Tarea 2 - Estructuras de Datos (INF-134)

Publicación: Viernes 09 de Mayo
Entrega: Viernes 30 de Mayo - 23:55h vía Aula

2025-1

1 Reglas

- La tarea debe realizarse en grupos de 2 personas. Cualquier situación excepcional debe ser aprobada por el ayudante coordinador (Tomás Barros).
- Debe usarse el lenguaje de programación C++. Al realizar la evaluación, la tarea será compilada usando el compilador g++ en Linux, usando el comando:

  `g++ tarea2.cpp -o tarea2 -Wall`.

  No se aceptarán entregas realizadas en Windows o compiladores online como Dev-C++, Replit, GDB, etc. En caso de usar Mac, se recomienda usar XCode, y de ser posible, revisar la tarea también en Linux.
- No se permite usar la biblioteca STL, así como ninguno de los contenedores y algoritmos definidos en ella (e.g. vector, list, etc.). En general no se permite importar ninguna estructura de datos, salvo por string. Está permitido utilizar librerías para obtener números aleatorios como *random*
- Una tarea que no compile tendrá nota 0. En dicha situación, el/la estudiante puede apelar con el ayudante coordinador.
- La tarea puede entregarse con máximo 1 día de atraso, en cuyo caso la nota máxima será 50.
- Solo una de las tres tareas del semestre puede tener nota inferior a 30.
- Si se detecta **COPIA** o abuso de herramientas de generación automática de código (ChatGPT), la situación será revisada por el ayudante coordinador y profesor coordinador. Si se considera que el estudiante no entiende el código que escribió, tendrá nota 0 en la tarea.

2 Problema a resolver

Deberán programar un videojuego de aventura en el cual la protagonista viaja de *habitación* en *habitación* resolviendo *combates* o *eventos*. Se tiene la siguiente lista de requisitos que el juego debe cumplir, que, podemos imaginar, fue entregada por un cliente tremendamente emocionado por su revolucionaria y original idea:

1. Cada habitación tiene que tener una pequeña descripción para que el jugador se introduzca en el mundo.

2. El jugador debe comenzar en una habitación introductoria, y de ahí elegir la siguiente habitación entre algunas opciones.

3. En cada habitación debe haber un combate o un evento.

4. Los combates se realizan automáticamente[1], entre el jugador y los enemigos presentes en la habitación, de acuerdo las siguientes reglas:

   (a) El jugador hace daño igual a su valor de ataque, al primer enemigo, con probabilidad igual a su valor de precisión. Es decir, hay una probabilidad (1 - precisión) de que el ataque no haga daño.

   (b) Cada enemigo hace daño igual a su valor de ataque, siguiendo el mismo esquema de precisión que el jugador.

   Esto se repite hasta que el jugador haya eliminado a todos los monstruos, o haya sido eliminado (fin del juego).

5. Después de cada combate, el jugador debe poder escoger una mejora para algunas de sus estadísticas.

6. Por otro lado, los eventos presentan una descripción breve junto a una decisión.

7. Después de cada combate o evento, el jugador debe recuperar un poco de vida. Este valor debe ser variable, para ser modificado por eventos y tras combates.

8. Debe haber por lo menos tres tipos de enemigos distintos, y cada combate debe comenzar seleccionando aleatoriamente una cantidad de ellos. Ej: En la habitación 1 puede haber un enemigo A, y en la habitación 2 pueden haber dos enemigos B y C. Enemigos más fuertes deben tener una probabilidad menor de aparecer.

9. Debe haber por lo menos tres eventos distintos y estos también deben tener distintas probabilidades de aparecer.

10. El mapa debe tener múltiples habitaciones finales, que presentan diferentes cierres de la historia.

11. La altura del mapa debe ser 5 como mínimo (altura: Camino más largo de raíz a hoja).

[1] sin input de usuario

La figura 1 muestra un ejemplo de mapa. El nodo inicial, se muestra en verde en la parte inferior del mapa. Los nodos finales se muestran en amarillo. En este caso se consideran 8 finales diferentes. Los nodos en rojo representan habitaciones con combates y los nodos en azul habitaciones con eventos. La altura de este mapa es cinco.

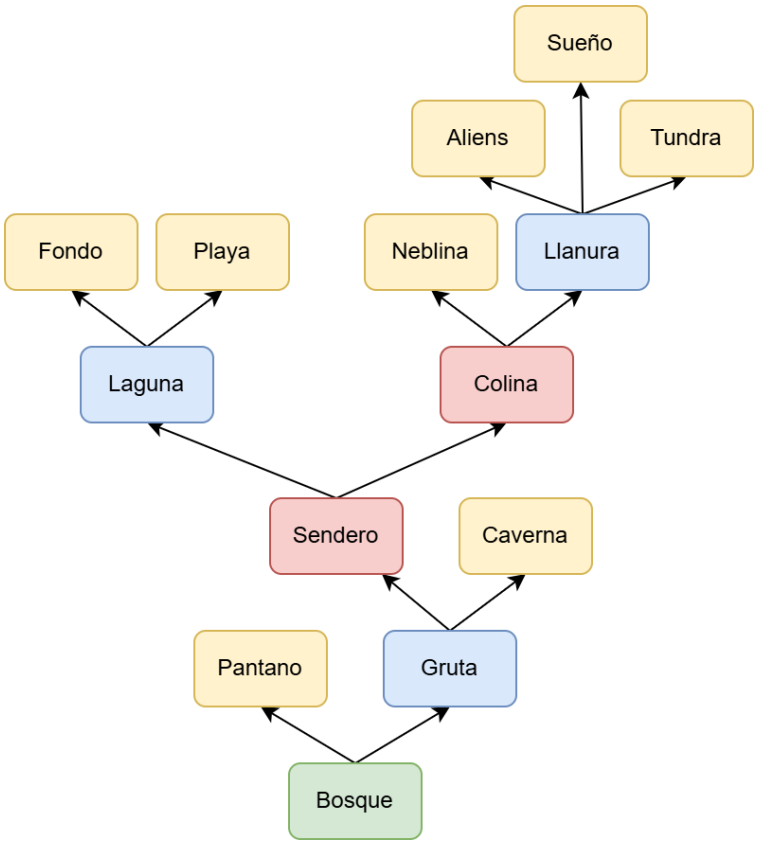

Figura 1: Visualización de un mapa. Nodos de combate marcados en rojo, nodos de evento marcados en azul y nodos finales marcados en amarillo. Notar que el árbol es ternario.

2.1 Requisito Especial (10 pts BONUS)

Como desarrolladores, pueden aportar una mecánica nueva que no haya sido previamente mencionada para obtener 10 puntos adicionales. Dicha mecánica debe incorporar una de las siguientes modificaciones del árbol de mapa:

1. Agregar un nodo durante la ejecución.
2. Agregar un camino durante la ejecución.
3. Retornar a un nodo pasado que no sea padre inmediato del nodo actual.

La nueva mecánica debe justificarse narrativamente, y dicha justificación, junto a la explicación de la mecánica, deben incluirse en el README de la tarea. Se ofrecen los siguientes ejemplos de mecánica, si bien se les invita a adaptarlas a sus narrativas:

1. Con probabilidad baja aparece un duende que les da la elección de viajar con él a una habitación secreta, ofreciendo un evento especial. (Implementado agregando un nodo durante la ejecución del programa).

2. Al introducir una clave secreta a la hora de seleccionar la siguiente habitación, se abre un pasadizo que lleva directamente a una habitación final del juego. (Implementado agregando un camino durante la ejecución).

3. Se define una habitación especial que permite viajar en el tiempo, retornando al nodo abuelo: Padre más izquierdo del padre del nodo. (Implementado retornando a un nodo pasado que no sea padre inmediato. Ojo que dicha implementación debe ser dinámica sobre los caminos del árbol, es decir, no es válido hardcodear que el jugador vuelve al nodo inicial, por ejemplo).

3 Detalles de implementación

Si bien el cliente entregó la lista anterior de requisitos del juego, se tiene, además una lista de requisitos técnicos que se deben seguir para cumplir con los estándares de la industria.

1. El mapa de habitaciones debe ser implementado con un TDA Árbol **ternario**[2] donde cada habitación no-final tiene como mínimo dos habitaciones siguientes.

2. El combate debe ser implementado usando un TDA Cola basado en listas enlazadas.

3. Los atributos del Jugador deben ser, como mínimo, los siguientes:

```
class Jugador{
private:
    int vida;
    int ataque;
    float precision;
    int recuperacion;
}
```

4 Notas adicionales

- Toda la memoria debe ser solicitada de manera dinámica, y debe ser liberada correctamente al finalizar el programa. Generalmente, la solicitud de memoria será en función de los contenidos del archivo `data.map`.

- Deben entregar un archivo data.map con su propia historia además de archivo del ejemplo adjunto. Su programa debe ser **robusto** a pequeñas modificaciones de dicho archivo. Debe ser factible agregar una habitación, cambiar una conexión entre dos habitaciones, cambiar el orden de las habitaciones en el archivo, etc.

- El *main* de su programa debe cargar el archivo `.map` y ejecutar el juego.

- Todo input del programa se recibirá por entrada estándar (terminal).

[2]Hint: Binario: 2 punteros, Ternario...

5 Formato de Entrega

Se debe entregar un archivo llamado `tarea2-apellido1-apellido2.zip`[3] que contenga, por lo menos:

1. El archivo que contiene su código fuente: `tarea2.cpp`.
2. El archivo `data.map` con su juego, además del archivo `ejemplo.map` adjunto en la tarea.
3. `README.txt` con los nombres y roles de los integrantes, además de un apartado con instrucciones de compilación y ejecución en caso de ser distintas a las esperadas. **Este archivo también deberá contener una explicación del formato de su archivo `data.map` y la justificación para optar a los 10 pts del requisito especial, en caso de haberlo implementado.**

Para esta tarea, pueden además incluir todos los archivos que consideren necesarios para la organización de su entrega (Archivos relacionados a estructuras de datos, por ejemplo).

Recordar que no seguir el formato de entrega conlleva hasta 10 puntos de descuento. *Favor adjuntar roles*. La compilación de su tarea es imperativa para la evaluación. **Una tarea que no compile tendrá nota 0.**

Recordar que es su responsabilidad asegurarse de que se adjunten los archivos correctos a la entrega.

6 Directrices de programación

Las directrices corresponden a buenas prácticas de programación, las cuales serán tomadas en consideración para la evaluación de la tarea. Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota. Un código entregado completamente sin comentarios, tendrá **nota 0**.

Cada función de código cuya función no sea inmediatamente aparente, debe tener un comentario de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

Además,

[3] 1 apellido por cada integrante. Se debe entregar el *primer* apellido.

- Se deben utilizar nombres de variables descriptivos.
- Se deben seguir buenas prácticas respecto al uso de funciones. Es decir, se deben utilizar funciones para evitar redundancia y para modularizar las componentes de su código.
- Se debe considerar la indentación correcta del código.

7 Ejemplos

Ejemplo de inicio

A continuación se presenta un ejemplo del inicio del juego.

```
-- Habitación Inicial --
Te adentras en un bosque donde nada es lo que parece. A tu izquierda
   ves una gruta cuyo interior emite un tibio fulgor. A tu derecha
   encuentras un sendero iluminado por la esperanza de un mejor andar.
    En la lejanía, una laguna sin orilla.

A donde quieres ir?
1. Gruta
2. Sendero
3. Laguna
(Presiona la tecla correspondiente)
```

Ejemplo de Combate

A continuación se presenta un ejemplo del combate. En la habitación se encontraron dos enemigos, obtenidos aleatoriamente de un conjunto predefinido. Notar el ciclo de combate: Jugador golpea al primer enemigo, luego cada enemigo golpea a Jugador.

```
Entras a una habitación y encuentras a dos monstruos: PIM y POMPLIN!
Jugador | Pim | Pomplin
30 | 10 | 12

Jugador golpea a Pim por 7 de daño!

Jugador | Pim | Pomplin
30 | 3 | 12

Pim golpea a Jugador por 2 de daño!

Jugador | Pim | Pomplin
28 | 3 | 12

Pomplin golpea a Jugador por 1 de daño!

Jugador | Pim | Pomplin
27 | 3 | 12
```

```
Jugador golpea a Pim por 7 de daño!

Jugador | Pim | Pomplin
27 | X | 12

Pomplin falla!

Jugador | Pim | Pomplin
27 | X | 12

Jugador golpea a Pomplin por 7 de daño!

Jugador | Pim | Pomplin
27 | X | 5

Pomplin golpea a Jugador por 1 de daño!

Jugador | Pim | Pomplin
26 | X | 5

Jugador golpea a Pomplin por 7 de daño!

Jugador | Pim | Pomplin
26 | X | X

Has sobrevivido el combate!
Recuperas 3 de vida tras el combate.
Debes decidir:
    1. Recuperar 5 de vida.
    2. Aumentar ataque en 1.
*Usuario inputa 1*
Recuperaste 5 de vida!
Ahora, te encuentras en un campo...
.
.
.
```

Tras sobrevivir el combate, el Jugador recupera vida segun su estadística de recuperación, y luego puede tomar una decisión menor para mejorar a su personaje. A continuación, se le da la elección de la próxima habitación para visitar.

Ejemplo de archivo .map

Nótese que un programa podría elaborar todas las habitaciones, como también las conexiones entre ellas, solo a partir de la información contenida en este archivo.

```
INICIO DE ARCHIVO
HABITACIONES
14
0 Bosque (INICIO)
Te adentras en un bosque donde nada es lo que parece
1 Gruta (EVENTO)
Investigas la gruta y te encuentras en una situacíón peculiar.
```

```
2 Sendero (COMBATE)
Caminas por el sendero, cuando se te acerca un grupo de sospechosos!
3 Laguna (EVENTO)
A la orilla de la laguna encuentras una gema...
4 Colina (COMBATE)
Subes la colina y te encuentras en una batalla!
5 Llanura (EVENTO)
Caminando por la llanura, cansado de tus viajes, notas algo extraño...
6 Pantano (FIN)
Te encuentras en un pantano. El fango se pega a tus pies, y te da
   tanto asco que mueres :(
7 Caverna (FIN)
Encuentras una apertura diminuta y oscura. Cualquier persona razonable
    se mantendría alejada, pero has visto muchos tiktok de exploración
    de cavernas. Te quedas atascado, y despues de una semana te mueres
    por abstinencia de redes sociales.
8 Fondo(FIN)
Se te olvida que no eres un pez y te adentras al fondo de la laguna.
   Te encuentras a un pez que se le olvidó que no es una persona. Sale
    al exterior y te reemplaza en la selección chilena. Quedas en el
   olvido, mientras un pez gana el mundial.
9 Playa (FIN)
La playa está tan cómoda que te acuestas de chill. Te quedas dormido,
   y despiertas ayer. *Vuelves al nodo ante-anterior!* (Este es un
   ejemplo del requisito especial 3)
10 Neblina (FIN)
Caminas en subida hasta alcanzar las nubes. Estas mirando el atardecer
    cuando recuerdas que dejaste el horno prendido! Aún peor, notas
   que lo que pensabas que era el sol era realmente tu casa a lo lejos
   ...
11 Tundra (FIN)
Qué es una tundra? Mueres.
12 Aliens (FIN)
Te recoge una nave alienígena. Te preguntan tu nombre y por mera
   casualidad es una palabra que hace cortocircuito en su cerebro!
   Salvaste a la humanidad :D
13 Suenho (FIN)
Te quedas dormido y sueñas con una tarea de estructuras de datos donde
    tienes que hacer un juego. Llegas a la parte de escribir la
   historia y no se te ocurre mejor final que uno en el que el jugador
    se queda dormido y sueña con una tarea de estructuras de datos
   donde tienes que hacer un juego. Llegas a la parte de escribir la
   historia...
ARCOS
13
0 -> 1
0 -> 6
1 -> 2
1 -> 7
2 -> 3
2 -> 4
3 -> 8
3 -> 9
4 -> 10
```

```
4 -> 5
5 -> 11
5 -> 12
5 -> 13
ENEMIGOS
4
Pim | Vida 10 | Ataque 2 | Precision 0.8 | Probabilidad 0.2
Pomplin | Vida 12 | Ataque 7 | Precision 0.3 | Probabilidad 0.4
Tomp | Vida 30 | Ataque 100 | Precision 0.001 | Probabilidad 0.1
Momplin | Vida 1 | Ataque 5 | Precision 1 | Probabilidad 0.3
EVENTOS
3
&
Tesoro
Probabilidad 0.3
Encuentras un tesoro!...
A: Abrirlo
Era una mímica! El terror afecta tu psique.
-0.2 PRECISION
B: Dejarlo
Se ve sus, lo dejas pasar.
Ninguna consecuencia.
&
Sentimiento
Probabilidad 0.3
Te empiezas a sentir extraño...
A: Te acuestas en el suelo
Te despiertas muy relajado y feliz!
+10 Vida
B: Tratas de continuar
Te mareas, te tropiezas, y te esguinzas la articulación metacarpofalá
   ngica!
-5 Vida
&
Silueta
Probabilidad 0.4
Ves una silueta misteriosa a lo lejos.
A: La tratas de dibujar.
Logras un dibujo excelente, impresionando a todos tus amigos!
+7 Vida
B: Lo tratas de identificar.
Tras horas de forzar la vista logras entender que era una mancha en el
    cielo simplemente. Tu vista se agotó.
-0.3 Precision
MEJORAS DE COMBATE
+4 Vida
+0.1 Precision
+1 Ataque
+1 Recuperacion
FIN DE ARCHIVO
```

Cada apartado del archivo comienza con la cantidad de entradas que posee (por ejemplo, hay 14 habitaciones). Cada apartado tiene su formato que debiese ser aparente, si bien cabe aclarar el más complejo: El apartado de EVENTOS comienza con la cantidad de eventos a

registrar. Luego, cada evento está separado por un &. A continuación, cada evento tiene un nombre, probabilidad, y una serie de opciones. Finalmente, cada una de estas opciones tiene un enunciado, descripción y consecuencia.

El archivo `ejemplo.map` se adjunta también con el PDF de la tarea. Se recomienda realizar toda la tarea con este archivo, para luego finalmente crear su propia historia usando el mismo formato. Recordar que su propia historia puede ser todo lo creativa que usted desee, pero es fundamental que sea respetuosa para los ayudantes y profesores.

8 Contacto

Profesores

- Elizabeth Montero (**Coordinadora**) (elizabeth.montero@usm.cl)
- Alondra Rojas (alondra.rojas@usm.cl)
- José Miguel Cazorla (jcazorla@usm.com)
- Ricardo Salas (ricardo.salas@usm)
- Juan Pablo Castillo (juan.castillog@sansano.usm.cl)
- Roberto Díaz (roberto.diazu@usm.cl)

Ayudantes (CC)

- Tomás Barros (**Coordinador**) (tomas.barros@sansano.usm.cl)
- Jaime Inostroza (jinostroza@usm.cl)
- Facundo Riquelme Lucero (friquelmel@usm.cl)
- Franco Cerca (franco.cerda@usm.cl)
- Bastián Jimenez (bjimenez@usm.cl)

Ayudantes (SJ)

- Juan Alegría (juan.alegria@usm.cl)
- Damián Rojas (damian.rojas@usm.cl)
- Ignacia Nettle Oliveri (inettle@usm.cl)
- Alvaro Aceituno Alarcon (alvaro.aceitunoa@usm.cl)
- Lucas Morrison (lucas.morrison@sansano.usm.cl)
- Javier Matamala Gonzalez (javier.matamalag@usm.cl)