UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Paradigmas de programación.

Manual de usuario

Andrés Felipe Muñoz Bravo
19.646.487-5

Profesor:
Roberto González Ibáñez

Santiago - Chile
2016

UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

UdeSantiago de Chile

TABLA DE CONTENIDOS

Tabla de contenido

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

Este programa es un juego basado en otro juego llamado “Battle ship”, su traducción literal es “Batalla Naval”. Este juego necesita dos jugadores, uno será la computadora y el otro será un usuario. El juego transcurre a través de turnos en los cuales cada jugador, realiza un ataque al oponente, el cual puede resultar en la destrucción de un barco. El juego finaliza cuando uno de los dos contrincantes no posee barcos.

En el presente manual, se pretende explicar la manera de utilizar de forma correcta el programa. Este se ha desarrollado en el lenguaje de programación Scheme, que opera bajo el paradigma de programación Funcional. Para esto se utilizó el intérprete DrRacket .

CAPÍTULO 2. FUNCIONALIDADES DEL PROGRAMA.

2.1 (CREATEBOARDRL N M SHIPS SEED) Y (CREATEBOARD N M SHIPS SEED)

Esta funciones que crean un tablero de dimensiones “N” x “M” donde se ubican en la parte del enemigo la cantidad “SHIPS” de barcos de forma aleatoria utilizando la semilla “SEED”.

- N: Cantidad de filas del tablero, debe ser par
- M: Cantidad de columnas del tablero.
- ships: Cantidad de barcos enemigos a ubicar aleatoriamente.
- seed: Semilla para utilizar las funciones aleatorias. Puede ser cualquier número.

```
> (createBoardRL 10 10 10 10)
{10
 0
 0
 0
 {{#\. #\E #\. #\. #\. #\. #\. #\. #\E #\.}
  {#\. #\. #\. #\. #\E #\. #\. #\E #\. #\.}
  {#\E #\. #\. #\. #\. #\E #\. #\. #\. #\.}
  {#\. #\E #\. #\. #\E #\. #\. #\. #\. #\.}
  {#\. #\. #\E #\. #\. #\. #\. #\E #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}}
 ()}
```

Ilustración 1: Función createBoardRL

```
> (createBoardRC 10 10 10 10)
{10
 0
 0
 0
 {{#\. #\E #\. #\. #\. #\. #\. #\. #\E #\.}
  {#\. #\. #\. #\. #\E #\. #\. #\E #\. #\.}
  {#\E #\. #\. #\. #\. #\E #\. #\. #\. #\.}
  {#\. #\E #\. #\. #\E #\. #\. #\. #\. #\.}
  {#\. #\. #\E #\. #\. #\. #\. #\E #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}}
 ()}
```

Ilustración 2: Función createBoardRC

2.2 (CHECKBOARD BOARD):

Función que verifica si un tablero es válido para el juego. Recibe un board (tablero) como parámetro, entrega verdadero (#t) o falso (#f) según el caso.

```
> (define board (createBoardRC 4 4 4 4) )
> (checkBoard board)
#t
> (checkBoard (list 1 2 3 4 (list 2 4 1)))
#f
>
```

Ilustración 3: Función checkBoard

2.3 (PUTSHIP BOARD POSITION SHIP)

Función que retorna un tablero con un barco ubicado según los parámetros de entrada.

- Board: tablero en el cual se ubicara un barco.
- Position: Coordenadas del barco a ubicar en la matriz:
    - Fila: Entre 0 y N/2. Donde N es la cantidad de filas del tablero.
    - Columna: Entre 0 y M. Donde M es la cantidad de columnas del tablero.
- Ship: Carácter que representa al barco.

```
> (putShip (createBoardRC 10 10 10 10) (createPosition 0 1) #\M)
{10
 1
 0
 0
 {{#\. #\E #\. #\. #\. #\. #\. #\. #\E #\.}
  {#\. #\. #\. #\. #\E #\. #\. #\E #\. #\.}
  {#\E #\. #\. #\. #\. #\E #\. #\. #\. #\.}
  {#\. #\E #\. #\. #\E #\. #\. #\. #\. #\.}
  {#\. #\. #\E #\. #\. #\. #\. #\E #\. #\.}
  {#\. #\M #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}
  {#\. #\. #\. #\. #\. #\. #\. #\. #\. #\.}}
 ()}
>
```

Ilustración 4: Función putShip

2.4 (PLAY BOARD SHIP POSITIONS SEED)

Función devuelve un tablero con una jugada realizada.

- Board: Tablero en el cual se hará una jugada.
- Ship: Carácter del barco que ataca.
- Positions: Lista de posiciones en las cuales atacara el barco en la parte del tablero enemigo:
    - Fila: Entre 0 y N/2. Donde N es la cantidad de filas del tablero.
    - Columna: Entre 0 y M. Donde M es la cantidad de columnas del tablero.
- Seed: semila para ocupar las funciones aleatorias. Puede ser cualquier número.

2.5 (BOARD->STRING BOARD SHOWCOMLPETE)

Función que devuelve un string del tablero, este string puede o no puede mostrar los barcos del enemigo dependiendo del parámetro showComplete.

```
> (board->string boardRC 0)
"....\n....\n....\n....\n"
> (board->string boardRC 1)
"E.E.\n.E.E\n....\n....\n"
```

Ilustración 5: Función board->string

```
> (display (board->string boardRC 0))
....
....
....
....
> (display (board->string boardRC 1))
E.E.
.E.E
....
....
>
```

Ilustración 6: Mostrar string del board

2.6 (GETSCORE BOARD)

Función que retorna el puntaje obtenido de usuario durante el juego.

- Board: Tablero del cual se obtiene el puntaje.

```
> (getScore boardRC)
0
> (getScore bPlay2)
2
>
```

Ilustración 7: Función getScore

También se realizó una función para obtener el puntaje obtenido por la computadora:

2.6.1 (get-score-cpu board):

Recibe el mismo parámetro que la función anterior.

```
> (get-score-cpu boardRC)
0
> (get-score-cpu bPlay2)
1
```

Ilustración 8: Función get-score-cpu

CAPÍTULO 3. POSIBLES ERRORES.

No se encontraron posibles errores dentro del programa.