

Propuesta Didáctica

Módulo: Programación – 1.º DAM

Simulación de gestión de memoria

2.1 ¿Qué supuesto queremos trabajar?

 Simular cómo un sistema operativo:

- Asigna y libera bloques de memoria
- Gestiona de forma **dinámica y controlada** los recursos
- Refleja el estado del sistema tras cada operación

 El alumnado desarrollará una aplicación didáctica que modele esta lógica.

2.2 Contextualización del alumnado

- 1.º curso del CFGS DAM
- Alumnado en etapa inicial de programación estructurada
- Nivel heterogéneo en cuanto a abstracción de bajo nivel
- Familiaridad progresiva con representación de memoria y estructuras

2.3 Conocimientos previos requeridos

- Variables y estructuras de datos básicas
- Funciones y paso de parámetros
- Condicionales y bucles
- Representación lógica de estados (arrays, booleanos...)
- Uso de entrada/salida

2.4 Objetivos de aprendizaje

- Comprender el concepto de **memoria contigua y fragmentación**
- Simular llamadas de reserva y liberación de memoria
- Visualizar dinámicamente el uso de recursos
- Familiarizarse con la lógica previa a la paginación
- Aplicar pensamiento algorítmico y gestión de estructuras

2.5 Metodología

- ◆ Activa y participativa
- ◆ Resolución guiada de casos paso a paso
- ◆ Enfoque visual del estado de memoria
- ◆ Desarrollo individual con asesoramiento continuo
- ◆ Explicación por descubrimiento: de lo concreto a lo abstracto

2.6 Material didáctico (DUA)

- Estructura visual de bloques de memoria (tabla o cuadrícula)
- Código base o pseudocódigo inicial
- Hoja de ejercicios con llamadas simuladas
- Alternativas de representación (digital o papel)
- Apoyo gráfico con ejemplos animados




2.7 Secuencia de acciones formativas

1. Introducción: ¿Qué es la memoria y cómo se gestiona?
2. Simulación paso a paso en pizarra con llamadas de ejemplo
3. Diseño de una estructura de memoria como array
4. Desarrollo del simulador por parejas
5. Visualización dinámica en consola o interfaz básica
6. Discusión de errores comunes: falta de hueco, huecos dispersos
7. Evaluación del trabajo + presentación voluntaria



2.8 Actividad principal

Simulación de gestión de memoria con llamadas desde un programa ficticio

- ◆ La memoria se modela como un array de N bloques contiguos
- ◆ El alumnado simula:
 - Solicitudes de reserva
 - Liberaciones
 - Consultas de estado
-  Cada operación se refleja **visualmente** en una tabla o cuadrícula
- ⚠ Se analiza **fragmentación externa** al fallar una reserva por falta de hueco contiguo

Relación con el contenido del tema

- Se reproducen operaciones reales de **asignación y liberación**
- Se introduce la lógica de **memoria contigua**
- Base conceptual para entender **paginación y segmentación** más adelante
- Promueve pensamiento algorítmico y comprensión de recursos limitados



2.9 Evaluación: Instrumentos y criterios



Criterios:

- Correcta implementación de asignación y liberación
- Manejo de estructura de memoria y visualización
- Claridad del código y representación del estado
- Participación activa y reflexión sobre los errores



Instrumentos:

- Rúbrica funcional
- Observación directa
- Defensa opcional del código
- Validación por casos de prueba

2.10 Inclusión y atención a la diversidad


- Proporcionar ejemplos guiados paso a paso
- Uso de código base con zonas completables
- Posibilidad de representar la memoria en papel si es más visual
- Acompañamiento individual para reforzar los conceptos clave
- Apoyo visual y verbal simultáneo

2.11 Actividades de ampliación

 Para el alumnado avanzado:

- Introducir **algoritmos de asignación**:
 - Primer ajuste
 - Mejor ajuste
- Añadir **compactación manual** tras liberaciones
- Simular carga de varios procesos con distintas necesidades
- Incluir lógica de paginación (simulada) o estadísticas de uso

Cierre

- ✓ Actividad útil para visualizar conceptos abstractos
 - ✓ Relaciona programación estructurada con sistemas reales
 - ✓ Fomenta análisis de errores, planificación de recursos y visualización de estado
-  ¡El sistema operativo no es magia... es lógica bien estructurada!