Aquí tienes una propuesta de \*\*criterios de desempeño\*\* para una materia de \*\*Programación en Lenguaje C\*\* dirigida a universitarios, estructurados en categorías clave:

---

### \*\*1. Dominio de la Sintaxis y Estructuras Básicas\*\*

- Escribe código en C que cumpla con la sintaxis estándar (declaraciones, tipos de datos, operadores, estructuras de control).

- Implementa programas que utilicen:

- Entrada/salida básica (`printf`, `scanf`).

- Estructuras condicionales (`if-else`, `switch`).

- Bucles (`for`, `while`, `do-while`).

### \*\*2. Manejo de Funciones y Modularización\*\*

- Diseña programas divididos en funciones con propósito claro y parámetros bien definidos.

- Utiliza correctamente el ámbito de variables (locales, globales) y el paso de parámetros (por valor/referencia).

### \*\*3. Gestión de Memoria y Punteros\*\*

- Demuestra comprensión de punteros: declaración, aritmética y su relación con arreglos.

- Gestiona memoria dinámica (`malloc`, `calloc`, `free`) sin errores (fugas de memoria, accesos inválidos).

### \*\*4. Uso de Arreglos y Estructuras de Datos Básicas\*\*

- Implementa algoritmos que manipulen arreglos unidimensionales y multidimensionales.

- Crea y gestiona estructuras (`struct`) para modelar datos complejos.

### \*\*5. Resolución de Problemas y Algoritmos\*\*

- Desarrolla programas que resuelvan problemas específicos con lógica clara y eficiencia aceptable.

- Aplica algoritmos básicos (búsqueda, ordenamiento) y los implementa en C.

### \*\*6. Depuración y Validación\*\*

- Identifica y corrige errores de compilación, lógicos y runtime usando herramientas como `gdb` o análisis estático.

- Prueba programas con casos de uso diversos (incluyendo entradas inválidas).

### \*\*7. Documentación y Buenas Prácticas\*\*

- Documenta código con comentarios relevantes y sigue convenciones de estilo (ej. nombres descriptivos, indentación).

- Organiza el código en módulos coherentes (archivos `.h` y `.c` cuando aplique).

### \*\*8. Creatividad y Aplicación Práctica\*\*

- Propone soluciones innovadoras a problemas planteados, optimizando recursos.

- Integra conocimientos para proyectos pequeños (ej. sistemas de inventario, calculadoras avanzadas).

---

### \*\*Escala de Evaluación Sugerida\*\*

| \*\*Nivel\*\* | \*\*Descripción\*\* |

|-----------------|-------------------------------------------------------------------------------|

| \*\*Excelente\*\* | Cumple todos los criterios con código eficiente, documentado y sin errores. |

| \*\*Satisfactorio\*\* | Cumple criterios básicos con errores menores o documentación incompleta. |

| \*\*En desarrollo\*\* | Logra objetivos parciales, pero con errores frecuentes o lógica ineficiente. |

| \*\*No satisfactorio\*\* | No resuelve el problema o el código no funciona correctamente. |

---

\*\*Nota:\*\* Estos criterios pueden ajustarse según el nivel del curso (introductorio o avanzado) y los objetivos específicos de la asignatura. ¿Necesitas añadir algún enfoque en particular (ej. bajo nivel, embedded systems)?