Universidad de las Fuerzas Armadas ESPE

Departamento: Ciencias de la computación

Carrera: Ingeniería en Electricidad y Automatización

U2T1: Vectores y funciones

1. Información General

- **Asignatura:** Fundamentos de Programación
- **Apellidos y nombres de los estudiantes:** Astudillo Adriana, Muñoz Sarahi, Nero Alan
- **NRC:** 20823
- **Fecha de realización:** 10/06/2025

Problema 2.1.3 Vector con término general dado. Sea la sucesión:
$k=k^2+3$,

desarrolle un programa que lea el número n de componentes que se quieren calcular de la sucesión y almacenarlas en un vector vec, tal que vec(i) = ı. Se mostraré vector por pantalla. Puede asumir que n será siempre menor o igual a 100.
Para calcular las componentes del vector se utilizará una iteración con un índice amando valores de 1 a n en diagrama de flujo (de 0 an-1 en C). A la vez, se ira calculando la componente (vec(i) = i^2+3) y mostrándola por pantalla.

- **Requisitos funcionales**

- **RF_1:** El sistema debe solicitar al usuario el número de términos (n) a generar, con un máximo de MAX=100

- **RF_2:** El sistema debe validar que n esté en el rango permitido (1 a 100). Si no, mostrará un mensaje de error y terminará.
- **RF_3:** El sistema debe calcular cada término de la sucesión usando la fórmula $Vk=k2+3$ $V_k=k_2+3$ y almacenarlo en un vector.
- **RF_4:** El sistema debe imprimir los términos generados en formato lista: Termino k=k²+3

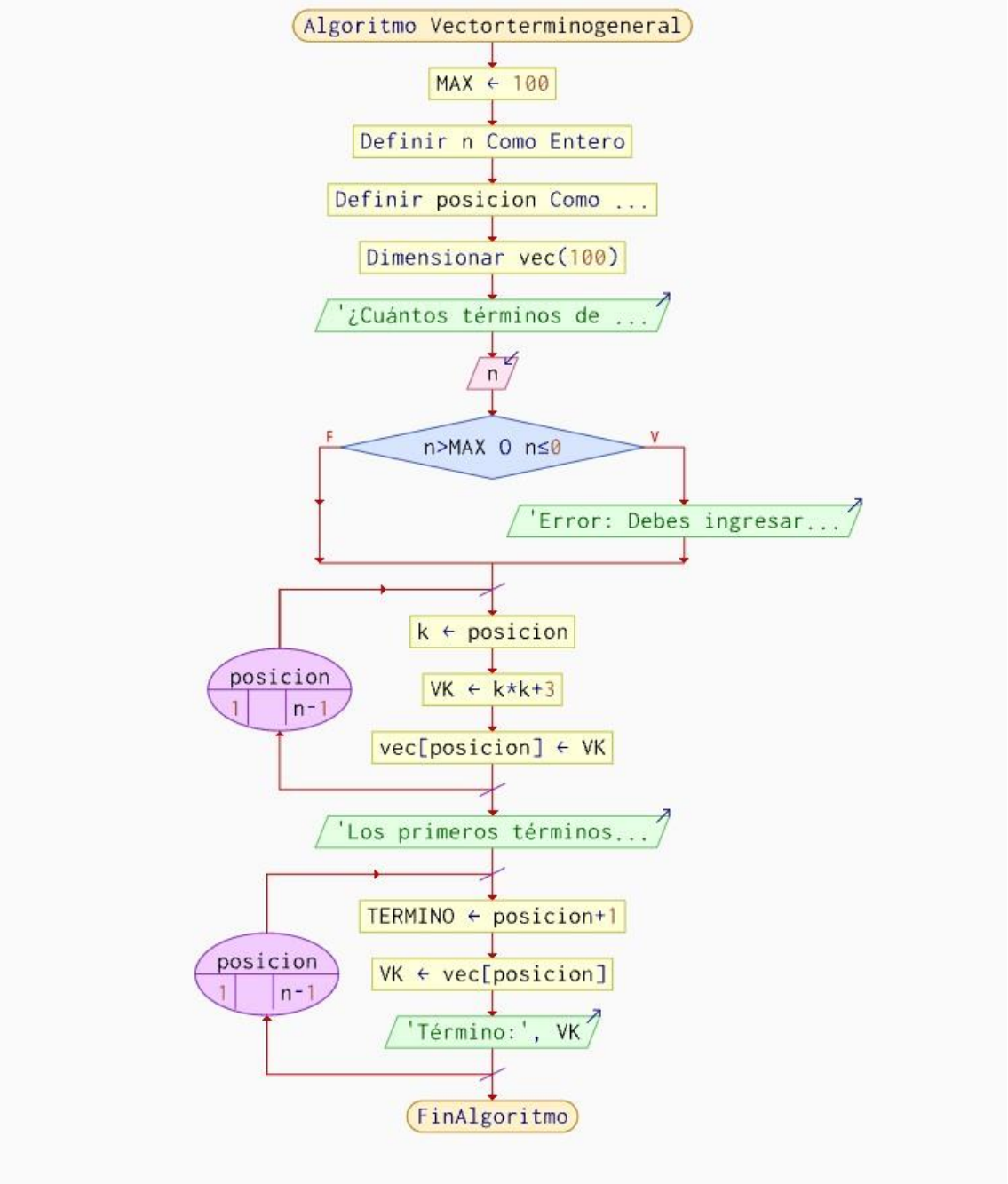
➢ **Tabla de objetos**

objetos	nombre	valor	Tipo
Máximo	MAX	constante	Entero
Numero de sucesión	n	variable	Entero
posición	posición	constante	entero
Numero de posición en la sucesión	K	Constante	Entero
Termino	Termino	constante	entero

➢ **Pseint**

```
1  Algoritmo Vectorterminogeneral
2      MAX=100
3      Definir n como entero
4      Definir posicion como entero
5      Dimensionar vec[100]
6      Escribir "¿Cuántos términos de la sucesión deseas generar?"
7      LEER n
8      SI n > MAX O n ≤ 0 ENTONCES
9          MOSTRAR "Error: Debes ingresar un número entre 1 y 100"
10     FIN SI
11     PARA posición desde 1 hasta n-1 HACER
12         k = posicion
13         VK = k * k + 3
14         vec[posición]= VK
15     FIN PARA
16     Escribir "Los primeros términos de la sucesión son:",VK
17
18     PARA posición desde 1 hasta n-1 HACER
19         término = posición + 1
20         VK = vec[posición]
21         MOSTRAR "Término:", VK
22     FIN PARA
23     FinAlgoritmo
```

- **DF**



- **Código c**

```
#include <stdio.h>
#define MAX 100
```

```
//El sistema debe solicitar al usuario el número de términos (n) a generar, con un máximo de MAX = 100.
//El sistema debe validar que n esté en el rango permitido (1 a 100). Si no, mostrará un mensaje de error y terminará.
//El sistema debe calcular cada término de la sucesión usando la fórmula y almacenarlo en un vector.
//El sistema debe imprimir los términos generados en formato lista.
int main() {
    int n;
    int vec[MAX];
    int k, VK;
    printf("¿Cuántos términos de la sucesión deseas generar? (1-%d): ", MAX);
    scanf("%d", &n);
    if (n > MAX || n <= 0) {
        printf("Error: Debes ingresar un número entre 1 y %d\n", MAX);
        return 1;
    }
    for (int posicion = 0; posicion < n; posicion++) {
        k = posicion + 1;
        VK = k * k + 3;
        vec[posicion] = VK;
    }
    printf("\nLos primeros %d términos de la sucesión son:\n", n);
    for (int posicion = 0; posicion < n; posicion++) {
        printf("Término %d: %d\n", posicion + 1, vec[posicion]);
    }

    return 0;
}
```

Problema 2.1.4 - Comprobar si dos valores pertenecen a un vector:

Realice un algoritmo que lea dos números enteros por teclado y determine si ambos valores forman parte de un vector de enteros previamente definido de dimensión n*n*. La solución se basa en dos variables bandera, que representan si uno de los números está en el vector. Se inicializan ambas a 0, y se recorre el vector comparando cada componente con los valores leídos por el teclado. Si alguno coincide, se cambia el valor de la bandera asociada a 1. Al finalizar, si ambas valen 1, el resultado será positivo.

- **Requisitos funcionales**

RF_1: El sistema debe inicializar un vector con 15 valores predefinidos
RF_2: El sistema debe solicitar al usuario dos números enteros (n_1 y n_2)
RF_3: El sistema debe buscar ambos números en el vector
RF_4: El sistema debe mostrar mensajes claros indicando:

- Si ambos números están presentes
- Si solo uno está presente
- Si ninguno está presente

- **Tabla de objetos**

Objeto	nombre	valor	Tipo
Numero 1	n_1	Variable	Entero
Número 2	n_2	Variable	Entero
Vector	vec	constante	Entero
Bandera n_1	bandera n_1	Variable	Entero
Banderan_2	bandera n_2	Variable	Entero

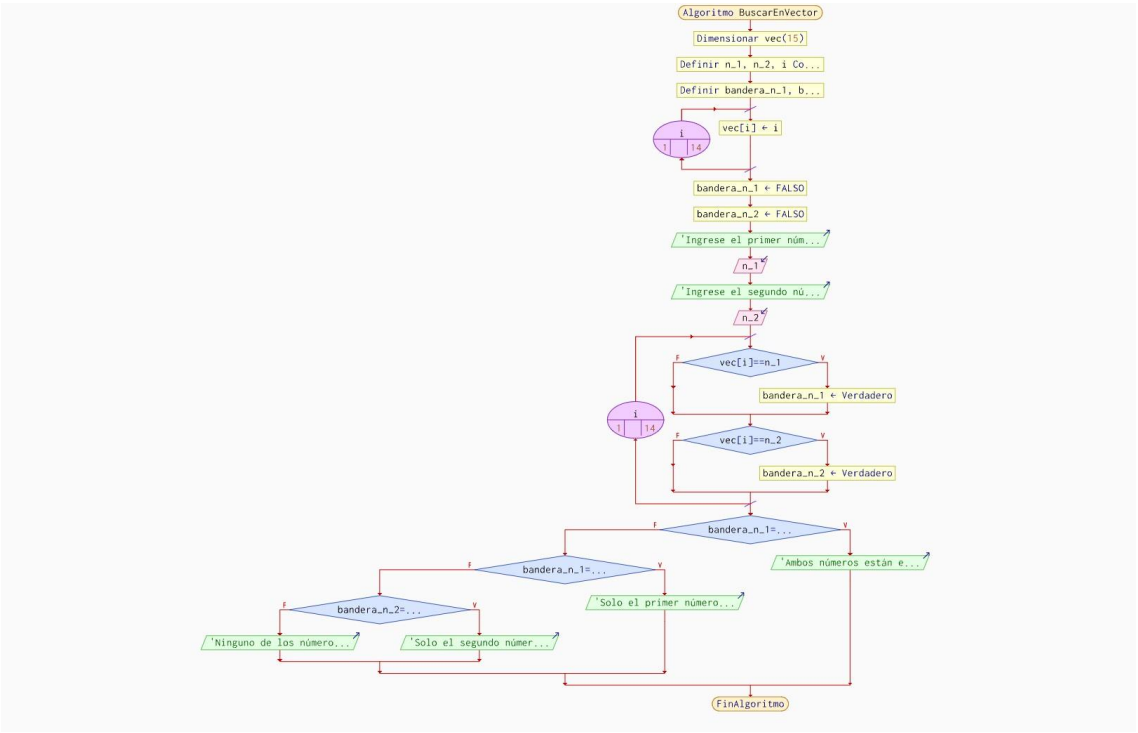
- **Pseint**

```
Algoritmo BuscarEnVector
Dimensionar vec[15]
Definir n_1, n_2, i como entero
Definir bandera_n_1, bandera_n_2 como logico
Para i ← 1 Hasta 14 Hacer
    vec[i] ← i
FinPara
    bandera_n_1 = FALSO
    bandera_n_2 = FALSO
    Escribir "Ingrese el primer número a buscar: "
    Leer n_1
    Escribir "Ingrese el segundo número a buscar: "
    Leer n_2
    Para i desde 1 hasta 14 Hacer
        Si vec[i] == n_1 Entonces
            bandera_n_1= Verdadero
        FinSi

        Si vec[i] == n_2 Entonces
            bandera_n_2 = Verdadero
        FinSi
    FinPara
    Si bandera_n_1 == Verdadero Y bandera_n_2 == Verdadero Entonces
        Escribir "Ambos números están en el vector"
    Sino Si bandera_n_1 == Verdadero Entonces
            Escribir "Solo el primer número está en el vector"
        Sino Si bandera_n_2 == Verdadero Entonces
                Escribir "Solo el segundo número está en el vector"
            Sino
                Escribir "Ninguno de los números está en el vector"
            FinSi
        FinSi
    FinSi

FinAlgoritmo
```

- **DF**



- **Código c**

```
#include <stdio.h>
#include <stdbool.h>
// El sistema debe inicializar un vector de tamaño 15 con valores del 1 al 14.
//El sistema debe solicitar al usuario dos números enteros (n_1 y n_2).
//El sistema debe buscar ambos números en el vector y actualizar las banderas correspondientes si los encuentra.
//El sistema debe mostrar un mensaje indicando si:
//Ambos números están en el vector
//Solo el primer número está en el vector
//Solo el segundo número está en el vector.
//Ninguno de los números está en el vector.

#define SIZE 15

int main() {
    int vec[SIZE];
    int n_1, n_2;
    bool bandera_n_1 = false;
```

```
    bool bandera_n_2 = false;

    for (int i = 1; i <= 14; i++) {
        vec[i] = i;
    }
    printf("Ingrese el primer número a buscar: ");
    scanf("%d", &n_1);
    printf("Ingrese el segundo número a buscar: ");
    scanf("%d", &n_2);
    for (int i = 1; i <= 14; i++) {
        if (vec[i] == n_1) {
            bandera_n_1 = true;
        }
        if (vec[i] == n_2) {
            bandera_n_2 = true;
        }
    }
    if (bandera_n_1 && bandera_n_2) {
        printf("Ambos números están en el vector\n");
    } else if (bandera_n_1) {
        printf("Solo el primer número está en el vector\n");
    } else if (bandera_n_2) {
        printf("Solo el segundo número está en el vector\n");
    } else {
        printf("Ninguno de los números está en el vector\n");
    }

    return 0;
}
```

Problema 2.1.5 - Vector de factoriales:

Dado un vector Vec, que contiene los primeros 15 números naturales, calcule un vector factor con sus factoriales y mostrarlo por pantalla.

Nota: ¿Por qué a partir del número 12 no funciona correctamente el cálculo del factorial? ¿Qué se podría hacer para evitarlo?

El algoritmo consiste en dos bucles anidados, uno externo, que da valores a la variable i*i* entre 1 y 15, y otro interno, que calcula el factorial de la componente e*n*ésima. El

algoritmo deja de funcionar para el valor 12 porque en la codificación en C se han empleado variables de tipo entero. El valor de 13! excede la capacidad de almacenamiento de una variable entera, y el cálculo se corrompe. Una posible solución es emplear variables de tipo *unsigned long int* (entero largo sin signo), que incrementan la capacidad de almacenamiento.

- **Requisitos funcionales**

- **RF_1:**El sistema debe almacenar números enteros del 1 al 15 en un arreglo (Vec).
- **RF_2:** El sistema debe calcular el factorial de cada número en Vec y almacenarlo en otro arreglo (Fact).
- **RF_3:** El sistema debe imprimir una tabla con dos columnas: Número y Factorial, formateada para alinear los valores.
- **RF_4:** El sistema debe usar el tipo unsigned long long para evitar desbordamiento en factoriales grandes.
-

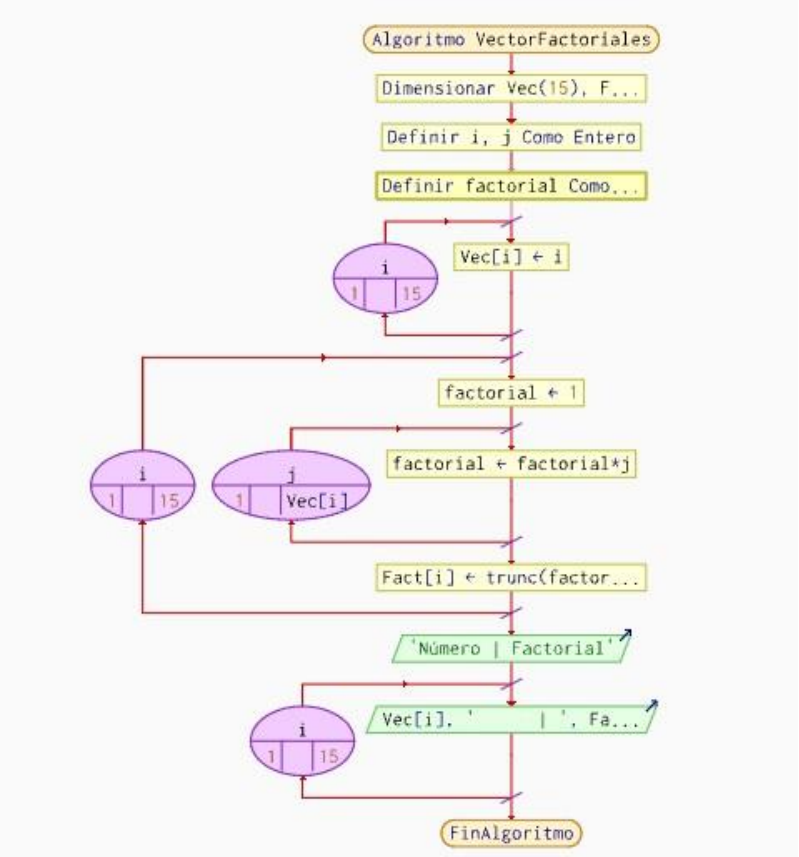
- **Tabla de objetos**

Objeto	Nombre	Valor	Tipo
Vector original	Vec	[1, 2, 3, ..., 15] Constante	Enteros
Vector factorial	Fact	[1!, 2!, 3!, ..., 15!] Variable	Enteros
Contador externo	i	1 a 15 Variable	Entero
Contador interno	j	1 a Vec[i] Variable	Entero
Valor factorial	factorial	Temporal para cálculo (1, 2, 6,..) Variable	Entero
Límite máximo	n	15 Constante	Entero

- **Pseint**

```
1   Algoritmo VectorFactoriales
2       Dimension Vec[15], Fact[15]
3       Definir i, j Como Entero
4       Definir factorial Como Real
5       Para i ← 1 Hasta 15 Hacer
6           Vec[i] ← i
7       FinPara
8
9       Para i ← 1 Hasta 15 Hacer
10          factorial ← 1
11
12          Para j ← 1 Hasta Vec[i] Hacer
13              factorial ← factorial * j
14          FinPara
15
16          Fact[i] ← trunc(factorial)
17      FinPara
18
19      Escribir "Número | Factorial"
20      Para i ← 1 Hasta 15 Hacer
21          Escribir Vec[i], "      | ", Fact[i]
22      FinPara
23  FinAlgoritmo
```

- **DF**



- **Código c**

```
#include <stdio.h>
#include <limits.h> // Para INT_MAX
//El sistema debe almacenar números enteros del 1 al 15 en un arreglo (Vec).
//El sistema debe calcular el factorial de cada número en Vec y almacenarlo en otro arreglo (Fact).
//El sistema debe imprimir una tabla con dos columnas: Número y Factorial, formateada para alinear los valores.
//El sistema debe usar el tipo unsigned long long para evitar desbordamiento en factoriales grandes.
int main() {
    const int n = 15;
    int Vec[n];

unsigned long long Fact[n];

    for(int i = 0; i < n; i++) {
        Vec[i] = i + 1;
    }

    for(int i = 0; i < n; i++) {
        Fact[i] = 1;

        for(int j = 1; j <= Vec[i]; j++) {
            Fact[i] *= j;
        }
    }

    printf("Número | Factorial\n");
    printf("-----------------\n");
    for(int i = 0; i < n; i++) {
        printf("%6d | %20llu\n", Vec[i], Fact[i]);
    }

    return 0;
}
```

Problema 2.1.6 - Ordenación de un vector:

Desarrolle un programa que ordene un vector de 10 componentes de mayor a menor valor. Asuma que el vector está ya leído y almacenado en memoria.

Existen varios métodos de ordenación. Se expondrán dos:

Ordenación iterativa: Esta solución emplea dos bucles anidados para comparar cada elemento del vector con los que le siguen, intercambiando las parejas de elementos fuera de orden. Por ejemplo, para ordenar un vector de n*n* elementos numerados de 0 a n−1*n*−1 (en DF de 1 a n*n*), se comienza comparando el elemento 0 con los que le siguen, es decir, las parejas de elementos (0-1), (0-2), ...(0-n-1). Si alguna pareja presenta un orden inverso al deseado, se intercambian las posiciones de sus elementos en el vector. Tras comparar la última pareja, se tiene la garantía de que el elemento de mayor valor está en la posición 0. El procedimiento se repite a continuación con el elemento 1, esto es, se estudian las parejas (1-2), (1-3), ...(1-n-1), intercambiando aquellas fuera de orden. El procedimiento se repite hasta llegar al elemento n−2*n*−2, que debe ser comparado con el último, es decir, n−2*n*−2 con n−1*n*−1.

- **Requisitos funcionales:**

- **RF_1:** El sistema debe permitir al usuario ingresar 10 valores enteros y almacenarlos en un vector.
- **RF_2:** El sistema debe ordenar los números en orden descendente (de mayor a menor).
- **RF_3:** El sistema debe imprimir el vector ordenado en formato lista.
- **RF_4:** El algoritmo de ordenamiento debe ser eficiente para entradas pequeñas

- **Tabla de objetos.**

Objetos	**Nombre**	**Valor**	**Tipo**
Vector	vector	constante	Entero
Valor i	i	Variable	Entero
Valor j	j	Variable	Entero

➢ **Pseint**

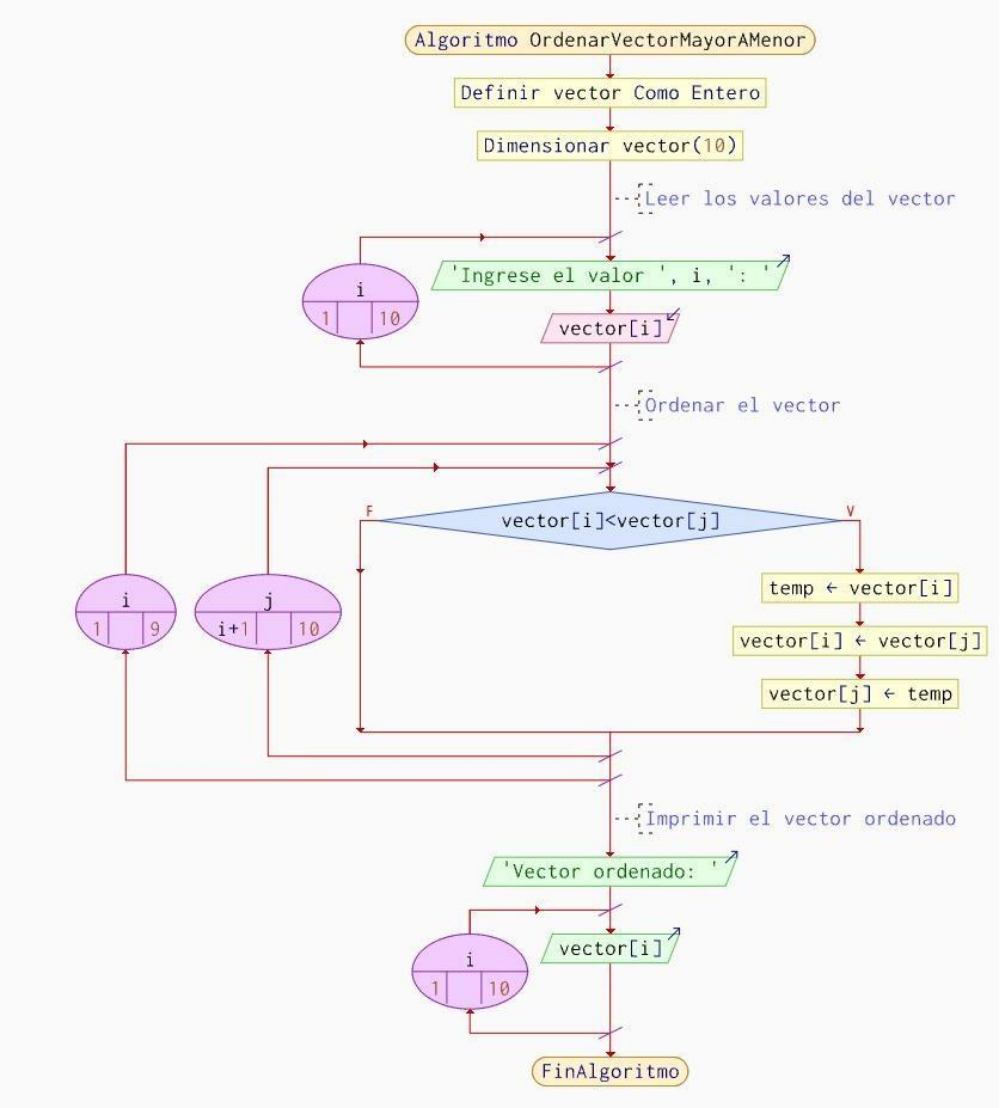
```
Algoritmo OrdenarVectorMayorAMenor
    Definir vector Como Entero
    Definir i, j, temp Como Entero
    Dimension vector[10]
    Para i ← 0 Hasta 9 Hacer
        Escribir "Ingrese el valor ", i+1, ": "
        Leer vector[i]
    FinPara

    Para i ← 0 Hasta 8 Hacer
        Para j ← i+1 Hasta 9 Hacer
            Si vector[i] < vector[j] Entonces

                temp ← vector[i]
                vector[i] ← vector[j]
                vector[j] ← temp
            FinSi
        FinPara
    FinPara

    Escribir "Vector ordenado (de mayor a menor):"
    Para i ← 0 Hasta 9 Hacer
        Escribir "Posición ", i+1, ": ", vector[i]
    FinPara
FinAlgoritmo
```

- **DF**



- **Código en c.**

```
#include <stdio.h>
int main() {
//El sistema debe permitir al usuario ingresar 10 valores enteros y almacenarlos en un
  vector.
//El sistema debe ordenar los números en orden descendente (de mayor a menor).
//El sistema debe imprimir el vector ordenado en formato lista.
//El algoritmo de ordenamiento debe ser eficiente para entradas pequeñas
    int i;
    int j;
```

```
    int temp;
    int vector[10];

    /* Leer los valores del vector */
    for (i = 0; i < 10; i++) {
        printf("Ingrese el valor %d: \n", i + 1);
        scanf("%d", &vector[i]);
    }

    /* Ordenar el vector */
    for (i = 0; i < 9; i++) {
        for (j = i + 1; j < 10; j++) {
            if (vector[i] < vector[j]) {
                temp = vector[i];
                vector[i] = vector[j];
                vector[j] = temp;
            }
        }
    }
    /* Imprimir el vector ordenado */
    printf("Vector ordenado: \n");
    for (i = 0; i < 10; i++) {
        printf("%d\n", vector[i]);
    }
    return 0;
}
```