**CI-2125 Computación I**

**Practica 6**

**Contenido: Arreglos. Funciones con Arreglos.**

1. Diga cuál será el resultado de correr los siguientes trozos de programa:

```
main ( ){
      intj,k;
      intprimero[21] = {7,1,8,2,9,3,10,4,-1,5,-2,6};
      int segundo[21];
      for (j=0; j<6; j++)
            segundo[j] = primero[2*j] + j;
      for (k=3; k<7; k++)
            printf("%d %d\n", primero[k+1], segundo[k-1]);
}
```

**Solución:**

La salida será 9 11
3 13
10 3
4 3

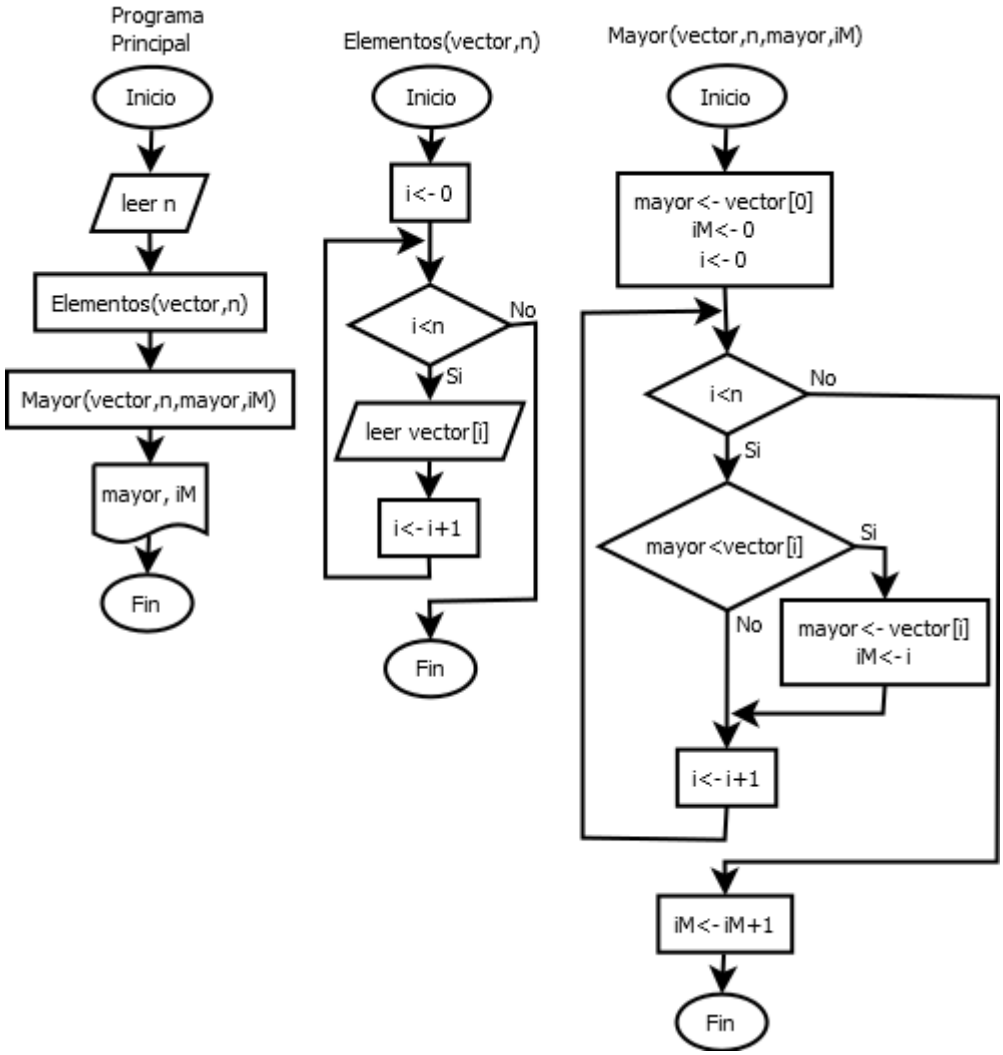
2. Diseñe un algoritmo que llene un vector de a lo sumo 20 elementos reales e imprima la posición y el valor del elemento mayor almacenado en el vector. ¿Qué debe modificarle al algoritmo para que también calcule el menor elemento? Escriba el programa equivalente en C.

**Solución:**

```
Programa Principal:
      ENTRADAS:   vector (arreglo de reales)
                  n (entero)
      PRECONDICIÓN: 0<n≤20

      SALIDAS:    mayor (real)
                  iM (entero) // la posición del mayor

      POSTCONDICIÓN: mayor=vector[iM] y 0≤iM<n

Procedimiento Elementos(vector,n)
      ENTRADAS: n (entero), parámetro por valor
      PRECONDICIÓN: 0<n≤20
      SALIDA: vector (arreglo de reales), parámetro por referencia
      POSTCONDICIÓN: vector contiene los valores leidos

FUNCION Mayor(vector,n,mayor,iM)
      ENTRADAS:
            vector (arreglo de reales)
            n (entero)
```

```
PRECONDICIÓN: 0<n≤20
SALIDAS:
      mayor (real), parámetro por referencia
      iM (entero), parámetro por referencia
POSTCONDICIÓN: mayor contiene el mayor valor del vector y su posición
               queda en iM
```

DIAGRAMA DE FLUJO:



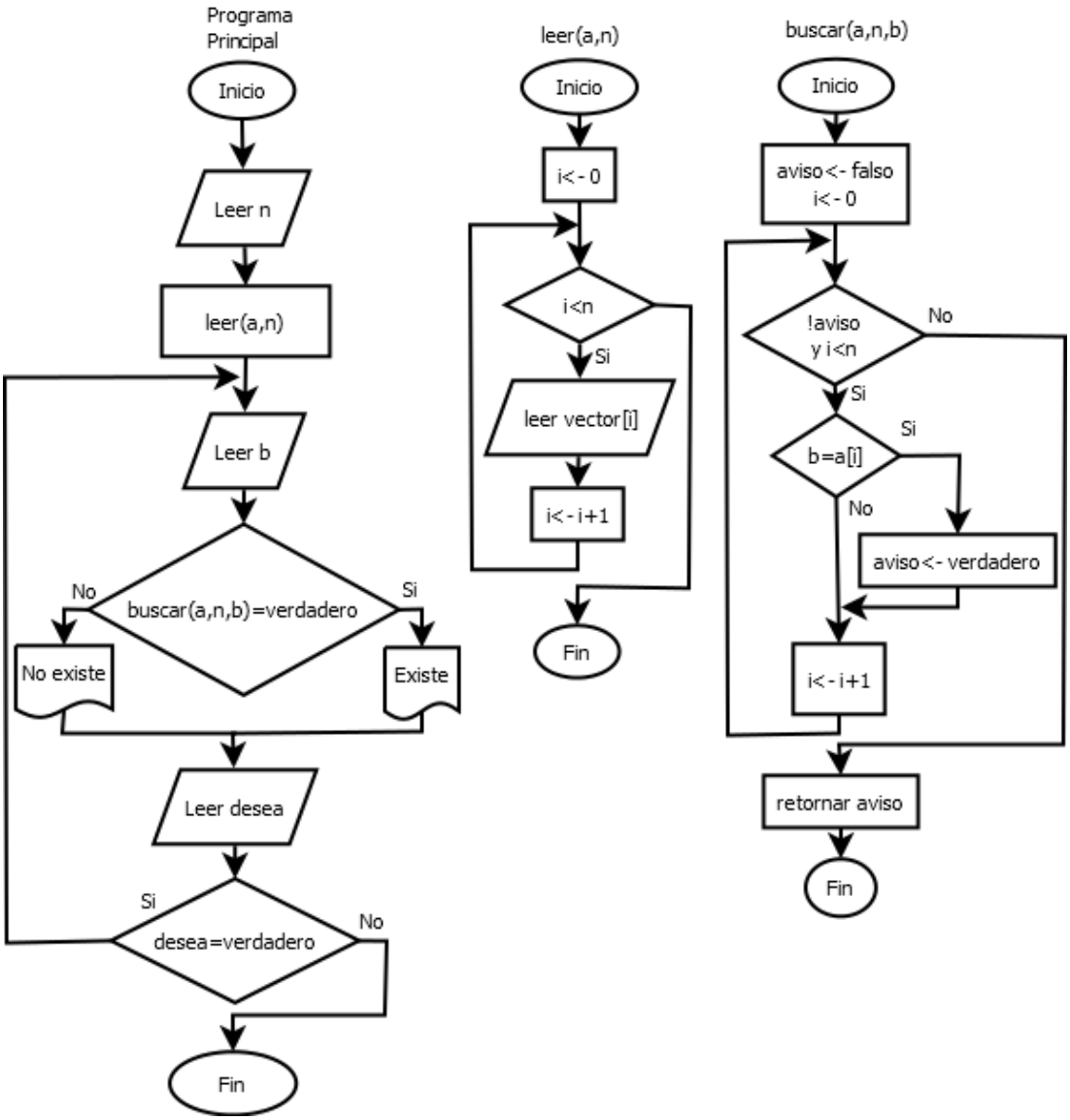

Para calcular el menor sólo se necesita agregar dos variables adicionales de salida menor y im, que almacenen el valor del menor y su posición. Estas variables son inicializadas antes del ciclo, con los mismos valores vector[0] y 0, respectivamente. En el ciclo se incluye otro if que cheque si vector[i] < menor y se hacen los mismos cambios que para el mayor.

3. Diseñe un algoritmo que lea N de enteros, los almacene en un arreglo A y luego permita que el usuario pregunte si existen valores en dicho arreglo. N es dado por el usuario y no puede ser mayor a 20. Luego que son leídos los elementos del arreglo A, el algoritmo pide al usuario un valor X y busca dicho valor en el arreglo, indicando si X está o no en A. Posteriormente se da la

opción de buscar otro valor X, permaneciendo en un ciclo hasta que el usuario lo desee. Escriba el programa equivalente en C.

**Solución:**

```
Programa Principal:
	ENTRADAS:
		A (arreglo de enteros)
		N (entero)
		X (entero)
	PRECONDICIÓN: 0<N<=20
	POSTCONDICION: dice si cada X leído está o no en el arreglo

Procedimiento leer(A,N)
	ENTRADAS: N (entero), parámetro por valor
	PRECONDICIÓN: 0<N<=20
	SALIDA: A (arreglo de enteros), parámetro por referencia
	POSTCONDICIÓN: A contiene los valores leidos

FUNCION buscar(A,N,X)
	ENTRADAS:
		A (arreglo de enteros)
		N (entero)
		X (entero)
	PRECONDICIÓN: 0<N<=20
	SALIDA:
		Retorna un valor booleano
	POSTCONDICIÓN: buscar devuelve VERDADERO si X está en el arreglo y
	FALSO en caso contrario

DIAGRAMA DE FLUJO: (siguiente página)
```

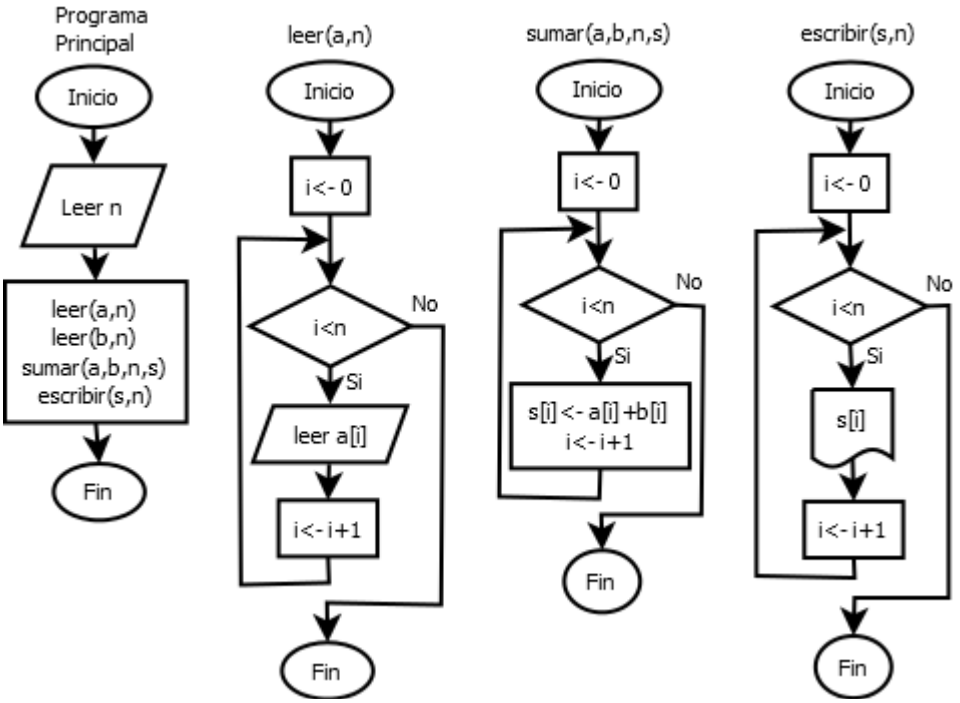



4. Diseñe un algoritmo que permita leer dos vectores de números reales, calcule la suma de ellos y escriba el vector suma. Para ello diseñe un subprograma que lea los valores de un vector de tamaño N. El valor de N es leído en el programa principal y no puede ser mayor que 10. Además, diseñe un segundo subprograma que calcule la suma de dos vectores de tamaño N y un tercer subprograma que escriba un vector de tamaño N en pantalla con la notación $(v_1,v_2,\ldots,v_N)$.

**Solución:**

```
Programa Principal:
      ENTRADAS:
            A,B (arreglos de reales)
            N (entero)
      PRECONDICIÓN: 0<N≤10
      SALIDAS: S (arreglo de reales)
```

```
        POSTCONDICION: S[i] = A[i]+B[i], para todo 0≤i<N

Procedimiento leer(V,N)
        ENTRADAS: N (entero), parámetro por valor
        PRECONDICIÓN: 0<N≤10
        SALIDA: V (arreglo de reales), parámetro por referencia
        POSTCONDICIÓN: V contiene los valores leídos del vector

Procedimiento sumar(N,A,B)
        ENTRADAS:
                N (entero)
                A, B (arreglos de reales)
        PRECONDICIÓN: 0<N≤10
        SALIDA:
                S (arreglo de reales)
        POSTCONDICIÓN: S[i] = A[i]+B[i], para todo 0≤i<N

Procedimiento escribir(N,V)
        ENTRADAS:
                N (entero)
                V (arreglo de reales)
        PRECONDICIÓN: 0<N≤10
        POSTCONDICIÓN: escribe en pantalla el contenido del arreglo V en notación
        vectorial (v1,..,vN)
```




5. Diseñe un algoritmo que permita leer los coeficientes de un polinomio y los almacene en un arreglo, luego lee un valor real X y calcula el resultado de evaluar el polinomio en X. Por ejemplo, para el polinomio $P(X) = X^2+3X-5$, si X = 2, el resultado de evaluar P(X) en 2 es 5. Los coeficientes del polinomio son valores reales y antes de leerlos debe pedirse al usuario el valor del grado que es un entero no negativo, el cual no puede ser mayor que 10. Use la técnica del análisis descendente para dividir el algoritmo en subprogramas. Escriba un programa equivalente en C.

Solución:

```
Programa Principal:
	ENTRADAS:
		grado (entero)
		P (arreglo de reales)
		X (real)
	PRECONDICIÓN: 0≤grado≤10
	SALIDAS: eval (real)
	POSTCONDICION: eval = P(X), el resultado de evaluar P en X

Procedimiento leer(P,grado)
	ENTRADAS: grado (entero), parámetro por valor
	PRECONDICIÓN: 0≤grado≤10
	SALIDA: P (arreglo de reales), parámetro por referencia
	POSTCONDICIÓN: P contiene los grado+1 coeficientes del polinomio


Función evaluar(grado,P,X)
	ENTRADAS:
		grado (entero)
		P (arreglo de reales)
		X (real)
	PRECONDICIÓN: 0≤grado≤10
	SALIDA:
		retorna un real
	POSTCONDICIÓN: evaluar contiene el valor de P(X)
```