Carrera: Electrónica y automatización
Tema: Ejercicios propuestos capítulo 4
Nombre: Dylan Tutillo
NRC: 29583

Asignatura: Fundamentos de programación
Docente: Jenny Ruiz
Fecha: 06/01/2026

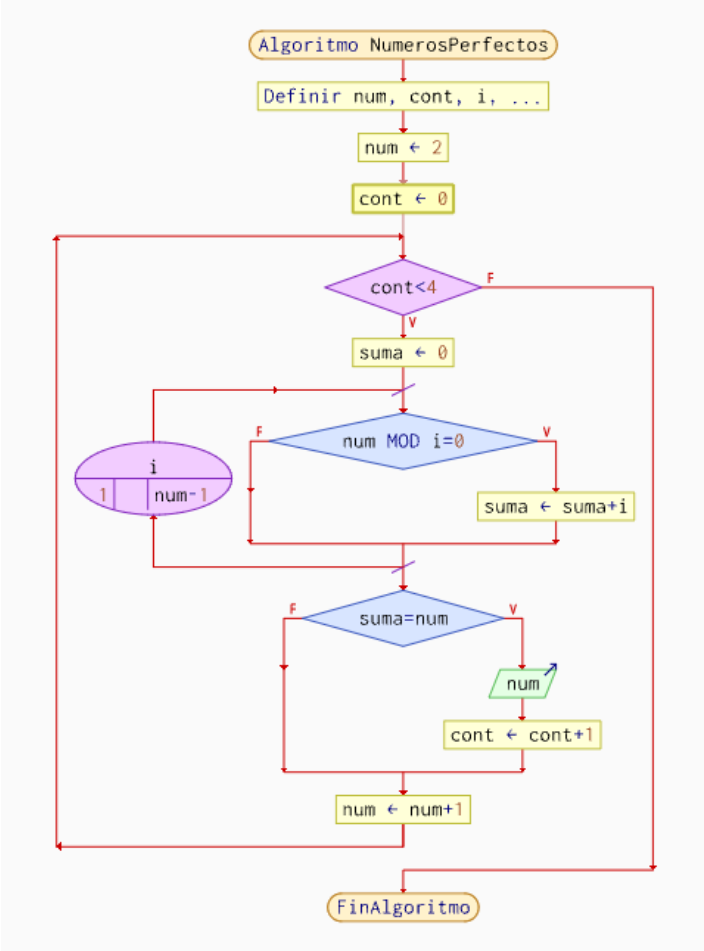
Problema 3.1.1: Número perfecto

Un número perfecto es un número natural que es igual a la suma de sus divisores propios positivos, sin incluirse a sí mismo. Por ejemplo, el número 6 es un número perfecto porque sus divisores propios son 1, 2 y 3; y 6= 1+2+3. El siguiente número perfecto es 28=1+2+4+7+14

Se pide:

1. Programar la función esperfecto, la cual recibe como argumento un número natural y devuelve el valor 1 si el número es perfecto, y 0 en caso contrario.
2. Desarrollar la función principal del programa que calcule y muestre por pantalla los cuatro primeros números perfectos.

Objeto	Tipo	Valor inicial
num	Entero	2
cont	Entero	0
i	Entero	1
suma	Entero	0




```
#include <stdio.h>

int main() {
   int num = 2;
   int cont = 0;
   int i;
   int suma;

   while (cont < 4) {
      suma = 0;

      for (i = 1; i <= num - 1; i++) {
         if (num % i == 0) {
            suma += i;
         }
      }

      if (suma == num) {
         printf("%d\n", num);
         cont++;
```

```
    }

    num++;
  }

  return 0;
}
```

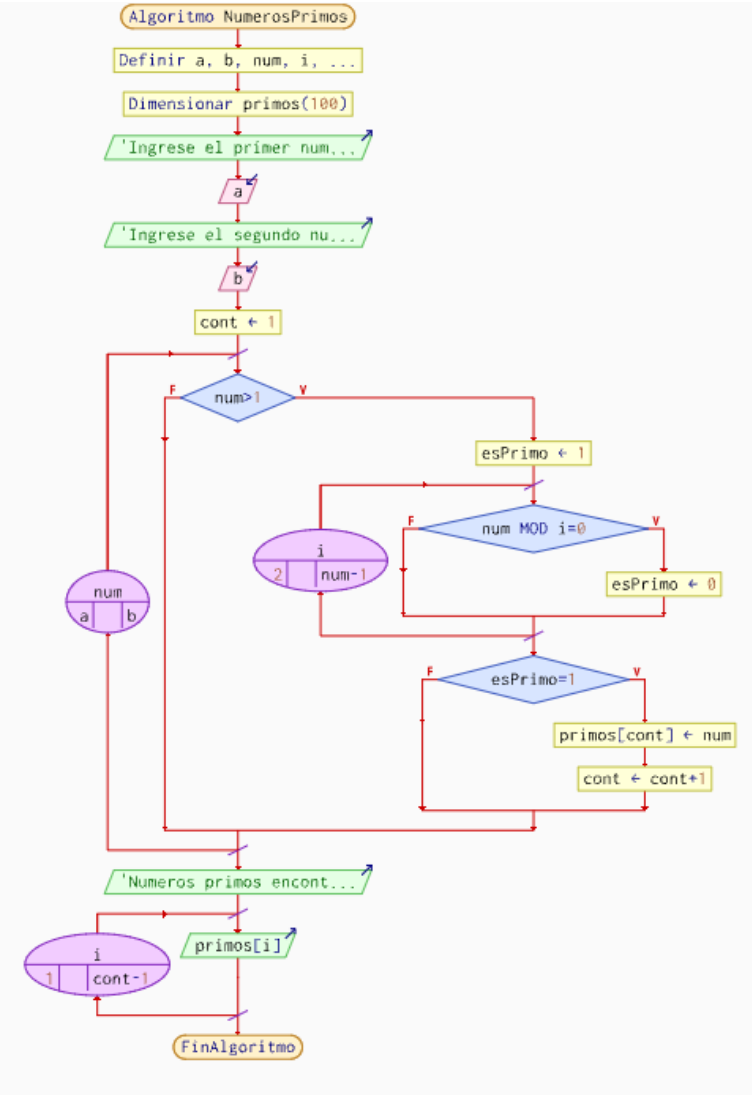
```
6
28
496
8128

Process returned 0 (0x0)   execution time : 0.086 s
Press any key to continue.
```

Problema 3.1.2: Números primos

Realice un programa que resuelva adecuadamente los siguientes apartados:

1. Programar la función esprimo, que recibe como argumento un número entero y devuelve el valor 1 si el número es primo y 0 en caso contrario.
2. Almacenar en un vector todos los números primos comprendidos entre dos números introducidos por teclado y luego imprimir dicho vector.

Objeto	Tipo	Valor inicial
a	Entero	Teclado
b	Entero	Teclado
num	Entero	a
i	Entero	2
cont	Entero	1
esPrimo	Entero	1
primos[]	Vector	Vacío (tamaño 100)




```
#include <stdio.h>

int main() {
   int a, b, num, i;
   int primos[100];
   int cont = 0;
   int esPrimo;

   printf("Ingrese el primer numero: ");
   scanf("%d", &a);

   printf("Ingrese el segundo numero: ");
   scanf("%d", &b);

   for (num = a; num <= b; num++) {
      if (num > 1) {
         esPrimo = 1;
```

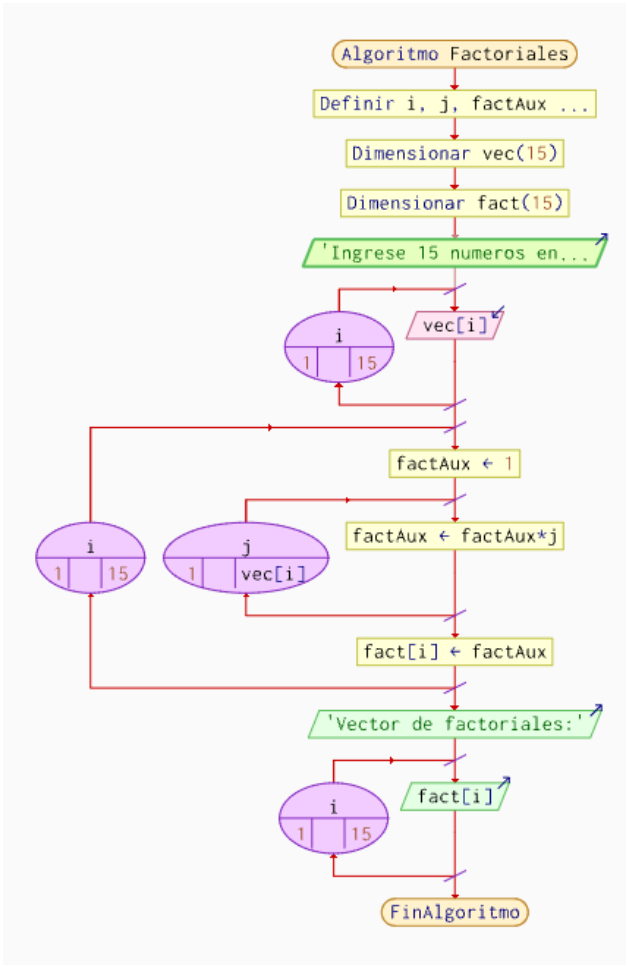
```
        for (i = 2; i <= num - 1; i++) {
            if (num % i == 0) {
                esPrimo = 0;
            }
        }

        if (esPrimo == 1) {
            primos[cont] = num;
            cont++;
        }
    }
  }

  printf("Numeros primos encontrados:\n");
  for (i = 0; i < cont; i++) {
    printf("%d\n", primos[i]);
  }

  return 0;
}
```

```
Ingrese el primer numero: 2
Ingrese el segundo numero: 20
Numeros primos encontrados:
2
3
5
7
11
13
17
19

Process returned 0 (0x0)   execution time : 4.282 s
Press any key to continue.
```

Problema 3.1.3: Factorial

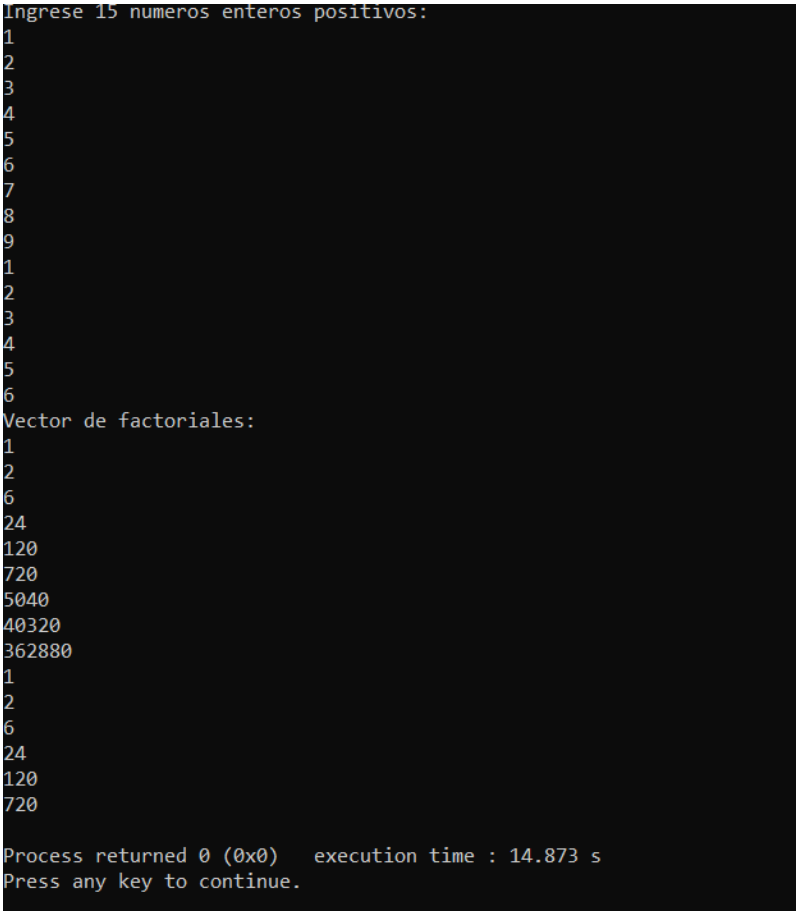
Programar la función calc_fact, que recibe como argumento un número entero y devuelve el valor de su factorial, usar dicha función en un programa que, dado un vector vec de 15 números enteros, calcule un vector fact con las factoriales correspondientes y los muestre por pantalla.

Objeto	Tipo	Valor inicial
i	Entero	1
j	Entero	1
Vec	Vector	Teclado (15 enteros)
Fact	Vector	Vacío
factAux	Entero	1




```
#include <stdio.h>

int main() {
   int vec[15];
   int fact[15];
   int i, j;
   int factAux;

   printf("Ingrese 15 numeros enteros positivos:\n");
   for (i = 0; i < 15; i++) {
      scanf("%d", &vec[i]);
   }

   for (i = 0; i < 15; i++) {
      factAux = 1;

      for (j = 1; j <= vec[i]; j++) {
         factAux *= j;
      }

      fact[i] = factAux;
```

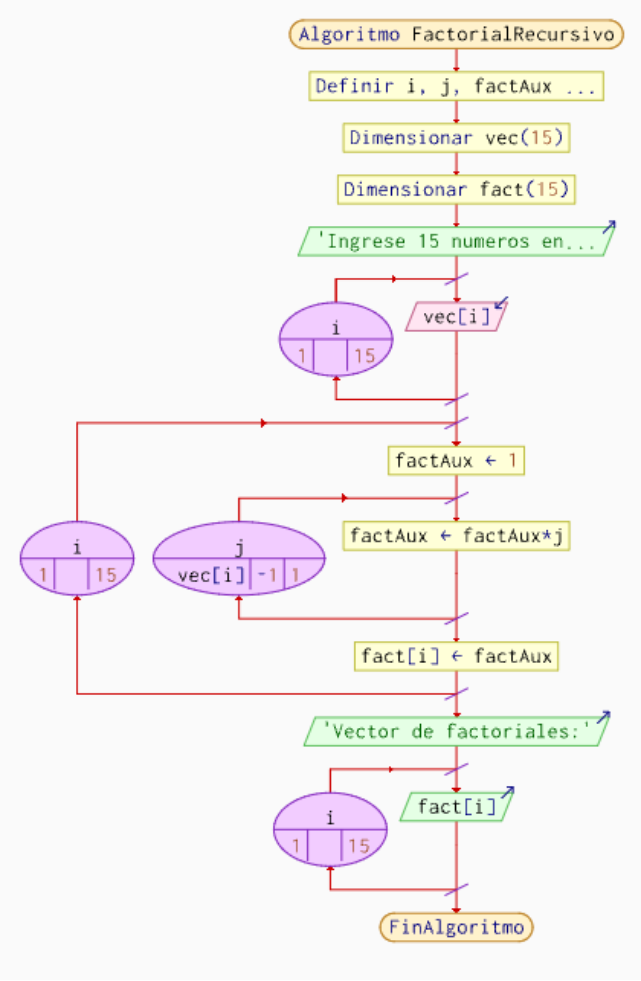
```
    }

    printf("Vector de factoriales:\n");
    for (i = 0; i < 15; i++) {
        printf("%d\n", fact[i]);
    }
    return 0;
}
```



Problema 3.1.4: Factorial recursivo

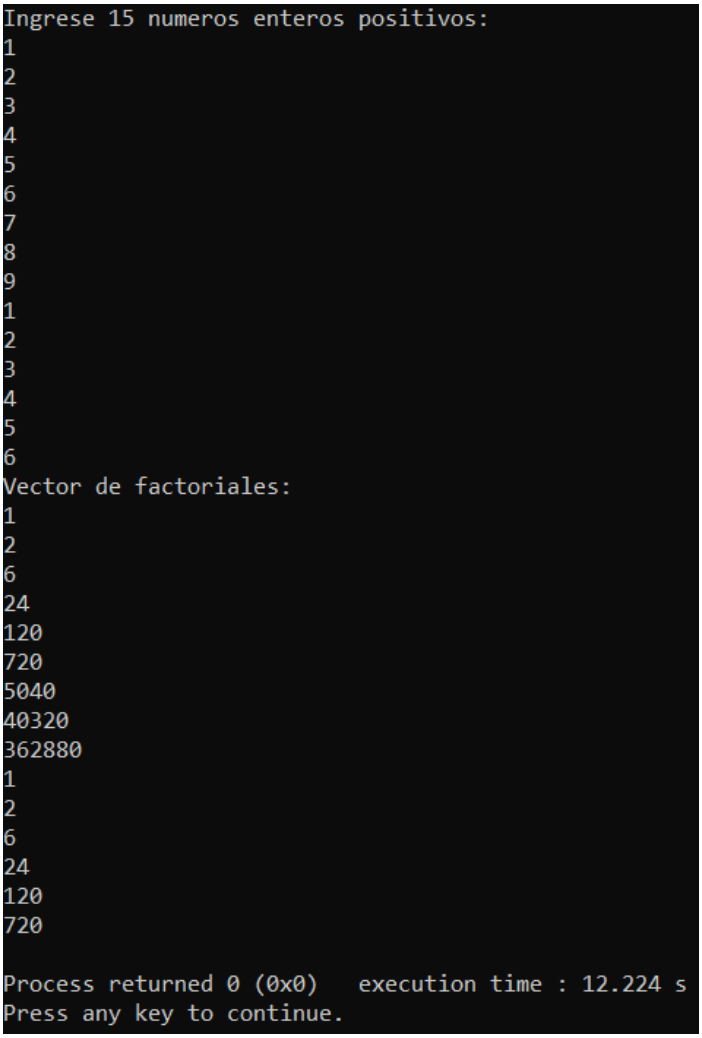
Programar la función calc_fact, que recibe como argumento un número entero y devuelve el valor de su factorial calculado de manera recursiva, usar dicha función en un programa que, dado un vector vec de 15 números enteros, calcule un vector fact con sus factoriales y los muestre por pantalla.

Objeto	Tipo	Valor inicial
i	Entero	1
j	Entero	—
vec	Vector	Teclado (15 enteros)
fact	Vector	Vacío
factAux	Entero	1
calc_fact	Función	Devuelve entero




```
#include <stdio.h>

int calc_fact(int n) {
   if (n == 0 || n == 1)
      return 1;
   else
      return n * calc_fact(n - 1);
}

int main() {
   int vec[15];
   int fact[15];
   int i;

   printf("Ingrese 15 numeros enteros positivos:\n");
   for (i = 0; i < 15; i++) {
      scanf("%d", &vec[i]);
   }

   for (i = 0; i < 15; i++) {
      fact[i] = calc_fact(vec[i]);
   }
```

```
    printf("Vector de factoriales:\n");
    for (i = 0; i < 15; i++) {
        printf("%d\n", fact[i]);
    }

    return 0;
}
```



Recomendaciones:

- Es importante saber y comprender correctamente el uso de las estructuras en C ya que permiten organizar de muchas formas la información, de forma clara y facilita el uso de fatos como matrices, puntos y mensajes.

- Finalmente, se recomienda practicar este tipo de ejercicios para reforzar el uso de estructuras, arreglos y funciones en el lenguaje C, ya que son fundamentales en el desarrollo de programas más avanzados.

Conclusiones:

- Para concluir, el uso de estructuras en lenguaje C permite organizar de manera ordenada la información de varias personas dentro de un programa.
- En conclusión, los arreglos de estructuras facilitan el almacenamiento y manejo de múltiples datos de forma eficiente.
- Mediante el desarrollo de cada ejercicio propuesto de logro aplicar el uso de estructuras en C, permitiéndonos de alguna forma almacenar y manipular información de forma organizada y eficiente.

- Concluimos que el aprendizaje viene de la manipulación entre otras palabras jugando se aprende a programar, las estructuras son muy complejas, pero la práctica logro que entendiéramos el funcionamiento de estas.