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 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.

Se ha introducido la definición de un nuevo tipo con la sentencia typedef.

```
struct persona
{
char nombre[100];
int edad;
char ciudad[100];
int rep; // Indica las veces que está repetido un nombre.
};

typedef struct persona per; // Declaración de nuevo tipo
```

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 a introducir y después los datos de cada una de ellas. Tras ello, calcule el nombre que se repite más en los datos introducidos y la media de edad de todas las personas.

Objeto	Tipo	Valor inicial
p	Vector de estructuras per	Tamaño 10
nombre	Cadena	Teclado
edad	Entero	Teclado
ciudad	Cadena	Teclado
rep	Entero	0
n	Entero	Teclado
i	Entero	0
j	Entero	0
sumaEdad	Entero	0
maxRep	Entero	0
nombreMax	Cadena	Vacío

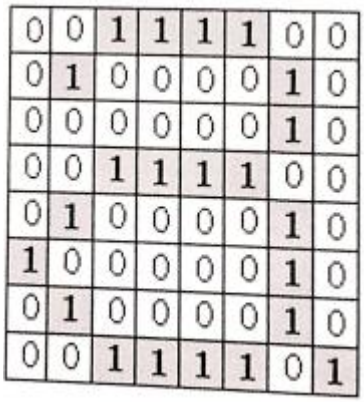
```
#include <stdio.h>
#include <string.h>

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

int main() {
   per p[10];
   int n, i, j;
   int sumaEdad = 0;
   int maxRep = 0;
   char nombreMax[100];

   printf("Ingrese el numero de personas: ");
   scanf("%d", &n);


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

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

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

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

        p[i].rep = 0;
        sumaEdad += p[i].edad;
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
          if (strcmp(p[i].nombre, p[j].nombre) == 0) {
             p[i].rep++;
            }
        }
    }
    for (i = 0; i < n; i++) {
        if (p[i].rep > maxRep) {
            maxRep = p[i].rep;
            strcpy(nombreMax, p[i].nombre);
        }
    }
    printf("\nNombre mas repetido: %s\n", nombreMax);
    printf("Media de edad: %.2f\n", (float)sumaEdad / n);

    return 0;
}
```

```
Ingrese el numero de personas: 3

Persona 1
Nombre: Dylan
Edad: 12
Ciudad: Quito

Persona 2
Nombre: Dylan
Edad: 19
Ciudad: Cuanca

Persona 3
Nombre: Jose
Edad: 17
Ciudad: Quito

Nombre mas repetido: Dylan
Media de edad: 16.00

Process returned 0 (0x0)   execution time : 58.015 s
Press any key to continue.
```

Problema 4.2 Reconocimiento de caracteres

Se pretende escribir un programa para reconocer 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 8x8 celdas. De dispone además de una tabla de estructuras del tipo struct letras, que puede suponer convenientemente creada e inicializada, y que contiene la descripción de las 27 letras del alfabeto, tal como se describe en el ejemplo.



Escriba la función **busca_caracter**, la cual recibe una matriz de 8 × 8 enteros que contiene el mapa de puntos de la letra que se quiere identificar (parámetro *mp*) y la tabla de estructuras de tipo letras que contiene la descripción de las 27 letras del alfabeto (*tab_let*). La función debe devolver el código ASCII del carácter que más se parezca.

Fila	Valores
0	0 0 1 1 1 1 0 0
1	0 1 0 0 0 0 1 0
2	0 0 0 0 0 0 1 0
3	0 0 1 1 1 1 0 0
4	0 1 0 0 0 0 1 0
5	1 0 0 0 0 0 1 0
6	0 1 0 0 0 0 1 0
7	0 0 1 1 1 1 0 1

```
#include <stdio.h>

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

char busca_caracter(int mapa[8][8], letras tab_let[]) {
   int f, c;

   for (f = 0; f < 8; f++) {
      for (c = 0; c < 8; c++) {
         if (mapa[f][c] != tab_let[0].mptos[f][c]) {
            return '?';
         }
```

```
        }
    }
    return tab_let[0].cod_ASCII;
}

int main() {
    letras tab_let[1] = {
        {'a',
        {
            {0,0,1,1,1,1,0,0},
            {0,1,0,0,0,0,1,0},
            {0,0,0,0,0,0,1,0},
            {0,0,1,1,1,1,0,0},
            {0,1,0,0,0,0,1,0},
            {1,0,0,0,0,0,1,0},
            {0,1,0,0,0,0,1,0},
            {0,0,1,1,1,1,0,1}
        }}
    };

    int mapa[8][8] = {
        {0,0,1,1,1,1,0,0},
        {0,1,0,0,0,0,1,0},
        {0,0,0,0,0,0,1,0},
        {0,0,1,1,1,1,0,0},
        {0,1,0,0,0,0,1,0},
        {1,0,0,0,0,0,1,0},
        {0,1,0,0,0,0,1,0},
        {0,0,1,1,1,1,0,1}
    };

    char c = busca_caracter(mapa, tab_let);

    printf("Caracter reconocido: %c\n", c);
    printf("Codigo ASCII: %d\n", c);

    return 0;
}
```

```
Caracter reconocido: a
Codigo ASCII: 97

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Problema 4.3 Polígono.
Un polígono es una figura geométrica cerrada delimitada por segmentos rectos (aristas). En este ejercicio se usa la estructura tipo polígono para almacenar la información del polígono, donde en vez de almacenarse sus aristas se almacenan sus vértices; De este modo, el polígono de la figura 4.2 quedará representado mediante una estructura de tipo polígono (p), que contiene el número de vértices del polígono (5), y otros tantos de estructuras que contienen las coordenadas de los vértices con dimensión 100.

Codigo:

```
#include <stdio.h>

struct punto {
   float x;
   float y;
};

struct poligono {
   int nvert;
   struct punto vert[100];
};

int main() {
   struct poligono p;
   int i;

   printf("Ingrese el numero de vertices del poligono: ");
   scanf("%d", &p.nvert);

   for (i = 0; i < p.nvert; i++) {
      printf("Vertice %d\n", i + 1);
      printf("x: ");
      scanf("%f", &p.vert[i].x);
      printf("y: ");
      scanf("%f", &p.vert[i].y);
   }
 printf("\nVertices del poligono:\n");
   for (i = 0; i < p.nvert; i++) {
      printf("(%f , %f)\n", p.vert[i].x, p.vert[i].y);
   }

   return 0;
}
```

```
Ingrese el numero de vertices del poligono: 5
Vertice 1
x: 2
y: 3
Vertice 2
x: 4
y: 1
Vertice 3
x: 7
y: 6
Vertice 4
x: 82
y: 34
Vertice 5
x: 4
y: 2

Vertices del poligono:
(2.000000 , 3.000000)
(4.000000 , 1.000000)
(7.000000 , 6.000000)
(82.000000 , 34.000000)
(4.000000 , 2.000000)

Process returned 0 (0x0)   execution time : 29.475 s
Press any key to continue.
```

Problema 4.4 Game Over.
Se pretende hacer un pequeño juego de palabras por ordenador en lenguaje C que muestre en pantalla los caracteres que aparecen en la figura 4.3 de fin de juego. Considere la siguiente estructura:

```
#define N 12
struct mensaje
{
   char game_over[N];
   char se_acabo[N];
};
typedef struct mensaje men;
```

Codigo:

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

#define N 12

struct mensaje {
   char game_over[N];
   char se_acabo[N];
};

typedef struct mensaje men;

int main() {
   men m;
   int i;

   strcpy(m.game_over, "GAME OVER");
   strcpy(m.se_acabo, "INSERT COIN");

   /* Mostrar GAME OVER progresivamente */
   for (i = 1; i <= strlen(m.game_over); i++) {
      printf("%.*s\n", i, m.game_over);
   }

   /* Mostrar INSERT COIN */
   printf("%s\n", m.se_acabo);

   return 0;
}
```

```
G
GA
GAM
GAME
GAME
GAME O
GAME OV
GAME OVE
GAME OVER
INSERT COIN

Process returned 0 (0x0)   execution time : 0.024 s
Press any key to continue.
```

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.

- Es recomendable leer y entender el problema a resolver sin importar cuanto tiempo se demore.

- Se sugiere que jugando a cambiar algo en los códigos podemos llegar a descubrir nuevas funciones, y si se cometen errores aprender de ellos.

- 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:

- 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.