Fundamendez - 2C 2026 - Práctica semana 4

Laboratorio juego

Milhouse tiene un desastre en su cuarto. Su madre le dijo que, hasta que no ordene un poco, no puede salir a jugar con Bart.

Bart le dijo que lo va a esperar, pero no por mucho tiempo.

El objetivo del juego es que Milhouse junte toda la ropa que está tirada en su cuarto antes de quedarse sin movimientos. Milhouse comienza con 100 movimientos disponibles. Cada vez que realiza un movimiento, pierde uno de ellos.

Además de ropa, también hay juguetes alrededor del cuarto. Si Milhouse se para sobre uno se distrae jugando y pierde 5 movimientos adicionales.

Se tiene el siguiente struct para representar el juego:

```
typedef struct coordenada {
 int fil;
 int col;
} coordenada_t;

typedef struct jugador {
 coordenada_t posicion;
 int movimientos_restantes;
} jugador_t;

typedef struct juego {
 jugador_t jugador;
 coordenada_t ropa[MAX_ROPA];
 int tope_ropa;
 coordenada_t juguetes[MAX_JUGUETES];
 int tope_jueguetes;
} juego_t;
```

a. Crear un procedimiento que inicialice el juego teniendo en cuenta que:
   - Hay 10 objetos de tipo ropa y 5 objetos de tipo juguete.
   - Tanto la ropa como los juguetes deben ubicarse **aleatoriamente** en el cuarto.

- Milhouse comienza con **100 movimientos disponibles** y en la posición 0,0

Para pensar: ¿cómo podemos hacer la inicialización para asegurar que los elementos no se pisen?

b. Crear un procedimiento que reciba un jugador y un movimiento, y actualice su posición según corresponda.
Los movimientos posibles son:
   - (A): izquierda
   - (S): abajo
   - (D): derecha
   - (W): arriba

   El cuarto tiene un tamaño de **20 × 20**, por lo que Milhouse no puede salir de sus límites. Si un movimiento intenta llevarlo fuera del cuarto, su posición debe permanecer sin cambios.

   Cada movimiento válido realizado por Milhouse consume 1 movimiento disponible.

c. Crear un procedimiento que reciba el juego y determine si Milhouse se encuentra sobre algún objeto.
   - Si Milhouse se encuentra sobre una **prenda de ropa**, debe recogerla. La ropa deja de estar disponible en el terreno.
   - Si Milhouse se encuentra sobre un **juguete**, se distrae jugando y pierde **5 movimientos adicionales**.
   - Una vez recogida una prenda, no debe volver a considerarse en movimientos posteriores.

Obtención de números aleatorios en C

Para obtener números aleatorios debe utilizarse la función **rand()**, la cual está disponible en la biblioteca *stdlib.h*.

Esta función devuelve números pseudo-aleatorios, esto quiere decir que, cuando uno ejecuta nuevamente el programa, los números, aunque aleatorios, son los mismos.

Para resolver este problema debe inicializarse una semilla, cuya función es determinar desde dónde empezarán a calcularse los números aleatorios.

Los números arrojados por **rand()** son enteros sin signo, generalmente queremos que estén acotados a un rango (queremos números aleatorios entre tal y tal). Para esto, podemos obtener el resto de la división de **rand()** por el valor máximo del rango que necesitamos.

Aquí dejamos un breve ejemplo de cómo obtener números aleatorios entre 10 y 30.

```c
#include <stdio.h>
#include <stdlib.h> // Para usar rand
#include <time.h>   // Para obtener una semilla desde el reloj

int main(){
   srand ((unsigned)time(NULL));
   int numero = rand() % 20 + 10; // la amplitud del rango es 20 y el valor mínimo
es 10.
   printf("El valor aleatorio es: %i\n", numero);

   return 0;
}
```