Fundamentos de Programación – 1º ASI

# Ejercicios de vectores y matrices resueltos

A continuación se presenta el código fuente de algunos problemas del tema 3

## Ejercicio 1

Crear un vector de 100 enteros e inicializarlo con el número -1.

```
int main(void)
{
    int v[100], i;

    // Inicializamos el vector
    for(i=0; i<100; i++)
       v[i] = -1;

    // Lo mostramos por la pantalla para comprobar que se ha inicializado bien
    for (i=0; i<100; i++)
       printf("v[%i] = %i\n", i, v[i]);

    system("PAUSE");
    return 0;
}
```

## Ejercicio 2

Crear dos vectores e introducir en uno de ellos los 100 primeros números pares y, en el otro, los 100 primeros números impares.

```
int main(void)
{
    int pares[100], impares[100];// Los dos vectores
    int i;

    // Inicializamos los vectores
    for (i = 0; i < 100; i++)
    {
       pares[i] = i * 2;
       impares[i] = (i * 2) + 1;
    }

    // Mostramos el contenido de los vectores en la pantalla
    // (para ver si todo ha ido bien)
  printf("\n\nPares: ");
    for (i = 0; i < 100; i++)
       printf("%i ", pares[i]);
    printf("\n\nImpares: ");
    for (i = 0; i < 100; i++)
       printf("%i ", impares[i]);

    system("PAUSE");
    return 0;
}
```

## Ejercicio 3

Escribe un programa que genere al azar una combinación para jugar a la lotería primitiva asegurándote de que ningún número se repite.

```
int main(void)
{
      int n, i, j, repetido, v[6];

      srand(time(NULL));// Inicialización del generador de números aleatorios
      for (i = 0; i < 6; i++)
```

```
	{
		// Sacamos números repetidamente hasta dar con uno que no haya salido todavía
		do
		{
			n = rand() % 49 + 1; // Sacamos un número al azar entre 1 y 49
			repetido = 0;
			for (j = 0; j < i; j++) // Comprobamos si el número ha salido ya
			{
				if (v[j] == n) // El número ya ha salido !!
					repetido = 1;
			}
		} while (repetido == 1); // Si el número ya ha salido, repetimos para sacar otro

		v[i] = n;		// Guardamos el número elegido en el vector v
	}

	// Mostramos la combinación
	printf("Combinación para la primitiva\n");
	for (i = 0; i < 6; i++)
		printf("%i\n", v[i]);

	return 0;
}
```

## Ejercicio 4

Escribe un programa que pida al usuario 10 números enteros y calcule el valor medio de todos ellos, mostrando luego en la pantalla los números que están por encima de la media y los que están por debajo de ella.

```
int main(void)
{
   int v[10], suma, i;
   float media;

   // Introducir los datos
   printf("Introduce 10 números enteros: \n");
   for (i = 0; i < 10; i++)
         scanf("%i", &v[i]);

   // Calcular la media
   suma = 0;
   for (i = 0; i < 10; i++)
         suma = suma + v[i];

   media = (float)suma / 10;
   printf("Media = %.2f\n", media);

   // Mostrar números por encima de la media
   printf("Los números por encima de la media son:\n");
   for (i = 0; i < 10; i++)
   {
         if (v[i] > media)
               printf("%i\n", v[i]);
   }

   // Mostrar números por debajo de la media
   printf("Los números por debajo de la media son:\n");
   for (i = 0; i < 10; i++)
   {
         if (v[i] < media)
               printf("%i\n", v[i]);
   }

   return 0;
}
```

## Ejercicio 5

Escribe un programa que pida al usuario 10 números reales positivos o negativos, y realice, por un lado, la suma de todos los positivos y, por otro, la suma de todos los negativos, mostrando al final el resultado

```
int main(void)
{
   float v[10];
   float suma_neg, suma_pos;
   int  i;

   printf("Introduce 10 números:");
   for (i=0; i<=9; i++)
     scanf("%f", &v[i]);

   suma_neg = 0 ;
   suma_pos = 0 ;
   for  (i=0; i<=9;  i++)
   {
     if (v[i] < 0)
                suma_neg = suma_neg + v[i];
         if  (v[i]  > 0)
                suma_pos = suma_pos + v[i];
   }

   printf("La suma de los negativos es: %.2f\n", suma_neg);

   printf("La suma de los positivos es: %.2f\n", suma_pos);

   system("PAUSE");
   return 0;
}
```

## Ejercicio 8 (con la burbuja)

Escribe un programa que defina un vector de 1000 números enteros e inicialízalo con números aleatorios entre 0 y 50.000. Una vez generado el vector, aplícale algún método de ordenación de los vistos en el ejercicio anterior.

Después de que el vector haya sido ordenado, el usuario debe introducir un número entre 0 y 50.000, y el programa debe buscar ese número en el vector, comunicando al usuario la posición que ocupa (si lo encuentra) o un mensaje de error (si no lo encuentra).

**Notas sobre esta solución:**

- Observa que, en esta solución, la longitud del vector se ha declarado en una constante llamada LONGITUD VECTOR. Así, puede usarse el mismo programa con varios tamaños de vector: basta con cambiar el valor de esa constante y volver a compilarlo.
- Observa también cómo se usa la función estándar clock() para medir el tiempo que tarda el algoritmo en ejecutarse. La función clock() devuelve el tiempo de CPU que el programa ha consumido desde que empezó a ejecutarse. Ese tiempo viene expresado en "clocks" o pasos de reloj, no en segundos, ni milisegundos, ni nada parecido. Para convertirlo a milésimas de segundo, hay que dividirlo entre la constante CLOCKS_PER_SEC (ver código).
- Se han implementado los dos métodos de búsqueda (secuencial y binaria), contando en cada caso el número de pasos necesarios para localizar un elemento. Revisa bien las implementaciones y asegúrate de entenderlas, en particular la de la búsqueda binaria, bastante más difícil que la secuencial.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define LONGITUD_VECTOR 50000
```

```
void genera_vector(int v[LONGITUD_VECTOR]);
void ordena_vector(int v[LONGITUD_VECTOR]);
void buscar_secuencial(int v[LONGITUD_VECTOR], int busc);
void buscar_binario(int v[LONGITUD_VECTOR], int busc);


int main(void)
{
  int v[LONGITUD_VECTOR], busc;
  clock_t tiempo;

  printf("\nORDENACIÓN DE UN VECTOR DE %i ELEMENTOS\n", LONGITUD_VECTOR);
  printf("MÉTODO DE INTERCAMBIO DIRECTO (BURBUJA)\n");

  // Se genera el vector aleatoriamente
  printf("Generando el vector...\n");
  genera_vector(v);

  // Se ordena el vector con el método de la burbuja
  printf("Ordenando el vector...\n");
  ordena_vector(v);

  // Se calcula y muestra el tiempo de ordenación
  tiempo = clock();
  tiempo = tiempo * 1000 / CLOCKS_PER_SEC;
  printf("Vector ordenado\nTiempo empleado: %i,%i segundos\n\n", tiempo/1000, tiempo%1000/100);

  // Se busca un dato por los métodos secuencial y binario
  printf("¿Qué número quieres buscar?");
  scanf("%i", &busc);

  buscar_secuencial(v, busc);
  buscar_binario(v, busc);

  system("pause");
  return 0;
}

void genera_vector(int v[LONGITUD_VECTOR])
{
  int i;
  for (i = 0; i < LONGITUD_VECTOR; i++)
     v[i] = rand()%50000;
}

void ordena_vector(int v[LONGITUD_VECTOR])
{
  int i, j, elem;
  for (i = 1; i < LONGITUD_VECTOR; i++)
  {
     for (j = LONGITUD_VECTOR - 1; j >=i; j--)
     {
        if (v[j-1] > v[j])
             {
                elem = v[j- 1];
                v[j- 1]  = v[j];
                v[j]   = elem;
             }
     }
  }
}

void buscar_secuencial(int v[LONGITUD_VECTOR], int busc)
{
   int i, encontrado;
```

```
  // Iniciamos una búsqueda secuencial
  encontrado = 0;
  i = 0;
  while ((i < LONGITUD_VECTOR) && (encontrado == 0))
  {
        if (v[i] == busc) // Lo hemos encontrado !!
              encontrado = 1;
   else
      i++;
  }

  if (encontrado == 1)
        printf("El dato %i ha sido encontrado en la posición %i\n", busc, i);
  else
        printf("El dato %i no está en el vector\n", busc);

  printf("Se han necesitado %i pasos con la búsqueda secuencial\n\n", i);
}


void buscar_binario(int v[LONGITUD_VECTOR], int busc)
{
  int izq, der, mitad, encontrado, pasos;

  // Iniciamos una búsqueda binaria
  encontrado = 0;
  pasos = 0;
  izq = 0;
  der = LONGITUD_VECTOR - 1;
  while ((izq < der-1) && (encontrado == 0))
  {
   printf("Paso %i, iz=%i, de=%i\n", pasos, izq, der);
        mitad = izq + ((der - izq) / 2);
        if (v[mitad] == busc)          // Lo hemos encontrado !!
              encontrado = 1;

     if (v[mitad] > busc)           // Seguimos buscando en la mitad izquierda
              der = mitad;
          if (v[mitad] < busc)       // Seguimos buscando en la mitad derecha
              izq = mitad;

        pasos++;
  }

  if (encontrado == 1)
        printf("El dato %i ha sido encontrado en la posición %i\n", busc, mitad);
  else
        printf("El dato %i no está en el vector\n", busc);

  printf("Se han necesitado %i pasos con la búsqueda binaria\n\n", pasos);
}
```

## Ejercicio 10

Escribe un programa que genere al azar un vector de 10 enteros *ordenado de menor a mayor*, y que lo muestre por la pantalla. Después, pedirá al usuario un número entre 0 y el número más grande que haya en el vector, y lo insertará en la posición adecuada del vector para no romper el orden, desplazando el resto de elementos a la derecha. El más grande se perderá. Por último, el vector se volverá a mostrar en la pantalla.

```
int main(void)
{
        int i, n, posicion_insercion, v[10];
        srand(time(NULL));    /* Inicialización de los números aleatorios */
```

```
	/* Para generar el vector ordenado, asignaremos un número de 0 a 9 a la primera posición,
	de 10 a 19 a la segunda, de 20 a 29 a la tercera, y así sucesivamente. Hay otras
	formas de generar vectores aleatorios ordenados */

	for (i = 0; i < 10; i++)
	{
		v[i] = rand()%10 + (i*10);
	}
	imprimir_vector(v);

	/* Pedimos al usuario que teclee un número entre 0 y v[9] */
	do {
		printf("Introduzca un número entre 0 y %i: ", v[9]);
		scanf("%i", &n);
	}
	while ((n < 0) || (n > v[9]));

	/* Buscamos la posición en la que hay que insertar el número n */
	i = 0;
	while (v[i] < n)
		i++;

	posicion_insercion = i;

	/* Desplazamos los números del vector desde la posición de inserción hasta el final */
	for (i = 9; i > posicion_insercion; i--)
		v[i] = v[i-1];

	/* Insertamos el número en la posición de inserción */
	v[posicion_insercion] = n;

	/* Mostramos el vector resultante */
	imprimir_vector(v);

	return 0;
}

void imprimir_vector(int v[10])
{
	int i;

	for (i=0; i<10; i++)
		printf("%i ", v[i]);
}
```

## Ejercicio 11

Escribe un programa capaz de sumar dos números enteros de gran tamaño. Los podemos llamar enteros gigantes. Por ejemplo, dos números enteros hasta 1000 dígitos cada uno.

```
int main()
{
   char gigante1[76], gigante2[76], gigante3[76];
   int n1, n2, n3, me_llevo, i1, i2, i3;

   /* Pedimos los datos de entrada. Los números gigantes se guardarán
      en cadenas de caracteres de 75 dígitos como máximo (se puede
      cambiar para aceptar tamaños mayores) */
   puts("\n\nENTEROS GIGANTES\n\n");
   puts("Teclee el primer número gigante: ");
   gets(gigante1);
   puts("Teclee el segundo número gigante: ");
   gets(gigante2);

   /* Establecemos los valores iniciales de los índices de cada cadena
      para recorrerlas hacia atrás. La cadena resultado (gigante3) medirá
```

```
    tanto como la más larga más uno, por si hay un acarreo al final de la suma */
  i1 = strlen(gigante1) - 1;
  i2 = strlen(gigante2) - 1;
  if (i1 > i2)
        i3  = i1  + 1;
  else
        i3  = i2  + 1;


  gigante3[i3+1] = '\0';    /* Colocamos el nulo en la cadena resultado */
  me_llevo = 0;           /* Acarreo */

  /* Comienza el bucle de la suma. Iremos sumando dígito a dígito, desde el
     último hacia atrás, construyendo el resultado en otra cadena (gigante3) */
  while ((i1 >= 0) || (i2 >= 0))
  {
        if  (i1  >= 0)
              n1 = gigante1[i1]  -  48;    /*  Convertimos  primer  sumando a número */
        else
              n1 = 0;
        if  (i2  >= 0)
              n2 = gigante2[i2]  -  48;    /*  Convertimos  segundo  sumando a número */
        else
              n2 = 0;

        n3 = (n1  + n2 + me_llevo)  % 10;    /*  Calculamos el  resultado  */
        gigante3[i3]  = n3 + 48;             /*  Lo almacenamos en gigante3 */
        me_llevo  = (n1  + n2 + me_llevo)  /  10;    /*  Acarreo  para  la  próxima suma */
        i1--;                                /*  Actualizamos los  índices  */
        i2--;
        i3--;
  }
  if (me_llevo == 0)              /* Colocamos el último acarreo (si lo hay) */
        gigante3[i3]  = '  ';
  else
        gigante3[i3]  = me_llevo  + 48;

  /* Mostramos el resultado (adaptado para números de máximo 75 dígitos) */
  printf("\n\n");
  printf("%+75s  +\n", gigante1);
  printf("%+75s\n", gigante2);
  printf("---------------------------------------------------------------------------\n");
  printf("%+75s\n\n\n", gigante3);

  return 0;
}
```

## Ejercicio 14

Escribe un programa en el que se defina una matriz de 10x10 números enteros. Inicializa todos los elementos al valor –1.

```
int main(void)

{
        int  m[10][10];
        int  fila,  columna;

        //  Inicializamos la  matriz
        for  (fila  = 0;  fila  < 10;  fila++)
        {
              for  (columna = 0;  columna < 10;  columna++)
              {
                    m[columna][fila]  = -1;
              }
```

```
	}

	// Mostramos la matriz en la pantalla
	for (fila = 0; fila < 10; fila++)
	{
		for (columna = 0; columna < 10; columna++)
		{
			printf("%i ", m[columna][fila]);
		}
		printf("\n");
	}

	return 0;
}
```

## Ejercicio 15

Repite el ejercicio anterior, inicializando ahora todas las filas pares al valor 0 y todas las filas impares al valor –1.

```
int main(void)
{
	int m[10][10];
	int fila, columna;

	// Inicializamos la matriz
	for (fila = 0; fila < 10; fila++)
	{
		for (columna = 0; columna < 10; columna++)
		{
			if (fila % 2 == 0)
				m[columna][fila] = 0;
			else
				m[columna][fila] = -1;
		}
	}

	// Mostramos la matriz en la pantalla
	for (fila = 0; fila < 10; fila++)
	{
		for (columna = 0; columna < 10; columna++)
		{
			printf("%i ", m[columna][fila]);
		}
		printf("\n");
	}

	return 0;
}
```

## Ejercicio 16

Escribe un programa que defina una matriz de 3x3 números enteros y luego pida al usuario que introduzca los valores de cada elemento. Después, el programa debe sumar los tres números de cada fila y de cada columna, mostrando los resultados.

```
int main(void)
{
	int m[3][3];
	int fila, columna, suma;

	// Inicializamos la matriz
	for (fila = 0; fila < 3; fila++)
	{
		for (columna = 0; columna < 3; columna++)
```

```
		{
			printf("Introduce el valor de la columna %i, fila %i: ", columna, fila);
			scanf("%i", &m[columna][fila]);
		}
	}

	// Mostramos la matriz en la pantalla
	for (fila = 0; fila < 3; fila++)
	{
		for (columna = 0; columna < 3; columna++)
		{
			printf("%3i ", m[columna][fila]);
		}
		printf("\n");
	}

	// Mostramos la suma de las filas
	for (fila = 0; fila < 3; fila++)
	{
		suma = 0;
		for (columna = 0; columna < 3; columna++)
		{
			suma = suma + m[columna][fila];
		}
		printf("La suma de la fila %i es: %i\n", fila, suma);
	}

	// Mostramos la suma de las columnas
	for (columna = 0; columna < 3; columna++)
	{
		suma = 0;
		for (fila = 0; fila < 3; fila++)
		{
			suma = suma + m[columna][fila];
		}
		printf("La suma de la columna %i es: %i\n", columna, suma);
	}

	return 0;
}
```

## Ejercicio 17

Escribe un programa que defina una matriz de 3x3 números enteros y luego pida al usuario que introduzca los valores de cada elemento. Después, debe permutar el contenido de la fila 3 por el de la fila 1, y mostrar por último el contenido de la matriz.

```
int main(void)
{
	int m[3][3];
	int fila, columna, aux;

	// Inicializamos la matriz
	for (fila = 0; fila < 3; fila++)
	{
		for (columna = 0; columna < 3; columna++)
		{
			printf("Introduce el valor de la columna %i, fila %i: ", columna, fila);
			scanf("%i", &m[columna][fila]);
		}
	}

	// Mostramos la matriz en la pantalla
	for (fila = 0; fila < 3; fila++)
	{
		for (columna = 0; columna < 3; columna++)
		{
```

```
                printf("%3i ", m[columna][fila]);
        }
        printf("\n");
    }

    // Permutamos la fila 3 y la fila 1 (es decir, los índices 2 y 0)
    for (columna = 0; columna < 3; columna++)
    {
        aux = m[columna][0];
        m[columna][0] = m[columna][2];
        m[columna][2] = aux;
    }

    // Volvemos a mostrar la matriz para ver cómo ha quedado
    for (fila = 0; fila < 3; fila++)
    {
        for (columna = 0; columna < 3; columna++)
        {
            printf("%3i ", m[columna][fila]);
        }
        printf("\n");
    }

    return 0;
}
```

## Ejercicio 18

Escribe un programa que defina dos matrices de 10x5 números enteros y las inicialice con números aleatorios entre 0 y 255. Posteriormente, cada elemento de la primera matriz debe ser sumado con el mismo elemento de la segunda matriz, guardando el resultado en una tercera matriz. Se deben sumar todas las parejas de elementos y mostrar el resultado en la pantalla.

```
// Prototipo de la función para imprimir matrices en la pantalla
void imprimir_matriz(int m[10][5])

int main(void)
{
    int m1[10][5], m2[10][5], msuma[10][5];
    int fila, columna;

    srand(time(NULL));    // Inicialización del generador de números aleatorios

    // Inicializamos las matrices m1 y m2
    for (fila = 0; fila < 5; fila++)
    {
        for (columna = 0; columna < 10; columna++)
        {
            m1[columna][fila] = rand()%256;
            m2[columna][fila] = rand()%256;
        }
    }

    // Calculamos la matriz suma (msuma)
    for (fila = 0; fila < 5; fila++)
    {
        for (columna = 0; columna < 10; columna++)
        {
            msuma[columna][fila] = m1[columna][fila] + m2[columna][fila];
        }
    }

    // Mostramos las tres matrices en la pantalla
    printf("MATRIZ 1:\n--------\n");
    imprimir_matriz(m1);
```

```
        printf("MATRIZ 2:\n---------\n");
        imprimir_matriz(m2);

        printf("MATRIZ SUMA:\n---------\n");
        imprimir_matriz(msuma);

        return 0;
}


void imprimir_matriz(int m[10][5])
{
        int fila, columna;

        for (fila = 0; fila < 5; fila++)
        {
                for (columna = 0; columna < 10; columna++)
                {
                        printf("%3i ", m[columna][fila]);
                }
                printf("\n");
        }
}
```

## Ejercicio 19

Escribe un programa que utilice una matriz de 5x5 enteros para:

1. Pedir por teclado el valor de todos sus elementos.
2. Imprimir por pantalla la diagonal principal.
3. Calcular la media de la triangular superior.

```
int main(void)
{
        int m[5][5];
        int fila, columna;
        int suma, cont, columna_inicial;
        float media;

        // Inicializamos la matriz
        for (fila = 0; fila < 5; fila++)
        {
                for (columna = 0; columna < 5; columna++)
                {
                        printf("Introduce el valor de la columna %i, fila %i: ", columna, fila);
                        scanf("%i", &m[columna][fila]);
                }
        }

        // Mostramos la matriz en la pantalla
        for (fila = 0; fila < 5; fila++)
        {
                for (columna = 0; columna < 5; columna++)
                {
                        printf("%3i ", m[columna][fila]);
                }
                printf("\n");
        }

        // Imprimimos la diagonal principal
        printf("Diagonal principal: ");
        fila = 0; columna = 0;
        while (fila < 5)
```

```
        {
                printf("%i ", m[columna][fila]);
                fila++;
                columna++;
        }
        printf("\n");

        // Calculamos la suma de la triangular superior
        suma = 0;
        cont = 0;
        for (fila = 0; fila < 5; fila++)
        {
                columna_inicial = fila;
                for (columna = columna_inicial; columna < 5; columna++)
                {
                        suma = suma + m[columna][fila];
                        cont++;
                }
        }

        // Calculamos la media de la triangular superior
        media = (float)suma / cont;
        printf("La media de la triangular superior es: %.2f ", media);
        printf("(suma = %i, cuenta = %i)\n", suma, cont);

        return 0;
}
```

## Ejercicio 20

Escribe un programa que genere al azar una matriz cuadrada de NxN números enteros (siendo N una constante definida al principio del programa con un valor cualquiera) y que luego muestre un pequeño menú de opciones al usuario con estas opciones:

- Opción 1: mostrar matriz. Esta función sacará el contenido de la matriz por la pantalla de manera que se puedan visualizar claramente sus elementos.
- Opción 2: perímetro. Mostrará los elementos que ocupan el borde de la matriz, partiendo de la esquina superior izquierda y recorriéndola hacia la derecha y luego hacia abajo.
- Opción 3: espiral. Hará un recorrido en espiral por la matriz partiendo de la esquina superior izquierda.
- Opción 4: salir. Hace que el programa finalice.

```
/* Tamaño de la matriz cuadrada */
#define N 4

/* Prototipos de funciones */
void mostrar_matriz(int m[N][N]);
void mostrar_perimetro(int m[N][N]);
void mostrar_espiral(int m[N][N]);

int main(void)
{
        int opc, f, c;
        int m[N][N];

        /* Rellenamos la matriz con números al azar */
        for (f = 0; f < N; f++)
                for (c = 0; c < N; c++)
                        m[c][f] = rand() % 10;

        /* Menú de opciones */
        do
        {
                printf("1. Mostrar matriz\n");
                printf("2. Perímetro\n");
                printf("3. Espiral\n");
```

```
                printf("4. Salir\n");

                printf("Introduzca opción (1-4): ");
                scanf("%i", &opc);

                switch(opc)
                {
                        case 1: mostrar_matriz(m); break;
                        case 2: mostrar_perimetro(m); break;
                        case 3: mostrar_espiral(m); break;
                }
        }
        while (opc != 4);

        return 0;
}


/* Esta función muestra por la pantalla la matriz pasada como parámetro */
void mostrar_matriz(int m[N][N])
{
        int f, c;

        for (f = 0; f < N; f++)
        {
                for (c = 0; c < N; c++)
                {
                        printf("%i ", m[c][f]);
                }
                printf("\n");
        }

}


/* Esta función muestra el perímetro de la matriz pasada como parámetro */
void mostrar_perimetro(int m[N][N])
{
        int f, c;

        printf("El perímetro es:\n");

        /* Mostramos fila superior (de izquierda a derecha) */
        for (c = 0; c < N; c++)
                printf("%i ", m[c][0]);
        /* Mostramos columna derecha (de arriba a abajo) */
        for (f = 1; f < N; f++)
                printf("%i ", m[N-1][f]);
        /* Mostramos fila inferior (de derecha a izquierda) */
        for (c = N-2; c >= 0; c--)
                printf("%i ", m[c][N-1]);
        /* Mostramos columna izquierda (de abajo a arriba) */
        for (f = N-2; f >= 1; f--)
                printf("%i ", m[0][f]);

        printf("\n");
}


/* Esta función muestra la espiral de la matriz pasada como parámetro.
   Para lograrlo, hacemos un perímetro de la matriz completa y luego
   vamos acotando los límites para hacer un perímetro de la matriz
   interna. */
void mostrar_espiral(int m[N][N])
{
        int f, c, cont, izq, der, arr, aba;
```

```
        printf("La espiral es:\n");

        izq = 0;/* Límites del perímetro */
        der = N;
        arr = 0;
        aba = N;

        for (cont = 0; cont <= N/2; cont++)
        {
                /* Mostramos fila superior (de izquierda a derecha) */
                for (c = izq; c < der; c++)
                        printf("%i ", m[c][arr]);
                /* Mostramos columna derecha (de arriba a abajo) */
                for (f = arr + 1; f < aba; f++)
                        printf("%i ", m[der-1][f]);
                /* Mostramos fila inferior (de derecha a izquierda) */
                for (c = der-2; c >= izq; c--)
                        printf("%i ", m[c][aba-1]);
                /* Mostramos columna izquierda (de abajo a arriba) */
                for (f = aba-2; f >= arr+1; f--)
                        printf("%i ", m[izq][f]);

                izq++;          /* Actualizamos límites del perímetro */
                der--;
                arr++;
                aba--;
        }

        printf("\n");
}
```