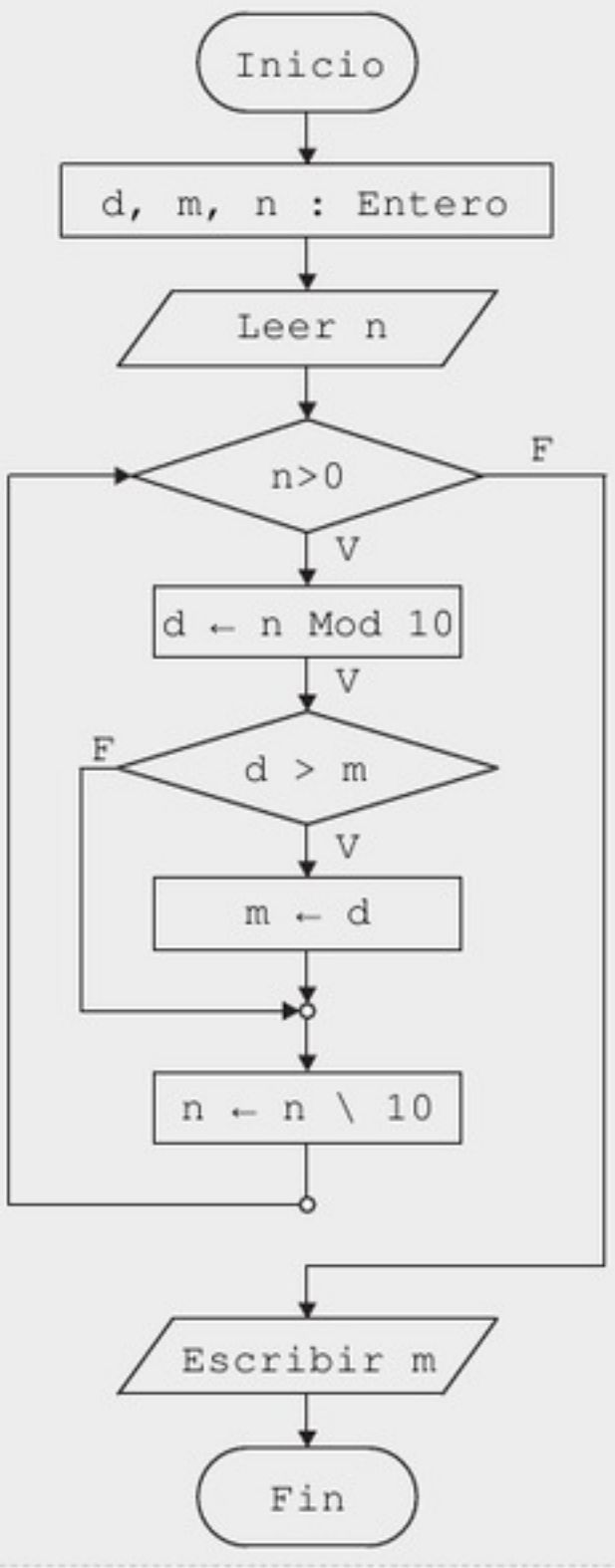
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   d, m, n : Entero

   //Entrada
   Leer n

   //Proceso
   Mientras n > 0
       d ← n Mod 10
       Si d > m Entonces
             m ← d
       Fin Si
       n ← n \ 10
   Fin Mientras

   //Salida
   Escribir m

Fin
```

**Codificación:**

```
'Variables
Dim d As Long
Dim m As Long
Dim n As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Do While n > 0
     d = n Mod 10
     If d > m Then
          m = d
     End If
     n = n \ 10
Loop

'Salida
Me.txtm.Text = Str(m)
```

## Problema n.° 48

**Enunciado**: Dados 2 números, diga si son amigos. Recuerde que dos números son amigos si la suma de sus divisores de uno de ellos es igual al otro y viceversa, por ejemplo, 220 y 284 son amigos.

Divisores de 220 son:

$$1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = \mathbf{284}$$

Divisores de 284 son:

$$1 + 2 + 4 + 71 + 142 = \mathbf{220}$$

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese dos números (n1 y n2); luego, que el sistema verifique y devuelva si son o no número amigos.

**Entrada**

- Números (n1, n2)

**Salida**

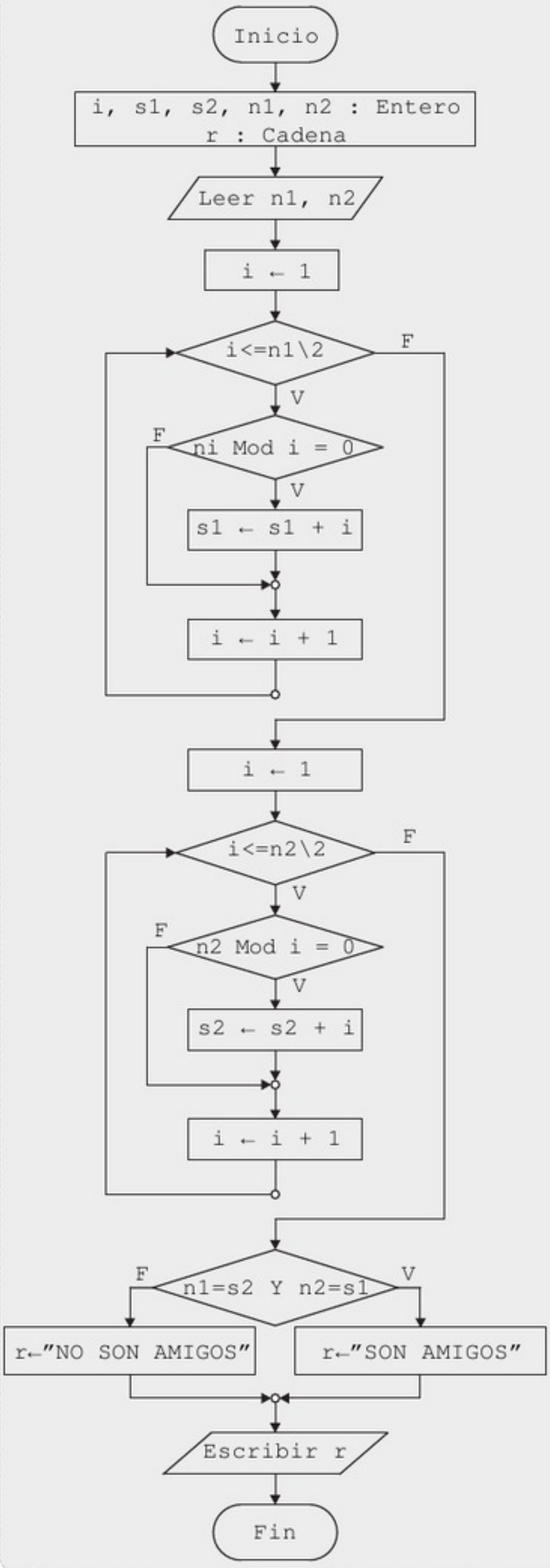
- Respuesta (r)
  - SON AMIGOS
  - NO SON AMIGOS

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, s1, s2, n1, n2 : Entero
   r : Cadena

   //Entrada
   Leer n1, n2

   //Proceso
   i ← 1
   Mientras i <= n1\2
       Si n1 Mod i = 0 Entonces
              s1 ← s1 + i
       Fin Si
       i = i + 1
   Fin Mientras

   i ← 1
   Mientras i <= n2\2
       Si n2 Mod i = 0 Entonces
              s2 ← s2 + i
       Fin Si
       i = i + 1
   Fin Mientras
   Si n1 = s2 Y n2 = n1 Entonces
       r ← "SON AMIGOS"
   SiNo
       r ← "NO SON AMIGOS"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim n1 As Integer
Dim n2 As Integer
Dim s1 As Integer
Dim s2 As Integer
Dim r As String

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
i = 1
Do While i <= n1 \ 2
     If n1 Mod i = 0 Then
          s1 = s1 + i
     End If
     i = i + 1
Loop

i = 1
Do While i <= n2 \ 2
     If n2 Mod i = 0 Then
          s2 = s2 + i
     End If
     i = i + 1
Loop

If n1 = s2 And n2 = s1 Then
     r = "SON AMIGOS"
Else
     r = "NO SON AMIGOS"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 49

**Enunciado**: Dado un número, devuelva el inverso del número.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el número; luego, que el sistema procese y devuelva el inverso del número.

**Entrada**

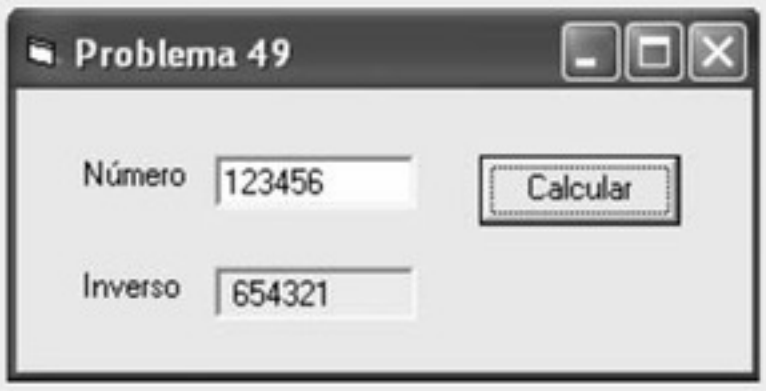
- Número (n)

**Salida**

- Número inverso (i)

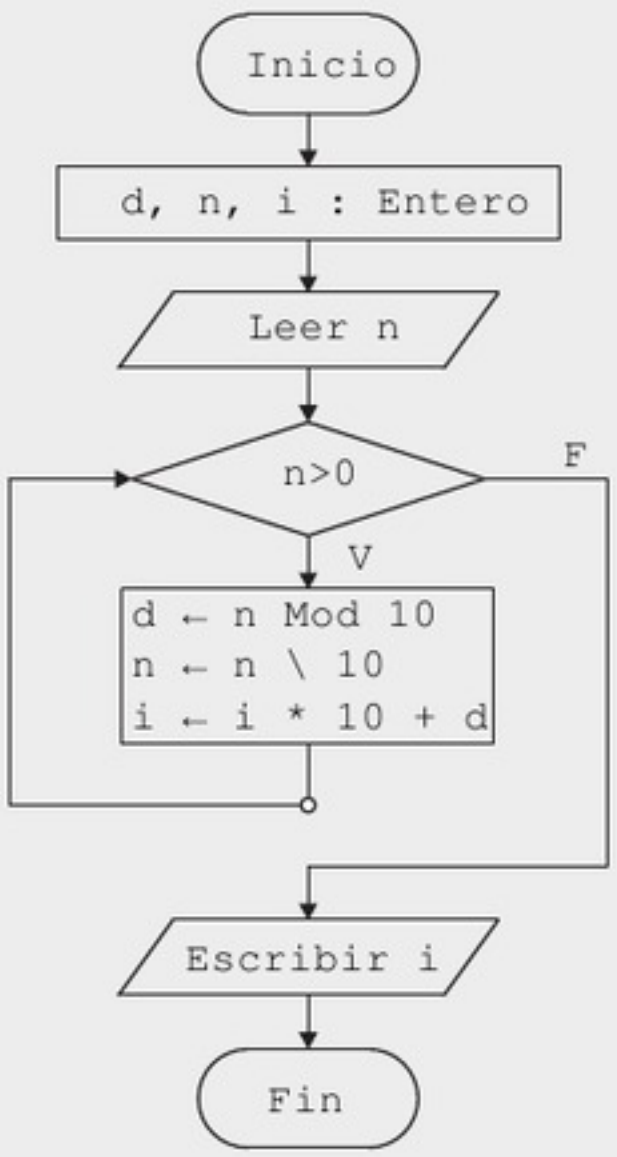
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

//Variables
d, n, i : Entero

//Entrada
Leer n

//Proceso
Mientras n > 0
    d ← n Mod 10
    n ← n \ 10
    i ← i * 10 + d
Fin Mientras

//Salida
Escribir i

Fin
```

**Codificación:**

```
'Variables
Dim d As Long
Dim n As Long
Dim i As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Do While n > 0
    d = n Mod 10
    n = n \ 10
    i = i * 10 + d
Loop

'Salida
Me.txti.Text = Str(i)
```

## Problema n.° 50

**Enunciado**: Crear un algoritmo que indique si un número es cubo perfecto (anstrong) o no; se dice que un número es cubo perfecto si al sumar los cubos de sus dígitos dan el mismo número. Por ejemplo: 153, los cubos de sus dígitos $1^3 + 5^3 + 3^3 = 153$. Por lo tanto, el número 153 es cubo perfecto.

**Análisis**: Para la solución de este problema se requiere que el usuario ingrese el número; luego, que el sistema procese y determine si es o no un cubo perfecto.

**Entrada**

- Número (n)

**Salida**

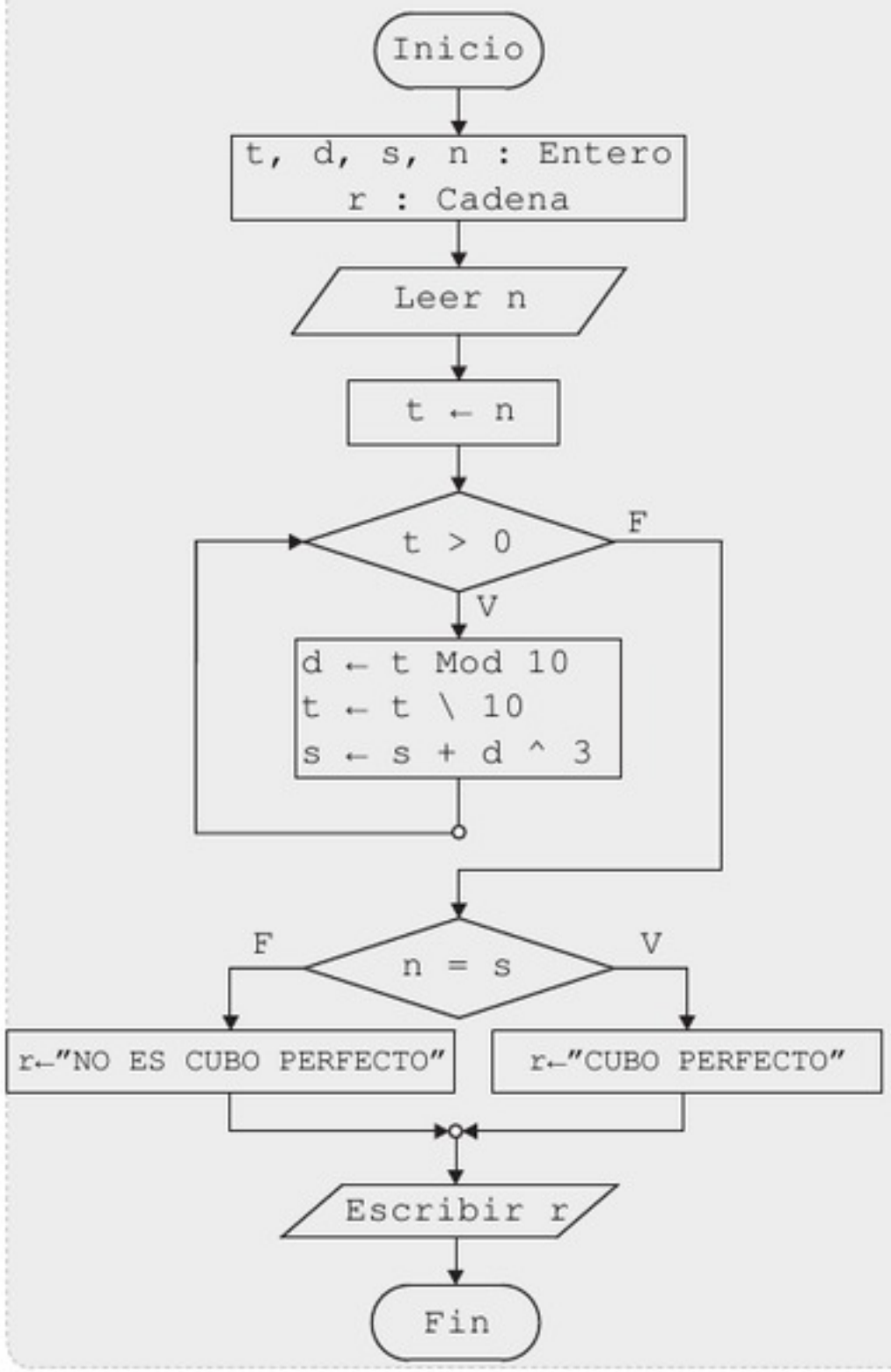
- Respuesta (r)
  - CUBO PERFECTO
  - NO ES CUBO PERFECTO

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   t, d, s, n : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   t ← n
   Mientras t > 0
       d ← t Mod 10
       t ← t \ 10
       s ← s + d ^ 3

   Fin Mientras

   Si n = s Entonces
       r ← "CUBO PERFECTO"
   SiNo
       r ← "NO ES CUBO PERFECTO"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim t As Integer
Dim d As Integer
Dim s  As Integer
Dim n As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
t = n
Do While t > 0
     d = t Mod 10
     t = t \ 10
     s = s + d ^ 3
Loop

If n = s Then
     r = "CUBO PERFECTO"
Else
     r = "NO ES CUBO PERFECTO"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 51

**Enunciado**: Obtenga el cociente y el residuo de una división mediante restas sucesivas, por ejemplo, si el dividendo es 3989 y el divisor es 1247, entonces:

3989 – 1247 = 2742 R(1)

2742 – 1247 = 1495 R(2)

1495 – 1247 = 248 R(3)

Ya no se puede seguir restando, pues 248 es menor a 1247; entonces el cociente es el número de veces restado (3) y el residuo es el último número obtenido (248).

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese la temperatura; luego, que el sistema verifique y determine el clima.

**Entrada**

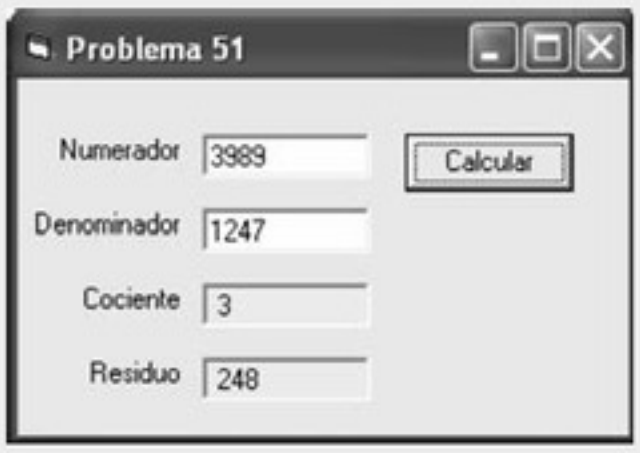
- Numerador (n)
- Denominador (d)

**Salida**

- Cociente (c)
- Residuo (r)

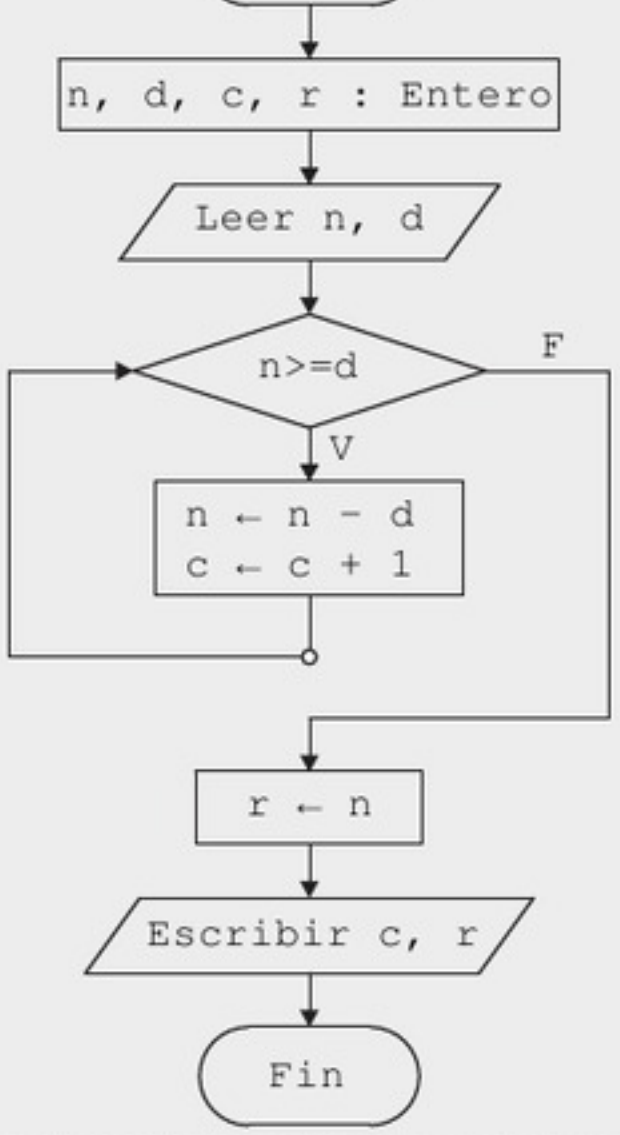
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, d, c, r : Entero

   //Entrada
   Leer n, d

   //Proceso
   Mientras n >= d
       n ← n - d
       c ← c + 1
   Fin Mientras
   r ← n

   //Salida
   Escribir c, r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim d As Integer
Dim c As Integer
Dim r As Integer

'Entrada
n = Val(Me.txtn.Text)
d = Val(Me.txtd.Text)

'Proceso
Do While n >= d
     n = n - d
     c = c + 1
Loop
r = n

'Salida
Me.txtc.Text = Str(c)
Me.txtr.Text = Str(r)
```

## Problema n.° 52

**Enunciado**: Determine si un número es capicúa o no. Se dice que un número capicúa es aquel cuyas cifras, al ser invertidas, dan el mismo número. Por ejemplo, 12 321 invertido es 12 321, entonces es un número capicúa.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema verifique y determine si es o no capicúa.

**Entrada**

- Número (n)

**Salida**

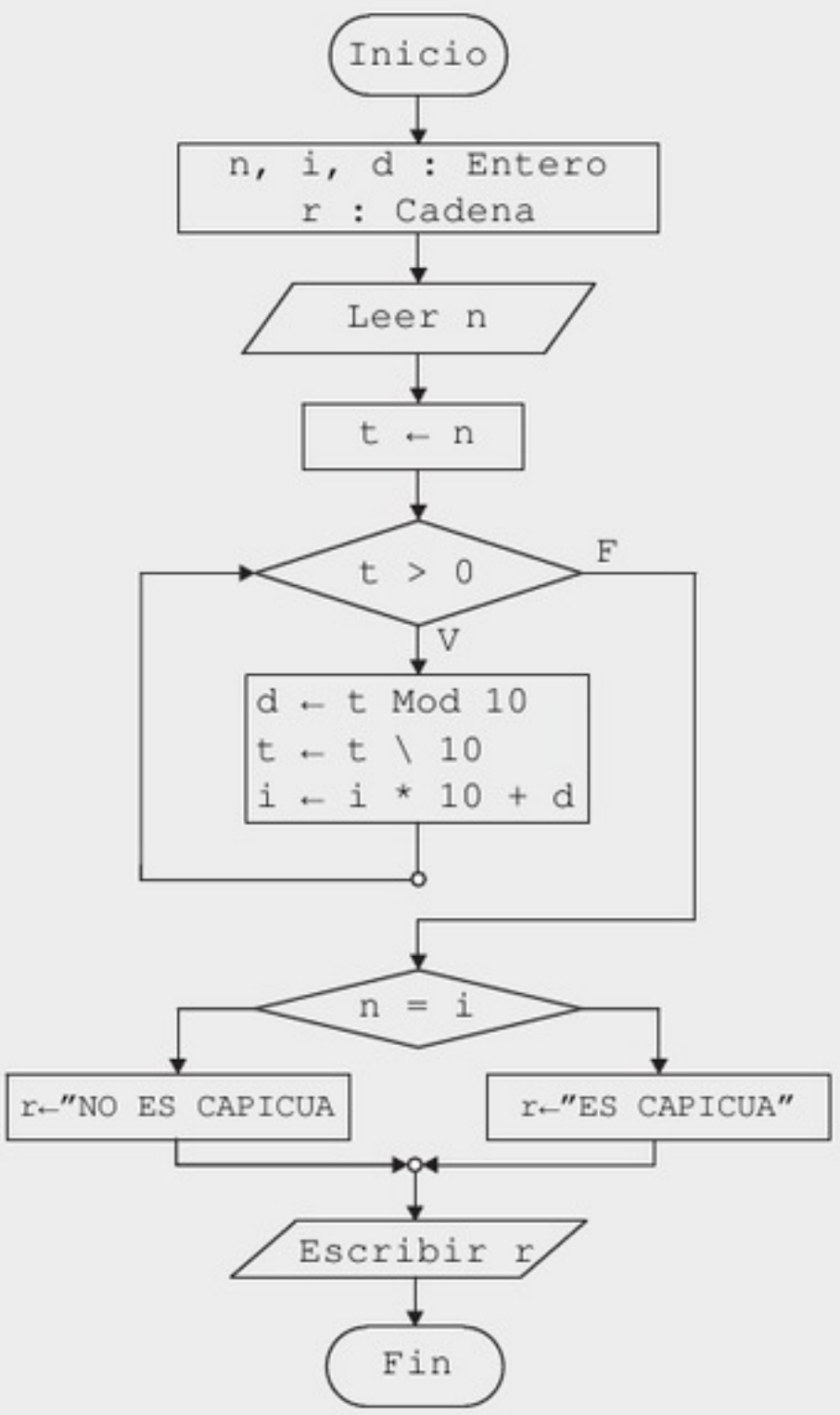
- Respuesta (r)
  - ES CAPICUA
  - NO ES CAPICUA

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, i, d : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   t ← n
   Mientras t > 0
       d ← t Mod 10
       t ← t \ 10
       i ← i * 10 + d

   Fin Mientras

   Si n = i Entonces
       r ← "ES CAPICUA"
   SiNo
       r ← "NO ES CAPICUA"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim i As Integer
Dim d As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
t = n
Do While t > 0
    d = t Mod 10
    t = t \ 10
    i = i * 10 + d
Loop

If n = i Then
    r = "ES CAPICUA"
Else
    r = "NO ES CAPICUA"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 53

**Enunciado**: Dado un número, determine si es primo, recuerde que un número primo es aquel que solo es divisible por 1 y por sí mismo.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema determine si es primo o no.

**Entrada**

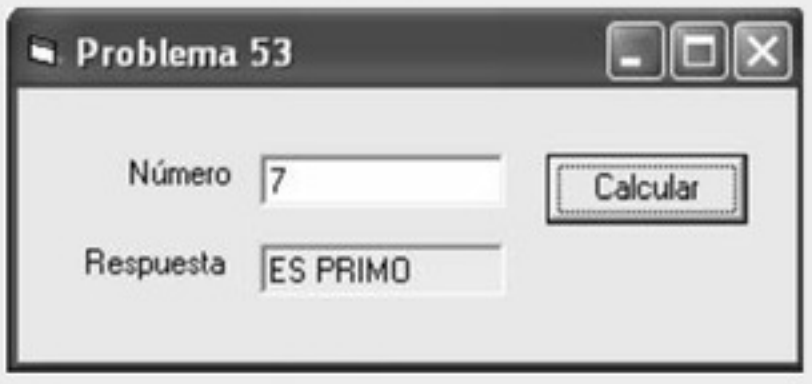
- Número (n)

**Salida**

- Respuesta (r)
  - ES PRIMO
  - NO ES PRIMO

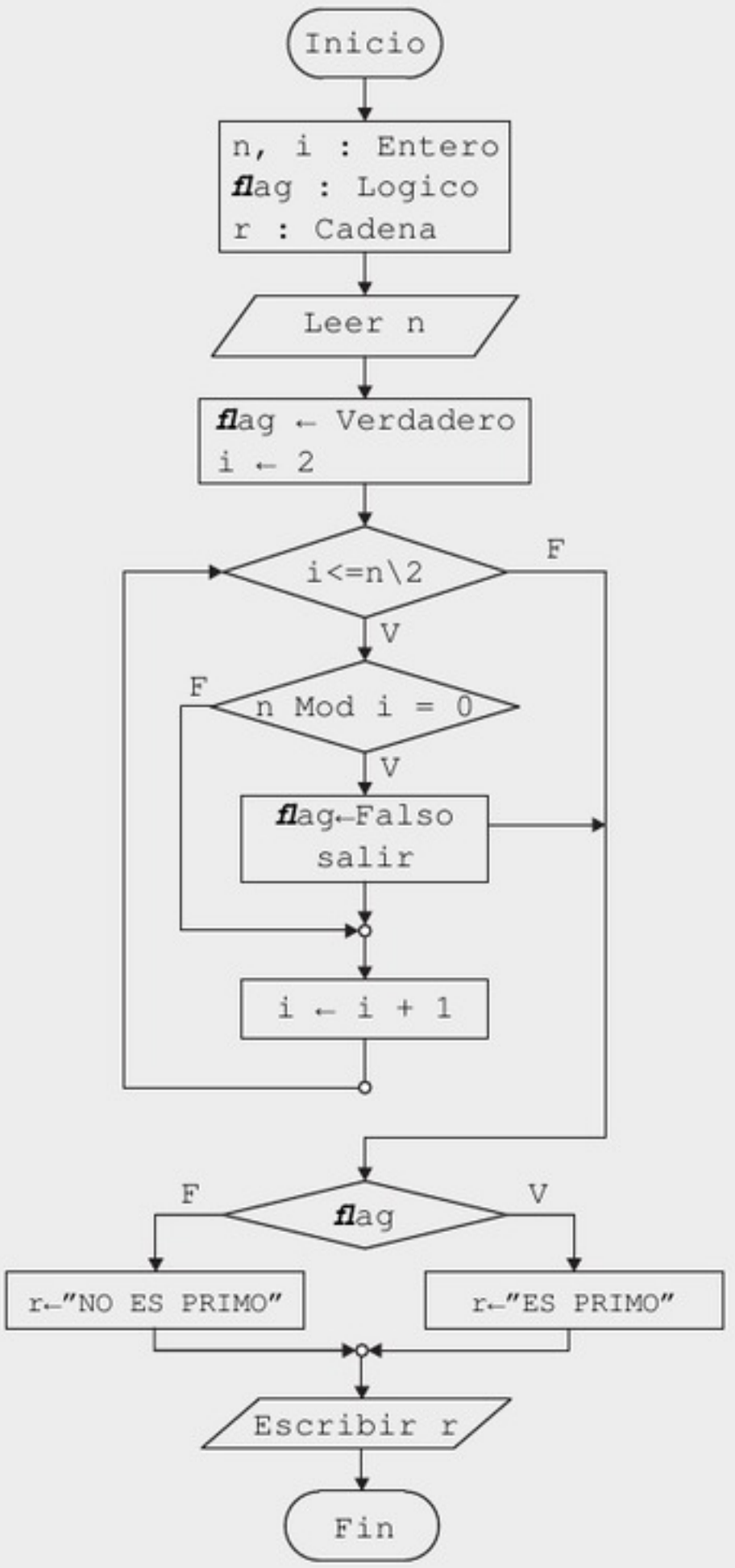
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, i : Entero
   flag : Logico
   r : Cadena

   //Entrada
   Leer n
   //Proceso
   flag ← Verdadero
   i ← 2
   Mientras i <= n\2
       Si n Mod i = 0
              flag ← Falso
              Salir
       Fin Si
       i ← i + 1
   Fin Mientras

   Si flag Entonces
       r ← "ES PRIMO"
   SiNo
       r ← "NO ES PRIMO"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim i As Integer
Dim flag As Boolean
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
flag = True
i = 2
Do While i <= n \ 2
    If n Mod i = 0 Then
        flag = False
        Exit Do
    End If
    i = i + 1
Loop

If flag Then
    r = "ES PRIMO"
Else
    r = "NO ES PRIMO"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 54

**Enunciado**: Dado un número y su base, determine si el número pertenece a la base ingresada. Recuerde que un número pertenece a un base si sus dígitos son menores a su base.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema verifique y determine si pertenece a la base o no.

**Entrada**

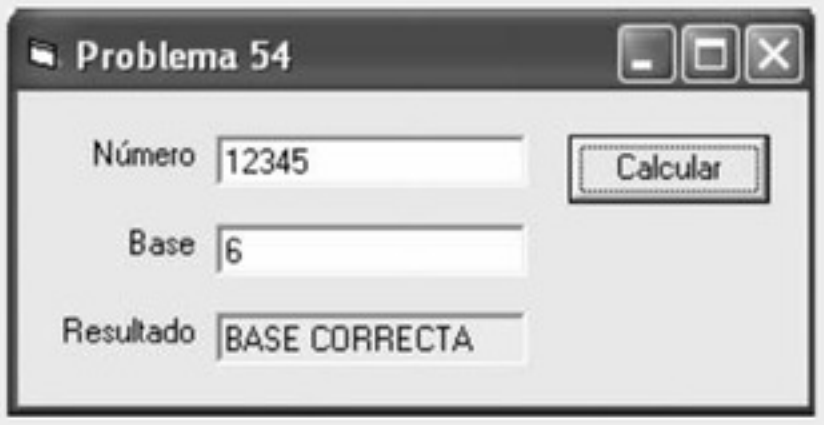
- Número (n)
- Base (b)

**Salida**

- Respuesta (r)
  - BASE CORRECTA
  - BASE INCORRECTA

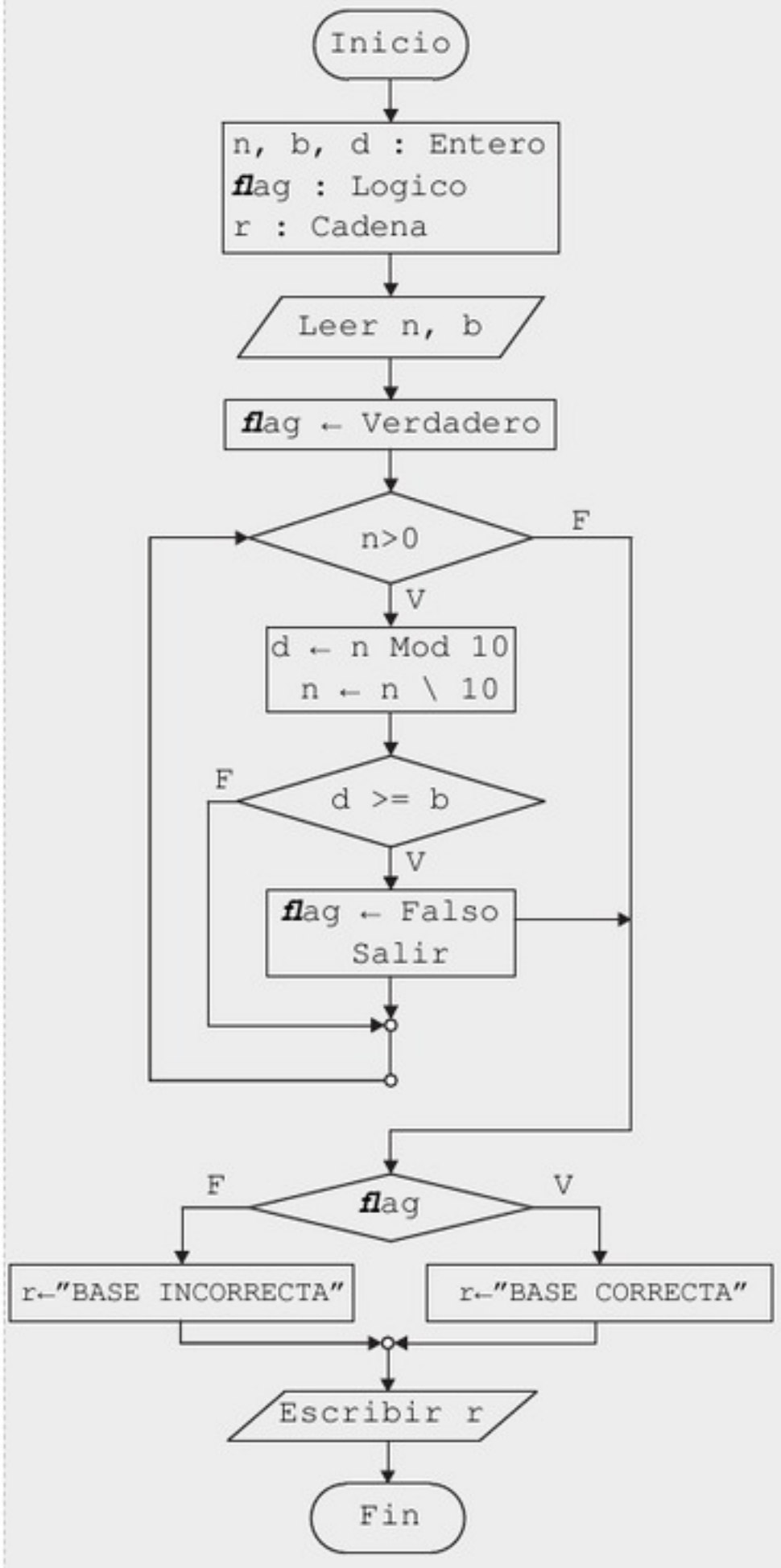
**Diseño:**

**Interfaz de usuario**




## Algoritmo

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, b, d : Entero
   flag : Logico
   r : Cadena

   //Entrada
   Leer n, b

   //Proceso
   flag ← Verdadero
   Mientras n > 0
       d ← n Mod 10
       n ← n \ 10
       Si d >= b Entonces
              flag ← Falso
              Salir
       Fin Si
   Fin Mientras

   Si flag Entonces
              r ← "BASE CORRECTA"
   SiNo
              r ← "BASE INCORRECTA"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Long
Dim b As Long
Dim d As Long
Dim flag As Boolean
Dim r As String

'Entrada
n = Val(Me.txtn.Text)
b = Val(Me.txtb.Text)

'Proceso
flag = True
Do While n > 0
    d = n Mod 10
    n = n \ 10
    If d >= b Then
        flag = False
        Exit Do
    End If
Loop
If flag Then
    r = "BASE CORRECTA"
Else
    r = "BASE INCORRECTA"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 55

**Enunciado**: Dado un número entero en base 10, convertir el número a otra base menor que 10.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese el número entero y la base a convertir; luego, que el sistema devuelva el número convertido a su nueva base.

**Entrada**

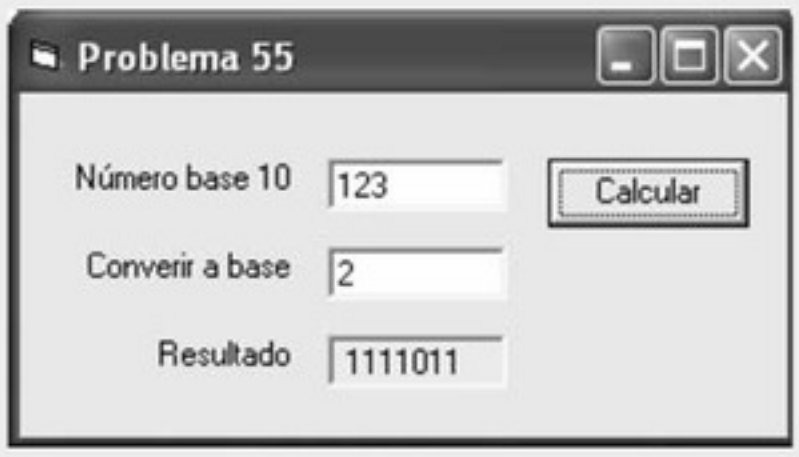
- Número (n)
- Base (b)

**Salida**

- Número convertido (r)

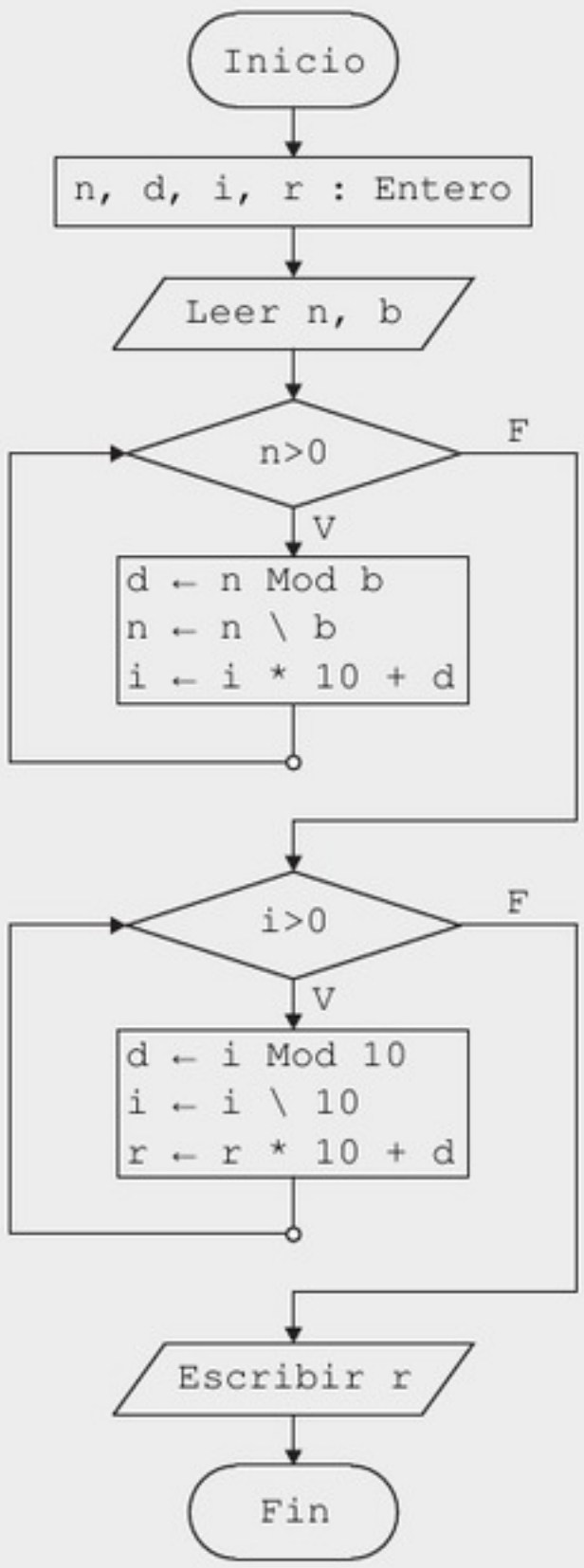
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, d, i, r : Entero

   //Entrada
   Leer n, b

   //Proceso
   Mientras n > 0
       d ← n Mod b
       n = n \ 10
       i = i * 10 + d
   Fin Mientras

   Mientras i > 0
       d ← i Mod 10
       i = i \ 10
       r = r * 10 + d
   Fin Mientras

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Long
Dim d As Long
Dim i As Long
Dim r As Long

'Entrada
n = Val(Me.txtn.Text)
b = Val(Me.txtb.Text)

'Proceso
Do While n > 0
    d = n Mod b
    n = n \ b
    i = i * 10 + d
Loop
Do While i > 0
    d = i Mod 10
    i = i \ 10
    r = r * 10 + d
Loop

'Salida
Me.txtr.Text = Str(r)
```

## 5.8 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 31

**Enunciado**: Obtener el factorial de un número; recuerde que el factorial de un número es el producto de 1 x 2 x 3 x ... x N.

### Propuesto n.° 32

**Enunciado**: Dado un rango de números enteros, obtener la cantidad de números pares e impares que contiene el rango, sin considerar los múltiplos de 5.

### Propuesto n.° 33

**Enunciado**: Calcular la suma y el producto de los N primeros números naturales múltiplos de 3.

### Propuesto n.° 34

**Enunciado**: Dado un número, determinar cuantos dígitos «0» contiene.

### Propuesto n.° 35

**Enunciado**: Se requiere saber si existe un determinado dígito en un número dado.

### Propuesto n.° 36

**Enunciado**: Dado un número, determinar cual es el porcentaje de números pares, impares y neutros (0).

### Propuesto n.° 37

**Enunciado**: Dado un rango de números, determine cuántos números primos contiene.

### Propuesto n.° 38

**Enunciado**: Dado un rango de números, determine cuántos números capicúa hay.

### Propuesto n.° 39

**Enunciado**: Dados 2 números, obtener el MCD (máximo común divisor); utilice el método de Euclides (divisiones sucesivas).

### Propuesto n.° 40

**Enunciado**: Dados 2 números, obtener el MCD (máximo común divisor), utilice el método factorización simultánea.

**Recuerde:** El máximo común divisor es el divisor mayor común de todos ellos.

# Capítulo 6

# Estructura repetitiva «Para»

## 6.1 Introducción

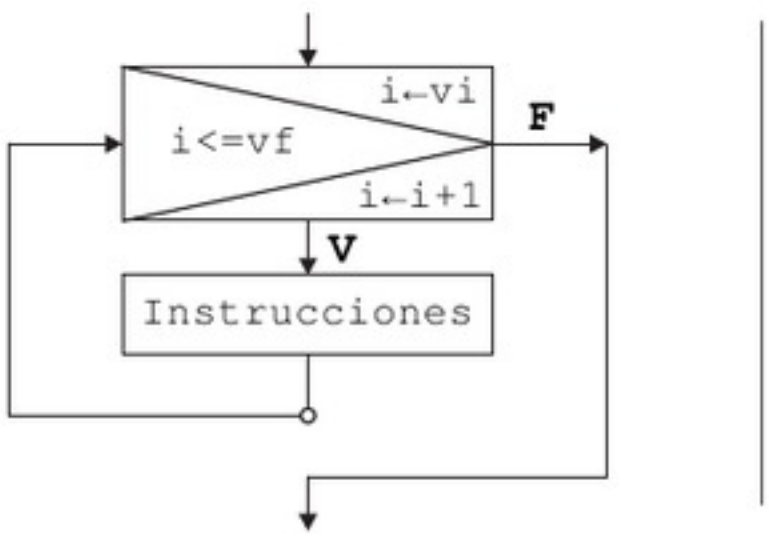
Cada vez que requiere repetir un proceso una determinada cantidad de veces, deberá usar la estructura repetitiva «**Para**» (for), que permitirá realizar en forma simple este trabajo.

Esta estructura usa una variable «**contador**», donde se establece el valor inicial (**vi**), valor final (**vf**) y el valor de incremento (**inc**), que determina las veces a repetir la instrucción

## 6.2 Estructura repetitiva «Para»

Permite repetir una o más instrucciones una cantidad de veces.

- **i** Es nuestra variable contador, donde establecemos el valor inicial.
- **vf** Representa el valor final de la variable contador.
- **+1** Valor de incremento.



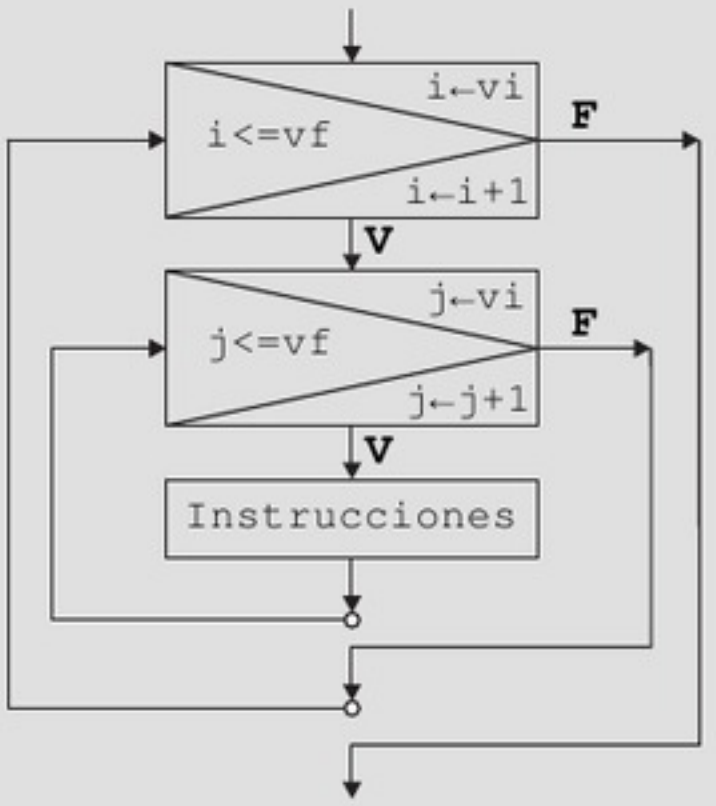

**Sintaxis Visual Basic**

```
For i=vi To vf Step 1
      <instrucciones>
Next
```

La palabra **Step** es opcional; por defecto, en Visual Basic, su valor de incremento es de 1.

## 6.3 Estructura repetitiva «Para» anidada

Dentro de la estructura repetitiva es posible colocar una o más estructuras repetitivas, y también otras estructuras.




```
Para i ← vi Hasta vf Inc +1
      Para j ← vi Hasta vf Inc +1
            Instrucciones
      Fin Para
Fin Para
```

**Sintaxis Visual Basic**

```
For i=vi To vf
      For j=vi To vf
            <instrucciones>
      Next
Next
```

### Problema n.° 56

**Enunciado**: Obtener la suma de los primeros N números naturales positivos.

**Análisis**: Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema realice el proceso para devolver la suma de los N primeros números.

**Entrada**

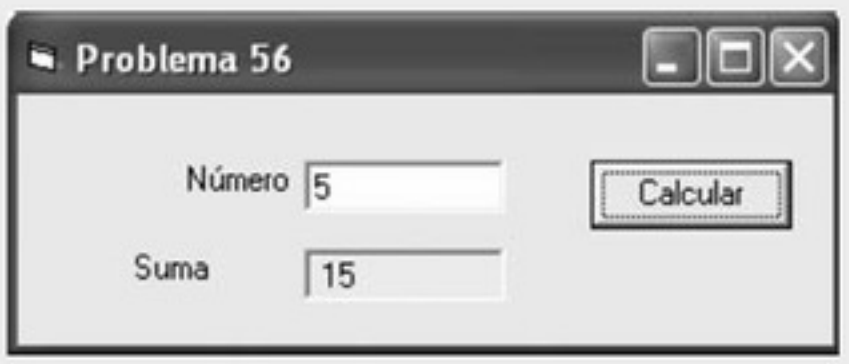
- Número (n)

**Salida**

- Suma (s)

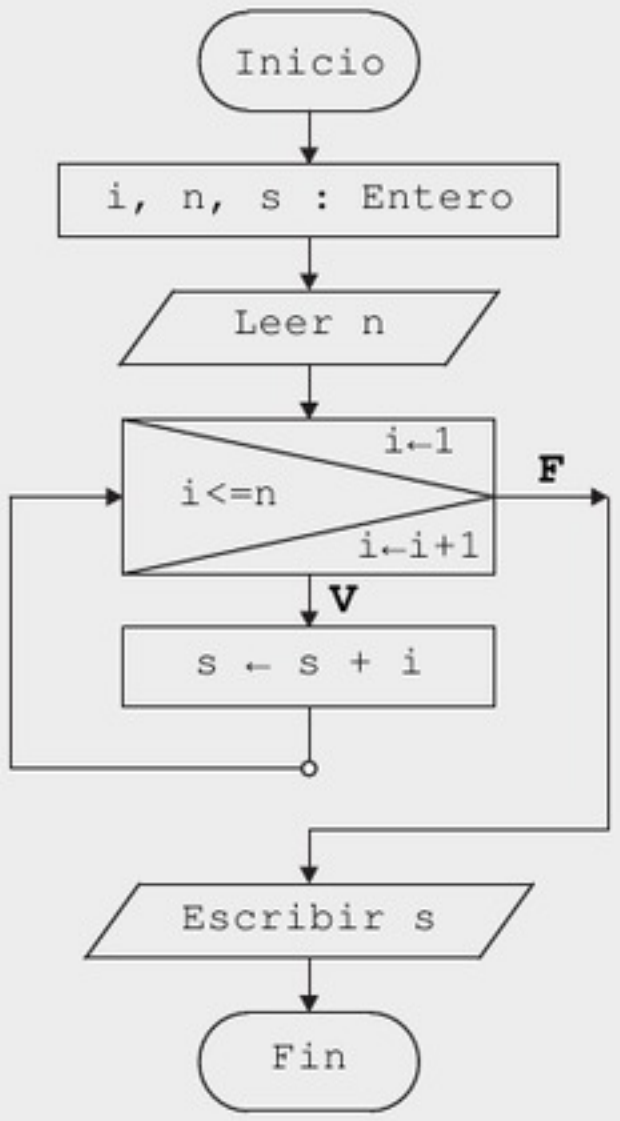
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, n, s : Entero

   //Entrada
   Leer n

   //Proceso
   Para i←1 Hasta n Inc 1
       s ← s + i
   Fin Para

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim n As Integer
Dim s As Integer

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i = 1 To n
     s = s + i
Next

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 57

**Enunciado:** Dado un rango de números enteros, obtener la cantidad de números enteros que contiene.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese el número inicial y final; luego, que el sistema procese y devuelva la cantidad de números enteros que contiene el rango.

**Entrada**

- Número inicial (ni)
- Número final (nf)

**Salida**

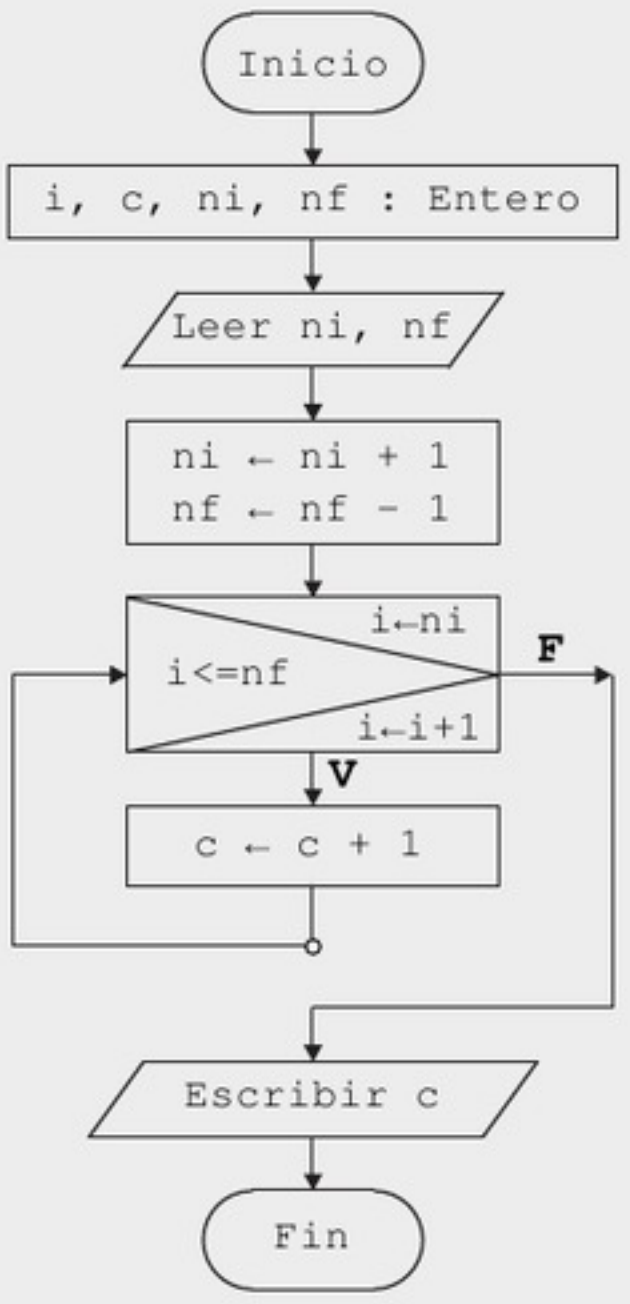
- Cantidad (c)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, c, ni, nf : Entero

   //Entrada
   Leer ni, nf

   //Proceso
   ni ← ni + 1
   nf ← nf - 1
   Para i←ni Hasta nf Inc 1
       c ← c + 1
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim ni As Integer
Dim nf As Integer
Dim c As Integer

'Entrada
ni = Val(Me.txtni.Text)
nf = Val(Me.txtnf.Text)

'Proceso
ni = ni + 1
nf = nf - 1
For i = ni To nf
    c = c + 1
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 58

**Enunciado:** Dado un rango de números enteros, obtener la cantidad de números pares que contiene.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese el número inicial y final; luego, que el sistema procese y devuelva la cantidad números pares que contiene el rango.

**Entrada**

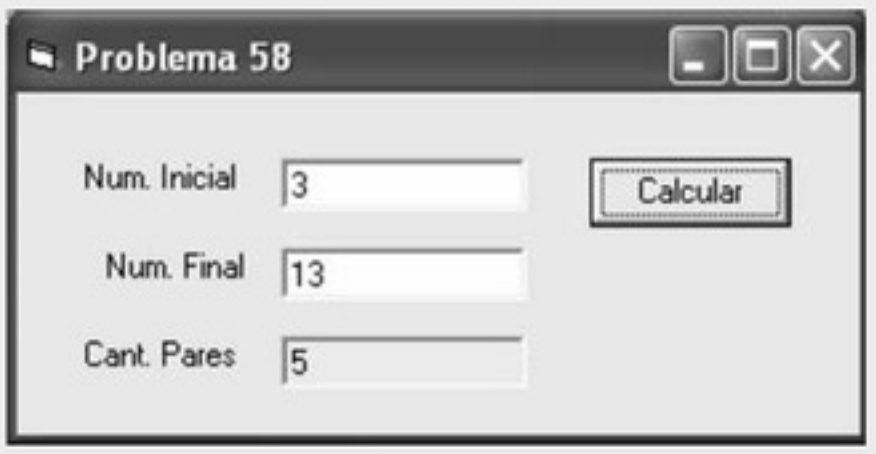
- Número inicial (ni)
- Número final (nf)

**Salida**

- Cantidad de pares (cp)

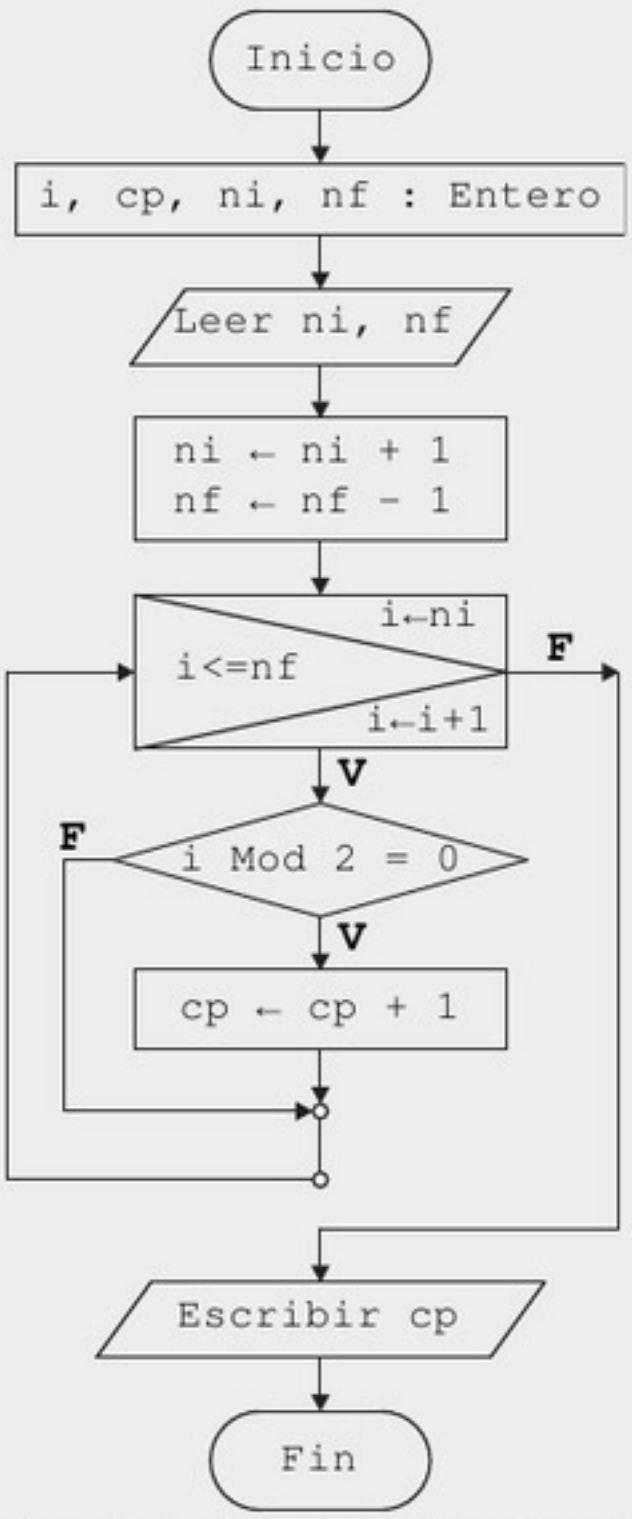
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, cp, ni, nf : Entero

   //Entrada
   Leer ni, nf

   //Proceso
   ni ← ni + 1
   nf ← nf - 1
   Para i←ni Hasta nf Inc 1
       Si i Mod 2 = 0 Entonces
              cp ← cp + 1
       Fin Si
   Fin Para

   //Salida
   Escribir cp

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim ni As Long
Dim nf As Long
Dim cp As Long

'Entrada
ni = Val(Me.txtni.Text)
nf = Val(Me.txtnf.Text)

'Proceso
ni = ni + 1
nf = nf - 1
For i=ni To nf
     If i Mod 2 = 0 Then
           cp = cp + 1
     End If
Next

'Salida
Me.txtcp.Text = cp
```

## Problema n.° 59

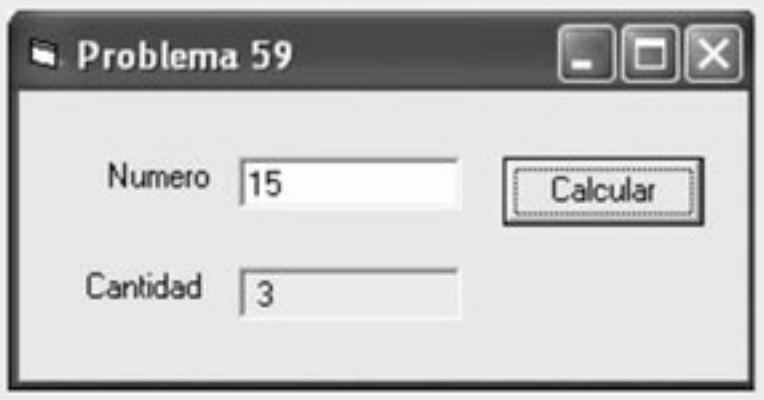
**Enunciado:** Obtener la cantidad de los primeros N números múltiplos de 5.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva la cantidad de números múltiplos de 5.

| Entrada | Salida |
|---|---|
| • Número (n) | • Cantidad (c) |

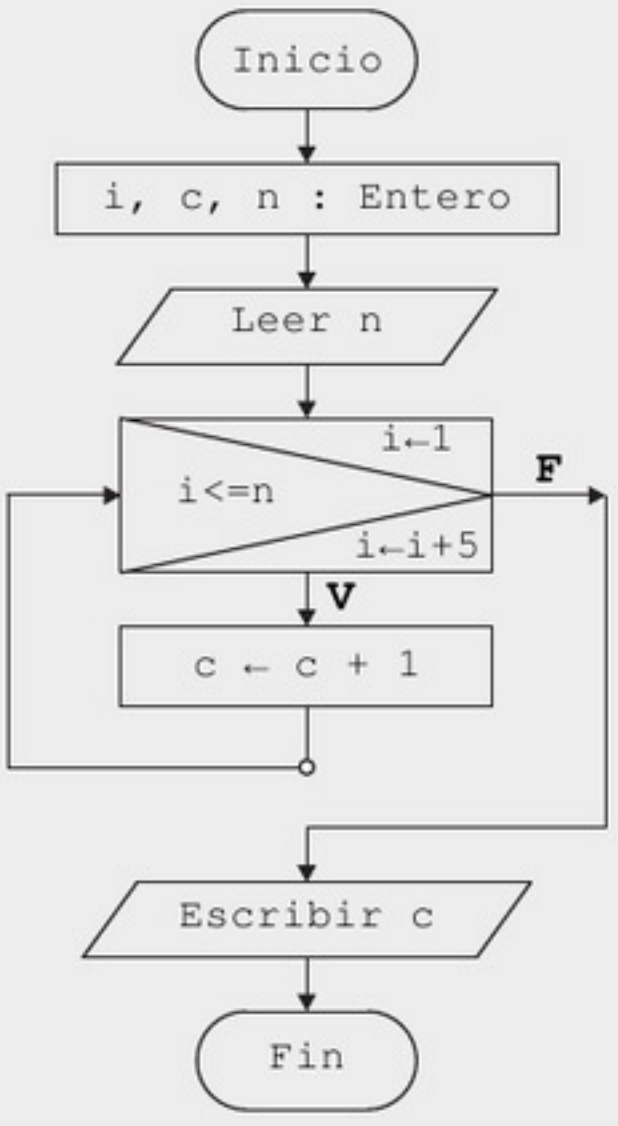
**Diseño:**

**Interfaz de usuario**




### Algoritmo

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, c, n : Entero

   //Entrada
   Leer n

   //Proceso
   Para i←1 Hasta n Inc 5
       c ← c + 1
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim n As Long
Dim c As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i=1 To n Step 5
    c = c + 1
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 60

**Enunciado:** Obtener la suma de pares e impares de los primeros N números enteros positivos.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva la suma de pares e impares.

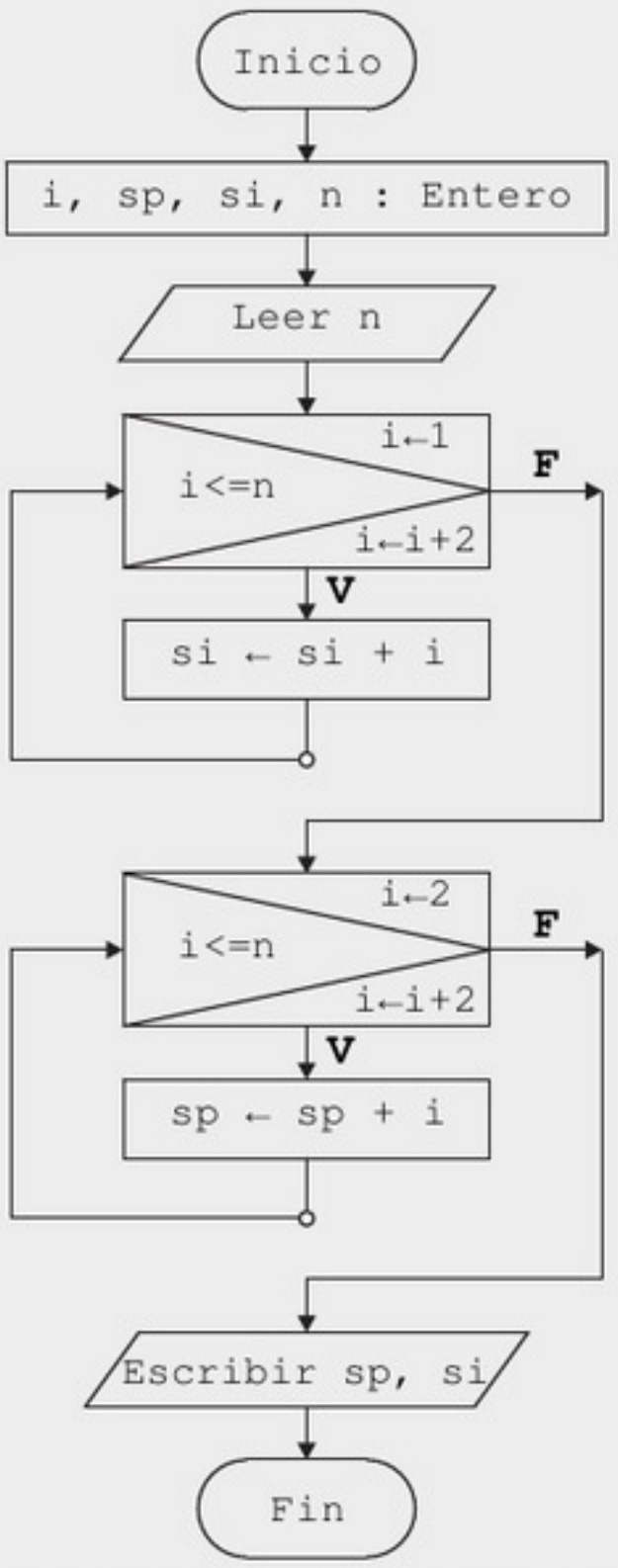
| Entrada | Salida |
|---|---|
| • Número (n) | • Suma pares (sp)<br>• Suma impares (si) |

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, sp, si, n : Entero

   //Entrada
   Leer n

   //Proceso
   Para i←1 Hasta n Inc 2
       si ← si + i
   Fin Para

   Para i←2 Hasta n Inc 2
       sp ← sp + i
   Fin Para

   //Salida
   Escribir sp, si

Fin
```

**Codificación:**

```
 'Variables
Dim i As Long
Dim sp As Long
Dim si As Long
Dim n As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i=1 To n Step 2
     si = si + 1
Next

For i=2 To n Step 2
     sp = sp + 1
Next

'Salida
Me.txtsp.Text = Str(sp)
Me.txtsi.Text = Str(si)
```

## Problema n.° 61

**Enunciado:** Hallar el cuadrado de un número usando la siguiente relación: $N^2 = 1 + 3 + 5 + \ldots\ldots + 2N-1$.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelve el cuadrado del número.

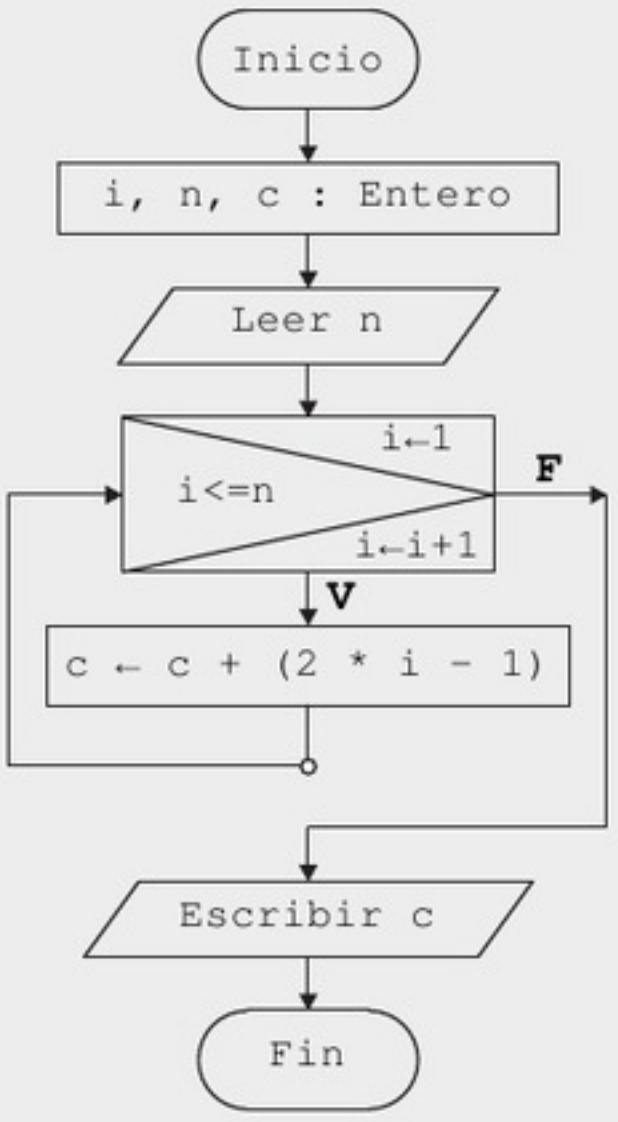
| Entrada | Salida |
|---|---|
| • Número (n) | • Cuadrado (c) |

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, n, c : Entero

   //Entrada
   Leer n

   //Proceso
   Para i←1 Hasta n Inc 1
       c ← c + (2 * i - 1)
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim i As Long
Dim c As Long
Dim n As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i=1 To n Step 1
    c = c + (2 * i - 1)
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 62

**Enunciado:** Crear el algoritmo que indique si un número es perfecto o no; se dice que un número es perfecto si la suma de sus divisores es igual al número. Por ejemplo, 6 tiene como divisores 1, 2 y 3; entonces 1 + 2 + 3 = 6, el número 6 es perfecto. Si el número es 9, tiene como divisores 1 y 3; entonces 1 + 3 = 4, por lo tanto no es perfecto.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva si el número es o no perfecto.

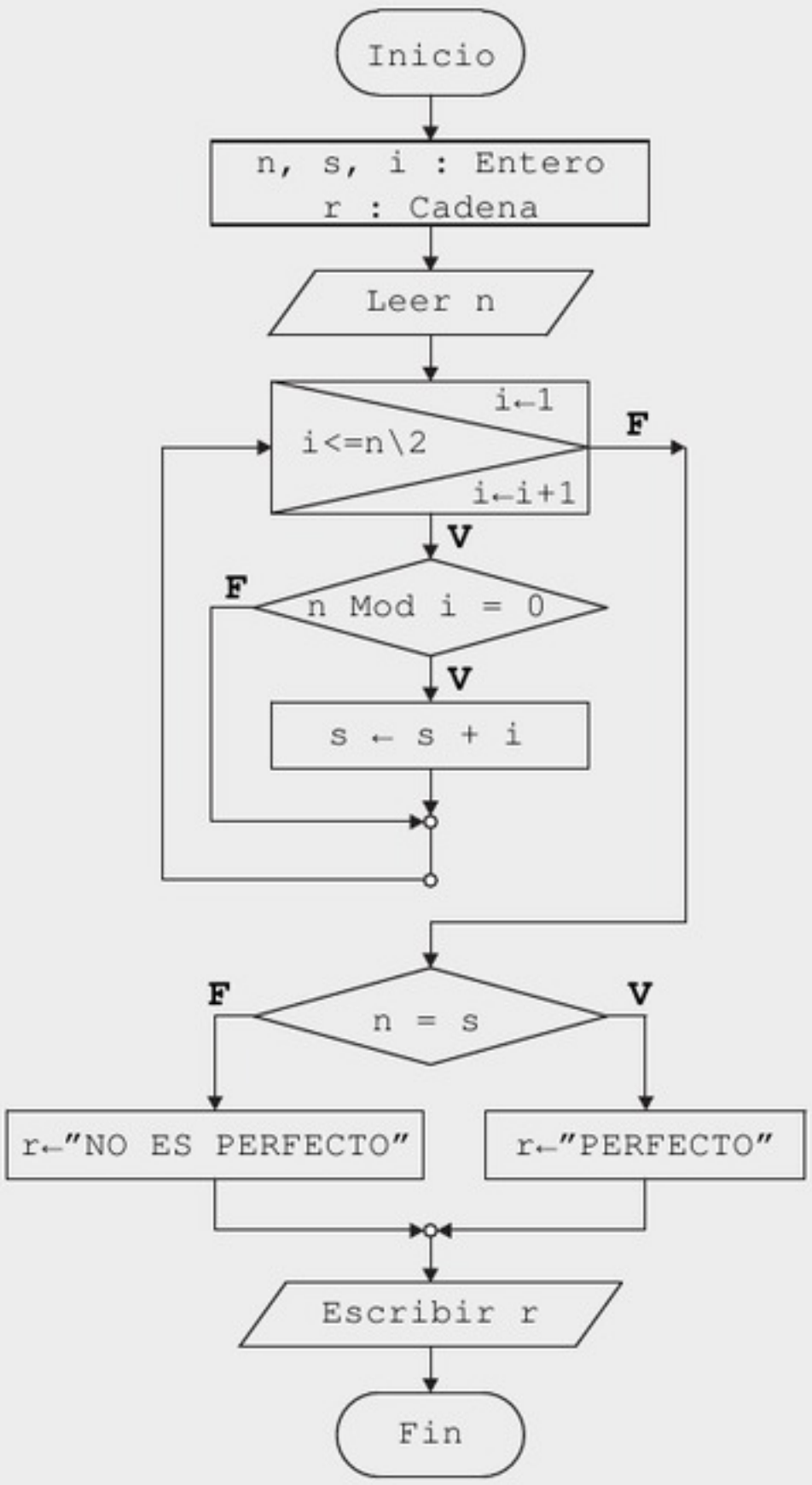
| Entrada | Salida |
|---|---|
| • Número (n) | • Respuesta (r) |

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, s, i : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   Para i←1 Hasta n\2 Inc 1
       Si n Mod i = 0 Entonces
              s ← s + i
       Fin Si
   Fin Para

   Si n = s Entonces
       r ← "PERFECTO"
   SiNo
       r ← "NO ES PERFECTO"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim s As Integer
Dim i As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i = 1 To n \ 2
     If n Mod i = 0 Then
          s = s + i
     End If
Next

If n = s Then
     r = "PERFECTO"
Else
     r = "NO ES PERFECTO"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 63

**Enunciado:** Dados 2 números, diga si son amigos o no. Recuerde que dos números son amigos si la suma de los divisores de uno de ellos es igual al otro y viceversa, por ejemplo 220 y 284 son amigos.

Divisores de 220 son: 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284

Divisores de 284 son: 1 + 2 + 4 + 71 + 142 = 220

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese dos números; luego, que el sistema devuelva el resultado para saber si los números son amigos o no.

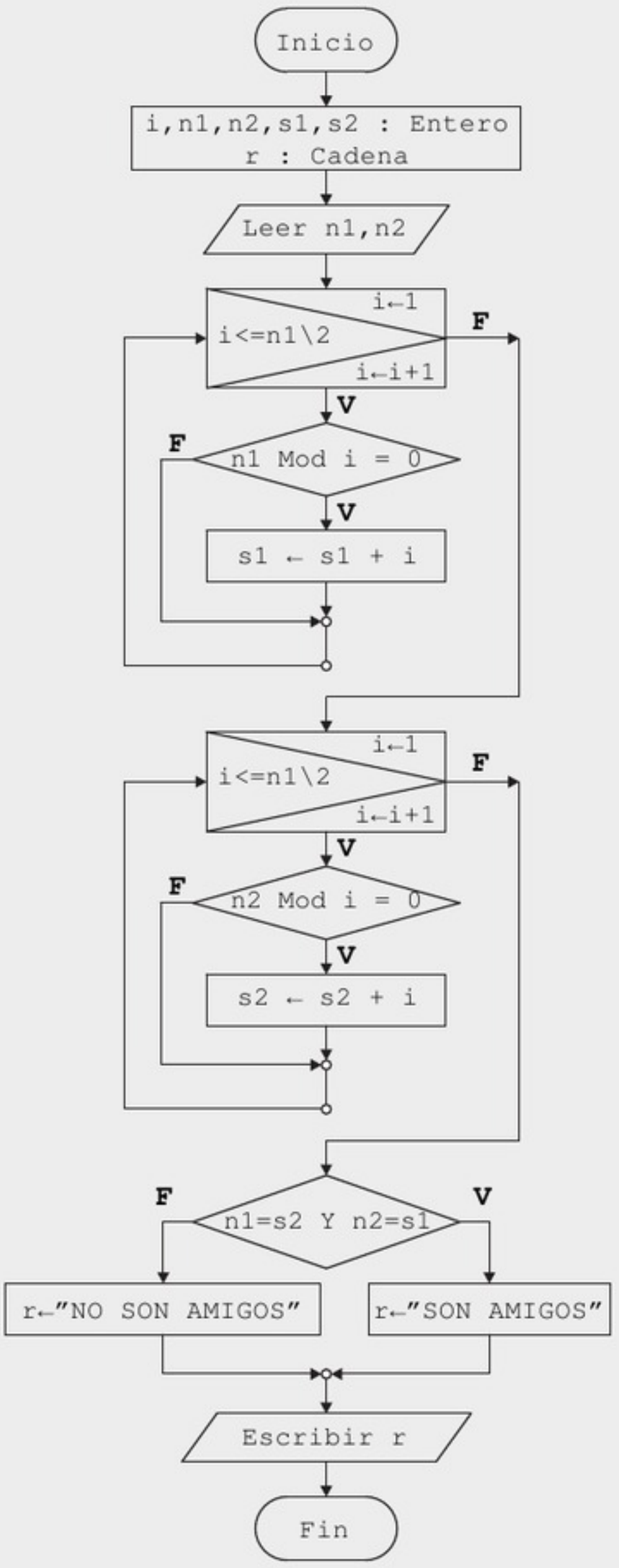
| Entrada | Salida |
|---|---|
| • Números (n1, n2) | • Resultado (r) |

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i,n1,n2,s1,s2 : Entero
   r : Cadena

   //Entrada
   Leer n1,n2

   //Proceso
Para i←1 Hasta n\2 Inc 1
    Si n1 Mod i = 0 Entonces
          s1 ← s1 + i
    Fin Si
Fin Para

Para i←1 Hasta n\2 Inc 1
    Si n2 Mod i = 0 Entonces
          s2 ← s2 + i
    Fin Si
Fin Para

Si n1=s2 Y n2=s1 Entonces
    r ← "SON AMIGOS"
SiNo
    r ← "NO SON AMIGOS"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim n1 As Integer
Dim n2 As Integer
Dim s1 As Integer
Dim s2 As Integer
Dim r As String

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
For i = 1 To n1 \ 2
    If n1 Mod i = 0 Then
        s1 = s1 + i
    End If
Next

For i = 1 To n2 \ 2
    If n2 Mod i = 0 Then
        s2 = s2 + i
    End If
Next

If n1 = s2 And n2 = s1 Then
    r = "SON AMIGOS"
Else
    r = "NO SON AMIGOS"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 64

**Enunciado:** Escriba un algoritmo que calcule la suma de la siguiente serie, hasta el número entero positivo N ingresado.

$\frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\frac{4}{N}$ , por ejemplo si N es 3 entonces: $\frac{1}{2}+\frac{2}{3}+\frac{7}{6}=1{,}1666667$

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva el resultado de la suma de quebrados.

**Entrada**

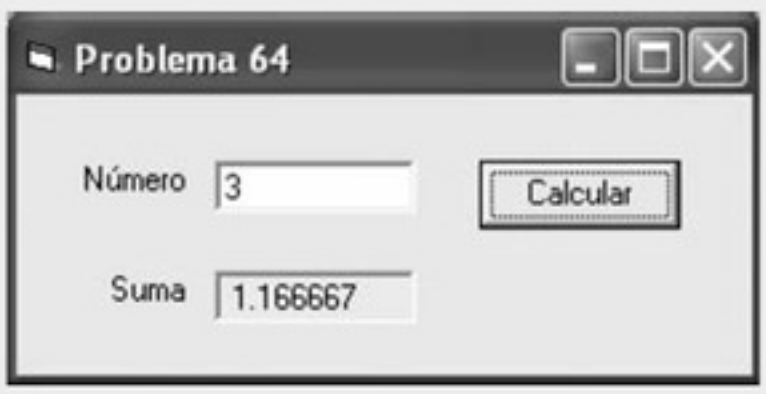
- Número (n)

**Salida**

- Suma (s)

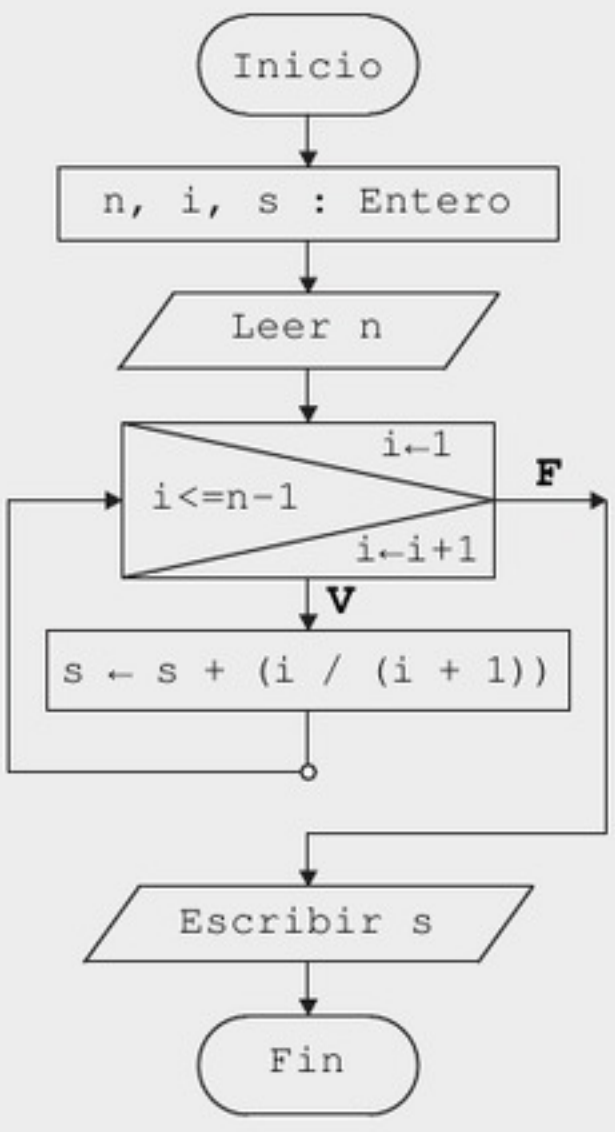
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, n,s : Entero

   //Entrada
   Leer n

   //Proceso
   Para i←1 Hasta n-1 Inc 1
       s ← s + (i / (i + 1))
   Fin Para

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim n As Single
Dim i As Single
Dim s As Single

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i = 1 To n - 1
    s = s + (i / (i + 1))
Next

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 65

**Enunciado:** Dado un rango numérico entero (número inicial y número final), obtener la cantidad de números positivos y negativos que existen en el rango.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese dos números; luego, que el sistema devuelva la cantidad de números positivos y negativos.

**Entrada**

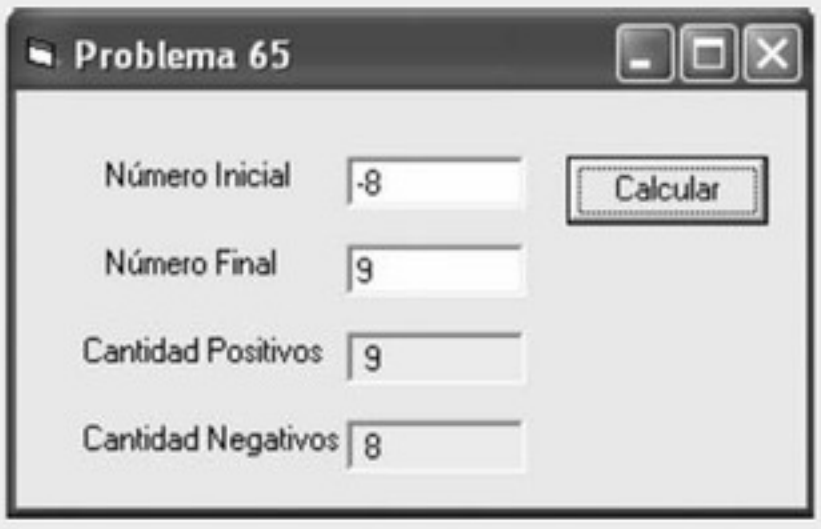
- Número inicial (ni)
- Número final (nf)

**Salida**

- Cantidad positivos (cp)
- Cantidad negativos (cn)

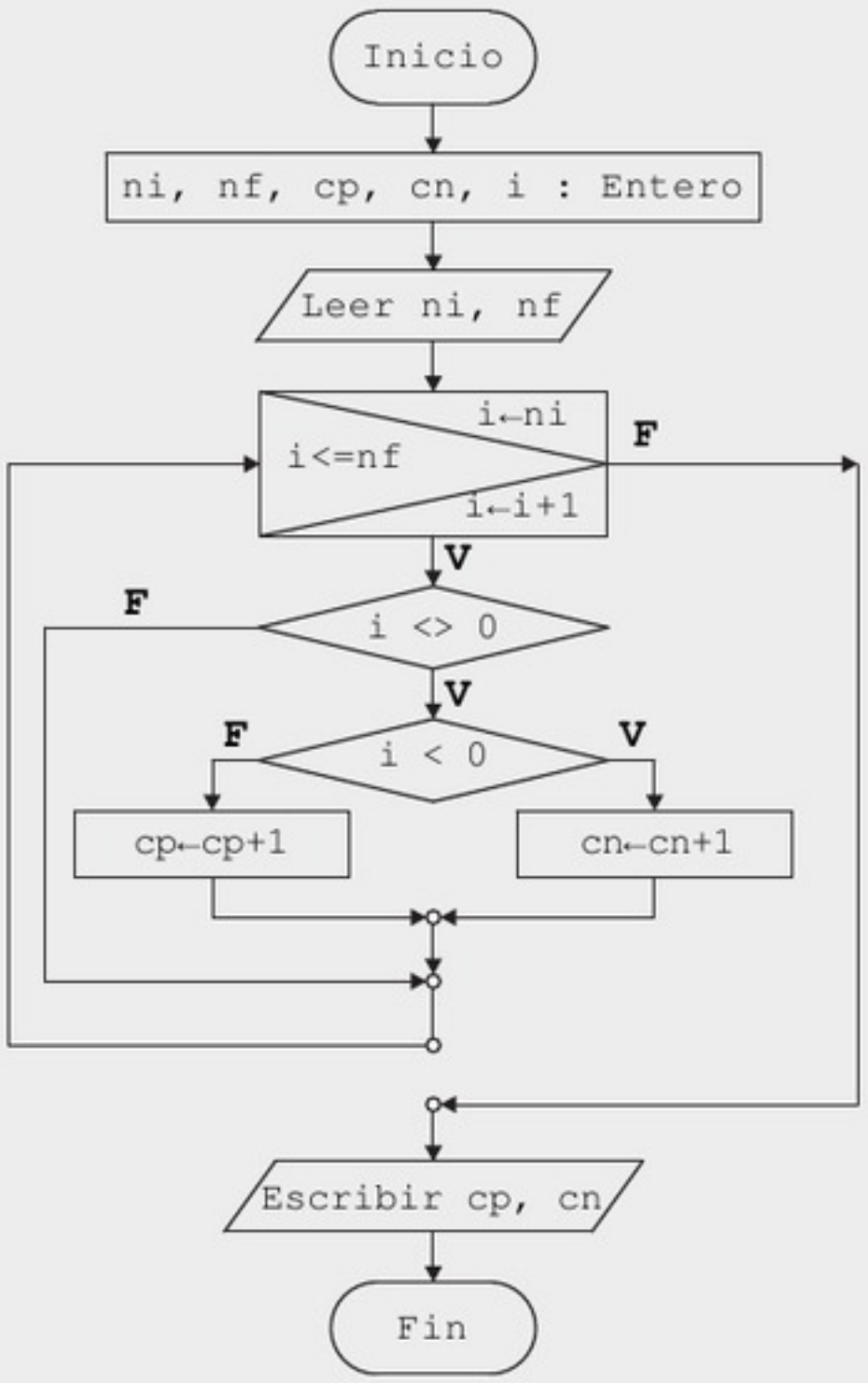
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   ni, nf, cp, cn, i : Entero

   //Entrada
   Leer ni, nf

   //Proceso
   Para i←ni Hasta nf Inc 1
       Si i <> 0 Entonces
          Si i<0 Entonces
                     cn ← cn + 1
          SiNo
                     cp ← cp + 1
          Fin Si
       Fin Si
   Fin Para

   //Salida
   Escribir cp, cn

Fin
```

**Codificación:**

```
'Variables
Dim ni As Integer
Dim nf As Integer
Dim cp As Integer
Dim cn As Integer
Dim i As Integer

'Entrada
ni = Val(Me.txtni.Text)
nf = Val(Me.txtnf.Text)

'Proceso
For i = ni To nf
    If i <> 0 Then
```

```
            If i < 0 Then
                  cn = cn + 1

            Else
                  cp = cp + 1
            End If
      End If
Next

'Salida
Me.txtcp.Text = Str(cp)
Me.txtcn.Text = Str(cn)
```

## Problema n.° 66

**Enunciado:** Hallar cuántos múltiplos de M hay en un rango de números enteros.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese tres números (num. inicial, num. final y num. múltiplo); luego, que el sistema devuelva la cantidad de múltiplos que hay en el rango.

**Entrada**

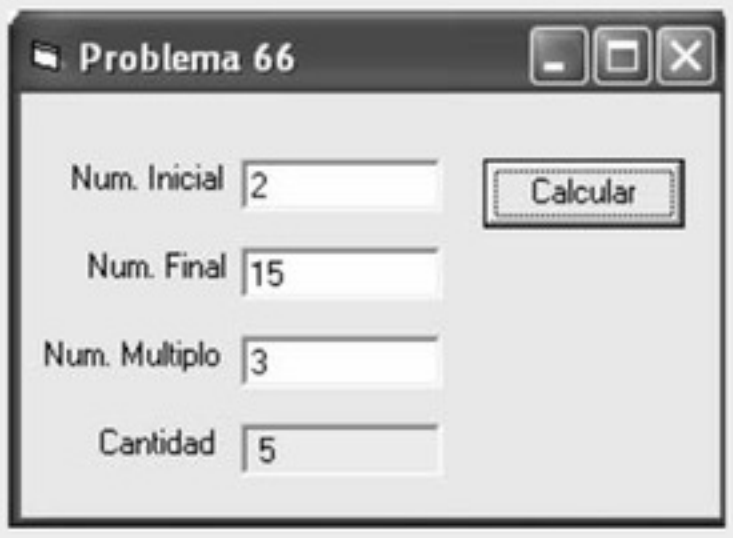
- Número inicial (ni)
- Número final (nf)
- Número múltiplo

**Salida**

- Cantidad (c)

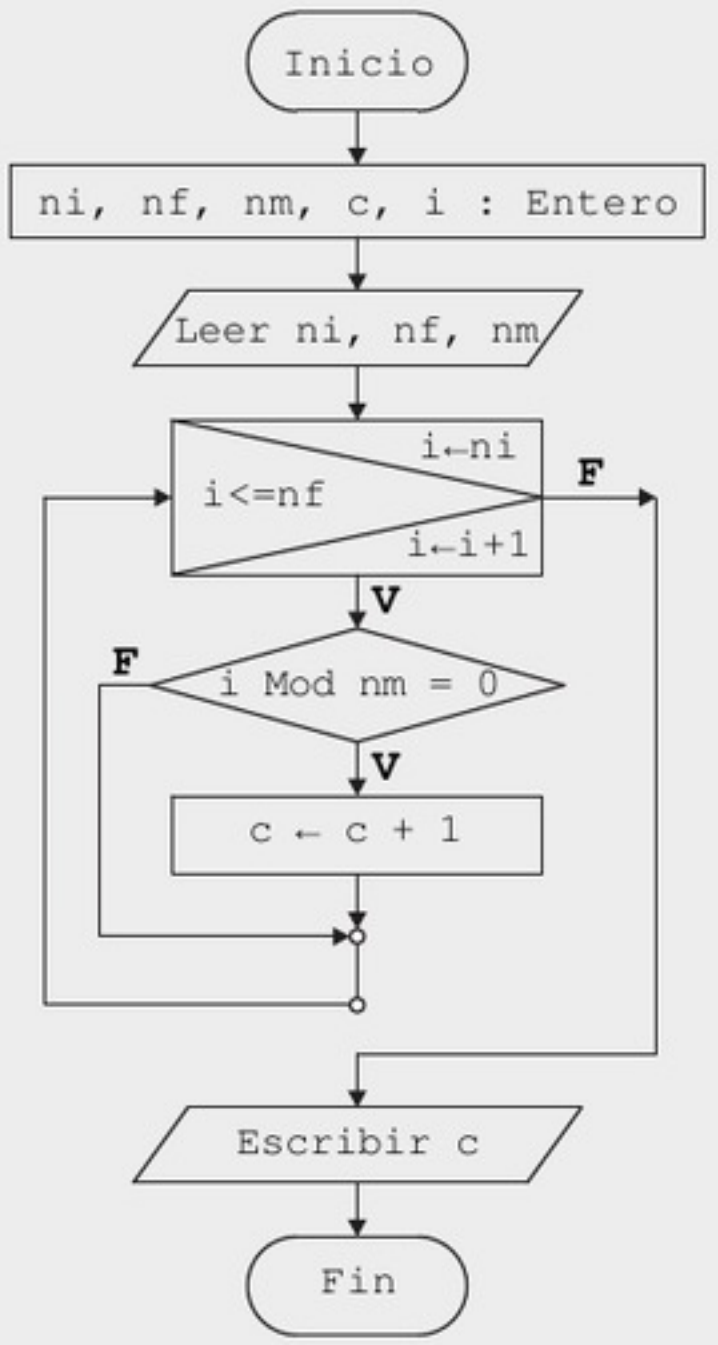
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   ni, nf, nm, c, i : Entero

   //Entrada
   Leer ni, nf, nm

   //Proceso
   Para i←ni Hasta nf Inc 1
       Si i Mod nm = 0 Entonces
              c ← c + 1
       Fin Si
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim ni As Integer
Dim nf As Integer
Dim nm As Integer
Dim c As Integer
Dim i As Integer

'Entrada
ni = Val(Me.txtni.Text)
nf = Val(Me.txtnf.Text)
nm = Val(Me.txtnm.Text)

'Proceso
For i = ni To nf
     If i Mod nm = 0 Then
          c = c + 1
     End If
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 67

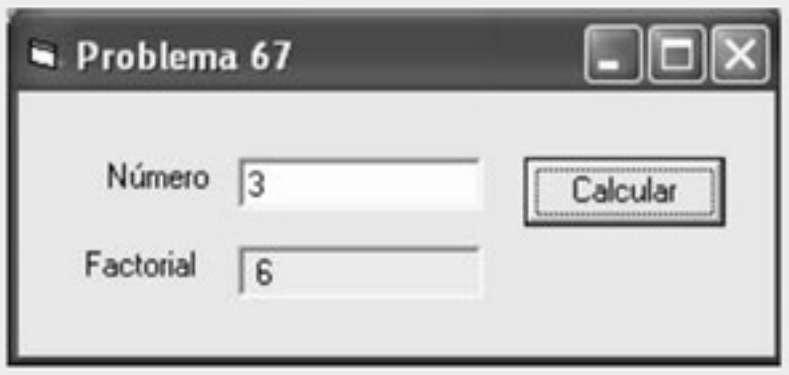
**Enunciado:** Crear un algoritmo para hallar el factorial de un número; el factorial es el producto de todos los números consecutivos, desde la unidad hasta el número. Por ejemplo, factorial de 3! (se denota !) es 1 X 2 X 3 = 6.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva el factorial del número.

| Entrada | Salida |
|---|---|
| • Número (n) | • Factorial (f) |

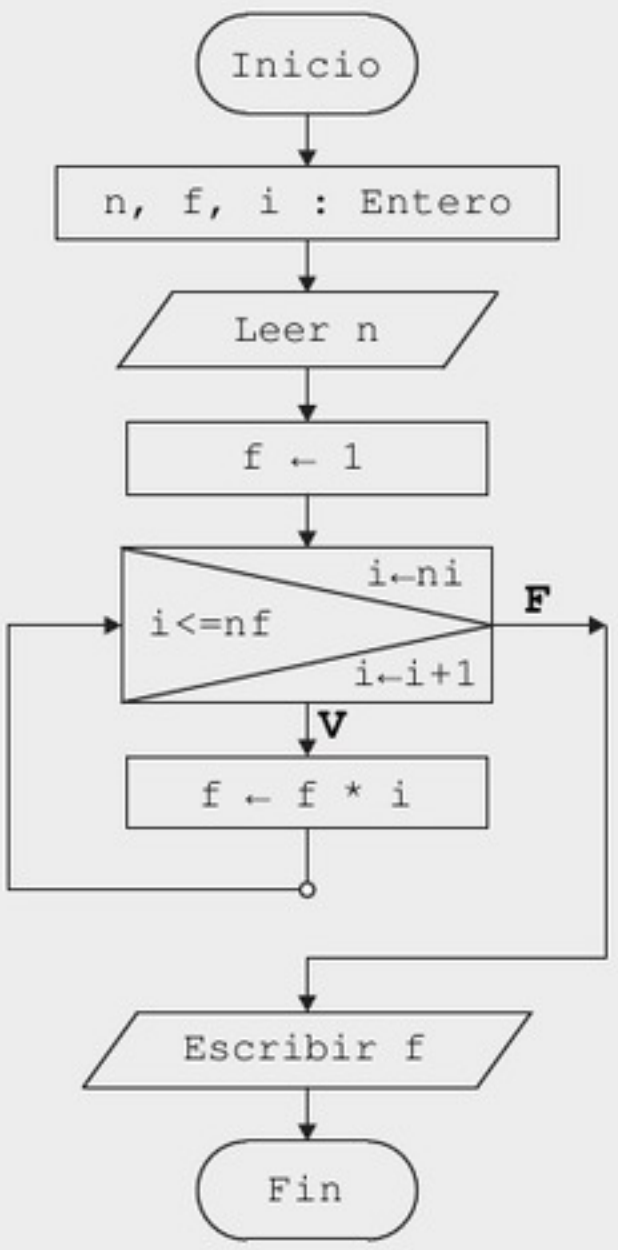
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, f, i : Entero

   //Entrada
   Leer n

   //Proceso
   f ← 1
   Para i←1 Hasta n Inc 1
       f ← f * i
   Fin Para

   //Salida
   Escribir f

Fin
```

**Codificación:**

```
'Variables
Dim n As Long
Dim f As Long
Dim i As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
f = 1
For i = 1 To n
    f = f * i
Next

'Salida
Me.txtf.Text = Str(f)
```

## Problema n.° 68

**Enunciado:** Determine si un número es primo; se dice que un número es primo si es divisible entre 1 y entre sí mismo.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva si el número es o no primo.

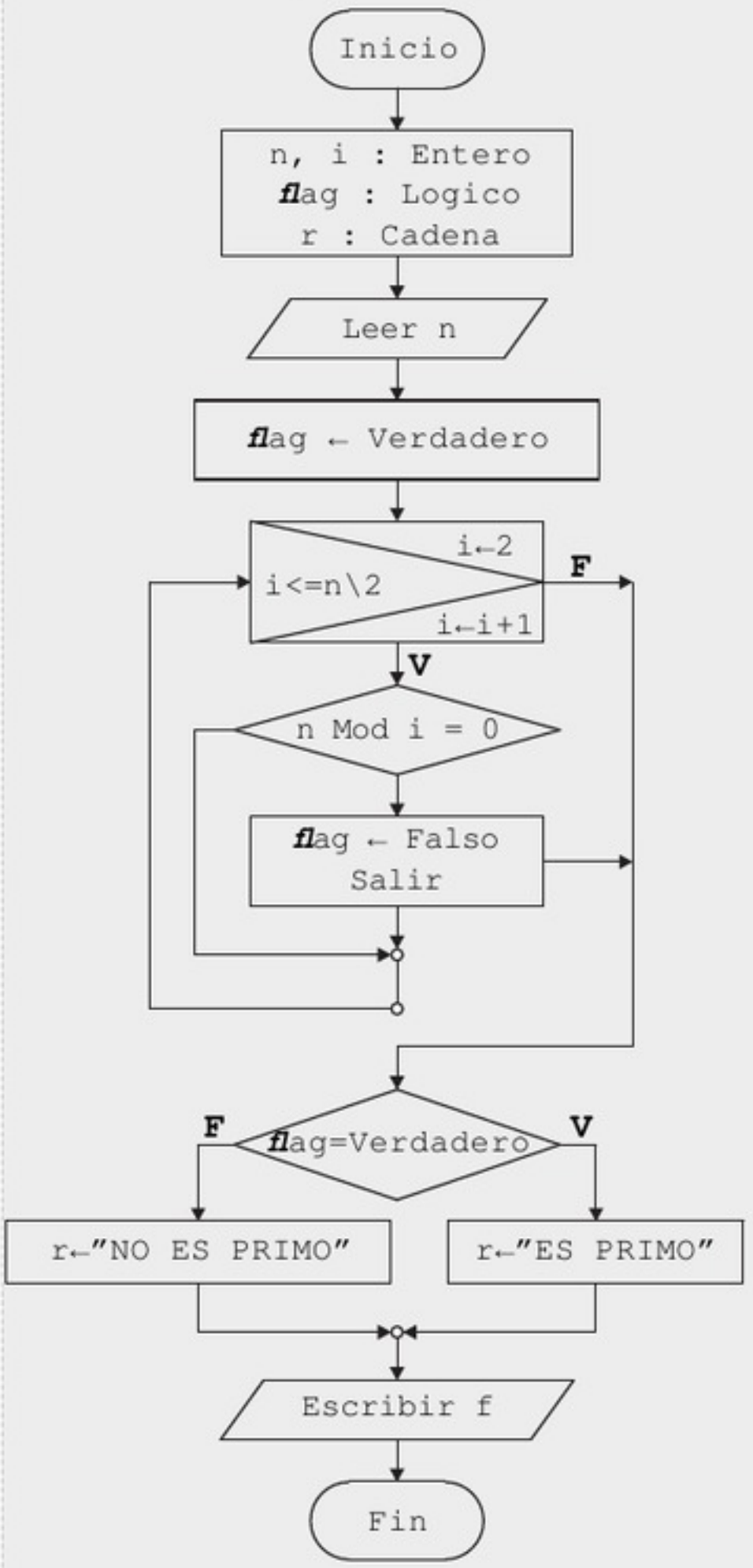
| Entrada | Salida |
|---|---|
| • Número (n) | • Respuesta (r) |

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, i : Entero
   flag : Logico
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   flag ← Verdadero
   Para i←1 Hasta n\2 Inc 1
       Si n Mod i = 0 Entonces
              flag ← Falso
              Salir
       Fin Si
   Fin Para

   Si flag = Verdadero Entonces
       r ← "ES PRIMO"
   SiNo
       r ← "NO ES PRIMO"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim i As Integer
Dim flag As Boolean
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
flag = True
For i = 2 To n \ 2
    If n Mod i = 0 Then
        flag = False
        Exit For
    End If
Next

If flag Then
    r = "ES PRIMO"
Else
    r = "NO ES PRIMO"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 69

**Enunciado:** Determine cuántos números primos hay en los primeros N números enteros positivos.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva la cantidad de números primos; por ejemplo, si ingresa 7, hay 4 números primos 1, 3, 5 y 7.

**Entrada**

- Número (n)

**Salida**

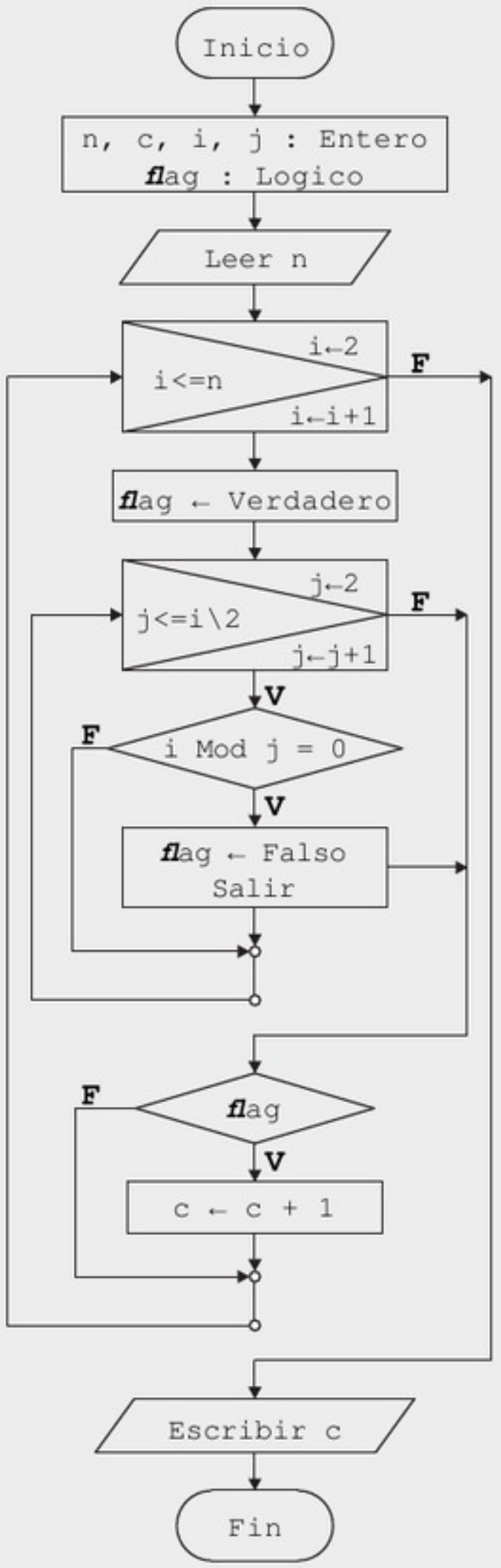
- Cantidad (c)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, c, i, j : Entero
   flag : Logico

   //Entrada
   Leer n

   //Proceso
   Para i←2 Hasta n Inc 1
      flag ← Verdadero
      Para j←2 Hasta i\2 Inc 1
           Si i Mod j = 0 Entonces
                flag ← Falso
                Salir
           Fin Si
      Fin Para

      Si flag Entonces
         c ← c + 1
      Fin Si
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim c As Integer
Dim i As Integer
Dim j As Integer
Dim flag As Boolean

'Entrada
n = Val(Me.txtn.Text)

'Proceso
For i = 2 To n
    flag = True
    For j = 2 To i \ 2
        If i Mod j = 0 Then
            flag = False
            Exit For
        End If
    Next
    If flag Then
        c = c + 1
    End If
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 70

**Enunciado:** Dado un número y un divisor, determine cuál es el número múltiplo antecesor al número ingresado; por ejemplo, si ingresa N = 21 y D = 3, entonces R = 18, porque es el número múltiplo de 3 antecesor de 21.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un número; luego, que el sistema devuelva el número múltiplo antecesor.

**Entrada**

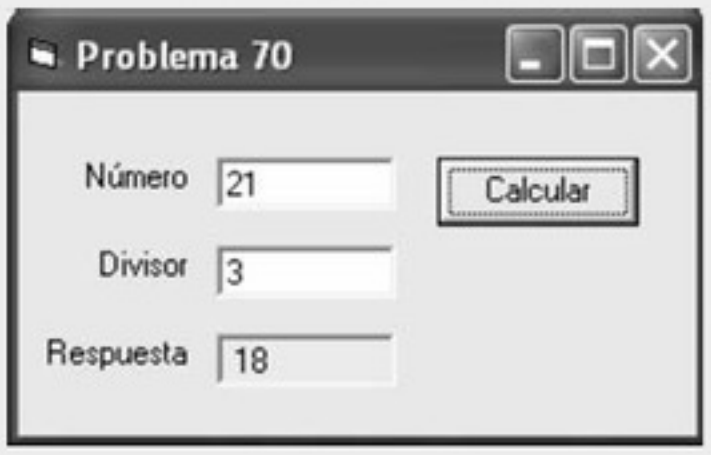
- Número (n)
- Divisor (d)

**Salida**

- Respuesta (r)

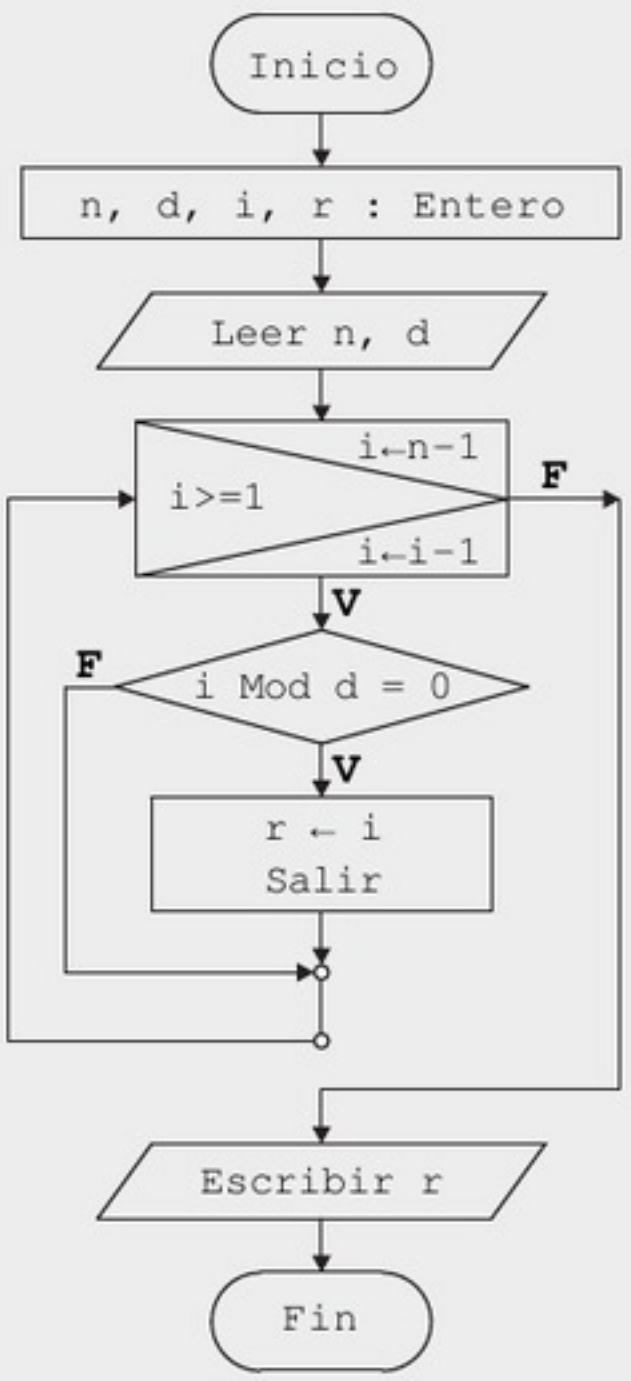
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, d, i, r : Entero

   //Entrada
   Leer n, d

   //Proceso
   Para i←n-1 Hasta 1 Inc -1
       Si i Mod d = 0 Entonces
              r ← i
              Salir
       Fin Si
   Fin Para

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As Integer
Dim d As Integer
Dim i As Integer
Dim r As Integer

'Entrada
n = Val(Me.txtn.Text)
d = Val(Me.txtd.Text)

'Proceso
For i = n - 1 To 1 Step -1
    If i Mod d = 0 Then
        r = i
        Exit For
    End If
Next

'Salida
Me.txtr.Text = Str(r)
```

## 6.4 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 41

**Enunciado:** Calcule la suma de los cuadrados y cubos de los N primeros números naturales.

### Propuesto n.° 42

**Enunciado:** Obtener la suma y la cantidad de los números divisibles por 3 y 5 a la vez, de los N primeros números naturales.

### Propuesto n.° 43

**Enunciado:** Dado un rango numérico entero positivo a y b, obtener la suma y la cantidad de los números pares, impares y múltiplos de 3.

### Propuesto n.° 44

**Enunciado:** Calcule la suma y la cantidad de números de la serie de fibonacci, menores a N. La serie de fibonacci es una secuencia de números cuya característica es que cada número de la serie debe ser igual a la suma de los 2 números anteriores; la serie empieza con 0 y 1, entonces, si el número N ingresado es 30, la serie sería menor a 30. Esto equivale a 0 1 1 2 3 5 8 13 21, y lo que se pide es la suma y la cantidad de números de la serie.

### Propuesto n.° 45

**Enunciado:** Dado un rango de números, determine cuántos números capicúa hay.

### Propuesto n.° 46

**Enunciado:** Dada la cantidad de cifras y un divisor, determine cuántos números múltiplos del divisor con dichas cifras existen.

### Propuesto n.° 47

**Enunciado:** Calcule la suma de la siguiente serie:

$$s = \frac{1}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots + \frac{x^n}{n!}$$

### Propuesto n.° 48

**Enunciado:** Calcule de cuántas formas se pueden ordenar **n** objetos.

### Propuesto n.° 49

**Enunciado:** Obtenga la cantidad de los números primos de n cifras.

### Propuesto n.° 50

**Enunciado:** Obtenga la cantidad de los números capicúas de n cifras.

# Capítulo 7

# Estructuras de datos. Arreglos (vectores y matrices)

## 7.1 Introducción

En muchas situaciones se necesita procesar una colección de datos que están relacionados entre sí, por ejemplo, la lista de notas de los alumnos, los participantes de una carrera deportiva, etc. Procesar ese conjunto de datos en forma independiente con variables simples (primitivas), es tremendamente difícil, por eso los lenguajes de programación incorporan un mecanismo que facilita la manipulación y organización para una colección de datos llamada «estructura de datos».

Para explicar todo lo relacionado a estructura de datos se necesita escribir todo un libro que detalle los temas involucrados, para este capítulo solo se está considerando una parte básica e importante en la estructura de datos, llamada *array* (arreglos).

Vector

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 15 | 12 | 18 | 14 | 12 |

Matriz

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 25 | 10 | 15 | 32 |
| 1 | 52 | 10 | 4 | 18 |
| 2 | 18 | 22 | 3 | 9 |

Las estructuras de datos están subdivididas por estáticas (espacio fijo establecido en memoria) y dinámicas (sin restricciones y limitaciones en el espacio usado en memoria).

**Estructuras de datos estáticas**

- *Arrays* (vectores y matrices)
- Cadenas
- Registros
- Ficheros

**Estructuras de datos dinámicas**

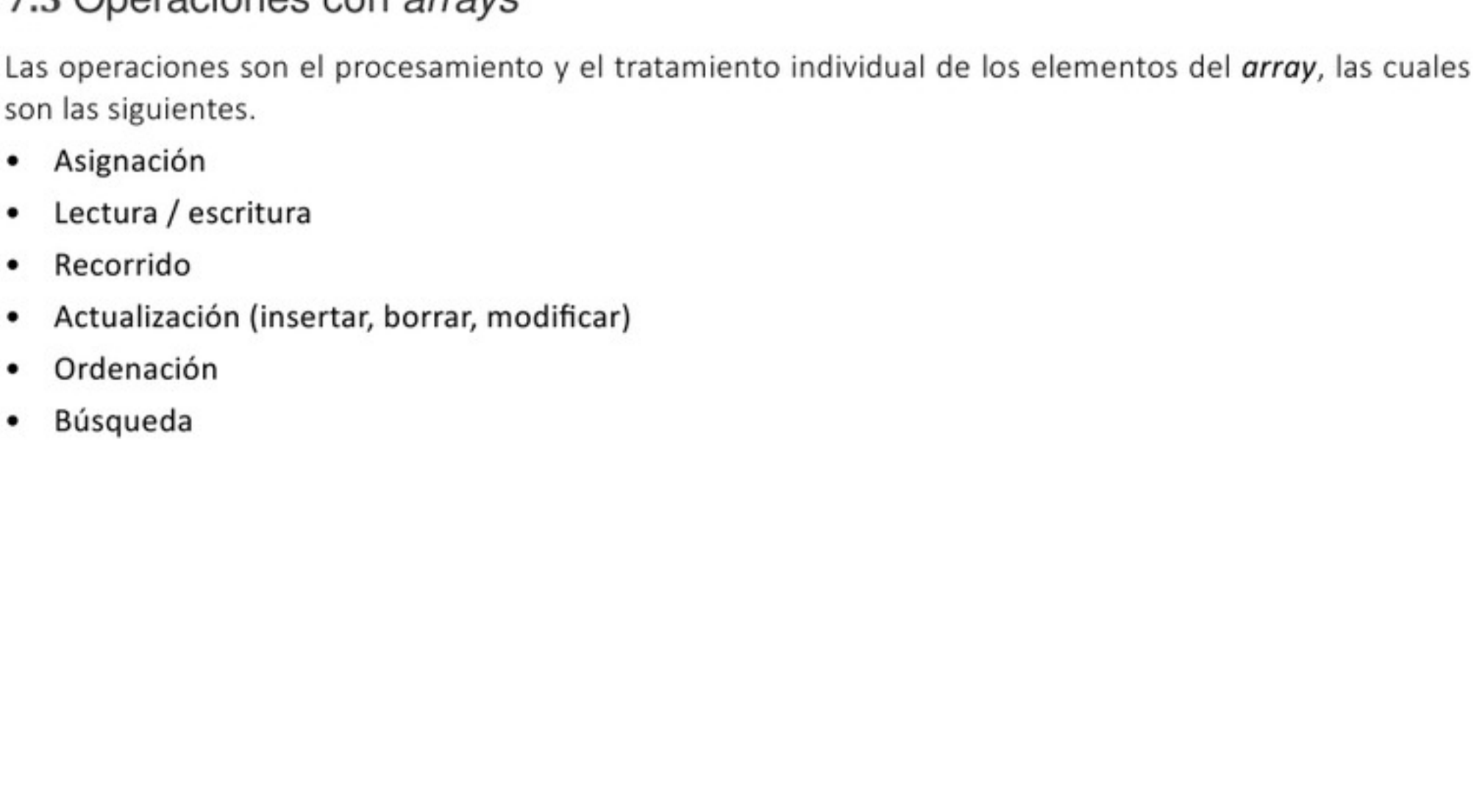
- Listas (pilas y colas)
- Listas enlazadas
- Árboles
- Grafos

La diferencia entre cada estructura es la forma en cómo se almacena y manipula el conjunto de datos, permitiendo así su eficiencia en el resultado de una operación sobre dichos datos.

## 7.2 *Arrays* (Arreglos)

Es un conjunto finito (tamaño fijo) y ordenado (usa un índice) de datos homogéneos (datos del mismo tipo). Los arreglos pueden ser de una dimensión (vector), dos dimensiones (matriz) y n dimensiones (multidimensional).

En todos los lenguajes de programación, los ***arrays*** usan un índice numérico para cada elemento que contiene, los cuales por lo general inician con el índice 0, llamado «Limite Inferior» (LI); y el último elemento tendrá el índice llamado «Limite Superior» (LS), que en sí es la cantidad de elementos del ***array*** menos 1.




## 7.3 Operaciones con *arrays*

Las operaciones son el procesamiento y el tratamiento individual de los elementos del ***array***, las cuales son las siguientes.

- Asignación
- Lectura / escritura
- Recorrido
- Actualización (insertar, borrar, modificar)
- Ordenación
- Búsqueda

## 7.4 Creación de *arrays*

Para la creación de un ***array*** se requiere conocer el nombre, las dimensiones, el tamaño de elementos y el tipo de dato.

**Pseudocódigo**

```
//Array de una dimensión (Vector)
// 5 elementos LI = 0 y LS = 4
N[5] : Entero

//Array de dos dimensiones (Matriz)
// 3X4 elementos
// 1era Dim. LI = 0 y LS = 2
// 2da Dim. LI = 0 y LS = 3
N[3][4] : Entero
```

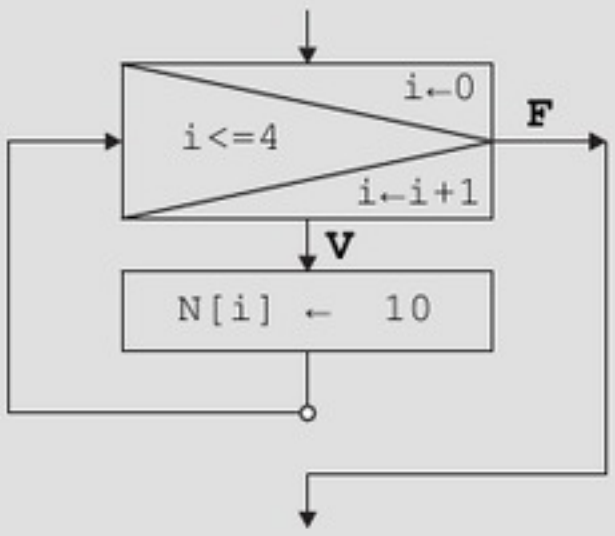
**Visual Basic**

```
'Array de una dimensión (Vector)
' 5 elementos LI = 0 y LS = 4
Dim N(4) As Integer

'Array de dos dimensiones (Matriz)
' 3X4 elementos
' 1era Dim. LI = 0 y LS = 2
' 2da Dim. LI = 0 y LS = 3
Dim N(2,3) As Integer
```

## 7.5 Recorrido por los elementos del *array*

Para realizar un recorrido por cada elemento del *array* utilizaremos la estructura repetitiva «**Para**» (For).

En el siguiente diagrama se tiene el **vector** N de 5 elementos, y se asigna el valor 10 a cada elemento.

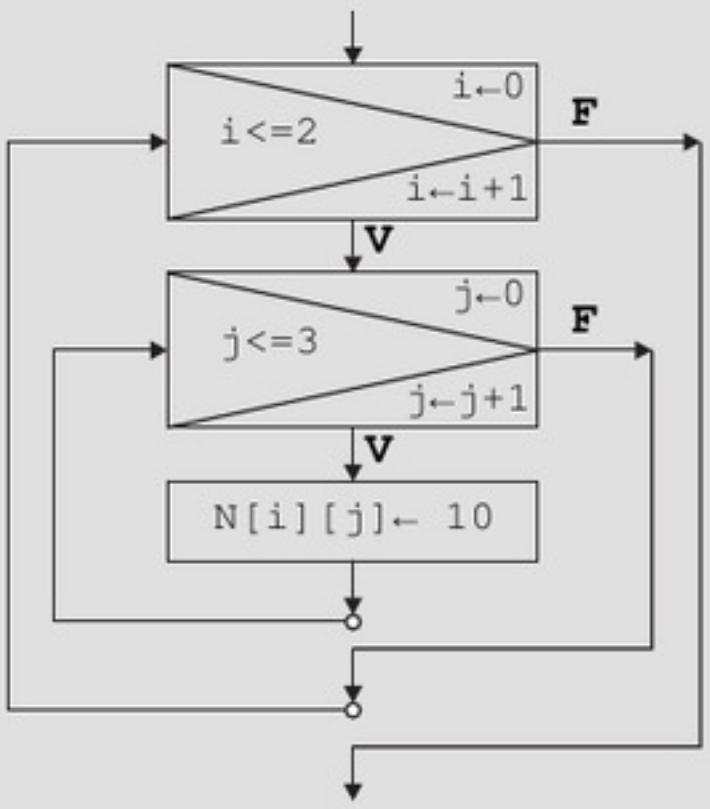



```
Para i← 0 Hasta 4 Inc +1
      N[i] ← 10
Fin Para
```

**Sintaxis Visual Basic**

```
For i=0 To 4
   N(i) = 10
Next
```

En el siguiente diagrama se tiene la **matriz** N de 3x4 elementos y se asigna el valor 10 a cada elemento.




```
Para i← 0 Hasta 2 Inc +1
      Para j← 0 Hasta 3 Inc +1
            N[i][j] ← 10
      Fin Para
Fin Para
```

**Sintaxis Visual Basic**

```
For i=0 To 2
      For j=0 To 3

            N(i,j) = 10

      Next
Next
```

## Problema n.° 71

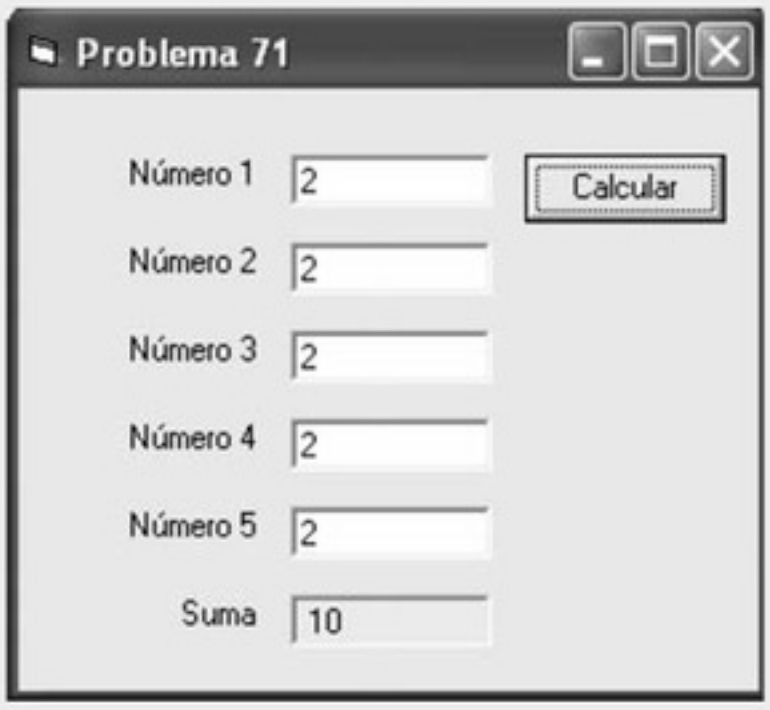
**Enunciado:** Dados 5 números, obtener la suma.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 5 números; luego, que el sistema realice el proceso para devolver la suma.

| Entrada | Salida |
|---|---|
| • 5 Números n[5] | • Suma (s) |

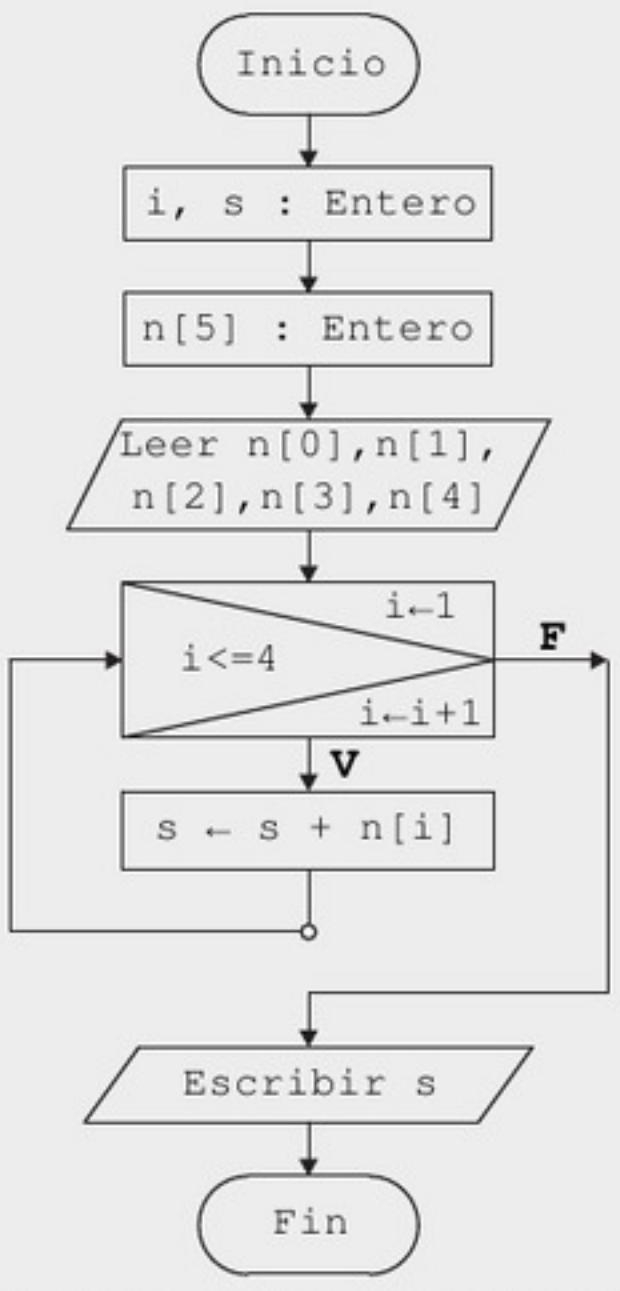
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, s : Entero

   //Arreglos (Vector)
   n[5] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3],n[4]

   //Proceso
   Para i←0 Hasta 4 Inc 1
       s ← s + n[i]
   Fin Para

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim s As Integer
Dim i As Integer

'Arreglos (Vector)
Dim n(4) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)
n(4) = Val(Me.txtn5.Text)

'Proceso
For i = 0 To 4
    s = s + n(i)
Next

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 72

**Enunciado:** Dados 5 números, obtener el número mayor.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 5 números; luego, que el sistema realice el proceso para devolver el mayor.

**Entrada**

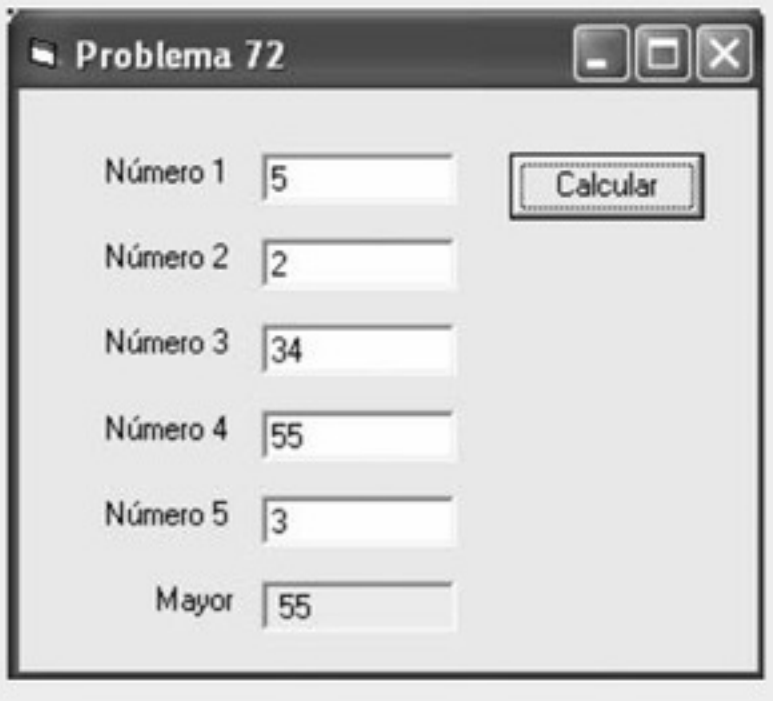
- 5 números n[5]

**Salida**

- Mayor (m)

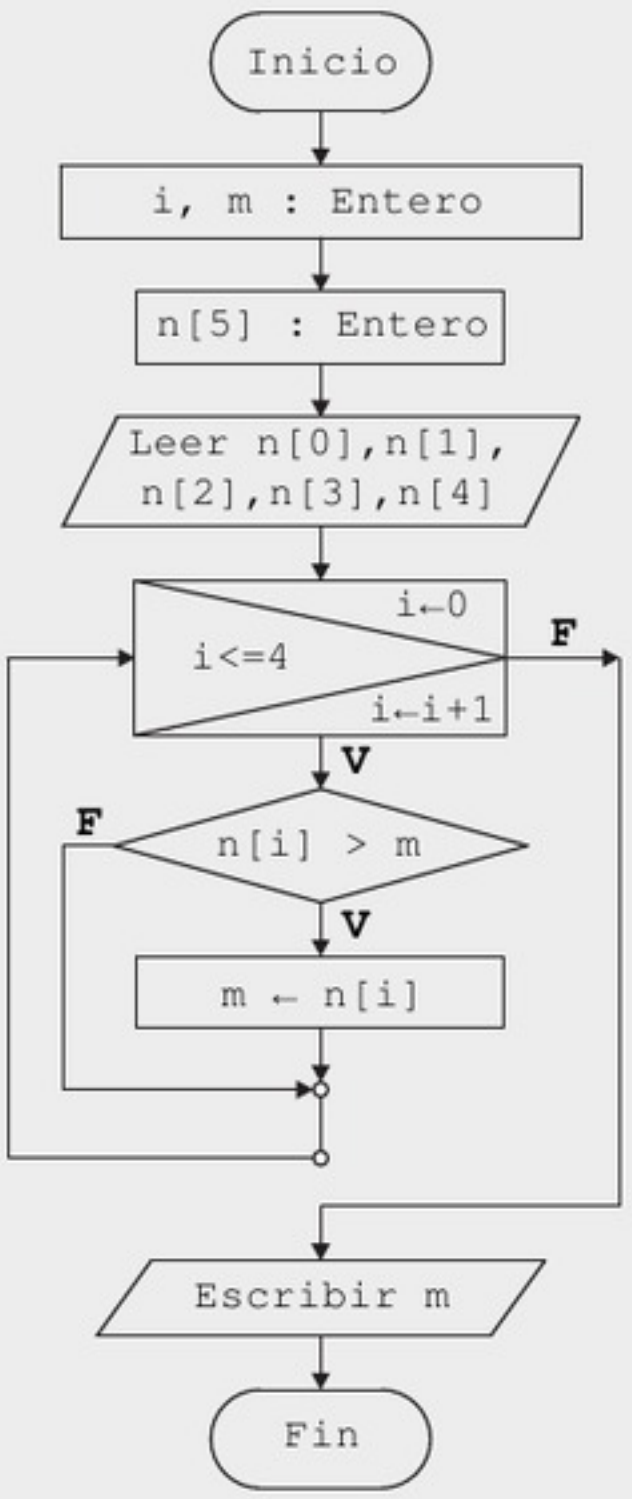
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, m : Entero

   //Arreglos (Vector)
   n[5] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3],n[4]

   //Proceso
   Para i←0 Hasta 4 Inc 1
       Si n[i] > m Entonces
          m ← n[i]
       Fin Si
   Fin Para

   //Salida
   Escribir m

Fin
```

**Codificación:**

```
'Variables
Dim m As Integer
Dim i As Integer

'Arreglos
Dim n(4) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)
n(4) = Val(Me.txtn5.Text)

'Proceso
For i = 0 To 4
     If n(i) > m Then
          m = n(i)
     End If
Next

'Salida
Me.txtm.Text = Str(m)
```

## Problema n.° 73

**Enunciado:** Dados 5 números y un divisor, determinar cuántos números múltiplos hay del divisor en los 5 números ingresados.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 5 números; luego, que el sistema procese y devuelva la cantidad de números múltiplos que hay.

**Entrada**

- 5 números (n[5])
- Divisor (d)

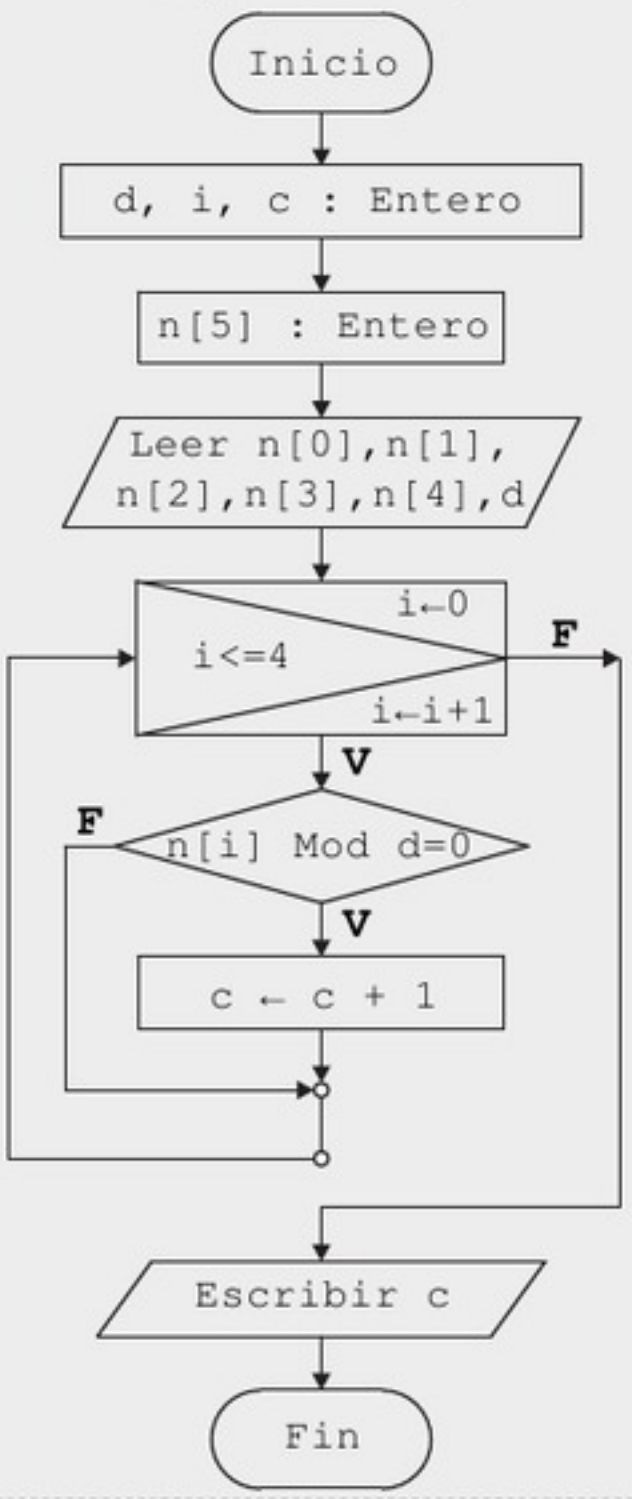
**Salida**

- Cantidad (c)

**Diseño:**

**Interfaz de usuario**


**Diagrama de flujo**




**Algoritmo**

**Pseudocódigo**

```
Inicio

   //Variables
   d, i, c : Entero

   //Arreglos (Vector)
   n[5] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3],n[4],d

   //Proceso
   Para i←0 Hasta 4 Inc 1
       Si n[i] Mod d = 0 Entonces
             c ← c + 1
       Fin Si
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim d As Integer
Dim i As Integer
Dim c As Integer

'Arreglos
Dim n(4) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)
n(4) = Val(Me.txtn5.Text)
d = Val(Me.txtd.Text)

'Proceso
For i = 0 To UBound(n, 1)
    If n(i) Mod d = 0 Then
        c = c + 1
    End If
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 74

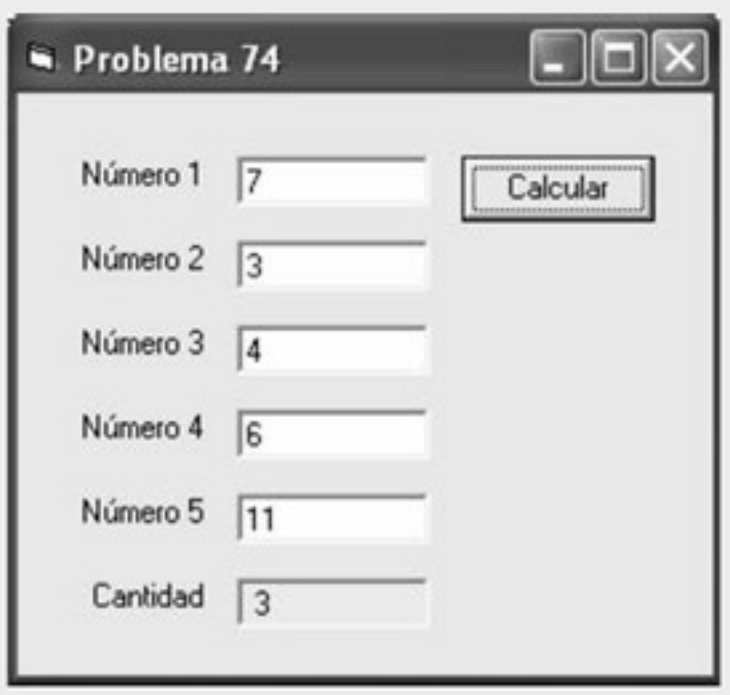
**Enunciado:** Dados 5 números, obtener la cantidad de números primos ingresados.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 5 números; luego, que el sistema procese y devuelva la cantidad números primos.

| Entrada | Salida |
|---|---|
| • 5 números (n[5]) | • Cantidad (c) |

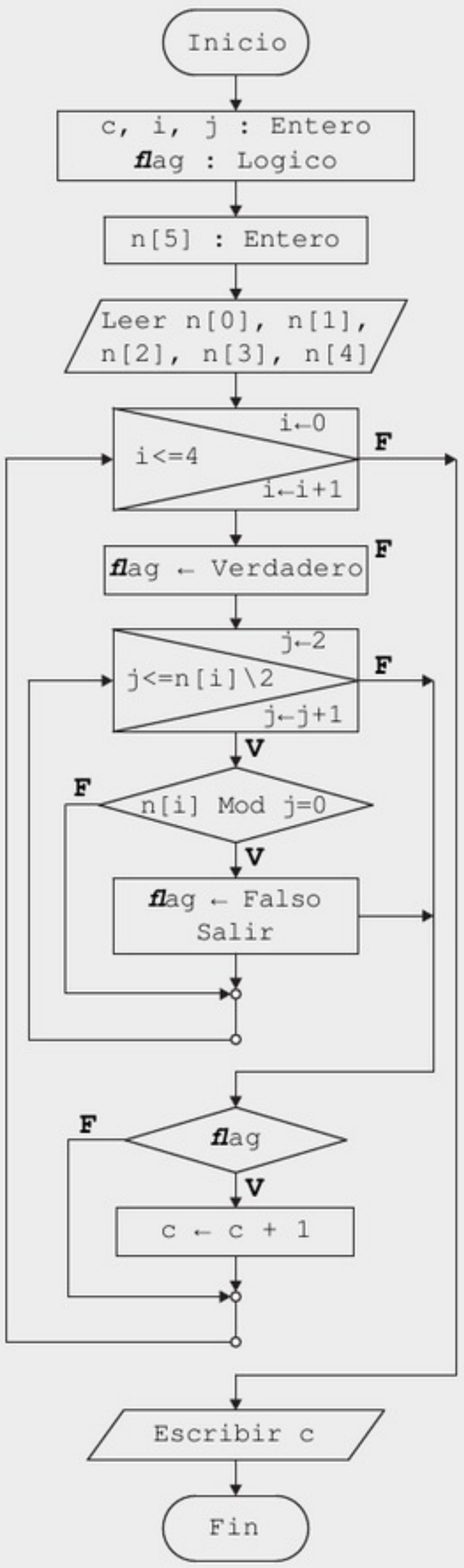
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio
   //Variables
   c, i, j : Entero
   flan : Logico

   //Arreglos (Vector)
   n[5] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3],n[4]

   //Proceso
   Para i←0 Hasta 4 Inc 1
      flag ← Verdadero
       Para j←2 Hasta n[i]\2 Inc 1
         Si n[i] Mod j=0 Entonces
            flag ← Falso
            Salir
         Fin Si
       Fin Para
      Si flag Entonces
         c ← c + 1
      Fin Si
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim c As Integer
Dim i As Integer
Dim j As Integer
Dim flag As Boolean

'Arreglos
Dim n(4) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)
n(4) = Val(Me.txtn5.Text)

'Proceso
For i = 0 To 4
     flag = True
     For j = 2 To n(i) \ 2
          If n(i) Mod j = 0 Then
               flag = False
               Exit For
          End If
     Next
     If flag Then
          c = c + 1
     End If
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 75

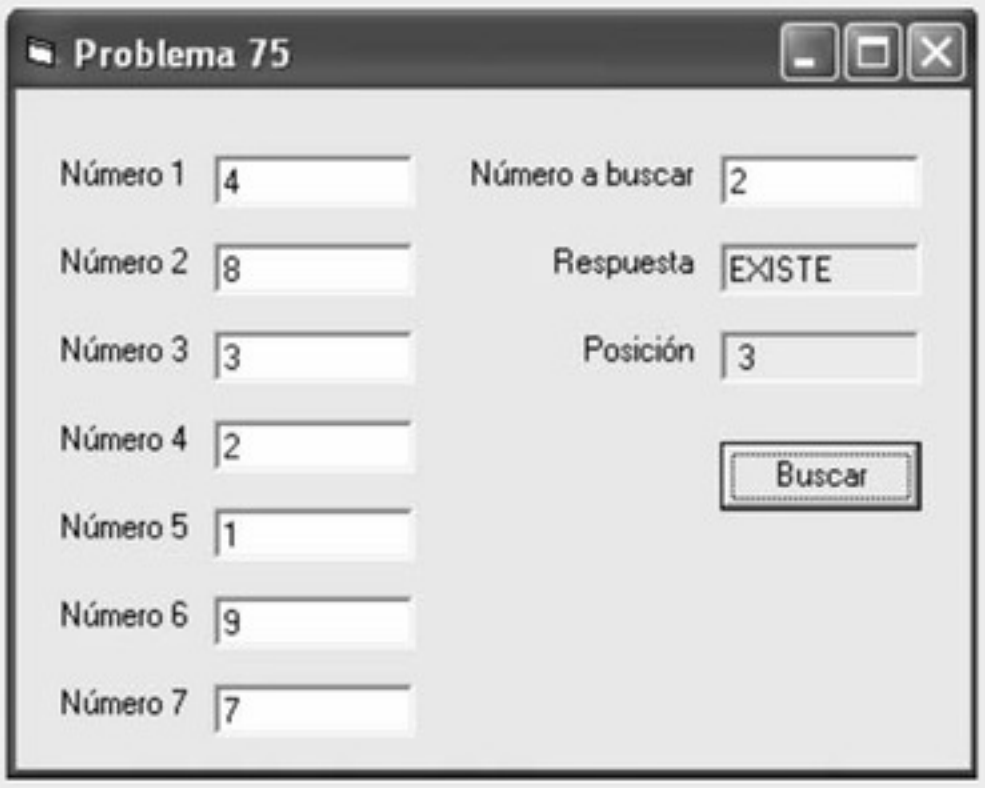
**Enunciado:** Busque un número en 7 números ingresados y determine la posición, y si existe o no el número buscado, use el método de búsqueda secuencial.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 7 números; luego, que el sistema devuelva la respuesta, si existe o no el número y la posición del número encontrado.

| Entrada | Salida |
|---|---|
| • 7 números (n[7]) | • Respuesta (r) |
| • Numero a buscar (nb) | • Posición (p) |

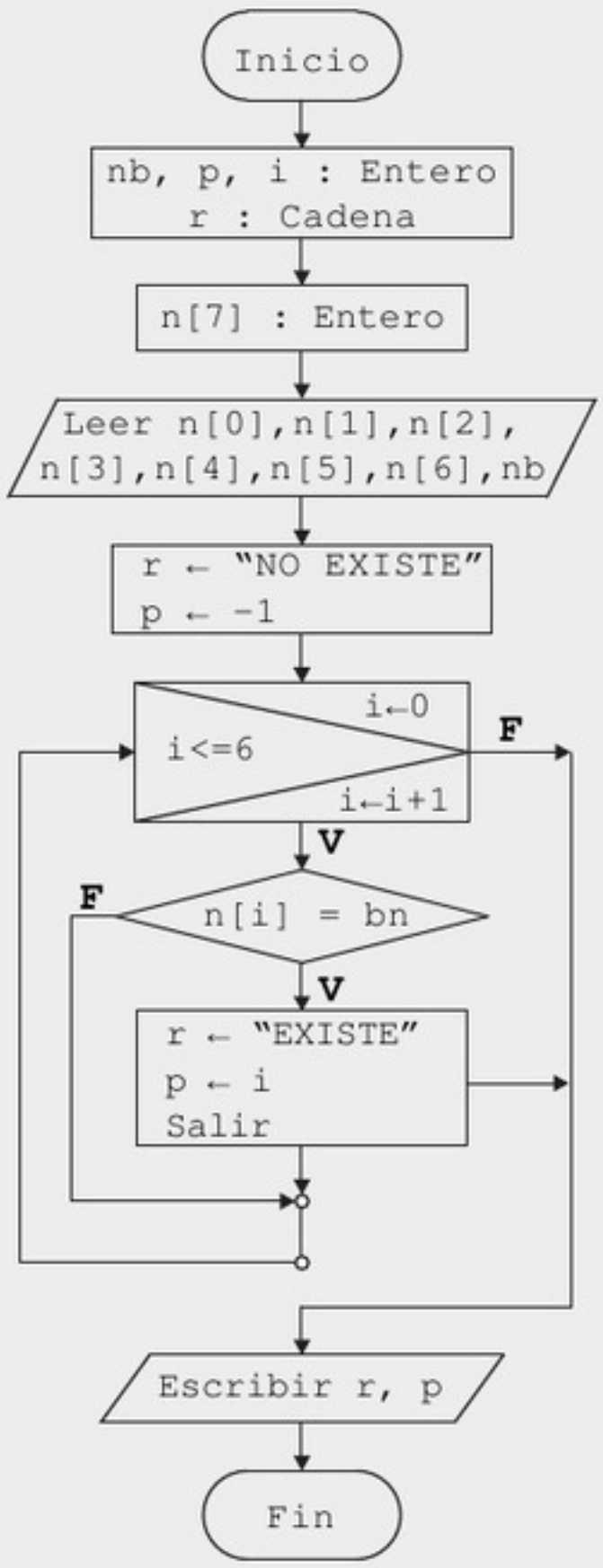
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   nb, p, i : Entero
   r : Cadena

   //Arreglos (Vector)
   n[7] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3],n[4],
        n[5],n[6],nb

   //Proceso
   r ← "NO EXISTE"
   p ← -1
   Para i←0 Hasta 6 Inc 1
       Si n[i] = nb Entonces
              r ← "EXISTE"
              p ← i
              Salir
       Fin Si
   Fin Para

   //Salida
   Escribir r, p

Fin
```

**Codificación:**

```
'Variables
Dim nb As Integer
Dim r As String
Dim p As Integer
Dim i As Integer

'Arreglos (Vector)
Dim n(6) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)
n(4) = Val(Me.txtn5.Text)
n(5) = Val(Me.txtn6.Text)
n(6) = Val(Me.txtn7.Text)
nb = Val(Me.txtnb.Text)

'Proceso
r = "NO EXISTE"
p = -1
For i = 0 To UBound(n, 1)
    If n(i) = nb Then
        r = "EXISTE"
        p = i
        Exit For
    End If
Next

'Salida
Me.txtr.Text = r
Me.txtp.Text = Str(p)
```

## Problema n.° 76

**Enunciado:** Lea 4 números y almacénelos en un vector llamado A; lo mismo con otros 4 números en un vector llamado B, y determine cuantos números de A se encuentran en B.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 8 números; luego, que el sistema devuelve la cantidad.

**Entrada**

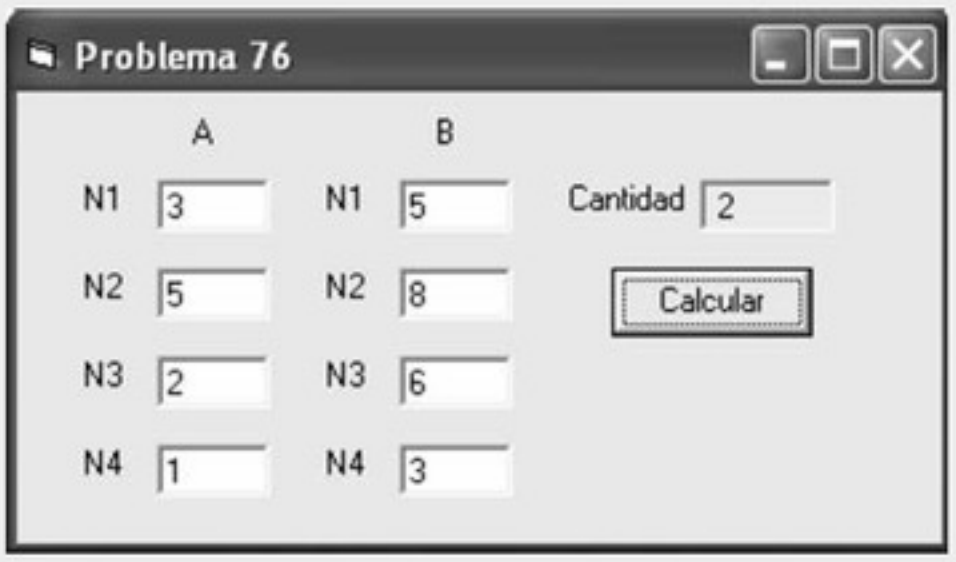
- 4 números (a[4])
- 4 números (b[4])

**Salida**

- Cantidad (c)

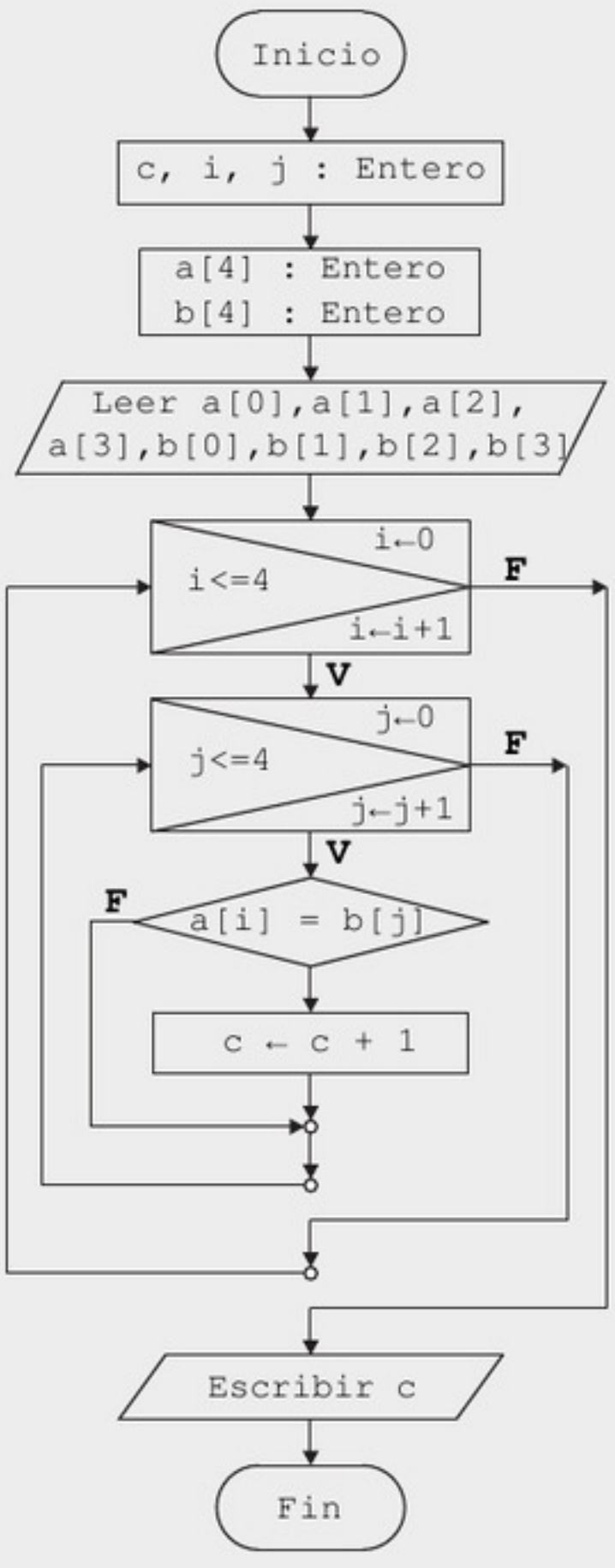
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c, i, j : Entero

   //Arreglos (Vector)
   a[4], b[4] : Entero

   //Entrada
   Leer a[0],a[1],a[2],a[3],
        b[0],b[1],b[2],b[3]

   //Proceso
   Para i←0 Hasta 4 Inc 1
       Para j←0 Hasta 4 Inc 1
      Si a[i]=b[j] Entonces
            c ← c + 1
         Fin Si
       Fin Para
   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim c As Integer
Dim i As Integer
Dim j As Integer

'Arreglos
Dim a(3) As Integer
Dim b(3) As Integer

'Entrada
a(0) = Val(Me.txta1.Text)
a(1) = Val(Me.txta2.Text)
a(2) = Val(Me.txta3.Text)
a(3) = Val(Me.txta4.Text)
b(0) = Val(Me.txtb1.Text)
b(1) = Val(Me.txtb2.Text)
b(2) = Val(Me.txtb3.Text)
b(3) = Val(Me.txtb4.Text)

'Proceso
For i = 0 To UBound(a, 1)
    For j = 0 To UBound(b, 1)
        If a(i) = b(j) Then
            c = c + 1
        End If
    Next
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 77

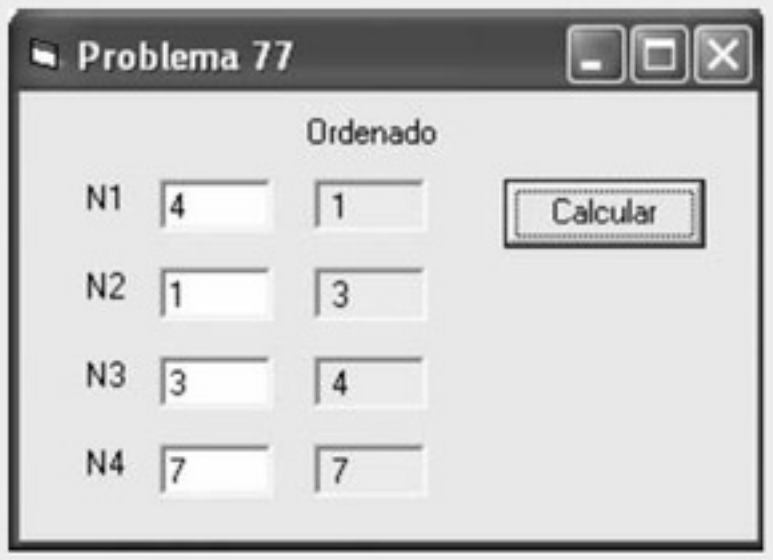
**Enunciado:** Ordene 4 números usando el método de ordenación por intercambio (burbuja).

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 4 números; luego, que el sistema devuelva los números ordenados.

| Entrada | Salida |
|---|---|
| • 4 números (n[4]) | • 4 números ordenados (n[4]) |

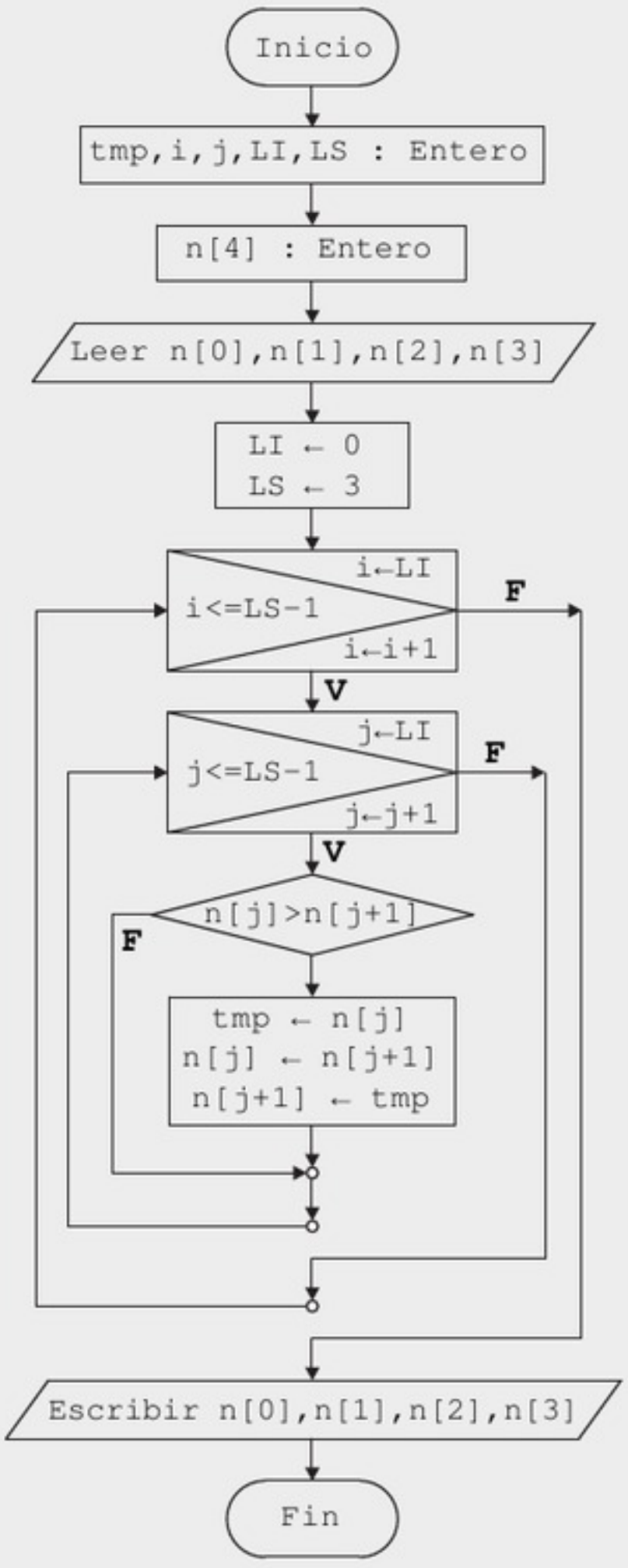
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   tmp,i,j,LI,LS : Entero

   //Arreglos (Vector)
   n[4] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3]

   //Proceso
   LI ← 0
   LS ← 3
   Para i←LI Hasta LS-1 Inc 1
       Para j←LI Hasta LS-1 Inc 1
           Si n[j]>n[j+1] Entonces
              tmp ← n[j]
              n[j] ← n[j+1]
              n[j+1] ← tmp
           Fin Si
       Fin Para
   Fin Para

   //Salida
   Escribir n[0],n[1],n[2],n[3]

Fin
```

**Codificación:**

```
'Variables
Dim tmp As Integer
Dim i As Integer
Dim j As Integer
Dim LI As Integer
Dim LS As Integer

'Arreglos (Vector)
Dim n(3) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)

'Proceso
LI = LBound(n, 1)
LS = UBound(n, 1)

For i = LI To LS - 1
    For j = LI To LS - 1
        If n(j) > n(j + 1) Then
            tmp = n(j)
            n(j) = n(j + 1)
            n(j + 1) = tmp
        End If
    Next
Next

'Salida
Me.txtno1.Text = Str(n(0))
Me.txtno2.Text = Str(n(1))
Me.txtno3.Text = Str(n(2))
Me.txtno4.Text = Str(n(3))
```

## Problema n.° 78

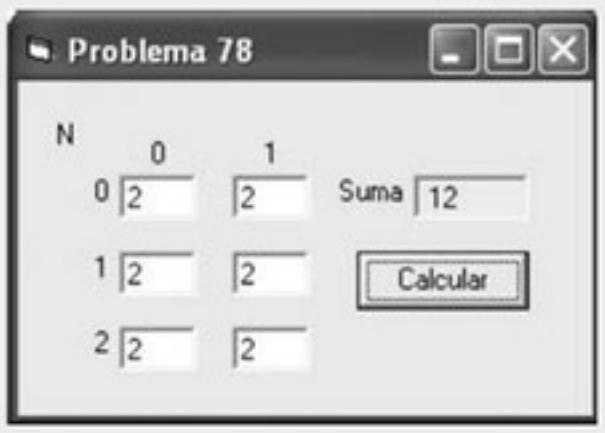
**Enunciado:** Ingrese 6 números en un arreglo de dos dimensiones (matriz) de 3x2, y obtenga la suma de los números ingresados.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 6 números; luego, que el sistema devuelva la suma de los números.

| Entrada | Salida |
|---|---|
| • 6 números (n[3][2]) | • Suma (s) |

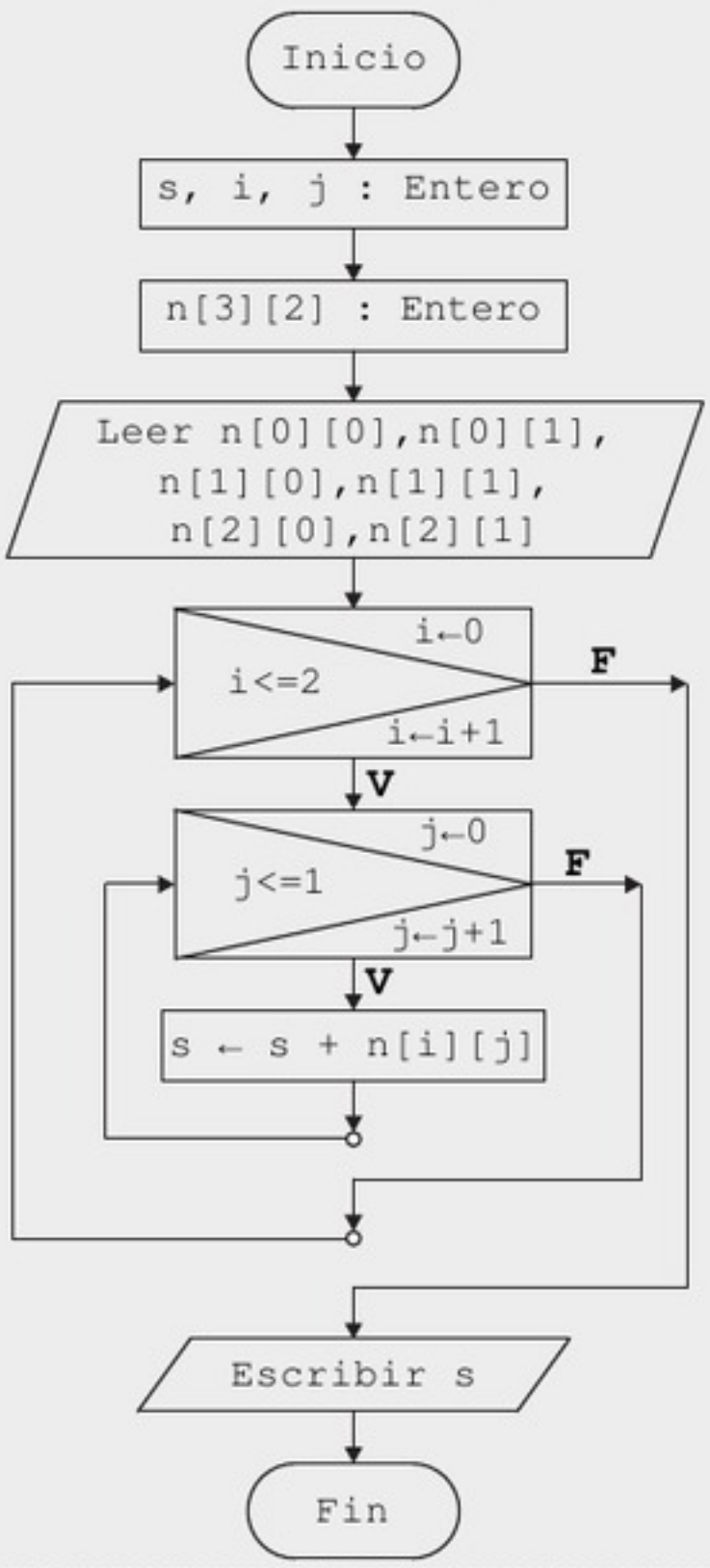
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   s,i,j : Entero

   //Arreglos (Matriz)
   n[3][2] : Entero

   //Entrada
   Leer n[0][0],n[0][1],
        n[1][0],n[0][1],
        n[2][0],n[0][1],

   //Proceso
   Para i←0 Hasta 2 Inc 1
       Para j←0 Hasta 1 Inc 1
          s ← s + n[i][j]
       Fin Para
   Fin Para

   //Salida
   Escribir s

Fin
```

**Codificación:**

```
'Variables
Dim s As Integer
Dim i As Integer
Dim j As Integer

'Arreglos
Dim n(2, 1) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)

'Proceso
For i = 0 To 2
    For j = 0 To 1
        s = s + n(i, j)
    Next
Next

'Salida
Me.txts.Text = Str(s)
```

## Problema n.° 79

**Enunciado:** Ingrese 12 números en un arreglo bidimensional (matriz) de 4x3, y obtenga la suma de cada columna.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 12 números; luego, que el sistema devuelva la suma de cada columna.

**Entrada**

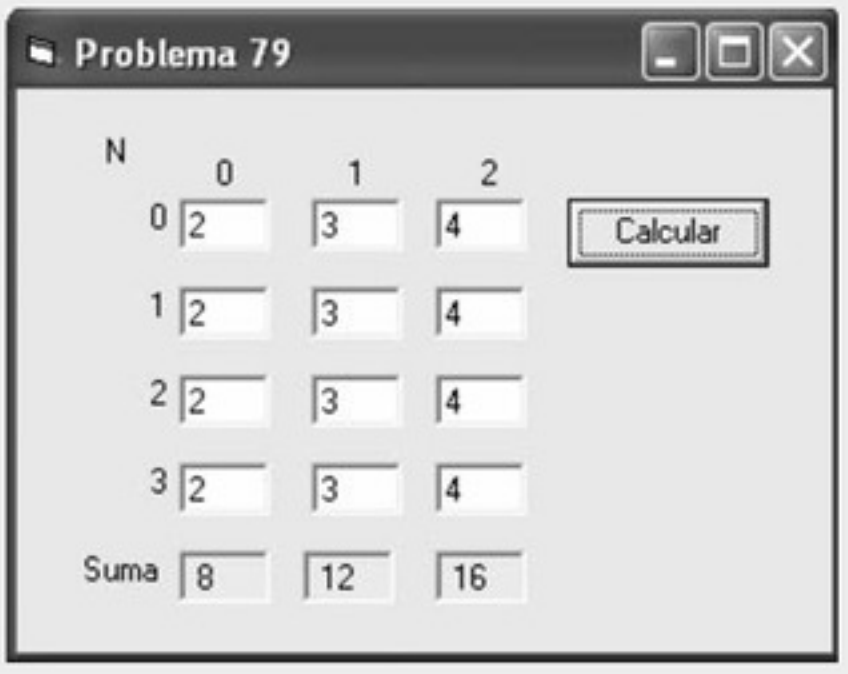
- 12 números (n[4][3])

**Salida**

- Suma columna 1 (s0)
- Suma columna 2 (s1)
- Suma columna 3 (s2)

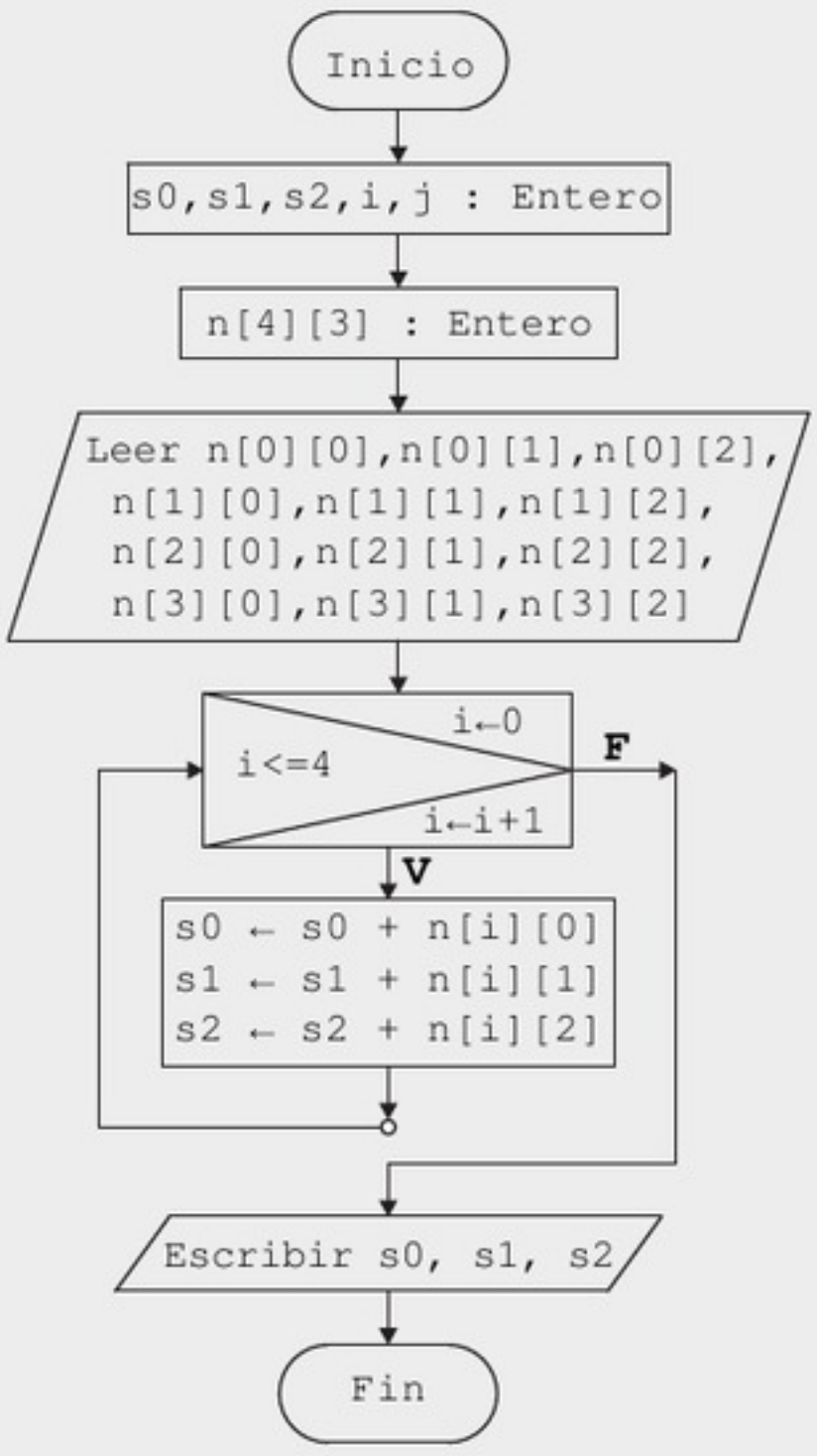
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   s0,s1,s2,i,j : Entero

   //Arreglos (Matriz)
   n[4][3] : Entero

   //Entrada
   Leer n[0][0],n[0][1],n[0][2],
        n[1][0],n[1][1],n[1][2],
        n[2][0],n[2][1],n[2][2],
        n[3][0],n[3][1],n[3][2],

   //Proceso
   Para i←0 Hasta 4 Inc 1
        s0 ← s0 + n[i][0]
        s1 ← s1 + n[i][1]
        s2 ← s2 + n[i][2]
   Fin Para

   //Salida
   Escribir s0, s1, s2

Fin
```

**Codificación:**

```
'Variables
Dim s0 As Integer
Dim s1 As Integer
Dim s2 As Integer
Dim i As Integer
Dim j As Integer

'Arreglos
Dim n(3, 2) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(0, 2) = Val(Me.txtn02.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(1, 2) = Val(Me.txtn12.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)
n(2, 2) = Val(Me.txtn22.Text)
n(3, 0) = Val(Me.txtn20.Text)
n(3, 1) = Val(Me.txtn21.Text)
n(3, 2) = Val(Me.txtn22.Text)

'Proceso
For i = 0 To 3
    s0 = s0 + n(i, 0)
    s1 = s1 + n(i, 1)
    s2 = s2 + n(i, 2)
Next

'Salida
Me.txts0.Text = Str(s0)
Me.txts1.Text = Str(s1)
Me.txts2.Text = Str(s2)
```

## Problema n.° 80

**Enunciado:** Almacene en una matriz de 3x2, 6 números y obtenga la cantidad de pares e impares.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 6 números; luego, que el sistema devuelva la cantidad de pares e impares.

**Entrada**

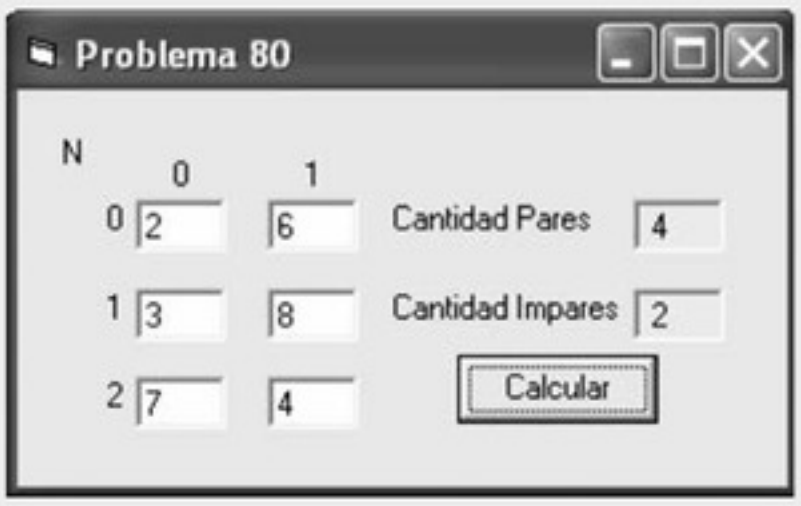
- 6 número (n[3][2])

**Salida**

- Cantidad de pares (cp)
- Cantidad de impares (ci)

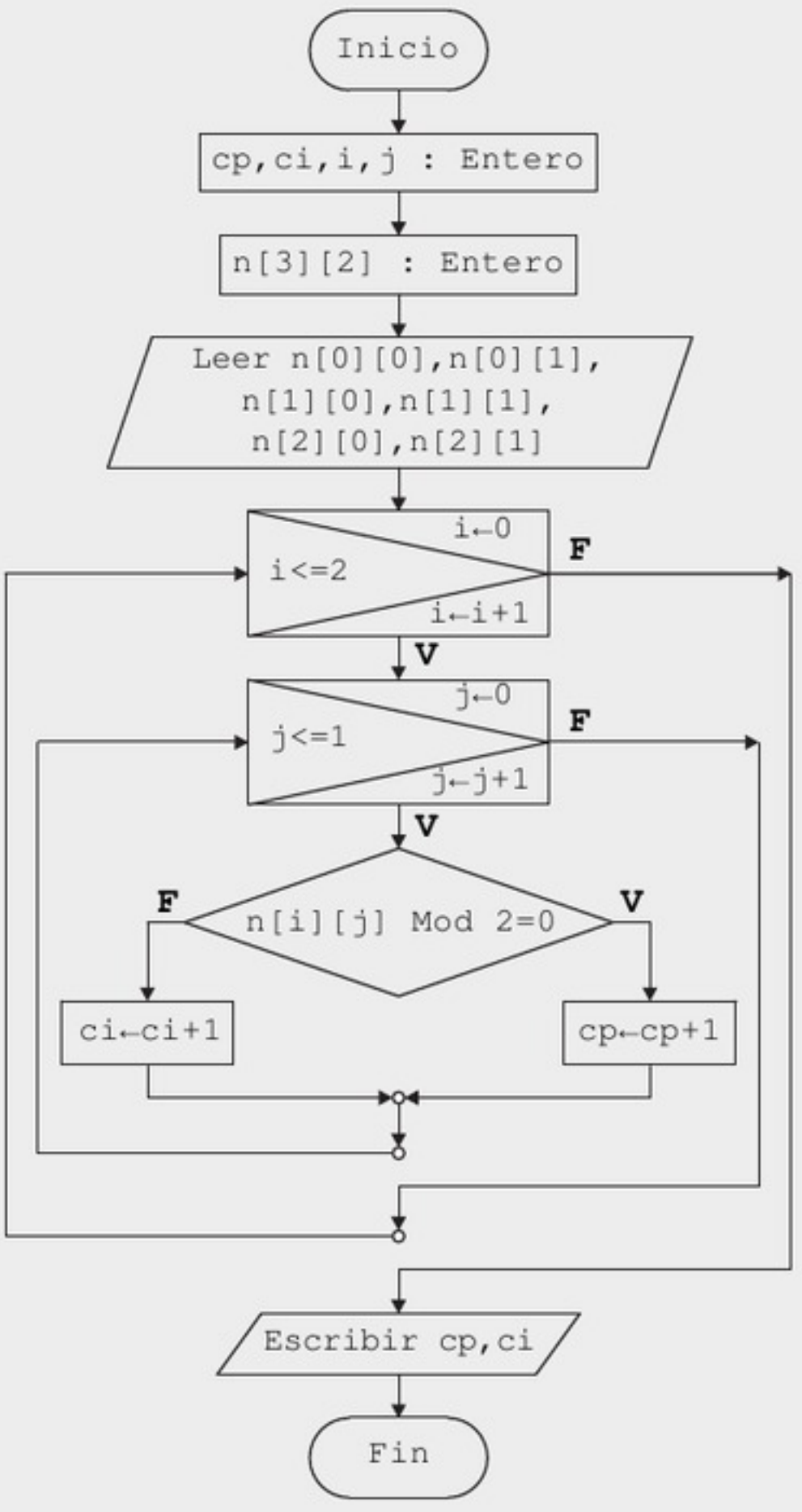
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   cp,ci,i,j : Entero

   //Arreglos (Matriz)
   n[3][2] : Entero

   //Entrada
   Leer n[0][0],n[0][1],
        n[1][0],n[1][1],
        n[2][0],n[2][1]

   //Proceso
   Para i←0 Hasta 2 Inc 1
       Para j←0 Hasta 1 Inc 1
      Si n[i][j] Mod 2=0 Entonces
       cp ← cp + 1
     SiNo
       ci ← ci + 1
     Fin Si
       Fin Para
   Fin Para

   //Salida
   Escribir cp, ci

Fin
```

**Codificación:**

```
'Variables
Dim cp As Integer
Dim ci As Integer
Dim i As Integer
Dim j As Integer

'Arreglos (Matriz)
Dim n(2, 1) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)

'Proceso
For i = 0 To 2
     For j = 0 To 1
          If n(i, j) Mod 2 = 0 Then
               cp = cp + 1
          Else
               ci = ci + 1
          End If
     Next
Next

'Salida
Me.txtcp.Text = Str(cp)
Me.txtci.Text = Str(ci)
```

## Problema n.° 81

**Enunciado:** Busque un número dentro de una matriz de 4x3 y determine la posición, y si existe o no el número buscado. Use el método de búsqueda secuencial.

**Análisis:** Para la solución de este problema se requiere que el usuario ingrese dos números; luego, que el sistema devuelva la cantidad de números positivos y negativos.

**Entrada**

- Matriz (n[4][3])
- Número a buscar (nb)

**Salida**

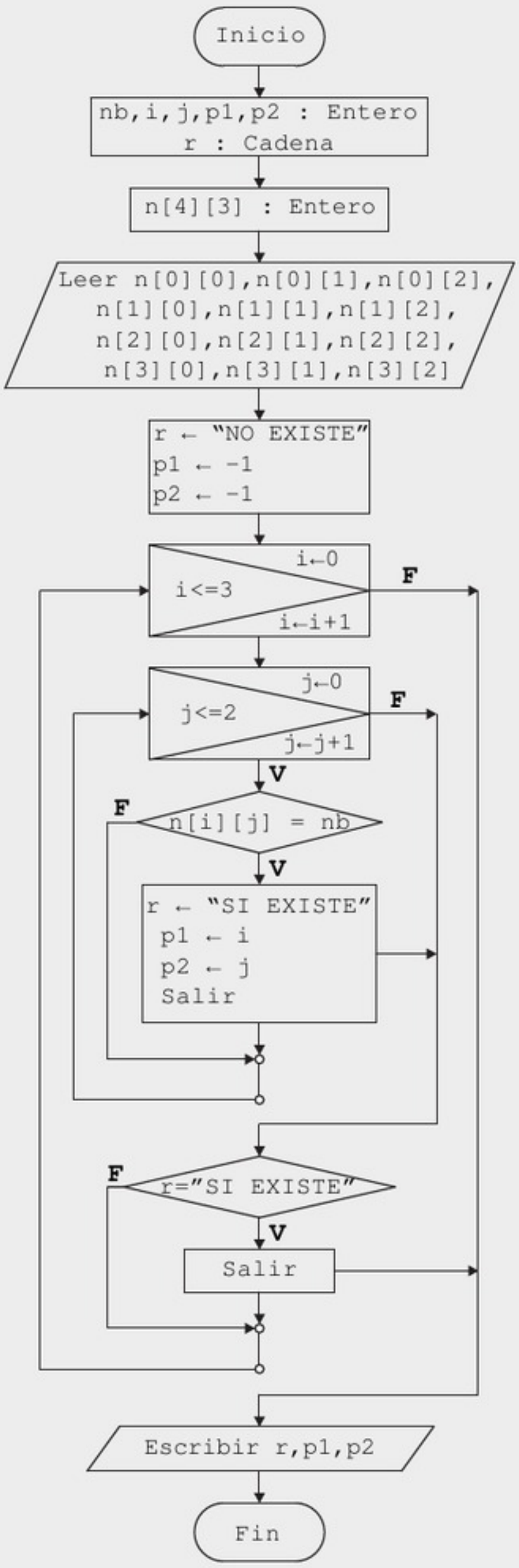
- Respuesta (r)
- Posición 1.ra dim. (p1)
- Posición 2.da dim. (p2)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   nb, i, j, p1, p2 : Entero
   r : Cadena

   //Arreglos (Matriz)
   n[4][3] : Entero

   //Entrada
   Leer n[0][0],n[0][1],n[0][2],
        n[1][0],n[1][1],n[1][2],
        n[2][0],n[2][1],n[2][2],
        n[3][0],n[3][1],n[3][2]

   //Proceso
   r ← "NO EXISTE"
   p1 ← -1
   p2 ← -1
   Para i←0 Hasta 3 Inc 1
       Para j←0 Hasta 2 Inc 1
          Si n[i][j]=nb Entonces
              r ← "SI EXISTE"
              p1 ← i
              p2 ← j
              Salir
          Fin Si
       Fin Para
       Si r="SI EXISTE" Entonces
          Salir
       Fin Si
   Fin Para

   //Salida
   Escribir r, p1, p2

Fin
```

**Codificación:**

```
'Variables
Dim nb As Integer
Dim i As Integer
Dim j As Integer
Dim r As String
Dim p1 As Integer
Dim p2 As Integer

'Arreglos
Dim n(3, 2) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(0, 2) = Val(Me.txtn02.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(1, 2) = Val(Me.txtn12.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)
n(2, 2) = Val(Me.txtn22.Text)
n(3, 0) = Val(Me.txtn30.Text)
n(3, 1) = Val(Me.txtn31.Text)
n(3, 2) = Val(Me.txtn32.Text)
nb = Val(Me.txtnb.Text)

'Proceso
r = "NO EXISTE"
p1 = -1
p2 = -1

For i = 0 To 3
    For j = 0 To 2
        If n(i, j) = nb Then
            r = "SI EXISTE"
            p1 = i
            p2 = j
            Exit For
        End If
    Next
    If r = "SI EXISTE" Then
        Exit For
    End If
Next

'Salida
Me.txtr.Text = r
Me.txtp1.Text = Str(p1)
Me.txtp2.Text = Str(p2)
```

## Problema n.° 82

**Enunciado:** Dada la matriz A de 2x2, la matriz B de 2x2, obtenga la suma de dichas matrices.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 8 números; luego, que el sistema devuelva la suma de matrices.

**Entrada**

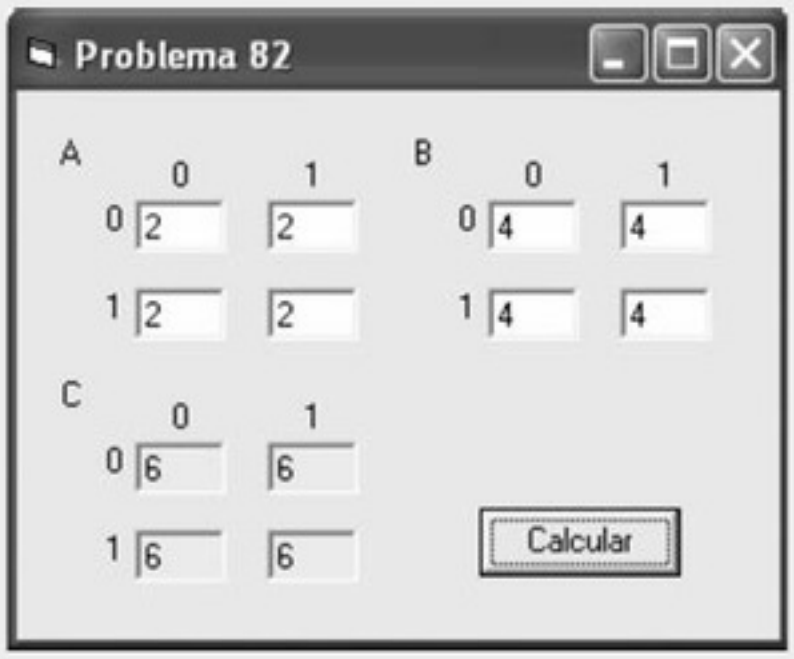
- 4 números matriz A (a[2][2])
- 4 números matriz B (b[2][2])

**Salida**

- 4 números matriz C (c[2][2])

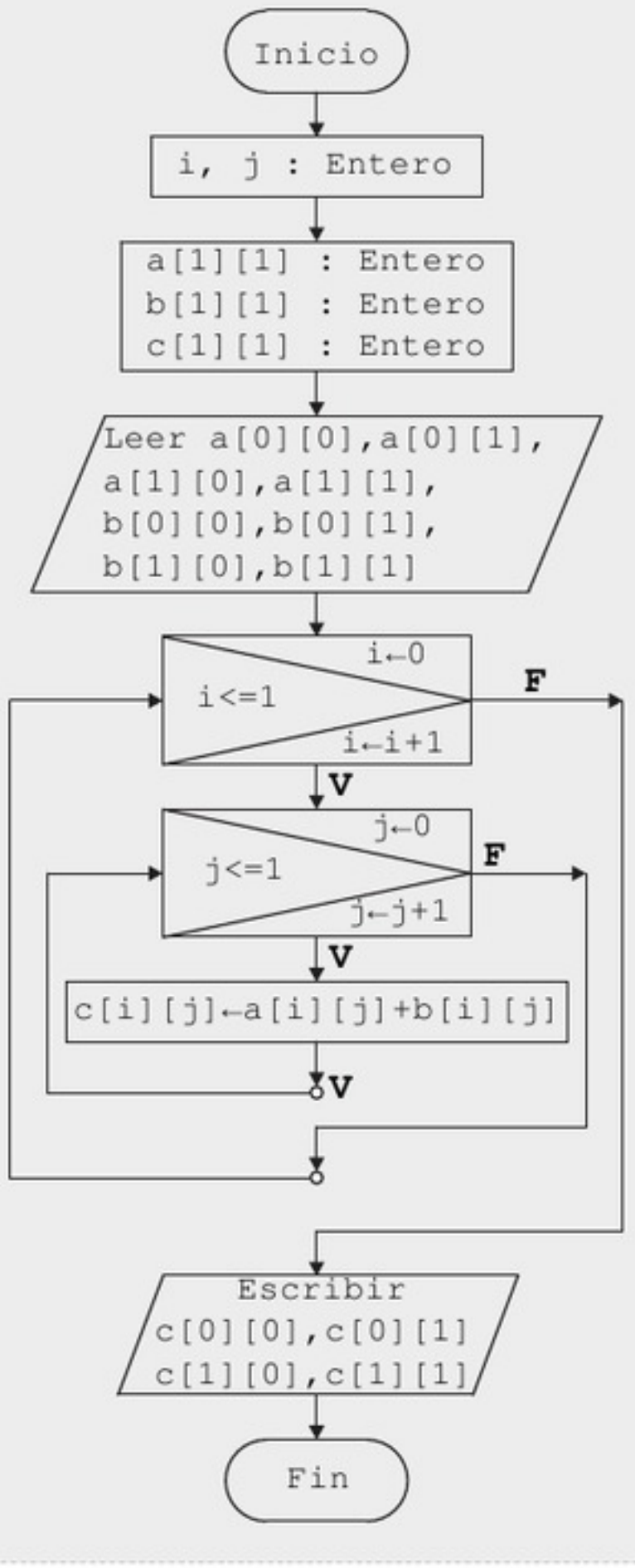
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   i, j : Entero

   //Arreglos (Matriz)
   a[1][1] : Entero
   b[1][1] : Entero
   c[1][1] : Entero

   //Entrada
   Leer a[0][0],a[0][1],
        a[1][0],a[1][1],
        b[0][0],b[0][1],
        b[1][0],b[1][1]

   //Proceso
   Para i←0 Hasta 1 Inc 1
        Para j←0 Hasta 1 Inc 1
       c[i][j]←a[i][j]+b[i][j]
        Fin Para
   Fin Para

   //Salida
   Escribir c[0][0],c[0][1],
            c[1][0],c[1][1]

Fin
```

**Codificación:**

```
'Variables
Dim i As Integer
Dim j As Integer

'Arreglos
Dim a(1, 1) As Integer
Dim b(1, 1) As Integer
Dim c(1, 1) As Integer

'Entrada (Matriz)
a(0, 0) = Val(Me.txta00.Text)
a(0, 1) = Val(Me.txta01.Text)
a(1, 0) = Val(Me.txta10.Text)
a(1, 1) = Val(Me.txta11.Text)

b(0, 0) = Val(Me.txtb00.Text)
b(0, 1) = Val(Me.txtb01.Text)
b(1, 0) = Val(Me.txtb10.Text)
b(1, 1) = Val(Me.txtb11.Text)

'Proceso
For i = 0 To 1
    For j = 0 To 1
        c(i, j) = a(i, j) + b(i, j)
    Next
Next

'Salida
Me.txtc00.Text = c(0, 0)
Me.txtc01.Text = c(0, 1)
Me.txtc10.Text = c(1, 0)
Me.txtc11.Text = c(1, 1)
```

## Problema n.° 83

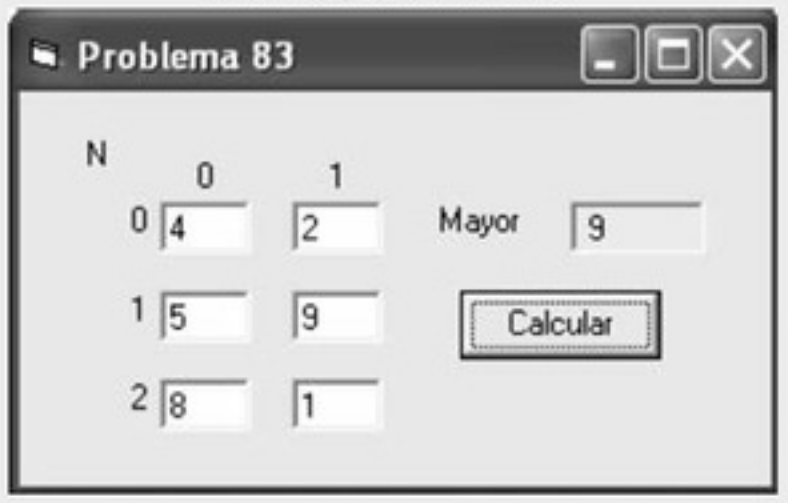
**Enunciado:** Ingrese 6 números en una matriz de 3x2, y obtenga el número mayor ingresado.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 6 números; luego, que el sistema devuelva el número mayor.

| Entrada | Salida |
|---|---|
| • 6 números (n[3][2]) | • Mayor (m) |

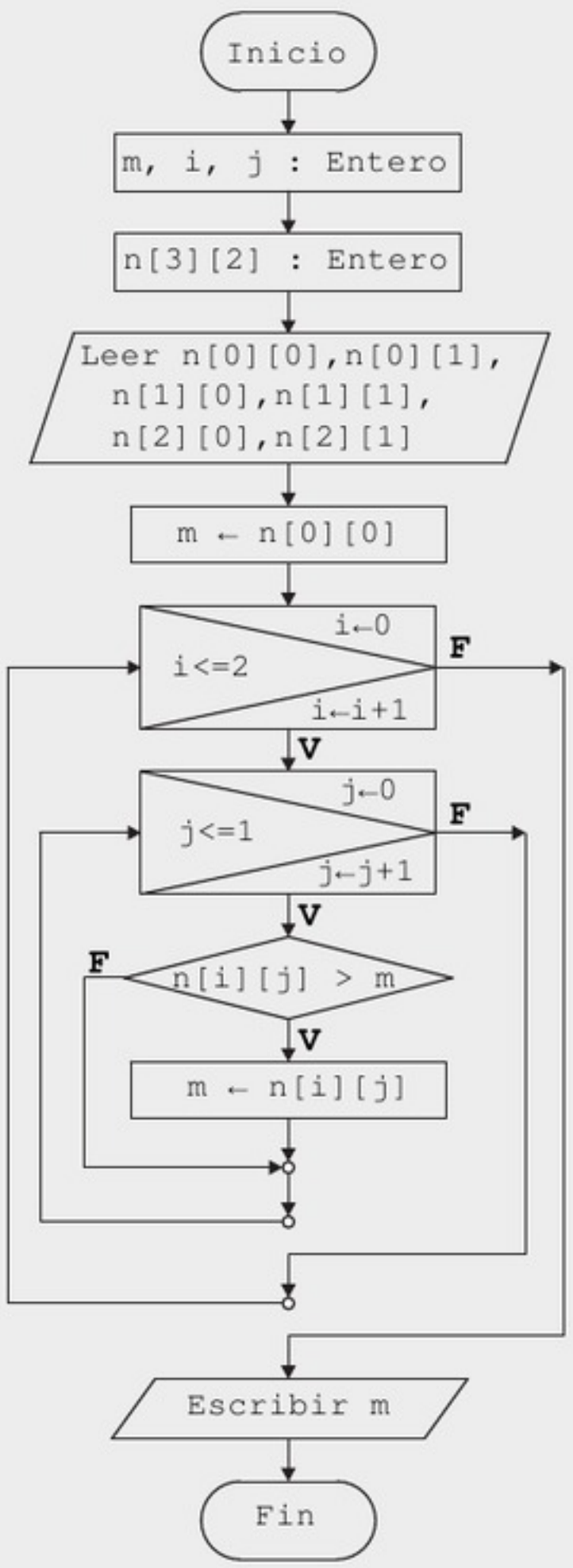
**Diseño:**

**Interfaz de usuario**




## Algoritmo

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   m, i, j : Entero

   //Arreglos (Matriz)
   n[3][2] : Entero

   //Entrada
   Leer n[0][0],n[0][1],
        n[1][0],n[1][1],
        n[2][0],n[2][1]

   //Proceso
   m ← n[0][0]
   Para i←0 Hasta 2 Inc 1
       Para j←0 Hasta 1 Inc 1
           Si n[i][j]>m Entonces
             m ← n[i][j]
           Fin Si
       Fin Para
   Fin Para

   //Salida
   Escribir m

Fin
```

**Codificación:**

```
'Variables
Dim m As Integer
Dim i As Integer
Dim j As Integer

'Arreglos
Dim n(2, 1) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)

'Proceso
m = n(0, 0)
For i = 0 To 2
    For j = 0 To 1
        If n(i, j) > m Then
            m = n(i, j)
        End If
    Next
Next

'Salida
Me.txtm.Text = Str(m)
```

## Problema n.° 84

**Enunciado:** Ingrese 6 números en una matriz de 3x2, y ordene los números de cada columna.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 6 números; luego, que el sistema devuelva las columnas ordenadas.

**Entrada**

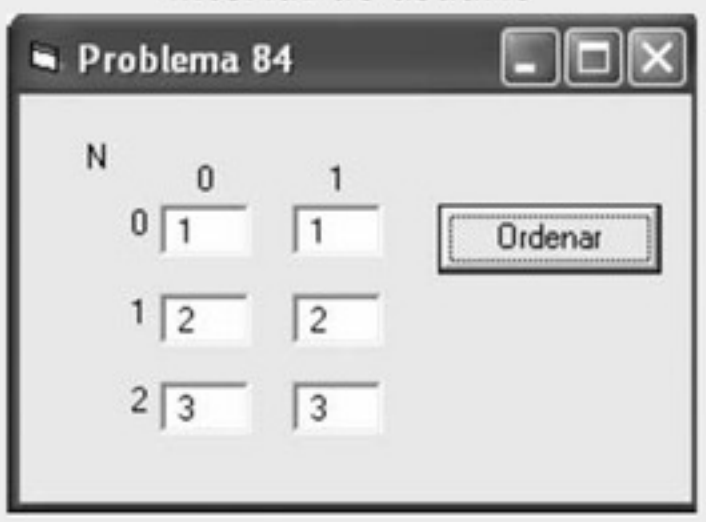
- 6 números (n[3][2])

**Salida**

- Cada columna ordenada (n[3][2])

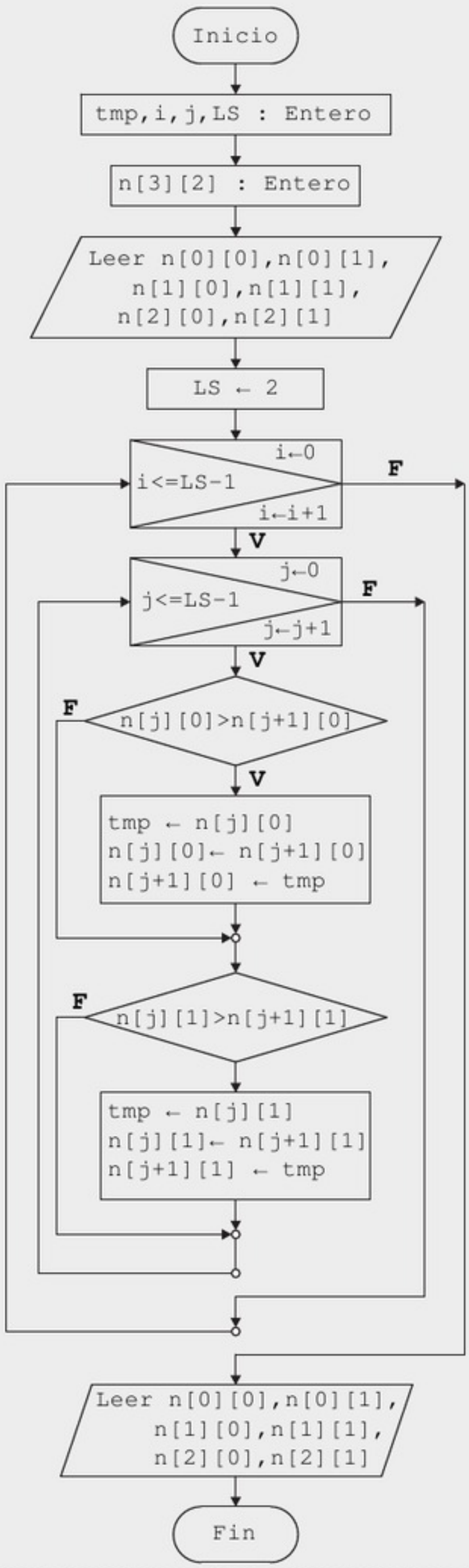
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   tmp,i,j,LS : Entero

   //Arreglos (Matriz)
   n[3][2] : Entero

   //Entrada
   Leer n[0][0],n[0][1],
        n[1][0],n[1][1],
        n[2][0],n[2][1]

   //Proceso
   LS ← 2
   Para i←0 Hasta LS-1 Inc 1
     Para j←0 Hasta LS-1 Inc 1

   Si n[j][0]>n[j+1][0] Entonces
        tmp ← n[j][0]
        n[j][0]← n[j+1][0]
        n[j+1][0] ← tmp
   Fin Si
   Si n[j][1]>n[j+1][1] Entonces
        tmp ← n[j][1]
        n[j][1]← n[j+1][1]
        n[j+1][1] ← tmp
   Fin Si
        Fin Para
   Fin Para
   //Salida
   Escribir n[0][0],n[0][1],
        n[1][0],n[1][1],
        n[2][0],n[2][1]

Fin
```

**Codificación:**

```
'Variables
Dim tmp As Integer
Dim i As Integer
Dim j As Integer
Dim LS As Integer

'Arreglos (Matriz)
Dim n(2, 1) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)

'Proceso
LS = 2
For i = 0 To LS - 1
    For j = 0 To LS - 1

        If n(j, 0) > n(j + 1, 0) Then
            tmp = n(j, 0)
            n(j, 0) = n(j + 1, 0)
            n(j + 1, 0) = tmp
        End If
        If n(j, 1) > n(j + 1, 1) Then
            tmp = n(j, 1)
            n(j, 1) = n(j + 1, 1)
            n(j + 1, 1) = tmp
        End If
    Next
Next

'Salida
Me.txtn00.Text = Str(n(0, 0))
Me.txtn01.Text = Str(n(0, 1))
Me.txtn10.Text = Str(n(1, 0))
Me.txtn11.Text = Str(n(1, 1))
Me.txtn20.Text = Str(n(2, 0))
Me.txtn21.Text = Str(n(2, 1))
```

## Problema n.° 85

**Enunciado:** Almacene 9 números en una matriz de 3x3, y obtenga los números ordenados.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese 9 números; luego, que el sistema devuelva la matriz con los números ordenados.

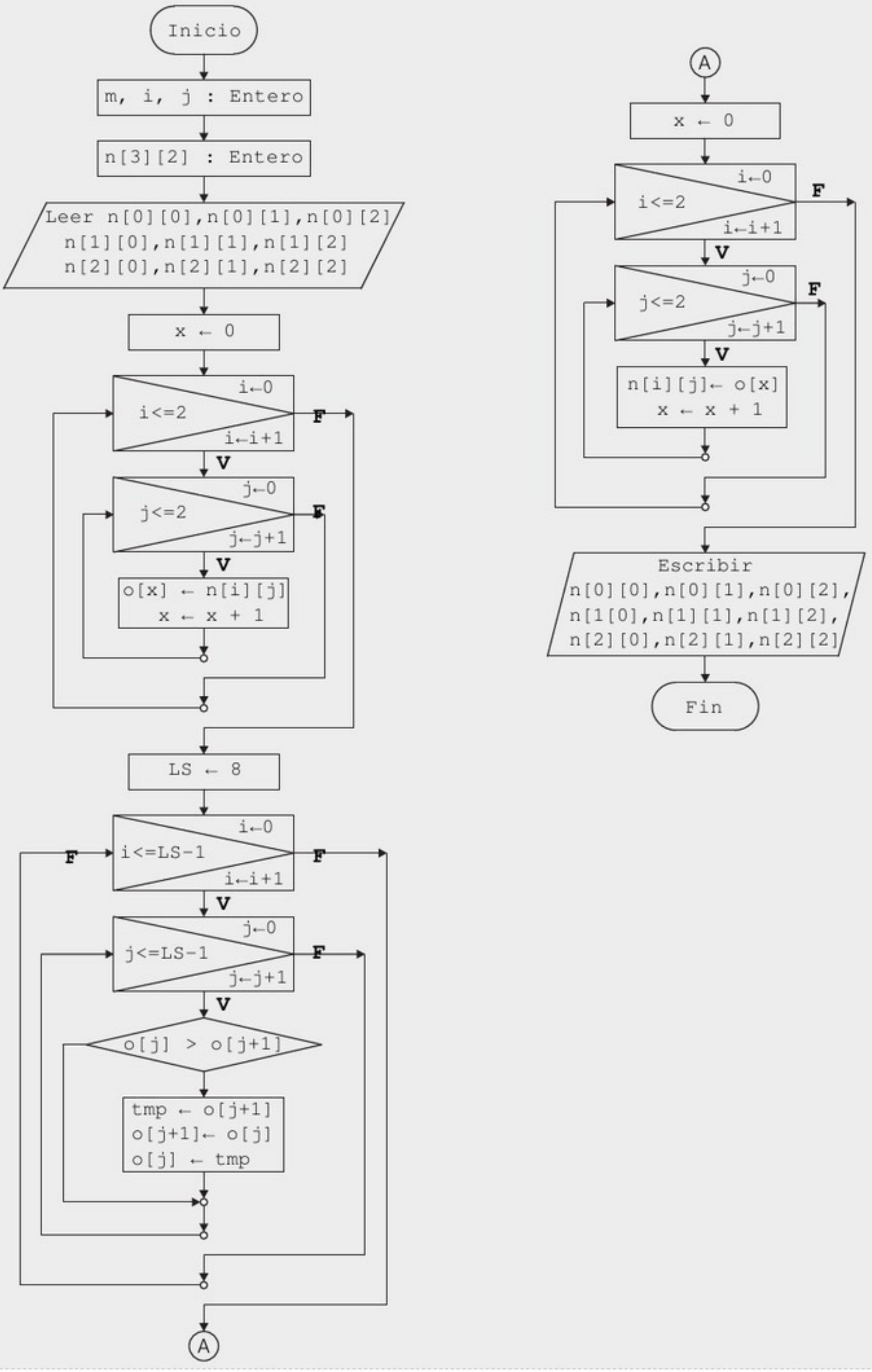
| Entrada | Salida |
|---|---|
| • 9 números (n[3][3]) | • 9 números ordenados (n[3][3]) |

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**



**Pseudocódigo**

```
Inicio

   //Variables
   tmp, i, j, x, LS : Entero
   flag : Logico

   //Arreglos (Matriz y Vector)
   n[3][3] : Entero
   O[9] : Entero

   //Entrada
   Leer n[0][0],n[0][1],n[0][2],
        n[1][0],n[1][1],n[1][2],
        n[2][0],n[2][1],n[2][2]

   //Proceso
   x ← 0
   Para i←0 Hasta 2 Inc 1
       Para j←0 Hasta 2 Inc 1
             o[x] ← n[i][j]
             x ← x + 1
       Fin Para
   Fin Para

   LS ← 8
   Para i←0 Hasta LS-1 Inc 1
      Para j←0 Hasta LS-1 Inc 1
         Si o[j] > o[j+1] Entonces
             tmp ← o(j + 1)
             o(j + 1) ← o(j)
             o(j) ← tmp
         Fin Si
       Fin Para
   Fin Para

   x ← 0
   Para i←0 Hasta 2 Inc 1
       Para j←0 Hasta 2 Inc 1
             n[i][j] ← o[x]
             x ← x + 1
       Fin Para
   Fin Para

   //Salida
   Escribir n[0][0],n[0][1],n[0][2],
       n[1][0],n[1][1],n[1][2],
       n[2][0],n[2][1],n[2][2]

Fin
```

**Codificación:**

```
'Variables
Dim tmp As Integer
Dim i As Integer
Dim j As Integer
Dim x As Integer
Dim LS As Integer

'Arreglos (Matriz)
Dim n(2, 2) As Integer
Dim o(8) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(0, 2) = Val(Me.txtn02.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(1, 2) = Val(Me.txtn12.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)
n(2, 2) = Val(Me.txtn22.Text)

'Proceso
x = 0
For i = 0 To 2
    For j = 0 To 2
        o(x) = n(i, j)
        x = x + 1
    Next
Next

LS = UBound(o, 1)
For i = 0 To LS - 1
    For j = 0 To LS - 1
        If o(j) > o(j + 1) Then
            tmp = o(j + 1)
            o(j + 1) = o(j)
            o(j) = tmp
        End If
    Next
```

```
Next

x = 0
For i = 0 To 2
     For j = 0 To 2
          n(i, j) = o(x)
          x = x + 1
     Next
Next

'Salida
Me.txtn00.Text = Str(n(0, 0))
Me.txtn01.Text = Str(n(0, 1))
Me.txtn02.Text = Str(n(0, 2))
Me.txtn10.Text = Str(n(1, 0))
Me.txtn11.Text = Str(n(1, 1))
Me.txtn12.Text = Str(n(1, 2))
Me.txtn20.Text = Str(n(2, 0))
Me.txtn21.Text = Str(n(2, 1))
Me.txtn22.Text = Str(n(2, 2))
```

## 7.6 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 51

**Enunciado:** Dados 4 números, almacénelos en un vector, luego obtenga la suma y el promedio de los valores almacenados.

### Propuesto n.° 52

**Enunciado:** Dados 4 números, almacénelos en un vector, el número mayor y menor.

### Propuesto n.° 53

**Enunciado:** Dado 6 números y almacénelo en un vector, luego obtenga cuantos números múltiplos de n ha ingresado.

### Propuesto n.° 54

**Enunciado:** Ordene 5 números según la forma que se indique: «A» (ascendente) o «D» (descendente).

### Propuesto n.° 55

**Enunciado:** Ingrese 6 números y determine cuántos números repetidos existen.

### Propuesto n.° 56

**Enunciado:** Ingrese 6 números en una matriz de 3x2 y obtenga la suma de cada fila.

### Propuesto n.° 57

**Enunciado:** Ingrese 6 números en una matriz de 3x2 y obtenga el promedio aritmético.

### Propuesto n.° 58

**Enunciado:** En una matriz de 2x3 ingrese 6 números, múltiplique su contenido por un valor K y obtenga la suma de los números de la matriz.

### Propuesto n.° 59

**Enunciado:** Cree una matriz «A» de 2x2, otra «B» de 2X2, y obtenga una matriz C =A * B.

### Propuesto n.° 60

**Enunciado:** Cree una matriz de 4x3 y obtenga los números mayores de cada columna.

# Capítulo 8

# Cadenas de caracteres

## 8.1 Introducción

Inicialmente, las computadoras fueron creadas con la finalidad de resolver problemas aritméticos; sin embargo, hoy en día el manejo de datos alfanuméricos (texto) es importante y de gran utilidad para el procesamiento de operaciones con caracteres (cadenas). Una cadena de caracteres es una secuencia de cero o más símbolos, que incluye letras del alfabeto, dígitos y caracteres especiales.

## 8.2 Juego de caracteres

Los lenguajes de programación utilizan un conjunto de caracteres para comunicarse con las computadoras, dentro de las cuales existen diferentes tipos de juego de caracteres, de los que destacan el ASCII, UNICODE, etc.

**Standard ASCII (Caracteres alfanuméricos)**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | ! | 49 | 1 | 65 | A | 81 | Q | 97 | a | 113 | q |
| 34 | “ | 50 | 2 | 66 | B | 82 | R | 98 | b | 114 | r |
| 35 | # | 51 | 3 | 67 | C | 83 | S | 99 | c | 115 | s |
| 36 | $ | 52 | 4 | 68 | D | 84 | T | 100 | d | 116 | t |
| 37 | % | 53 | 5 | 69 | E | 85 | U | 101 | e | 117 | u |
| 38 | & | 54 | 6 | 70 | F | 86 | V | 102 | f | 118 | v |
| 39 | ‘ | 55 | 7 | 71 | G | 87 | W | 103 | g | 119 | w |
| 40 | ( | 56 | 8 | 72 | H | 88 | X | 104 | h | 120 | x |
| 41 | ) | 57 | 9 | 73 | I | 89 | Y | 105 | i | 121 | y |
| 42 | * | 58 | : | 74 | J | 90 | Z | 106 | j | 122 | z |
| 43 | + | 59 | ; | 75 | K | 91 | [ | 107 | k | 123 | { |
| 44 | , | 60 | < | 76 | L | 92 | \ | 108 | l | 124 | \| |
| 45 | - | 61 | = | 77 | M | 93 | ] | 109 | m | 125 | } |
| 46 | . | 62 | > | 78 | N | 94 | ^ | 110 | n | 126 | ~ |
| 47 | / | 63 | ? | 79 | O | 95 | _ | 111 | o | 127 | ⌂ |
| 48 | 0 | 64 | @ | 80 | P | 96 | ` | 112 | p | | |

**Caracteres extendidos de ASCII**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 128 | € | 144 | • | 160 | | 176 | ° | 193 | Á | 209 | Ñ | 225 | á | 241 | ñ |
| 129 | • | 145 | ‘ | 161 | ¡ | 177 | ± | 194 | Â | 210 | Ò | 226 | â | 242 | ò |
| 130 | ‚ | 146 | ’ | 162 | ¢ | 178 | ² | 195 | Ã | 211 | Ó | 227 | ã | 243 | ó |
| 131 | *ƒ* | 147 | “ | 163 | £ | 179 | ³ | 196 | Ä | 212 | Ô | 228 | ä | 244 | ô |
| 132 | „ | 148 | ” | 164 | ¤ | 180 | ´ | 197 | Å | 213 | Õ | 229 | å | 245 | õ |
| 133 | … | 149 | • | 165 | ¥ | 181 | µ | 198 | Æ | 214 | Ö | 230 | æ | 246 | ö |
| 134 | † | 150 | – | 166 | ¦ | 182 | ¶ | 199 | Ç | 215 | × | 231 | ç | 247 | ÷ |
| 135 | ‡ | 151 | — | 167 | § | 183 | · | 200 | È | 216 | Ø | 232 | è | 248 | ø |
| 136 | ˆ | 152 | ˜ | 168 | ¨ | 184 | ¸ | 201 | É | 217 | Ù | 233 | é | 249 | ù |
| 137 | ‰ | 153 | ™ | 169 | © | 185 | ¹ | 202 | Ê | 218 | Ú | 234 | ê | 250 | ú |
| 138 | Š | 154 | š | 170 | ª | 186 | V | 203 | Ë | 219 | Û | 235 | ë | 251 | û |
| 139 | ‹ | 156 | œ | 171 | « | 187 | » | 204 | Ì | 220 | Ü | 236 | ì | 252 | ü |
| 140 | Œ | 157 | • | 172 | ¬ | 188 | ¼ | 205 | Í | 221 | Ý | 237 | í | 253 | ý |
| 141 | • | 158 | ž | 173 | | 189 | ½ | 206 | Î | 222 | Þ | 238 | î | 254 | þ |
| 142 | Ž | 159 | Ÿ | 174 | ® | 190 | ¾ | 207 | Ï | 223 | ß | 239 | ï | 255 | ÿ |
| 143 | • | 192 | À | 175 | ¯ | 191 | ¿ | 208 | Ð | 224 | à | 240 | ð | | |

## 8.3 Carácter (*char*)

Representa un solo valor de tipo carácter; por lo general, se representa con comillas simples.

**Pseudocódigo**

```
//Crear una variable caracter
c : Caracter

//Asignar un valor
c ← 'A'
```

**Visual Basic**

```
'Uno o más caracteres
Dim c As String

'Asignar un valor
c = "A"
```

## 8.4 Cadena de caracteres (*string*)

Representa un conjunto de caracteres y, por lo general, lo representamos entre comillas dobles.

**Pseudocódigo**

```
//Crear una variable cadena
c : Cadena

//Asignar un valor
c ← "ABC"
```

**Visual Basic**

```
'Uno o más caracteres
Dim c As String

'Asignar un valor
c = "ABC"
```

## 8.5 Operaciones con cadena

Para la manipulación de las cadenas, los lenguajes de programación incorporan una variedad de funciones y/o métodos que permiten realizar operaciones con cadenas.

Las operaciones con cadenas más usadas son:

- Concatenación
- Comparación
- Cálculo de longitud
- Extracción de cadenas (subcadenas)
- Búsqueda de cadenas
- Conversiones

## 8.6 Concatenación

Unir varias cadenas en una sola.

**Pseudocódigo**

```
    //Unir cadenas
    c ← "ABC" + "XYZ"
```

**Visual Basic**

```
'Unir cadenas
c = "ABC" & "XYZ"
c = "ABC" + "XYZ"
```

## 8.7 Comparación

Igualdad y desigualdad de cadenas.

**Pseudocódigo**

```
//Igualdad (Falso)
"AAA" = "aaa"

//Desigualdad (Verdadero)
"LUISA" > "LUIS"
```

**Visual Basic**

```
'Igualdad (Falso)
"AAA" = "aaa"

'Desigualdad (Verdadero)
"LUISA" > "LUIS"
```

## 8.8 Cálculo de longitud

Obtener la cantidad de caracteres de una cadena.

**Pseudocódigo**

```
//Retorna 3
l ← Longitud("aaa")
```

**Visual Basic**

```
'Retorna 3
l = Len("aaa")
```

## 8.9 Extracción de cadenas (subcadenas)

Extraer una parte especifica de la cadena, por lo general cada carácter de una cadena se representa por una posición que inicia con 0, es decir "JUAN" consta de 4 caracteres J es el primer carácter cuya posición es 0, U segundo carácter posición 1, así sucesivamente.

En Visual Basic las posiciones de los caracteres de una cadena inician con 1.

**Pseudocódigo**

```
//Extraer el primer caracter A
// 1 cantidad a extraer
c ← Izquierda("ABC",1)

//También se usa
// 0 posicion
// 1 cantidad a extraer
c ← subcadena("ABC",0,1)

//Extraer el último caracter C
// 1 cantidad a extraer
c ← Derecha("ABC",1)

//También se usa
// 2 posicion
// 1 cantidad a extraer
c ← subcadena("ABC",2,1)

//Extraer el segundo caracter B
// 2 posicion
// 1 cantidad a extraer
c ← Extraer("ABC",1,1)
c ← subcadena("ABC",1,1)
```

**Visual Basic**

```
'Extraer el primer caracter A
c = Left("ABC",1)

'También se usa
' 1 posicion
' 1 cantidad a extraer
c =  Mid("ABC",1,1)
```

```
'Extraer el último caracter C
c = Right("ABC",1)
'También se usa
' 3 posicion
' 1 cantidad a extraer
c =  Mid("ABC",3,1)

'Extraer el segundo caracter B
' 2 posicion
' 1 cantidad a extraer
c = Mid("ABC",2,1)
```

## 8.10 Búsqueda de cadenas

Buscar si una cadena se encuentra dentro de otra cadena mas grande, si fuera así la función devuelve la posición de la cadena encontrada caso contrario retorna-1.

En Visual Basic la función que cumple la tarea de buscar retorna 0 si no encuentra la otra cadena, porque las cadenas inician con posición 1.

**Pseudocódigo**

```
//Retorna 1
p ←  Indice("ABC", "B")
```

**Visual Basic**

```
'Retorna 2
p = InStr("ABC","B")
```

## 8.11 Conversiones

Convertir números a cadena o cadena de números a números, es una tarea frecuente en los lenguajes de programación, así como convertir a mayúscula o minúscula una cadena de caracteres u obtener el valor ASCII de un carácter, o devolver el carácter de un código ASCII.

**Pseudocódigo**

```
//Convertir a cadena un número, de 32 a "32"
c ←  Cadena(32)

//Convertir una cadena de números a dato numérico, de "32" a 32
n ←  Valor("32")
```

```
//Convertir a mayúscula (ABC)
c ←  Mayus("abc")

//Convertir a minúscula (abc)
c ←  Minus("ABC")

//Obtener el valor ASCII (A = 65)
c ←  Codigo("A")

//Obtener el carácter de un código ASCII (65 = A)
c ←  Caracter(65)
```

**Visual Basic**

```
'Convertir a cadena un número, de 32 a "32"
c = Str(32)

'Convertir una cadena de números a dato numérico, de "32" a 32
n = Val("32")

'Convertir a mayúscula (ABC)
c = UCase("abc")

'Convertir a minúscula (abc)
c = LCase("ABC")

'Obtener el valor ASCII (A = 65)
c = Asc("A")

'Obtener el carácter de un código ASCII (65 = A)
c = Chr(65)
```

## Problema n.° 86

**Enunciado:** Dado un nombre, obtener la cantidad de caracteres que contiene.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese una cadena de caracteres; luego, que el sistema devuelva la cantidad de caracteres que contiene.

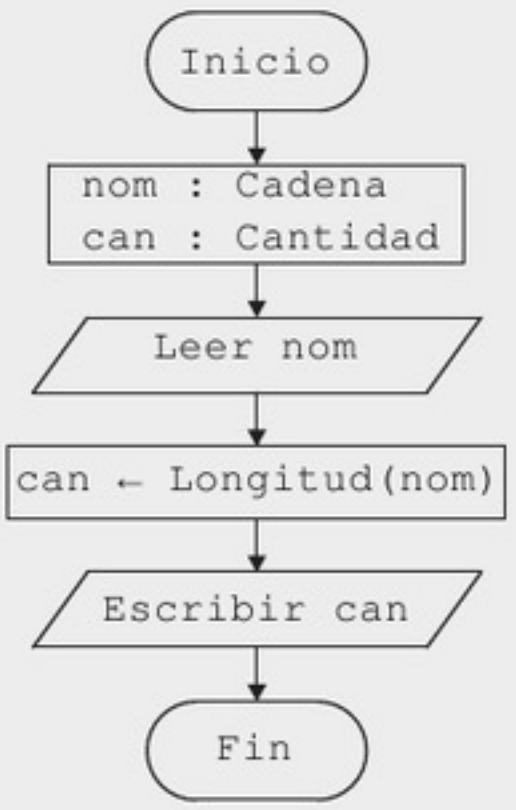
| Entrada | Salida |
|---|---|
| • Cadena de caracteres (nom) | • Cantidad (can) |

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   nom : Cadena
   can : Entero

   //Entrada
   Leer nom

   //Proceso
   can ← Longitud(nom)

   //Salida
   Escribir can

Fin
```

**Codificación:**

```
'Variables
Dim nom As String
Dim can As Integer

'Entrada
nom = Me.txtnom.Text

'Proceso
can = Len(nom)

'Salida
Me.txtcan.Text = Str(can)
```

## Problema n.° 87

**Enunciado:** Ingrese su nombre y apellido, y obtenga su nombre y apellido en mayúscula separado por una coma XXXXX, XXXXX.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese su nombre y apellido; luego, que el sistema devuelva su nombre y apellido separado por una coma y en mayúscula.

**Entrada**

- Nombre (nom)
- Apellido (ape)

**Salida**

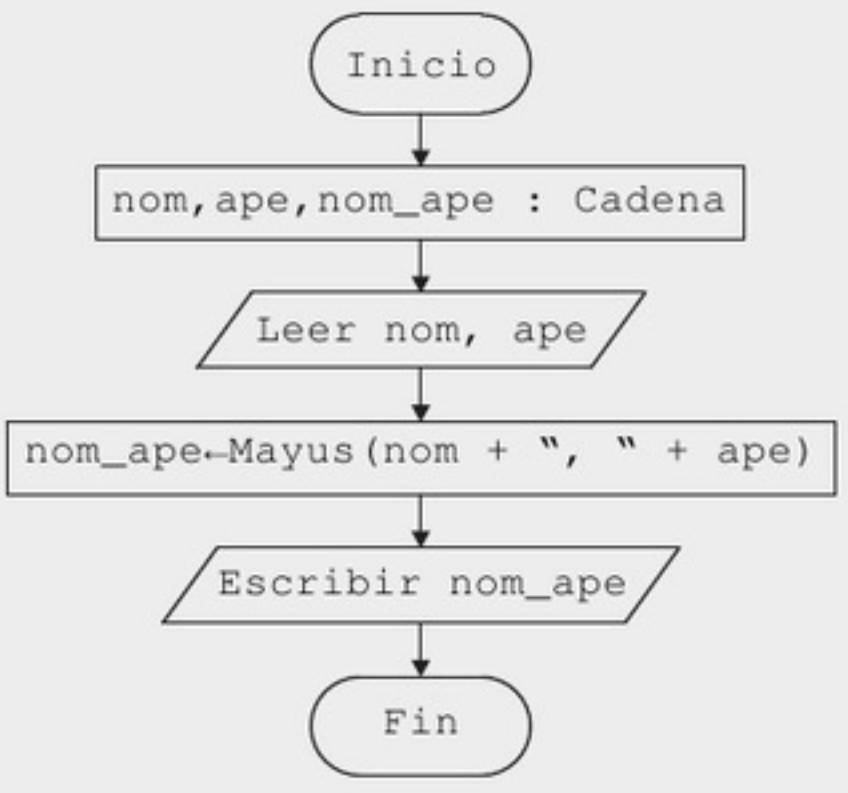
- Nombre y apellido (nom_ape)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   nom,ape,nom_ape : Cadena

   //Entrada
   Leer nom, ape

   //Proceso
   nom_ape←Mayus(nom + ", " + ape)

   //Salida
   Escribir nom_ape

Fin
```

**Codificación:**

```
'Variables
Dim nom As String
Dim ape As String
Dim nom_ape As String

'Entrada
nom = Me.txtnom.Text
ape = Me.txtape.Text

'Proceso
nom_ape = UCase(nom & ", " & ape)

'Salida
Me.txtnomape.Text = nom_ape
```

## Problema n.° 88

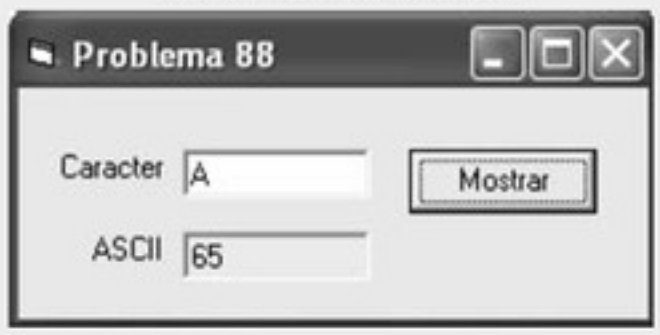
**Enunciado:** Dado un carácter, devolver su código ASCII.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un carácter; luego, que el sistema devuelva el ASCII.

| Entrada | Salida |
|---|---|
| • Carácter (c) | • ASCII (a) |

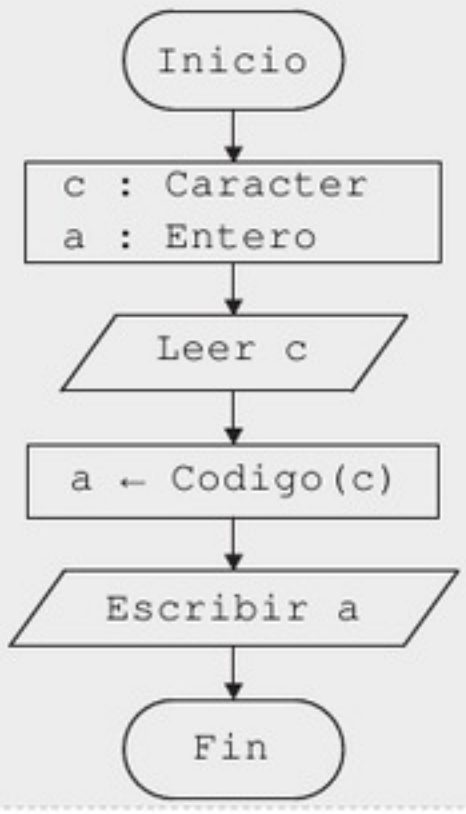
**Diseño:**

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c : Caracater
   a : Entero

   //Entrada
   Leer c

   //Proceso
   a ← Codigo(c)

   //Salida
   Escribir a

Fin
```

**Codificación:**

```
'Variables
Dim c As String
Dim a As Integer

'Entrada
c = Me.txtc.Text

'Proceso
a = Asc(c)

'Salida
Me.txta.Text = a
```

## Problema n.° 89

**Enunciado:** Al ingresar una letra, determine si es una vocal.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese una letra; luego, que el sistema devuelva si es o no una vocal.

**Entrada**

- Letra (l)

**Salida**

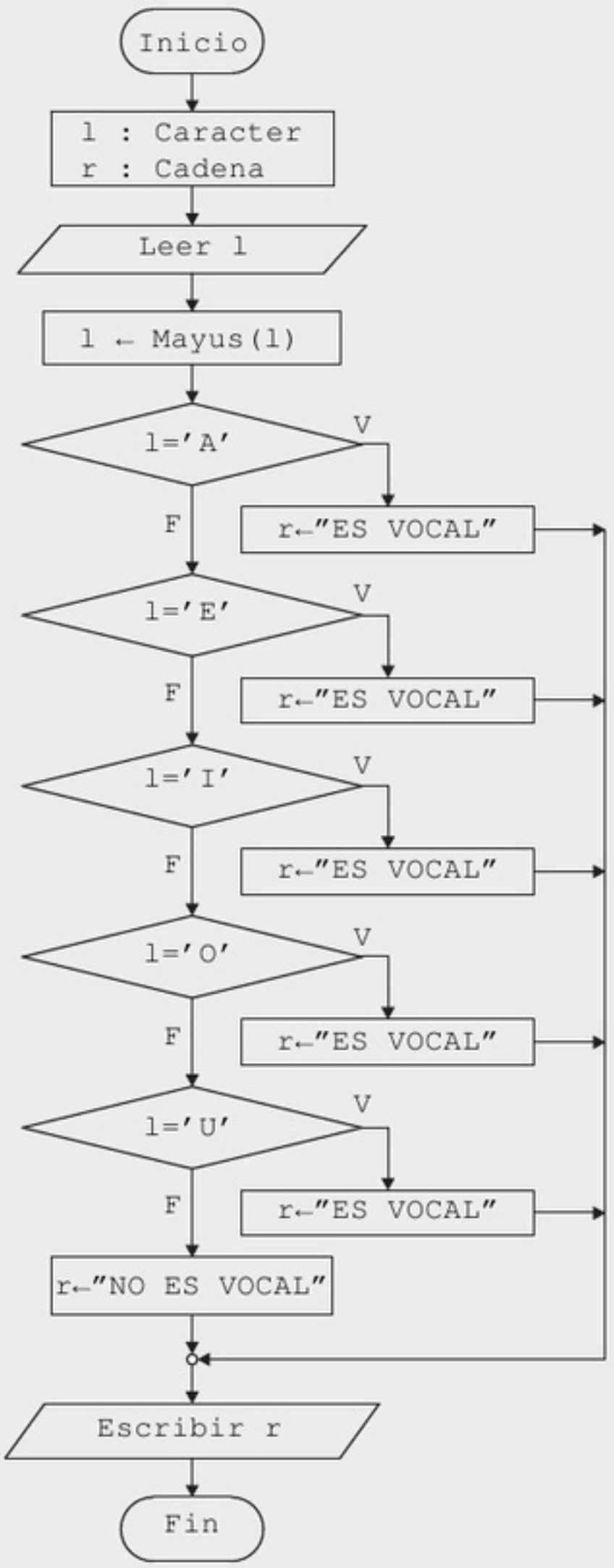
- Respuesta (r)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   l : Caracter
   r : Cadena

   //Entrada
   Leer l

   //Proceso
   l ← Mayus(l)
   Si l='A' Entonces
       R ← "ES VOCAL"
   SiNoSi l='E' Entonces
       R ← "ES VOCAL"
   SiNoSi l='I' Entonces
       R ← "ES VOCAL"
   SiNoSi l='O' Entonces
       R ← "ES VOCAL"
   SiNoSi l='U' Entonces
       R ← "ES VOCAL"
   SiNo
       R ← "NO ES VOCAL"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim l As String
Dim r As String

'Entrada
l = Me.txtl.Text

'Proceso
l = UCase(l)
If l = "A" Then
     r = "ES VOCAL"
ElseIf l = "E" Then
     r = "ES VOCAL"
ElseIf l = "I" Then
     r = "ES VOCAL"
ElseIf l = "O" Then
     r = "ES VOCAL"
ElseIf l = "U" Then
     r = "ES VOCAL"
Else
     r = "NO ES VOCAL"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 90

**Enunciado:** Dado un carácter, determine si es una letra, número o símbolo.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un carácter; luego, que el sistema devuelva si es letra, número o símbolo.

**Entrada**

- Caracter (c)

**Salida**

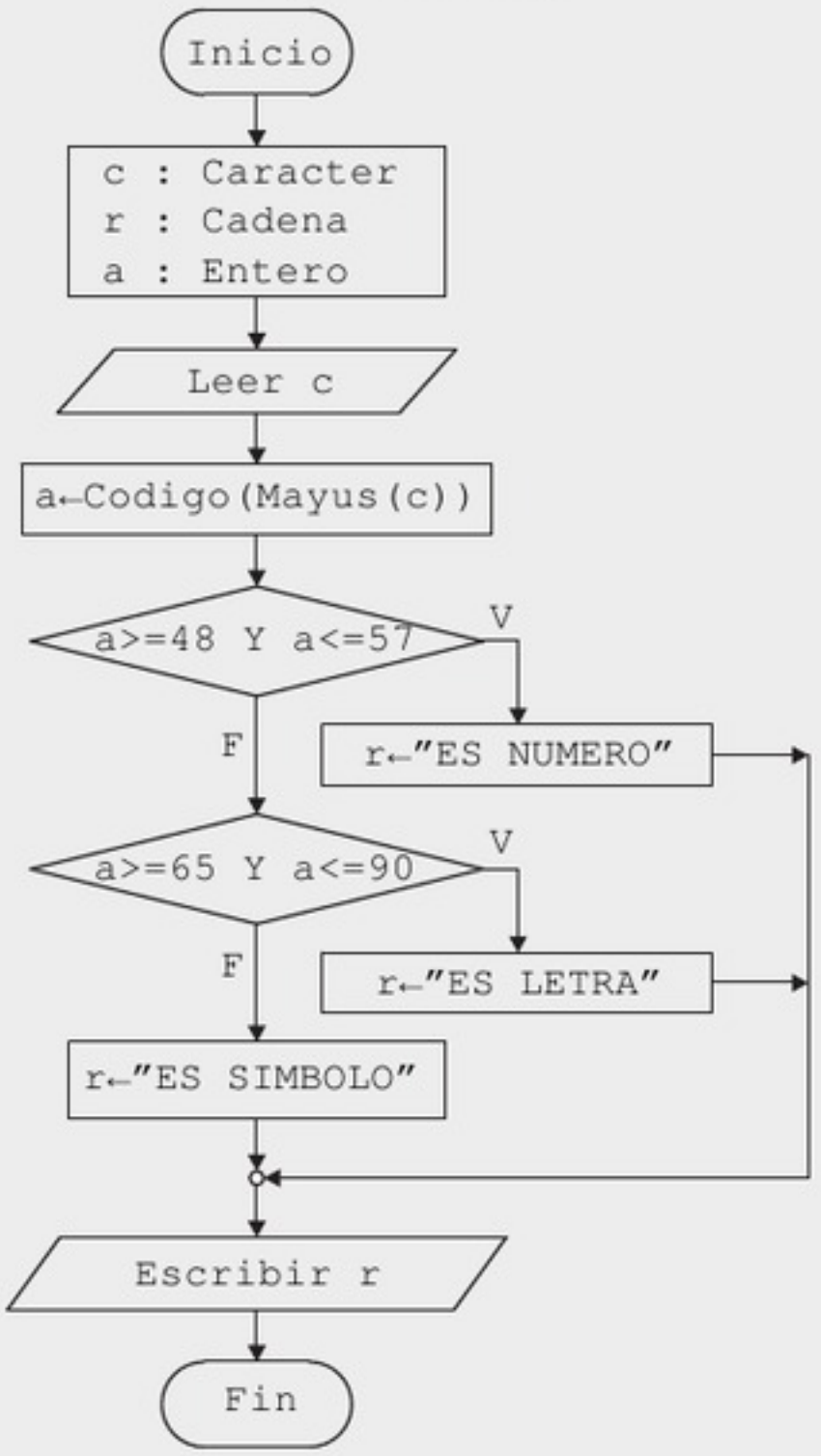
- Respuesta (r)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   c : Caracter
   r : Cadena
   a : Entero

   //Entrada
   Leer c

   //Proceso
   a ← Codigo(Mayus(c))
   Si a>=48 Y a<=57 Entonces
       r ← "ES NUMERO"
   SiNoSi a>=65 Y a<=90 Entonces
       r ← "ES LETRA"
   SiNo
       r ← "ES SIMBOLO"
   Fin Si

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim c As String
Dim r As String
Dim a As Integer

'Entrada
c = Me.txtc.Text

'Proceso
a = Asc(UCase(c))
If a >= 48 And a <= 57 Then
     r = "ES NUMERO"
ElseIf a >= 65 And a <= 90 Then
     r = "ES LETRA"
Else
     r = "ES SIMBOLO"
End If

'Salida
Me.txtr.Text = r
```

## Problema n.° 91

**Enunciado:** Se desea obtener los N primeros caracteres de un nombre.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un nombre y una cantidad; luego, que el sistema devuelva los primeros caracteres indicados por la cantidad.

**Entrada**

- Nombre (n)
- Cantidad (c)

**Salida**

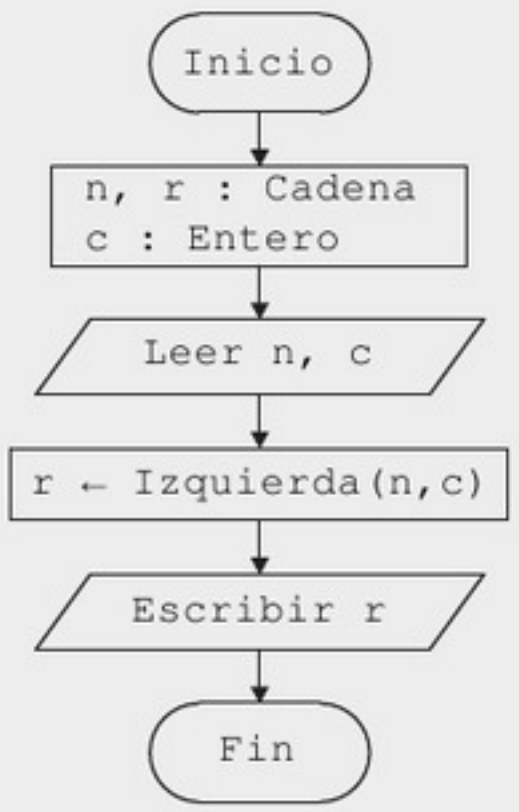
- Respuesta (r)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   n, r : Cadena
   c : Entero

   //Entrada
   Leer n, c

   //Proceso
   r ← Izquierda(n,c)

   //Salida
   Escribir r

Fin
```

**Codificación:**

```
'Variables
Dim n As String
Dim c As Integer
Dim r As String

'Entrada
n = Me.txtn.Text
c = Val(Me.txtc.Text)

'Proceso
r = Left(n, c)

'Salida
Me.txtr.Text = r
```

## Problema n.° 92

**Enunciado:** Según las siguientes especificaciones, genere un código basado en el nombre ingresado.

Especificaciones para generar el código:

1.er carácter del código: Primer carácter del nombre.

2.° carácter del código: Tercer carácter del nombre.

3.er carácter del código: Último carácter del nombre.

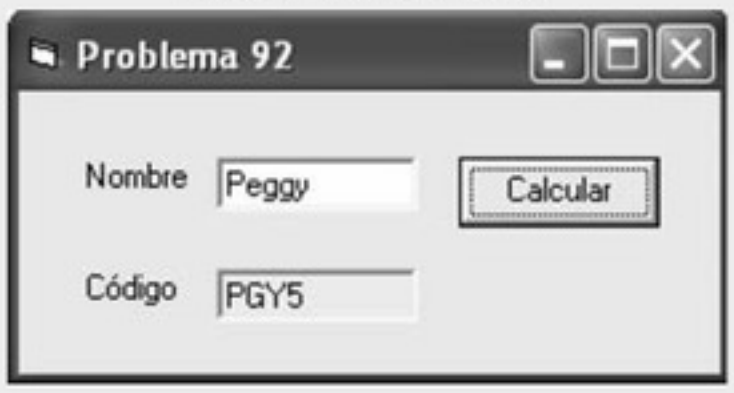
4.° carácter del código: Cantidad de caracteres del nombre..

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese un nombre; luego, que el sistema procese y obtenga el código generado.

| Entrada | Salida |
|---|---|
| • Nombre (n) | • Código (c) |

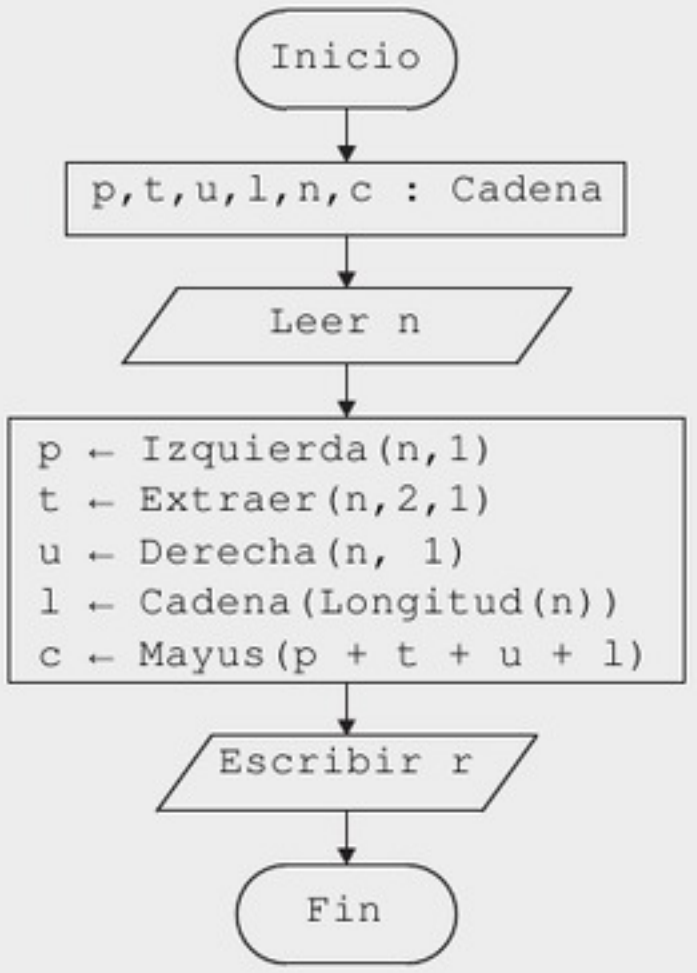
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

    //Variables
    p,t,u,l,n,c : Cadena

    //Entrada
    Leer n

    //Proceso
    p ← Izquierda(n,1)
    t ← Extraer(n,2,1)
    u ← Derecha(n, 1)
    l ← Cadena(Longitud(n))
    c ← Mayus(p + t + u + l)

    //Salida
    Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim p As String
Dim t As String
Dim u As String
Dim l As String
Dim n As String
Dim c As String

'Entrada
n = Me.txtn.Text

'Proceso
p = Left(n, 1)
t = Mid(n, 3, 1)
u = Right(n, 1)
l = Str(Len(n))
c = UCase(p & t & u & l)

'Salida
Me.txtc.Text = c
```

## Problema n.° 93

**Enunciado:** Determine cuántas veces se repite una letra en una frase dada.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese una frase y una letra; luego, que el sistema devuelva la cantidad de veces que se repite la letra en la frase.

**Entrada**

- Frase (f)
- Letra (l)

**Salida**

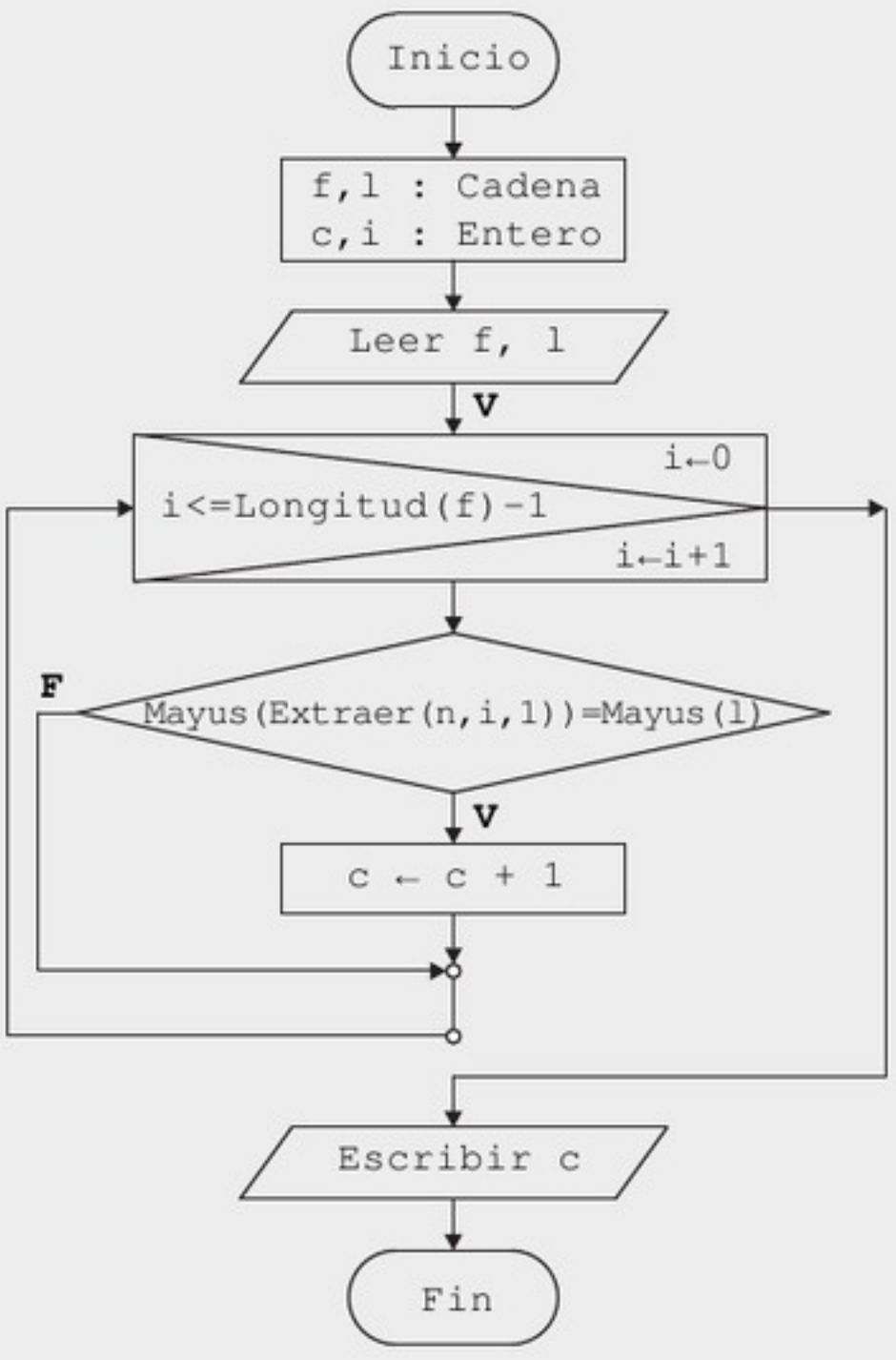
- Cantidad (c)

**Diseño:**

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   f,l : Cadena
   c,i : Entero

   //Entrada
   Leer f,l

   //Proceso
   Para i←0 Hasta Longitud(f)-1 Inc 1

   Si Mayus(Extraer(n,i,1))=Mayus(l)
       Entonces
       c ← c + 1
     Fin Si

   Fin Para

   //Salida
   Escribir c

Fin
```

**Codificación:**

```
'Variables
Dim f As String
Dim l As String
Dim c As Integer
Dim i As Integer

'Entrada
f = Me.txtf.Text
l = Me.txtl.Text

'Proceso
For i = 1 To Len(f)
    If UCase(Mid(f, i, 1)) = UCase(l) Then
        c = c + 1
    End If
Next

'Salida
Me.txtc.Text = Str(c)
```

## Problema n.° 94

**Enunciado:** Dado una frase, devolver la frase sin espacio en blancos.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese una frase; luego, que el sistema devuelva la frase sin espacios en blancos.

**Entrada**

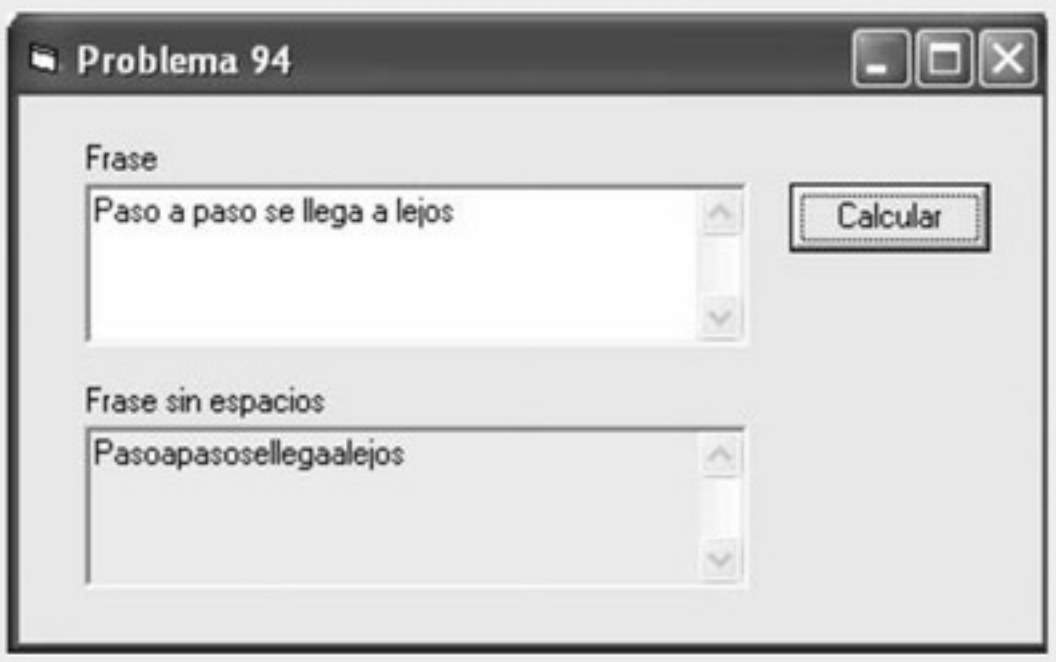
- Frase (f1)

**Salida**

- Frase sin espacios en blanco (f2)

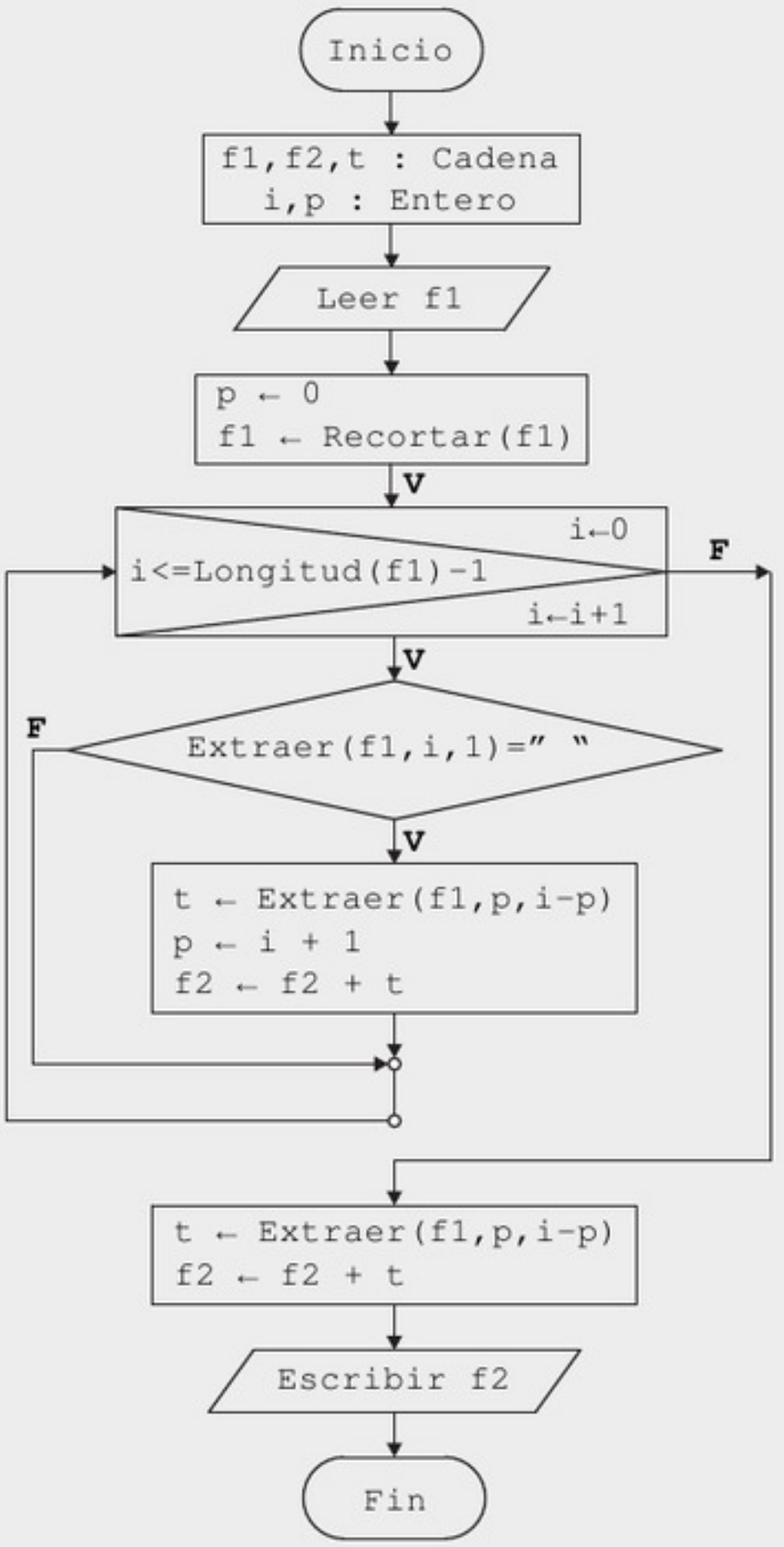
**Diseño:**

**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   f1,f2,t : Cadena
   i,p : Entero

   //Entrada
   Leer f1

   //Proceso
   p ← 0
   f1 ← Recortar(f1)

   Para i←0 Hasta Longitud(f)-1 Inc 1

   Si Extraer(f1,i,1)=" " Entonces

      t ← Extraer(f1,p,i-p)
      p ← i + 1
      f2 ← f2 + t

     Fin Si

   Fin Para

     t ← Extraer(f1,p,i-p)
     f2 ← f2 + t

   //Salida
   Escribir f2

Fin
```

**Codificación:**

```
'Variables
Dim f1 As String
Dim f2 As String
Dim t As String
Dim i As Integer
Dim p As Integer

'Entrada
f1 = Me.txtf1.Text

'Proceso
p = 1
f1 = Trim(f1)
For i = 1 To Len(f1)
     If Mid(f1, i, 1) = " " Then
        t = Mid(f1, p, i - p)
            p = i + 1
            f2 = f2 & t
     End If
Next

t = Mid(f1, p, i - p)
f2 = f2 & t

'Salida
Me.txtf2.Text = f2
```

## Problema n.° 95

**Enunciado:** Dada una frase, devuelva la frase en forma encriptada usando el método de convertir al siguiente caracteres del ASCII; ejemplo si el carácter es A = 65, devolverá B=66.

**Análisis:** Para la solución de este problema, se requiere que el usuario ingrese una frase; luego, que el sistema devolverá la frase en formato encriptado.

**Entrada**

- Frase (f1)

**Salida**

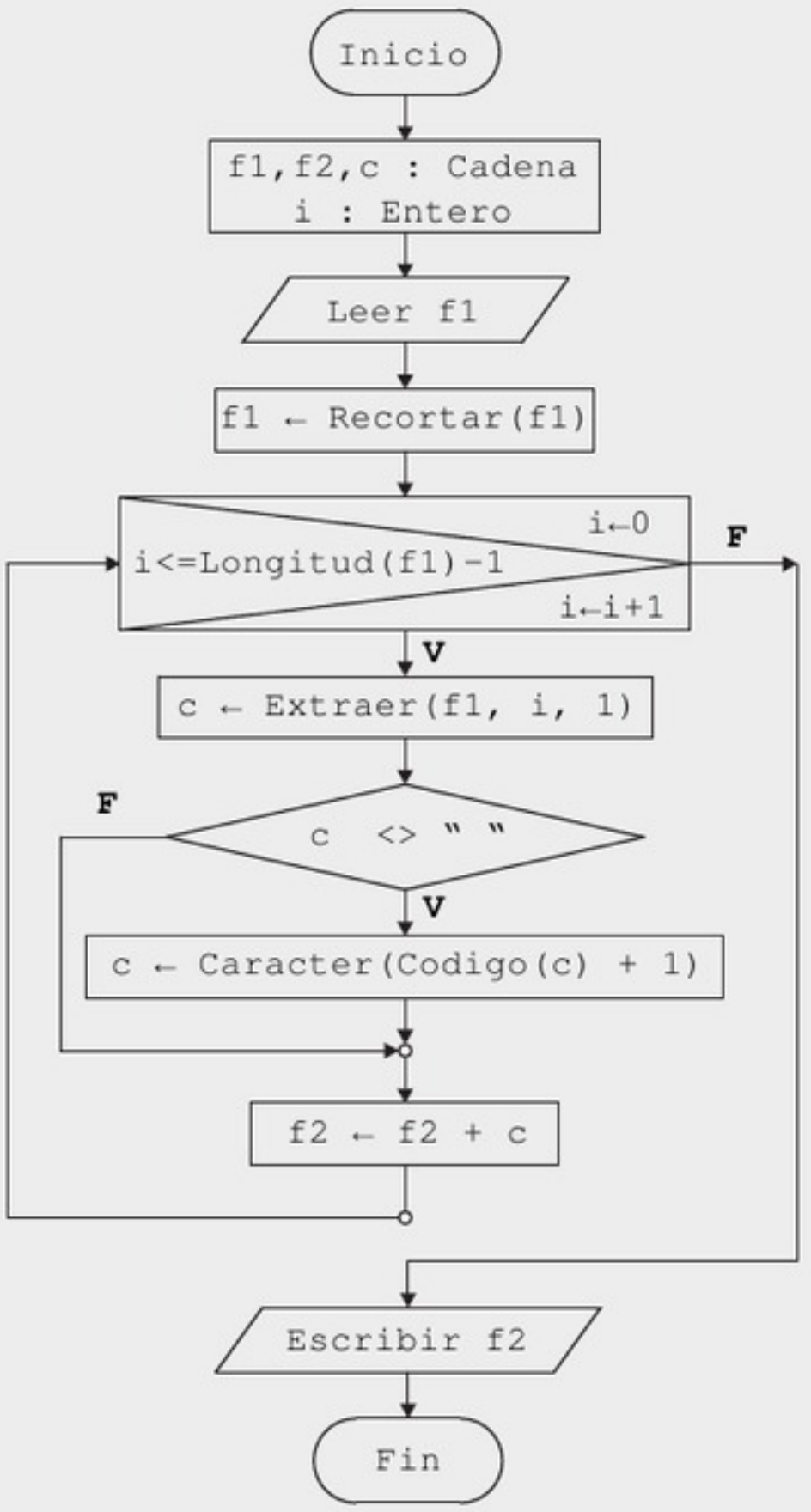
- Frase encriptada (f2)

**Diseño:**

**Interfaz de usuario**

**Algoritmo**

**Diagrama de flujo**




**Pseudocódigo**

```
Inicio

   //Variables
   f1,f2,c : Cadena
   i : Entero

   //Entrada
   Leer f1

   //Proceso
   f1 ← Recortar(f1)
   Para i←0 Hasta Longitud(f1)-1 Inc 1

       c ← Extraer(f1,i,1)
       Si c  <> " " Entonces
       c ← Caracter(Codigo(c) + 1)

       Fin Si
       f2 ← f2 + c
   Fin Para

   //Salida
   Escribir f2

Fin
```

**Codificación:**

```
'Variables
Dim f1 As String
Dim f2 As String
Dim c As String
Dim i As Integer

'Entrada
f1 = Me.txtf1.Text

'Proceso
f1 = Trim(f1)
For i = 1 To Len(f1)
     c = Mid(f1, i, 1)
     If c <> " " Then
          c = Chr(Asc(c) + 1)
     End If
     f2 = f2 & c
Next

'Salida
Me.txtf2.Text = f2
```

## 8.12 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 61

**Enunciado:** Dado el nombre de una persona, obtenga el mensaje: «Bienvenido, Sr(a) Gustavo, a su tienda de preferencia».

### Propuesto n.° 62

**Enunciado:** Dado un nombre, obtenga el nombre en forma invertida; por ejemplo, «Julio» invertido es «oiluJ».

### Propuesto n.° 63

**Enunciado:** Dada una frase, devuelva la frase con asteriscos en lugar de espacios en blanco.

### Propuesto n.° 64

**Enunciado:** Dada una letra, determine si está en minúscula o mayúscula.

### Propuesto n.° 65

**Enunciado:** Lea una frase y una palabra, y determine si existe o no la palabra en la frase.

### Propuesto n.° 66

**Enunciado:** Dada una palabra, determinar si es palíndromo (una palabra es palíndromo si se lee igual de izquierda a derecha o de derecha a izquierda), por ejemplo ANA.

### Propuesto n.° 67

**Enunciado:** Dada una frase, determine cuántas palabras palíndromos ha ingresado.

### Propuesto n.° 68

**Enunciado:** Dada una frase, determine cuántas palabras se repiten.

### Propuesto n.º 69

**Enunciado:** Cree el algoritmo para encriptar una frase con el valor del carácter ASCII, sumando 2 posiciones.

### Propuesto n.° 70

**Enunciado:** Cree el algoritmo para desencriptar la frase generada por el algoritmo anterior.

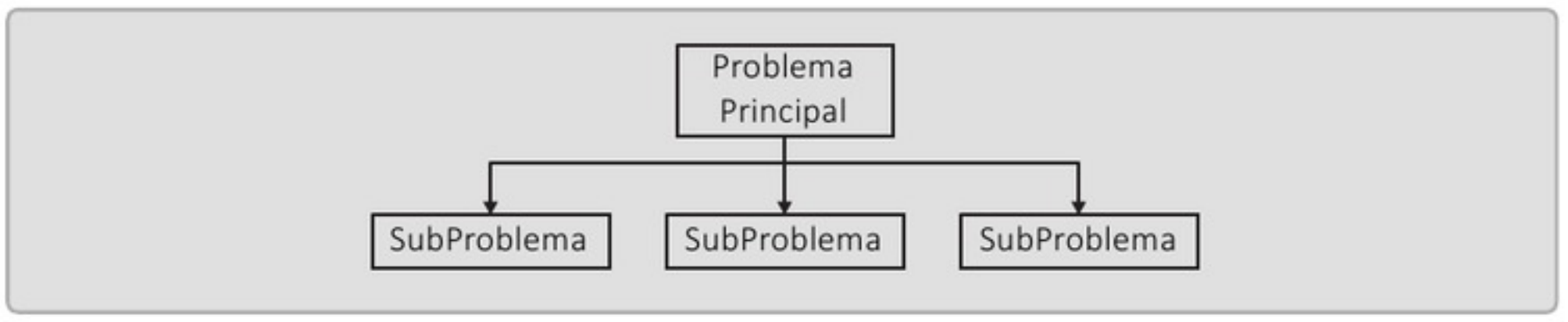
# Capítulo 9

# SubAlgoritmos (procedimientos y funciones)

## 9.1 Introducción

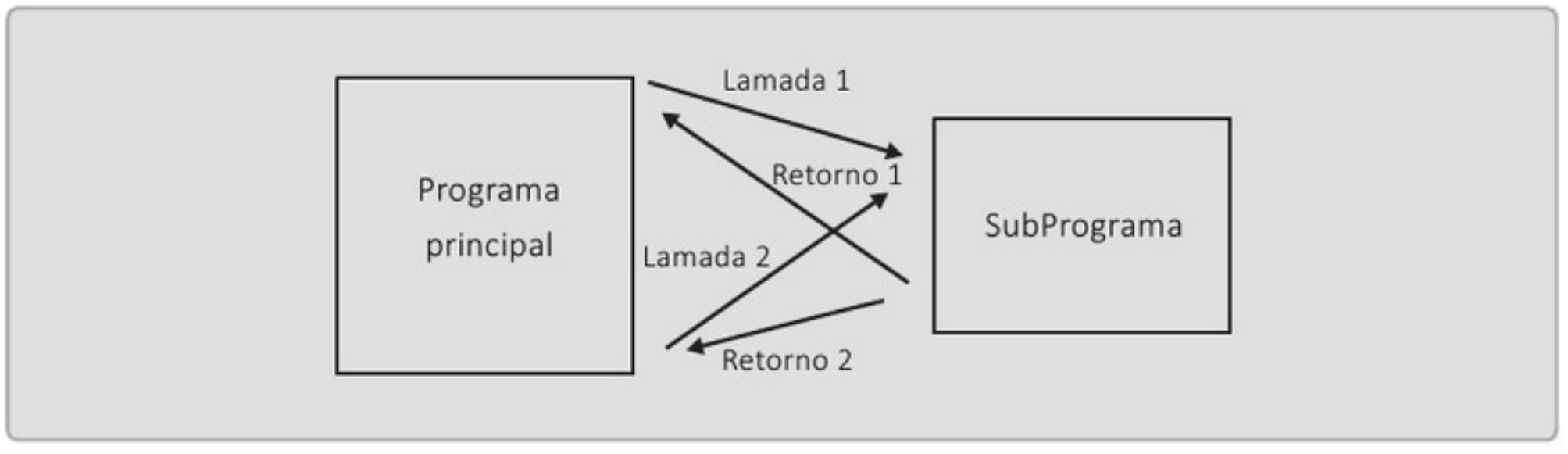
Una frase bastante usada en el mundo de la informática, para resolver problemas complejos, es: «Divide y vencerás». Esta es aplicada al tema de subalgoritmos (subprogramas), y consiste en dividir un problema grande en problemas más pequeños que se encargarán de resolver temas específicos. Los subalgoritmos (subprogramas) se dividen en dos de tipos procedimientos (subrutinas) y funciones, que evitarán la duplicidad de código y ayudarán a crear módulos más pequeños para un mejor mantenimiento, y permite reutilizarlo muchas veces.

El método para diseñar la solución de un problema principal (***main***) en subproblemas se conoce como diseño descendente (***top-dow design***), difundida por la programación modular.




El problema principal corresponde al programa o algoritmo principal, y la solución de los subproblemas mediantes subprogramas (procedimientos y funciones), en el lenguaje algorítmico se conoce como subalgoritmos.

El subprograma recibe datos y es invocado desde el programa principal, después de terminar el proceso que tuvo que realizar el subprograma devuelve el resultado correspondiente al programa principal.



## 9.2 Procedimientos

Los procedimientos se caracterizan por realizar una tarea específica y no retornar un resultado; sin embargo, sí es posible implementar que devuelva resultados por intermedio de parámetros llamados de salida o por referencia.

**Pseudocódigo**

```
//Crear un procedimiento
Procedimiento Proc1(E:Param1:Entero)

      <Instrucciones>

Fin Procedimiento

//Invocar el procedimiento
Llamar Proc1(10)
```

**Visual Basic**

```
'Crear un procedimiento
Private Sub Proc1(ByVal Param1 As Integer)

      <Instrucciones>

End Sub

'Invocar el procedimiento
Call Proc1(10)
```

## 9.3 Funciones

Son más conocidos por devolver un valor como resultado de la tarea realizada; los lenguajes de programación incorporan funciones que realizan algunas tareas ya programadas, conocidas como funciones internas, pero las funciones programadas por el usuario (programador) se conocen como externas o funciones definidas por el usuario (FDU).

**Pseudocódigo**

```
//Crear una función
Funcion Func1(E:Param1:Entero):Cadena

      <Instrucciones>

      Retorna <Valor>

Fin Funcion

//Invocar la función
c ← Func1(10)
```

**Visual Basic**

```
'Crear una función
Private Function Func1(ByVal Param1 As Integer) As String

      <Instrucciones>

      Func1 = <Valor>

End Function

'Invocar la función
c = Func1(10)
```

## 9.4 Paso de parámetros

Muchas veces los procedimientos y funciones requieren que le envíen una lista de valores llamados parámetros (argumentos), para usarlos en la solución de la tarea encomendada. Los parámetros son variables, muchas veces de entrada (reciben valores) y de salida (devuelven resultados) o ambos de entrada/salida.

Estos parámetros también toman el nombre de **parámetros por valor** (entrada) y **parámetros por referencias** (salida).

## 9.5 Parámetros por valor (entrada)

Los valores que se envían a los parámetros son asignados como una copia de los valores originales, desconectando el programa principal con el subprograma; es decir, si los valores de los parámetros cambian dentro del subprograma no afecta al programa principal.

**Pseudocódigo**

```
//Crear una función
Funcion Incrementar(E:N:Entero):Entero

      N ← N + 1 //Modifica el valor de N

      Retorna N

Fin Funcion

//Invocar la función
Num ← 5
Res ← Incrementar(Num) //El valor de Num se copia en N
Imprimir Num  //su valor sigue siendo 5
Imprimir Res  //su valor es 6
```

**Visual Basic**

```
'Crear una función
Private Function Incrementar(ByVal N As Integer) As Integer

      N = N + 1 'Modifica el valor de N

      Incrementar = N

End Function

'Invocar la función
Num = 5
Res = Incrementar(Num) 'El valor de Num se copia en N
Print Num  'su valor sigue siendo 5
Print Res  'su valor es 6
```

## 9.6 Parámetros por referencia (salida)

Se asignan las referencias de las variables (dirección de memoria de la variable) a los parámetros, conectando el programa principal con el subprograma; es decir, si los valores de los parámetros cambian dentro del subprograma, afecta a las variables del programa principal.

**Pseudocódigo**

```
//Crear una función
Funcion Incrementar(S:N:Entero):Entero

      N ← N + 1 //Modifica el valor de N

      Retorna N

Fin Funcion

//Invocar la función
Num ← 5
Res ← Incrementar(Num) //El parámetro N hace referencia a Num
Imprimir Num  //su valor ahora es 6
Imprimir Res  //su valor es 6
```

**Visual Basic**

```
'Crear una función
Private Function Incrementar(ByRef N As Integer) As Integer

      N = N + 1 'Modifica el valor de N
```

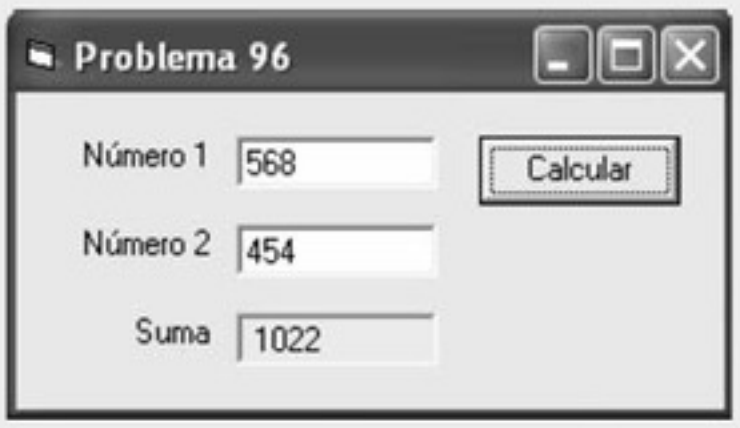
```
        Incrementar = N
    End Function

    'Invocar la función
    Num = 5
    Res = Incrementar(Num) 'El parámetro N hace referencia a Num
    Print Num  'su valor ahora es 6
    Print Res  'su valor es 6
```

En Visual Basic los arreglos (*array*) y los objetos son enviados siempre como parámetros por referencia.

## Problema n.° 96

**Enunciado:** Dados dos números enteros, hallar la suma. Cree una función para resolver el problema.

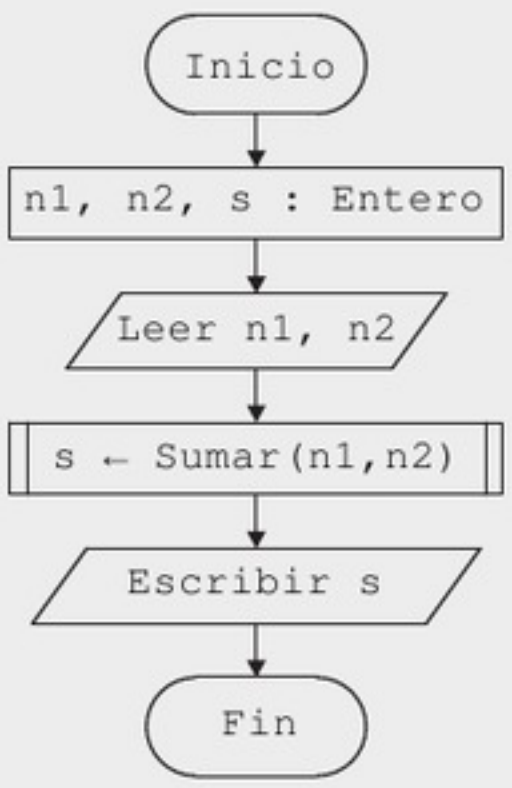
**Sumar**(E:Num1:Entero, E:Num2:Entero):Entero




### Algoritmo

**Diagrama de flujo**

**Principal**



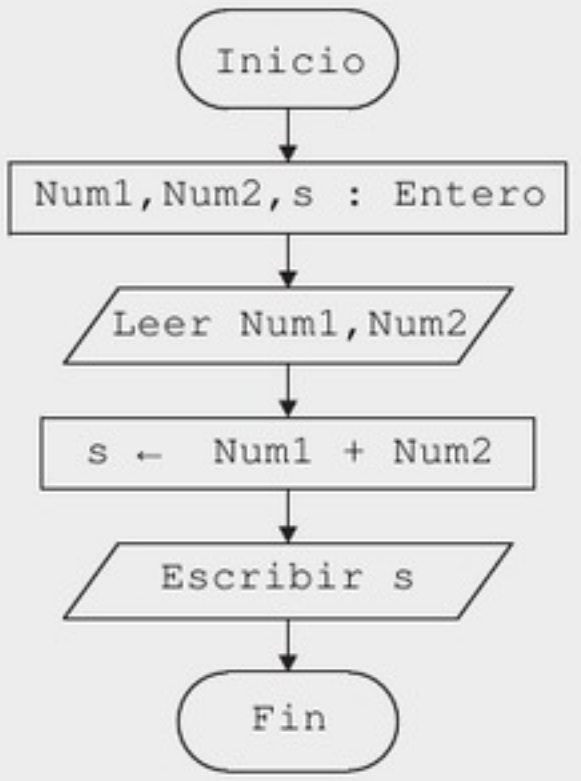

**Pseudocódigo**

```
Principal
Inicio

   //Variables
   n1, n2, s : Entero

   //Entrada
   Leer n1, n2

   //Proceso
   s ← Sumar(n1 + n2)

   //Salida
   Escribir s

Fin
```

**SubAlgoritmo**

**Diagrama de flujo**

**Sumar**




**Pseudocódigo**

```
Funcion Sumar(E:Num1:Entero,
              E:Num2:Entero):Entero

        //Variables locales
        s : Entero

        //Proceso
        s ← Num1 + Num2

        //Salida
        Retornar s

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim n1 As Integer
Dim n2 As Integer
Dim s As Integer

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
s = Sumar(n1, n2)

'Salida
Me.txts.Text = Str(s)
```

**Codificación Sumar:**

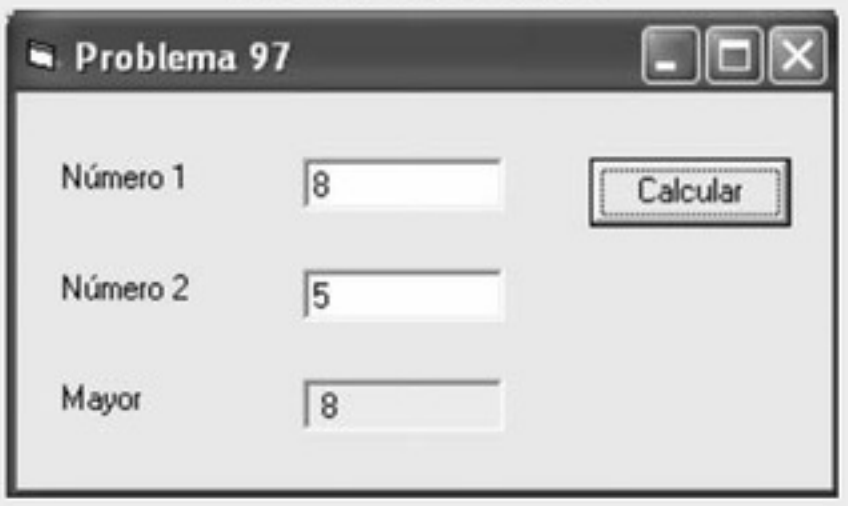
```
Private Function Sumar(ByVal Num1 As Integer, _
                       ByVal Num2 As Integer) As Integer

    'Variables locales
    Dim s As Integer

    'Proceso
    s = Num1 + Num2

    'Salida
    Sumar = s

End Function
```

## Problema n.° 97

**Enunciado:** Dado dos números enteros diferentes, devolver el número mayor. Cree una función para resolver el problema.

**Mayor**(E:n1:Entero, E:n2:Entero):Entero

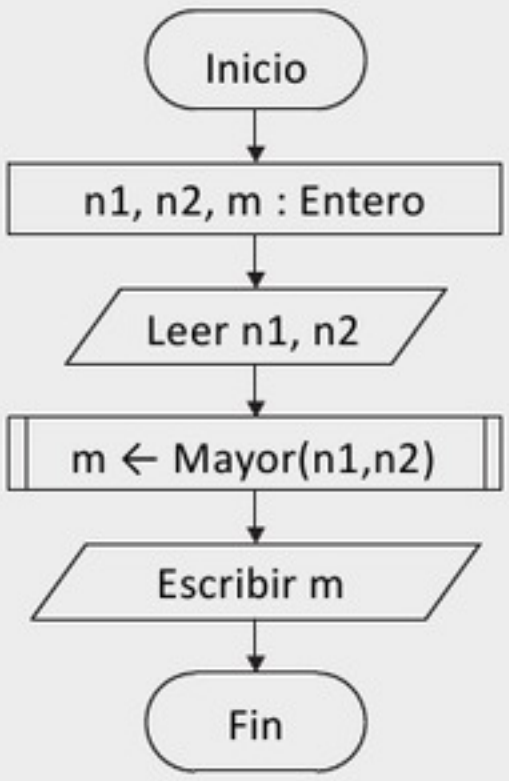
**Interfaz de usuario**




### Algoritmo

**Diagrama de flujo**

**Principal**




**Pseudocódigo**

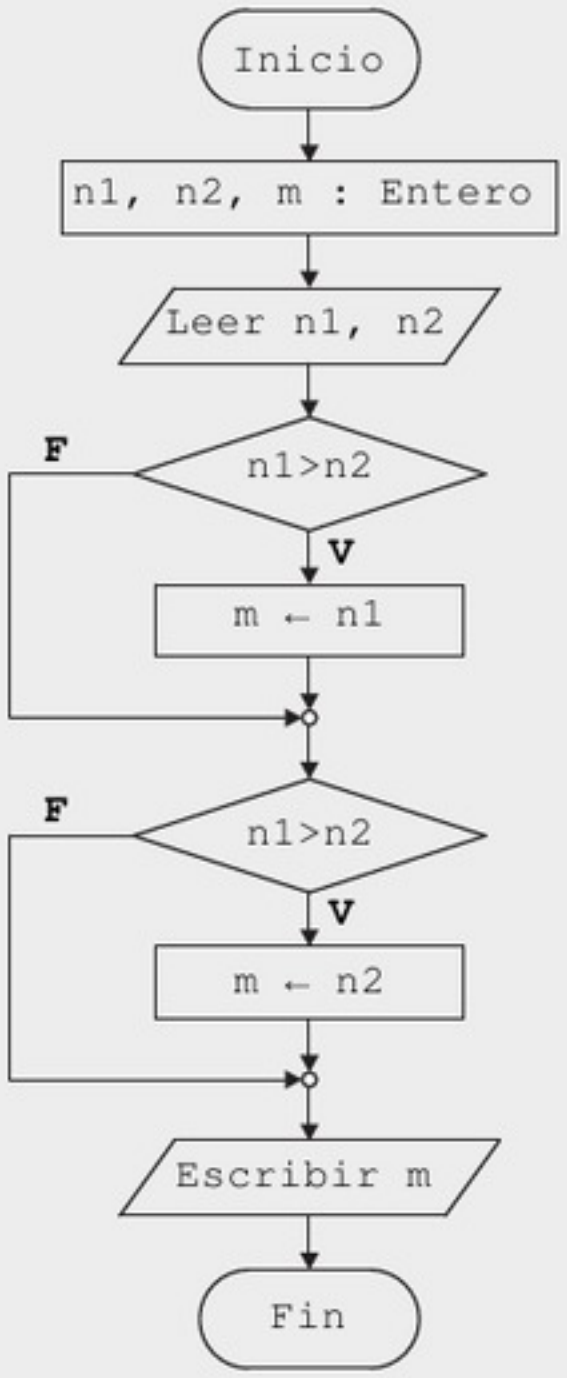
**Principal**

```
Inicio

//Variables
n1, n2, m : Entero

//Entrada
Leer n1, n2

//Proceso
m ← Mayor(n1,n2)

//Salida
Escribir m

Fin
```

**SubAlgoritmo**

**Diagrama de flujo**

**Mayor**




**Pseudocódigo**

```
Funcion Mayor(E:n1:Entero,
      E:n2:Entero):Entero

   //Variables locales
   m : Entero

   //Proceso
   Si n1 > n2 Entonces
       m ← n1
   Fin Si

   Si n2 > n1 Entonces
       m ← n2
   Fin Si

   //Salida
   Retorna m

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim n1 As Integer
Dim n2 As Integer
Dim m As Integer

'Entrada
n1 = Val(Me.txtn1.Text)
n2 = Val(Me.txtn2.Text)

'Proceso
m = Mayor(n1,n2)

'Salida
Me.txtm.Text = Str(m)
```

**Codificación Mayor:**

```
Private Function Mayor(ByVal n1 As Integer, _
                       ByVal n2 As Integer) As Integer

    ' Variables locales
    Dim m As Integer

    ' Proceso
    If n1 > n2 Then
        m = n1
    End If
    If n2 > n1 Then
        m = n2
    End If

    ' Salida
    Mayor = m

End Function
```

## Problema n.° 98

**Enunciado:** Determinar si un número entero es par o impar. Cree un procedimiento para resolver el problema.

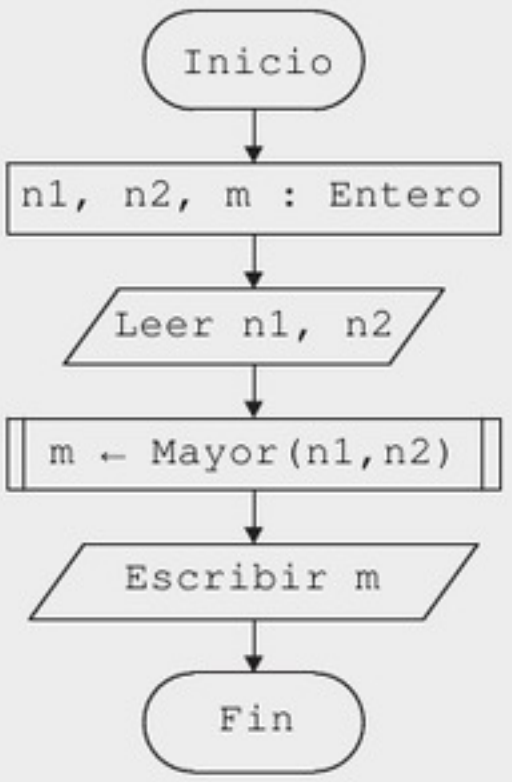
**ParImpar**(E:num:Entero, S:res:Entero)

**Interfaz de usuario**

## Algoritmo

**Diagrama de flujo**

**Principal**



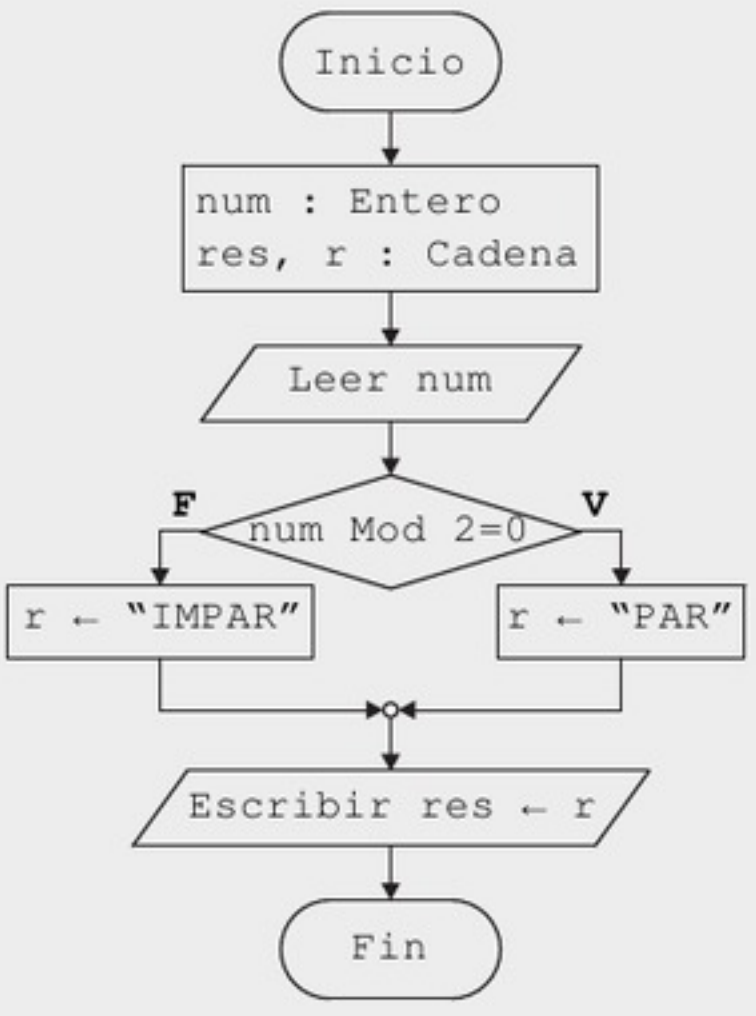

**Pseudocódigo**

```
Principal
Inicio
   //Variables
   n : Entero
   r : Cadena

   //Entrada
   Leer n

   //Proceso
   ParImpar(n, r)

   //Salida
   Escribir r
Fin
```

## SubAlgoritmo

**Diagrama de flujo**

**ParImpar**




**Pseudocódigo**

```
Procedimiento ParImpar(E:num:Entero,
                       S:res:Cadena)

   //Variables locales
   r : Cadena

   //Proceso
   Si num Mod 2 = 0 Entonces
       r ← "PAR"
   SiNo
       r ← "IMPAR"
   Fin Si

   //Salida
   res ← r

Fin Procedimiento
```

**Codificación Principal:**

```
'Variables
Dim n As Integer
Dim r As String

'Entrada
n = Val(Me.txtn.Text)

'Proceso
Call ParImpar(n, r)

'Salida
Me.txtr.Text = r
```

**Codificación ParImpar:**

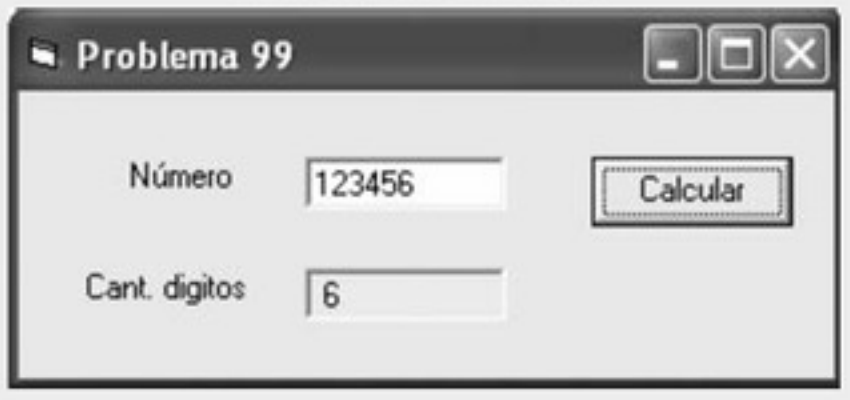
```
Private Sub ParImpar(ByVal num As Integer, ByRef res As String)
    'Variables
    Dim r As String

    'Procesar
    If num Mod 2 = 0 Then
        r = "PAR"
    Else
        r = "IMPAR"
    End If

    'Salida
    res = r
End Sub
```

## Problema n.° 99

**Enunciado:** Dado un número, determinar cuántos dígitos tiene. Cree una función para resolver el problema.

**CantidadDigitos**(E:num:Entero):Entero

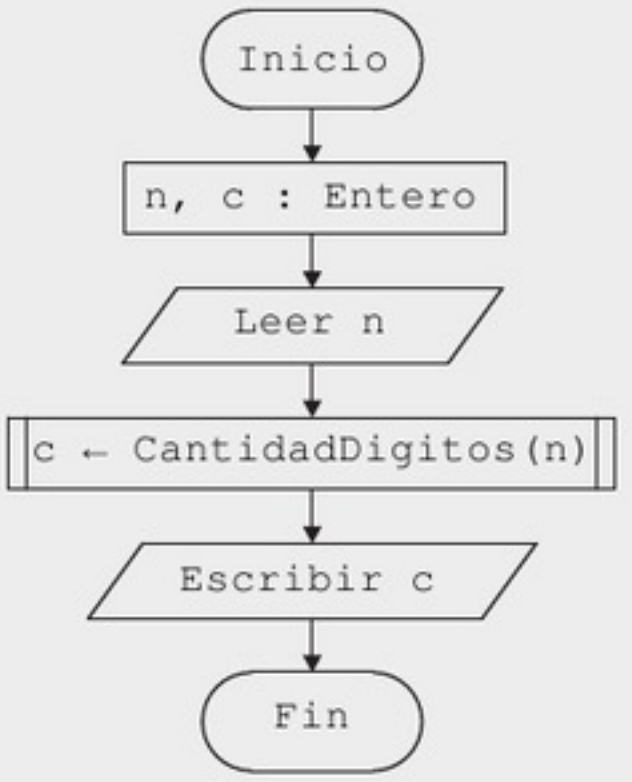
**Interfaz de usuario**



## Algoritmo

### Diagrama de flujo

**Principal**




### Pseudocódigo

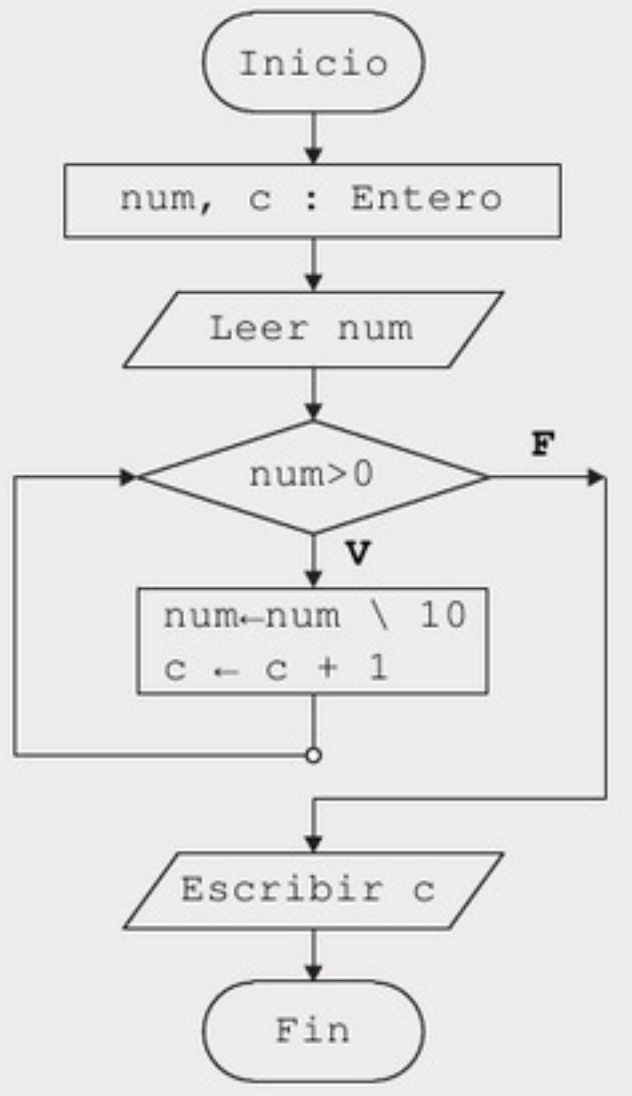
**Principal**

```
Inicio

   //Variables
   n, c : Entero

   //Entrada
   Leer n

   //Proceso
        c ← CantidadDigitos(n)

   //Salida
   Escribir c

Fin
```

## SubAlgoritmo

### Diagrama de flujo

**CantidadDigitos**




### Pseudocódigo

```
Funcion CantidadDigitos
(E:num:Entero):Entero

   //Variables locales
   c : Entero

   //Proceso
   Mientras num>0
        num ← num \ 10
        c ← c + 1
   Fin Mientras

   //Salida
   Retornar c

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim n As Long
Dim c As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
c = CantidadDigitos(n)

'Salida
Me.txtc.Text = Str(c)
```

**Codificación CantidadDigitos:**

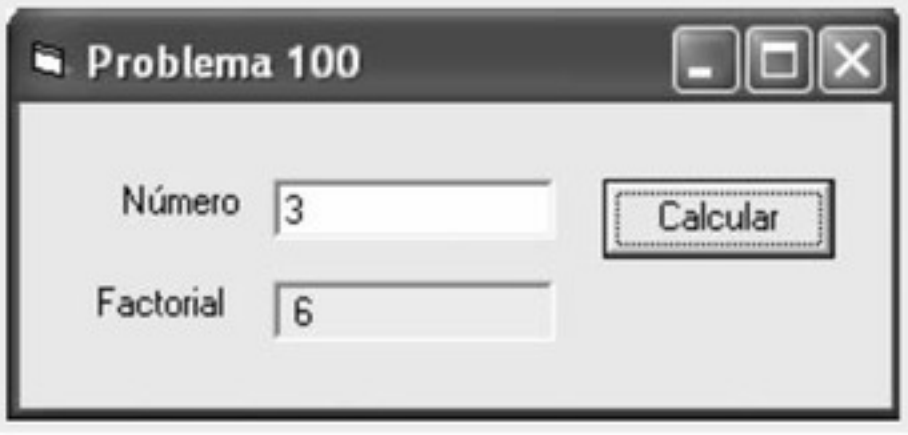
```
Private Function CantidadDigitos(ByVal num As Long) As Long
    'Variables locales
    Dim c As Long

    'Proceso
    Do While num > 0
        num = num \ 10
        c = c + 1
    Loop

    'Salida
    CantidadDigitos = c

End Function
```

## Problema n.° 100

**Enunciado:** Crear un algoritmo para hallar el factorial de un número, el factorial es el producto de todos los números consecutivos, desde la unidad hasta el número. Por ejemplo, factorial de 3! (se denota !) es 1 X 2 X 3 = 6. Cree una función para resolver el problema.

**Factorial**(E:num:Entero):Entero

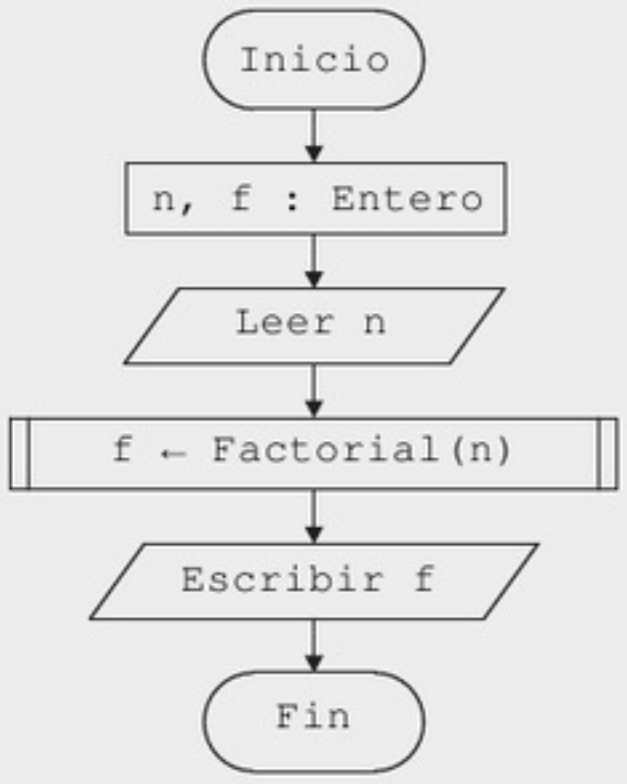
**Interfaz de usuario**



**Algoritmo**

**Diagrama de flujo**

**Principal**




**Pseudocódigo**

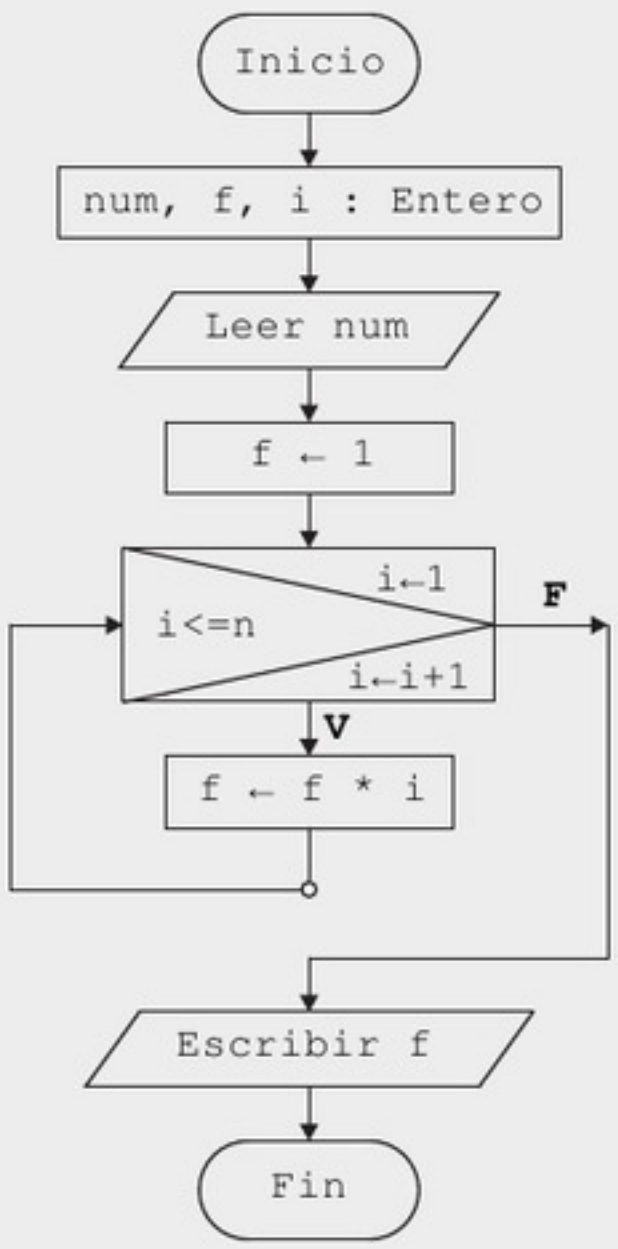
**Principal**

```
Inicio

   //Variables
   n, f : Entero

   //Entrada
   Leer n

   //Proceso
   f ← Factorial(n)

   //Salida
   Escribir f

Fin
```

**Algoritmo**

**Diagrama de flujo**

**Factorial**




**Pseudocódigo**

```
Funcion Factorial(E:num:Entero):Entero

   //Variables locales
   f, i : Entero

   //Proceso
   f ← 1
   Para i←1 Hasta num Inc 1
       f ← f * i
   Fin Para

   //Salida
   Retornar f

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim n As Long
Dim f As Long

'Entrada
n = Val(Me.txtn.Text)

'Proceso
    f = Factorial(n)

'Salida
Me.txtf.Text = Str(f)
```

**Codificación Factorial:**

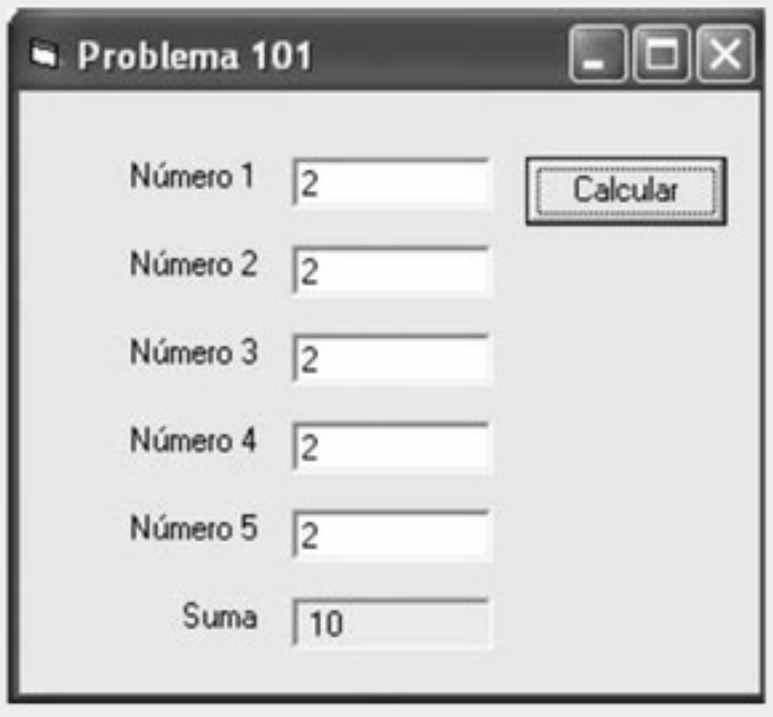
```
Private Function Factorial(ByVal num As Long) As Long
    'Variables locales
    Dim i As Long
    Dim f As Long

    'Proceso
    f = 1
    For i = 1 To num
        f = f * i
    Next

    'Salida
    Factorial = f

End Function
```

## Problema n.° 101

**Enunciado:** Dados 5 números, obtener la suma. Cree una función para resolver el problema.

**Sumar**(E:num[]:Entero):Entero

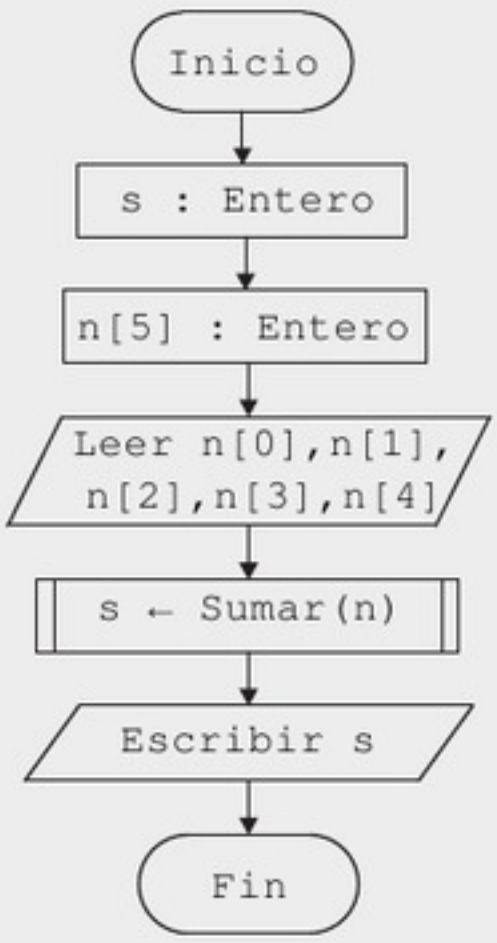
**Interfaz de usuario**



## Algoritmo

### Diagrama de flujo

**Principal**




### Pseudocódigo

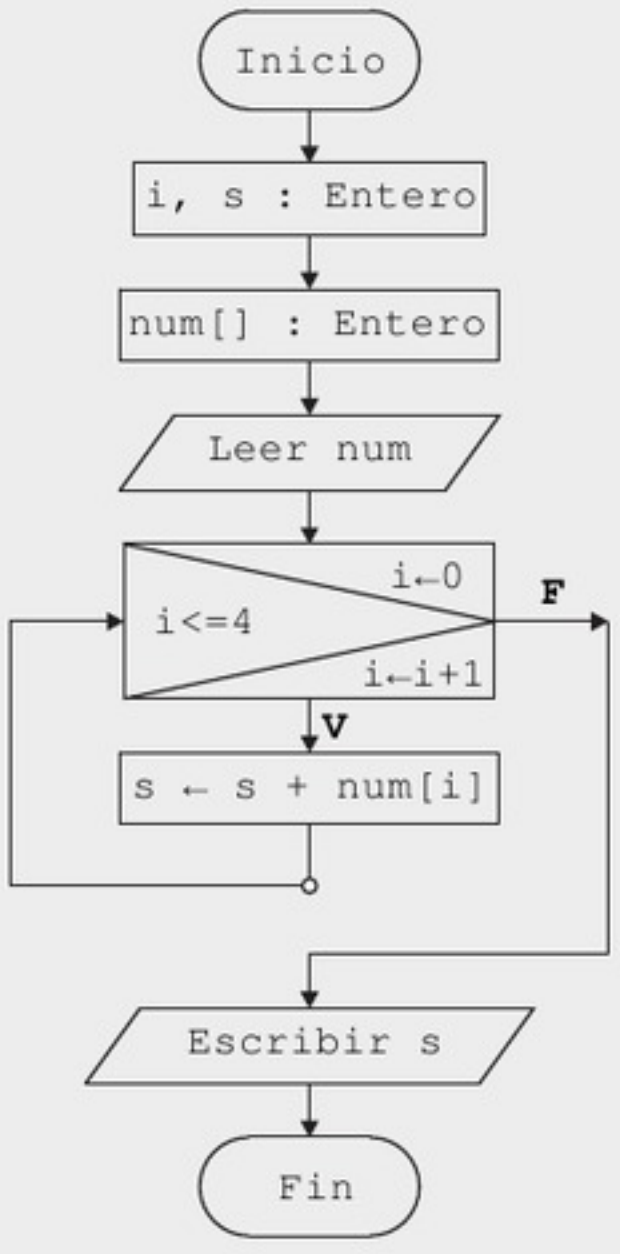
**Principal**

```
Inicio

   //Variables
   s : Entero

   //Arreglos (Vector)
   n[5] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3],n[4]

   //Proceso
   s ← Sumar(n)

   //Salida
   Escribir s

Fin
```

## SubAlgoritmo

### Diagrama de flujo

**Sumar**




### Pseudocódigo

```
Funcion Sumar(E:num[]:Entero):Entero

   //Variables locales
   i, s : Entero

   //Proceso
   Para i←0 Hasta 4 Inc 1
       s ← s + num[i]
   Fin Para

   //Salida
   Retornar s

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim s As Integer

'Arreglos (Vector)
Dim n(4) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)
n(4) = Val(Me.txtn5.Text)

'Proceso
s = Sumar(n)

'Salida
Me.txts.Text = Str(s)
```

**Codificación Sumar:**

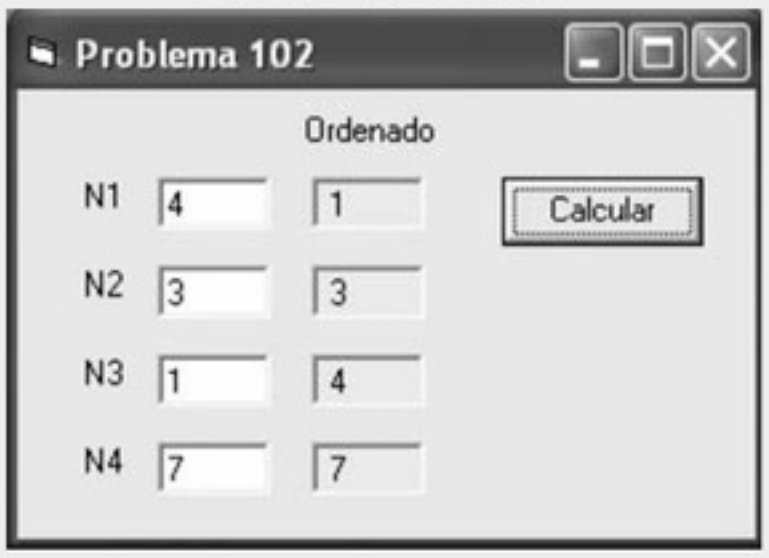
```
Private Function Sumar(ByRef num() As Integer) As Integer

    'Variables locales
    Dim s As Integer
    Dim i As Integer

    'Proceso
    For i = 0 To UBound(num, 1)
        s = s + num(i)
    Next

    'Salida
    Sumar = s

End Function
```

## Problema n.° 102

**Enunciado:** Ordene 4 números usando el método de ordenación por intercambio (burbuja). Cree un procedimiento para resolver el problema.

**Ordenar** (S:num[]:Entero)

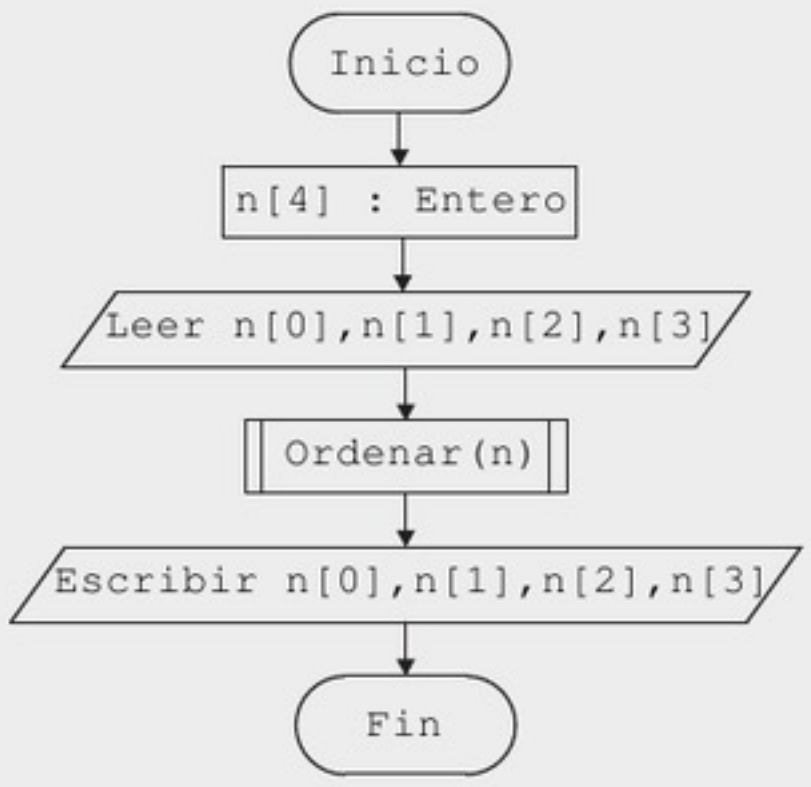
**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**

**Principal**



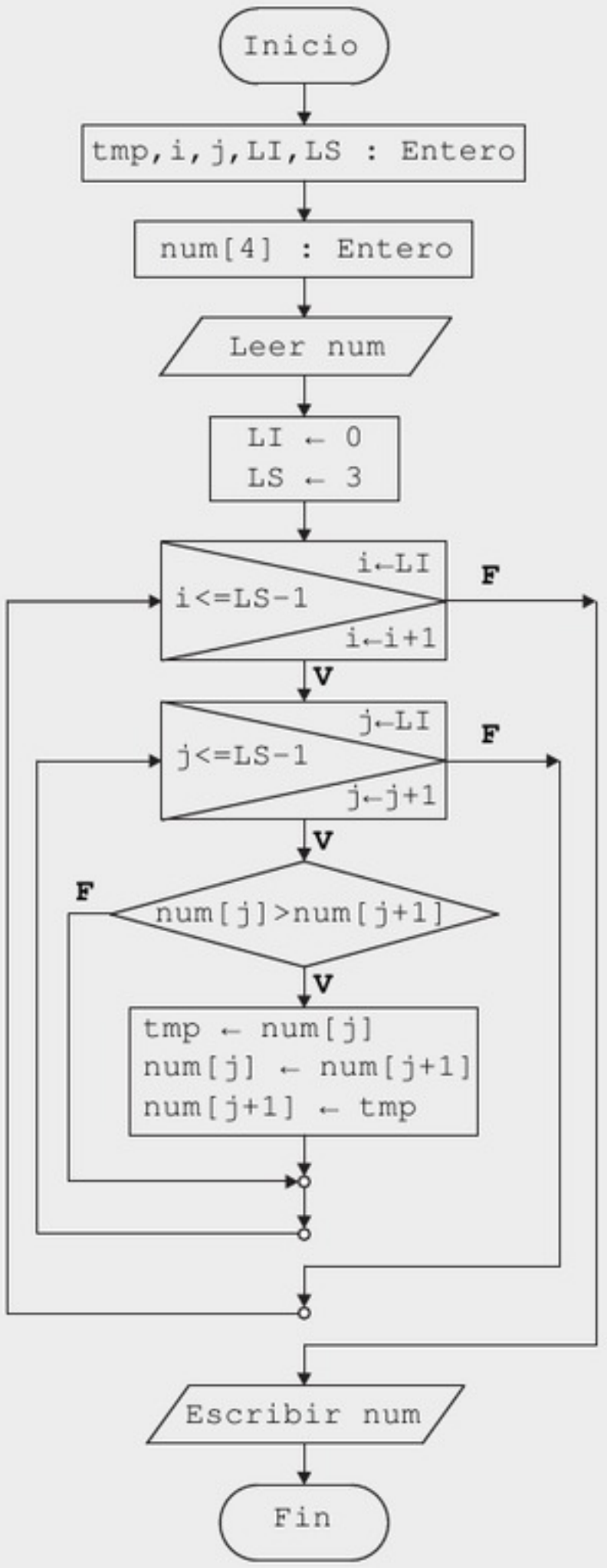

**Pseudocódigo**

```
Principal

Inicio

   //Arreglos (Vector)
   n[4] : Entero

   //Entrada
   Leer n[0],n[1],n[2],n[3]

   //Proceso
   Ordenar(n)

   //Salida
   Escribir n[0],n[1],n[2],n[3]

Fin
```

**SubAlgoritmo**

**Diagrama de flujo**

**Ordenar**




**Pseudocódigo**

```
Procedimiento Ordenar(S:num[]:Entero)

   //Variables locales
   tmp,i,j,LI,LS : Entero

   //Proceso
   LI ← 0
   LS ← 3
   Para i←LI Hasta LS-1 Inc 1
       Para j←LI Hasta LS-1 Inc 1
          Si num[j]>num[j+1] Entonces
             tmp ← num[j]
             num[j] ← num[j+1]
             num[j+1] ← tmp
          Fin Si
       Fin Para
   Fin Para

   //Salida
   Escribir num

Fin Procedimiento
```

**Codificación Principal:**

```
'Arreglos (Vector)
Dim n(3) As Integer

'Entrada
n(0) = Val(Me.txtn1.Text)
n(1) = Val(Me.txtn2.Text)
n(2) = Val(Me.txtn3.Text)
n(3) = Val(Me.txtn4.Text)

'Proceso
Ordenar(n)

'Salida
Me.txtno1.Text = Str(n(0))
Me.txtno2.Text = Str(n(1))
Me.txtno3.Text = Str(n(2))
Me.txtno4.Text = Str(n(3))
```

**Codificación Ordenar:**

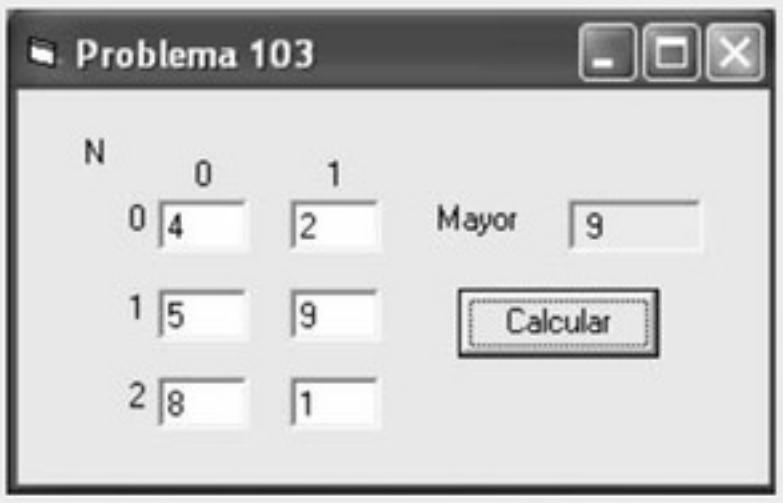
```
Private Sub Ordenar(ByRef num() As Integer)

    'Variables locales
    Dim tmp As Integer
    Dim i As Integer
    Dim j As Integer
    Dim LI As Integer
    Dim LS As Integer

    'Proceso
    LI = LBound(num, 1)
    LS = UBound(num, 1)

    For i = LI To LS - 1
        For j = LI To LS - 1
            If num(j) > num(j + 1) Then
                tmp = num(j)
                num(j) = num(j + 1)
                num(j + 1) = tmp
            End If
        Next
    Next

End Sub
```

## Problema n.° 103

**Enunciado:** Ingrese 6 números en una matriz de 3x2 y obtenga el número mayor ingresado. Cree una función para resolver el problema.

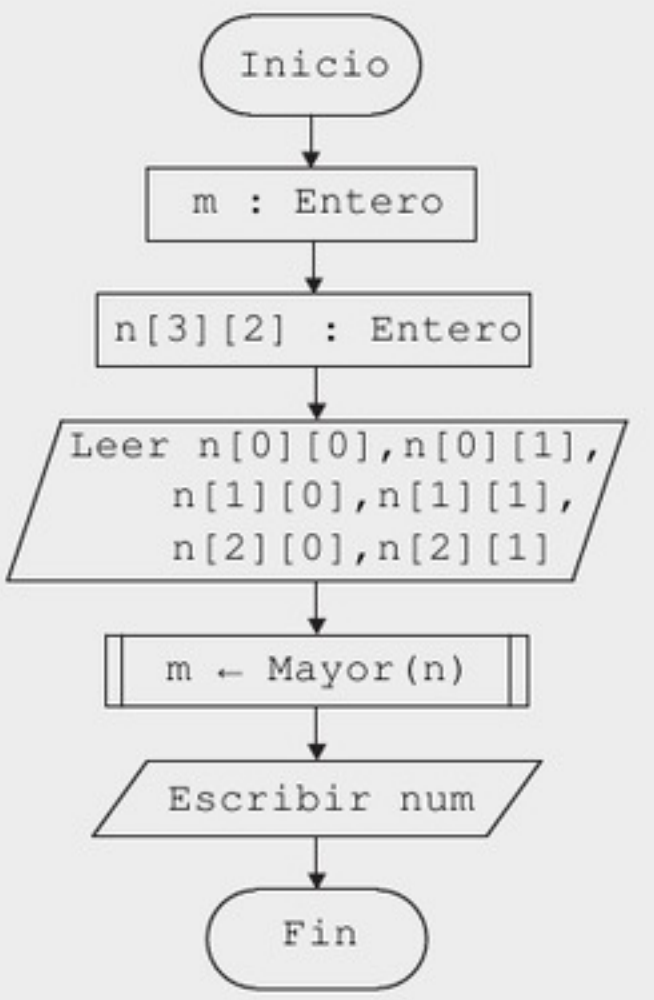
**Mayor**(E:num[]:Entero):Entero

**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**



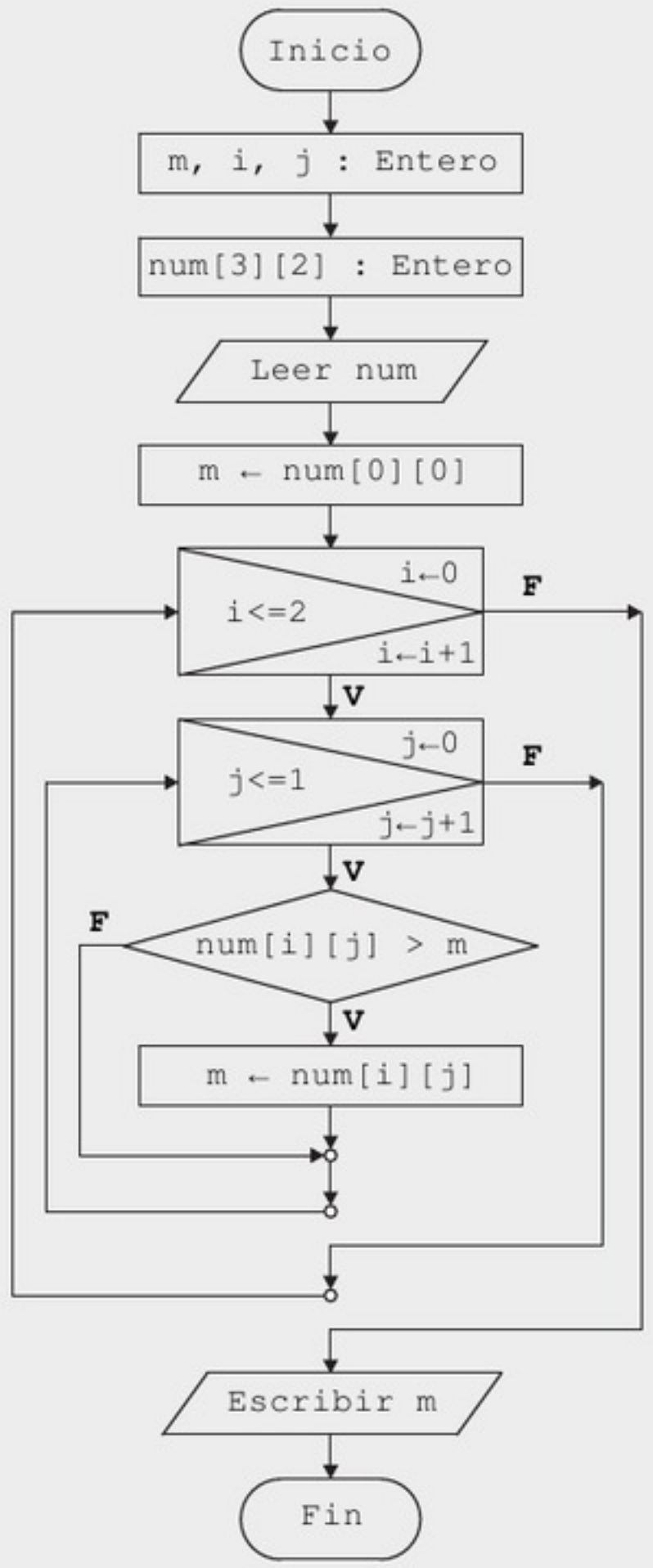

**Pseudocódigo**

```
Inicio

   //Variables
   m : Entero

   //Arreglos (Matriz)
   n[3][2] : Entero

   //Entrada
   Leer n[0][0],n[0][1],
        n[1][0],n[1][1],
        n[2][0],n[2][1]

   //Proceso
   m ← Mayor(n)

   //Salida
   Escribir m

Fin
```

**SubAlgoritmo**

**Mayor**

**Diagrama de flujo**




**Pseudocódigo**

```
Funcion Mayor(E:num[][]:Entero):Entero

   //Variables locales
   m, i, j : Entero

   //Arreglos (Matriz)
   num[3][2] : Entero

   //Proceso
   m ← num[0][0]
   Para i←0 Hasta 2 Inc 1
       Para j←0 Hasta 1 Inc 1
           Si num[i][j]>m Entonces
             m ← num[i][j]
           Fin Si
       Fin Para
   Fin Para

   //Salida
   Escribir m

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim m As Integer

'Arreglos
Dim n(2, 1) As Integer

'Entrada
n(0, 0) = Val(Me.txtn00.Text)
n(0, 1) = Val(Me.txtn01.Text)
n(1, 0) = Val(Me.txtn10.Text)
n(1, 1) = Val(Me.txtn11.Text)
n(2, 0) = Val(Me.txtn20.Text)
n(2, 1) = Val(Me.txtn21.Text)

'Proceso
m = Mayor(n)

'Salida
Me.txtm.Text = Str(m)
```

**Codificación Mayor:**

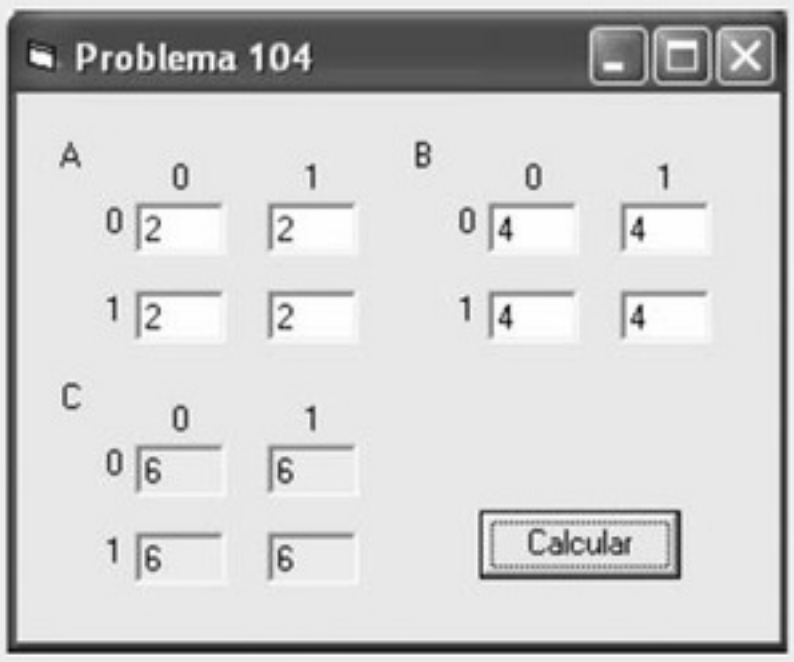
```
Private Function Mayor(ByRef num() As Integer) As Integer

    'Variables locales
    Dim m As Integer
    Dim i As Integer
    Dim j As Integer

    'Proceso
    m = num(0, 0)
    For i = 0 To 2
        For j = 0 To 1
            If num(i, j) > m Then
                m = num(i, j)
            End If
        Next
    Next

    'Salida
    Mayor = m

End Function
```

## Problema n.° 104

**Enunciado:** Dado la matriz A de 2X2, la matriz B de 2X2, obtenga la suma de dichas matriz. Cree una función para resolver el problema.

**Mayor**(E:num[]:Entero):Entero

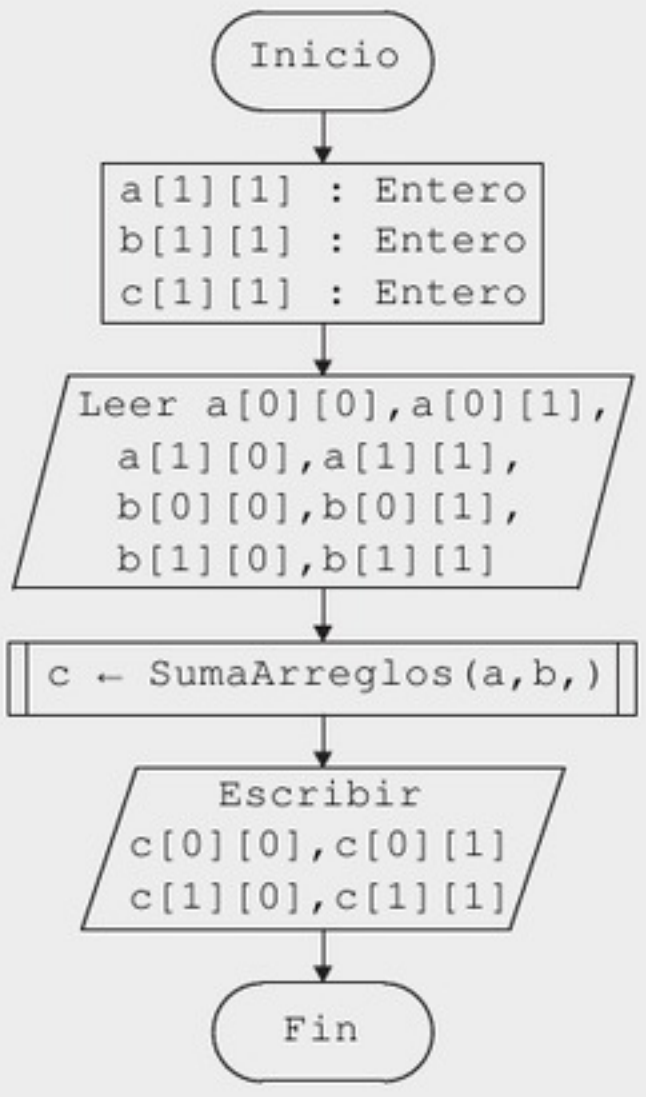
**Interfaz de usuario**




**Algoritmo**

**Diagrama de flujo**

**Principal**




**Pseudocódigo**

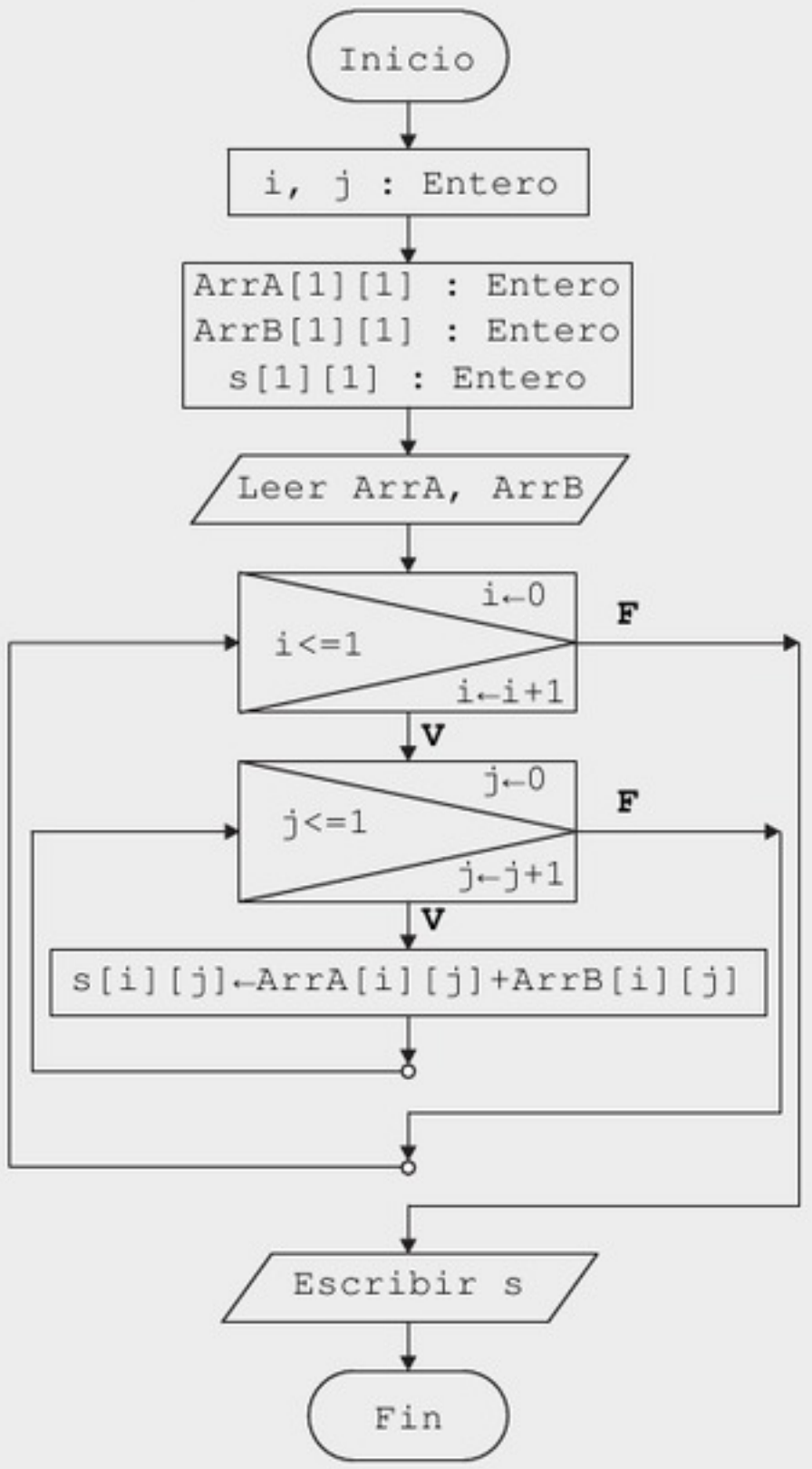
**Principal**

```
Inicio

   //Arreglos (Matriz)
   a[1][1] : Entero
   b[1][1] : Entero
   c[1][1] : Entero

   //Entrada
   Leer a[0][0],a[0][1],
        a[1][0],a[1][1],
        b[0][0],b[0][1],
        b[1][0],b[1][1]

   //Proceso
   c ← SumaArreglos(a,b)

   //Salida
   Escribir c[0][0],c[0][1],
            c[1][0],c[1][1]

Fin
```

**Algoritmo**

**Diagrama de flujo**

**SumaArreglos**




**Pseudocódigo**

```
Funcion SumaArreglos(E:ArrA[][]:Entero, E:ArrB[][]:Entero):Entero[][]

   //Variables locales
   i, j : Entero

   //Arreglos (Matriz)
   s[1][1] : Entero

   //Proceso
   Para i←0 Hasta 1 Inc 1
       Para j←0 Hasta 1 Inc 1
        s[i][j]←ArrA[i][j]+ArrB[i][j]
       Fin Para
   Fin Para

   //Salida
   Retornar s

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim i As Integer
Dim j As Integer

'Arreglos
Dim a(1, 1) As Integer
Dim b(1, 1) As Integer
Dim c(1, 1) As Integer

'Entrada (Matriz)
a(0, 0) = Val(Me.txta00.Text)
a(0, 1) = Val(Me.txta01.Text)
a(1, 0) = Val(Me.txta10.Text)
a(1, 1) = Val(Me.txta11.Text)
```

```
b(0, 0) = Val(Me.txtb00.Text)
b(0, 1) = Val(Me.txtb01.Text)
b(1, 0) = Val(Me.txtb10.Text)
b(1, 1) = Val(Me.txtb11.Text)

'Proceso
c = SumaArreglos(a,b)

'Salida
Me.txtc00.Text = c(0, 0)
Me.txtc01.Text = c(0, 1)
Me.txtc10.Text = c(1, 0)
Me.txtc11.Text = c(1, 1)
```

**Codificación SumaArreglos:**

```
Private Function SumaArreglos(ByRef ArrA() As Integer, ByRef ArrB()
As Integer) As Integer()
    'Variables
    Dim i As Integer
    Dim j As Integer

    'Arreglos
    Dim s(1, 1) As Integer

    'Proceso
    For i = 0 To 1
        For j = 0 To 1
            s(i, j) = ArrA(i, j) + ArrB(i, j)
        Next
    Next

    'Salida
    SumaArreglos = s

End Function
```

## Problema n.° 105

**Enunciado:** Dada una frase, devolver la frase sin espacio en blanco. Cree una función para resolver el problema.

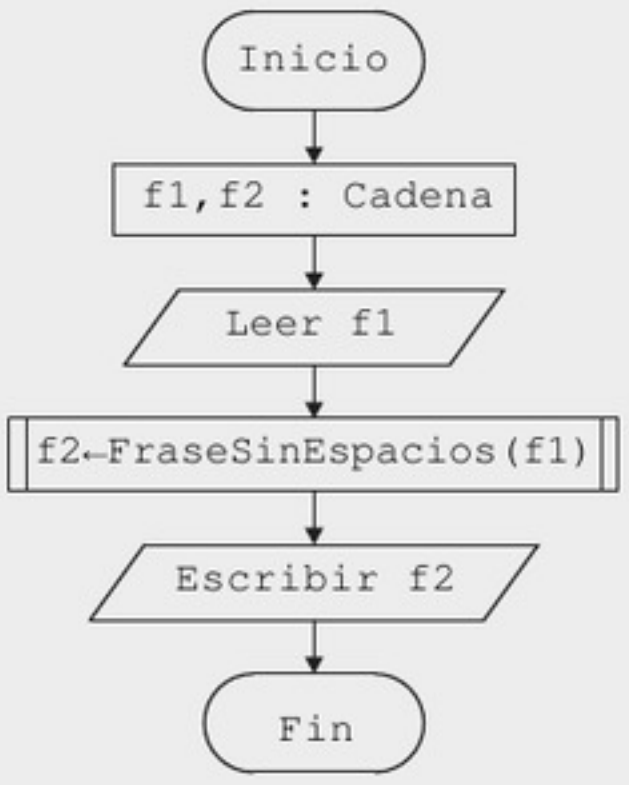
**FraseSinEspacios**(E:Frase:Cadena):Cadena

**Interfaz de usuario**


**Algoritmo**

**Diagrama de flujo**

**Principal**




**Pseudocódigo**

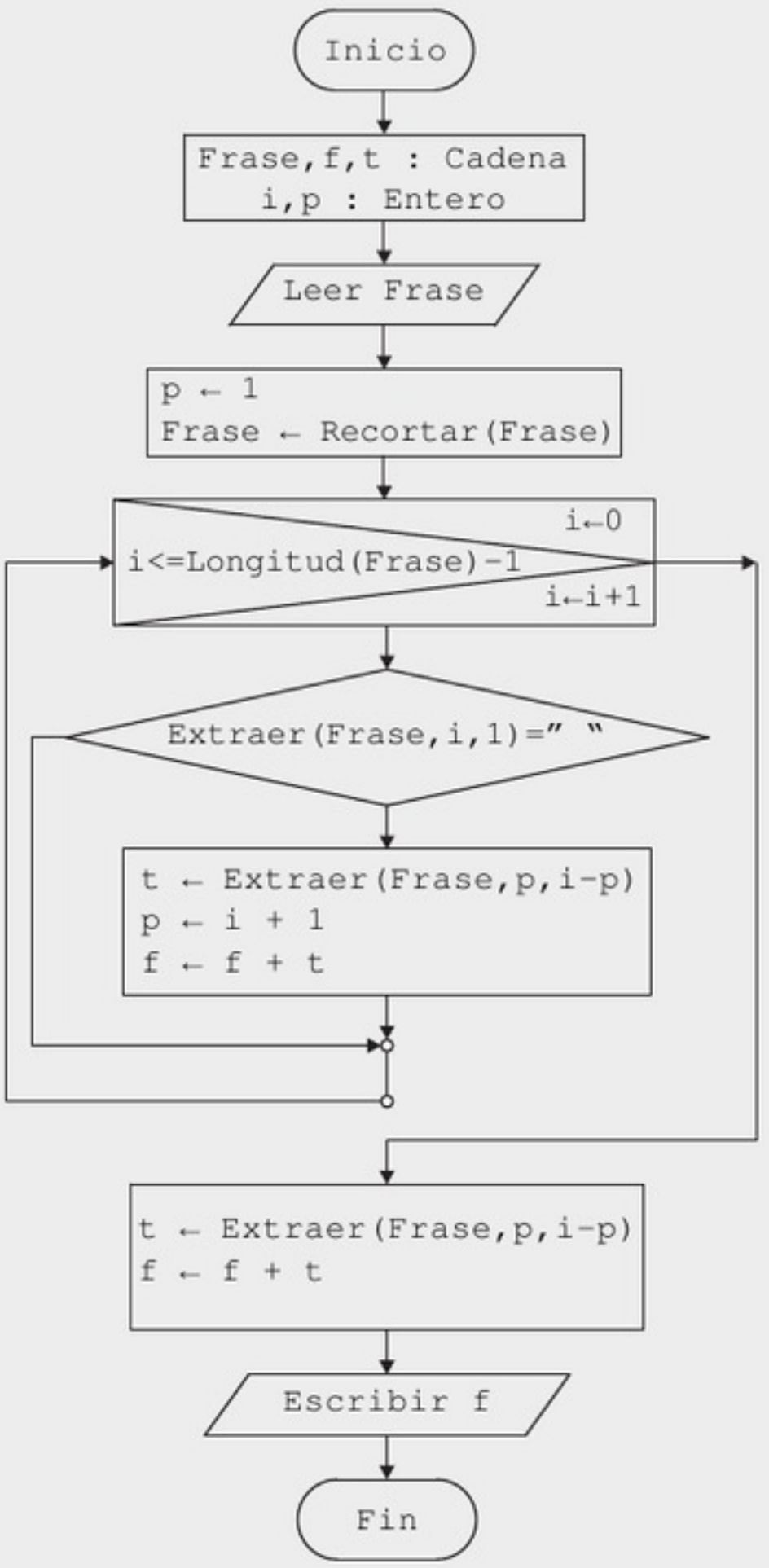
**Principal**

```
Inicio

   //Variables
   f1,f2 : Cadena

   //Entrada
   Leer f1

   //Proceso
   f2 ← FraseSinEspacios(f1)

   //Salida
   Escribir f2

Fin
```

## Algoritmo

### Diagrama de flujo

**FrasesSinEspacios**




### Pseudocódigo

```
Funcion FraseSinEspacios
        (E:Frase:Cadena):Cadena

   //Variables
   f2,t : Cadena
   i,p : Entero

   //Proceso
   p ← 1
   Frase ← Recortar(Frase)

   Para i←0 Hasta Longitud(Frase)-1
   Inc 1

   Si Extraer(Frase,i,1)=" " Entonces

        t ← Extraer(Frase,p,i-p)
      p ← i + 1
      f ← f + t

     Fin Si

   Fin Para

     t ← Extraer(Frase,p,i-p)
     f ← f + t

   //Salida
   Retornar f

Fin Funcion
```

**Codificación Principal:**

```
'Variables
Dim f1 As String
Dim f2 As String

'Entrada
f1 = Me.txtf1.Text

'Proceso
f2 = FraseSinEspacios(f1)

'Salida
Me.txtf2.Text = f2
```

**Codificación FraseSinEspacios:**

```
Private Function FraseSinEspacios(ByVal Frase As String) As String
    'Variables locales
    Dim f As String
    Dim t As String
    Dim i As Integer
    Dim p As Integer

    'Proceso
    p = 1
    Frase = Trim(Frase)
    For i = 1 To Len(Frase)
        If Mid(Frase, i, 1) = " " Then
            t = Mid(Frase, p, i - p)
            p = i + 1
            f = f & t
        End If
    Next

    t = Mid(Frase, p, i - p)
    f = f & t

    'Salida
    FraseSinEspacios = f

End Function
```

## 9.7 Problemas propuestos

Los siguientes problemas le servirán para medir su aprendizaje, es importante que los resuelva.

### Propuesto n.° 71

**Enunciado:** Hallar el área y el perímetro de un de un cuadrado, cree un procedimiento para realizar dicha tarea.

**Cuadrado** (E:Lado:Real, S:Area:Real, S:Perimetro:Real)

### Propuesto n.° 72

**Enunciado:** Dadas tres notas, obtenga el promedio de las dos notas mayores, cree un procedimiento para realizar dicha tarea.

**Promedio**(E:N1:Real,E:N2:Real,E:N3:Real,S:Promedio:Real)

### Propuesto n.° 73

**Enunciado:** Dada la edad de una persona, determine en qué etapa de su vida se encuentra. Cree un procedimiento para realizar dicha tarea.

**Etapa** (E:Edad:Entero, S:Etapa:Cadena)

| Edad | Etapa |
|---|---|
| Entre 0 y 2 | Bebé |
| Entre 3 y 5 | Niño |
| Entre 6 y 12 | Pubertad |
| Entre 13 y 18 | Adolescente |
| Entre 19 y 25 | Joven |
| Entre 26 y 60 | Adulto |
| Mayor a 60 | Anciano |

### Propuesto n.° 74

**Enunciado:** Dado un número, obtener la suma de sus dígitos pares e impares.

**Recuerde:** Crear un procedimiento que realice la tarea.

### Propuesto n.° 75

**Enunciado:** Dado un carácter, determinar si es vocal, letra mayúscula, letra minúscula, número o símbolo.

**Recuerde:** Crear un procedimiento que realice la tarea.

### Propuesto n.° 76

**Enunciado:** Hallar el área de un rectángulo, cree una función para realizar dicha tarea.

**AreaRectangulo**(E:Base:Real, E:Altura:Real):Real

## Propuesto n.° 77

**Enunciado:** Un negocio tiene dos tipos de cliente: publico en general (G) y cliente afiliado (A). Acepta dos formas de pago: al contado (C) o en plazos (P). Nos piden crear un programa que al ingresar el monto de la compra se obtenga el monto del descuento o el monto del recargo y el total a pagar, según la siguiente tabla:

| Tipo | Contado (C) descuento | Plazos (P) recargo |
|---|---|---|
| Público en general (G) | 15 % | 10 % |
| Cliente afiliado (A) | 20 % | 5 % |

- Cree una función para obtener el % de recargo

  **Recargo**(E:Tipo:Carácter):Real

- Cree una función para obtener el % del descuento

  **Descuento**(E:Tipo:Carácter):Real

## Propuesto n.° 78

**Enunciado:** Lea un número y devuelva el número en forma inversa. Por ejemplo, si ingresa 123, su número invertido es 321; si ingresa 12345, su número invertido 54321.

**Recuerde**: Crear una función que realice la tarea.

## Propuesto n.° 79

**Enunciado:** Dada una palabra, determinar si es palíndromo (una palabra es palíndromo si se lee igual de izquierda a derecha o de derecha a izquierda), por ejemplo ANA.

**Recuerde:** Crear una función que realice la tarea.

## Propuesto n.° 80

**Enunciado:** Cree una matriz de A de 2x2 y otra B de 2x2, y obtenga una matriz C =A * B.

**Recuerde**: Crear una función que realice la tarea.

Impreso en los talleres gráficos de

EDITORIAL MACRO®

Surquillo