Estructuras de Datos y Algoritmos I

2° CUAT 2025 | JUEVES NOCHE

DOCENTES: ING. Muriel CUCCETTI | TEC. Darío LALANNE | Delfina GÓMEZ

PUNTO DE CONTROL

Descripción de la situación:

Un pequeño estudio de videojuegos quiere desarrollar un juego de cartas con personajes que luchan entre sí.

Consideraciones iniciales:

Cada personaje es representado en el programa como un **`struct Personaje`** con los siguientes campos:

- `int id` : Número único que identifica al personaje
- `char nombre[30]` Nombre del personaje (no puede repetirse).
- `int nivel` Comienza en 1.
- `int vida` Valor inicial 10.
- `int ataque` Valor inicial 1.
- `int defensa` Valor inicial 1.
- `int puntosMejora` Puntos disponibles para asignar a atributos.

Al iniciar el programa se pregunta cuántos personajes como máximo se podrán registrar.

Se reserva memoria dinámicamente para almacenar la cantidad de personajes indicada.

Funcionalidades del juego:

1. **Crear personaje**
    a. Verifica que no se supere el máximo de personajes (definido al inicio).
    b. Asigna id único automáticamente.
    c. Pide nombre (único: hay que validar que no exista).
    d. Inicializa los campos de atributos con valores predeterminados:
        i. vida = 10
        ii. ataque = 1
        iii. defensa = 1
        iv. nivel = 1
        v. puntosMejora = 10
    e. Permite distribuir los 10 puntos de mejora iniciales entre vida, ataque y defensa hasta que se agoten.

2. **Mejorar personaje**
    a. Busca un personaje por nombre.
    b. Si tiene puntosMejora > 0: Pregunta dónde asignarlos (vida / ataque / defensa).
    c. Actualiza valores.

3. **Luchar**
    a. Muestra listado de personajes (id, nombre, atributos).
    b. Se eligen dos personajes diferentes por id (Atacante y defensor).
    c. Se ejecuta la mecánica de combate

    `daño = ataque atacante – defensa defensor`

    Si queda en 0 o menos: Gana el defensor y recibe +2 puntos de mejora

    Si es mayor a cero: Gana el atacante y se le resta el daño a la vida del defensor. El atacante recibe +1 punto de mejora o +3 y sube de nivel si el defensor muere.

0. **Salir**

Se espera que el alumno organice el programa en funciones:

Algunas sugeridas (no es obligatorio que usen exactamente estos nombres, pero sí se deben cubrir las funcionalidades):

- `void crearPersonaje(Personaje* personajes, int* cantidad);`
- `void mejorarPersonaje(Personaje* personajes, int cantidad);`
- `void luchar(Personaje* personajes, int cantidad);`
- `Personaje* buscarPorNombre(Personaje* personajes, int cantidad, const char* nombre);`
- `Personaje* buscarPorId(Personaje* personajes, int cantidad, int id);`
- `void mostrarPersonaje(Personaje p);`
- `void mostrarTodos(Personaje* personajes, int cantidad);`

Importante:

- El programa debe manejar memoria dinámica con *malloc y free*
- Se debe utilizar punteros y aritmética de punteros para recorrer el vector de personajes.
- Validar todas las entradas del usuario (ejemplo: nombre repetido, puntos de mejora incorrectos, ids inválidos).

Análisis de escenarios:

Para cada uno de los siguientes escenarios indicar cómo se desarrolla el flujo de ejecución paso a paso teniendo en cuenta en cada caso:

- Instrucciones que se ejecutan (Ej: Llamado a función, estructura selectiva, asignación, operación aritmética) explicando brevemente lo que hace y qué resultado produce.

- Valor de las variables involucradas luego de ejecutar cada instrucción (Por ej: Contadores, campos de los struct Personaje involucrados)

Escenario A: Creación de Personaje

- Máximo de personajes: 3
- Se crea "Mario" (id = 1).
- Distribuye sus puntos: vida +5, ataque +3, defensa +2.

Escenario B: Mejora de Personaje

- Mario (id 1) tenía 2 puntos de mejora
- Decide (id 2) poner 2 puntos en defensa.

Escenario C: Lucha

- Atacante: Mario (id 1, ataque 4).
- Defensor: Luigi (id 2, defensa 6).

Escenario D: Lucha

- Atacante: Mario (id 1,ataque 8).
- Defensor: Luigi (id 2,defensa 3, vida 12).

Escenario E: Lucha

- Atacante: Mario (id 1, ataque 12).
- Defensor: Luigi (id 2, defensa 2, vida 10).

Pautas de entrega y evaluación:

Modalidad:

- Escrito e Individual con defensa
- Código desarrollado exclusivamente en lenguaje C

Condiciones para la entrega del punto de control:

- **Fecha límite: 01/10/2025 – 23:59 hs**. Luego de ese horario cierra la actividad y no se aceptan más entregas.
- El punto de control se entrega como tarea **por evea**. No se reciben entregas vía email, discord u otro medio diferente al propuesto por los docentes.
- El código deberá entregarse en **un único archivo .c** cuyo nombre respete el siguiente **formato:** *DNI_APELLIDO_NOMBRE.c | Ejemplo: 12345678_PEREZ_JUAN.c*
- El análisis de escenarios deberá entregarse en **un único archivo .pdf** cuyo nombre respete el siguiente **formato:** *DNI_APELLIDO_NOMBRE.pdf | Ejemplo: 12345678_PEREZ_JUAN.pdf*
- **No se reciben entregas en papel, u otro tipo de archivo**. El código debe estar en un archivo .c que se pueda compilar y ejecutar correctamente.

Criterios de evaluación

El punto de control tiene dos instancias:

1- **Entrega de código:**

Se evalúan los siguientes aspectos:

- **Entrega en tiempo y forma** (Excluyente)
- Código **sin errores** de compilación (Excluyente)
- **Funcionalidades desarrolladas correctamente**.
- **Uso de buenas prácticas:** Comentarios, nombres representativos de variables, indentación del código.

2- **Defensa: Se desarrollará en la clase del 02/10 - 18:00 hs (presencial en UPE)**

El punto de control se considera desaprobado en primera instancia si:

- No se cumplen con las condiciones de entrega.
- No compila o no están implementadas las funcionalidades mínimas que se piden.
- Se utilizan variables globales o sentencias “continue” o “break” (Esta última solo se permite en estructuras switch).
- Si dos alumnos entregan el mismo código. (El parcial es individual)
- Si se desarrolla con cualquier otro lenguaje que no sea ANSI C.