ACTIVIDAD

1. DATOS INFORMATIVOS

Carrera: Electrónica y automatización

Asignatura: Fundamentos de programación

Tema: Ejercicios

Docente: Jenny Ruiz

Integrantes: Omar Alquinga

Fecha: 02/01/2026 **Paralelo:** 29583

2. DESARROLLO

Desarrollar los siguientes ejercicios con todo lo viste desde la primera unidad:

- **PROBLEMA 3.1.2 NÚMEROS PRIMOS.**

Realice un programa que resuelva adecuadamente los siguientes apartados:

1. Programe la función es primo, que recibe como argumento un número entero, y devuelve un
valor 1 si el número es primo y 0 en caso contrario.

2. Almacene en un vector todos los números primos comprendidos entre dos nú meros introducidos por teclado y luego imprima dicho vector.

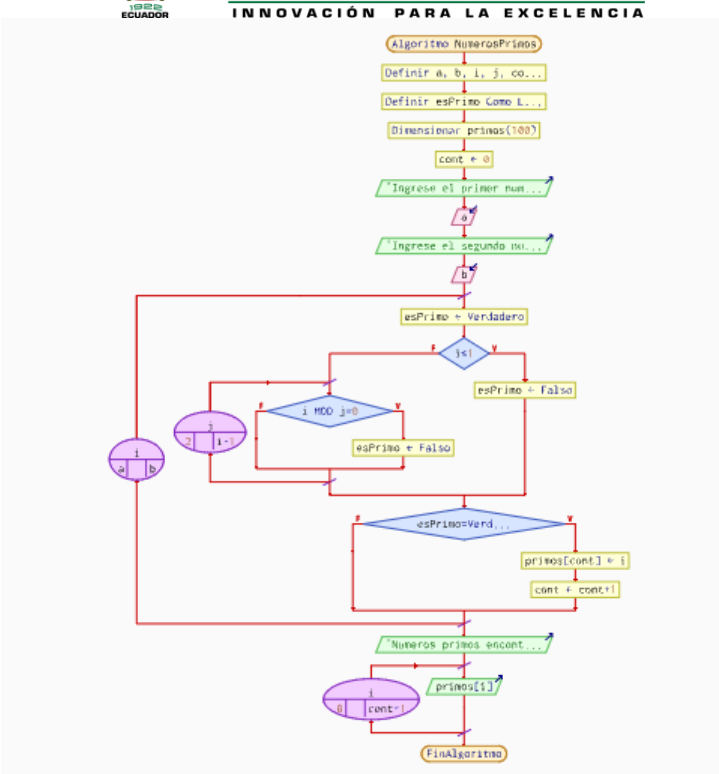
- **Tabla de datos**

Objeto	Nombre	Tipo	Descripción
Número inicial del rango	a	var	entero
Número final del rango	b	var	entero
Contador del recorrido del rango	i	var	entero
Contador para verificar divisiones	j	var	entero
Contador de números primos	cont	var	entero
Indicador de número primo	esPrimo	var	lógico
Vector de números primos	primos	var	entero
Límite máximo del vector	100	cte	entero

- **Pseudocódigo**

```
Algoritmo NumerosPrimos
   Definir a, b, i, j, cont Como Entero
   Definir esPrimo Como Logico
   Dimension primos[100]
   cont <- 0

   Escribir "Ingrese el primer numero:"
   Leer a
   Escribir "Ingrese el segundo numero:"
   Leer b

   Para i <- a Hasta b Hacer
      esPrimo <- Verdadero

      Si i <= 1 Entonces
         esPrimo <- Falso
      Sino
         Para j <- 2 Hasta i - 1 Hacer
            Si i MOD j = 0 Entonces
               esPrimo <- Falso
            FinSi
         FinPara
      FinSi

      Si esPrimo = Verdadero Entonces
         primos[cont] <- i
         cont <- cont + 1
      FinSi
   FinPara

   Escribir "Numeros primos encontrados:"
   Para i <- 0 Hasta cont - 1 Hacer
      Escribir primos[i]
   FinPara
FinAlgoritmo
```

- **Diagrama de Flujo**

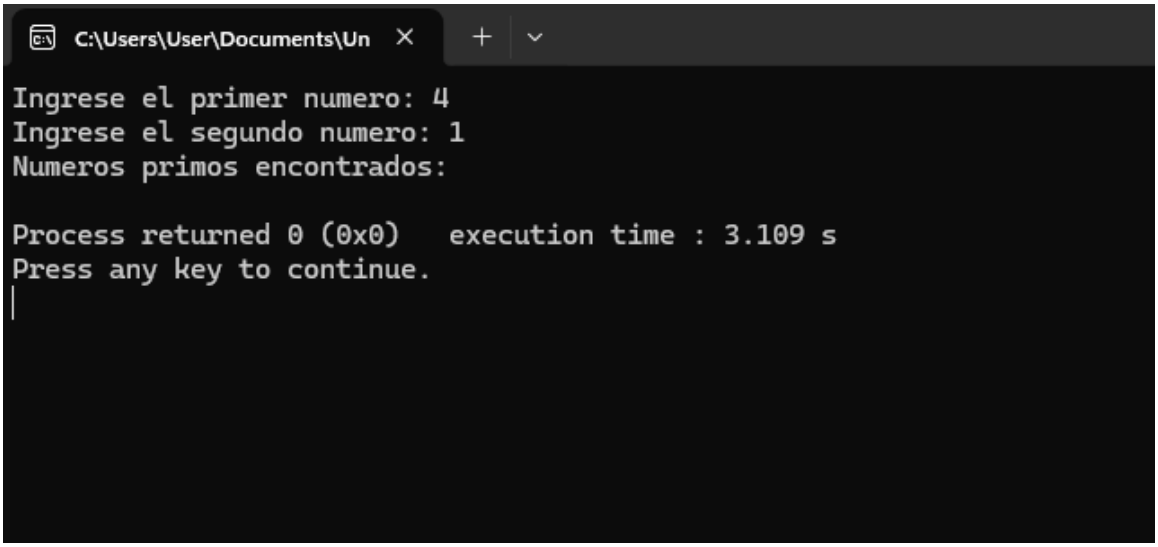


- **Código del programa:**

```
#include <stdio.h>

int main() {
    int a, b, i, j;
    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 (i = a; i <= b; i++) {
        esPrimo = 1;

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

```
            }
        }

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

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

}
```

- **Captura del programa:**



- **PROBLEMA 3.1.3 FACTORIAL.**

Programe la función calc_fact, que recibe como argumento un número entero, y devuelva el valor del factorial. A continuación, use esta función en un programa que, dado un vector de 15
números enteros vec, calcule un vector fact con sus factoriales y lo muestre por pantalla.

- **Tabla de objetos**

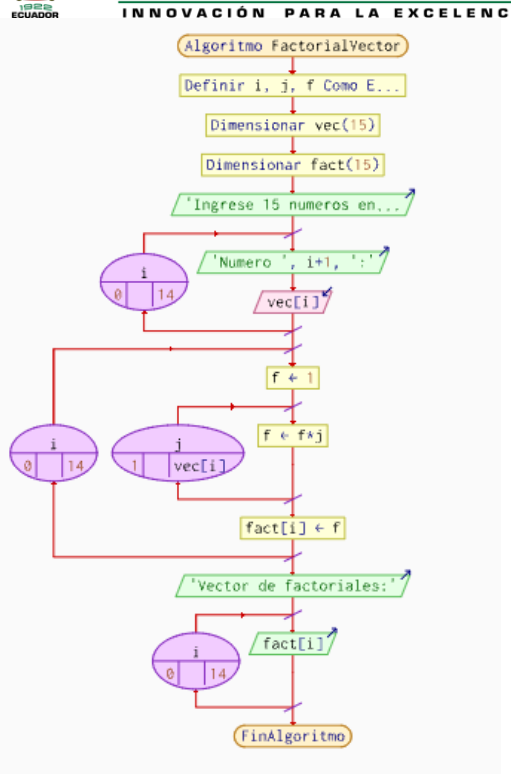
Objeto	Nombre	Tipo	Descripción
Vector de números de entrada	vec	var	entero
Vector de factoriales	fact	var	entero
Contador del vector	i	var	entero

Contador para el cálculo del factorial	j	var	entero
Variable acumuladora del factorial	f	var	entero
Tamaño del vector	15	cte	entero
Número uno	1	cte	entero

- **Pseudocódigo**

```
Algoritmo FactorialVector
   Definir i, j, f Como Entero
   Dimension vec[15]
   Dimension fact[15]

   Escribir "Ingrese 15 numeros enteros:"

   Para i <- 0 Hasta 14 Hacer
      Escribir "Numero ", i + 1, ":"
      Leer vec[i]
   FinPara

   Para i <- 0 Hasta 14 Hacer
      f <- 1

      Para j <- 1 Hasta vec[i] Hacer
         f <- f * j
      FinPara

      fact[i] <- f
   FinPara

   Escribir "Vector de factoriales:"
   Para i <- 0 Hasta 14 Hacer
      Escribir fact[i]
   FinPara
FinAlgoritmo
```

- **Diagrama de flujo**

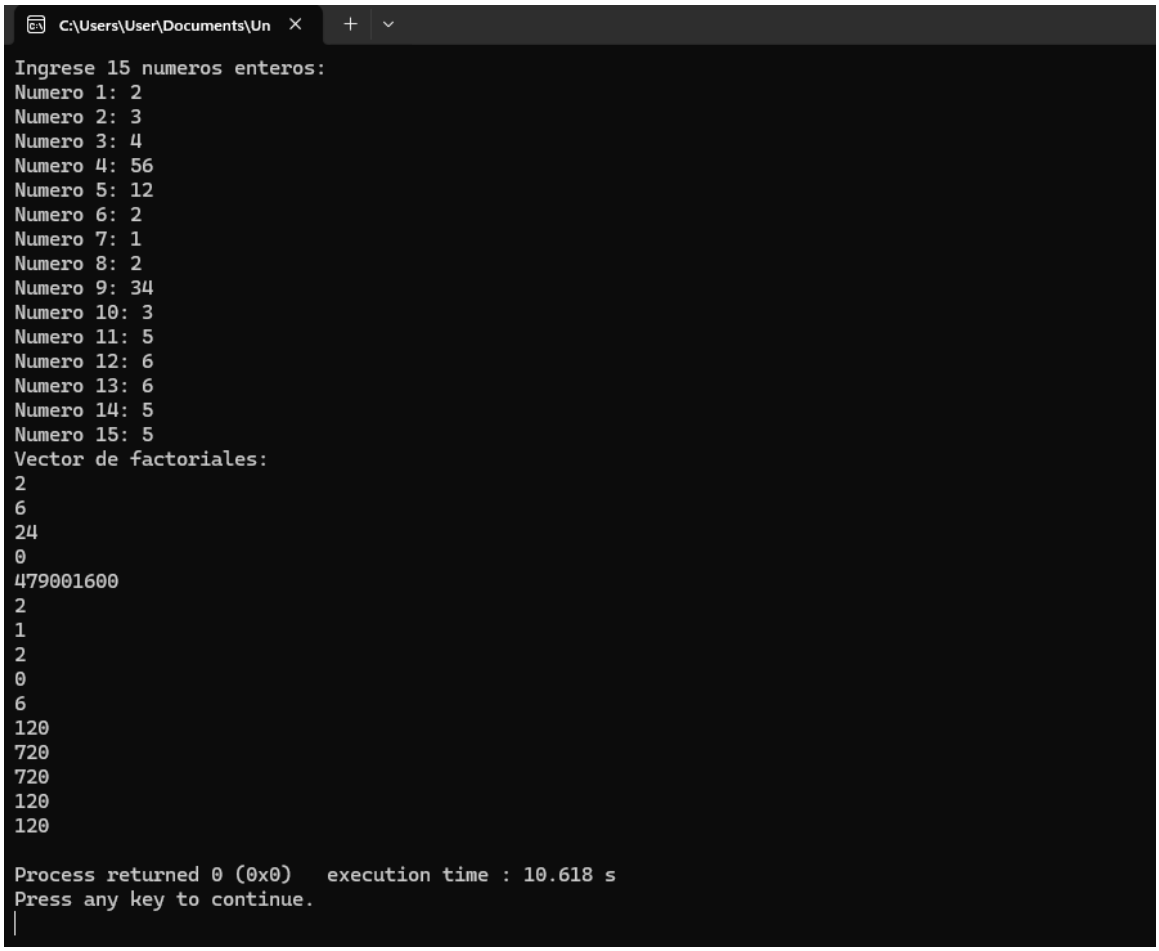


- **Código del programa:**

```
#include <stdio.h>

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

    printf("Ingrese 15 numeros enteros:\n");
    for (i = 0; i < 15; i++) {
        printf("Numero %d: ", i + 1);
        scanf("%d", &vec[i]);
    }

    for (i = 0; i < 15; i++) {
        f = 1;
        for (j = 1; j <= vec[i]; j++) {
            f = f * j;
        }
        fact[i] = f;
    }

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

```
    }

}
```

- **Captura del programa funcionando:**



- **PROBLEMA 3.1.4 FACTORIAL RECURSIVO.**

Programe la función calc_fact, que recibe como argumento un número entero, y devuelva el valor del factorial realizando el cálculo de manera recursiva. A continuación use esta función en un programa que, dado un vector de 15 números enteros vec, calcule un vector fact con sus
factoriales y lo muestre por pantalla.

- **Tabla de objetos:**

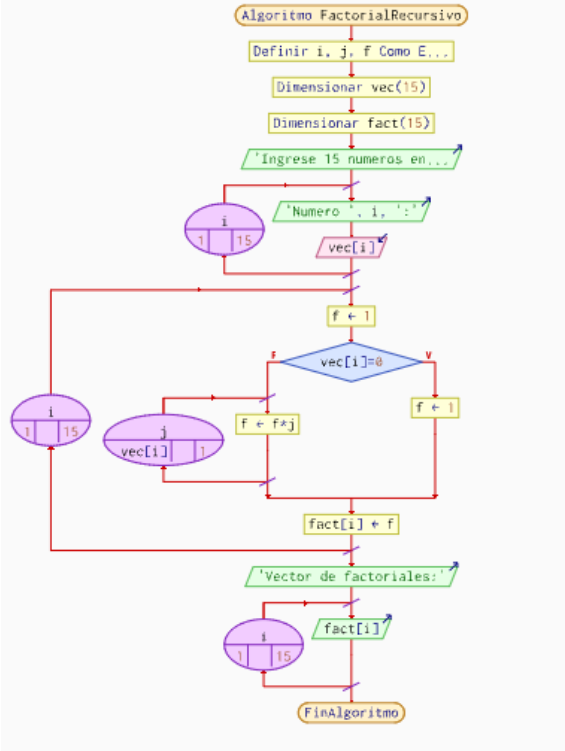
Objeto	Nombre	Tipo	Descripción
Vector de números de entrada	vec	var	entero
Vector de factoriales	fact	var	entero
Contador del vector	i	var	entero

Contador auxiliar	j	var	entero
Acumulador del factorial	f	var	entero
Tamaño del vector	15	cte	entero
Número uno	1	cte	entero
Número cero	0	cte	entero

- **Pseudocódigo:**

```
Algoritmo FactorialRecursivo
   Definir i, j, f Como Entero
   Dimension vec[15]
   Dimension fact[15]

   Escribir "Ingrese 15 numeros enteros:"

   Para i <- 1 Hasta 15 Hacer
      Escribir "Numero ", i, ":"
      Leer vec[i]
   FinPara

   Para i <- 1 Hasta 15 Hacer
      f <- 1

      Si vec[i] = 0 Entonces
         f <- 1
      Sino
         Para j <- vec[i] Hasta 1 Hacer
            f <- f * j
         FinPara
      FinSi

      fact[i] <- f
   FinPara

   Escribir "Vector de factoriales:"
   Para i <- 1 Hasta 15 Hacer
      Escribir fact[i]
   FinPara
FinAlgoritmo
```

- **Diagrama de flujo:**

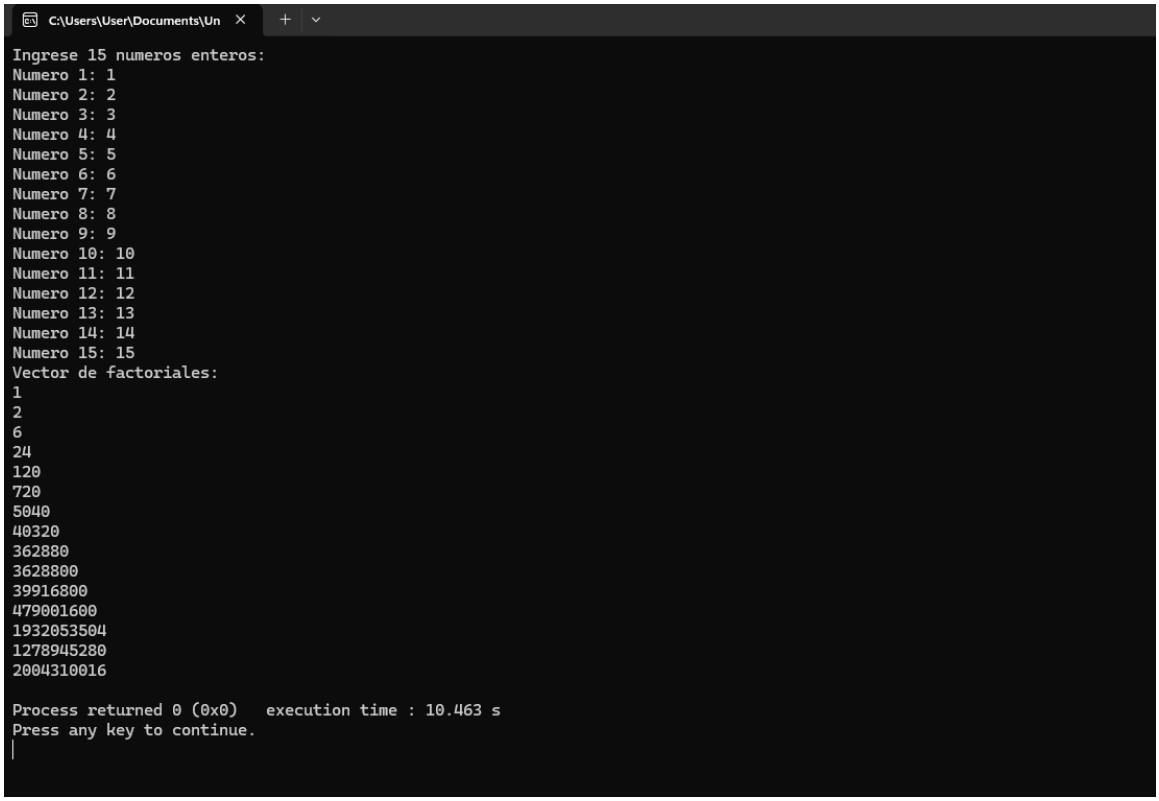


- **Código del programa:**

```
#include <stdio.h>

int calc_fact(int n) {
    if (n == 0 || n == 1) {
        return 1;              // caso base
    } else {
        return n * calc_fact(n - 1);   // llamada recursiva
    }
}

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

    printf("Ingrese 15 numeros enteros:\n");
    for (i = 0; i < 15; i++) {
        printf("Numero %d: ", i + 1);
        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]);
    }

}
```

- **Captura de pantalla del programa:**



- **PROBLEMA 4.1: DATOS POBLACIÓN.**

Se desea hacer un programa que almacene datos sobre un determinado colectivo de personas, solicitando el nombre, la edad y la ciudad de residencia. Realice los siguientes apartados:
1. Diseñe una estructura de datos para almacenar la información relativa a cada individuo. Añada un campo adicional que
almacene el número de veces que se repite el nombre en el colectivo.

2. Realice el programa principal que declare una tabla de estructuras de dimensión 10 para almacenar la información sobre personas (se ha supuesto que el número de personas no será mayor de 10). A continuación debe pedir por teclado el número de personas para introducir y después los datos de cada una de ellas. Tras ello, calcule el nombre que se repita más en los datos introducidos y la media de edad de todas las personas.

- **Tabla de objetos:**

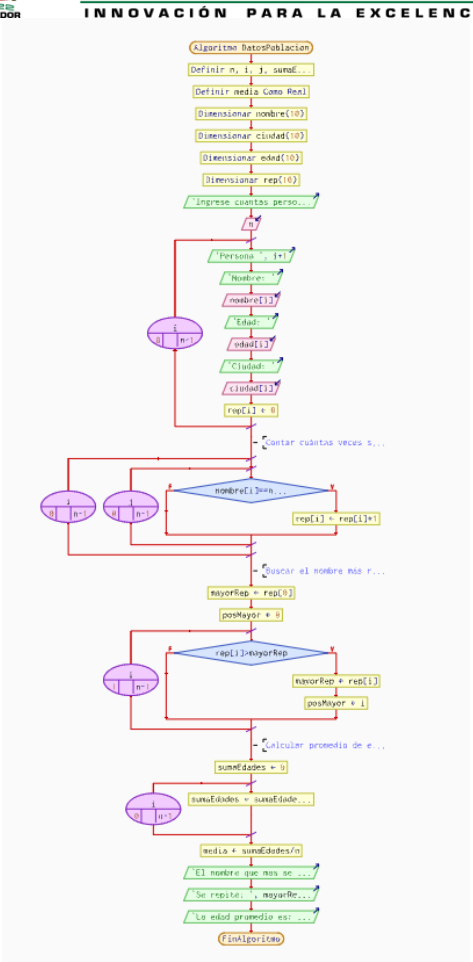
Objeto	Nombre	Valor	Tipo
Número de personas	n	var	entero
Contador	i	var	entero
Contador	j	var	entero
Suma de edades	sumaEdades	var	entero
Mayor repetición	mayorRep	var	entero
Posición mayor	posMayor	var	entero
Promedio	media	var	real
Personas	personas	var	estructura
Nombre	nombre	var	cadena
Edad	edad	var	entero
Ciudad	ciudad	var	cadena
Repeticiones	rep	var	entero
Límite personas	10	cte	entero

- **Pseudocódigo del programa:**

```
Proceso DatosPoblacion

   Definir n, i, j, sumaEdades, mayorRep, posMayor Como Entero
   Definir media Como Real

   Dimension nombre[10]
   Dimension ciudad[10]
   Dimension edad[10]
   Dimension rep[10]

   Escribir "Ingrese cuantas personas va a registrar (max 10): "
   Leer n

   Para i = 0 Hasta n-1 Hacer
      Escribir "Persona ", i+1

      Escribir "Nombre: "
      Leer nombre[i]

      Escribir "Edad: "
      Leer edad[i]
```

```
    Escribir "Ciudad: "
    Leer ciudad[i]

    rep[i] = 0
  FinPara

  Para i = 0 Hasta n-1 Hacer
    Para j = 0 Hasta n-1 Hacer
      Si nombre[i] == nombre[j] Entonces
        rep[i] = rep[i] + 1
      FinSi
    FinPara
  FinPara

  mayorRep = rep[0]
  posMayor = 0

  Para i = 1 Hasta n-1 Hacer
    Si rep[i] > mayorRep Entonces
      mayorRep = rep[i]
      posMayor = i
    FinSi
  FinPara

  sumaEdades = 0
  Para i = 0 Hasta n-1 Hacer
    sumaEdades = sumaEdades + edad[i]
  FinPara

  media = sumaEdades / n

  Escribir "El nombre que mas se repite es: ", nombre[posMayor]
  Escribir "Se repite: ", mayorRep, " veces"
  Escribir "La edad promedio es: ", media

FinProceso
```

- **Diagrama de flujo:**



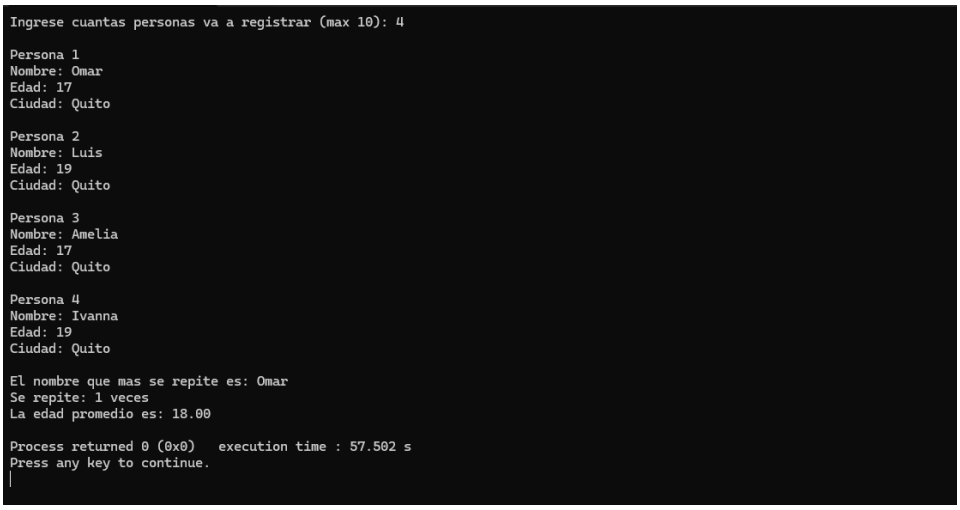
- **Código del programa:**

```
#include <stdio.h>
#include <string.h>

struct persona
{
   char nombre[100];
   int edad;
   char ciudad[100];
   int rep;
};

typedef struct persona per;

int main()
{
   per personas[10];
   int n, i, j;
   int sumaEdades = 0;
```

```
int mayorRep = 0;
int posMayor = 0;
float media;

printf("Ingrese cuantas personas va a registrar (max 10): ");
scanf("%d", &n);

for(i = 0; i < n; i++)
{
   printf("\nPersona %d\n", i + 1);

   printf("Nombre: ");
   scanf("%s", personas[i].nombre);

   printf("Edad: ");
   scanf("%d", &personas[i].edad);

   printf("Ciudad: ");
   scanf("%s", personas[i].ciudad);

   personas[i].rep = 0;
}

for(i = 0; i < n; i++)
{
   for(j = 0; j < n; j++)
   {
      if(strncmp(personas[i].nombre, personas[j].nombre, 100) == 0)
      {
         personas[i].rep++;
      }
   }
}

mayorRep = personas[0].rep;
posMayor = 0;

for(i = 1; i < n; i++)
{
   if(personas[i].rep > mayorRep)
   {
      mayorRep = personas[i].rep;
      posMayor = i;
   }
}
```

```
    for(i = 0; i < n; i++)
    {
        sumaEdades += personas[i].edad;
    }

    media = (float)sumaEdades / n;

    printf("\nEl nombre que mas se repite es: %s", personas[posMayor].nombre);
    printf("\nSe repite: %d veces", mayorRep);
    printf("\nLa edad promedio es: %.2f\n", media);

}
```

- **Captura del programa funcionando:**



- **PROBLEMA 4.2 RECONOCIMIENTO DE CARACTERES.**

Se pretende escribir un programa para reconocer caracteres a partir de un mapa de puntos. El mapa de puntos describe la forma de un carácter como una matriz de unos y ceros de 8×8 celdas. Se dispone además de una tabla de estructuras de tipo struct letras, que puede suponer convenientemente creada e inicializada, y que contiene la descripción de las 27 letras del alfabeto.

- **Tabla de objetos:**

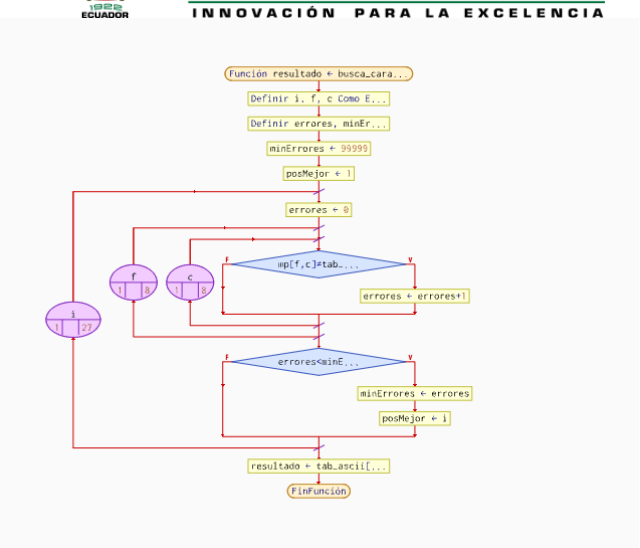
Objeto	nombre	valor	tipo
Matriz	**mp**	**var**	**entero**
Tabla de letras	**tab_let**	**var**	**struct letras**
Código ASCII	**cod_ASCII**	**var**	**carácter**
Matriz de puntos	**mptos**	**var**	**entero**
Máximo de coincidencias	**max_coincidencias**	**var**	**entero**

Coincidencias actuales	**coincidencias_actuales**	**var**	**entero**
Total de letras	**27**	**cte**	**entero**
Dimensiones matriz	**8**	**cte**	**entero**

- **Pseudocódigo del programa:**

```
Algoritmo ReconocimientoCaracteres
   Dimension tab_let_mptos[27, 8, 8]
   Dimension tab_let_ascii[27]
   Dimension mp[8, 8]

   Escribir "El caracter detectado es: ", busca_caracter(mp, tab_let_mptos, tab_let_ascii)
FinAlgoritmo

Funcion retorno <- busca_caracter(mp, mptos, letras_ascii)
   Definir max_coincidencias, indice_mejor, coincidencias_actuales Como Entero
   max_coincidencias <- -1
   indice_mejor <- 0

   Para k <- 1 Hasta 27 Con Paso 1 Hacer
      coincidencias_actuales <- 0
      Para i <- 1 Hasta 8 Con Paso 1 Hacer
         Para j <- 1 Hasta 8 Con Paso 1 Hacer
            Si mp[i,j] == mptos[k,i,j] Entonces
               coincidencias_actuales <- coincidencias_actuales + 1
            FinSi
         FinPara
      FinPara

      Si coincidencias_actuales > max_coincidencias Entonces
         max_coincidencias <- coincidencias_actuales
         indice_mejor <- k
      FinSi
   FinPara

   retorno <- letras_ascii[indice_mejor]
FinFuncion
```

- **Diagrama de flujo:**

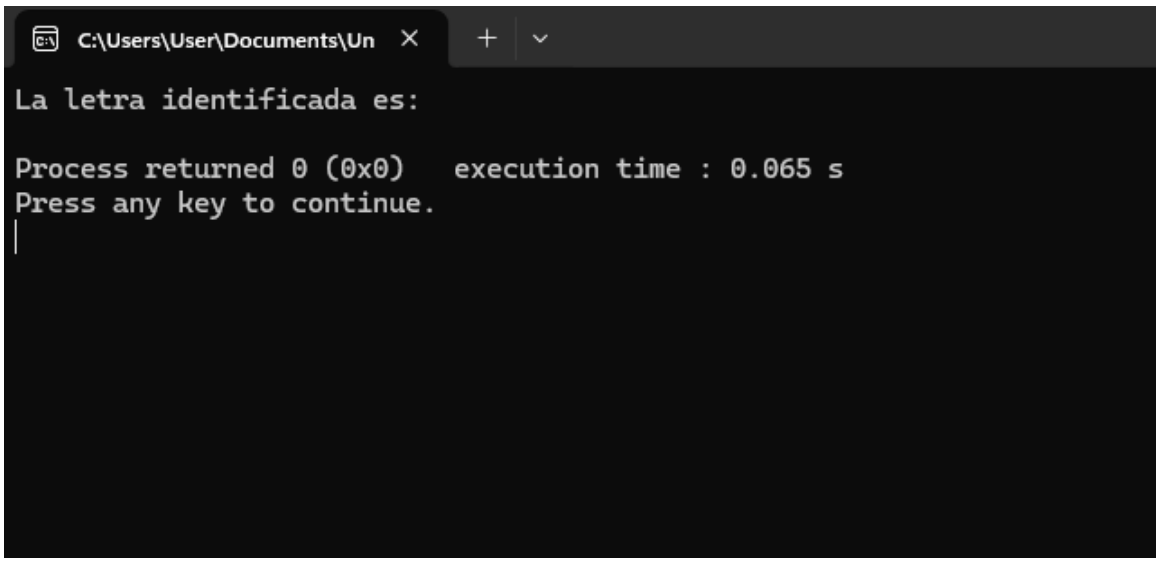


- **Código del programa:**

```
#include <stdio.h>

struct letras {
    char cod_ASCII;
    int mptos[8][8];
};

char busca_caracter(int mp[8][8], struct letras tablet[27]) {
    int max_coincidencias = -1;
    char mejor_letra = '?';

    for (int k = 0; k < 27; k++) {
        int coincidencias_actuales = 0;

        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                if (mp[i][j] == tablet[k].mptos[i][j]) {
                    coincidencias_actuales++;
                }
            }
        }

        if (coincidencias_actuales > max_coincidencias) {
            max_coincidencias = coincidencias_actuales;
            mejor_letra = tablet[k].cod_ASCII;
        }
    }
```

```
    return mejor_letra;
}

int main() {
    struct letras tab_let[27];
    int mi_matriz[8][8] = {0};

    char resultado = busca_caracter(mi_matriz, tab_let);
    printf("La letra identificada es: %c\n", resultado);

}
```

- **Captura del programa funcionando:**



- **Conclusiones:**

1. Estos ejercicios sirven para aprender a manejar varios datos juntos usando estructuras y arreglos, en vez de usar muchas variables separadas.
2. Con los ciclos y las comparaciones se puede analizar la información por ejemplo para saber qué nombre se repite más o qué letra se parece más a otra.
3. Hacer primero la tabla de objetos y el pseudocódigo ayuda bastante a que el programa salga más ordenado y no tenga tantos errores.

- **Recomendaciones:**

1. Leer bien el enunciado antes de empezar a programar para saber exactamente qué pide el ejercicio.
2. Probar el programa varias veces para ver si da el resultado correcto.
3. Guardar el trabajo seguido para no perder lo que ya está hecho.