.UBAfiuba
FACULTAD DE INGENIERÍA

FUNDAMENTOS DE PROGRAMACIÓN - CURSO MENDEZ

# Trabajo Práctico Nº 1

## Desastre nuclear

| | |
|---|---|
| Fecha presentación | 03/04/2025 |
| Fecha entrega | 01/05/2025 |

Primer cuatrimestre 2025

# 1. Introducción

Después de un largo día en la planta nuclear, Homero sale apurado rumbo al bar de Moe. Pero distraído y con la mente puesta en su cerveza, tropieza con una caja repleta de barras de plutonio, desparramándolas por todos lados. Ayudalo a juntarlas todas antes de que se quede sin energía y decida que lo mejor es rendirse e irse a tomar una cerveza.

# 2. Objetivo

El presente trabajo práctico tiene como objetivo que el alumno:

- Diseñe y desarrolle las funcionalidades de una biblioteca con un contrato preestablecido.
- Se familiarice con y utilice correctamente los tipos de datos estructurados.
- Desarrolle una interfaz gráfica amigable y entendible para el usuario.

Por supuesto, se requiere que el trabajo cumpla con las **buenas prácticas de programación** profesadas por la cátedra. Se considerarán críticos la modularización, reutilización y claridad del código.

# 3. Enunciado

El juego consiste en ayudar a Homero a juntar las barras de plutonio en un cuarto oscuro antes de quedarse sin energía. Para poder ver los elementos en el cuarto, Homero debe colocarse sobre los interruptores que están repartidos en el lugar. A su vez, en el terreno va a haber herramientas y obstáculos que pueden dificultar o facilitar el trabajo.

Al comenzar el juego, se deberá posicionar los elementos en posiciones aleatorias en el terreno. El mismo estará a oscuras, es decir, únicamente se podrá visualizar a Homero y los Interruptores, los demás elementos no serán visibles al usuario, sino que se podrán ver únicamente cuando Homero se posicione sobre un interruptor o utilice una linterna.

***IMPORTANTE*** **Ningún elemento puede inicializarse por fuera de los límites del terreno ni pisar otros elementos ya inicializados**.

## 3.1. Homero

Homero se inicializará en una posición aleatoria y podrá moverse por todo el terreno para cumplir el objetivo. Comenzará el juego con 400 puntos de energía, y con cada movimiento perderá un punto. Si Homero se queda sin energía antes de haber recolectado todas las barras de plutonio, el juego se dará por perdido.

## 3.2. Barras

Habrá 10 barras de plutonio que deberán inicializarse en posiciones aleatorias. Para recolectar una, Homero debe posicionarse encima de la misma. Una vez recolectada, la barra deberá desaparecer del terreno eliminándola del vector. Al juntar las 10 barras, el juego se dará por ganado.

## 3.3. Herramientas

### 3.3.1. Donas

Se tendrán 5 donas dispersas por el terreno de forma aleatoria.
Las donas le sumarán 10 puntos de energía a Homero.
Para recolectar una dona, Homero deberá estar en la misma posición que una. Luego, esa deberá eliminarse del vector de herramientas.

### 3.3.2. Interruptores

Habrá 4 interruptores distribuídos en el terreno, todos deberán inicializarse en posiciones aleatorias.
Cuando Homero se posicione encima de cualquiera de los interruptores, esto hará que se prenda la luz y por lo tanto se podrán ver todos los elementos del terreno. Cuando Homero no esté sobre algún interruptor, la luz estará apagada, por lo que no se podrá ver ningún elemento que no sean Homero o los Interruptores.

#### 3.3.3. Linternas

Homero comenzará el juego con 5 linternas.

Cuando las active, se le restará una linterna al personaje y este podrá ver los elementos que se encuentren hasta una distancia manhattan 2 de él. Una vez que Homero activa una linterna, esta permanecerá prendida durante sus siguientes 5 movimientos.

### 3.4. Obstáculos

#### 3.4.1. Ratas radioactivas

Se tendrán 5 ratas radioactivas dispersas en el terreno de forma aleatoria.

Cada vez que se encienda una luz, ya sea con un interruptor o con la linterna, las ratas se moverán a un lugar aleatorio (mientras la linterna esté prendida, no seguirán moviéndose, solo cambiarán de posición en el momento en que se active). Es decir, cada vez que se use un interruptor o se encienda la linterna, todas las ratas aparecerán en una nueva ubicación al azar dentro del terreno, sin pisar ningún elemento.

Si Homero choca con alguna de las ratas, perderá una linterna.

#### 3.4.2. Barriles

Se tendrán 15 barriles dispersos en el terreno de forma aleatoria. Si Homero choca con alguno de ellos (está en la misma posición que un barril) perderá 15 puntos de energía.

### 3.5. Propulsores

Estos son elementos que le permite al personaje moverse más rápido por el terreno permitiéndole ahorrar energía, pero que a su vez tiene que tener cuidado de usar, porque puede enviarlo a posiciones peligrosas. Pueden ser escaleras o charcos.

#### 3.5.1. Escaleras

Habrá 3 escaleras que deberán inicializarse en posiciones aleatorias. Cuando Homero se coloque sobre una, se transportará a otro lugar del terreno dependiendo del movimiento que haya realizado para colocarse sobre la escalera. **Homero va a moverse hasta cumplir 3 movimientos o hasta que llegue al límite del terreno**.

- DERECHA: se moverá 3 lugares hacia la **diagonal arriba derecha**.
- IZQUIERDA: se moverá 3 lugares hacia la **diagonal arriba izquierda**.
- ABAJO o ARRIBA: se moverá 3 lugares hacia **arriba**.

Esto quiere decir que si por ejemplo, para colocarse sobre la escalera Homero se movió a la derecha, en ese mismo movimiento deberá moverse 3 lugares hacia la diagonal arriba derecha.

Si hay elementos en esos 3 lugares que Homero se movió, deberá interactuar con todos.

Al realizar estos 3 movimientos se pierde solo un punto de energía.

#### 3.5.2. Charcos

Habrá 3 charcos que deberán inicializarse en posiciones aleatorias. Cuando Homero se coloque sobre una, se transportará a otro lugar del terreno dependiendo del movimiento que haya realizado para colocarse sobre el charco. **Homero va a moverse hasta cumplir 3 movimientos o hasta que llegue al límite del terreno**.

- DERECHA: se moverá 3 lugares hacia la **diagonal abajo derecha**.
- IZQUIERDA: se moverá 3 lugares hacia la **diagonal abajo izquierda**
- ABAJO o ARRIBA: se moverá 3 lugares hacia **abajo**.

Si hay elementos en esos 3 lugares que Homero se movió, deberá interactuar con todos.

Al realizar estos 3 movimientos se pierde solo un punto de energía.

### 3.6. Terreno

El terreno está compuesto por los elementos mencionados anteriormente, cada uno de ellos con una coordenada (x, y) con x siendo un valor entre 0 y 19 inclusives e y siendo un valor entre 0 y 19 inclusives.

### 3.7. Modo de juego

Al moverse, Homero no puede pasarse de los límites del terreno. Por ejemplo, si Homero está en la fila 0 y el usuario lo mueve para arriba, Homero debería quedarse ahí, no se debería mover porque estaría saliéndose del terreno y ese movimiento no restaría energía ya que no se movió.

**Es importante que antes de realizar el movimiento, se valide lo que ingrese el usuario**, volviéndole a preguntar hasta que ingrese un movimiento correcto.

Homero se podrá mover en 4 direcciones:

- **Arriba**: W
- **Abajo**: S
- **Derecha**: D
- **Izquierda**: A

Además, para activar las linternas deberá ingresar la siguiente instrucción:

- **Linternas**: L.

Luego de realizar una acción en caso de chocar o estar a la distancia de reacción con un elemento se activará la reacción relacionada al mismo que afectará a Homero y el estado del juego.

El juego finaliza cuando:

- Homero recolecta las 10 barras. En este caso, el juego se dará por ganado
- Homero se queda sin energía. En este caso, el juego se dará por perdido.

## 4. Especificaciones

### 4.1. Convenciones

- **Homero**: H.
- **Interruptorres**: I.
- **Barras**: B.

Se deberá utilizar la siguiente convención para los obstáculos:

- **Ratas radioactivas**: R.
- **Barriles**: A.

Para las herramientas:

- **Donas**: D.

Para los propulsores:

- **Escaleras**: E.
- **Charcos**: C.

## 4.2. Funciones y procedimientos

Para poder cumplir con lo pedido, se pedirá implementar las siguientes funciones y procedimientos.

```c
#ifndef __PLUTONIO_H__
#define __PLUTONIO_H__

#include <stdbool.h>

#define MAX_HERRAMIENTAS 30
#define MAX_OBSTACULOS 30
#define MAX_BARRAS 10
#define MAX_PROPULSORES 10

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

typedef struct objeto {
    coordenada_t posicion;
    char tipo;
    bool visible;
} objeto_t;

typedef struct personaje {
    coordenada_t posicion;
    int cantidad_linternas;
    int cantidad_barras;
    int energia;
    bool linterna_activada;
    int mov_linterna;
} personaje_t;

typedef struct juego {
    personaje_t homero;
    objeto_t herramientas[MAX_HERRAMIENTAS];
    int cantidad_herramientas;
    objeto_t obstaculos[MAX_OBSTACULOS];
    int cantidad_obstaculos;
    objeto_t barras[MAX_BARRAS];
    int cantidad_barras;
    objeto_t propulsores[MAX_PROPULSORES];
    int cantidad_propulsores;
} juego_t;

/*
 * Pre condiciones: -
 * Post condiciones: Inicializará el juego, cargando toda la información inicial de Homero, las
    herramientas y los obstáculos.
 */
void inicializar_juego(juego_t *juego);

/*
 * Pre condiciones: El juego debe estar inicializado previamente con `inicializar_juego` y la acción
 * debe ser válida.
 * Post condiciones: Realizará la acción recibida por parámetro actualizando el juego.
 */
void realizar_jugada(juego_t *juego, char accion);

/*
 * Pre condiciones: El juego debe estar inicializado previamente con `inicializar_juego `.
 * Post condiciones: Imprime el juego por pantalla.
 */
void mostrar_juego(juego_t juego);

/*
 * Pre condiciones: El juego deberá estar inicializado previamente con `inicializar_juego `
 * Post condiciones: Devuelve:
 * --> 1 si es ganado
 * --> -1 si es perdido
 * --> 0 si se sigue jugando
 * El juego se dará por ganado cuando Homero junta todas las barras de plutonio.
 * Se dará por perdido si se le termina la energía a Homero y no juntó todas las barras de plutonio.
 */
int estado_juego(juego_t juego);

```

```
73
74 #endif /* __PLUTONIO_H__ */
```

**Observación:** Queda a criterio del alumno/a hacer o no más funciones y/o procedimientos para resolver los problemas presentados. No se permite agregar funciones al .h presentado por la cátedra, como tampoco modificar las funciones ni los structs dados.

# 5. Resultado esperado

El tp es un juego. Esperamos que el trabajo cumpla la funcionalidad explicada anteriormente. Se deberá:

- Implementar todas las funciones especificadas en la biblioteca.
- Inicializar **todos** los campos del registro juego_t.
- Pedirle al usuario que ingrese una acción **válida** a realizar cada turno.
- Imprimir el terreno de forma clara con información que pueda serle útil al usuario (cuántas linternas tiene, las barras, etc)
- Respetar las buenas prácticas de programación que profesamos en la cátedra.

# 6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado plutonio.c, lo que sería la implementación de la biblioteca plutonio.h.

**Aclaración:** Además del desarrollo de la biblioteca, se deberá implementar un main, el cual tiene que estar desarrollado en un archivo llamado juego.c. En este se manejará todo el flujo del programa, utilizando las funciones de la biblioteca realizada e incluyendo el pedido del movimiento al usuario y la validación del mismo.

El trabajo debe ser compilado sin errores al correr el siguiente comando:

```
1 gcc juego.c plutonio.c -o juego -std=c99 -Wall -Wconversion -Werror -lm
```

Por último debe ser entregado en la plataforma de corrección de trabajos prácticos **AlgoTrón** (patente pendiente), en la cual deberá tener la etiqueta **¡Exito!** significando que ha pasado las pruebas a las que la cátedra someterá al trabajo.

**IMPORTANTE!** ***Esto no implica necesariamente haber aprobado el trabajo ya que además será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación. Para que el trabajo sea corregido por un colaborador, el mismo debe pasar todas las pruebas mínimas.***.

Para la entrega en **AlgoTrón** (patente pendiente), recuerde que deberá subir un archivo **zip** conteniendo únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

# 7. Anexos

## 7.1. FAQ

En este link encontrarán el documento de FAQ del TP, donde se irán cargando dudas realizadas con sus respuestas. Todo lo que esté en ese documento es válido y oficial para la realización del TP.

## 7.2. Obtención de números aleatorios

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 como 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;
}
```

## 7.3. Limpiar la pantalla durante la ejecución de un programa

Muchas veces nos gustaría que nuestro programa pueda verse siempre en la pantalla sin ver texto anterior.

Para esto, podemos utilizar la llamada al sistema **clear**, de esta manera, limpiaremos todo lo que hay en nuestra terminal hasta el momento y podremos dibujar la información actualizada.

Y se utiliza de la siguiente manera:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
        printf("Escribimos algo\n");
        printf("que debería\n");
        printf("desaparecer...\n");

        system("clear"); // Limpiamos la pantalla

        printf("Solo deberiamos ver esto...\n");
        return 0;
}
```

## 7.4. Distancia Manhattan

Para obtener la distancia entre 2 puntos mediante este método, se debe conocer a priori las coordenadas de dichos puntos.

Luego, la distancia entre ellos es la suma de los valores absolutos de las diferencias de las coordenadas. Se ve claramente en los siguientes ejemplos:

- La distancia entre los puntos (0,0) y (1,1) es 2 ya que: $|0 - 1| + |0 - 1| = 1 + 1 = 2$
- La distancia entre los puntos (10,5) y (2,12) es 15 ya que: $|10 - 2| + |5 - 12| = 8 + 7 = 15$
- La distancia entre los puntos (7,8) y (9,8) es 2 ya que: $|7 - 9| + |8 - 8| = 2 + 0 = 2$