UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Nombre: Richard Casa **NRC:** 29583
Fecha: 05/01/2026
Tema: Estructuras

1.

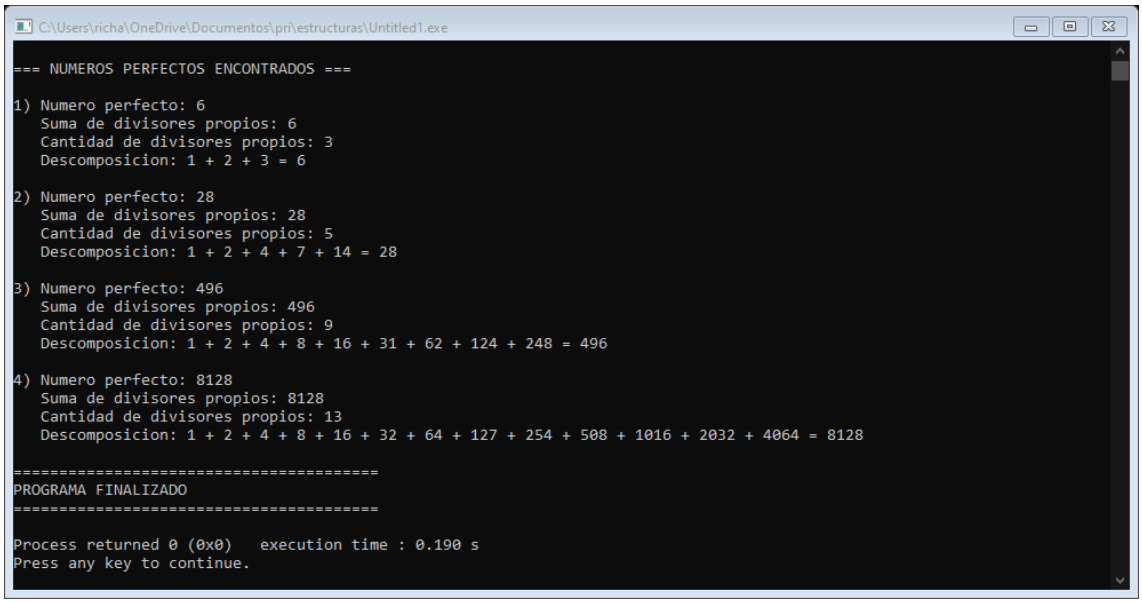
```
#include <stdio.h>
#include <stdbool.h>  // Para usar el tipo bool

// Estructura para almacenar información sobre un número perfecto
struct NumeroPerfecto {
   int valor;          // El número perfecto en sí
   int sumaDivisores;   // La suma de sus divisores propios
   int numDivisores;    // Cantidad de divisores propios
};

// Estructura para almacenar los resultados de búsqueda
struct ResultadoBusqueda {
   struct NumeroPerfecto numeros[4];  // Los 4 números perfectos
   int encontrados;                // Cuántos se han encontrado
   int numerosProbados;            // Cuántos números se han verificado
};

// Función que verifica si un número es perfecto
// Recibe: numero - el número a verificar
// Devuelve: true si es perfecto, false si no lo es
bool esPerfecto(int numero) {
   int suma = 0;

   // Calcular la suma de todos los divisores propios
   for (int i = 1; i <= numero / 2; i++) {
      if (numero % i == 0) {
```

```
            suma += i;
        }
    }

    // Verificar si la suma es igual al número original
    return suma == numero;
}

// Función que encuentra divisores y los almacena en una estructura
// Recibe: numero - el número a analizar
// Devuelve: Una estructura NumeroPerfecto con la información
struct NumeroPerfecto analizarNumero(int numero) {
    struct NumeroPerfecto resultado;
    resultado.valor = numero;
    resultado.sumaDivisores = 0;
    resultado.numDivisores = 0;

    // Encontrar todos los divisores propios
    for (int i = 1; i <= numero / 2; i++) {
        if (numero % i == 0) {
            resultado.sumaDivisores += i;
            resultado.numDivisores++;
        }
    }

    return resultado;
}

// Función que busca los primeros 4 números perfectos
// Devuelve: Una estructura con los resultados de la búsqueda
struct ResultadoBusqueda buscarPrimerosPerfectos() {
```

```
    struct ResultadoBusqueda resultado;
    resultado.encontrados = 0;
    resultado.numerosProbados = 0;

    int numero = 1;  // Empezamos desde el 1

    // Continuar hasta encontrar 4 números perfectos
    while (resultado.encontrados < 4) {
        if (esPerfecto(numero)) {
            // Almacenar la información del número perfecto encontrado
            resultado.numeros[resultado.encontrados] = analizarNumero(numero);
            resultado.encontrados++;
        }
        resultado.numerosProbados++;
        numero++;
    }

    return resultado;
}

// Función para mostrar los resultados de forma organizada
void mostrarResultados(struct ResultadoBusqueda resultado) {
    printf("=== RESULTADOS DE LA BUSQUEDA ===\n");
    printf("Numeros probados: %d\n", resultado.numerosProbados);
    printf("Numeros perfectos encontrados: %d\n\n", resultado.encontrados);

    printf("=== NUMEROS PERFECTOS ENCONTRADOS ===\n");
    for (int i = 0; i < resultado.encontrados; i++) {
        struct NumeroPerfecto np = resultado.numeros[i];

        printf("\n%d) Numero perfecto: %d\n", i + 1, np.valor);
```

```
        printf("   Suma de divisores propios: %d\n", np.sumaDivisores);
        printf("   Cantidad de divisores propios: %d\n", np.numDivisores);

        // Mostrar la descomposición (1 + 2 + 3 + ...)
        printf("   Descomposicion: ");
        int contador = 0;
        for (int j = 1; j <= np.valor / 2; j++) {
            if (np.valor % j == 0) {
                printf("%d", j);
                contador++;
                if (contador < np.numDivisores) {
                    printf(" + ");
                }
            }
        }
        printf(" = %d\n", np.valor);
    }
}

// Función principal del programa
int main() {
    printf("========================================\n");
    printf("BUSCADOR DE NUMEROS PERFECTOS\n");
    printf("========================================\n\n");

    printf("Un numero perfecto es igual a la suma de sus divisores propios.\n");
    printf("Ejemplo: 6 = 1 + 2 + 3\n");
    printf("Buscando los primeros 4 numeros perfectos...\n\n");

    // Buscar los números perfectos
    struct ResultadoBusqueda resultado = buscarPrimerosPerfectos();
```

```
    // Mostrar los resultados
    mostrarResultados(resultado);

    printf("\n========================================\n");
    printf("PROGRAMA FINALIZADO\n");
    printf("========================================\n");

    return 0;
}
```

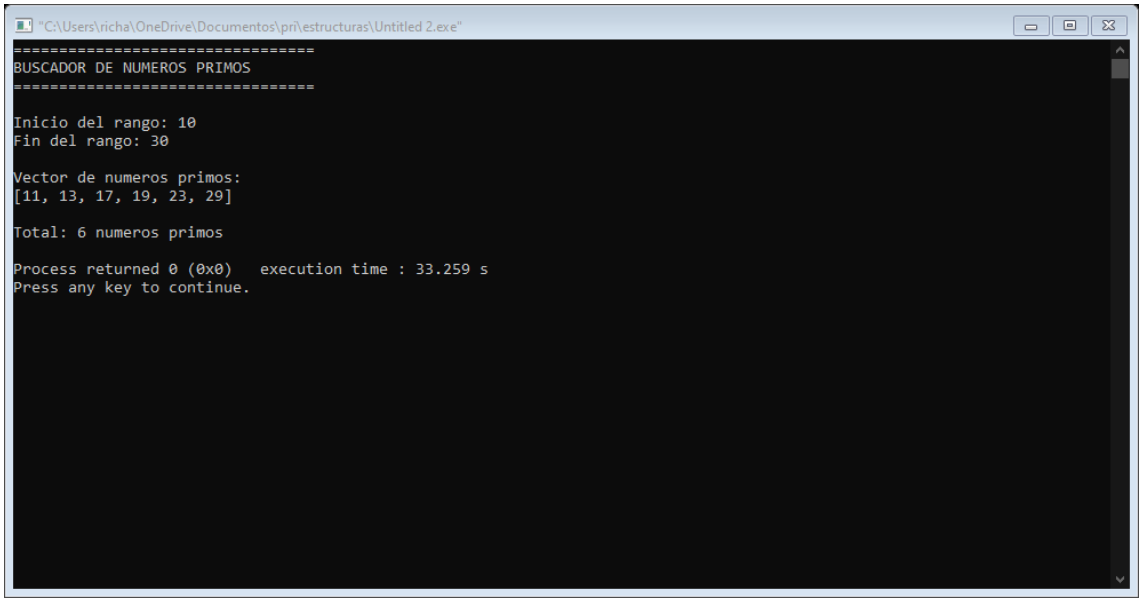


https://onlinegdb.com/gWMsW63ju

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

// Estructura simple para almacenar primos
struct Resultado {
    int primos[1000];  // Tamaño fijo para simplificar
    int cantidad;
};
```

```
// Función esprimo exactamente como se pide
bool esprimo(int n) {
    if (n <= 1) return false;
    if (n == 2) return true;
    if (n % 2 == 0) return false;

    int limite = sqrt(n);
    for (int i = 3; i <= limite; i += 2) {
        if (n % i == 0) return false;
    }

    return true;
}

int main() {
    int inicio, fin;

    printf("==================================\n");
    printf("BUSCADOR DE NUMEROS PRIMOS\n");
    printf("==================================\n\n");

    // Leer rango
    printf("Inicio del rango: ");
    scanf("%d", &inicio);
    printf("Fin del rango: ");
    scanf("%d", &fin);

    // Crear estructura para resultados
    struct Resultado resultado;
    resultado.cantidad = 0;
```

```
    // Buscar primos
    for (int i = inicio; i <= fin; i++) {
        if (esprimo(i)) {
            resultado.primos[resultado.cantidad] = i;
            resultado.cantidad++;
        }
    }

    // Mostrar vector de primos
    printf("\nVector de numeros primos:\n");
    printf("[");
    for (int i = 0; i < resultado.cantidad; i++) {
        printf("%d", resultado.primos[i]);
        if (i < resultado.cantidad - 1) printf(", ");
    }
    printf("]\n");

    // Resumen
    printf("\nTotal: %d numeros primos\n", resultado.cantidad);

    return 0;
}
```

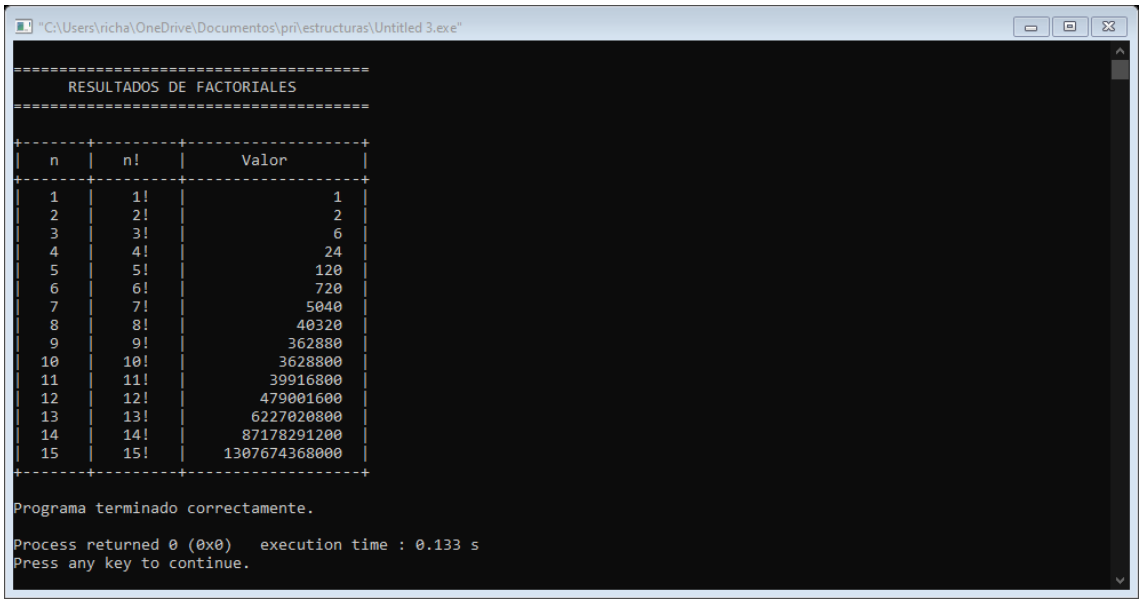


https://onlinegdb.com/XDYcCfeiz

```
#include <stdio.h>

// ============== ESTRUCTURAS ==============
struct FactorialData {
    int numero;
    long long factorial;
};

// ============== FUNCION CALC_FACT ==============
long long calc_fact(int n) {
    long long resultado = 1;

    // Calcular factorial
    for(int i = 1; i <= n; i++) {
        resultado = resultado * i;
    }

    return resultado;
}
```

```
// ============== PROGRAMA PRINCIPAL ==============
int main() {
   int i;

   // 1. Crear vector de 15 números (vec)
   int vec[15];
   printf("Vector de 15 numeros:\n");
   for(i = 0; i < 15; i++) {
      vec[i] = i + 1;  // Números del 1 al 15
      printf("%d ", vec[i]);
   }

   // 2. Crear estructura para factoriales
   struct FactorialData datos[15];

   // 3. Calcular factoriales usando calc_fact
   printf("\n\nCalculando factoriales...\n");
   for(i = 0; i < 15; i++) {
      datos[i].numero = vec[i];
      datos[i].factorial = calc_fact(vec[i]);
   }

   // 4. Mostrar resultados
   printf("\n=======================================\n");
   printf("      RESULTADOS DE FACTORIALES      \n");
   printf("=======================================\n");

   printf("\n+-------+---------+-------------------+\n");
   printf("|   n   |   n!    |      Valor        |\n");
   printf("+-------+---------+-------------------+\n");
```

```
    for(i = 0; i < 15; i++) {
        printf("|  %2d   |   %2d!   | %15lld  |\n",
               datos[i].numero,
               datos[i].numero,
               datos[i].factorial);
    }

    printf("+-------+---------+-------------------+\n");

    printf("\nPrograma terminado correctamente.\n");

    return 0;
}
```

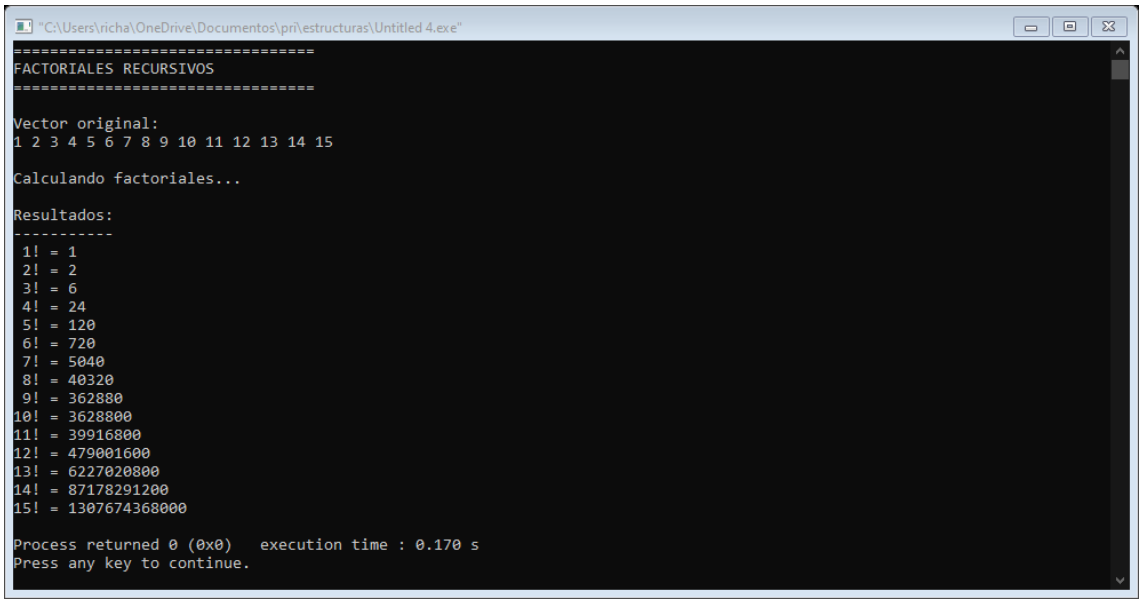


https://onlinegdb.com/JPAekl2cS

```
#include <stdio.h>

struct FactData {
    int n;
    long long fact;
};
```

```
long long calc_fact(int n) {
    if (n <= 1) return 1;
    return n * calc_fact(n - 1);
}

int main() {
    struct FactData datos[15];

    printf("================================\n");
    printf("FACTORIALES RECURSIVOS\n");
    printf("================================\n\n");

    printf("Vector original:\n");
    for (int i = 0; i < 15; i++) {
        datos[i].n = i + 1;
        printf("%d ", datos[i].n);
    }

    printf("\n\nCalculando factoriales...\n\n");

    for (int i = 0; i < 15; i++) {
        datos[i].fact = calc_fact(datos[i].n);
    }

    printf("Resultados:\n");
    printf("-----------\n");
    for (int i = 0; i < 15; i++) {
        printf("%2d! = %lld\n", datos[i].n, datos[i].fact);
    }
```

```
    return 0;
}
```



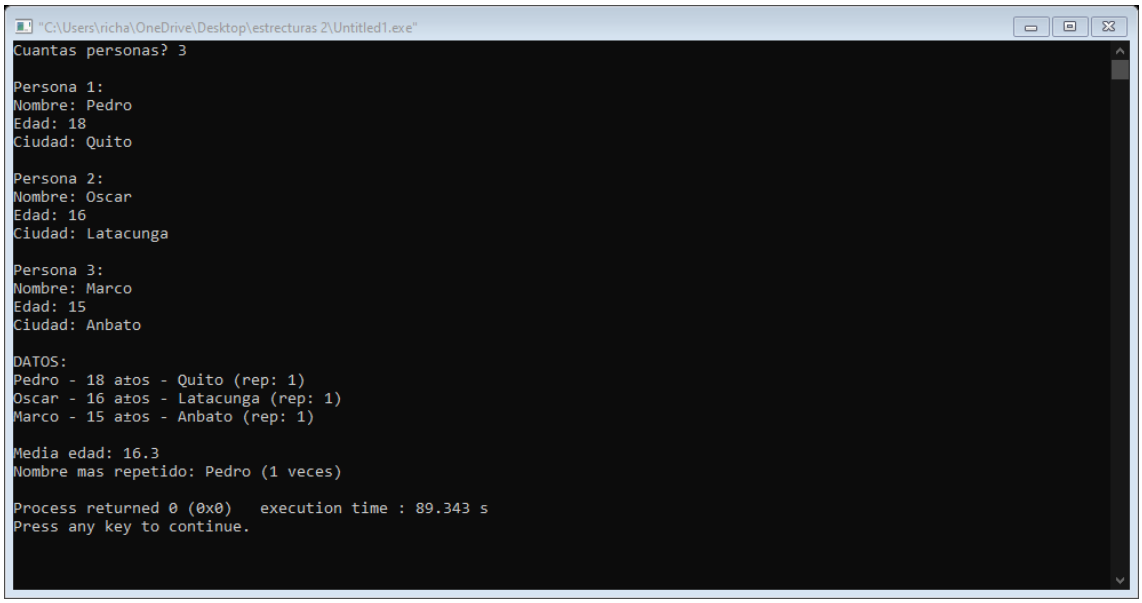
https://onlinegdb.com/DylEnJyKN

2.

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

struct p {
    char nom[50];
    int edad;
    char ciu[50];
    int rep;
};

int main() {
    struct p per[10];
    int n, i, j, suma = 0, max = 0;
    char mas_repetido[50];

    printf("Cuantas personas? ");
    scanf("%d", &n);
```

```
getchar();

// Entrada
for(i = 0; i < n; i++) {
   printf("\nPersona %d:\n", i+1);
   printf("Nombre: "); fgets(per[i].nom, 50, stdin);
   per[i].nom[strlen(per[i].nom)-1] = '\0';

   printf("Edad: "); scanf("%d", &per[i].edad);
   getchar();
   suma += per[i].edad;

   printf("Ciudad: "); fgets(per[i].ciu, 50, stdin);
   per[i].ciu[strlen(per[i].ciu)-1] = '\0';

   per[i].rep = 1;
}

// Calcular repeticiones
for(i = 0; i < n; i++) {
   for(j = i+1; j < n; j++) {
      if(strcmp(per[i].nom, per[j].nom) == 0) {
         per[i].rep++;
         per[j].rep++;
      }
   }
   if(per[i].rep > max) {
      max = per[i].rep;
      strcpy(mas_repetido, per[i].nom);
   }
}
```

```
    // Salida
    printf("\nDATOS:\n");
    for(i = 0; i < n; i++) {
        printf("%s - %d años - %s (rep: %d)\n",
               per[i].nom, per[i].edad, per[i].ciu, per[i].rep);
    }

    printf("\nMedia edad: %.1f\n", (float)suma/n);
    printf("Nombre mas repetido: %s (%d veces)\n", mas_repetido, max);

    return 0;
}
```



https://onlinegdb.com/R4hJTzCJa

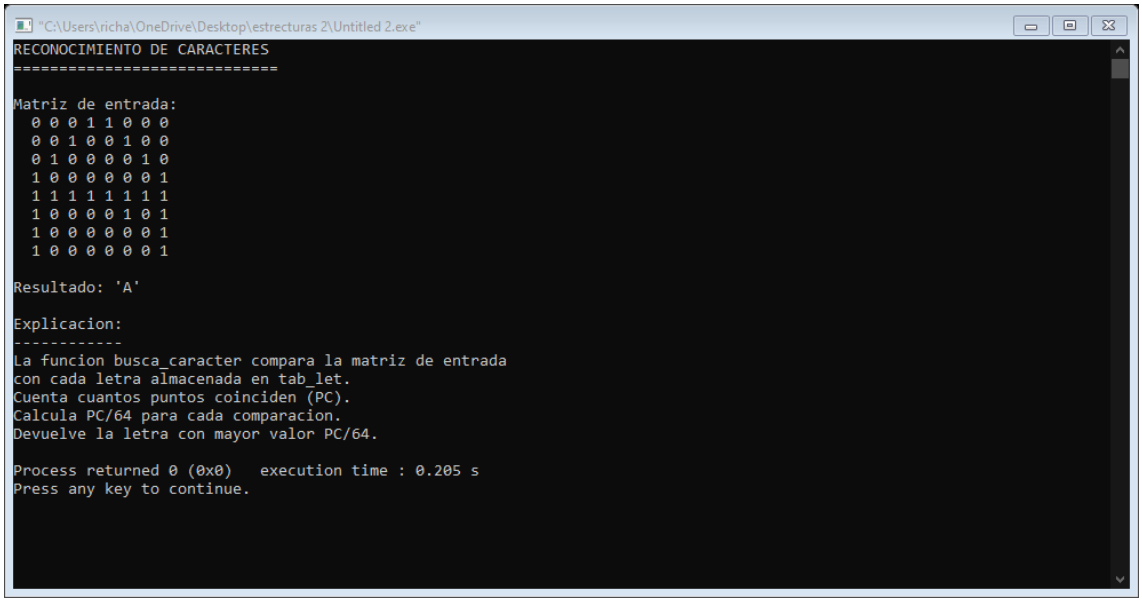
```
#include <stdio.h>

// ==================== ESTRUCTURAS ====================

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

```
};

// ==================== FUNCIÓN BUSCA_CARACTER ====================

char busca_caracter(int mp[8][8], struct letras tab_let[27]) {
   float max_similitud = 0;
   char mejor_letra = '?';

   // Comparar con cada letra
   for (int l = 0; l < 27; l++) {
      int coincidencias = 0;

      // Contar puntos que coinciden
      for (int i = 0; i < 8; i++) {
         for (int j = 0; j < 8; j++) {
            if (mp[i][j] == tab_let[l].mptos[i][j]) {
               coincidencias++;
            }
         }
      }

      // Calcular PC/64
      float similitud = (float)coincidencias / 64.0;

      // Guardar la mejor
      if (similitud > max_similitud) {
         max_similitud = similitud;
         mejor_letra = tab_let[l].cod_ASCII;
      }
   }
```

```
    return mejor_letra;
}

// ==================== EJEMPLO DE USO ====================

int main() {
    struct letras alfabeto[27];
    int mi_matriz[8][8];

    // 1. Inicializar algunas letras (ejemplo simplificado)
    // Letra 'A'
    alfabeto[0].cod_ASCII = 'A';
    int ejemplo_A[8][8] = {
        {0,0,0,1,1,0,0,0},
        {0,0,1,0,0,1,0,0},
        {0,1,0,0,0,0,1,0},
        {1,0,0,0,0,0,0,1},
        {1,1,1,1,1,1,1,1},
        {1,0,0,0,0,0,0,1},
        {1,0,0,0,0,0,0,1},
        {1,0,0,0,0,0,0,1}
    };

    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            alfabeto[0].mptos[i][j] = ejemplo_A[i][j];
        }
    }

    // Letra 'B' (simplificada)
    alfabeto[1].cod_ASCII = 'B';
```

```
for (int i = 0; i < 8; i++) {
   for (int j = 0; j < 8; j++) {
      // Matriz simple para B
      alfabeto[1].mptos[i][j] = 0;
   }
}

// 2. Crear matriz de prueba (una letra 'A' con pequeños cambios)
for (int i = 0; i < 8; i++) {
   for (int j = 0; j < 8; j++) {
      mi_matriz[i][j] = ejemplo_A[i][j];
   }
}
// Añadir algo de "ruido"
mi_matriz[3][4] = 0;
mi_matriz[5][5] = 1;

// 3. Usar la función busca_caracter
char resultado = busca_caracter(mi_matriz, alfabeto);

printf("RECONOCIMIENTO DE CARACTERES\n");
printf("============================\n\n");

// Mostrar matriz de entrada
printf("Matriz de entrada:\n");
for (int i = 0; i < 8; i++) {
   printf("  ");
   for (int j = 0; j < 8; j++) {
      printf("%d ", mi_matriz[i][j]);
   }
   printf("\n");
```

```
    }

    printf("\nResultado: '%c'\n", resultado);

    // Explicación
    printf("\nExplicacion:\n");
    printf("------------\n");
    printf("La funcion busca_caracter compara la matriz de entrada\n");
    printf("con cada letra almacenada en tab_let.\n");
    printf("Cuenta cuantos puntos coinciden (PC).\n");
    printf("Calcula PC/64 para cada comparacion.\n");
    printf("Devuelve la letra con mayor valor PC/64.\n");

    return 0;
}
```



https://onlinegdb.com/f2DdSZekT

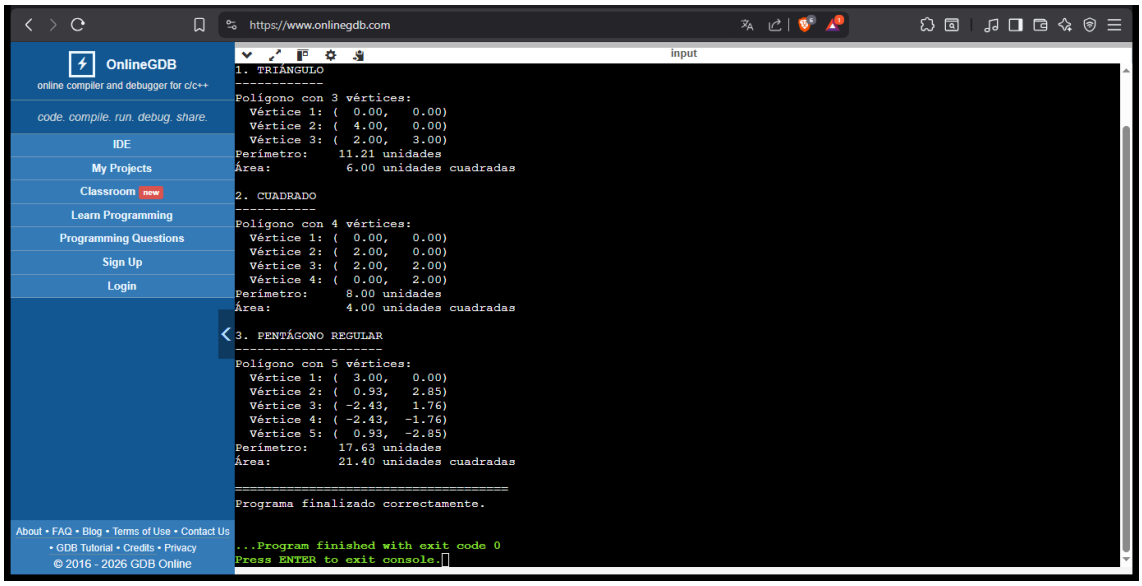
```
#include <stdio.h>
#include <math.h>

#define MAX_VERTICES 100
```

```
typedef struct {
   float x, y;
} Punto;

typedef struct {
   int nvert;
   Punto vertices[MAX_VERTICES];
} Poligono;

Poligono crear_poligono() {
   Poligono p;
   p.nvert = 0;
   return p;
}

int agregar_vertice(Poligono *pol, float x, float y) {
   if (pol->nvert >= MAX_VERTICES) return 0;

   pol->vertices[pol->nvert].x = x;
   pol->vertices[pol->nvert].y = y;
   pol->nvert++;
   return 1;
}

float distancia(Punto a, Punto b) {
   float dx = b.x - a.x;
   float dy = b.y - a.y;
   return sqrt(dx*dx + dy*dy);
}

float calcular_perimetro(Poligono pol) {
```

```
    if (pol.nvert < 2) return 0;

    float perimetro = 0;
    for (int i = 0; i < pol.nvert; i++) {
        int j = (i + 1) % pol.nvert;
        perimetro += distancia(pol.vertices[i], pol.vertices[j]);
    }
    return perimetro;
}

float calcular_area(Poligono pol) {
    if (pol.nvert < 3) return 0;

    float area = 0;
    for (int i = 0; i < pol.nvert; i++) {
        int j = (i + 1) % pol.nvert;
        area += pol.vertices[i].x * pol.vertices[j].y;
        area -= pol.vertices[j].x * pol.vertices[i].y;
    }
    return fabs(area) / 2;
}

void mostrar_poligono(Poligono pol) {
    printf("Polígono con %d vértices:\n", pol.nvert);
    for (int i = 0; i < pol.nvert; i++) {
        printf("  Vértice %d: (%6.2f, %6.2f)\n", i+1, pol.vertices[i].x, pol.vertices[i].y);
    }
}

int main() {
    printf("====================================\n");
```

```
printf("       SISTEMA DE POLÍGONOS\n");
printf("====================================\n\n");

// Triángulo
printf("1. TRIÁNGULO\n");
printf("------------\n");
Poligono triangulo = crear_poligono();
agregar_vertice(&triangulo, 0, 0);
agregar_vertice(&triangulo, 4, 0);
agregar_vertice(&triangulo, 2, 3);

mostrar_poligono(triangulo);
printf("Perímetro: %8.2f unidades\n", calcular_perimetro(triangulo));
printf("Área:      %8.2f unidades cuadradas\n\n", calcular_area(triangulo));

// Cuadrado
printf("2. CUADRADO\n");
printf("-----------\n");
Poligono cuadrado = crear_poligono();
agregar_vertice(&cuadrado, 0, 0);
agregar_vertice(&cuadrado, 2, 0);
agregar_vertice(&cuadrado, 2, 2);
agregar_vertice(&cuadrado, 0, 2);

mostrar_poligono(cuadrado);
printf("Perímetro: %8.2f unidades\n", calcular_perimetro(cuadrado));
printf("Área:      %8.2f unidades cuadradas\n\n", calcular_area(cuadrado));

// Pentágono
printf("3. PENTÁGONO REGULAR\n");
printf("--------------------\n");
```

```
    Poligono pentagono = crear_poligono();
    float radio = 3;
    for (int i = 0; i < 5; i++) {
        float angulo = 2 * 3.14159 * i / 5;
        agregar_vertice(&pentagono, radio * cos(angulo), radio * sin(angulo));
    }

    mostrar_poligono(pentagono);
    printf("Perímetro: %8.2f unidades\n", calcular_perimetro(pentagono));
    printf("Área:      %8.2f unidades cuadradas\n\n", calcular_area(pentagono));

    printf("====================================\n");
    printf("Programa finalizado correctamente.\n");

    return 0;
}
```



https://onlinegdb.com/69XATHaf7

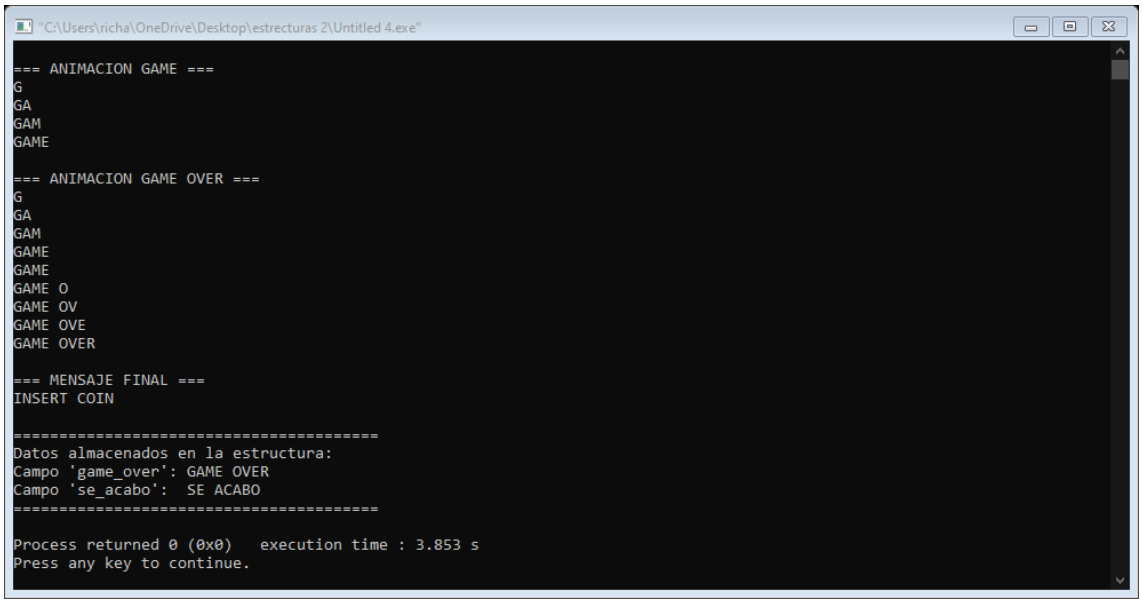
```
#include <stdio.h>
#include <string.h>
```

```
// Definición de constantes según el enunciado
#define N 10
#define M 12

// Estructura para almacenar los mensajes
struct mensaje {
   char game_over[N];
   char se_acabo[N];
};

typedef struct mensaje men;

// Función para mostrar la animación completa
void mostrarAnimacionCompleta() {
   printf("\n=== ANIMACION GAME ===\n");
   printf("G\n");
   printf("GA\n");
   printf("GAM\n");
   printf("GAME\n");

   printf("\n=== ANIMACION GAME OVER ===\n");
   printf("G\n");
   printf("GA\n");
   printf("GAM\n");
   printf("GAME\n");
   printf("GAME \n");
   printf("GAME O\n");
   printf("GAME OV\n");
   printf("GAME OVE\n");
   printf("GAME OVER\n");
```

```
    printf("\n=== MENSAJE FINAL ===\n");
    printf("INSERT COIN\n");
}

// Función para mostrar la animación paso a paso
void mostrarAnimacionPasoAPaso() {
    printf("Presione Enter para continuar cada paso...\n\n");

    // Animación GAME
    printf("Paso 1: G\n"); getchar();
    printf("Paso 2: GA\n"); getchar();
    printf("Paso 3: GAM\n"); getchar();
    printf("Paso 4: GAME\n"); getchar();

    printf("\n--- Continuando con GAME OVER ---\n\n");

    // Animación GAME OVER
    printf("Paso 5: GAME \n"); getchar();
    printf("Paso 6: GAME O\n"); getchar();
    printf("Paso 7: GAME OV\n"); getchar();
    printf("Paso 8: GAME OVE\n"); getchar();
    printf("Paso 9: GAME OVER\n"); getchar();

    printf("\n=== MENSAJE FINAL ===\n");
    printf("INSERT COIN\n");
}

int main() {
    // Inicializar estructura
    men mensaje_final;
    strcpy(mensaje_final.game_over, "GAME OVER");
```

```
strcpy(mensaje_final.se_acabo, "SE ACABO");

printf("=======================================\n");
printf("    GAME OVER - JUEGO DE PALABRAS\n");
printf("=======================================\n");
printf("Seleccione una opcion:\n");
printf("1. Ver animacion completa\n");
printf("2. Ver animacion paso a paso (con Enter)\n");
printf("3. Solo mostrar estructura\n");
printf("Opcion: ");

int opcion;
scanf("%d", &opcion);

switch(opcion) {
   case 1:
      mostrarAnimacionCompleta();
      break;
   case 2:
      // Limpiar buffer
      while(getchar() != '\n');
      mostrarAnimacionPasoAPaso();
      break;
   case 3:
      printf("\n=== ESTRUCTURA MENSAJE ===\n");
      printf("game_over: %s\n", mensaje_final.game_over);
      printf("se_acabo:  %s\n", mensaje_final.se_acabo);
      break;
   default:
      printf("Opcion no valida\n");
}
```

```
    // Mostrar estructura siempre
    printf("\n=======================================\n");
    printf("Datos almacenados en la estructura:\n");
    printf("Campo 'game_over': %s\n", mensaje_final.game_over);
    printf("Campo 'se_acabo':  %s\n", mensaje_final.se_acabo);
    printf("=======================================\n");


    return 0;
}
```



https://onlinegdb.com/E1geI5GBE