INF-253 Lenguajes de Programación

Profesores: José Luis Martí Lara, Wladimir Ormazabal Orellana
Ayudante coordinador: Bastián Ríos Lastarria
Ayudantes: Fernanda Araya, Diego Duarte, Andrés Jablonca, Cristian Tapia

7 de abril de 2025

1. McSansano: Cocina Frenética

Tu anterior jefe en Soluciones Sansaniles te ha despedido en un brusco cambio de ánimo, por lo que, para poder llegar a fin de mes, has terminado trabajando en McSansano, un caótico local de comida rápida. Para evitar el despido (¡otra vez!), debes crear un programa en C que gestione los pedidos, recolecte ingredientes y los cocine en estaciones de trabajo para así poder entregar el mejor servicio. El local está al borde del colapso: órdenes se acumulan, ingredientes se mezclan mal, y los clientes se van furiosos. ¡Usa tus habilidades para salvar el negocio!

2. Descripción general

El jugador controla un único chef que se mueve por un tablero de tamaño configurable a través de un sistema por turnos. En la cocina aparece el chef, junto a cuatro estaciones de trabajo (tabla de cortar, cocina, almacén, extintor) de manera aleatoria esparcidas por el tablero, en las estaciones se pueden cortar, cocinar o buscar ingredientes según corresponda.
Estas estaciones pueden aparecer en cualquier parte del tablero de forma aleatoria, siempre debe haber 1 de cada tipo.

2.1. Descripción general de las estaciones y el inventario del chef:

- **Tabla de cortar:** En la tabla, el jugador puede cortar ingredientes obtenidos, costándole una cantidad de turnos determinada por el ingrediente.
- **Almacén:** Dentro del almacén, el jugador puede encontrar ingredientes (como pan, lechuga, tomate, arroz, pasta, hamburguesa, pollo, papa...) de forma aleatoria. Cada intento de búsqueda cuesta un turno, y al obtener un ingrediente se puede optar por devolverlo o agregarlo al inventario (si hay espacio disponible).
- **Cocina:** En la cocina, el jugador puede cocinar los ingredientes obtenidos costándole una cantidad determinada por el ingrediente de turnos,**y hay una probabilidad de causar un incendio al usar la cocina dependiendo del ingrediente que**

se esté cocinando.
En caso de incendio, el jugador se debe reposicionar a un lugar seguro.

- **Extintor:** El extintor se usa para apagar incendios y debe ser recogido previamente desde la estación correspondiente.
  En caso de incendio, el ingrediente que se estuviera cocinando en la estación se pierde.
  El extintor solo puede ser recogido con el inventario vacío.

- **Inventario:** El chef dispone de un inventario estático (con espacio limitado a 5 elementos) para recoger y almacenar ingredientes, y ocasionalmente el extintor. **El inventario no puede contener otros elementos si se transporta el extintor.**

Mecánicas del Juego

1. **Selección de Dificultad**
   - **Fácil:** 5x5, 60 turnos, 3 pedidos.
   - **Medio:** 8x8, 50 turnos, 4 pedidos.
   - **Difícil:** 10x10, 45 turnos, 5 pedidos.
2. **Movimiento:**
   - El jugador ingresa una dirección junto con la cantidad de espacios a moverse, el input puede ser seleccionando la opción correspondiente (1 para moverse, por ejemplo) para luego indicar dirección, o directo de la siguiente manera:

     - W: Avanzar verticalmente hacia arriba
     - A: Avanzar horizontalmente hacia la izquierda
     - S: Avanzar verticalmente hacia abajo
     - D: Avanzar horizontalmente hacia la derecha

     Luego de seleccionar la dirección se le pide el input de un número, que corresponderá a la cantidad de unidades que avanzará. Ejemplo:

```
1  Tablero 5x5              (luego de "W" y "3")
2  Turnos restantes: 40      Turnos restantes: 39
3  [T][ ][ ][ ][E]            [V][ ][ ][ ][E]
4  [ ][ ][C][ ][ ]            [ ][ ][C][ ][ ]
5  [ ][ ][ ][ ][ ] -------> [ ][ ][ ][ ][ ]
6  [O][ ][ ][ ][ ]            [ ][ ][ ][ ][ ]
7  [ ][ ][ ][ ][A]            [ ][ ][ ][ ][A]
8  input: W
9  Cuantas unidades desea avanzar?
10 input: 3
```

     En el primer tablero, T representa la tabla de cortar, O es el chef, E es el extintor y A es el almacén, **en el segundo caso, la 'O' cambia por una 'V' para señalar la nueva posición del chef**, ya que el chef se encuentra ahora en la tabla de cortar.

Como pueden ver, el tablero presente es enumerado de abajo hacia arriba y de izquierda a derecha, pueden buscar alternativas a esto mientras la representación gráfica del tablero sea legible y correcta como se muestra en el ejemplo.

- Todos los movimientos costarán 1 turno independiente de la distancia recorrida.
- El jugador no puede atravesar una estación de cocina en llamas, en caso de que lo intente, se prenderá en llamas y perderá 3 turnos.

3. **Acciones:**
   - **Buscar ingrediente:** Si el chef se encuentra en el almacén, puede buscar un ingrediente al azar. Se le mostrará qué ingrediente encontró y se le ofrece la opción de devolverlo o agregarlo al inventario.
   - **Preparar ingrediente:** Si el chef se ubica en una estación de cocina o tabla de cortar, se invoca una función asociada acorde a esa estación (usando punteros a función) que permite procesar el ingrediente. Existe probabilidad de incendio al usar la cocina.
4. **Ver Inventario:** Muestra el contenido actual del inventario, que corresponde a una lista estática donde se almacenan los ingredientes recolectados o el extintor. Además, ofrece la opción de descartar cualquier ingrediente o el extintor para liberar espacio en caso de ser necesario. El espacio del inventario siempre será de 5 espacios.
5. **Entregar plato:** El jugador debe poseer en su inventario los ingredientes que solicita el pedido, y seleccionarlos desde su inventario para formar el plato requerido por la orden actual. Solo se permite la entrega si todos los ingredientes seleccionados están listos y disponibles.

2.2. Archivos requeridos

La tarea debe contar con los siguientes 8 archivos, estos son:

- **acciones.h:** Declaraciones de Estacion, Pedido y funciones.
- **acciones.c:** Implementación de las funciones de acción y lógica de incendios.
- **inventario.h:** Declaración de Ingrediente y funciones para gestionar el inventario.
- **inventario.c:** Implementación de agregar_ingrediente(), eliminar_ingrediente(), etc.
- **tablero.h:** Declaración de funciones para manejar el tablero (inicializarTablero(), etc).
- **tablero.c:** Implementación de las funciones del tablero.
- **main.h:** La estructura `Juego y Jugador` y la configuración global.
- **main.c** Lógica principal y ciclo de juego.

2.3. Sobre los pedidos e ingredientes

Los pedidos deben ser elegidos de manera aleatoria cada partida de la lista de posibles pedidos, se pueden repetir si así se desea.
Aquí hay una lista de ingredientes y pedidos a usar, siéntanse libres de crear sus propias recetas y agregar cuántos ingredientes quieran, esto es solo la base!
Adicional a esto, las probabilidades de que aparezca cada ingrediente dentro del almacén se deja a decisión de ustedes, idealmente cada ingrediente debe tener la misma probabilidad de ser encontrado en el almacén pero si sus recetas son dependientes en su gran mayoria de un ingrediente en específico, son libres de aumentar las chances de éste.

- Ingredientes:
  - Papa:
    - posibles estados: crudo, cortado, papas fritas (cortadas y fritas)
    - turnos en cortar: 1
    - turnos en freir: 4
    - probabilidad de quemar la cocina: 2 %
  - Pollo:
    - posibles estados: crudo, frito
    - turnos en cocinar: 5
    - probabilidad de quemar la cocina: 5 %
  - Hamburguesa:
    - posibles estados: crudo, cocinado
    - turnos en cocinar: 3
    - probabilidad de quemar la cocina: 3 %
  - Arroz:
    - posibles estados: crudo, cocinado
    - turnos en cocinar: 2
    - probabilidad de quemar la cocina: 1 %
  - Pasta:
    - posibles estados: crudo, cocinado (olla)
    - turnos en cocinar: 2
    - probabilidad de quemar la cocina: 1 %
  - Tomate:
    - posibles estados: crudo, cortado
    - turnos en cortar: 1
  - Lechuga:
    - posibles estados: crudo, cortado
    - turnos en cocinar: 1
  - Pan: no es necesario prepararlo
- Pedidos:
  - 'el McCharly': [papa frita,pollo frito,papa frita]
  - 'Pasta a la Sansini': [pasta, tomate]
  - 'El Certamen': [pan,lechuga,tomate,hamburguesa]
  - 'El Clasico': [arroz,pollo frito]

3. Flujo del juego y ejemplos:

El programa inicia preguntándole al usuario qué tamaño de tablero desea jugar con un mensaje de "Selecciona la dificultad". Luego de escoger, el juego generará el tablero, comenzando a colocar las 4 estaciones de trabajo y al jugador en lugares al azar Con esto listo, parte el primer turno del juego.
A continuación se muestra un ejemplo gráfico de como se vería esto a través de la terminal.

```
Seleccione dificultad:
1.Facil
```

```
2.Medio
3.Dificil
>1
pedido en curso! prepara 'el McCharly' y otros 2 pedidos mas en: 60 turnos.
'el McCharly': [papa, pollo, papa]. Turnos restantes: 60
A . O . .
. . . . .
. . . T .
. . . . .
C . E . .
1. Moverse (W, A, S, D)
2. Accion
3. Ver inventario
4. Entregar plato
5. Salir del juego
>A
Ingresa la cantidad de espacios: 2
'el McCharly': [papa, pollo, papa]. Turnos restantes: 59
V . . . .
. . . . .
. . . T .
. . . . .
C . E . .
1. Moverse (W, A, S, D)
2. Accion
3. Ver inventario
4. Entregar plato
5. Salir del juego
>2
Has encontrado pollo! Deseas guardarlo en el inventario?(1.Si/2.No) >1
pollo ha sido almacenado en tu inventario. 'el McCharly': [papa, pollo, papa]. Turnos restantes: 58
V . . . .
. . . . .
. . . T .
. . . . .
C . E . .
1. Moverse (W, A, S, D)
2. Accion
3. Ver inventario
4. Entregar plato
5. Salir del juego
>2
Has encontrado hamburguesa! Deseas guardarlo en el inventario?(1.Si/2.No) >1
hamburguesa ha sido almacenado en tu inventario. 'el McCharly': [papa, pollo, papa]. Turnos restantes: 57
V . . . .
. . . . .
. . . T .
. . . . .
C . E . .
1. Moverse (W, A, S, D)
2. Accion
```

```
3. Ver inventario
4. Entregar plato
5. Salir del juego
>S
Ingresa la cantidad de espacios: 4
'el McCharly': [papa, pollo, papa]. Turnos restantes: 56
A . . . .
. . . . .
. . . T .
. . . . .
V . E . .
1. Moverse (W, A, S, D)
2. Accion
3. Ver inventario
4. Entregar plato
5. Salir del juego
>3
   pollo, hamburguesa
2/5 espacios utilizados.
1. Moverse (W, A, S, D)
2. Accion
3. Ver inventario
4. Entregar plato
5. Salir del juego
>2
Has consumido 5 turnos! has obtenido [pollo frito]. Deseas seguir cocinando?(1.Si/2.No) >2
'el McCharly': [papa, pollo, papa]. Turnos restantes: 51
A . . . .
. . . . .
. . . T .
. . . . .
V . E . .
1. Moverse (W, A, S, D)
2. Accion
3. Ver inventario
4. Entregar plato
5. Salir del juego
>6
```

Esta es una ejecucion simple de la idea de ciclo de juego. Se recomienda usar este mismo formato o uno similar para su tarea, en caso de usar uno distinto, procure que el jugador pueda ver la información relevante del juego para poder tomar sus decisiones, como el estado del tablero, cuándo debe colocar un input y qué opciones puede colocar, etc.

Funciones y structs a implementar

Adicional a este documento, habrá un pdf con structs y funciones de carácter **obligatorio**, si estas funciones y structs no están al menos **definidas** en sus respectivos archivos .h, **la tarea no será revisada**. Son libres de crear otros Structs y funciones adicionales a las mencionadas en el pdf, pero **no pueden eliminar ni editar los existentes a menos que se encuentre algun error que impida su uso correcto**. Su uso también es de carácter obligatorio, sobre todo las funciones en `acciones.h` y el struct de `tablero.h`, ya que en estas se basa gran parte

del puntaje asociado a la tarea.

4. Sobre la entrega

- El código debe venir indentado y ordenado. De no existir orden, se realizará descuento.
- Se debe entregar los siguientes archivos:
  - Makefile
  - README
  - main.h, main.c, tablero.h, tablero.c, inventario.h, inventario.c, acciones.h, acciones.c

  Donde main.c debe contener la función main, los archivos .h debe contener las declaraciones de las estructuras, funciones, variables y definiciones. Y los archivos .c debe contener todas las implementaciones de las funciones de su header correspondiente.
- Se puede crear nuevas funciones, estructuras y/o definiciones que estime conveniente. Como también agregar nuevos atributos a las estructuras existentes.
- La falta de declaración de los structs y funciones descritos en el documento 'Requerimientos_Tarea_C.pdf' en sus respectivos archivos de encabezado (.h) resultará en una calificación de 0 en la nota final de la tarea.
- Es de **uso obligatorio la estructura Tablero y sus atributos para la entrega mínima**, de lo contrario significará un 0 en la nota final de la tarea.
- No se permite cambiar las firmas, parámetros y retorno, como también agregar nuevos parámetros en las funciones mencionadas en el documento siguiente a éste.
- Las funciones deben ir comentadas, explicando clara y brevemente lo que realiza, los parámetros que recibe y los que devuelve (en caso de que devuelva algo). Se deja en libertad al formato del comentario.
- Debe estar presente el archivo **MAKEFILE** para que se efectúe la revisión, este debe compilar TODOS los archivos mediante compilación separada.
- Se utilizará Valgrind para detectar las fugas de memoria.
- El trabajo es individual obligatoriamente.
- La entrega debe realizarse en un archivo comprimido en tar.gz y debe llevar el nombre: Tarea2LP_RolAlumno.tar.gz.
  Ej.: Tarea2LP_202373000-k.tar.gz.
- El archivo **README.txt** debe contener nombre y rol del alumno, junto a instrucciones detalladas para la correcta compilación y ejecucion del programa.
- La entrega es vía aula y el plazo máximo de entrega es hasta el **23 de abril a las 23:55 vía aula.**
- Las consultas se deben realizar mediante el foro de la tarea disponible en AULA.
- Solo se responderán consultas hasta el dia 21 de abril 23:50 horas. Cualquier consulta enviada con hora posterior a la mencionada no será respondida.
- Por cada día (o fracción) de atraso se descontarán 20 puntos. 10 puntos dentro de la primera hora.
- Las copias serán evaluadas con nota 0 y se informará a las respectivas autoridades.

5. Clasificación

Para la clasificación de su tarea, se debe realizar una entre con requerimientos mínimos que otorgarán hasta 30 pts base. Luego se le entregará puntaje dependiendo de los otros requerimientos que llegue a cumplir.

5.1. Entrega mínima

Para obtener el puntaje mínimo en la entrega, el programa de cumplir con los siguientes requerimientos:

- Inicializa el tablero según la dificultad seleccionada fácil, asignando espacio en la memoria *heap* para la matriz **void*** celdas** dentro del struct Tablero. La inicialización debe configurar solo el tamaño mínimo pedido, es decir 5x5. (10 pts)
- El tablero se debe inicializarse y borrarse a través de **malloc** y **free**, no puede haber *leaks* de memoria. (10 pts)
- El programa genera a las estaciones y al jugador dentro del tablero en posiciones aleatorias. (10 pts)

5.2. Entrega

Luego de cumplir con la Entrega Mínima, puede obtener más puntaje cumpliendo con los siguientes puntos (puede haber puntaje parcial por cada punto):

5.2.1. General. (10 puntos)

- Utiliza los Header (archivo .h) para declarar las funciones, estructuras, variables y definiciones correspondientes. (5 puntos)
- El programa permite seleccionar las dificultades Fácil, Medio y Difícil, creando el tablero, las estaciones, el chef y el límite de turno correspondiente. (5 puntos)

5.2.2. Implementación de acciones mediante punteros a funciones. (30 puntos)

- Definición y uso de punteros a funciones para acciones en estaciones y su invocación correcta según la estación donde se encuentra el jugador. (*cortar(), cocinar(), buscar(), entregar_pedido() y apagar_incendio()*). (5 puntos c/u) (25 total)
- Lógica de probabilidades de incendio y reposicionamiento seguro. (5 pts)

5.2.3. Manejo del inventario y lógica principal del juego. (30 puntos)

- Movimiento del chef (entrar en estaciones, costo de 1 turno, bloqueo en estaciones incendiadas, actualizar celda). (10 puntos)
- Gestión del inventario (agregar/eliminar ingredientes, extintor). (5 puntos)
- Procesamiento de ingredientes (cortar: turnos según tipo, cocinar: riesgo de incendio). (5 puntos)
- Generación aleatoria de estaciones y posiciones iniciales. (5 puntos)
- Entrega de pedidos (validación de ingredientes, actualización de órdenes completadas). (5 puntos)

6. Descuentos

- Falta de declaraciones en archivos .h (-100 pts)
- Representación gráfica del tablero inconsistente o confusa (-10 puntos)
- Porcentaje de leak de memoria (bytes asignados) ((1 - 5) % -3 pts, (6 - 50) % -15 pts, (51 - 90) % -30 pts, (91 - 100) % - 40 pts))
- Falta de orden (Max -20 pts)
- Código no compila (-100 pts)
- Compilación Unificada (-5 pts)
- Warnings (-5 pts c/u)
- Falta de Makefile (-100 pts)
- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Mal nombre en algún archivo entregado o información errónea (-5 pts c/u)
- Día de atraso (-20 pts por día (o fracción), -10 pts dentro de la primera hora)
- En caso de existir nota negativa, esta será remplazada por un 0.