



UNIVERSITAT  
ROVIRA I VIRGILI

Departament d'Enginyeria Informàtica i Matemàtiques

## **El joc d'enfonsar la flota**

**ALUMNE:** Christian Callau Romero

**PROFESSOR:** Xavier Mallafré Porta

**ASSIGNATURA:** Fonaments de programació

**ENSENYAMENT:** 1r Grau

**GRUP:** T1

**DATA:** 31 / gener / 2016

## Índex

<b>1</b>	<b>ESPECIFICACIONS.....</b>	<b>1</b>
<b>2</b>	<b>ANÀLISI.....</b>	<b>10</b>
<b>3</b>	<b>DISSENY .....</b>	<b>11</b>
3.1	ESTRUCTURA DE DADES .....	11
3.2	ALGORISME .....	12
3.3	DISSENY DEL JOC DE PROVES .....	44
<b>4</b>	<b>AVALUACIÓ.....</b>	<b>46</b>

# 1 Especificacions

## 1.1.1 El joc d'enfonsar vaixells

El projecte final tracta de programar un joc, que consisteix a enfonsar tots els vaixells del jugador contrari. Els vaixells de cada jugador estan col·locats dins d'un taulell quadrat (matriu), i les seves dimensions poden ser de 8x8, 9x9 o 10x10 caselles. Els vaixells poden ocupar diverses caselles (contigües) del taulell, en funció del seu tipus:

- Submarí: 1 casella
- Dragamines: 2 caselles
- Destructor: 3 caselles
- Portaavions: 4 caselles

Es disposa de 4 submarins, 3 dragamines, 2 destructors i 1 portaavions. Els vaixells es poden col·locar tant en horitzontal com en vertical. Els vaixells no poden superposar-se ni tocar-se, és a dir, entre un vaixell i un altre ha d'haver-hi, com a mínim, una casella buida (aigua).

El joc ha de començar amb un menú amb les següents opcions com a mínim:

1. Crear un nou joc
2. Carregar un joc emmagatzemat
3. Jugar partida
4. Emmagatzemar el joc
5. Veure pòdium
6. Sortir del joc

L'objectiu del joc és enfonsar tots els vaixells de l'oponent llençant (per torns) bombes sobre els vaixells de l'altre jugador i anotant els resultats (Aigua, Tocat, Enfosat) sobre el seu taulell de llançaments.

## 1.1.2 Taulells dels jugadors

Cada jugador disposa de dos taulells: taulell de vaixells i taulell de llançaments.

### Taulell de vaixells

El taulell de vaixells permet al jugador fixar la posició dels seus vaixells i anotar els resultats dels trets de l'oponent, per saber quan li han enfonsat els seus vaixells.

Per crear el taulell de vaixells existeixen dos possibilitats:

- Automàticament. Es pot crear un nou taulell amb els vaixells col·locats automàticament, executant el codi *inicia\_taulell (dim, taulell\_vaixells)*, que es dona fet.

- Manualment. El jugador també podrà crear el seu propi taulell de vaixells posicionant tots els vaixells. Per a col·locar cada vaixell només caldrà demanar al jugador la posició de la primera casella que ocuparà el vaixell (fila, columna) i la seva orientació (H/V). Es posicionaran tots els vaixells, del més grand al més petit. El programa no ha de permetre que se sobreposin vaixells, ni que es toquin, ni que ocupin posicions de fora del taulell.

Les anotacions dels trets del jugador contrari sobre el taulell de vaixells es faran automàticament.

	1	2	3	4	5	6	7	8	9	10
A	@	.	.	.	.	.	.	.	.	.
B	.	.	.	-	X	.	.	.	@	@
C	-	.	.	.	.	.	.	.	.	.
D	.	.	@	@	@	@	.	@	.	.
E	.	.	.	.	.	.	.	@	.	.
F	.	.	.	@	.	.	.	@	-	.
G	@	.	.	.	.	-	.	.	.	@
H	@	.	.	.	@	X	X	.	.	.
I	.	.	.	.	-	.	.	.	.	@
J	.	.	.	.	.	.	.	.	.	@

**Figura 1.** Possible vista del taulell de vaixells amb els vaixells no tocats (@), els vaixells tocats (X), i els trets a l'aigua (-).

### Taulell de llançaments

El taulell de llançaments permet anotar els resultats dels trets del jugador que té el torn.

Inicialment apareixerà buit (caràcters '?'), i després caldrà marcar les caselles a mesura que es van llançant les bombes, diferenciant les dues possibilitats: aigua, amb el caràcter '.', quan la bomba no ha tocat cap vaixell. O tocat, amb el caràcter '@', quan la bomba cau en una casella ocupada per un vaixell.

Les anotacions sobre el taulell de llançaments es faran després de cada llançament segons el resultat del mateix. En el cas d'enfonsar un vaixell, també es marcaran com aigua les caselles del voltant.

	1	2	3	4	5	6	7	8	9	10
A	?	?	?	?	?	?	?	?	?	?
B	?	?	?	?	?	?	?	?	?	?
C	?	?	?	.	.	.	?	?	?	?
D	?	?	?	.	@	.	?	?	?	?
E	?	?	?	.	.	.	?	?	?	?
F	?	?	?	?	?	?	?	?	?	.
G	?	?	?	?	?	?	?	?	?	?
H	?	?	.	?	?	?	?	?	?	?
I	?	?	@	?	?	?	?	?	?	?
J	?	?	?	?	?	?	?	?	?	?

**Figura 2.** Possible vista del taulell de llançaments on anotar els resultats de cada tret, marcant amb diferents caràcters les caselles no visitades ('?'), les que sabem que hi ha aigua('.') i les que sabem que hi ha un vaixell tocat o enfonsat ('@').

### ***1.1.3 Dinàmica del joc***

Durant la partida, l'usuari haurà d'indicar les coordenades de la casella on vol llançar la bomba; amb aquestes coordenades, el programa haurà d'invocar a la funció *dispara(f,c,taulell\_vaixells)* (aquest codi que es dóna fet i es detalla més endavant). Aquesta funció actualitza el taulell de vaixells i informa del resultat del llançament:

- Casella repetida (0). Quan ja s'ha disparat abans sobre la casella.
- Aigua (1)
- Tocat (2)
- Tocat i enfonsat (3)

Abans de fer un llançament s'haurà de comprovar que el jugador proporciona les coordenades correctes (dins del taulell).

Després de cada tret, s'hi haurà d'actualitzar el taulell de llançaments amb el resultat.

Si la partida és de dos jugadors, el jugador que encerta continua jugant. Si falla, el torn passa a l'altre jugador.

El joc acaba quan un jugador enfonsa tots els vaixells del contrari; aquest és el guanyador.

Durant la partida s'ha de poder aturar el joc i tornar al menú principal per poder emmagatzemar-lo i, eventualment, sortir del programa.

### ***1.1.4 Configuració del joc***

Per a començar un joc hi ha dues possibilitats:

#### **Iniciar un nou joc (opció 1)**

Primer s'ha de demanar a l'usuari la mida del taulell i el nombre de jugadors. En iniciar un nou joc es crearà un nou taulell de vaixells per cada jugador. Els taulells de vaixells es crearan de manera automàtica.

Si es tria l'opció de dos jugadors, l'usuari podrà crear el seu propi taulell de vaixells de manera automàtica o manual.

#### **Carregar un joc emmagatzemat (opció 2)**

El joc es pot emmagatzemar en un fitxer binari i recuperar-lo després per a continuar amb la partida.

En aquest cas, en el moment de carregar el joc, el programa informará l'usuari de les dimensions del taulell i de la modalitat del joc.

### 1.1.5 Modalitats de joc

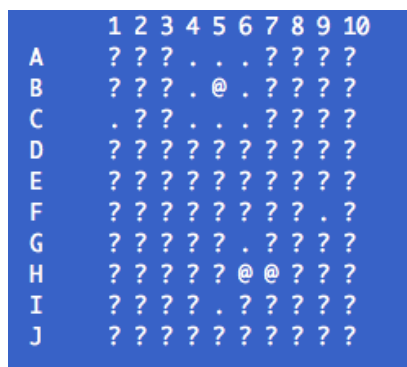
Hi ha tres modalitats de joc, segons el nombre de jugadors: cap, un o dos jugadors.

#### Cap jugador (nombre de jugadors = 0)

En aquesta modalitat, la màquina (el programa) juga contra ella mateixa.

El programa a desenvolupar simularà la lògica de joc de la màquina. S'indicarà les coordenades de cada tret: fila (lletra) i columna (número), que es mostraran per pantalla i llançarà la bomba utilitzant el codi `dispara(f,c,taulell_vaixells)` que es dona fet. Segons el resultat del llançament, el programa haurà d'anotar els resultats al taulell de llançaments i decidir la següent casella de llançament.

Per poder veure el desenvolupament del joc, es mostrarà per pantalla el taulell de llançaments. L'usuari indicarà, mitjançant el teclat, quan es farà el següent llançament.



	1	2	3	4	5	6	7	8	9	10
A	?	?	?	.	.	.	?	?	?	?
B	?	?	?	.	@	.	?	?	?	?
C	.	?	?	.	.	.	?	?	?	?
D	?	?	?	?	?	?	?	?	?	?
E	?	?	?	?	?	?	?	?	?	?
F	?	?	?	?	?	?	?	?	.	?
G	?	?	?	?	?	.	?	?	?	?
H	?	?	?	?	?	@	@	?	?	?
I	?	?	?	?	.	?	?	?	?	?
J	?	?	?	?	?	?	?	?	?	?

**Figura 3.** Possible vista del taulell de llançaments amb els resultats de cada tret.

#### Un jugador (nombre de jugadors = 1)

L'usuari jugarà contra la màquina i intentarà enfonsar tots els vaixells d'aquesta. En aquesta modalitat, la màquina no dispara, únicament indicarà el resultat de cada tret.

Durant la partida, en tot moment s'ha de visualitzar a la pantalla el taulell de llançaments del jugador, inicialment buit (tot marcat amb caràcters '?'). Caldrà marcar les caselles a mesura que es van llançant les bombes, diferenciant les dues possibilitats: aigua (quan la bomba no ha tocat cap vaixell) i tocat (quan la bomba cau en una casella ocupada per un vaixell).

L'usuari indicarà, mitjançant el teclat, les coordenades de la casella on vol llançar la bomba: la fila (lletra) i la columna (número). El programa haurà d'informar de possibles errors si l'usuari proporciona coordenades incorrectes. Amb les coordenades correctes, el programa efectuarà el llançament (amb el codi `dispara(f,c,taulell_vaixells)`) i obtenirà el resultat del mateix.

	1	2	3	4	5	6	7	8
A	?	?	?	.	.	.	?	?
B	?	?	?	.	@	.	?	?
C	.	?	?	.	.	.	?	?
D	?	?	?	?	?	?	?	?
E	?	?	?	?	?	?	?	?
F	?	?	?	?	?	?	?	?
G	?	?	?	?	?	.	?	?
H	@	?	?	?	?	@	?	?

**Figura 4.** Possible vista del taulell de llançaments de l'usuari, de dimensions 8x8, amb els resultats de cada tret.

### Dos jugadors (nombre de jugadors = 2)

En aquest cas, l'usuari jugarà contra la màquina i viceversa. L'usuari posarà els seus vaixells al seu taulell de vaixells (de manera automàtica o manual) i intentarà enfonsar tots els vaixells de la màquina. Per la seva banda, la màquina fixarà la posició dels seus vaixells de manera automàtica i intentarà enfonsar tots els vaixells de l'usuari. Guanya el jugador que primer enfonsa tots els vaixells del seu contrincant.

Durant la partida, en tot moment s'ha de visualitzar a la pantalla el taulell de llançaments i el taulell de vaixells del jugador. Caldrà anar marcant les caselles dels dos taulells a mesura que es van llançant i rebent bombes.

L'usuari indicarà, mitjançant el teclat, les coordenades de la casella on vol llançar la bomba: la fila (lletra) i la columna (número). La màquina informarà del seu llençament: la fila (lletra) i la columna (número), per pantalla.

El programa haurà d'informar de possibles errors si l'usuari proporciona coordenades incorrectes. Amb les coordenades correctes, el programa efectuarà el llençament (amb el codi de *dispara()*). Els resultats dels llançaments s'anotaran al taulell corresponent.

	1	2	3	4	5	6	7	8	9	10
A	?	?	?	?	?	?	?	?	?	?
B	?	?	?	?	?	?	?	?	?	?
C	?	?	?	.	.	.	?	?	?	?
D	?	?	?	.	@	.	?	?	?	?
E	?	?	?	.	.	.	?	?	?	?
F	?	?	?	?	?	?	?	?	?	.
G	?	?	?	?	?	?	?	?	?	?
H	?	?	.	?	?	?	?	?	?	?
I	?	?	@	?	?	?	?	?	?	?
J	?	?	?	?	?	?	?	?	?	?

	1	2	3	4	5	6	7	8	9	10
A	@	.	.	.	.	.	.	.	.	.
B	.	.	.	.	@	.	.	.	@	@
C	r	.	.	.	.	.	.	.	.	.
D	.	.	@	@	@	@	.	@	.	.
E	.	.	.	.	.	.	.	@	.	.
F	.	.	.	@	.	.	.	@	.	.
G	@	.	.	.	.	.	.	.	.	@
H	@	.	.	.	@	X	X	.	.	.
I	.	.	.	.	.	.	.	.	.	@
J	.	.	.	.	.	.	.	.	.	@

**Figura 5.** Possible vista del taulell de llançaments (a l'esquerra) i del taulell de vaixells (a la dreta), tots dos de l'usuari, amb els resultats de cada tret.

### 1.1.6 Fi del joc

El joc finalitza quan un dels jugadors enfonsa tots els vaixells de l'altre. En acabar la partida, el joc ha de mostrar el pòdium amb els deu millors resultats, ordenats, i els punts aconseguits pel guanyador.

La puntuació ha de ser un **valor enter** i es calcularà segons la següent fórmula:

$$puntuació = 100 * \frac{dimensió\_taulell}{total\_llançaments} \sum (resultat\_llançament - 1)$$

**Equació 1.** Equació per a calcular la puntuació d'un jugador.

On:

- *dimensió\_taulell*. Dependrà de la configuració inicial del joc: 8, 9 o 10, per a taulells de 8x8, 9x9 i 10x10 respectivament.
- *resultat\_llançament*. El seu valor és el mateix que el resultat donat pel llançament: 0, 1, 2, 3 (Veure *Dinàmica del joc*).
- *total\_llançaments*. És el total de llançaments fets pel jugador.

Els rècords estaran emmagatzemats en un fitxer de text i s'actualitzaran amb els resultats de cada partida: si la puntuació aconseguida per un dels jugadors és superior a la màs baixa dels rècords actuals, s'afegirà la nova puntuació, amb el nom del jugador (usuari/màquina), a la llista de rècords, els rècords s'ordenaran i s'emmagatzemaran de nou al fitxer de rècords.

Amb la pulsació d'una tecla es tornarà al menú principal.

Un joc finalitzat no es pot emmagatzemar!!

## 1.2 Especificacions del nivell de desenvolupament

En aquest apartat heu de indicar les especificacions relatives al nivell de desenvolupament del projecte: nivell 1, 2 o 3.

### Nivell 1 o bàsic (No vàlid per a segona convocatòria)

El nivell 1 o bàsic consisteix a implementar el joc per a la modalitat de cap jugador: la màquina contra ella mateixa.

El joc tindrà els següents elements.

- Menú inicial.
- Configurar del joc per a la màquina (cap jugador): crear un joc nou (*inicia\_taulell()*).
- Crear la lògica de joc per descobrir els vaixells de la màquina: la funció *jugar()* indicarà les coordendades de llançament de la màquina. Aquesta funció representa



el cervell de la màquina, per tant, es tracta de dissenyar l'algorisme que permet a la màquina endivinar la posició dels vaixells amb el número mínim de trets.

- Dissenyar l'algorisme general de joc per a què pugui jugar la màquina: obtenir les coordenades del següent tret amb la funció `jugar()`, efectuar el tret, amb la funció `dispara()`, marcar al taulell de llançament els resultats, controlar el final del joc...
- Emmagatzemar i recuperar els rècords. Com a màxim s'emmagatzemaren 25 rècords al fitxer.

Per dissenyar la lògica de joc de la màquina, heu de dissenyar el procediment `jugar()`.

- *acció jugar (var f:caracter, var c:enter; dim:enter, taulell\_llancament:taula de caràcters);*

El procediment `jugar()` rep, com paràmetres, les coordenades del darrer llençament (*f* i *c*), les dimensions del taulell (*dim*) i el taulell de llançaments (*taulell\_llancaments*) i retornarà, per referència, les noves coordenades per fer el següent llançament (*f* i *c*).

Les coordenades han de ser vàlides.

Per emmagatzemar i recuperar els rècord, també heu de dissenyar dos procediments específics:

- *funció emmagatzema\_records (fitxer\_record:taula de caràcters, num\_records:enter, records:taula de record\_tipus) retorna booleà.*

Aquest procediment, rep el nom del fitxer de rècords per paràmetre (*fitxer\_record*) i, emmagatzema al fitxer de records els rècords actuals i informarà de si tot ha anat bé. Els rècords s'emmagatzemen quan finaliza una partida si el resultat de la partida és millor que el darrer rècord.

*num\_records* indica el nombre total de rècords emmagatzemats.

- *funció recupera\_records (fitxer\_record:taula de caràcters, num\_records:enter, var records:taula de record\_tipus) retorna booleà.*

Aquest procediment recupera els rècords emmagatzemats i informa si tot ha anat bé. Rep el nom del fitxer de rècords per paràmetre (*fitxer\_record*).

Els rècords es recuperen quan s'executa el joc per primera vegada. S'ha de decidir què fer en cas que no existeixi cap fitxer amb rècords.

A la implementació es pot utilitzar el fitxer objecte *vaixells.obj* amb els procediments necessaris per a inicialitzar el taulell de vaixells i per a disparar:

- *acció inicia\_taulell (dim:enter, var taulell\_vaixells:taula de caràcters);*

Inicialitza la disposició de tots els tipus de vaixells dins d'un nou taulell de vaixells.

- *funció dispara (f:caràcter, c:enter, var taulell\_vaixells:taula de caràcters) retorna enter;*

Retorna el resultat del llançament (repetit-0, aigua-1...) i actualitza el taulell de vaixells amb les anotacions pertinents.

A més dels procediments indicats, podeu dissenyar tots els procediments addicionals que necessiteu.

## **Nivell 2 o intermedi**

El nivell 2 o intermedi consisteix a completar el joc del nivell 1 per a permetre dues modalitats de joc diferents: cap jugador (juga la màquina) o un jugador (juga l'usuari contra la màquina).

A més a més, permetrà emmagatzemar el joc i continuar la partida a partir d'un joc emmagatzemat.

- Recuperar un joc emmagatzemat.
- Emmagatzemar el joc.
- Dissenyar l'algorisme general de joc per a què pugui jugar l'usuari: demanar a l'usuari les coordenades i confirmar que són vàlides, disparar amb la funció `dispara()`, marcar al taulell de llançament els resultats dels llançaments, controlar el final del joc...

Per a emmagatzemar i recuperar un joc s'han de dissenyar els següents procediments:

- *funció emmagatzema\_joc (nom\_fitxer:taula de caràcters, dim:enter, num\_jugadors:enter, jugadors:taula de jugador\_tipus) retorna booleà;*

Aquest procediment emmagatzema el joc actual, per això rep el nom del fitxer binari (*nom\_fitxer*), les dimensions del taulell (*dim*), el nombre de jugadors (*num\_jugadors*) i la informació dels jugadors (*jugadors*): nom, taulells, total de trets... I retorna si tot ha anat bé.

- *funció recupera\_joc (nom\_fitxer:taula de caràcters, var dim:enter, var num\_jugadors:enter, var jugadors:taula de jugador\_tipus) retorna booleà;*

Aquest procediment recupera la informació del joc emmagatzemat al fitxer binari que s'indica per paràmetre (*nom\_fitxer*) i informarà si tot ha anat bé.

A més de la informació dels jugadors ha d'indicar les dimensions del taulell i el nombre de jugadors.

Per demanar les coordenades al jugador heu de dissenyar el procediment *nova\_jugada*.

- *nova\_jugada (dim:enter, var coor:coor\_tipus);*

El procediment rep les dimensions del joc (*dim*) i retornarà les coordenades per disparar. L'acció no retornarà fins que l'usuari proporcioni unes coordenades vàlides.

A la implementació es pot utilitzar el fitxer objecte *vaixells.obj* amb les funcions necessàries per a crear el taulell de vaixells i disparar: *inicia\_taulell()* i *dispara()*.

### Nivell 3 o avançat

El nivell 3 o avançat consisteix a completar el joc del nivell 2 per a permetre jugar a totes les modalitats de joc:

- Cap jugador (nivell 1)
- Un jugador (nivell 2)
- Dos jugadors.

El joc tindrà els següents elements nous:

- Permetre al jugador triar entre iniciar el taulell de vaixells de forma automàtica (cridant a la funció *inicia\_taulell()*) o crear el seu propi taulell de vaixells amb la funció *inicia\_elmeu\_taulell()*.
- Dissenyar l'algorisme general de joc per a què pugui jugar l'usuari contra la màquina: controlar els torns dels jugadors, demanar les coordenades de forma automàtica amb la funció *jugar()* o de forma manual amb la funció *nova\_jugada()*, efectuar els trets, tant de l'usuari com de la màquina, amb la funció *dispara()*, marcar als taulells de llançament els resultats, controlar el final del joc...

S'ha de dissenyar una nova funció que permeti a l'usuari crear el seu propi taulell de vaixells.

- acció *inicia\_elmeu\_taulell* (*dim:enter, var taulell\_vaixells:taula de caràcters*);

Permet al jugador crear el seu propi taulell de vaixells. S'ha de garantir que el taulell sigui vàlid.

A més a més heu de dissenyar els procediments: *inicia\_taulell()* i *dispara()*.

S'ha de crear una llibreria *vaixells.lib* amb tots o part dels procediments implements.

## 2 Anàlisi

El treball plantejat es el famós joc d'enfonsar la flota, *battleship* en angles. La versió del joc que es demana es correspon a la variant russa que es diferencia de l'americana per el nombre i llargària dels vaixells i l'obligació de posicionar els vaixells sense que es toquin ni es superposen.

Es demanen tres modes de joc, el primer es tant sols la IA jugant tota sola, el segon es l'usuari jugant tot sol i finalment el tercer mode es la implementació completa de la mecànica del joc. També es demanen funcions extra com calcular la puntuació amb una formula donada, guardar les puntuacions, pausa i sortir, guardar la partida i recuperar una partida guardada.

Per al funcionament del joc necessitarem les següents funcions:

- Acció que generi el taulell de vaixells automàticament, per al jugador i per a la maquina i una modificació d'aquesta per a que l'usuari pugui iniciar el taulell de forma manual.

- Acció que pregunti al usuari el tipus de joc que vol i generi i actualitzi els taulells i variables pertinents.

- Funció que retorni el resultat dels llançaments i actualitzi el taulell de vaixells corresponent.

- Acció que actualitzi el taulell de llançaments, en cas de tocat i enfonsat necessitarem una altra funció per descobrir els voltants del vaixell enfonsat.

- Funció que s'encarregui de jugar per la màquina, ha de retornar coordenades vàlides i seguir amb la lògica del joc.

- Acció que s'encarregui de la lògica del joc, usant les funcions de disparar i actualitzar el taulell de llançaments, la funció jugar i que obtingué coordenades valides del usuari, controli els torns i el final del joc. També a de determinar qui es el guanyador i calcular la puntuació.

- Per als records necessitarem dues funcions, una per recuperar els records i l'altra per emmagatzemar-los. També necessitarem una funció per ordenar-los de mes gran a mes petit.

- Per a guardar la partida també necessitarem dues funcions per guardar i per emmagatzemar.

- Finalment necessitarem el menú principal que controli les opcions disponibles i s'encarregui d'executar les funcions necessàries.

## 3 Disseny

### 3.1 Estructura de dades

Se'ns demana guardar les puntuacions mes altes i com a mínim una partida amb tota la informació necessària per poder restaurar-la (taules, resultats...). Per cada un d'aquests problemes farem servir tipus que ens permeti guardar i treballar aquestes dades en mes facilitat.

Per al les puntuacions tant sols necessitem un enter que guardi el valor de la puntuació i una cadena de caràcters que guardi el nom. Després tant sols farà falta inicialitzar una taula amb tants elements com s'especifiquin.

Per restaurar una partida guardarem tota la informació d'un jugador en un mateix tipus. Ha de guardar la taula on estan disposats els vaixells, que serà una matriu de caràcters, la taula de llançaments, el nom del jugador que serà necessari per saber qui a guanyat i assignar-li la puntuació, les coordenades i el resultat del últim tret, el nombre de vaixells enfonsats, el nombre de trets realitzats i la suma dels resultats dels llançaments, els dos darrers son necessaris per calcular la puntuació.

També crearem un tipus específic per a guardar les coordenades i un altre per al tipus que guardarà els noms. Els valors com el màxim de caràcters del tipus nom o la dimensió màxima i mínima de les taules les guardarem en constants.

constants

```
DIM_MAX:=10;  
DIM_MIN:=8;  
MAX_NOM:=10;  
MAX_RECORDS:=25;
```

fconst

tipus

```
nom_tipus : taula [MAX_NOM] de caràcters;  
registre record_tipus és  
    nom : nom_tipus;  
    punts : enter;  
fregistre;  
registre coor_tipus és  
    lletra : caràcter;  
    nombre : enter;  
fregistre;  
registre jugador_tipus és  
    nom : nom_tipus;  
    coor : coor_tipus;  
    vaixells, llancaments : taula [DIM_MAX][DIM_MAX] de caràcters;  
    vaixells_enfonsats, sum_resultats, num_trets, resultats :  
    enter;  
fregistre
```

```

ftipus
var
    jugadors : taula [2] de jugador_tipus;
    records : taula [MAX_RECORDS] de record_tipus;
fvar

```

## 3.2 Algorisme

### 3.2.1 Programa principal

En el programa principal haurem d'inicialitzar les variables jugador\_tipus i record\_tipus, la dimensió i el nombre de jugadors que seran enters i el nombre de records que també serà un enter. El programa comença amb un menú per tant haurem de cridar la funció del menú, en quan aquest ens retorni una opció vàlida hem de cridar les funcions corresponents, ho farem amb una opció que estarà dins d'un mentre controlat per un booleà per a que tant sols es pugi sortir del joc quan l'usuari ho indiqui.

He decidit que les variables de record\_tipus i num\_records s'inicialitzin automàticament quan s'executa el programa, per tant si la funció recupera\_records trobar un arxiu vàlid amb records guarda el contingut a la taula de record\_tipus, guarda el nombre de records a la variable num\_records, ordena els records en cas de que no estiguin ordenats i els torna a guardar al arxiu de text.

### 3.2.2 Acció jugar();

Aquesta acció l'he modificat varies vegades, mentre treballava amb altres aspectes del joc vaig fer l'algorisme amb la funcionalitat mínima, tant sols retornava caselles aleatòries que estaven per descobrir, de manera que no disparava mai dues vegades en la mateixa casella però estava molt lluny de se òptim.

Mes endavant vaig afegir el mode "buscar i destruir" amb dues fases. En la fase buscar es dedica a disparar aleatòriament a caselles que estiguin per descobrir fins que toca un vaixell, si no enfonsa el vaixell entra en mode destruir. El mode destruir identifica si hi han altres parts del vaixell al voltant de la casella tocada, si no en troba cap tria un punt cardinal aleatori i dispara just una casella per sobre, sota o als costats de la casella del vaixell. En canvi, si hi ha mes d'una casella del vaixell descoberta identifica si esta posicionat en horitzontal o vertical i dispara a les caselles oportunes fins enfonsar-lo.

Aquest sistema va incrementar notablement la eficiència del algorisme amb una mitjana de jugades per partida molt mes alta que la primera versió.

Mitjana de jugades per partida, 100 milions de simulacions:

- Aleatori : 76
- Buscar i destruir: 60

Vaig quedar satisfet amb el funcionament de la fase destruir però vaig arribar a la conclusió que la fase de buscar es podia millorar.

Finalment vaig dissenyar el mode buscar i destruir amb probabilitats, es el mateix que l'anterior però en el mode buscar en comptes de triar una casella aleatòria entre totes les possibles calcula la probabilitat de que hi hagi un vaixell en cada casella pendent per descobrir i tant sols dispara a una casella aleatòria entre les que tinguin la probabilitat mes alta.

Per fer-ho s'inicia una taula nova cada cop on es guardarà la probabilitat de cada casella que serà un enter positiu. Identifico els vaixells que queden per enfonsar i ho guardo en les variables enteres port, destr i drag, després recorro casella per casella la taula de probabilitats intentant col·locar els vaixells en horitzontal i vertical a la taula de llançaments, si hi caben sumo 1 a les caselles que ocupa el vaixell a la taula de probabilitats, així amb tots els vaixells.

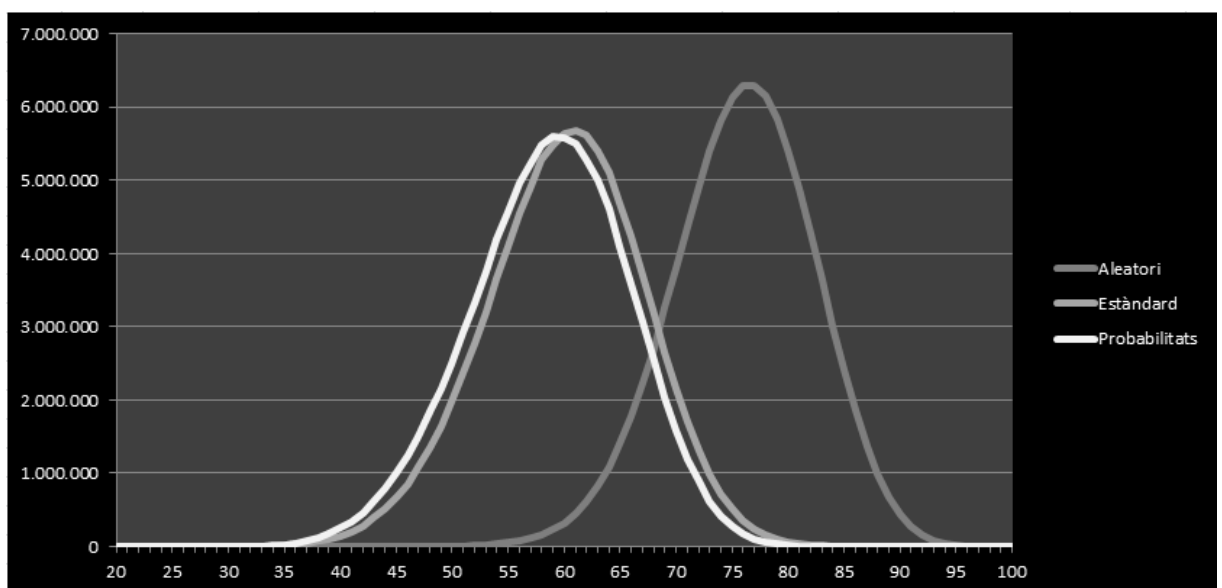
Finalment la probabilitat mes alta es guarda en la variable pmax i a l'hora de tirar les coordenades aleatòries també es verifica que corresponguin a una casella amb la probabilitat mes alta.

Després de tot aquest treball em vaig donar compte que hi ha millora respecte al algorisme anterior però és mínima.

Mitjana de jugades per partida, 100 milions de simulacions:

- Aleatori : 76
- Buscar i destruir: 60
- Buscar i destruir amb probabilitats: 58,6

Distribució del nombre de partides segons el nombre de jugades dels tres algorismes:

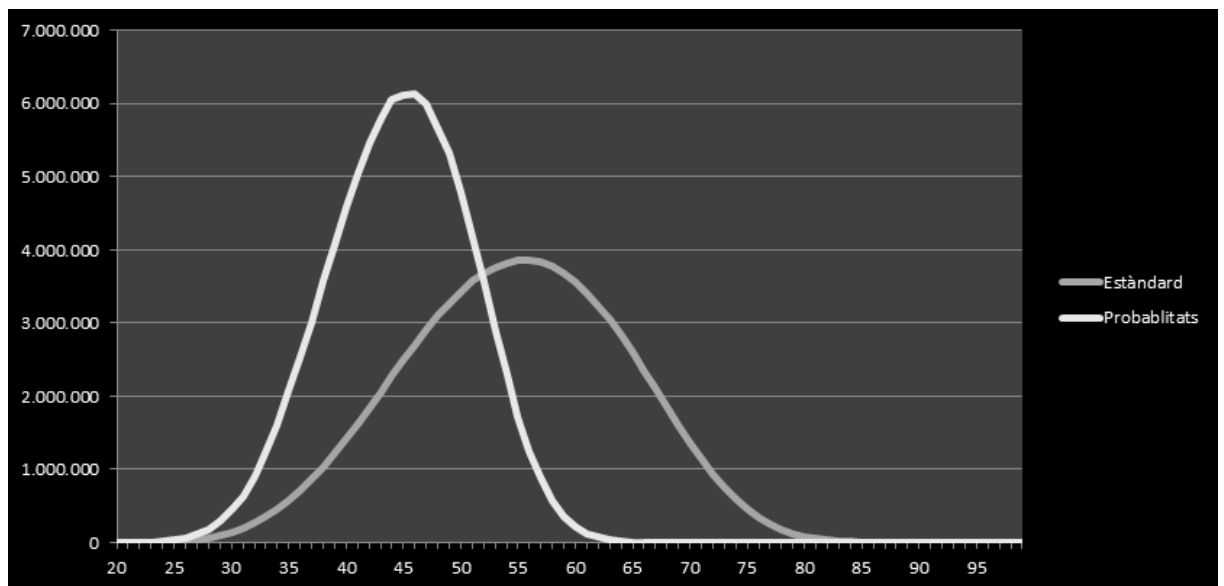


Tenia la sospita que la culpa de que els resultats no fossin suficientment bons era pels vaixells d'una casella que afegien molta aleatorietat al joc. Al moment que vaig finalitzar l'acció que inicia el taulell dels vaixells la vaig utilitzar per generar taulells amb un portaavions, dos destructors, tres dragamines però en aquest cas sense cap submarí. Aquest van ser els resultats:

Mitjana de jugades per partida, 100 milions de simulacions (sense submarins):

- Buscar i destruir: 56
- Buscar i destruir amb probabilitats: 46

Distribució del nombre de partides segons el nombre de jugades:



Com es pot veure al gràfic no tant sols obté una millor mitjana de jugades per partida sinó que en general te un comportament molt menys aleatori respecte dels altres algorismes.



### ***3.2.3 Acció descobrir\_voltants***

Descobrir els voltants d'un vaixell quan aquest es tocat i enfonsat es pot fer de moltes maneres, mentre pensava la solució vaig trobar tres algorismes diferents, al final domes he usat el que em semblava mes simple.

El funcionament es el següent, es delimita el quadrat que conte el vaixell amb, si estan dins dels límits del taulell, totes les caselles d'aigua que el toquen. Per fer-ho es necessiten dos punts cada un amb les seves coordenades. Un es situa al extrem nord-oest del vaixell i després es mou una casella en diagonal, l'altre fa el mateix però movent-se cap a l'extrem sud-est.

Finalment es recorre totes les caselles dins del quadrat i es substitueixen les caselles buides per aigua.

### ***3.2.4 Acció incia\_taulell***

L'objectiu d'aquest acció es generar taulells de vaixells vàlids, amb un portaavions de quatre caselles, dos destructors de tres caselles, tres dragamines de dues caselles i quatre submarins de una casella, tots aquest poden estar situats a qualsevol lloc del taulell en horitzontal o vertical sempre i quan no es toquin, no es superposin i hi haguí una casella d'aigua lliure entre les caselles de vaixells diferents.

Per fer-ho posiciona els vaixells del mes gran al mes petit, començant pel portaavions, genera coordenades i orientació aleatòria, comprova si hi cap al taulell, si no genera coordenades i orientació nova i si hi cap posa el vaixell al taulell i a mes canvia les caselles que estan tocant al vaixell per caselles d'aigua tocada.

Tot seguit fica els altres vaixells continuant pels destructors, fa el mateix que abans però en aquest cas vigila que cap casella del vaixell estigui damunt d'una casella d'un altre vaixell o duna casella d'aigua tocada.

Per als dragamines i els submarins pot haver-hi la situació, sobretot en dimensions 8 i 9, de que no hi hagi lloc perquè el taulell esta ple, en aquest cas comprova si hi ha lloc buscant almenys una casella d'aigua, si no en troba cap surt dels bucles i torna a començar.

### 3.2.5 Algorisme complet

#### Algorisme vaixells és

##### constants

```
DIM_MAX:=10;
DIM_MIN:=8;
MAX_NOM:=10;
MAX_RECORDS:=25;
FITXER_RECORDS:="records.txt";
FITXER_JOC:="partida.bin";
HORITZONTAL:=0;
VERTICAL:=1;
CASELLA_BUIDA:='?';
CASELLA_AIGUA:='.';
CASELLA_VAIXEL:='@';
CASELLA_VAIXEL_TOCAT:='X';
CASELLA_AIGUA_TOCADA:='-';
ERROR:=-1;
REPETIT:=0;
AIGUA:=1;
TOCAT:=2;
ENFONSAT:=3;
```

##### fconst

##### tipus

```
nom_tipus : taula [MAX_NOM] de caràcters;
registre record_tipus és
    nom : nom_tipus;
    punts : enter;
fregistre;
registre coor_tipus és
    lletra : caràcter;
    nombre : enter;
fregistre;
registre jugador_tipus és
    nom : nom_tipus;
    coor : coor_tipus;
    vaixells, llançaments : taula [DIM_MAX][DIM_MAX] de caràcters;
    vaixells_enfonsats, sum_resultats, num_trets,
    resultats : enter;
fregistre
```

##### ftipus

##### var

```
dim, num_jugadors, num_records : enter;
fin, joc : booleà;
jugadors : taula [2] de jugador_tipus;
records : taula [MAX_RECORDS] de record_tipus;
```

```

fvar
inici
    fin:=fals;
    joc:=fals;
    si recupera_records (FITXER_RECORDS, num_records, records)
    llavors
        ordena_records (num_records, records);
        emmagatzema_records (FITXER_RECORDS, num_records, records);
    fsi
    mentre no fin fer
        opcio menu_principal (joc)
        1:
            crea_joc (dim, num_jugadors, jugadors);
            joc:=cert;
        2:
            si recupera_joc (FITXER_JOC, dim, num_jugadors, jugadors)
            llavors
                joc:=cert;
                escriure("S'ha carregat una partida guardada.");
                escriure("Dimensio: ", dim);
                escriure("Nombre de jugadors: ", num_jugadors);
            sino
                escriure("No hi ha cap joc emmagatzemat.");
            fsi
        3:
            si joc llavors
                si (num_jugadors = 0) o (num_jugadors = 1) llavors
                    joc:=jugar_partida (dim, num_jugadors, jugadors,
                    num_records, records);
                fsi
                si (num_jugadors = 2) llavors
                    joc:=jugar_partida_dos (dim, jugadors, num_records,
                    records);
                fsi
            fsi
        4:
            si joc llavors
                si emmagatzema_joc (FITXER_JOC, dim, num_jugadors,
                jugadors) llavors
                    escriure("Joc emmagatzemat!");
                sino
                    escriure("Error! No s'ha pogut emmagatzemar el
                    joc.");
                fsi
            fsi
        5:
            veure_records (MAX_RECORDS, num_records, records);
        6:
            fin:=cert;
        fopcio
    fmente

```

## falgorisme

```
$***** copiar_cadena *****
acció copiar_cadena (VAR c:taula de caràcters, s:taula de caràcters)
és
var
    i:enter;
fvar
inici
    i:=0;
    mentre s[i] fer
        c[i]:=s[i];
        i:=i+1
    fmentre
    c:='\0';
facció
$***** menu_principal *****
funció menu_principal (joc:booleà) retorna enter és
var
    i, j:enter;
fvar
inici
    j:=0;
    si (no joc) llavors
        j:=1;
    fsi
    escriure ("1. Crear un nou joc");
    escriure ("2. Carrega un joc emmagatzemat");
    si (joc) llavors
        escriure ("3. Jugar partida");
        escriure ("4. Emmagatzemar el joc");
    fsi
    escriure ("5. Veure p\225dium");
    escriure ("6. Sortir del joc");
    escriure ("Selecciona una opció: ");
    mentre (llegir (i) != 1) o (i < 1) o (i > 6) o (i = 3*j) o (i =
4*j) fer
        escriure ("1. Crear un nou joc");
        escriure ("2. Carrega un joc emmagatzemat");
        si (joc) llavors
            escriure ("3. Jugar partida");
            escriure ("4. Emmagatzemar el joc");
        fsi
        escriure ("5. Veure p\225dium");
        escriure ("6. Sortir del joc");
        escriure ("Error! Selecciona una opció correcta: ");
    fmentre
    retorna i;
ffunció
$***** crea_joc *****
acció crea_joc (VAR dim:enter, VAR num_jugadors:enter, VAR
jugadors:taula de jugador_tipus) és
```

```

var
    opció:enter;
fvar
inici
    escriure ("Tria la mida del taulell (minim: , maxim: )", DIM_MIN,
DIM_MAX);
    escriure ("Mida: ");
    mentre (llegir (dim) != 1) o (dim < DIM_MIN) o (dim > DIM_MAX)
    fer
        escriure ("Tria la mida del taulell (minim: , maxim: )",
DIM_MIN, DIM_MAX);
        escriure ("Error! Selecciona una mida correcta: ");
    fmentre
    escriure ("1. Cap jugador");
    escriure ("2. Un jugador");
    escriure ("3. Dos jugadors");
    escriure ("Selecciona una opció: ");
    mentre (llegir (num_jugadors) != 1) o (*num_jugadors < 1) o
(*num_jugadors > 3) fer
        escriure ("1. Cap jugador");
        escriure ("2. Un jugador");
        escriure ("3. Dos jugadors");
        escriure ("Error! Selecciona una opció correcta: ");
    fmentre
    num_jugadors:=num_jugadors-1;
    si (num_jugadors = 2) llavors
        escriure ("De quina manera vols posicionar els vaixells?");
        escriure ("1. Automàtica");
        escriure ("2. Manual");
        escriure ("Selecciona una opció: ");
        mentre (llegir (opció ) != 1) o (opció < 1) o (opció > 2)
        fer
            escriure ("De quina manera vols posicionar els vaixells?");
            escriure ("1. Automàtica");
            escriure ("2. Manual");
            escriure ("Error! Selecciona una opció correcta: ");
        fmentre
    fsi
    inicia_taulell_custom (dim, jugadors[0].vaixells);
    inicia_taulell_llancaments (dim, jugadors[0].llancaments);
    jugadors[0].vaixells_enfonsats:=0;
    jugadors[0].sum_resultats:=0;
    jugadors[0].num_trets:=0;
    jugadors[0].resultat:=0;
    jugadors[0].coord.nombre:=-1;
    si (num_jugadors = 0) o (num_jugadors = 2) llavors
        copiar_cadena (jugadors[0].nom, "maquina");
    fsi
    si (num_jugadors = 1) llavors
        copiar_cadena (jugadors[0].nom, "jugador");
    fsi
    si (num_jugadors = 2) llavors
        si (opció = 1) llavors

```

```

        inicia_taulell_custom (dim, jugadors[1].vaixells);
    fsi
    si (opció = 2) llavors
        inicia_elmeu_taulell (dim, jugadors[1].vaixells);
    fsi
    inicia_taulell_llançaments (dim, jugadors[1].llançaments);
    jugadors[1].vaixells_enfonsats:=0;
    jugadors[1].sum_resultats:=0;
    jugadors[1].num_trets:=0;
    jugadors[1].resultat:=0;
    jugadors[1].coor.nombre:=-1;
    copiar_cadena (jugadors[1].nom, "jugador");
    fsi
facció
$***** imprimir_taules *****
acció imprimir_taules (dim:enter, taula_1, taula_2:taula de
caràcters) és
var
    f, c:enter;
    lletra:taula de caràcters;
fvar
inici
    lletra:="ABCDEFGHJIJ";
    escriure ("Taulell de vaixells \t Taulell de llançaments");
    per (c:=0; c<dim; c:=c+1) fer
        escriure ("%c ", lletra[c]);
    fper
    escriure ("\t");
    per (c:=0; c<dim; c:=c+1) fer
        escriure ("%c ", lletra[c]);
    fper
    per (f:=0; f<dim; f:=f+1) fer
        escriure (f+1);
        per (c:=0; c<dim; c:=c+1) fer
            escriure (taula_1[c][f]);
            escriure ("\t");
            escriure (f+1);
        fper
        per (c:=0; c<dim; c++) fer
            escriure (taula_2[c][f]);
        fper
    fper
facció
$***** imprimir_taula *****
acció imprimir_taula (dim:enter, taula:taula de caràcters) és
var
    f, c:enter;
    lletra:taula de caràcters;
fvar
inici
    lletra:="ABCDEFGHJIJ";
    per (c:=0; c<dim; c:=c+1) fer

```

```

        escriure (lletra[c]);
    fper
per (f:=0; f<dim; f:=f+1) fer
    escriure (f+1);
    per (c:=0; c<dim; c:=c+1) fer
        escriure (taula[c][f]);
    fper
fper
facció
$***** inicia_taulell_llancaments *****
acció inicia_taulell_llancaments (dim:enter, VAR
taulell_llancaments:taula de caràcters) és
var
    i, j:enter;
fvar
inici
    per (i:=0; i<dim; i:=i+1) fer
        per (j:=0; j<dim; j:=j+1) fer
            taulell_llancaments[i][j]:=CASELLA_BUIDA;
        fper
    fper
facció
$***** nova_jugada *****
acció nova_jugada (dim:enter, VAR coor:coor_tipus) és
inici
    mentre (llegir (coor.lletra, coor.nombre) != 2) o (coor.nombre <
    1) o (coor.nombre > dim) o (coor.lletra < 64) o (coor.lletra >
    dim+64) fer
        escriure ("Coordenades no vàlides! Torna a enterentar-ho: ");
    fmentre
facció
$***** nova_orientacio *****
acció nova_orientacio (VAR orientacio:enter) és
inici
    mentre (llegir (orientacio) != 1) o (orientacio < 0) o
    (orientacio > 1 ) fer
        escriure ("Orientació no vàlida! Torna a enterentar-ho: ");
    fmentre
facció
$***** descobrir_voltants *****
acció descobrir_voltants (x1, y1, dim:enter, VAR taula:taula de
caràcters) és
var
    x2, y2, i, j:enter;
fvar
inici
    x2:=x1;
    y2:=y1;
    mentre (taula[x1-1][y1] = CASELLA_VAIXELL) i (x1 - 1 >= 0) fer
        x1:=x1-1;
    fmentre
    mentre (taula[x1][y1-1] = CASELLA_VAIXELL) i (y1 - 1 >= 0) fer
        y1:=y1-1;

```

```

fmentre
si (x1-1 >= 0) llavors
    x1:=x1-1;
fsi
si (y1-1 >= 0) llavors
    y1:=y1-1;
fsi
mentre (taula[x2+1][y2] = CASELLA_VAIXELL) i (x2 + 1 < dim) fer
    x2:=x2+1;
fmentre
mentre (taula[x2][y2+1] = CASELLA_VAIXELL) i (y2 + 1 < dim) fer
    y2:=y2+1;
fmentre
si (x2+1 < dim) llavors
    x2:=x2+1;
fsi
si (y2+1 < dim) llavors
    y2:=y2+1;
fsi
per (i:=x1; i <= x2; i:=i+1) fer
    per (j:=y1; j <= y2; j:=j+1) fer
        si (taula[i][j] = CASELLA_BUIDA) llavors
            taula[i][j]:=CASELLA_AIGUA;
        fsi
    fper
fper
facció
$***** actualitza *****
acció actualitza (f:caràcter, c, res, dim:enter, VAR taula:taula de
caràcters) és
var
    x, y:enter;
fvar
inici
    x:=f-65;
    y:=c-1;
    opció (res)
        1:
            taula[x][y]:=CASELLA_AIGUA;
        2:
            taula[x][y]:=CASELLA_VAIXELL;
        3:
            taula[x][y]:=CASELLA_VAIXELL;
            descobrir_voltants (x, y, dim, taula);
    fopció
facció
$***** dispara_custom *****
funció dispara_custom (f:caràcter, c:enter, VAR t:taula de
caràcters) retorna enter és
var
    i, x, y:enter;
fvar

```



```

inici
  x:=f-65;
  y:=c-1;
  si (t[x][y] = CASELLA_AIGUA_TOCADA) o (t[x][y] =
CASELLA_VAIXELL_TOCAT) llavors
    i:=0;
  sino si (t[x][y] = CASELLA_AIGUA) llavors
    t[x][y]:=CASELLA_AIGUA_TOCADA;
    i:=1;
  sino si (t[x][y] = CASELLA_VAIXELL) llavors
    t[x][y]:=CASELLA_VAIXELL_TOCAT;
    i:=3;
  $ es posiciona a la caslla mes al nord-oest del vaixell
  mentre ((t[x-1][y] = CASELLA_VAIXELL) o (t[x-1][y] =
CASELLA_VAIXELL_TOCAT) i x-1 >= 0) fer
    x:=x-1;
  fmentre
  mentre ((t[x][y-1] = CASELLA_VAIXELL) o (t[x][y-1] =
CASELLA_VAIXELL_TOCAT) i y-1 >= 0) fer
    y:=y-1;
  fmentre
  $ segons si el vaixell esta en vertical) o (horitzontal abança
caslla per caslla, si detecta una caslla CASELLA_VAIXELL vol
dir que el vaixell encara esta per enfonsar per tant retorna 2
  mentre (t[x+1][y] = CASELLA_VAIXELL) o (t[x+1][y] =
CASELLA_VAIXELL_TOCAT) fmentre
    si (t[x][y] = CASELLA_VAIXELL) llavors
      i:=2;
    fsi
    x:=x+1;
  fmentre
  si (t[x][y] = CASELLA_VAIXELL) llavors
    i:=2;
  fsi
  mentre (t[x][y+1] = CASELLA_VAIXELL) o (t[x][y+1] =
CASELLA_VAIXELL_TOCAT) fer
    si (t[x][y] = CASELLA_VAIXELL) llavors
      i:=2;
    fsi
    y:=y+1;
  fmentre
  si (t[x][y] = CASELLA_VAIXELL) llavors
    i:=2;
  fsi
  sino
    i:=-1;
  fsi
  retorna i;
ffunció
$***** inicia_taulell_custom *****
acció inicia_taulell_custom (dim:enter, VAR taula:taula de
caràcters) és
var

```

```

x, y , i , j, cont:enter;
fin, hi_ha_lloc:booleà;
fvar
inici
    fin:=fals;
    mentre (no fin) i (dim >= 8) i (dim <= DIM_MAX) fer
    $ inicia el taulell
        per (i:=0; i < dim; i:=i+1) fer
            per (j:=0; j < dim; j:=j+1) fer
                taula[i][j]:=CASELLA_AIGUA;
            fper
        fper
    cont:=0;
    $ tria dues coordenades aleatòries i enteres per posar el vaixell
    de quatre caselles, també canvia les caselles que envolten el
    vaixell per aigua tocada, serà útil més endavant
    mentre (cont < 1) fer
        x:=aleatori mod dim;
        y:=aleatori mod dim;
        si (aleatori mod 2 = HORIZONTAL) i (taula[x][y] =
        CASELLA_AIGUA) i (taula[x+1][y] = CASELLA_AIGUA) i
        (taula[x+2][y] = CASELLA_AIGUA) i (taula[x+3][y] =
        CASELLA_AIGUA i x+3 < dim) llavors
            per (i:=x - 1; i <= x + 4; i++) fer
                per (j:=y - 1; j <= y + 1; j++) fer
                    si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                        taula[i][j]:=CASELLA_AIGUA_TOCADA;
                    fsi
                fper
            fper
            per (i:=0; i <= 3; i++) fer
                taula[x+i][y]:=CASELLA_VAIXELL;
            fper
            cont++;
        sino si (taula[x][y] = CASELLA_AIGUA) i (taula[x][y+1] =
        CASELLA_AIGUA) i (taula[x][y+2] = CASELLA_AIGUA) i
        (taula[x][y+3] = CASELLA_AIGUA) i (y+3 < dim) llavors
            per (i:=x - 1; i <= x + 1; i++) fer
                per (j:=y - 1; j <= y + 4; j++) fer
                    si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                        taula[i][j]:=CASELLA_AIGUA_TOCADA;
                    fsi
                fper
            fper
            per (i:=0; i <= 3; i++) fer
                taula[x][y+i]:=CASELLA_VAIXELL;
            fper
            cont++;
        fsi
    fmentre
    cont:=0;
    $ el mateix que abans, en aquest cas el loop es repeteix dues
    vegades, tan sols fica el vaixell si totes les caselles son

```

**CASELLA\_AIGUA, per tant no es tocara amb els altres vaixells  
ja que estan envoltats de CASELLA\_AIGUA\_TOCADA**

```
mentre (cont < 2) fer
  x:=aleatori mod dim;
  y:=aleatori mod dim;
  si (aleatori mod 2 = HORITZONTAL) i (taula[x][y] =
CASELLA_AIGUA) i (taula[x+1][y] = CASELLA_AIGUA) i
(taula[x+2][y] = CASELLA_AIGUA) i (x+2 < dim) llavors
    per (i:=x - 1; i <= x + 3; i++) fer
      per (j:=y - 1; j <= y + 1; j++) fer
        si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
          taula[i][j]:=CASELLA_AIGUA_TOCADA;
        fsi
      fper
    fper
  per (i:=0; i <= 2; i++) fer
    taula[x+i][y]:=CASELLA_VAIXELL;
  fper
  cont++;
sino si (taula[x][y] = CASELLA_AIGUA) i (taula[x][y+1] =
CASELLA_AIGUA) i (taula[x][y+2] = CASELLA_AIGUA) i (y+2 <
dim) llavors
  per (i:=x - 1; i <= x + 1; i++) fer
    per (j:=y - 1; j <= y + 3; j++) fer
      si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
        taula[i][j]:=CASELLA_AIGUA_TOCADA;
      fsi
    fper
  fper
  per (i:=0; i <= 2; i++) fer
    taula[x][y+i]:=CASELLA_VAIXELL;
  fper
  cont++;
fsi
fmentre
cont:=0;
$ en aquest cas comproba que hi hagi lloc, en cas contrari
modsiica la variable cont per a sortir del loop i tornar a
començar
```

```
mentre (cont < 3) fer
  hi_ha_lloc:=fals;
  per (i:=0; i < dim; i++) fer
    per (j:=0; j < dim; j++) fer
      si (taula[i][j] = CASELLA_AIGUA)
        hi_ha_lloc:=cert;
      fsi
    fper
  fper
  si (no hi_ha_lloc) llavors
    cont:=5;
  fsi
  x:=aleatori mod dim;
  y:=aleatori mod dim;
```

```

si (aleatori mod 2 = HORIZONTAL) i (taula[x][y] =
CASELLA_AIGUA) i (taula[x+1][y] = CASELLA_AIGUA) i (x+1 <
dim) llavors
    per (i:=x - 1; i <= x + 2; i++) fer
        per (j:=y - 1; j <= y + 1; j++) fer
            si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                taula[i][j]:=CASELLA_AIGUA_TOCADA;
            fsi
        fper
    fper
    per (i:=0; i <= 1; i++) fer
        taula[x+i][y]:=CASELLA_VAIXEL;
    fper
    cont++;
sino si (taula[x][y] = CASELLA_AIGUA) i (taula[x][y+1] =
CASELLA_AIGUA) i (y+1 < dim)
    per (i:=x - 1; i <= x + 1; i++) fer
        per (j:=y - 1; j <= y + 2; j++) fer
            si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                taula[i][j]:=CASELLA_AIGUA_TOCADA;
            fsi
        fper
    fper
    per (i:=0; i <= 1; i++) fer
        taula[x][y+i]:=CASELLA_VAIXEL;
    fper
    cont++;
fsi
fmentre
cont:=0;
mentre (cont < 4) fer
    hi_ha_lloc:=fals;
    per (i:=0; i < dim; i++) fer
        per (j:=0; j < dim; j++) fer
            si (taula[i][j] = CASELLA_AIGUA) llavors
                hi_ha_lloc:=cert;
            fsi
        fper
    fper
    si (no hi_ha_lloc) llavors
        cont:=5;
    fsi
    x:=aleatori mod dim;
    y:=aleatori mod dim;
    si (taula[x][y] = CASELLA_AIGUA) llavors
        per (i:=x - 1; i <= x + 1; i++) fer
            per (j:=y - 1; j <= y + 1; j++) fer
                si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                    taula[i][j]:=CASELLA_AIGUA_TOCADA;
                fsi
            fper
        fper
        taula[x][y]:=CASELLA_VAIXEL;

```

```

        cont++;
    fsi
fmentre
$ si hem arribat aqui i cont:=4 vol dir que s'han ficat tots
els vaixells per tant final:=cert
si (cont = 4) llavors
    fin:=cert;
fsi
$ canvia les caslles d'aigua tocada per aigua
per (i:=0; i < dim; i++) fer
    per (j:=0; j < dim; j++) fer
        si (taula[i][j] = CASELLA_AIGUA_TOCADA)
            taula[i][j]:=CASELLA_AIGUA;
        fsi
    fper
fper
fmentre
facció
$***** imp_resultat *****
acció imp_resultat (res:enter) és
inci
    opció (res)
        ERROR:
            escriure ("Error");
        REPETIT:
            escriure ("Repetit");
        AIGUA:
            escriure ("Aigua");
        TOCAT:
            escriure ("Tocat");
        ENFONSAT:
            escriure ("Tocat i enfonsat");
        en altre cas:
    fopció
facció
$***** continuar *****
funció continuar () retorna booleà és
var
    i:enter;
fvar
inci
    escriure ("Continuar? (0/1): ");
    mentre (llegir (i) != 1) o ((i != 0) i (i != 1)) fer
        escriure ("Error! Continuar? (0/1): ");
    fmentre
    retorna i;
ffunció
$***** jugar_partida *****
funció jugar_partida (dim:enter, num_jugadors:enter, VAR
jugadors:taula de jugador_tipus, VAR num_records:enter, VAR
records:taula de record_tipus) retorna booleà és
var
    pts:enter;

```

```

    retorn:booleà;
fvar
inici
    retorn:=cert;
    imprimir_taules (dim, jugadors[0].vaixells,
jugadors[0].llancaments);
    escriure ("Jugades: ", jugadors[0].num_trets);
    escriure ("0:=NO, 1:=SI");
    mentre (jugadors[0].vaixells_enfonsats < 10) i (continuar()) fer
        si (num_jugadors = 0) llavors
            jugar (jugadors[0].coor.lletra, jugadors[0].coor.nombre,
dim, jugadors[0].llancaments);
        fsi
        si (num_jugadors = 1) llavors
            escriure ("Fila [A-"dim+64"] Columna [1-"dim"]): ");
            nova_jugada(dim, jugadors[0].coor);
        fsi
        jugadors[0].resultat:=dispara_custom (jugadors[0].coor.lletra,
jugadors[0].coor.nombre, jugadors[0].vaixells);
        actualitza (jugadors[0].coor.lletra, jugadors[0].coor.nombre,
jugadors[0].resultat, dim, jugadors[0].llancaments);
        jugadors[0].sum_resultats += jugadors[0].resultat - 1;
        si (jugadors[0].resultat = ENFONSAT) llavors
            jugadors[0].vaixells_enfonsats++;
            jugadors[0].num_trets++;
        fsi
        imprimir_taules (dim, jugadors[0].vaixells,
jugadors[0].llancaments);
        escriure ("Coordenades anteriors: ", jugadors[0].coor.lletra,
jugadors[0].coor.nombre);
        escriure ("Resultat: ");
        imp_resultat(jugadors[0].resultat);
        escriure ("Jugades: ", jugadors[0].num_trets);
    fmentre
    si (jugadors[0].vaixells_enfonsats = 10) llavors
        escriure ("Partida finalitzada! Prem enterro per veure la
puntuació");
        borra_joc (FITXER_JOC);
        retorn:=fals;
        pts:=calcular_puntuacio(jugadors[0].num_trets,
jugadors[0].sum_resultats, dim);
        escriure ("Puntuació: ", pts);
        si (num_records = MAX_RECORDS) i
            (pts > records[num_records-1].punts) llavors
            records[num_records-1].punts:=pts;
            copiar_cadena (records[num_records-1].nom,
jugadors[0].nom);
        sino si (num_records < MAX_RECORDS) llavors
            records[num_records].punts:=pts;
            copiar_cadena (records[num_records].nom, jugadors[0].nom);
            num_records++;
        fsi

```

```

        ordena_records (num_records, records);
        emmagatzema_records (FITXER_RECORDS, num_records, records);
        veure_records (10, num_records, records);
    fsi
    retorna retorn;
ffunció
$***** jugar_partida_dos *****
funció jugar_partida_dos (dim:enter, VAR jugadors:taula de
jugador_tipus, VAR num_records:enter, VAR records:taula de
record_tipus) retorna booleà és
var
    pts, guanyador:enter;
    retorn:booleà;
fvar
inici
    retorn:=cert;
    imprimir_taules (dim, jugadors[1].vaixells,
jugadors[1].llancaments);
    si (jugadors[0].coor.nombre >= 1) i (jugadors[0].coor.nombre <=
dim) i (jugadors[0].coor.lletra >= 64) i (jugadors[0].coor.lletra
<= dim+64) llavors
        escriure ("Coordenades anteriors: ", jugadors[1].coor.lletra,
jugadors[1].coor.nombre);
        escriure ("Resultat: ");
        imp_resultat(jugadors[1].resultat);
    fsi
    si (jugadors[0].coor.nombre >= 1) i (jugadors[0].coor.nombre <=
dim i jugadors[0].coor.lletra >= 64) i (jugadors[0].coor.lletra
<= dim+64) llavors
        escriure ("La maquina dispara a les coordenades: ",
jugadors[0].coor.lletra, jugadors[0].coor.nombre);
        escriure ("Resultat: ");
        imp_resultat(jugadors[0].resultat);
    fsi
    escriure ("0:=NO, 1:=SI");
    mentre (jugadors[0].vaixells_enfonsats < 10) i
(jugadors[1].vaixells_enfonsats < 10) i (continuar()) fer
        $ torn jugador
        si (jugadors[0].resultat != 2 i jugadors[0].resultat != 3)
            llavors
                escriure ("Fila [A-"dim+64"] Columna [1-"dim"] : ");
                nova_jugada(dim, jugadors[1].coor);
                jugadors[1].resultat:=dispara_custom
                (jugadors[1].coor.lletra, jugadors[1].coor.nombre,
jugadors[0].vaixells);
                actualitza (jugadors[1].coor.lletra,
jugadors[1].coor.nombre, jugadors[1].resultat, dim,
jugadors[1].llancaments);
                jugadors[1].sum_resultats += jugadors[1].resultat - 1;
                si (jugadors[1].resultat = ENFONSAT) llavors
                    jugadors[1].vaixells_enfonsats++;
                fsi
                jugadors[1].num_trets++;

```

```

fsi
$ torn maquina
si (jugadors[1].resultat != 2 i jugadors[1].resultat != 3)
llavors
    jugar (jugadors[0].coor.lletra, jugadors[0].coor.nombre,
    dim, jugadors[0].llancaments);
    jugadors[0].resultat:=dispara_custom
    (jugadors[0].coor.lletra, jugadors[0].coor.nombre,
    jugadors[1].vaixells);
    actualitza (jugadors[0].coor.lletra,
    jugadors[0].coor.nombre, jugadors[0].resultat, dim,
    jugadors[0].llancaments);
    jugadors[0].sum_resultats += jugadors[0].resultat - 1;
    si (jugadors[0].resultat = ENFONSAT) llavors
        jugadors[0].vaixells_enfonsats++;
    fsi
    jugadors[0].num_trets++;
fsi
$ imprimir resultats
imprimir_taules (dim, jugadors[1].vaixells,
jugadors[1].llancaments);
si (jugadors[0].coor.nombre >= 1 i jugadors[0].coor.nombre <=
dim i jugadors[0].coor.lletra >= 64 i jugadors[0].coor.lletra
<= dim+64) llavors
    escriure ("Coordenades anteriors: ",
    jugadors[1].coor.lletra, jugadors[1].coor.nombre);
    escriure ("Resultat: ");
    imp_resultat(jugadors[1].resultat);
fsi
si (jugadors[0].coor.nombre >= 1 i jugadors[0].coor.nombre <=
dim i jugadors[0].coor.lletra >= 64 i jugadors[0].coor.lletra
<= dim+64) llavors
    escriure ("La maquina dispara a les coordenades: ",
    jugadors[0].coor.lletra, jugadors[0].coor.nombre);
    escriure ("Resultat: ");
    imp_resultat(jugadors[0].resultat);
fsi
fmentre
si (jugadors[0].vaixells_enfonsats = 10) o
(jugadors[1].vaixells_enfonsats = 10) llavors
    escriure ("Partida finalitzada!Prem enterro per veure la
    puntuació . . .");
    borra_joc (FITXER_JOC);
    retorn:=fals;
si (jugadors[0].vaixells_enfonsats = 10) llavors
    guanyador:=0;
fsi
si (jugadors[1].vaixells_enfonsats = 10) llavors
    guanyador:=1;
fsi
pts:=calcular_puntuacio (jugadors[guanyador].num_trets,
jugadors[guanyador].sum_resultats, dim);
escriure ("Guanyador: ", jugadors[guanyador].nom);

```



```

    escriure ("Puntuació: ", pts);
    si (num_records = MAX_RECORDS i pts > records[num_records-
1].punts) llavors
        records[num_records-1].punts:=pts;
        copiar_cadena (records[num_records-1].nom,
            jugadors[guanyador].nom);
    sino si (num_records < MAX_RECORDS) llavors
        records[num_records].punts:=pts;
        copiar_cadena (records[num_records].nom,
            jugadors[guanyador].nom);
        num_records++;
    fsi
ordena_records (num_records, records);
emmagatzema_records (FITXER_RECORDS, num_records, records);
veure_records (10, num_records, records);
fsi
retorna retorn;
ffunció
$***** calcular_puntuacio *****
funció calcular_puntuacio (jugades, resultats, dim:enter) retorna
enter és
inci
    retorna ( dim div jugades ) * resultats * 100;
ffunció
$***** veure_records *****
acció veure_records (num:enter, num_records:enter, records:taula de
record_tipus) és
var
    i:enter;
fvar
inici
    si (num_records > 0) llavors
        i:=0;
        escriure ("Jugador\t\tPuntuació");
        mentre (i < num) i (i < num_records) fer
            escriure (i+1 records[i].nom"\t\t"records[i].punts);
            i++;
        fmentre
    sino
        escriure ("No hi ha rècords per veure.");
    fsi
facció
$***** ordena_records *****
acció ordena_records (num_records:enter, VAR records:taula de
record_tipus) és
var
    i, j, temp:enter;
    ctemp:nom_tipus;
fvar
inici
    per (i:=1; i < num_records; i:=i+1) fer
        j:=i;
        mentre (j > 0 i records[j].punts > records[j-1].punts) fer

```

```

        temp:=records[j].punts;
        records[j].punts:=records[j-1].punts;
        records[j-1].punts:=temp;
        copiar_cadena (ctemp, records[j].nom);
        copiar_cadena (records[j].nom, records[j-1].nom);
        copiar_cadena (records[j-1].nom, ctemp);
        j:=j-1;
    fmentre
fper
facció
$***** jugar *****
acció jugar (var f:caràcter, var c:enter, dim:enter,
taulell_llancament:taula de caràcters) és
var
    i, j, port, dest, drag, pmax:enter;
    fin:booleà;
    prob:taula [DIM_MAX][DIM_MAX] de caràcters;
fvar
inici
    i:=0;
    j:=0;
    port:=1;
    dest:=2;
    drag:=3;
    pmax:=0;
    fin:=fals;
$ recorre totes les caslles buscan un vaixell que estigue per
enfonsar
mentre (i<dim) i (no fin) fer
    j:=0;
    mentre (j<dim) i (no fin) fer
        si (taula[i][j] = CASELLA_VAIXELL) i (((i-1>-1) i
        (taula[i-1][j] = CASELLA_BUIDA) o (j-1>-1) i
        (taula[i][j-1] = CASELLA_BUIDA) o (i+1<dim) i
        (taula[i+1][j] = CASELLA_BUIDA) o (j+1<dim) i
        (taula[i][j+1] = CASELLA_BUIDA))) llavors
            $ comproba si el vaixell esta en posicio horitzontal,
vertical o si no es sap tria un punt cardinal aleatori
on disparar
            si ((i-1>-1) i (taula[i-1][j] = CASELLA_VAIXELL) o
            (i+1<dim) i (taula[i+1][j] = CASELLA_VAIXELL)) llavors
                si (i-1>-1) i (taula[i-1][j] = CASELLA_BUIDA)
                    llavors
                        f:=i;
                        c:=j + 1;
                        fin:=cert;
            sino
                mentre (taula[i][j] = CASELLA_VAIXELL) fer
                    i++;
            fmente

```

```

        f:=i + 1;
        c:=j + 1;
        fin:=cert;
    fsi
sino si ((j-1>-1) i (taula[i][j-1] = CASELLA_VAIXELL) o
(j+1<dim) i (taula[i][j+1] = CASELLA_VAIXELL)) llavors
    si (j-1>-1) i (taula[i][j-1] = CASELLA_BUIDA)
    llavors
        f:=i + 1;
        c:=j;
        fin:=cert;
sino
    mentre (taula[i][j] = CASELLA_VAIXELL) fer
        j++;
    fmente
        f:=i + 1;
        c:=j + 1;
        fin:=cert;
    fsi
sino
    mentre (no fin) fer
        opció (aleatori mod 4)
        0:
            si (i-1>-1) i (taula[i-1][j] = CASELLA_BUIDA)
            llavors
                f:=i;
                c:=j + 1;
                fin:=cert;
            fsi
        1:
            si (j-1>-1) i (taula[i][j-1] = CASELLA_BUIDA)
            llavors
                f:=i + 1;
                c:=j;
                fin:=cert;
            fsi
        2:
            si (i+1<dim) i (taula[i+1][j] = CASELLA_BUIDA)
            llavors
                f:=i + 2;
                c:=j + 1;
                fin:=cert;
            fsi
        3:
            si (j+1<dim) i (taula[i][j+1] = CASELLA_BUIDA)
            llavors

```

```

                                f:=i + 1;
                                c:=j + 2;
                                fin:=cert;
                                fsi
                                fopció
                                fmente
                                fsi
                                fsi
                                j++;
                                fmente
                                i++;
                                fmente
$ si no hi ha cap vaixell penden d'enfonsar
si (no fin) llavors
    $ inicia la taula de probabilitats a 0
    per (i:=0; i < dim; i++) fer
        per (j:=0; j < dim; j++) fer
            prob[i][j]:=0;
        fper
    fper
$ identifica quins vaixells estan enfonsats i quins queden per enfonsar
per (i:=0; i < dim; i++) fer
    per (j:=0; j < dim; j++) fer
        si (taula[i][j] = CASELLA_VAIXELL) llavors
            si (i+1 < dim) i (taula[i+1][j] = CASELLA_VAIXELL)
                llavors
                    si (i+2 < dim) i
                        (taula[i+2][j] = CASELLA_VAIXELL) llavors
                            si (i+3 < dim) i
                                (taula[i+3][j] = CASELLA_VAIXELL) llavors
                                    port--;
                                sino si (i-1 < 0) o ((i-1 > -1) i
                                    (taula[i-1][j] != CASELLA_VAIXELL)) llavors
                                        dest--;
                                fsi
                            sino si (i-1 < 0) o ((i-1 > -1) i
                                (taula[i-1][j] != CASELLA_VAIXELL)) llavors
                                    drag--;
                                fsi
                    sino si (j+1 < dim) i
                        (taula[i][j+1] = CASELLA_VAIXELL) llavors
                            si (j+2 < dim i taula[i][j+2] = CASELLA_VAIXELL)
                                llavors
                                    si (j+3 < dim) i
                                        (taula[i][j+3] = CASELLA_VAIXELL) llavors
                                            port--;
                                        sino si (j-1 < 0) o ((j-1 > -1) i

```

```

        (taula[i][j-1] != CASELLA_VAIXELL)) llavors
            dest--;
        fsi
    sino si (j-1 < 0) o ((j-1 > -1) i
        (taula[i][j-1] != CASELLA_VAIXELL)) llavors
            drag--;
        fsi
    fsi
fper
per (i:=0; i < dim; i++) fer
    per (j:=0; j < dim; j++) fer
        si (port > 0) llavors
            si (taula[i][j] = CASELLA_BUIDA) i
                (taula[i+1][j] = CASELLA_BUIDA i i+1 < dim) i
                (taula[i+2][j] = CASELLA_BUIDA i i+2 < dim) i
                (taula[i+3][j] = CASELLA_BUIDA i i+3 < dim) llavors
                    prob[i][j]++;
                    prob[i+1][j]++;
                    prob[i+2][j]++;
                    prob[i+3][j]++;
            fsi
            si (taula[i][j] = CASELLA_BUIDA) i
                (taula[i][j+1] = CASELLA_BUIDA i j+1 < dim) i
                (taula[i][j+2] = CASELLA_BUIDA i j+2 < dim) i
                (taula[i][j+3] = CASELLA_BUIDA i j+3 < dim) llavors
                    prob[i][j]++;
                    prob[i][j+1]++;
                    prob[i][j+2]++;
                    prob[i][j+3]++;
            fsi
        fsi
    si (dest > 0) llavors
        si (taula[i][j] = CASELLA_BUIDA) i
            (taula[i+1][j] = CASELLA_BUIDA) i (i+1 < dim) i
            (taula[i+2][j] = CASELLA_BUIDA i i+2 < dim) llavors
                prob[i][j]++;
                prob[i+1][j]++;
                prob[i+2][j]++;
            fsi
        si (taula[i][j] = CASELLA_BUIDA) i
            (taula[i][j+1] = CASELLA_BUIDA) i (j+1 < dim) i
            (taula[i][j+2] = CASELLA_BUIDA i j+2 < dim) llavors
                prob[i][j]++;
                prob[i][j+1]++;
                prob[i][j+2]++;
            fsi
        fsi
    fsi

```

**\$ calcula les probabilitats de cada caslla segons els vaixells que quedin per enfonsar**

```

        si (drag > 0) llavors
            si (taula[i][j] = CASELLA_BUIDA) i
                (taula[i+1][j] = CASELLA_BUIDA) i (i+1 < dim)
            llavors
                prob[i][j]++;
                prob[i+1][j]++;
            fsi
            si (taula[i][j] = CASELLA_BUIDA) i
                (taula[i][j+1] = CASELLA_BUIDA) i (j+1 < dim)
            llavors
                prob[i][j]++;
                prob[i][j+1]++;
            fsi
        fsi
    fper
fper
$ busca la probabilitat mes alta i la guarda a la variable
pmax
per (i:=0; i < dim; i++) fer
    per (j:=0; j < dim; j++) fer
        si (prob[i][j] > pmax) llavors
            pmax:=prob[i][j];
        fsi
    fper
fper
$ tria una coordenada aleatoria entre les caslles que tingin
la probabilitat mes alta
mentre (f < 1) o (c < 1) o (f > dim) o (c > dim) o
(taula[f-1][c-1] != CASELLA_BUIDA) o (prob[f-1][c-1] != pmax)
fer
    c:=aleatori mod dim + 1;
    f:=aleatori mod dim + 1;
fmentre
f:=f+64;
fsi
facció
$***** emmagatzema_records *****
funció emmagatzema_records (fitxer_record:taula de caràcters, var
num_records:enter, records:taula de record_tipus) retorna booleà és
var
    f:fitxer;
    retorn:booleà;
    i:enter;
fvar
inici
    f:=obrir(fitxer_record, "w");
    si (f = NULL) llavors
        retorn:=fals;
    sino
        per (i:=0; i < *num_records; i++) fer
            escriuref (f, "\t", records[i].nom, records[i].punts);

```

```

        fper
        tancar(f);
        retorn:=cert;
    fsi
    retorna retorn;
ffunció
$***** recupera_records *****
funció recupera_records (fitxer_record:taula de caràcters, var
num_records:enter, records:taula de record_tipus) retorna booleà és
var
    f:fitxer;
    retorn:booleà;
fvar
inici
    f:=obrir(fitxer_record, "r");
    num_records:=0;
    si (f = NULL) llavors
        retorn:=fals;
    sino
        mentre (!feof(f) i num_records < MAX_RECORDS) fer
            si (llegirf (f, records[num_records].nom,
records[num_records].punts) = 2) llavors
                num_records++;
            fsi
        fmentre
        tancar(f);
        retorn:=cert;
    fsi
    retorna retorn;
ffunció
$***** emmagatzema_joc *****
funcio emmagatzema_joc (nom_fitxer:taula de caràcters, dim,
num_jugadors: enter, jugadors:taula de jugador_tipus) retorna booleà
és
var
    f:fitxer;
    retorn:booleà;
fvar
inici
    f:=obrir(fitxer_record, "wb");
    si (f = NULL) llavors
        retorn:=fals;
    sino
        escriuref (dim, 1, f);
        escriuref (num_jugadors, 1, f);
        si (num_jugadors = 0) o (num_jugadors = 1) llavors
            escriuref (jugadors , 1, f);
        fsi
        si (num_jugadors = 2) llavors
            escriuref (jugadors, 2, f);
        fsi
        tancar(f);
        retorn:=cert;

```

```

    fsi
    retorna retorn;
ffunció
$***** recupera_joc *****
funcio recupera_joc (nom_fitxer:taula de caràcters, var dim, var
num_jugadors: enter, var jugadors:taula de jugador_tipus) retorna
booleàea és
var
    f:fitxer;
    retorn:booleà;
fvar
inici
    f:=obrir(fitxer_record, "rb");
    retorn:booleà;
    si (f = NULL) llavors
        retorn:=fals;
    sino
        llegirf (dim, 1 , f);
        llegirf (num_jugadors, 1 , f);
        si (num_jugadors = 0) o (*num_jugadors = 1) llavors
            llegirf (jugadors, 1, f);
        fsi
        si (num_jugadors = 2) llavors
            llegirf (jugadors, 2, f);
        fsi
        tancar(f);
        retorn:=cert;
    fsi
    retorna retorn;
ffunció
$***** inicia_elmeu_taulell *****
accio inicia_elmeu_taulell (dim:enter, VAR taulell_vaixells:taula de
caràcters) és
var
    i, j, x, y, orientacio, cont, cambi:enter;
    fin, hi_ha_lloc:booleà;
    coor:coor_tipus;
fvar
inici
    fin:=fals;
    hi_ha_lloc:=cert;
    $ el mateix que inicia_taulell_custom pero preguntan les
    coordenades i l'orientacio en comptes de generar-ho aleatoriament

mentre (no fin i dim >= 8 i dim <= DIM_MAX) fer
    per (i:=0; i < dim; i++) fer
        per (j:=0; j < dim; j++) fer
            taulell_vaixells[i][j]:=CASELLA_AIGUA;
        fper
    fper
    cont:=0;
    imprimir_taula (dim, taulell_vaixells);
    escriure ("Vaixell: Portaavions (4 caslles)");

```



```

escriure("Coordenades Fila [A-"dim+64"] Columna [1-"dim"]): ");
nova_jugada (dim, coor);
escriure ("Orientació, Horitzontal ["HORITZONTAL"] Vertical
["VERTICAL"]): ");
nova_orientacio (orientacio);
x:=coor.lletra - 65;
y:=coor.nombre - 1;
mentre (cont < 1) fer
si (orientacio = HORITZONTAL) i
(taulell_vaixells[x][y] = CASELLA_AIGUA) i
(taulell_vaixells[x+1][y] = CASELLA_AIGUA) i
(taulell_vaixells[x+2][y] = CASELLA_AIGUA) i
(taulell_vaixells[x+3][y] = CASELLA_AIGUA) i (x+3 < dim)
llavors
  per (i:=x - 1; i <= x + 4; i++) fer
    per (j:=y - 1; j <= y + 1; j++) fer
      si (i >= 0) i (i < dim i j >= 0) i (j < dim) llavors
        taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
      fsi
    fper
  fper
per (i:=0; i <= 3; i++) fer
  taulell_vaixells[x+i][y]:=CASELLA_VAIXELL;
fper
cont++;
sino si (taulell_vaixells[x][y] = CASELLA_AIGUA) i
(taulell_vaixells[x][y+1] = CASELLA_AIGUA) i
(taulell_vaixells[x][y+2] = CASELLA_AIGUA) i
(taulell_vaixells[x][y+3] = CASELLA_AIGUA) i (y+3 < dim)
llavors
  per (i:=x - 1; i <= x + 1; i++) fer
    per (j:=y - 1; j <= y + 4; j++) fer
      si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
        taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
      fsi
    fper
  fper
per (i:=0; i <= 3; i++) fer
  taulell_vaixells[x][y+i]:=CASELLA_VAIXELL;
fper
cont++;
sino
  imprimir_taula (dim, taulell_vaixells);
  escriure ("Posició no vàlida!");
  escriure ("Coordenades Fila [A-%c] Columna [1-]: ",
dim+64, dim);
  nova_jugada (dim, coor);
  escriure ("Orientació (:=Horitzontal, :=Vertical): ",
HORITZONTAL, VERTICAL);
  nova_orientacio (orientacio);
  x:=coor.lletra - 65;

```

```

        y:=coor.nombre - 1;
    fsi
fmente
cambi:=cont;
cont:=0;
mentre (cont < 2) fer
    si (cambi != cont) llavors
        cambi:=cont;
        imprimir_taula (dim, taulell_vaixells);
        escriure ("Vaixell: Destructor (3 caslles)");
        escriure ("Coordenades, Fila [A-"dim+64"] Columna [1-
        "dim"] : ");
        nova_jugada (dim, coor);
        escriure ("Orientació, Horitzontal ["HORITZONTAL"]
        Vertical ["VERTICAL"] : ");
        nova_orientacio (orientacio);
        x:=coor.lletra - 65;
        y:=coor.nombre - 1;
    si (orientacio = HORITZONTAL) i (taulell_vaixells[x][y] =
    CASELLA_AIGUA) i (taulell_vaixells[x+1][y] = CASELLA_AIGUA)
    i (taulell_vaixells[x+2][y] = CASELLA_AIGUA) i (x+2 < dim)
    llavors
        per (i:=x - 1; i <= x + 3; i++) fer
            per (j:=y - 1; j <= y + 1; j++) fer
                si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                    taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
                fsi
            fper
        fper
        per (i:=0; i <= 2; i++) fer
            taulell_vaixells[x+i][y]:=CASELLA_VAIXEL;
        fper
        cont++;
    sino si (taulell_vaixells[x][y] = CASELLA_AIGUA) i
    (taulell_vaixells[x][y+1] = CASELLA_AIGUA) i
    (taulell_vaixells[x][y+2] = CASELLA_AIGUA) i (y+2 < dim)
    llavors
        per (i:=x - 1; i <= x + 1; i++) fer
            per (j:=y - 1; j <= y + 3; j++) fer
                si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                    taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
                fsi
            fper
        fper
        per (i:=0; i <= 2; i++) fer
            taulell_vaixells[x][y+i]:=CASELLA_VAIXEL;
        fper
        cont++;
    sino
        imprimir_taula (dim, taulell_vaixells);
        escriure ("Posició no vàlida!");

```

```

        escriure ("Coordenades, Fila [A-"dim+64"] Columna [1-
        "dim"]: ");
        nova_jugada (dim, coor);
        escriure ("Orientació, Horitzontal ["HORITZONTAL"]
        Vertical ["VERTICAL"]: ");
        nova_orientacio (orientacio);
        x:=coor.lletra - 65;
        y:=coor.nombre - 1;
    fsi
fmentre
cambi:=cont;
cont:=0;
mentre (cont < 3 i hi_ha_lloc) fer
    si (cambi != cont) llavors
        cambi:=cont;
        imprimir_taula (dim, taulell_vaixells);
        escriure ("Vaixell: Dragamines (2 caslles)");
        escriure ("Coordenades, Fila [A-"dim+64"] Columna [1-
        "dim"]: ");
        nova_jugada (dim, coor);
        escriure ("Orientació, Horitzontal ["HORITZONTAL"]
        Vertical ["VERTICAL"]: ");
        nova_orientacio (orientacio);
        x:=coor.lletra - 65;
        y:=coor.nombre - 1;
    fsi
    si (orientacio = HORITZONTAL) i
    (taulell_vaixells[x][y] = CASELLA_AIGUA) i
    (taulell_vaixells[x+1][y] = CASELLA_AIGUA i x+1 < dim)
    llavors
        per (i:=x - 1; i <= x + 2; i++) fer
            per (j:=y - 1; j <= y + 1; j++) fer
                si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                    taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
            fsi
        fper
    fper
    per (i:=0; i <= 1; i++) fer
        taulell_vaixells[x+i][y]:=CASELLA_VAIXELL;
    fper
    cont++;
sino si (taulell_vaixells[x][y] = CASELLA_AIGUA) i
    (taulell_vaixells[x][y+1] = CASELLA_AIGUA) i (y+1 < dim)
    llavors
        per (i:=x - 1; i <= x + 1; i++) fer
            per (j:=y - 1; j <= y + 2; j++) fer
                si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                    taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
            fsi
        fper
    fper

```

```

        fper
        per (i:=0; i <= 1; i++) fer
            taulell_vaixells[x][y+i]:=CASELLA_VAIXELL;
        fper
        cont++;
    sino
        imprimir_taula (dim, taulell_vaixells);
        escriure ("Posició no vàlida!");
        escriure ("Coordenades, Fila [A-"dim+64"] Columna [1-
            "dim"]: ");
        nova_jugada (dim, coor);
        escriure ("Orientació, Horitzontal ["HORITZONTAL"]
            Vertical ["VERTICAL"]: ");
        nova_orientacio (orientacio);
        x:=coor.lletra - 65;
        y:=coor.nombre - 1;
    fsi
    hi_ha_lloc:=fals;
    per (i:=0; i < dim; i++) fer
        per (j:=0; j < dim; j++) fer
            si (taulell_vaixells[i][j] = CASELLA_AIGUA) llavors
                hi_ha_lloc:=cert;
            fsi
        fper
    fper
    fmentres
    cambi:=cont;
    cont:=0;
    mentre (cont < 4 i hi_ha_lloc) fer
        si (cambi != cont) llavors
            cambi:=cont;
            imprimir_taula (dim, taulell_vaixells);
            escriure ("Vaixell: Submar\241 (1 caslla)");
            escriure ("Coordenades, Fila [A-"dim+64"] Columna [1-
                "dim"]: ");
            nova_jugada (dim, coor);
            x:=coor.lletra - 65;
            y:=coor.nombre - 1;
        fsi
        si (taulell_vaixells[x][y] = CASELLA_AIGUA) llavors
            per (i:=x - 1; i <= x + 1; i++) fer
                per (j:=y - 1; j <= y + 1; j++) fer
                    si (i >= 0 i i < dim i j >= 0 i j < dim) llavors
                        taulell_vaixells[i][j]:=CASELLA_AIGUA_TOCADA;
                    fsi
                fper
            fper
            taulell_vaixells[x][y]:=CASELLA_VAIXELL;
            cont++;
        sino
            imprimir_taula (dim, taulell_vaixells);

```

```

        escriure ("Posició no vàlida!");
        escriure ("Coordenades, Fila [A-"dim+64"] Columna [1-
        "dim"]: ");
        nova_jugada (dim, coor);
        x:=coor.lletra - 65;
        y:=coor.nombre - 1;
    fsi
    hi_ha_lloc:=fals;
    per (i:=0; i < dim; i++) fer
        per (j:=0; j < dim; j++) fer
            si (taulell_vaixel·ls[i][j] = CASELLA_AIGUA) llavors
                hi_ha_lloc:=cert;
            fsi
        fper
    fper
    fmentre
    si no hi_ha_lloc llavors
        imprimir_taula (dim, taulell_vaixel·ls);
        escriure ("No hi ha lloc! Torna a començar");
    sino
        per (i:=0; i < dim; i++) fer
            per (j:=0; j < dim; j++) fer
                si (taulell_vaixel·ls[i][j] = CASELLA_AIGUA_TOCADA)
                    llavors
                        taulell_vaixel·ls[i][j]:=CASELLA_AIGUA;
                    fsi
            fper
        fper
        fin:=cert;
        imprimir_taula (dim, taulell_vaixel·ls);
        escriure ("Taula completa!");
    fsi
    fmentre
facció

```

### 3.3 Disseny del joc de proves

Farem un joc de proves per a cada funció:

menu\_principal, crea\_joc

Cas	Descripció	Entrada	Sortida teórica
1	Valors per sobre i per sota de les opcions possibles.	0, -325, 7, 346	error
2	Caracters aleatoris.	a, fifwf, -dq&	error
3	Opcions correctes	1, 2, 5, 6	ok
4	Si es carrega un joc opcions correctes	3, 4	ok
5	Si no hi ha joc opcions no disponible	3, 4	error

nova\_jugada

Cas	Descripció	Entrada	Sortida teórica
1	Nombres	0, -325, 7, 346	error
2	Caracters aleatoris.	a, fifwf, -dq&	error
3	Opcions correctes	A1, B1, J10	ok
4	Opcions incorrectes	J11, K1	error
5	Si la dimensio es diferen de 10	J10, J9, H10, I10	error

nova\_orientacio, continuar

Cas	Descripció	Entrada	Sortida teórica
1	Valors per sobre i per sota de les opcions possibles.	-1, -325, 2, 346	error
2	Caracters aleatoris.	a, fifwf, -dq&	error
3	Opcions correctes	0, 1	ok

inicia\_taulell\_custom

Cas	Descripció	Entrada	Sortida teórica
1	Dimensions per sota de 8 les ignora	0, 1, 7	no entra en bucle

recupera\_records

Cas	Descripció	Entrada	Sortida teórica
1	archiu de text en mes de 25 records	records.txt	domes llegeix MAX_RECO RDS records
2	archiu vuit	records.txt	num_records = 0
3	no hi ha archiu		retorna fals

recupera\_joc

Cas	Descripció	Entrada	Sortida teórica
1	Recupera el joc correctamen en mode 0 jugador		si
2	Recupera el joc correctamen en mode 1 jugador		si
3	Recupera el joc correctamen en mode 2 jugador		si
4	no hi ha archiu		retorna fals

inicia\_elmeu\_taulell

Cas	Descripció	Entrada	Sortida teórica
1	No permet que dos vaixells es toquin		si
2	Si no hi ha lloc torna a començar		si
3	Genera taulells valids		si

## 4 Avaluació

A l'avaluació hem de detallar els resultats de fer les proves especificades al disseny i justificar els resultats reals obtinguts.

menu\_principal, crea\_joc

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	Valors per sobre i per sota de les opcions possibles.	0, -325, 7, 346	Error	Error	Ok
2	Caracters aleatoris.	a, fifwf, -dq&	Error	Error	Ok
3	Opcions correctes	1, 2, 5, 6	Ok	Ok	Ok
4	Si es carrega un joc opcions correctes	3, 4	Ok	Ok	Ok
5	Si no hi ha joc opcions no disponible	3, 4	Error	Ok	Ok

nova\_jugada

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	Nombres	0, -325, 7, 346	Error	Error	Ok
2	Caracters aleatoris.	a, fifwf, -dq&	Error	Error	Ok
3	Opcions correctes	A1, B1, J10	Ok	Ok	Ok
4	Opcions incorrectes	J11, K1	Error	Error	Ok
5	Si la dimensio es diferen de 10	J10, J9, H10, I10	Error	Error	Ok

nova\_orientacio, continuar

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	Valors per sobre i per sota de les opcions possibles.	-1, -325, 2, 346	Error	Error	Ok
2	Caracters aleatoris.	a, fifwf, -dq&	Error	Error	Ok
3	Opcions correctes	0, 1	Ok	Ok	Ok

inicia\_taulell\_custom

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	Dimensions per sota de 8 les ignora	0, 1, 7	No entra en bucle	No entra en bucle	Ok



## recupera\_records

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	archiu de text en mes de 25 records	records.txt	Domes llegeix MAX_RECO RDS records	Domes llegeix MAX_RECO RDS records	Ok
2	archiu vuit	records.txt	num_records = 0	num_records = 0	Ok
3	no hi ha archiu		Retorna fals	Retorna fals	Ok

## recupera\_joc

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	Recupera el joc correctamen en mode 0 jugador		Ok	Ok	Ok
2	Recupera el joc correctamen en mode 1 jugador		Ok	Ok	Ok
3	Recupera el joc correctamen en mode 2 jugador		Ok	Ok	Ok
4	no hi ha archiu		Retorna fals	Retorna fals	Ok

## inicia\_elmeu\_taulell

Cas	Descripció	Entrada	Sortida teórica	Sortida real	Resultat
1	No permet que dos vaixells es toquin		Si	Si	Ok
2	Si no hi ha lloc torna a començar		Si	Si	Ok
3	Genera taulells valids		Si	Si	Ok

Totes les proves donen correcte.

Segons les proves realitzades, el programa funciona correctament i d'acord amb les especificacions inicials.