Práctica 1 SSOO

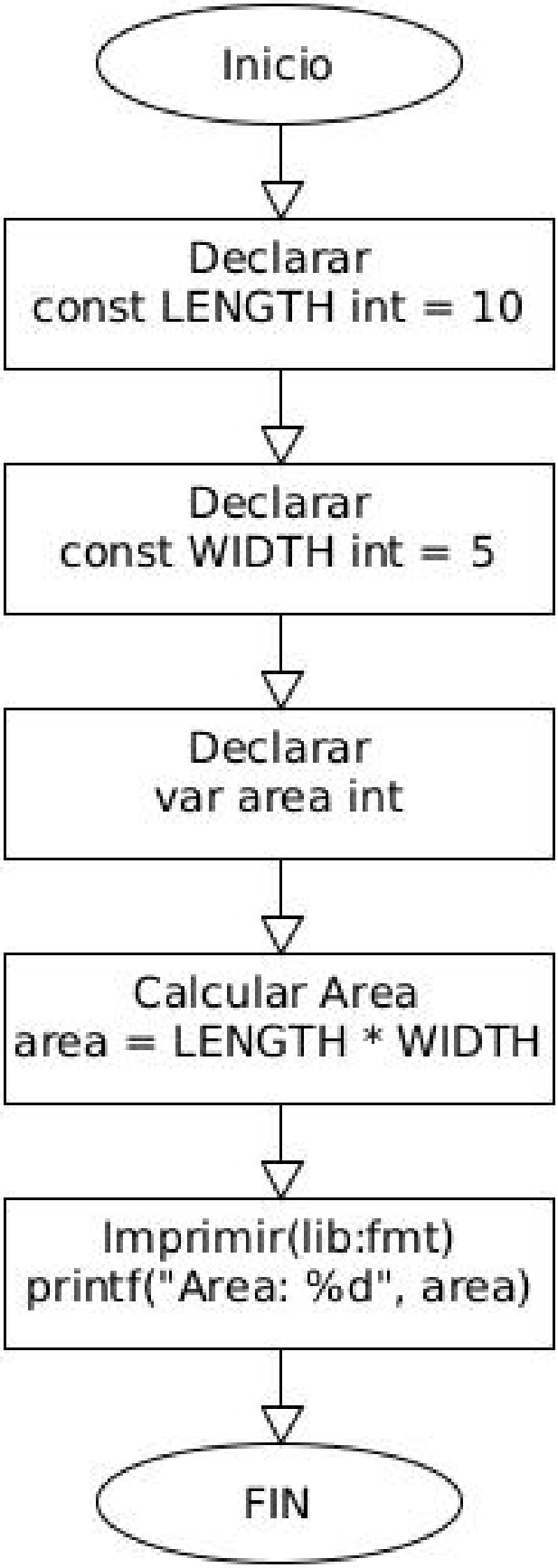
Adrián Montes Linares

1 Códigos

1.1 Ejercicio 1

Describir el siguiente código en un diagrama UML y un pseudocódigo.

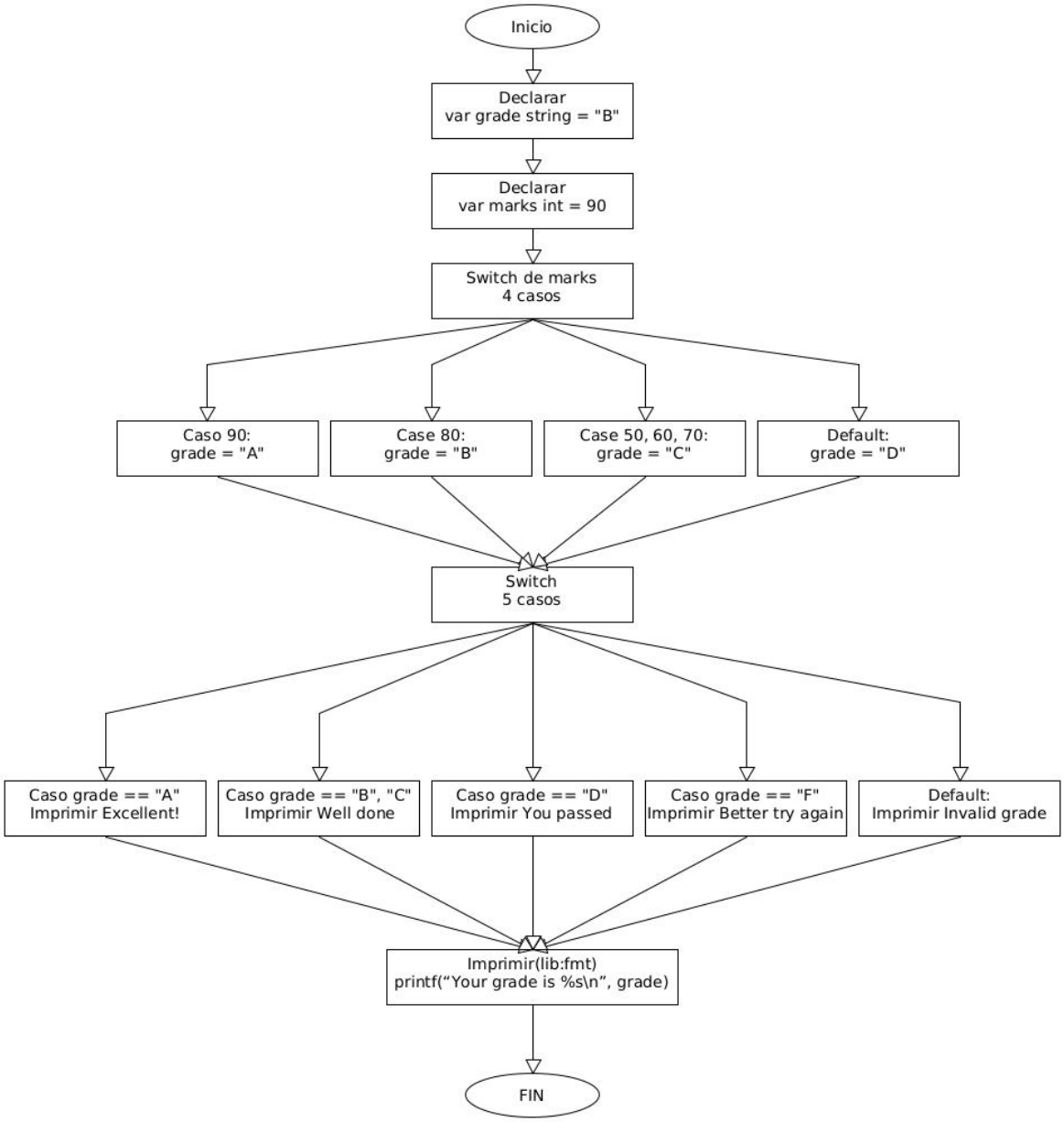
```
1. Función main:
    1.1 Definir constante LENGTH = 10
    1.2 Definir constante WIDTH = 5
    1.3 Declarar variable área de tipo entero
    1.4 Asignar a área el valor de LENGTH * WIDTH
    1.5 Mostrar mensaje "Valor del área: área"
```


Inicio
Declarar
const LENGTH int = 10
Declarar
const WIDTH int = 5
Declarar
var area int
Calcular Area
area = LENGTH * WIDTH
Imprimir(lib:fmt)
printf("Area: %d", area)
FIN

1.2 Ejercicio 2

Describir el siguiente código en un diagrama UML y un pseudocódigo.

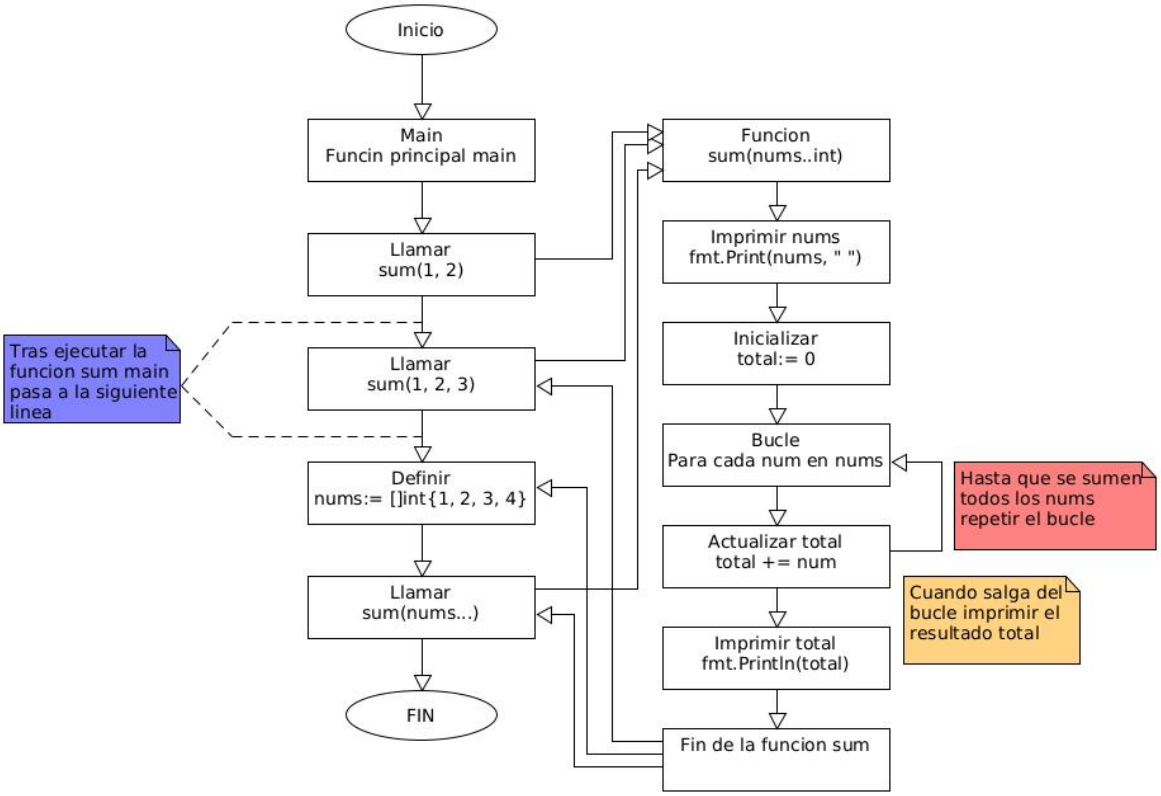
```
1. Función main:
    1.1 Declarar variable grade de tipo string con valor \B"
    1.2 Declarar variable mark de tipo string con valor 90
    1.3 Estructura switch para marks
        1.3.1 Caso 90: Asignar \A" a grade
        1.3.2 Caso 80: Asignar \B" a grade
        1.3.3 Caso 50, 60, 70: Asignar \C" a grade
        1.3.4 Default: Asignar \D" a grade
    1.4 Estructura switch sin expresión
        1.4.1 Caso grade == \A": Mostrar \Excellent!"
        1.4.2 Caso grade == \B" o grade == \C": Mostrar \Well done"
        1.4.3 Caso grade == \D": Mostrar \You passed"
        1.4.4 Caso grade == \F": Mostrar \Better try again"
        1.4.5 Default: Mostrar \Invalid grade"
    1.5 Mostrar mensaje \Your grade is grade"
```


Inicio
Declarar
var grade string = "B"
Declarar
var marks int = 90
Switch de marks
4 casos
Caso 90:
grade = "A"
Case 80:
grade = "B"
Case 50, 60, 70:
grade = "C"
Default:
grade = "D"
Switch
5 casos
Caso grade == "A"
Imprimir Excellent!
Caso grade == "B", "C"
Imprimir Well done
Caso grade == "D"
Imprimir You passed
Caso grade == "F"
Imprimir Better try again
Default:
Imprimir Invalid grade
Imprimir(lib:fmt)
printf("Your grade is %s\n", grade)
FIN

1.3 Ejercicio 3

Describir el siguiente código en un diagrama UML y un pseudocódigo.

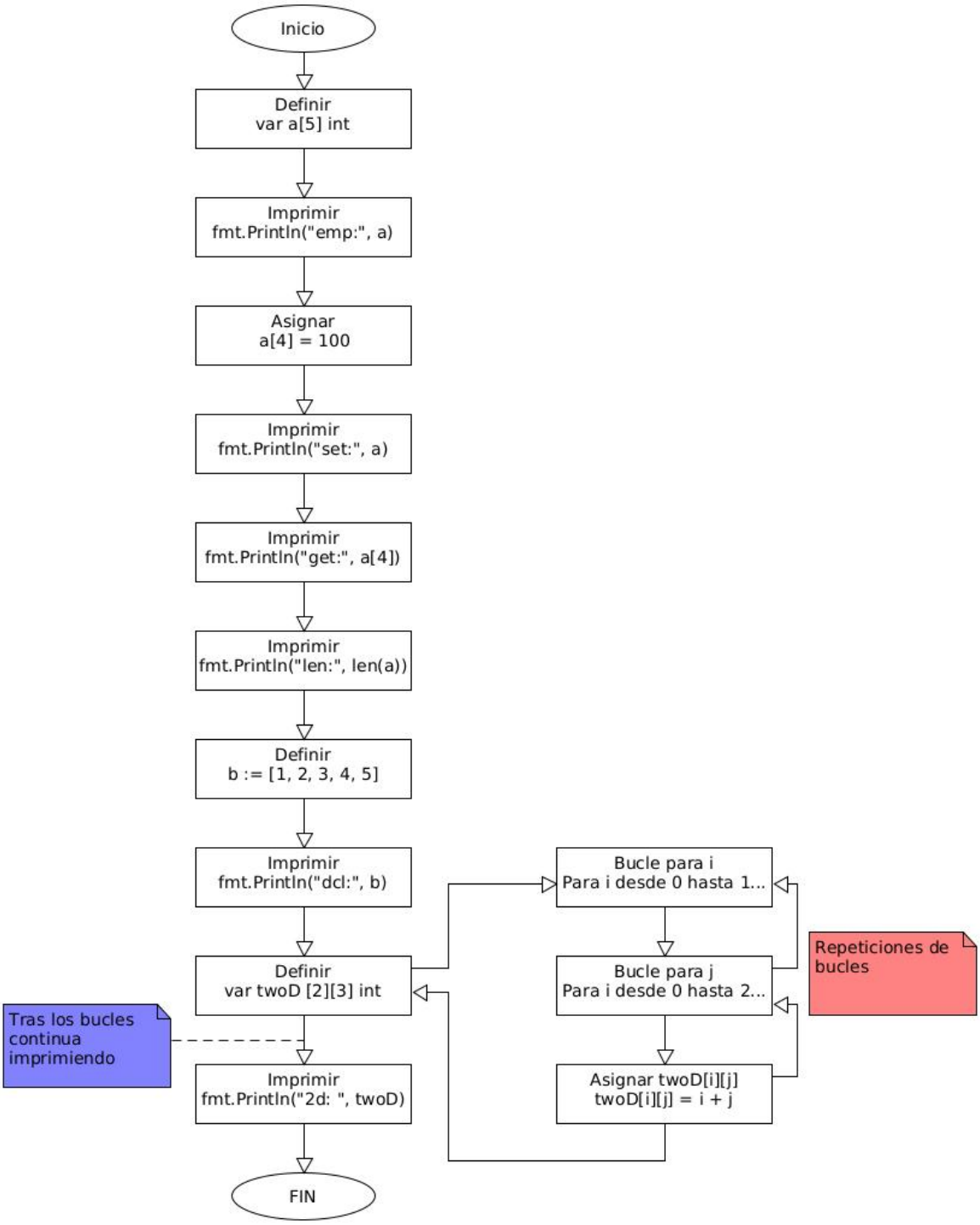
```
1. Función sum(nums ...int):
    1.1 Mostrar nums en la consola
    1.2 Declarar la variable local de tipo entero e inicializarla a 0
    1.3 Para cada num en nums
        1.3.1 Sumar num a total
    1.4 Mostrar total en la consola

2. Función main:
    2.1 Llamar a sum(1, 2)
    2.2 Llamar a sum(1, 2, 3)
    2.3 Declarar variable nums como un array de enteros con valores [1, 2, 3, 4]
    2.4 Llamar a sum(nums...)
```


Inicio
Main
Funcin principal main
Llamar
sum(1, 2)
Llamar
sum(1, 2, 3)
Definir
nums:= []int{1, 2, 3, 4}
Llamar
sum(nums...)
FIN
Tras ejecutar la funcion sum main pasa a la siguiente linea
Funcion
sum(nums..int)
Imprimir nums
fmt.Print(nums, " ")
Inicializar
total:= 0
Bucle
Para cada num en nums
Actualizar total
total += num
Imprimir total
fmt.Println(total)
Fin de la funcion sum
Hasta que se sumen todos los nums repetir el bucle
Cuando salga del bucle imprimir el resultado total

1.4 Ejercicio 4

Describir el siguiente código en un diagrama UML y un pseudocódigo.

```
1. Función main:
    1.1 Declarar array a de tipo entero con tamaño 5
    1.2 Mostrar mensaje \emp: a"
    1.3 Asignar el valor 100 a a[4]
    1.4 Mostrar mensaje \set: a"
    1.5 Mostrar mensaje \get: a[4]"
    1.6 Mostrar mensaje \len: len(a)"
    1.7 Declarar array b de tipo entero con valores [1, 2, 3, 4, 5]
    1.8 Mostrar mensaje \dcl: b"
    1.9 Declarar array bidimensional twoD de tamaño [2][3] de tipo entero
    1.10 Para i desde 0 hasta 1:
    1.11 Para j desde 0 hasta 2:
        1.11.1 Asignar twod[i][j] = i + j
    1.12 Mostrar mensaje \2d: twoD"
```


Inicio
Definir
var a[5] int
Imprimir
fmt.Println("emp:", a)
Asignar
a[4] = 100
Imprimir
fmt.Println("set:", a)
Imprimir
fmt.Println("get:", a[4])
Imprimir
fmt.Println("len:", len(a))
Definir
b := [1, 2, 3, 4, 5]
Imprimir
fmt.Println("dcl:", b)
Definir
var twoD [2][3] int
Tras los bucles
continua
imprimiendo
Imprimir
fmt.Println("2d: ", twoD)
FIN
Bucle para i
Para i desde 0 hasta 1...
Bucle para j
Para i desde 0 hasta 2...
Asignar twoD[i][j]
twoD[i][j] = i + j
Repeticiones de
bucles

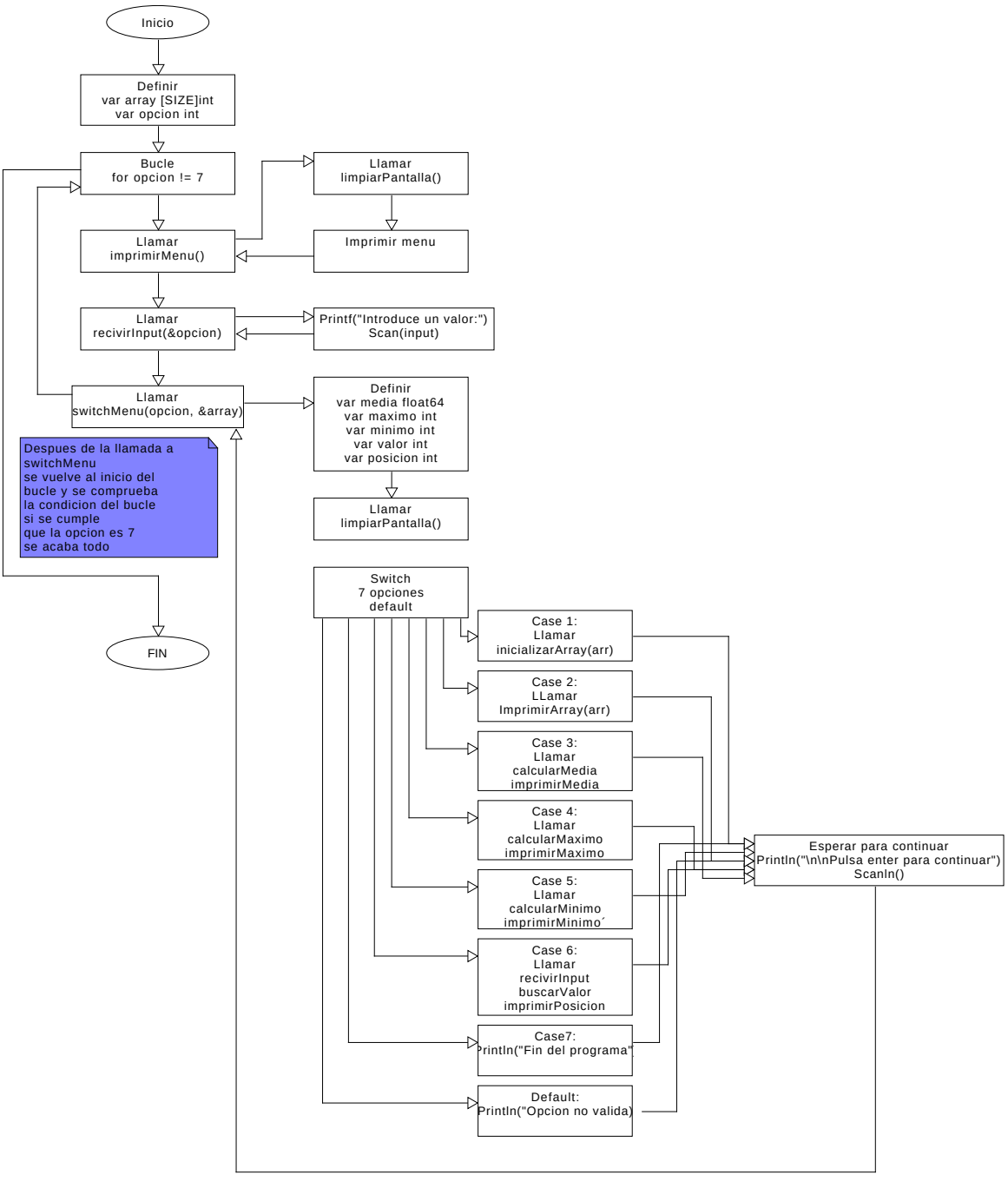
2 Programas

2.1 Pseudocódigo

```
1. Definir constante global SIZE = 5

2. Definir constante global OPCIONES = 7

3. Función main:
    a. Declarar array array de tamaño SIZE
    b. Declarar variable opcion
    c. Mientras opcion != 7:
        i. Llamar a imprimirMenu()
        ii. Llamar a recivirInput(&opcion)
        iii. Llamar a switchMenu(opcion, &array)
    d. Llamar a os.Exit(0) para salir del programa

4. Función inicializarArray(arr):
    a. Para i = 0 hasta SIZE-1:
        i. Leer valor para arr[i]

5. Función imprimirArray(arr):
    a. Si len(arr) == 0, mostrar "El array está vacío"
    b. Si no, para i = 0 hasta SIZE-1:
        i. Mostrar "El valor de la posición i es arr[i]"

6. Función calcularMedia(arr):
    a. Inicializar suma = 0
    b. Para i = 0 hasta SIZE-1:
        i. Sumar arr[i] a suma
    c. Retornar media = suma / SIZE

7. Función calcularMaximo(arr):
    a. Inicializar maximo = arr[0]
    b. Para i = 1 hasta SIZE-1:
        i. Si arr[i] > maximo, asignar arr[i] a maximo
    c. Retornar maximo

8. Función calcularMinimo(arr):
    a. Inicializar minimo = arr[0]
    b. Para i = 1 hasta SIZE-1:
        i. Si arr[i] < minimo, asignar arr[i] a minimo
    c. Retornar minimo

9. Función buscarValor(arr, valor):
    a. Inicializar posicion = -1
    b. Para i = 0 hasta SIZE-1:
        i. Si arr[i] == valor, asignar i a posicion y romper el bucle
```

```
    c. Retornar posicion

10. Función imprimirMenu():
    a. Limpiar pantalla
    b. Mostrar opciones del menú:
        i. 1. Inicializar array
        ii. 2. Imprimir array
        iii. 3. Calcular media
        iv. 4. Calcular valor máximo
        v. 5. Calcular valor mínimo
        vi. 6. Buscar valor
        vii. 7. Salir

11. Función recivirInput(input):
    a. Leer entrada del usuario para input

12. Función switchMenu(opcion, arr):
    a. Según opcion:
        i. Caso 1: Llamar a inicializarArray(arr)
        ii. Caso 2: Llamar a imprimirArray(arr)
        iii. Caso 3: Llamar a calcularMedia(arr) y mostrar el resultado
        iv. Caso 4: Llamar a calcularMaximo(arr) y mostrar el resultado
        v. Caso 5: Llamar a calcularMinimo(arr) y mostrar el resultado
        vi. Caso 6: Llamar a recivirInput(&valor), luego a buscarValor(arr, valor) y
        vii. Caso 7: Mostrar "Fin del programa"
        viii. default: Mostrar "Opción no válida"
```

2.2 Diagrama UML


Inicio
Definir
var array [SIZE]int
var opcion int
Bucle
for opcion != 7
Llamar
limpiarPantalla()
Llamar
imprimirMenu()
Imprimir menu
Llamar
recivirInput(&opcion)
Printf("Introduce un valor:")
Scan(input)
Llamar
switchMenu(opcion, &array)
Definir
var media float64
var maximo int
var minimo int
var valor int
var posicion int
Llamar
limpiarPantalla()
Despues de la llamada a
switchMenu
se vuelve al inicio del
bucle y se comprueba
la condicion del bucle
si se cumple
que la opcion es 7
se acaba todo
Switch
7 opciones
default
FIN
Case 1:
Llamar
inicializarArray(arr)
Case 2:
LLamar
ImprimirArray(arr)
Case 3:
Llamar
calcularMedia
imprimirMedia
Case 4:
Llamar
calcularMaximo
imprimirMaximo
Case 5:
Llamar
calcularMinimo
imprimirMinimo´
Case 6:
Llamar
recivirInput
buscarValor
imprimirPosicion
Case7:
Println("Fin del programa"
Default:
Println("Opcion no valida)
Esperar para continuar
Println("\n\nPulsa enter para continuar")
Scanln()

2.3 Código

```go
package main

import (
  "fmt"
  "os"
)

// Constante tama o del array
const SIZE = 5

// Constante para el n mero de opciones del men
const OPCIONES = 7

/*
1. Implementa una funci n para inicializar las posiciones del array
2. Implementa una funci n para imprimir las posiciones del array
3. Implementa una funci n que devuelva la media
4. Implementa una funci n que devuelva el valor m ximo
5. Implementa una funci n que devuelva el valor m nimo
6. Implementa una funci n que dado un valor, devuelva la posici n del
    array
7. Escribe un programa en el que te declares un array y desde el
   invoques a las anteriores
funciones a modo de men .
*/

// 1. Funci n para inicializar las posiciones del array
func inicializarArray(arr *[SIZE]int) {
  for i := 0; i < SIZE; i++ {
    fmt.Printf("Introduce un n mero para la posici n %d: ", i)
    fmt.Scan(&arr[i])
  }
}

// 2. Funci n para imprimir las posiciones del array
func imprimirArray(arr *[SIZE]int) {
  if len(arr) == 0 {
    fmt.Println("El array est  vac o")
  } else {
    for i := 0; i < SIZE; i++ {
      fmt.Printf("El valor de la posici n %d  es %d\n", i, arr[i])
    }
  }

}

// 3. Funci n que devuelva la media
func calcularMedia(arr *[SIZE]int) float64 {
  // Definimos una variable para almacenar la suma de los valores del
     array
  var suma int
  // Definimos una variable para almacenar la media
  var media float64

  // Recorremos el array y sumamos los valores
  for i := 0; i < SIZE; i++ {
```

```
54      suma += arr[i]
55    }
56
57    // Calculamos la media
58    media = float64(suma) / float64(SIZE)
59
60    return media
61
62
63  }
64
65  // 4. Funci n que devuelva el valor m ximo
66  func calcularMaximo(arr *[SIZE]int) int{
67    // Definimos una variable para almacenar el valor m ximo
68    var maximo int = arr[0]
69    for i := 1; i < SIZE; i++ {
70      if arr[i] > maximo {
71        maximo = arr[i]
72      }
73    }
74
75    return maximo
76  }
77
78  // 5. Funci n que devuelva el valor m nimo
79  func calcularMinimo(arr *[SIZE]int) int{
80    // Definimos una variable para almacenar el valor m nimo
81    var minimo int = arr[0]
82    for i := 1; i < SIZE; i++ {
83      if arr[i] < minimo {
84        minimo = arr[i]
85      }
86    }
87
88    return minimo
89  }
90
91  // 6. Funci n que dado un valor, devuelva la posici n del array
92  func buscarValor(arr *[SIZE]int, valor int) int {
93    // Definimos una variable para almacenar la posici n del valor
94    var posicion int = -1
95    for i := 0; i < SIZE; i++ {
96      if arr[i] == valor {
97        posicion = i
98        break
99      }
100   }
101
102   return posicion
103 }
104
105 // Funci n para imprimir la posici n del valor solicitado
106 func imprimirPosicion(posicion int, valor int) {
107   // Si la posici n es -1, significa que el valor no se encuentra en
        el array
108   if posicion == -1 {
109     fmt.Printf("El valor %d no se encuentra en el array\n", valor)
```

```
110     // Si la posici n es distinta de -1, significa que el valor se
            encuentra en la posici n indicada
111     } else {
112       fmt.Printf("El valor %d se encuentra en la posici n %d\n", valor,
            posicion)
113     }
114   }
115
116
117   // Funcion para imprimir el valor m nimo
118   func imprimirMinimo(minimo int) {
119     fmt.Printf("El valor m nimo es %d\n", minimo)
120   }
121
122   // Funci n para imprimir el valor m ximo
123   func imprimirMaximo(maximo int) {
124     fmt.Printf("El valor m ximo es %d\n", maximo)
125   }
126
127   // Funci n para imprimir la media
128   func imprimirMedia(media float64) {
129     fmt.Printf("La media es %.2f\n", media)
130   }
131
132   // Funcion que limpia la pantalla
133   func limpiarPantalla() {
134     fmt.Println("\033[H\033[2J")
135   }
136
137   // Funci n para imprimir el men
138   func imprimirMenu() {
139     // Antes de imprimir el men , limpiamos la pantalla
140     limpiarPantalla()
141
142     // Imprimimos el men
143     fmt.Println("1. Inicializar array")
144     fmt.Println("2. Imprimir array")
145     fmt.Println("3. Calcular media")
146     fmt.Println("4. Calcular valor m ximo")
147     fmt.Println("5. Calcular valor m nimo")
148     fmt.Println("6. Buscar valor")
149     fmt.Println("7. Salir")
150   }
151
152   // Funcion para leer del teclado
153   func recivirInput(input *int) {
154     // Pedimos al usuario que introduzca un input
155     fmt.Printf("Introduce un valor: ")
156     // Recibimos el input
157     fmt.Scan(input)
158   }
159
160
161
162   // Funcion switch para llamar a las funciones seg n la opci n elegida
163   func switchMenu(opcion int, arr *[SIZE]int) {
164     // Definimos una variable para almacenar la media, maximo, minimo,
            valor y posicion(pide que als funciones devuelvan un valor)
```

```
165     var media float64
166     var maximo int
167     var minimo int
168     var valor int
169     var posicion int
170 
171 
172     // Limpiamos la pantalla
173     limpiarPantalla()
174 
175     // Llamamos a la funci n correspondiente seg n la opci n elegida
176     switch opcion {
177     case 1:
178       // Llamamos a la funci n inicializarArray
179       inicializarArray(arr)
180     case 2:
181       // Llamamos a la funci n imprimirArray
182       imprimirArray(arr)
183     case 3:
184       // Llamamos a la funci n calcularMedia
185       media = calcularMedia(arr)
186       imprimirMedia(media)
187     case 4:
188       // Llamamos a la funci n calcularMaximo
189       maximo = calcularMaximo(arr)
190       imprimirMaximo(maximo)
191     case 5:
192       // Llamamos a la funci n calcularMinimo
193       minimo = calcularMinimo(arr)
194       imprimirMinimo(minimo)
195     case 6:
196       // Pedimos al usuario que introduzca un valor a buscar
197       recivirInput(&valor)
198       posicion = buscarValor(arr, valor)
199       imprimirPosicion(posicion, valor)
200     case 7:
201       // Salimos del programa
202       fmt.Println("Fin del programa")
203     default:
204       // Si la opci n no es v lida, mostramos un mensaje de error
205       fmt.Println("Opci n no v lida")
206     }
207     // Esperar unos segundos para que el usuario pueda leer el mensaje
208     fmt.Println("\n\nPulsa enter para continuar")
209     fmt.Scanln()
210 
211 }
212 
213 func main() {
214     // Definimos el array con el tama o SIZE
215     var array [SIZE]int
216     // Definimos una variable para almacenar la opci n del men
217     var opcion int
218 
219     // Mientras que la opci n no sea 7, se seguir  ejecutando el
          programa
220     for opcion != 7 {
221       // Imprimimos el men
```

```
222     imprimirMenu()
223     // Recivimos la opci n elegida
224     recivirInput(&opcion)
225     // Llamamos a la funci n switchMenu
226     switchMenu(opcion, &array)
227   }
228
229   // Salimos del programa
230   os.Exit(0)
231
232 }
```

Listing 1: Código del programa