SOLUCIONES

Tecnólogo en Informática • Mariano Zunino

Lenguaje de Programación C • Estructura de Datos y Algoritmos

1. PUNTEROS

Ejercicios Prácticos

Ejercicios Básicos de Punteros

Ejercicio 1

```c
int *punt;
int x = 7, y = 5;
punt = &x;
*punt = 4;
printf("%d, %d", x, y);
```

Seguimiento:

- `punt = &x` ⟶ punt apunta a x
- `*punt = 4` ⟶ modifica x a través del puntero (x = 4)

Solución: imprime `4, 5`

Ejercicio 2

```c
int *punt;
int x = 7, y = 5;
punt = &x;
x = 4;
printf("%d, %d", *punt, y);
```

Seguimiento:

- `punt = &x` ⟶ punt apunta a x
- `x = 4` ⟶ modifica x directamente
- `*punt` imprime el valor de x (4)

Solución: imprime `4, 5`

Ejercicio 3

```c
int *punt;
int x = 7, y = 5;
punt = &x;
x = 4;
punt = &y;
printf("%d, %d", *punt, x);
```

Seguimiento:

- `punt = &x` ⟶ punt apunta a x
- `x = 4` ⟶ modifica x
- `punt = &y` ⟶ punt ahora apunta a y

- `*punt` imprime el valor de y (5)

Solución: imprime `5, 4`

Ejercicio 4

```c
int *punt;
int x = 7, y = 5;
punt = &x;
*punt = 3;
punt = &y;
*punt = x;
x = 9;
printf("%d, %d", *punt, x);
```

Seguimiento:

- `*punt = 3` ⟶ modifica x = 3
- `punt = &y` ⟶ punt apunta a y
- `*punt = x` ⟶ modifica y = 3 (valor actual de x)
- `x = 9` ⟶ modifica x = 9
- `*punt` imprime y (3), x imprime 9

Solución: imprime `3, 9`

Ejercicio 5

```c
int *punta, *puntb;
int x = 7, y = 5;
punta = &x;
*punta = 3;
puntb = &y;
*puntb = x;
x = 9;
printf("%d, %d", *puntb, x);
```

Seguimiento:

- `*punta = 3` ⟶ modifica x = 3
- `*puntb = x` ⟶ modifica y = 3 (valor actual de x)
- `x = 9` ⟶ modifica x = 9
- `*puntb` imprime y (3)

Solución: imprime `3, 9`

Ejercicio 6

```c
int *punta, *puntb;
int x = 7, y = 5;
punta = &x;
*punta = 3;
puntb = &y;
*puntb = x;
x = 9;
printf("%d, %d", *puntb, *punta);
```

Seguimiento:
- `*punta = 3` ⟶ modifica x = 3
- `*puntb = x` ⟶ modifica y = 3
- `x = 9` ⟶ modifica x = 9
- `*puntb` imprime y (3), `*punta` imprime x (9)

Solución: imprime `3, 9`

Ejercicio 7

```c
int *punta, *puntb;
int x = 7, y = 5;
punta = &x;
*punta = 3;
puntb = &y;
*puntb = x;
x = 9;
puntb = punta;
printf("%d, %d", *puntb, y);
```

Seguimiento:
- `*punta = 3` ⟶ modifica x = 3
- `*puntb = x` ⟶ modifica y = 3
- `x = 9` ⟶ modifica x = 9
- `puntb = punta` ⟶ puntb apunta a x
- `*puntb` imprime x (9), y mantiene valor 3

Solución: imprime `9, 3`

Ejercicios de Punteros y Arrays

Ejercicio 8

```c
int *punt, i;
int x[5] = {1, 2, 3, 4, 5};
punt = x;
*punt = 9;
for(i = 0; i < 5; i++)
    printf("%d,", x[i]);
```

Seguimiento:
- Array inicial: `[1,2,3,4,5]`
- `punt = x` ⟶ punt apunta a x[0]
- `*punt = 9` ⟶ modifica x[0] = 9

Solución: imprime `9,2,3,4,5,`

Ejercicio 9

```c
int *punt, i;
int x[5] = {1, 2, 3, 4, 5};
punt = &x[0];
*punt = 9;
```

```
punt[3] = 7;
for(i = 0; i < 5; i++)
    printf("%d,", x[i]);
```

Seguimiento:

- `punt = &x[0]` ⟶ punt apunta a x[0]
- `*punt = 9` ⟶ modifica x[0] = 9
- `punt[3] = 7` ⟶ modifica x[3] = 7

Solución: imprime `9,2,3,7,5,`

Ejercicio 10

```
int *punt, i;
int x[5] = {1, 2, 3, 4, 5};
punt = x;
*x = 11;
*(punt + 3) = 9;
for(i = 0; i < 5; i++)
    printf("%d,", x[i]);
```

Seguimiento:

- `*x = 11` ⟶ modifica x[0] = 11
- `*(punt + 3) = 9` ⟶ modifica x[3] = 9

Solución: imprime `11,2,3,9,5,`

Ejercicio 11

```
int *punt, i;
int x[5] = {1, 2, 3, 4, 5};
punt = x;
*(punt + 2) = 9;
*(x + 3) = 7;
punt[1] = 11;
for(i = 0; i < 5; i++)
    printf("%d,", *(punt + i));
```

Seguimiento:

- `*(punt + 2) = 9` ⟶ modifica x[2] = 9
- `*(x + 3) = 7` ⟶ modifica x[3] = 7
- `punt[1] = 11` ⟶ modifica x[1] = 11

Solución: imprime `1,11,9,7,5,`

Ejercicio 12

```
int *punt, i;
int x[5] = {1, 2, 3, 4, 5};
punt = x + 4;
*(punt - 2) = 9;
punt--;
*punt = 7;
```

```
punt[1] = 11;
for(i = 0; i < 5; i++)
    printf("%d,", *(x + i));
```

Seguimiento:

- `punt = x + 4` ⟶ punt apunta a x[4]
- `*(punt - 2) = 9` ⟶ modifica x[2] = 9
- `punt--` ⟶ punt apunta a x[3]
- `*punt = 7` ⟶ modifica x[3] = 7
- `punt[1] = 11` ⟶ modifica x[4] = 11

Solución: imprime `1,2,9,7,11,`

Ejercicio 13

```
int *punt, i;
int x[5] = {1, 2, 3, 4, 5};
punt = &x[0] + 3;
*(punt - 2) = 9;
punt--;
*punt = 7;
punt[1] = 11;
punt = x;
for(i = 0; i < 5; i++)
    printf("%d,", punt[i]);
```

Seguimiento:

- `punt = &x[0] + 3` ⟶ punt apunta a x[3]
- `*(punt - 2) = 9` ⟶ modifica x[1] = 9
- `punt--` ⟶ punt apunta a x[2]
- `*punt = 7` ⟶ modifica x[2] = 7
- `punt[1] = 11` ⟶ modifica x[3] = 11

Solución: imprime `1,9,7,11,5,`

Estructura de Datos y Algoritmos - Ejercicios de Punteros y Estructuras

Ejercicios de Punteros y Funciones

Ejercicio 14

```c
void suma_dos(int *x, int *y, int *z) {
    *x = *x + 2;
    *y = *y + 2;
    *z = *z + 2;
}

int main() {
    int x = 3, y = 10, z = 15;
    suma_dos(&x, &y, &z);
    printf("%d %d %d", x, y, z);
    return 0;
}
```

Seguimiento:

- La función recibe las direcciones de x, y, z
- Suma 2 a cada valor a través de los punteros
- Variables originales se modifican: x=5, y=12, z=17

Solución: imprime `5 12 17`

Ejercicio 15

```c
void datos(int *x, float *y, char *c) {
    *x = 8;
    *y = 4.2;
    *c = 'g';
}

int main() {
    int x = 9;
    float y = 44.6;
    char c = 'a';
    datos(&x, &y, &c);
    printf("%d %.1f %c", x, y, c);
    return 0;
}
```

Seguimiento:

- La función modifica las variables originales
- x = 8, y = 4.2, c = “g”

Solución: imprime `8 4.2 g`

Ejercicio 16

```c
void datos(int *x, float *y, char *c) {
    printf("%p %.1f %c ", x, *y, *c);
    *x = 8;
    *y = 4.2;
    *c = 'g';
}
```

```c
int main() {
    int x = 9;
    float y = 44.6;
    char c = 'a';
    datos(&x, &y, &c);
    printf("%d %.1f %c", x, y, c);
    return 0;
}
```

Seguimiento:
- Función imprime dirección de x, valores de y y c
- Luego modifica las variables originales

Solución: imprime `[dirección_x] 44.6 a 8 4.2 g`

Ejercicio 17

```c
void datos(int x, float y, char c) {
    printf("%d %.1f %c ", x, y, c);
    x = 8;
    y = 4.2;
    c = 'g';
}

int main() {
    int x = 9;
    float y = 44.6;
    char c = 'a';
    datos(x, y, c);
    printf("%d %.1f %c", x, y, c);
    return 0;
}
```

Seguimiento:
- Paso por valor ⟶ modificaciones en función no afectan variables originales
- Variables originales mantienen valores iniciales

Solución: imprime `9 44.6 a 9 44.6 a`

Ejercicio 18

```c
int datos(int x, float y, char c) {
    printf("%d %.1f %c ", x, y, c);
    x = 8;
    y = 4.2;
    c = 'g';
    return x;
}

int main() {
    int x = 9;
    float y = 44.6;
    char c = 'a';
    x = datos(x, y, c);
```

```
    printf("%d %.1f %c", x, y, c);
    return 0;
}
```

Seguimiento:

- Función retorna 8 y se asigna a x
- y y c mantienen valores originales (paso por valor)

Solución: imprime `9 44.6 a 8 44.6 a`

Ejercicio 19

```
char datos(int *x, float *y, char *c) {
    printf("%p %.1f %c ", x, *y, *c);
    *x = 8;
    *y = 4.2;
    *c = 'g';
    return 'h';
}

int main() {
    int x = 9;
    float y = 44.6;
    char c = 'a';
    c = datos(&x, &y, &c);
    printf("%d %.1f %c", x, y, c);
    return 0;
}
```

Seguimiento:

- Función modifica x=8, y=4.2 y retorna “h” que se asigna a c

Solución: imprime `[dirección_x] 44.6 a 8 4.2 h`

2. ESTRUCTURAS (STRUCTS)

Ejercicios Prácticos

Ejercicio 1: Comprensión Básica

```c
struct jugador {
    char nombre[50];
    int edad;
    float altura;
};

int main() {
    struct jugador mijugador;
    strcpy(mijugador.nombre, "MarcoPolo");
    mijugador.edad = 45;
    mijugador.altura = 1.8;
    printf("Nombre: %s\nAltura: %.1f\nEdad: %d\n",
           mijugador.nombre, mijugador.altura, mijugador.edad);
    return 0;
}
```

Solución: imprime:

```
Nombre: MarcoPolo
Altura: 1.8
Edad: 45
```

Ejercicio 2: Arrays de Estructuras

```c
struct medidas {
    int alto, ancho, largo;
};

int main() {
    int i;
    struct medidas cubiletes[5];

    for(i = 0; i < 5; i++) {
        cubiletes[i].alto = 4;
        cubiletes[i].ancho = 2 * i;
        cubiletes[i].largo = i + 1;
    }

    for(i = 0; i < 5; i++) {
        printf("Cubilete %d: %d alto, %d ancho, %d largo\n",
               i, cubiletes[i].alto, cubiletes[i].ancho, cubiletes[i].largo);
    }
    return 0;
}
```

Solución: imprime:

```
Cubilete 0: 4 alto, 0 ancho, 1 largo
Cubilete 1: 4 alto, 2 ancho, 2 largo
Cubilete 2: 4 alto, 4 ancho, 3 largo
```

```
Cubilete 3: 4 alto, 6 ancho, 4 largo
Cubilete 4: 4 alto, 8 ancho, 5 largo
```

Ejercicio 3: Sistema de Gestión de Jugadores

Implementación completa del sistema con menú:

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

struct jugador {
    char nombre[50];
    int edad;
    float altura;
};

void listar_nombres(struct jugador jugadores[], int cantidad) {
    printf("\n=== NOMBRES ===\n");
    for(int i = 0; i < cantidad; i++) {
        printf("%d. %s\n", i+1, jugadores[i].nombre);
    }
}

void listar_alturas(struct jugador jugadores[], int cantidad) {
    printf("\n=== ALTURAS ===\n");
    for(int i = 0; i < cantidad; i++) {
        printf("%d. %.2f m\n", i+1, jugadores[i].altura);
    }
}

void listar_edades(struct jugador jugadores[], int cantidad) {
    printf("\n=== EDADES ===\n");
    for(int i = 0; i < cantidad; i++) {
        printf("%d. %d años\n", i+1, jugadores[i].edad);
    }
}

int main() {
    struct jugador jugadores[10];
    int opcion;

    // Solicitar datos de jugadores
    printf("=== INGRESO DE JUGADORES ===\n");
    for(int i = 0; i < 10; i++) {
        printf("\nJugador %d:\n", i+1);
        printf("Nombre: ");
        scanf("%s", jugadores[i].nombre);
        printf("Edad: ");
        scanf("%d", &jugadores[i].edad);
        printf("Altura (m): ");
        scanf("%f", &jugadores[i].altura);
    }

    // Menú principal
    do {
```

```
        printf("\n=== MENU PRINCIPAL ===\n");
        printf("1. Listar nombres\n");
        printf("2. Listar alturas\n");
        printf("3. Listar edades\n");
        printf("4. Salir\n");
        printf("Opción: ");
        scanf("%d", &opcion);

        switch(opcion) {
            case 1:
                listar_nombres(jugadores, 10);
                break;
            case 2:
                listar_alturas(jugadores, 10);
                break;
            case 3:
                listar_edades(jugadores, 10);
                break;
            case 4:
                printf("¡Hasta luego!\n");
                break;
            default:
                printf("Opción inválida\n");
        }
    } while(opcion != 4);

    return 0;
}
```

Solución: Programa interactivo que solicita datos de 10 jugadores y presenta un menú con las opciones solicitadas.

Ejercicio 4: Sistema Avanzado de Jugadores
Implementación completa con funcionalidades extendidas:

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

struct jugador {
    char nombre[50];
    int edad;
    float altura;
};

// Funciones del ejercicio anterior (listar_nombres, etc.)
// ... (código anterior) ...

void buscar_jugador(struct jugador jugadores[], int cantidad) {
    char nombre_buscar[50];
    int encontrado = 0;

    printf("\nIngrese nombre a buscar: ");
    scanf("%s", nombre_buscar);

    for(int i = 0; i < cantidad; i++) {
```

```
        if(strcmp(jugadores[i].nombre, nombre_buscar) == 0) {
            printf("\n=== JUGADOR ENCONTRADO ===\n");
            printf("Nombre: %s\n", jugadores[i].nombre);
            printf("Edad: %d años\n", jugadores[i].edad);
            printf("Altura: %.2f m\n", jugadores[i].altura);
            encontrado = 1;
            break;
        }
    }

    if(!encontrado) {
        printf("Jugador no encontrado.\n");
    }
}

void jugador_mas_alto(struct jugador jugadores[], int cantidad) {
    int indice_mayor = 0;

    for(int i = 1; i < cantidad; i++) {
        if(jugadores[i].altura > jugadores[indice_mayor].altura) {
            indice_mayor = i;
        }
    }

    printf("\n=== JUGADOR MÁS ALTO ===\n");
    printf("Nombre: %s\n", jugadores[indice_mayor].nombre);
    printf("Edad: %d años\n", jugadores[indice_mayor].edad);
    printf("Altura: %.2f m\n", jugadores[indice_mayor].altura);
}

void edad_promedio(struct jugador jugadores[], int cantidad) {
    int suma_edades = 0;

    for(int i = 0; i < cantidad; i++) {
        suma_edades += jugadores[i].edad;
    }

    float promedio = (float)suma_edades / cantidad;
    printf("\n=== EDAD PROMEDIO ===\n");
    printf("Promedio: %.2f años\n", promedio);
}

int main() {
    struct jugador jugadores[10];
    int opcion;

    // Solicitar datos (mismo código del ejercicio anterior)
    // ... (código de ingreso) ...

    // Menú extendido
    do {
        printf("\n=== MENU PRINCIPAL ===\n");
        printf("1. Listar nombres\n");
        printf("2. Listar alturas\n");
        printf("3. Listar edades\n");
        printf("4. Buscar jugador\n");
```

```
        printf("5. Jugador más alto\n");
        printf("6. Edad promedio\n");
        printf("7. Salir\n");
        printf("Opción: ");
        scanf("%d", &opcion);

        switch(opcion) {
            case 1: listar_nombres(jugadores, 10); break;
            case 2: listar_alturas(jugadores, 10); break;
            case 3: listar_edades(jugadores, 10); break;
            case 4: buscar_jugador(jugadores, 10); break;
            case 5: jugador_mas_alto(jugadores, 10); break;
            case 6: edad_promedio(jugadores, 10); break;
            case 7: printf("¡Hasta luego!\n"); break;
            default: printf("Opción inválida\n");
        }
    } while(opcion != 7);

    return 0;
}
```

Solución: Sistema completo de gestión con búsqueda, análisis de datos y cálculo de promedios.

Ejercicio 5: Geometría con Estructuras
Implementación completa del cálculo de perímetro:

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

struct punto {
    float x, y;
};

float calcular_distancia(struct punto p1, struct punto p2) {
    float dx = p2.x - p1.x;
    float dy = p2.y - p1.y;
    return sqrt(dx*dx + dy*dy);
}

int main() {
    struct punto p1, p2, p3;
    float lado1, lado2, lado3, perimetro;

    printf("=== CÁLCULO DE PERÍMETRO DE TRIÁNGULO ===\n");

    printf("Ingrese coordenadas del punto 1 (x y): ");
    scanf("%f %f", &p1.x, &p1.y);

    printf("Ingrese coordenadas del punto 2 (x y): ");
    scanf("%f %f", &p2.x, &p2.y);

    printf("Ingrese coordenadas del punto 3 (x y): ");
    scanf("%f %f", &p3.x, &p3.y);

    // Calcular distancias entre puntos
```

```
    lado1 = calcular_distancia(p1, p2);
    lado2 = calcular_distancia(p2, p3);
    lado3 = calcular_distancia(p3, p1);

    // Calcular perímetro
    perimetro = lado1 + lado2 + lado3;

    printf("\n=== RESULTADOS ===\n");
    printf("Punto 1: (%.2f, %.2f)\n", p1.x, p1.y);
    printf("Punto 2: (%.2f, %.2f)\n", p2.x, p2.y);
    printf("Punto 3: (%.2f, %.2f)\n", p3.x, p3.y);
    printf("\nDistancias:\n");
    printf("P1-P2: %.2f\n", lado1);
    printf("P2-P3: %.2f\n", lado2);
    printf("P3-P1: %.2f\n", lado3);
    printf("\nPerímetro del triángulo: %.2f\n", perimetro);

    return 0;
}
```

Nota: Compilar con `gcc programa.c -lm -o programa`

Solución: Programa que calcula y muestra el perímetro de un triángulo formado por tres puntos ingresados por el usuario.