Modulo PlayGame

Descrizione del modulo: Contiene tutte le funzioni relative allo svolgimento di una partita

INTERFACCIA

playGame

INPUT

Nome	Descrizione	Tipo	Vincoli
round	Round della partita	Round	/
numFile	Numero di file di salvataggio della partita	Intero	numFile >= 1 AND numFile <= 7
c	Carattere letto da tastiera	Carattere	/

OUTPUT

Nome	Descrizione	Tipo	Vincoli
file	File di salvataggio della partita	File binario	/

LAVORO

Nome	Descrizione	Tipo	Vincoli
turn	Turno di gioco	Intero	turn > 0
whoPlay	Indica il player che attacca	Intero	1 = Attacca il player 1 2 = Attacca il player2

pause	Indica lo stato di pausa della partita	Intero	1 = Uscire dalla partita 0 = Continuare a giocare
player1	Player che attacca	Player	/
player2	Player che subisce l'attacco	Player	/
end	Indica se la partita è terminata o meno	Intero	1 = partita terminata 2 = partita in corso
availableShipsControl	Numero di navi rimaste al giocatore che subisce l'attacco	Intero	availableShipsControl >= 0 AND availableShipsControl <= MAX_SHIP_AMOUNT
idWinner	Id del player vincitore	Intero	1 = Player 1 2 = Player 2
swap	Player temporaneo necessario per il salvataggio	Player	/

ALGORITMO

```
cleanScreen()
player1 := getActivePlayer(round)
player2 := getPassivePlayer(round)
turn := getGameRound(round)
whoPlay := getWhoPlay(round)
pause:= getPause(round)
end := 0
MENTRE((pause=0) AND (end=0))
  SE (whoPlay = 1)
    ALLORA
      StampaAVideo("ORA GIOCA IL GIOCATORE 1")
      round := setActivePlayer(round, player1)
      round := setPassivePlayer(round, player2)
      round := newTurn(round)
      pause := getPause(round)
      SE (pause = 0)
        ALLORA
          player1 := getActivePlayer(round)
          player2 := getPassivePlayer(round)
          whoPlay := 2
          round := setWhoPlay(round, whoPlay)
      FINE
```

```
  ALTRIMENTI
      StampareAVideo("ORA GIOCA IL GIOCATORE 2")
      round := setActivePlayer(round, player2)
      round := setPassivePlayer(round, player1)
      round := newTurn(round)
      pause := getPause(round)
      SE (pause = 0)
        ALLORA
          player2 := getActivePlayer(round)
          player1 := getPassivePlayer(round)
          whoPlay := 1
          round := setWhoPlay(round, whoPlay)
      FINE
    SE (pause = 0)
      ALLORA
        availableShipsControl := getAvailableShips(getPassivePlayer(round))
        SE (availableShipsControl = 0)
          ALLORA
            idWinner := getId(getActivePlayer(round))
            StampaAVideo("Partita finita: ha vinto il giocatore " idWinner)
            end := 1
            c := clickToContinue()
          ALTRIMENTI
            turn := turn + 1
            round := setGameRound(round, turn)
        FINE
        SE (whoPlay = 1)
          ALLORA
              swap := getActivePlayer(round)
              round := setActivePlayer(round, getPassivePlayer(round))
              round := setPassivePlayer(round, swap)
        FINE
        saveGame(round, numFile)
  FINE
  cleanScreen()
FINE
```

---

newTurn

INPUT

Nome	Descrizione	Tipo	Vincoli

round	Round della partita da giocare	Round	/
AIRSTRIKE_TURN	Turno dal quale si può iniziare ad utilizzare l'Airstrike	Numero intero	10
c	Carattere letto da tastiera	Carattere	/

OUTPUT

Nome	Descrizione	Tipo	Vincoli
round	Round della partita giocato	Round	/

LAVORO

Nome	Descrizione	Tipo	Vincoli
activePlayer	Player che attacca	Player	/
activePlayerPlayground	Playground del player che attacca	Array bidimensionale di caratteri	Dimensione 16x16 Caratteri consentiti: Da 'a' a 'o' per le navi '~' per il mare 'x' nave colpita '*' nave affondata
activePlayerHeatMap	HeatMap del player che attacca	Array bidimensionale di caratteri	Dimensione 16x16 '~' per il mare '?' per ignoto '!' per nave colpita '*' nave colpita e affondata '#' nave scansionata
turn	Turno di gioco	Intero	turn > 0
pause	Indica lo stato di pausa del gioco	Intero	1 = Uscire dalla partita 0 = Continuare a giocare
choice	Scelta della prossima mossa	Carattere	Choice >='1' AND choice <= '5'
row	Riga scelta	Intero	Row >= TABLE_MIN AND row <= TABLE_MAX
column	Colonna scelta	Intero	Column >= TABLE_MIN AND column <= TABLE_MAX

axis	Asse scelto	Carattere	R = Riga C = Colonna
activePlayerLongShot	Numero di longShot di activePlayer ancora disponibili	Intero	activePlayerLongShot >= 0 AND <= MAX_LONG_SHOT
activePlayerAirStrike	Numero di airStrike di activePlayer ancora disponibili	Intero	activePlayerAirStrike >= 0 AND activePlayerAirStrike <= MAX_AIR_STRIKE
activePlayerRadar	Numero di radar di activePlayer ancora disponibili	Intero	activePlayerRadar >= 0 AND activePlayerRadar <= MAX_RADAR
error	Indica una scelta non valida	Intero	1 = scelta non valida 0 = scelta valida

ALGORITMO

```
activePlayer := getActivePlayer(round)
activePlayerPlayground := getPlayground(activePlayer)
activePlayerHeatMap := getHeatMap(activePlayer)
StampaAVideo("ECCO LA TUA ATTUALE MAPPA DI GIOCO")
showMap(activePlayerPlayground)
StampaAVideo("ECCO LA TUA ATTUALE CONOSCENZA DEL CAMPO AVVERSARIO")
showMap(activePlayerHeatMap)
activePlayerLongShot := getLongShot(activePlayer)
activePlayerAirStrike := getAirStrike(activePlayer)
activePlayerRadar := getRadar(activePlayer)
turn := getGameRound(round)
ESEGUI
  error := 0
  choice := nextMoveChoice()
  SE (choice = 1)
  ALLORA
    column := columnChoice()
    row := rowChoice()
    round := hit(row, column, round)
    c := clickToContinue()
  ALTRIMENTI
    SE (choice = 2)
      ALLORA
        SE (activePlayerLongShot > 0)
          ALLORA
```

```
        column := columnChoice()
        row := rowChoice()
        round := longShot(row, column, round)
        activePlayer:=getActivePlayer(round)
        activePlayer := setLongShots(activePlayer, (activePlayerLongShot - 1))
        round := setActivePlayer(round, activePlayer)
        c := clickToContinue()
      ALTRIMENTI
        Error := 1
        StampaAVideo("ERRORE: NON HAI COLPI A LARGO RAGGIO A DISPOSIZIONE")
    FINE
  ALTRIMENTI
    SE (choice = 3)
      ALLORA
        SE (turn > AIRSTRIKE_TURN)
          ALLORA
            SE (activePlayerAirStrike > 0)
              ALLORA
                axis = axisChoice()
                SE (axis = 'R')
                  ALLORA
                    row := rowChoice()
                    round := airStrikeRow(round, row)
                    activePlayer:=getActivePlayer(round)
                    activePlayer := setAirStrike(activePlayer, (activePlayerAirStrike – 1))
                    round := setActivePlayer(round, activePlayer)
                    c := clickToContinue()
                  ALTRIMENTI
                    column := columnChoice()
                    round := airStrikeColumn(round, column)
                    activePlayer := getActivePlayer(round)
                    activePlayer := setAirStrike(activePlayer, (activePlayerAirStrike – 1))
                    round := setActivePlayer(round, activePlayer)
                    c := clickToContinue()
                FINE
            ALTRIMENTI
                error := 1
                StampaAVideo("Errore: non hai il bombardamento aereo a disposizione")
            FINE
          ALTRIMENTI
            error := 1
            StampaAVideo("Errore: bombardamento aereo non attivo")
```

```
        FINE
      ALTRIMENTI
        SE (choice = 4)
          ALLORA
            SE (activePlayerRadar > 0)
                    ALLORA
                      column := columnChoice()
                      row := rowChoice()
                      round := radar(round, row, column)
                      activePlayer:=getActivePlayer(round)
                      activePlayer := setRadar(activePlayer, (activePlayerRadar - 1))
                      round := setActivePlayer(round, activePlayer)
                      c := clickToContinue()
                    ALTRIMENTI
                      error := 1
                      StampaAVideo("Errore: non hai il radar a disposizione")
            FINE
            ALTRIMENTI
                  SE (choice = 5)
                    ALLORA
                      pause := 1
                      round := setPause(round,pause)
                  FINE
        FINE
FINCHE'(error = 1)
FINE
```

---

nextMoveChoice

INPUT

Nome	Descrizione	Tipo	Vincoli
choice	Scelta della prossima mossa	Carattere	Choice >= '1' AND choice <= '5'

OUTPUT

Nome	Descrizione	Tipo	Vincoli
choice	Scelta della prossima mossa	Carattere	Choice >= '1' AND choice <= '5'

LAVORO

Nome	Descrizione	Tipo	Vincoli

error	Indica se la scelta effettuata è valida	Intero	1 = Scelta non valida 0 = scelta valida

ALGORITMO

```
ESEGUI
  error := 0
  StampaAVideo(“Mosse:”)
  StampaAVideo(“1: Colpo normale”)
  StampaAVideo(“2: Colpo a largo raggio”)
  StampaAVideo(“3: Bombardamento aereo”)
  StampaAVideo(“4: Radar”)
  StampaAVideo(“5: Esci.”)
  StampaAVideo(“Seleziona la tua mossa: ”)
  choice := LeggereDaTastiera()
  SE ( (choice < ’1’) OR (choice > ’5’))
    error := 1
    StampaAVideo(“Errore: inserire una scelta valida”)
  FINE
FINCHE(error = 1)
FINE
```

---

rowChoice

INPUT

Nome	Descrizione	Tipo	Vincoli
row	Scelta della riga da colpire	Intero	Row >= TABLE_MIN AND row <= TABLE_MAX
TABLE_MIN	Riga/Colonna minima	Numero intero	1
TABLE_MAX	Riga/Colonna massima	Numero intero	16

OUTPUT

Nome	Descrizione	Tipo	Vincoli
row	Scelta della riga da colpire	Intero	Row >= TABLE_MIN AND row <= TABLE_MAX

LAVORO

Nome	Descrizione	Tipo	Vincoli
error	Indica una scelta non valida	Intero	1 = scelta non valida 0 = scelta valida

ALGORITMO

```
ESEGUI
  error := 0
  StampaAVideo(“Inserire la riga da colpire: “)
  row := LeggereDaTastiera()
  SE (row < TABLE_MIN OR row > TABLE_MAX)
    ALLORA
      error := 1
      StampaAVideo(“Errore: inserire un valore di riga valido”)
  FINE
FINCHE’ (error = 1)
FINE
```

---

columnChoice

INPUT

Nome	Descrizione	Tipo	Vincoli
column	Scelta della colonna da colpire, da convertire in un intero	Carattere	column >= ‘A’ AND column <= ‘P’

OUTPUT

Nome	Descrizione	Tipo	Vincoli
intColumn	Scelta della colonna da colpire, convertita in un intero	Intero	column >= TABLE_MIN AND column <= TABLE_MAX

LAVORO

Nome	Descrizione	Tipo	Vincoli
error	Indica una scelta non valida	Intero	1 = scelta non valida 0 = scelta valida

ALGORITMO

```
ESEGUI
  error := 0
  StampaAVideo(“Inserire la colonna da colpire: “)
  column := LeggereDaTastiera()
```

```
  column := toUpperCase(column)
  SE (column < 'A' OR column > 'P')
    ALLORA
      error := 1
      StampaAVideo("Errore: inserire un valore di colonna valido")
  FINE
FINCHE' (error = 1)
FINE
intColumn := getIntegerColumn(column)
```

---

hit

INPUT:

Nome	**Descrizione**	**Tipo**	**Vincoli**
row	Riga del colpo	intero	row >= TABLE_MIN AND row <= TABLE_MAX
column	Colonna del colpo	intero	column >= TABLE_MIN AND column <= TABLE_MAX
match	Contiene le informazioni relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	\
WATER	Carattere che indica la presenza di acqua nel playground e nell'heatMap	Carattere	'~'
PLAYGROUND_HIT	Carattere che indica una nave colpita nel playground	Carattere	'X'
START_LOWERCASE_ASCII	Prima lettera dell'alfabeto minuscola in ASCII	Numero intero	97

SUNK	Carattere che indica la presenza una nave affondata nel playground e nell'heatMap	Carattere	'*'
HEAT_MAP_HIT	Carattere che indica una nave colpita nell'heatMap	Carattere	'!'
START_UPPERCASE_ASCII	Prima lettera dell'alfabeto maiuscola in ASCII	Numero intero	65

OUTPUT:

Nome	**Descrizione**	**Tipo**	**Vincoli**
match	Contiene le informazioni aggiornate relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	\

LAVORO:

Nome	**Descrizione**	**Tipo**	**Vincoli**
passivePlayground	Campo da gioco del giocatore player che subisce l'attacco	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: da 'a' a 'o' per le navi '~' per il mare 'X' per nave colpita '*' per nave affondata
activeHeatMap	heatMap del giocatore player che attacca	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: '~' per il mare '!' per nave colpita '*' per nave affondata '#' per nave scansionata
cell	Contenuto della cella del passivePlayground	Carattere	Caratteri consentiti: da 'a' a 'o' per le navi

			'~' per il mare 'X' per nave colpita '*' per nave affondata
passivePlayer	Giocatore che subisce l'attacco	Player	\
activePlayer	Giocatore che attacca	Player	\
Ship	Nave del giocatore che subisce l'attacco	Ship	\
lifePoints	lifePoints della nave che ha subito l'attacco	Intero	lifePoints >= 0 AND lifePoints <= MAX_SHIP_SIZE
passivePlayerShips	Numero di navi rimaste al giocatore che ha subito l'attacco	Intero	passivePlayerShips >= 0 AND passivePlayerShips <= MAX_SHIP_AMOUNT
direction	Direzione della nave	Carattere	V = Verticale O = Orizzontale

ALGORITMO

passivePlayer := getPassivePlayer(match)
activePlayer := getActivePlayer(match)
passivePlayground := getPlayground(passivePlayer)
ActiveHeapMap := getHeatMap(activePlayer)
cell := elemento in riga row e colonna column di passivePlayground
SE ((elemento in riga row e colonna column di passivePlayground = WATER) OR (elemento in riga row e colonna column di passivePlayground = PLAYGROUND_HIT) OR (elemento in riga row e colonna column di passivePlayground = SUNK)
  ALLORA
    SE (elemento in riga row e colonna column di activeHeatMap = UNKNOWN)
      ALLORA
        elemento in riga row e colonna column di activeHeatMap := WATER
    FINE
    StampaAVideo((codice ASCII di (column + START_UPPERCASE_ASCII) - 1), row ": ACQUA!")
  ALTRIMENTI
    ship := getShip(passivePlayer, (codice ASCII di (cell - START_LOWERCASE_ASCII) + 1)
    lifePoints := getLifePoints(ship)
    SE (lifePoints = 1)
      ALLORA
        StampaAVideo((codice ASCII di (column + START_UPPERCASE_ASCII) -1), row ": COLPITO E AFFONDATO!")

```
        passivePlayerShips := getAvailableShips(passivePlayer)
        passivePlayerShips := passivePlayerShips – 1
        passivePlayer := setAvailableShips(passivePlayer, passivePlayerShips)
        passivePlayerCoords := getCoords(ship)
        direction := getDirection(ship)
        row := pullRow(ship)
        column := pullColumn(ship)
        SE (direction = 'V')
          ALLORA
            MENTRE (elemento in riga row e colonna column di passivePlayground <> WATER AND
            row <= TABLE_MAX)
              elemento in riga row e colonna column di passivePlayground := SUNK
              elemento in riga row e colonna column di activeHeatMap := SUNK
                row := row + 1
            FINE
          ALTRIMENTI
            MENTRE (elemento in riga row e colonna column di passivePlayground <> WATER AND
            column <= TABLE_MAX)
              elemento in riga row e colonna column di passivePlayground := SUNK
                elemento in riga row e colonna column di activeHeatMap := SUNK
                column := column + 1
            FINE
          FINE
      ALTRIMENTI
        elemento in riga row e colonna column di passivePlayground := PLAYGROUND_HIT
        elemento in riga row e colonna column di activeHeatMap := HEAT_MAP_HIT
        StampaAVideo( (codice ASCII di (column + START_UPPERCASE_ASCII) - 1), row ":
COLPITO!")
    FINE
  ship := setLifePoints(ship, (lifePoints – 1))
  passivePlayer := setShip(passivePlayer, (codice ASCII di (cell – START_LOWERCASE_ASCII) + 1),
ship)
FINE
passivePlayer := setPlayground(passivePlayer, passivePlayground)
activePlayer := setHeatMap(activePlayer, activeHeatMap)
match := setActivePlayer(match, activePlayer)
match := setPassivePlayer(match, passivePlayer)
```

---

longShot

INPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	/
row	riga del colpo	numero intero	Row >= TABLE_MIN AND row <= TABLE_MAX
column	colonna del colpo	numero intero	Column >= TABLE_MIN AND column <= TABLE_MAX
TABLE_MIN	Riga/Colonna minima	Numero intero	1
TABLE_MAX	Riga/Colonna massima	Numero intero	16

OUTPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni aggiornate relative al colpo effettuato (player che ha attaccato, player che ha subito l'attacco, numero del turno)	Round	/

LAVORO

Nome	Descrizione	Tipo	Vincoli
passivePlayground	Campo da gioco del giocatore player che subisce l'attacco	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: da 'a' a 'o' per le navi '~' per il mare 'x' per nave colpita '*' per nave affondata
activeHeatMap	heatMap del giocatore player che attacca	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: '~' per il mare '!' per nave colpita

			'*' per nave affondata '#' per nave scansionata
pivotRow	Riga di inizio del controllo	Numero intero	pivotRow >= TABLE_MIN AND pivotRow <= TABLE_MAX
pivotColumn	Colonna di inizio del controllo	Numero intero	pivotColumn >= TABLE_MIN AND pivotColumn <= TABLE_MAX
endRow	Riga di fine del controllo	Numero intero	endRow >= TABLE_MIN AND endRow <= TABLE_MAX
endColumn	Colonna di fine del controllo	Numero intero	endColumn >= TABLE_MIN AND endColumn <= TABLE_MAX
i	Contatore della colonna in passivePlayground	Numero intero	i > 0
passivePlayer	Giocatore che subisce l'attacco	Player	\
activePlayer	Giocatore che attacca	Player	\

ALGORITMO

```
passivePlayer := getPassivePlayer(match)
activePlayer := getActivePlayer(match)
passivePlayground:= getPlayground(passivePlayer)
activeHeatMap := getHeatMap(activePlayer)
pivotColumn := column - 1
pivotRow := row - 1
SE (pivotColumn < TABLE_MIN)
  ALLORA
    pivotColumn := TABLE_MIN
FINE
SE (pivotRow < TABLE_MIN)
  ALLORA
    pivotRow := TABLE_MIN
FINE
endRow := row + 1
endColumn := column + 1
```

```
SE (endRow > TABLE_MAX)
  ALLORA
    endRow := TABLE_MAX
FINE
SE (endColumn > TABLE_MAX)
  ALLORA
    endColumn := TABLE_MAX
FINE
MENTRE (pivotRow <= endRow AND getAvailableShips(getPassivePlayer(match)) > 0)
  i := pivotColumn
  MENTRE (i <= endColumn AND getAvailableShips(getPassivePlayer(match)) > 0)
    match := hit(pivotRow, i, match)
    i := i + 1
  FINE
  pivotRow := pivotRow + 1
FINE
```

---

axisChoice

INPUT

Nome	Descrizione	Tipo	Vincoli
Axis	Scelta dell'asse da colpire	Carattere	C = colonna R = riga

OUTPUT

Nome	Descrizione	Tipo	Vincoli
Axis	Scelta dell'asse da colpire	Carattere	C = colonna R = riga

LAVORO

Nome	Descrizione	Tipo	Vincoli
Error	indica una scelta non valida	Intero	1 = scelta non valida 0 = scelta valida

ALGORITMO

```
ESEGUI
  error := 0
  StampaAVideo("Inserire R se si vuole colpire la riga, oppure C se si vuole colpire la colonna: ")
  axis := LeggereDaTastiera()
  axis := toUpperCase(axis)
```

```
  SE ((axis <> 'C') AND (axis <> 'R'))
    ALLORA
      error := 1
      StampaAVideo("Errore: effettuare una scelta corretta")
  FINE
FINCHE' (error = 1)
FINE
```

---

airStrikeRow

INPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	/
row	Riga del colpo	numero intero	row >= TABLE_MIN AND row <= TABLE_MAX
TABLE_MAX	Riga/Colonna massima	Numero intero	16

OUTPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni aggiornate relative al colpo effettuato (player che ha attaccato, player che ha subito l'attacco, numero del turno)	Round	/

LAVORO

Nome	Descrizione	Tipo	Vincoli
passivePlayground	Campo da gioco del giocatore player che subisce l'attacco	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: da 'a' a 'o' per le navi '~' per il mare 'x' per nave colpita

			'*' per nave affondata
activeHeatMap	heatMap del giocatore player che attacca	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: '~' per il mare '!' per nave colpita '*' per nave affondata '#' per nave scansionata
i	Contatore della colonna in passivePlayground	Numero intero	i > 0
passivePlayer	Giocatore che subisce l'attacco	Player	\
activePlayer	Giocatore che attacca	Player	\

ALGORITMO

```
passivePlayer := getPassivePlayer(match)
activePlayer := getActivePlayer(match)
passivePlayground:= getPlayground(passivePlayer)
activeHeatMap:= getHeatMap(activePlayer)
i := 1
MENTRE (i <= TABLE_MAX AND getAvailableShips(getPassivePlayer(match)) > 0)
  match := hit(row, i, match)
  i := i + 1
FINE
```

---

airStrikeColumn

INPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	/
column	colonna del colpo	numero intero	Column >= TABLE_MIN AND column <= TABLE_MAX

TABLE_MAX	Riga/Colonna massima	Numero intero	16

OUTPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni aggiornate relative al colpo effettuato (player che ha attaccato, player che ha subito l'attacco, numero del turno)	Round	/

LAVORO

Nome	Tipo	Descrizione	Vincoli
passivePlayground	Array di caratteri a due dimensioni	Campo da gioco del giocatore player che subisce l'attacco	Dimensione 16x16 Caratteri consentiti: da 'a' a 'o' per le navi '~' per il mare 'x' per nave colpita '*' per nave affondata
activeHeatMap	Array di caratteri a due dimensioni	heatMap del giocatore player che attacca	Dimensione 16x16 Caratteri consentiti: '~' per il mare '!' per nave colpita '*' per nave affondata '#' per nave scansionata
i	Numero intero	Contatore della colonna in passivePlayground	i > 0
passivePlayer	Player	Giocatore che subisce l'attacco	\
activePlayer	Player	Giocatore che attacca	\

ALGORITMO

passivePlayer := getPassivePlayer(match)
activePlayer := getActivePlayer(match)
passivePlayground := getPlayground(passivePlayer)
activeHeatMap := getHeatMap(activePlayer)
i := 1

```
MENTRE (i <= TABLE_MAX AND getAvailableShips(getPassivePlayer(match)) > 0)
  match := hit(i, column, match)
  i := i + 1
FINE
```

---

scan

INPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	/
row	riga del colpo	numero intero	Row >= TABLE_MIN AND row <= TABLE_MAX
column	colonna del colpo	numero intero	Column >= TABLE_MIN AND column <= TABLE_MAX
WATER	Carattere che indica la presenza di acqua nel playground e nell'heatMap	Carattere	'~'
PLAYGROUND_HIT	Carattere che indica una nave colpita nel playground	Carattere	'X'
HEAT_MAP_SUCCESSFUL_SCAN	Carattere che indica la presenza di una nave nell'heatMap a seguito di una scansione	Carattere	'#'

START_UPPERCASE_ASCII	Prima lettera dell'alfabeto maiuscola in ASCII	Numero intero	65

OUTPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni aggiornate relative al colpo effettuato (player che ha attaccato, player che ha subito l'attacco, numero del turno)	Round	/

LAVORO

Nome	Tipo	Descrizione	Vincoli
passivePlayground	Array di caratteri a due dimensioni	Campo da gioco del giocatore player che subisce l'attacco	Dimensione 16x16 Caratteri consentiti: da 'a' a 'o' per le navi '~' per il mare 'x' per nave colpita '*' per nave affondata
activeHeatMap	Array di caratteri a due dimensioni	heatMap del giocatore player che attacca	Dimensione 16x16 Caratteri consentiti: '~' per il mare '!' per nave colpita '*' per nave affondata '#' per nave scansionata
passivePlayer	Player	Giocatore che subisce l'attacco	\
activePlayer	Player	Giocatore che attacca	\

ALGORITMO

passivePlayer := getPassivePlayer(match)

activePlayer := getActivePlayer(match)

passivePlayground:= getPlayground(passivePlayer)

activeHeatMap:= getHeatMap(activePlayer)

SE ((elemento in riga row e colonna column di passivePlayground = WATER) OR (elemento in riga row e colonna column di passivePlayground = PLAYGROUND_HIT) OR (elemento in riga row e colonna column di passivePlayground = SUNK)

```
    ALLORA
      SE (elemento in riga row e colonna column di activeHeatmap = UNKNOWN)
        ALLORA
          elemento in riga row e colonna column di activeHeatMap := WATER
      FINE
      StampareAVideo((codice ASCII di (column +START_UPPERCASE_ASCII) -1), row ": VUOTO")
    ALTRIMENTI
      elemento in riga row e colonna column di activeHeatMap := HEAT_MAP_SUCCESSFUL_SCAN
      StampareAVideo((codice ASCII di (column +START_UPPERCASE_ASCII) -1), row ": NAVE")
  FINE
  passivePlayer := setPlayground(passivePlayer, passivePlayground)
  activePlayer := setHeatMap(activePlayer, activeHeatMap)
  match := setActivePlayer(match, activePlayer)
  match := setPassivePlayer(match, passivePlayer)
```

---

radar

INPUT

Nome	**Descrizione**	**Tipo**	**Vincoli**
match	Contiene le informazioni relative al turno corrente (player che attacca, player che subisce l'attacco, numero del turno)	Round	/
row	riga del colpo	numero intero	Row >= TABLE_MIN AND row <= TABLE_MAX
column	colonna del colpo	numero intero	Column >= TABLE_MIN AND column <= TABLE_MAX
TABLE_MIN	Riga/Colonna minima	Numero intero	1
TABLE_MAX	Riga/Colonna massima	Numero intero	16

OUTPUT

Nome	Descrizione	Tipo	Vincoli
match	Contiene le informazioni	Round	/

	aggiornate relative al colpo effettuato (player che ha attaccato, player che ha subito l'attacco, numero del turno)		

LAVORO

Nome	**Descrizione**	**Tipo**	**Vincoli**
passivePlayground	Campo da gioco del giocatore player che subisce l'attacco	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: da 'a' a 'o' per le navi '~' per il mare 'x' per nave colpita '#' per nave affondata
activeHeatMap	heatMap del giocatore player che attacca	Array di caratteri a due dimensioni	Dimensione 16x16 Caratteri consentiti: '~' per il mare '!' per nave colpita '*' per nave affondata '#' per nave scansionata
i	Contatore della colonna in passivePlayground	Numero intero	i > 0
passivePlayer	Giocatore che subisce l'attacco	Player	\
activePlayer	Giocatore che attacca	Player	\
pivotRow	Riga di inizio del controllo	Numero intero	pivotRow >= TABLE_MIN AND pivotRow <= TABLE_MAX
pivotColumn	Colonna di inizio del controllo	Numero intero	pivotColumn >= TABLE_MIN AND pivotColumn <= TABLE_MAX
endRow	Riga di fine del controllo	Numero intero	endRow >= TABLE_MIN AND endRow <= TABLE_MAX
endColumn	Colonna di fine del controllo	Numero intero	endColumn >= TABLE_MIN AND

			endColumn <= TABLE_MAX

ALGORITMO

```
passivePlayer := getPassivePlayer(match)
activePlayer := getActivePlayer(match)
passivePlayground:= getPlayground(passivePlayer)
activeHeatMap:= getHeatMap(activePlayer)
pivotColumn := column - 1
pivotRow := row - 1
SE (pivotColumn < TABLE_MIN)
  ALLORA
    pivotColumn := TABLE_MIN
FINE
SE (pivotRow < TABLE_MIN)
  ALLORA
    pivotRow := TABLE_MIN
FINE
endRow := row + 1
endColumn := column + 1
SE (endColumn > TABLE_MAX)
  ALLORA
    endColumn := TABLE_MAX
FINE
SE (endRow > TABLE_MAX)
  ALLORA
    endRow := TABLE_MAX
FINE
MENTRE (pivotRow <= endRow)
  i := pivotColumn
  MENTRE (i <= endColumn)
    match := scan(pivotRow, i, match)
    i := i + 1
  FINE
  pivotRow := pivotRow + 1
FINE
```

---

clickToContinue

INPUT

Nome	Descrizione	Tipo	Vincoli

c	Carattere letto da tastiera	Carattere	/

OUTPUT

/

ALGORITMO

StampaAVideo(“PREMI INVIO PER CONTINUARE:”)

c := LeggereDaTastiera()