# MAIN

**Dati input**

**Dati output**

**Dati locali**

scelta (intero, può assumere valori da 0 a 5)

Fsave (di tipo FILE)

localp (di tipo “partita” contiene tutte le informazioni riguardo una nuova partita non ancora salvata oppure di una partita caricata)

c\_imp (di tipo “config”, i campi vengono inizializzati nel seguente modo:

pg1: 2,

pg2: 1,

ng1: Giocatore 1,

ng2: Giocatore 2)

r (di tipo “rec”, i campi vengono inizializzati nel seguente modo:

ora: 0

minuti: 0

secondi: 0

diff\_t: 0

nomerecord: vuoto)

**ALGORITMO**

**INIZIO**

ESEGUI

Chiamare “stampare\_titolo()”

Chiamare “stampare\_interfaccia\_menu()”

Stampare a video (“Inserire scelta: ”)

Acquisire scelta da tastiera

SE (**scelta** minore di 0 o maggiore di 5)

ALLORA

Stampare messaggio di errore “ERRORE! Inserire un numero compreso tra 0 e 5”

ALTRIMENTI

SE (**scelta** uguale a 1)

ALLORA

Assegnare a localp il valore restituito dalla funzione “nuova\_partita(r, c\_imp)”

ALTRIMENTI

SE (**scelta** uguale a 2)

ALLORA

Assegnare a localp il valore restituito dalla funzione “carica\_partita(Fsave)”

ALTRIMENTI

SE (**scelta** uguale a 3)

ALLORA

Chiamare “salva\_partita(Fsave, localp)”

ALTRIMENTI

SE (**scelta** uguale a 4)

ALLORA

Chiamare “impostazioni(c\_imp)”

ALTRIMENTI

SE (**scelta** uguale a 5)

ALLORA

Chiamare “topten()”

FINE

FINE

FINE

FINE

FINE

FINCHE’ (**scelta** diverso da 0)

FINE

**FINE**

# NUOVA PARTITA

**Dati input**

recptr (di tipo “rec” , contiene tutte le informazioni riguardo il tempo Record realizzato durante partita)

imp (di tipo “config”, contiene tutte le informazioni riguardo alle impostazioni di gioco)

**Dati output**

recptr (di tipo “rec” )

game (di tipo “partita”, contiene tutte le informazioni relative alla partita giocata; i seguenti campi dell’archivio vengono settati nel seguente modo:

save:FALSO

recordp

ora: 0

minuti: 0

secondi: 0

diff\_t: 0

nomerecord:vuoto

nomepartita:vuoto

turn:0)

**Dati locali**

**ALGORITMO**

**INIZIO**

Assegnare al campo **save** dell’archivio **game:** FALSO

Assegnare al campo **ora** dell’archivio “**recordp”** contenuto nell’archivio “**game”:** 0

Assegnare al campo **minuti** dell’archivio “**recordp”** contenuto nell’archivio “**game”:** 0

Assegnare al campo **secondi** dell’archivio “**recordp”** contenuto nell’archivio “**game”:** 0

Assegnare al campo **diff\_t** dell’archivio “**recordp”** contenuto nell’archivio “**game”:** 0

Inizializzare il campo **nomerecord** dell’archivio “**recordp”** contenuto nell’archivio “**game”:** vuoto

Inizializzare il campo **nomepartita** dell’archivio “**game”:** vuoto

Assegnare al campo **turn** dell’archivio “**game”:** 0

Chiamare “creare\_matrice\_pedine(game, imp)”

Chiamare “creare\_matrice\_grafica()”

Chiamare “stampare\_interfaccia\_campo(game)”

Assegnare al campo **nomeg1** dell’archivio “**game”** il campo **ng1** dell’archivio “**imp”**

Assegnare al campo **nomeg2** dell’archivio “**game”** il campo **ng2** dell’archivio “**imp”**

Chiamare “effettuare\_mossa(game)”

**FINE**

# CARICA PARTITA

**Dati input**

fpt (di tipo FILE, contenente tutte le partite salvate)

codESC (intero uguale a 27, indica il codice ASCII del tasto ESC sulla tastiera)(dato globale)

**Dati output**

part (di tipo “partita”, permette all’utente di riprendere una partita precedentemente salvata)

**Dati locali**

scelta (intero, può assumere valori compresi tra 0 e 4 e ”codESC”)

esito (di tipo logico, viene inizializzato a FALSO, indica se posso caricare o meno la partita selezionata)

**ALGORITMO**

**INIZIO**

ESEGUI

Chiamare “stampare\_interfaccia\_carica\_partita()”

Chiamare “leggere\_file\_salvataggio(fpt)”

SE (il file esiste)

ALLORA

Eseguire le operazioni per caricare la partita

Assegnare a **scelta** il valore restituito dalla funzione “scegliere\_slot()”

SE (scelta è diverso da codESC)

ALLORA

Assegnare ad **esito** il valore restituito dalla funzione “controllare\_slot\_caricamento(fpt, scelta)”

SE (**esito** è uguale a VERO)

ALLORA

Assegnare a **part** il valore restituito dalla funzione “estrarre\_partita(ftp, scelta)”

Chiamare “riprendere\_partita(part)”

FINE

ALTRIMENTI

Assegnare a **esito** il valore VERO

FINE

ALTRIMENTI

Assegnare a **esito** il valore VERO

FINE

FINCHE’ (**esito** è diverso da VERO)

FINE

**FINE**

# SALVA PARTITA

**Dati input**

fpt (di tipo FILE, contenente tutte le partite salvate)

p (di tipo “partita”, contiene la partita in corso)

codESC (intero uguale a 27, indica il codice ASCII del tasto ESC sulla tastiera)(dato globale)

**Dati output**

fpt (di tipo FILE)

**Dati locali**

esito (di tipo logico,)

scelta (intero, può assumere valori compresi tra 0 e 4 e “codESC”)

**ALGORITMO**

**INIZIO**

ESEGUI

Chiamare “stampare\_interfaccia\_salva\_partita()”

SE (il file non è presente)

ALLORA

Chiamare “creare\_file\_salvataggio(fpt)”

FINE

Chiamare “leggere\_file\_salvataggio(fpt)”

Assegnare a **scelta** il valore restituito dalla funzione “scegliere\_slot()”

SE (**scelta** è diverso da codESC)

ALLORA

Assegnare ad **esito** il valore restituito dalla funzione “controllare\_slot\_salvataggio(fpt, scelta)”

SE (**esito** è uguale a VERO)

ALLORA

Chiamare “effettuare\_salvataggio\_partita(fpt, scelta, p)”

FINE

ALTRIMENTI

Assegnare ad **esito** il valore VERO

FINE

FINCHE’ (**esito** è diverso da VERO)

FINE

**FINE**

# IMPOSTAZIONI

**Dati input**

imp (di tipo “config”, contiene tutte le informazioni riguardo alle impostazioni)

**Dati output**

imp (di tipo “config”, contiene tutte le informazioni riguardo alle impostazioni)

**Dati locali**

cgx (intero, inizializzato a 5, indica la riga grafica)

scelta (intero, può assumere valori compresi tra 0 e 2)

**ALGORITMO**

**INIZIO**

ESEGUI

Chiamare “stampare\_interfaccia\_impostazioni()”

Inserire **scelta**

SE (**scelta** è minore di 0 o maggiore di 2)

ALLORA

Stampare messaggio di errore (“ERRORE! Inserire un numero compreso tra 0 e 2”)

ALTRIMENTI SE (**scelta** uguale a 1)

ALLORA

Chiamare “cambiare\_simbolo\_pedine(imp)”

ALTRIMENTI SE (**scelta** uguale a 2)

ALLORA

Chiamare “cambiare\_nome\_giocatori (imp)”

FINE

FINE

FINE

FINCHE’ (**scelta** diversa da 0)

FINE

**FINE**

# TOPTEN

**Dati input**

**Dati output**

**Dati locali**

fpt (di tipo FILE, contenente i tempi di Record conseguiti)

**ALGORITMO**

**INIZIO**

Chiamare “stampare\_interfaccia\_topten()”

SE (il file non esiste)

ALLORA

Stampare messaggio di errore (“Errore! File TopTen inesistente”)

ALTRIMENTI

Chiamare “leggere\_file\_topten(fpt)”

FINE

**FINE**

# stampare\_titolo()

**Dati input**

**Dati output**

**Dati locali**

fpt (di tipo FILE, contiene il titolo del gioco)

**ALGORITMO**

**INIZIO**

SE (il file non è presente)

ALLORA

Stampare “File titolo non trovato”

ALTRIMENTI

Leggere il file carattere per carattere

FINE

Chiudere il file “titolo.txt”

**FINE**

# stampare\_interfaccia\_menu()

**Dati input**

**Dati output**

**Dati locali**

**ALGORITMO**

**INIZIO**

Stampare a video “1. Nuova Partita”

Stampare a video “2. Carica Partita”

Stampare a video “3. Salva Partita”

Stampare a video “4. Impostazioni”

Stampare a video “5. Record”

Stampare a video “0. Fine”

**FINE**

# creare\_matrice\_pedine()

**Dati input**

p (di tipo “partita”)

imp (di tipo “config”, contiene tutte le informazioni relative alle impostazioni)

righe (intero uguale a 5, indica le righe della “matrice\_pedine” linearizzata) (dato globale)

colonne (intero uguale a 9, indica le colonne della “matrice\_pedine” linearizzata) (dato globale)

**Dati output**

p (di tipo “partita”)

**Dati locali**

i (intero, contatore, indica le righe)

j (intero, contatore, indica le colonne)

**ALGORITMO**

**INIZIO**

Assegnare al campo **pedinag1** dell’archivio ”p” il valore del campo **pg1** dell’archivio “imp”

Assegnare al campo **pedinag2** dell’archivio ”p” il valore del campo **pg2** dell’archivio “imp”

Assegnare a **i** il valore 0

MENTRE (**i** è minore di **righe**)

Assegnare a **j** il valore 0

MENTRE (**j** è minore di colonne)

SE(**i** è minore o uguale a 1)

ALLORA

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag2** dell’archivio “p”

ALTRIMENTI

SE (**i** è maggiore o uguale a 3)

ALLORA

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag1** dell’archivio “p”

ALTRIMENTI

SE (**i** è uguale a 2)

ALLORA

SE (**j** è minore di 4)

ALLORA

SE (**j** diviso 2 dà come resto 0)

ALLORA

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag2** dell’archivio “p”

ALTRIMENTI

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag1** dell’archivio “p”

FINE

ALTRIMENTI

SE (**j** è maggiore di 4)

ALLORA

SE (**j** diviso 2 dà come resto 0)

ALLORA

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag1** dell’archivio “p”

ALTRIMENTI

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag2** dell’archivio “p”

FINE

ALTRIMENTI

Assegnare al campo **matrice\_pedine** nella posizione (i\*colonne+j) dell’archivio “p” il valore del campo **pedinag2** dell’archivio “p”

FINE

FINE

FINE

Incrementare **j** di 1;

FINE

Incrementare **i** di 1;

FINE

FINE

FINE

**FINE**

# creare\_matrice\_grafica()

**Dati input**

colonneg (intero uguale a 40, indica le colonne della matrice grafica linearizzata) (dato globale)

**Dati output**

**Dati locali**

fpt (di tipo FILE, contenente la struttura base del campo da gioco)

c (di tipo carattere, contiene un carattere letto dal file)

d (di tipo carattere, conserva il penultimo carattere letto dal file)

ri (intero, indica rispettivamente la riga della “matrice\_pedine”)

co (intero, indica rispettivamente la colonna della “matrice\_pedine”)

**ALGORITMO**

**INIZIO**

Aprire il file

SE (il file non è stato trovato)

ALLORA

Stampare messaggio di errore (“File campo da gioco non trovato”)

ALTRIMENTI

Leggere un carattere dal file

Assegnare a **d** il valore di c

Assegnare a **ri** il valore 0

Assegnare a **co** il valore 0

MENTRE (**c** è diverso dalla fine del file)

SE (**d** è uguale al carattere di nuova linea )

ALLORA

Assegnare a **co** il valore 0

Incrementare **ri** di 1

FINE

Assegnare alla **matrice\_grafica** in posizione (ri\*colonne+co) il valore di **c**

Incrementare **co** di 1

Assegnare a **d** il valore di **c**

Leggere un carattere dal file

FINE

Chiudere il file

FINE

**FINE**

# stampare\_interfaccia\_campo()

**Dati input**

p (di tipo “partita”)

colonne (intero uguale a 9, indica le colonne del campo da gioco) (dato globale)

**Dati output**

**Dati locali**

ri (intero, indica rispettivamente la riga della “matrice\_pedine”)

co (intero, indica rispettivamente la colonna della “matrice\_pedine”)

fpt (di tipo FILE, contenente la struttura base del campo da gioco)

c (di tipo carattere, contiene un carattere letto dal file )

**ALGORITMO**

**INIZIO**

SE (il file non esiste)

ALLORA

Stampare messaggio di errore (“Errore! File campo da gioco non trovato”)

ALTRIMENTI

Stampare a video i numeri da 1 a 9 che rappresentano le colonne del campo da gioco

Assegnare a **ri** il valore 0

Assegnare a **co** il valore 0

Leggere il campo da gioco dal file carattere per carattere

Assegnare a **c** un carattere letto dal file

MENTRE (c è diverso dalla fine del file)

Ogni quattro righe lette dal file scrivere a video la lettera relativa alla riga corrispondente, incrementare ri di 1 e assegnare a co il valore 0

SE (il carattere letto è uguale a “x”)

ALLORA

SE (il campo **matrice\_pedine** dell’archivio “**p**” in posizione ri\*colonne+co è uguale a 0)

ALLORA

Assegnare a **c** il carattere “O”

ALTRIMENTI

Assegnare a **c** il valore del campo **matrice\_pedine** dell’archivio “**p**” in posizione ri\*colonne+co

FINE

Incrementare **co** di 1

FINE

Stampare a video “**c”**

Assegnare a **c** il carattere letto da file

FINE

FINE

**FINE**

# stampare\_interfaccia\_topten()

**Dati input**

maxtt(intero uguale a 11, indica la dimensione massima del vettore di schede di tipo “rec”)(dato globale)

**Dati output**

**Dati locali**

i (intero, contatore)

**ALGORITMO**

**INIZIO**

Stampare a video “TOP TEN”

Stampare a video “hh:mm:ss <nomerecord>”

Assegnare a **i** il valore 1

MENTRE (**i** è minore di maxtt)

Stampare a video il valore di **i**

Incrementare **i** di 1

FINE

**FINE**

# effettuare\_mossa()

**Dati input**

p (di tipo “partita”)

codESC (intero uguale a 27, indica il codice ASCII del tasto ESC sulla tastiera) (dato globale)

colonne (intero uguale a 9, indica le colonne del campo da gioco) (dato globale)

**Dati output**

p (di tipo “partita”)

**Dati locali**

da (di tipo “pedina”, indica le coordinate di partenza della pedina)

verso (di tipo “pedina”, indica le coordinate di arrivo della pedina)

checkspia (di tipo “check”)

reclocale (di tipo ”rec”, i campi dell’archivio vengono inizializzati nel seguente modo:

ora: 0

minuti: 0

secondi: 0

diff\_t: 0

nomerecord: 0)

ftxt (di tipo FILE )

errore (di tipo logico)

vittoria (di tipo logico, indica se una partita è stata conclusa o meno, viene inizializzato a FALSO)

vicino (di tipo logico, riceve il valore VERO se la modalità di cattura è per avvicinamento)

lontano (di tipo logico, riceve il valore VERO se la modalità di cattura è per allontanamento)

esito (intero)

scelta (intero)

ESC (intero)

**ALGORITMO**

**INIZIO**

Assegnare al campo ora dell’archivio reclocale il valore 0

Assegnare al campo minuti dell’archivio reclocale il valore 0

Assegnare al campo secondi dell’archivio reclocale il valore 0

Assegnare al campo diff\_t dell’archivio reclocale il valore 0

Inizializzare il campo nomerecord dell’archivio reclocale a vuoto

Inizializzo il contatore tempo

ESEGUI

Chiamare “contatore\_tempo(reclocale, p)”

SE (il campo **turn** dell’archivio p diviso 2 da resto 0)

ALLORA

Stampare a video il contenuto del campo nomeg2 dell’archivio p

Stampare a video il contenuto del campo pedinag2 dell’archivio p

ALTRIMENTI

Stampare a video il contenuto del campo nomeg1 dell’archivio p

Stampare a video il contenuto del campo pedinag1 dell’archivio p

FINE

Stampare a video (“muove la pedina: “)

ESEGUI

Assegnare a **esito** il valore 0

Stampare a video (“Da: riga ( ) colonna ( )”)

Assegnare ad **ESC** il valore restituito dalla funzione “inserire\_coordinate\_spostamento\_pedina(da)”

SE (**ESC** è diverso da codESC)

ALLORA

Assegnare a **esito** il valore restituito dalla funzione “controllare\_coordinate\_spostamento(da)”

FINE

SE (**esito** è diverso da 0)

ALLORA

Stampare a video messaggio di errore ("Coordinate errate!! Premere un tasto...")

FINE

Chiamare “contatore\_tempo(reclocale, p)”

FINCHE’ (**esito** è diverso da 0)

FINE

SE (**ESC** è diverso da codESC)

ALLORA

ESEGUI

Assegnare ad **esito** il valore 0

Stampare a video ("A: riga ( ) colonna ( )")

Assegnare a **ESC** il valore restituito dalla funzione ” inserire\_coordinate\_ spostamento\_pedina(verso)”

SE (**ESC** è diverso da codESC)

ALLORA

Assegnare ad **esito** il valore restituito dalla funzione “controlla\_ coordinate\_spostamento(verso)”

FINE

SE (**esito** è diverso da 0)

ALLORA

Stampare a video messaggio di errore ("Coordinate errate! Premere un tasto...")

FINE

Chiamare “contatore\_tempo(reclocale, p)”

FINCHE’ (**esito** è diverso da 0)

FINE

SE (**ESC** è diverso da codESC)

ALLORA

SE (il campo **turn** dell’archivio “p” diviso 2 da resto 0)

ALLORA

SE (il campo di **matrice\_pedine** in posizione (del campo cordx dell’archivio “da” \*colonne+ il campo cordy dell’archivio “da”) è uguale al campo **pedinag2** dell’archivio “p”)

ALLORA

Assegnare a **errore** il valore restituito dalla funzione “controllare\_ spostamento\_pedina(p, da, verso, checkspia)”

ALTRIMENTI

Assegnare ad **errore** il valore VERO

FINE

ALTRIMENTI

SE (il campo di **matrice\_pedine** in posizione (del campo cordx dell’archivio “da” \*colonne+ il campo cordy dell’archivio “da”) è uguale al campo **pedinag1** dell’archivio “p”)

ALLORA

Assegnare a **errore** il valore restituito dalla funzione “controllare\_ spostamento\_pedina(p, da, verso, checkspia)”

ALTRIMENTI

Assegnare ad **errore** il valore VERO

FINE

FINE

SE (**errore** è uguale a FALSO)

ALLORA

SE (il campo **turn** dell’archivio “p” diviso 2 da resto 0)

ALLORA

Chiamare “controllare\_modalita\_cattura(p, checkspia, da, verso, il campo pedinag1 dell’archivio “p”, vicino, lontano)”

SE (**vicino** è uguale a VERO e **lontano** è uguale a VERO)

ALLORA

Assegnare a **scelta** il valore restituito da “scegliere\_modalita\_ cattura()”

SE (**scelta** è uguale a 1)

ALLORA

Chiamare “mangiare\_pedina\_avvicinamento(p, checkspia, da, verso, il campo pedinag1 dell’archivio “p”)”

ALTRIMENTI

SE (**scelta** è uguale a 2)

ALLORA

Chiamare “mangiare\_pedina\_allontanamento(p, checkspia, da, verso, il campo pedinag1 dell’archivio “p”)”

FINE

FINE

ALTRIMENTI

SE (**vicino** è uguale a VERO)

ALLORA

Chiamare “mangiare\_pedina\_avvicinamento(p, checkspia, da, verso, il campo pedinag1 dell’archivio “p”)”

ALTRIMENTI

SE (**lontano** è uguale a VERO)

ALLORA

Chiamare “mangiare\_pedina\_allontanamento(p, checkspia, da, verso, il campo pedinag1 dell’archivio “p”)”

FINE

FINE

FINE

Assegnare a vittoria il valore restituito dalla funzione “controllare\_pedine\_ campo (p, campo pedinag1 dell’archivio “p”)”

ALTRIMENTI

Chiamare “controllare\_modalita\_cattura(p, checkspia, da, verso, il campo pedinag1 dell’archivio “p”, vicino, lontano)”

SE (**vicino** è uguale a VERO e **lontano** è uguale a VERO)

ALLORA

Assegnare a **scelta** il valore restituito da “scegliere\_modalita\_ cattura()”

SE (**scelta** è uguale a 1)

ALLORA

Chiamare “mangiare\_pedina\_avvicinamento(p, checkspia, da, verso, il campo pedinag2 dell’archivio “p”)”

ALTRIMENTI

SE (**scelta** è uguale a 2)

ALLORA

Chiamare “mangiare\_pedina\_allontanamento(p, checkspia, da, verso, il campo pedinag2 dell’archivio “p”)”

FINE

FINE

ALTRIMENTI

SE (**vicino** è uguale a VERO)

ALLORA

Chiamare “mangiare\_pedina\_avvicinamento(p, checkspia, da, verso, il campo pedinag2 dell’archivio “p”)”

ALTRIMENTI

SE (**lontano** è uguale a VERO)

ALLORA

Chiamare “mangiare\_pedina\_allontanamento(p, checkspia, da, verso, il campo pedinag2 dell’archivio “p”)”

FINE

FINE

FINE

Assegnare a vittoria il valore restituito dalla funzione “controllare\_pedine\_ campo (p, campo pedinag1 dell’archivio “p”)”

FINE

Incrementare il campo turn di 1

ALTRIMENTI

Stampare messaggio di errore (“Impossibile effettuare mossa”)

FINE

FINE

FINE

Chiamare “contatore\_tempo(reclocale, p)”

FINCHE’ (ESC è diverso da codESC e vittoria è diverso da VERO)

FINE

Assegnare al campo recordp dell’archivio “p” il valore di reclocale

SE (vittoria è uguale a VERO)

ALLORA

Stampa a video (“Complimenti ha vinto la partita”)

SE (il campo (**turn-1)** dell’archivio “p” diviso 2 da resto 0)

ALLORA

Stampare a video il contenuto del campo nomeg2 dell’archivio “p”

ALTRIMENTI

Stampare a video il contenuto del campo nomeg1 dell’archivio “p”

FINE

FINE

Aprire il file

SE (il file non esiste)

ALLORA

Creare il file

Chiudere il file

FINE

Assegnare a esito il valore restituito dalla funzione “controllare\_nuovo\_record(p)”

SE (esito è uguale a VERO)

ALLORA

Stampare messaggio (“Nuovo record conseguito!”)

Stampare messaggio (“Inserire nome Record”)

Acquisire da tastiera il campo nomerecord dell’archivio “recordp” contenuto nell’archivio “p”

Chiamare “ordinamento\_file\_topten”

FINE

**FINE**

# stampare\_interfaccia\_carica\_partita()

**Dati input**

**Dati output**

**Dati locali**

i (intero, contatore)

**ALGORITMO**

**INIZIO**

Stampare a video “CARICA PARTITA”

Assegnare a **i** il valore 0

MENTRE (**i** è minore di 5)

Stampare a video il valore di **i**

Incrementare **i** di 1

FINE

**FINE**

# leggere\_file\_salvataggio()

**Dati input**

leggi (di tipo FILE)

**Dati output**

**Dati locali**

part (di tipo “partita”, riceve un archivio di tipo “partita” letto dal file)

**ALGORITMO**

**INIZIO**

Aprire il file

SE (il file di salvataggio non è presente)

ALLORA

Stampare a video messaggio di errore (“Errore apertura file: File inesistente")

ALTRIMENTI

MENTRE (non si è raggiunta la fine de file)

Leggere un archivio di tipo “part” dal file

SE (è stata letta correttamente una struttura dati dal file)

ALLORA

SE (il campo **save** della struttura “part” ha come valore FALSO)

ALLORA

Stampare a video (“VUOTO”)

ALTRIMENTI

Stampare a video il contenuto del campo “nomepartita” della struttura “part”

FINE

FINE

FINE

FINE

Chiudere il file

**FINE**

# stampare\_interfaccia\_salva\_partita()

**Dati input**

**Dati output**

**Dati locali**

i (intero, contatore)

**ALGORITMO**

**INIZIO**

Stampare a video “SALVA PARTITA”

Assegnare a **i** il valore 0

MENTRE (**i** è minore di 5)

Stampare a video il valore di **i**

Incrementare **i** di 1

FINE

**FINE**

# creare\_file\_salvataggio()

**Dati input**

crea (di tipo FILE)

dimax (intero uguale a 5, indica la dimensione massima di schede di tipo “partita” salvabili sul file di salvataggio partite) (dato globale)

**Dati output**

crea (di tipo FILE)

**Dati locali**

i (intero, contatore)

part (di tipo “partita”, con il campo save inizializzato a FALSO)

**ALGORITMO**

**INIZIO**

Creare il file di salvataggio

Assegnare ad **i** il valore 0

MENTRE (**i** è minore di dimax)

Scrivo su file una struttura dati fi tipo “partita”

Incrementare **i** di 1

FINE

Chiudere il file

**FINE**

# stampare\_interfaccia\_impostazioni()

**Dati input**

**Dati output**

**Dati locali**

**ALGORITMO**

**INIZIO**

Stampare a video “IMPOSTAZIONI”

Stampare a video "1. Cambia simbolo delle pedine”

Stampare a video “2. Cambia nome Giocatori”

Stampare a video “0. Torna al menu principale”

**FINE**

# cambiare\_simbolo\_pedine()

**Dati di input**

**imp** (puntatore di tipo “config”)

**Dati di output**

**Dati di lavoro**

**ALGORITMO**

Chiamare “stampare\_interfaccia\_cambiare\_simbolo\_pedine()”

Chiamare “scegliere\_coppia\_pedine()”

# cambiare\_nome\_giocatori()

**Dati input**

imp (di tipo “config”)

**Dati output**

imp (di tipo “config”)

**Dati locali**

**ALGORITMO**

**INIZIO**

Chiamare “stampare\_interfaccia\_cambiare\_nomi\_giocatori()”

Acquisire da tastiera il campo ng1 dell’archivio “imp”

Acquisire da tastiera il campo ng2 dell’archivio “imp”

**FINE**

# contatore\_tempo()

**Dati input**

reclc (di tipo “rec”)

p (di tipo “partita”)

**Dati output**

reclc (di tipo “rec”)

**Dati locali**

maxtemp (intero, inizializzato a 60, serve per la divisione sessagesimale per il calcolo del tempo)

**ALGORITMO**

**INIZIO**

Fermare il contatore tempo

Calcolare il valore del campo diff\_t della struttura reclc

Assegnare al campo minuti di reclc il valore del campo diff\_t di reclc diviso maxtemp

Assegnare al campo secondi di reclc il valore del campo diff\_t di reclc meno (il campo minuti di reclc\*maxtemp)

Aggiungere al campo minuti di reclc il valore del campo minuti dell’archivio “recordp” contenuto nell’archivio “p”

Aggiungere al campo secondi di reclc il valore del campo secondi dell’archivio “recordp” contenuto nell’archivio “p”

MENTRE (il campo secondi dell’archivio reclc è maggiore di maxtemp)

Incrementare di 1 il campo minuti dell’archivio reclc

Sottrarre al campo secondi dell’archivio reclc il valore di maxtemp

FINE

MENTRE (il campo minuti dell’archivio reclc è maggiore di maxtemp)

Incrementare di 1 il campo ora dell’archivio reclc

Sottrarre al campo minuti dell’archivio reclc il valore di maxtemp

FINE

Aggiungere al campo ora dell’archivio reclc il campo ora del’archivio “recordp” contenuto nell’archivio “p”

Aggiungere al campo diff\_t dell’archivio reclc il campo diff\_t del’archivio “recordp” contenuto nell’archivio “p”

Stampare a video messaggio (“TIME”)

Stampare a video contenuto dei campi (ora, minuti e secondi) dell’archivio reclc

**FINE**

# inserire\_coordinate\_spostamento\_pedina()

**Dati input**

p (di tipo “pedina”, contiene tutte le informazioni riguardo le coordinate di spostamento della pedina)

codESC (intero uguale a 27, indica il codice ASCII del tasto ESC sulla tastiera) (dato globale)

**Dati output**

exit (intero, restituisce l’ultimo valore letto da tastiera)

**Dati locali**

**ALGORITMO**

**INIZIO**

Acquisire da tastiera il campo cordx dell’archivio “p”

Assegnare a exit il campo cordx dell’archivio “p”

SE (exit è diverso da codESC)

ALLORA

Acquisire da tastiera il campo cordy dell’archivio “p”

FINE

**FINE**

# controllare\_coordinate\_spostamento()

**Dati input**

p (di tipo “pedina”)

**Dati output**

ctrl (intero, inizializzato a 2, viene decrementato a 0 se le coordinate sono state inserite correttamente)

**Dati locali**

**ALGORITMO**

**INIZIO**

Assegnare a **ctrl** il valore 2

Controllare se il campo cordx dell’archivio “p” è una lettera maiuscola o minuscola

SE (la lettera è minuscola)

ALLORA

Controllare che il campo contenga una lettera compresa tra “a” ed “e”

SE (il valore del campo **cordx** dell’archivio “p” è una lettera compresa tra “a” ed “e”)

Decrementare **ctrl** di 1

FINE

ALTRIMENTI

SE (il valore del campo **cordx** dell’archivio “p” è una lettera compresa tra “A” ed “E”)

Decrementare **ctrl** di 1

FINE

FINE

Controllare se il campo cordy dell’archivio “p” è un numero

SE (è un numero)

ALLORA

Controllare che il campo contenga un numero compreso tra “1” ed “9”

SE (il valore del campo **cordy** dell’archivio “p” è un numero compreso tra “1” ed “9”)

Decrementare **ctrl** di 1

FINE

FINE

**FINE**

# controllare\_spostamento\_pedina()

**Dati input**

game (di tipo “partita”)

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

verificaptr (di tipo “check”)

**Dati output**

stop (di tipo logico, inizializzato a FALSO, indica se il giocatore ha inserito delle coordinate giuste per effettuare una mossa)

**Dati locali**

esitoR (intero inizializzato a 0, indica se è possibile spostarsi lungo la riga)

esitoC (intero inizializzato a 0, indica se è possibile spostarsi lungo la colonna)

move (di tipo logico, inizializzato a FALSO, indica se è possibile effettuare lo spostamento in diagonale)

**ALGORITMO**

**INIZIO**

SE (il campo **cordy** dell’archivio “dap” è uguale al campo **cordy** dell’archivio “versop” e il campo **cordx** dell’archivio “dap” è uguale al campo **cordx** dell’archivio “versop”)

ALLORA

Assegnare a **stop** il valore VERO

ALTRIMENTI

Assegnare a **esitoR** il valore restituito dalla funzione “controllare\_riga (dap, versop, verificaptr)”

SE (**esitoR** è diverso da zero)

ALLORA

Assegnare a **esitoC** il valore restituito dalla funzione “controllare\_colonna(dap, versop, verificaptr)”

SE (**esitoC** è diverso da 0)

ALLORA

SE (**esitoR** è diverso da 2 e **esitoC** è diverso da 2)

ALLORA

Assegnare a **move** il valore restituito da “controllare\_linea\_campo(dap, verificaptr)”

SE (**move** è uguale a VERO)

ALLORA

Assegnare a **stop** il valore restituito da “effettuare\_spostamento(game, dap, versop)”

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

ALTRIMENTI

Assegnare a **stop** il valore restituito da “effettuare\_spostamento(game, dap, versop)”

FINE

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

FINE

**FINE**

# controllare\_colonna()

**Dati input**

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

ck (di tipo “check”)

**Dati output**

ck (di tipo “check”)

ctrl (intero, può assumere valori compresi tra 0 e 2)

**Dati locali**

**ALGORITMO**

**INIZIO**

Assegnare al campo sx dell’archivio “ck” il valore FALSO

Assegnare al campo dx dell’archivio “ck” il valore FALSO

SE (il campo **cordy** dell’archivio “dap” è maggiore del campo **cordy** dell’archivio “versop”)

ALLORA

SE (il campo **cordy** dell’archivio “dap” è uguale al campo (**cordy+1)** dell’archivio “versop”))

ALLORA

Assegnare al campo sx dell’archivio “ck” il valore VERO

Assegnare a **ctrl** il valore 1

FINE

ALTRIMENTI

SE (il campo **cordy** dell’archivio “versop” è maggiore del campo **cordy** dell’archivio “dap”)

ALLORA

SE (il campo **cordy** dell’archivio “versop” è uguale al campo (**cordy+1)** dell’archivio “dap”))

ALLORA

Assegnare al campo dx dell’archivio “ck” il valore VERO

Assegnare a **ctrl** il valore 1

FINE

ALTRIMENTI

Assegnare a **ctrl** il valore 2

FINE

FINE

**FINE**

# controllare\_riga()

**Dati input**

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

ck (di tipo “check”)

**Dati output**

ck (di tipo “check”)

ctrl (intero, può assumere valori compresi tra 0 e 2)

**Dati locali**

**ALGORITMO**

**INIZIO**

Assegnare al campo up dell’archivio “ck” il valore FALSO

Assegnare al campo down dell’archivio “ck” il valore FALSO

SE (il campo **cordx** dell’archivio “dap” è maggiore del campo **cordx** dell’archivio “versop”)

ALLORA

SE (il campo **cordx** dell’archivio “dap” è uguale al campo (**cordx+1)** dell’archivio “versop”))

ALLORA

Assegnare al campo up dell’archivio “ck” il valore VERO

Assegnare a **ctrl** il valore 1

FINE

ALTRIMENTI

SE (il campo **cordx** dell’archivio “versop” è maggiore del campo **cordx** dell’archivio “dap”)

ALLORA

SE (il campo **cordx** dell’archivio “versop” è uguale al campo (**cordx+1)** dell’archivio “dap”))

ALLORA

Assegnare al campo down dell’archivio “ck” il valore VERO

Assegnare a **ctrl** il valore 1

FINE

ALTRIMENTI

Assegnare a **ctrl** il valore 2

FINE

FINE

**FINE**

# controllare\_linea\_campo()

**Dati input**

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

ck (di tipo “check”)

colonneg (intero uguale a 40, indica le colonne della matrice grafica linearizzata)

matrice\_grafica(matrice linearizzata di caratteri di righeg\*colonneg, contiene la scacchiera di base del campo da gioco)

**Dati output**

muovi (di tipo logico, inizializzato a FALSO, autorizza lo spostamento lungo una diagonale)

**Dati locali**

xR (intero, inizializzato a 4, moltiplicatore della riga per spostarsi correttamente nella “matrice\_grafica” linearizzata)

xC (intero, inizializzato a 5, moltiplicatore della colonna per spostarsi correttamente nella “matrice\_grafica” linearizzata)

**ALGORITMO**

**INIZIO**

SE (il campo **up** dell’archivio “ck” è uguale a VERO e il campo **dx** dell’archivio “ck” è uguale a vero)

ALLORA

SE (il contenuto della **matrice\_grafica** in posizione [(il campo cordx di “dap”\*xR\*colonneg+2)+(il campo

cordy di “dap”\*xC-colonneg)] è uguale a “/”)

ALLORA

Assegnare a **muovi** il valore VERO

ALTRIMENTI

SE (il campo **up** dell’archivio “ck” è uguale a VERO e il campo **sx** dell’archivio “ck” è uguale a vero)

ALLORA

SE (il contenuto della **matrice\_grafica** in posizione [(il campo cordx di “dap”\*xR\*colonneg-2)+(il campo cordy di “dap”\*xC-colonneg)] è uguale a “\”)

ALLORA

Assegnare a **muovi** il valore VERO

ALTRIMENTI

SE (il campo **down** dell’archivio “ck” è uguale a VERO e il campo **dx** dell’archivio “ck” è uguale a vero)

ALLORA

SE (il contenuto della **matrice\_grafica** in posizione [(il campo cordx di “dap”\*xR\*colonneg+1)+(il campo cordy di “dap”\*xC+colonneg)] è uguale a “\”)

ALLORA

Assegnare a **muovi** il valore VERO

ALTRIMENTI

SE (il campo **down** dell’archivio “ck” è uguale a VERO e il campo s**x** dell’archivio “ck” è uguale a vero)

ALLORA

SE (il contenuto della **matrice\_grafica** in posizione [(il campo cordx di “dap”\*xR\*colonneg-1)+(il campo cordy di “dap”\*xC+colonneg)] è uguale a “/”)

ALLORA

Assegnare a **muovi** il valore VERO

FINE

FINE

FINE

FINE

FINE

FINE

FINE

**FINE**

# effettuare\_spostamento()

**Dati input**

g (di tipo “partita”)

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

colonne (intero uguale a 9, indica le colonne del campo da gioco) (dato globale)

**Dati output**

exit (di tipo logico, inizializzato a FALSO, indica se è possibile effettuare o meno lo spostamento)

g (di tipo “partita”)

**Dati locali**

**ALGORITMO**

**INIZIO**

SE (il campo **matrice\_pedine** dell’archivio “g” in posizione (il campo cordx di “versop”\*colonne+il campo cordy di “versop”) è uguale a 0)

ALLORA

SE (il campo turn dell’archivio “g” diviso 2 da resto 0)

ALLORA

Assegnare al campo **matrice\_pedine** dell’archivio “g” in posizione (il campo cordx di “versop”\*colonne+il campo cordy di “versop”) il valore del campo **pedinag2** dell’archivio “g”

ALTRIMENTI

Assegnare al campo **matrice\_pedine** dell’archivio “g” in posizione (il campo cordx di “versop”\*colonne+il campo cordy di “versop”) il valore del campo **pedinag1** dell’archivio “g”

FINE

Assegnare al campo **matrice\_pedine** dell’archivio “g” in posizione (il campo cordx di “dap”\*colonne+il campo cordy di “dap”) il valore 0

ALTRIMENTI

Assegnare ad **exit** il valore VERO

FINE

**FINE**

# scegliere\_coppia\_pedine()

**Dati input**

imp (di tipo “config”)

**Dati output**

imp (di tipo “config”)

**Dati locali**

scelta (intero, può assumere valori compresi tra 0 e 3)

**ALGORITMO**

**INIZIO**

ESEGUI

Stampare a video “Inserire scelta”

Acquisire da tastiera il valore di scelta

SE (**scelta** è minore di 0 o maggiore di 3)

ALLORA

Stampare a video messaggio di errore (“ERRORE! Inserire un numero compreso tra 0 e 3”)

ALTRIMENTI

SE (scelta è uguale a 1)

ALLORA

Assegnare al campo **pg1** dell’archivio “imp” il carattere ASCII 2 “☻”

Assegnare al campo **pg2** dell’archivio “imp” il carattere ASCII 1 “☺”

ALTRIMENTI

SE (**scelta** è uguale a 2)

ALLORA

Assegnare al campo **pg1** dell’archivio “imp” il carattere ASCII 15 “☼”

Assegnare al campo **pg2** dell’archivio “imp” il carattere ASCII 14 “♫”

ALTRIMENTI

SE (**scelta** è uguale a 3)

ALLORA

Assegnare al campo **pg1** dell’archivio “imp” il carattere ASCII 36 “$”

Assegnare al campo **pg2** dell’archivio “imp” il carattere ASCII 156 “£”

FINE

FINE

FINE

FINE

FINCHE’ (**scelta** è minore di 0 o maggiore di 3)

FINE

**FINCHE’**

# scegliere\_slot()

**Dati input**

codESC (intero uguale a 27, indica il codice ASCII del tasto ESC sulla tastiera)

**Dati output**

pos (intero, indica lo slot inserito dall’utente)

**Dati locali**

test (di tipo logico, viene inizializzato a FALSO, indica se l’utente ha inserito uno slot corretto o meno)

**ALGORITMO**

**INIZIO**

ESEGUI

Stampare a video “Inserire numero slot:”

Acquisire da tastiera il valore di **pos**

SE (**pos** è diverso da codESC)

ALLORA

Controllare se il carattere inserito è un numero

SE (pos è un numero)

ALLORA

SE (**pos** è compreso tra “0” e “4”)

ALLORA

Assegnare a **test** il valore VERO

ALTRIMENTI

Stampare messaggio di errore “Errore! Inserire un numero compreso tra 0 e 4”

FINE

ALTRIMENTI

Stampare messaggio di errore "Errore! Carattere non corretto: non hai inserito un numero”

FINE

ALTRIMENTI

Assegnare a **test** il valore VERO

FINE

FINCHE’ (**test** è diverso da VERO)

FINE

**FINE**

# riprendere\_partita()

**Dati input**

p (di tipo “partita”)

**Dati output**

p (di tipo “partita”)

**Dati locali**

**ALGORITMO**

**INIZIO**

Chiamare “creare\_matrice\_grafica()”

Chiamare “stampare\_interfaccia\_campo(p)”

Chiamare “effettuare\_mossa(p)”

**FINE**

# effettuare\_salvataggio\_partita()

**Dati input**

f (di tipo FILE)

pos (intero, che indica la posizione di salvataggio nel file)

p (di tipo “partita”, riceve un partita in corso)

**Dati output**

f (di tipo FILE)

**Dati locali**

**ALGORITMO**

**INZIO**

Aprire il file

Stampare a video (“Inserire nome partita:”)

Acquisire da tastiera il campo **nome partita** della struttura “p”

Assegnare al campo **save** della struttura “**p**” il valore VERO

Salvare nel file la struttura “p” nella posizione **pos**

Chiudere il file

**FINE**

# controllare\_slot\_salvataggio()

**Dati input**

f (di tipo FILE)

pos (intero, indica la posizione di salvataggio nel file)

**Dati output**

test (di tipo logico, riceve VERO se lo slot è vuoto e FALSO in caso contrario)

**Dati locali**

done (di tipo logico, inizializzato a FALSO, riceve il valore VERO se si è inserito un carattere corretto)

scelta (intero, può assumere i caratteri ASCII delle lettere “S”, “s”, “N”, “n”)

part (di tipo “partita”)

**ALGORITMO**

**INIZIO**

Aprire il file

Leggere nel file una struttura di tipo “part” nella posizione pos

SE (il campo **save** della struttura “part” è uguale a FALSO)

ALLORA

Assegnare a **test** il valore VERO

ALTRIMENTI

ESEGUI

Stampare a video (“Stai cercando di salvare nello slot numero”)

Stampare a video il valore di **pos**

Stampare a video (“sei sicuro di voler sovrascrivere lo slot? (s/n)”)

Acquisire da tastiera il valore di **scelta**

Controllare se scelta è una lettera

SE (scelta è una lettera minuscola)

ALLORA

SE (scelta è uguale al carattere ASCII di “s”)

ALLORA

Assegnare a test il valore VERO

Assegnare a done il valore VERO

ALTRIMENTI

SE (**scelta** è uguale al carattere ASCII di “n”)

ALLORA

Assegnare a **test** il valore FALSO

Assegnare a **done** il valore VERO

FINE

FINE

ALTRIMENTI

SE (scelta è uguale al carattere ASCII di “S”)

ALLORA

Assegnare a test il valore VERO

Assegnare a done il valore VERO

ALTRIMENTI

SE (**scelta** è uguale al carattere ASCII di “N”)

ALLORA

Assegnare a **test** il valore FALSO

Assegnare a **done** il valore VERO

FINE

FINE

FINE

SE (done è uguale a FALSO)

ALLORA

Stampare messaggio di errore (“Errore! Hai inserito un carattere non accettato: inserire s(si)/n(no)”)

FINE

FINCHE’ (**done** è diverso da VERO)

FINE

FINE

Chiudere il file

**FINE**

# controllare\_slot\_caricamento()

**Dati input**

f (di tipo FILE)

pos (intero, indica lo slot di caricamento della partita nel file)

**Dati output**

test (di tipo logico, riceve VERO se lo slot contiene una partita salvata)

**Dati locali**

part (di tipo “partita”)

**ALGORITMO**

**INIZIO**

Aprire il file

Leggere nel file una struttura di tipo “part” nella posizione pos

SE (**save** è uguale a vero)

ALLORA

Assegnare a **test** il valore VERO

ALTRIMENTI

Stampare messaggio di errore ("Errore! Impossibile caricare la partita: slot vuoto”)

FINE

Chiudere il file

**FINE**

# estrarre\_partita()

**Dati input**

f (di tipo FILE)

pos (intero, indica lo slot di caricamento della partita nel file)

**Dati output**

cpart (di tipo “partita”, contiene tutte le informazioni riguardo alla partita caricata)

**Dati locali**

**ALGORITMO**

**INIZIO**

Aprire il file

Leggere un archivio di tipo “part” dal file alla posizione indicata da pos

Chiudere il file

**FINE**

# stampare\_interfaccia\_cambiare\_simbolo\_pedine()

**Dati input**

**Dati output**

**Dati locali**

**ALGORITMO**

**INIZIO**

Stampare a video “IMPOSTAZIONI-Cambia simbolo delle pedine”

Stampare a video “Giocatore 1 Giocatore 2”

Stampare a video “1. ☻ ☺”

Stampare a video “2. ☼ ♫”

Stampare a video “3. $ £”

Stampare a video “0. Torna al menù Impostazioni”

**FINE**

# stampare\_interfaccia\_cambiare\_nomi\_giocatori()

**Dati input**

**Dati output**

**Dati locali**

**ALGORITMO**

**INIZIO**

Stampare a video “IMPOSTAZIONI-Cambia nome Giocatori”

Stampare a video “Giocatore 1:”

Stampare a video “Giocatore 2:”

**FINE**

# scegliere\_modalita\_cattura()

**Dati input**

**Dati output**

quale (intero, può assumere i valori 1 e 2)

**Dati locali**

**ALGORITMO**

**INIZIO**

ESEGUI

Stampare a video “Scegliere modalità di cattura”

Stampare a video “1. Avvicinamento”

Stampare a video “2. Allontanamento”

Acquisire da tastiera il valore **quale**

SE (**quale** è minore di 1 o maggiore di 2)

Stampare a video messaggio di errore "Errore! Numeri accettati: 1 e 2"

FINE

FINCHE’ (**quale** è minore di 1 o maggiore di 2)

FINE

**FINE**

# controllare\_modalita\_cattura()

**Dati input**

p (di tipo “partita”)

cordpt (di tipo “check”)

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

pedg (intero, indica il giocatore che sta effettuando la mossa)

vicinopt (di tipo logico, inizializzato a FALSO, cambia il suo valore in VERO se la modalità di cattura è per “avvicinamento”)

lontanopt (di tipo logico, inizializzato a FALSO, cambia il suo valore in VERO se la modalità di cattura è per “allontanamento”)

colonne (intero uguale a 9, indica le colonne del campo da gioco) (dato globale)

**Dati output**

vicinopt (di tipo logico)

lontanopt (di tipo logico)

**Dati locali**

ctrl (di tipo “pedina”, i campi assumono diversi valori in base alle direzioni di spostamento della pedina)

j (intero, uguale a 1)

**ALGORITMO**

**INIZIO**

Inizializzare lontanopt a FALSO

Inizializzare vicinopt a FALSO

Inizializzare j a 1

SE (il campo **up** di “cordpt” è uguale a VERO e il campo **sx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo (**cordx**+**j)** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **(cordy**+**j)** (di tipo “dap”)

SE (il campo **matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **(cordx**-**j)** di “versop”)

Assegnare al campo **cordy** di “ctrl” il valore del campo (**cordy**-**j**) di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

ALTRIMENTI

SE (il campo **up** di “cordpt” è uguale a VERO e il campo **dx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo (**cordx**+**j)** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **(cordy**-**j)** (di tipo “dap”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **(cordx**-**j)** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo (**cordy**+**j**) di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

ALTRIMENTI

SE (il campo **down** di “cordpt” è uguale a VERO e il campo **sx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo (**cordx**-**j)** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **(cordy**+**j)** (di tipo “dap”)

SE (il campo **matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy) di ”ctrl” è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **(cordx**+**j)** di “versop”)

Assegnare al campo **cordy** di “ctrl” il valore del campo (**cordy**-**j**) di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

ALTRIMENTI

SE (il campo **down** di “cordpt” è uguale a VERO e il campo **dx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo (**cordx-j)** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **(cordy**-**j)** (di tipo “dap”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **(cordx+j)** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo (**cordy**+**j**) di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

ALTRIMENTI

SE (il campo **up** di “cordpt” è uguale a FALSO e il campo **down** di “cordpt” è uguale a FALSO)

ALLORA

SE (il campo **sx** di tipi “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx** di tipo “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **(cordy**-**j)** (di tipo “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”) è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx** di “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo (**cordy**+**j**) di “dap”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

FINE

SE (il campo **dx** di tipi “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **(cordy**-**j)** (di tipo “dap”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”) è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo (**cordy**+**j**) di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

FINE

ALTRIMENTI

SE (il campo **sx** di “cordpt” è uguale a FALSO e il campo **dx** di “cordpt” è uguale a FALSO)

ALLORA

SE (il campo **up** di tipi “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo (**cordx+j)** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy** (di tipo “dap”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”) è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **(cordx-j)** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy** di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

FINE

SE (il campo **down** di tipi “cordpt” è uguale a VERO)

ALLORA

Assegnare al campo **cordx** di “ctrl” il valore del campo (**cordx**-**j)** di tipo “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy** (di tipo “dap”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”) è uguale a **pedg**)

ALLORA

Assegnare a **lontanopt** il valore VERO

FINE

Assegnare al campo **cordx** di “ctrl” il valore del campo **(cordx**+**j)** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy** di “versop”)

SE (**matrice\_pedine** in posizione (cordx di “ctrl”\*colonne+cordy di ”ctrl”] è uguale a **pedg**)

ALLORA

Assegnare a **vicinopt** il valore VERO

FINE

FINE

FINE

FINE

FINE

FINE

FINE

FINE

**FINE**

# mangiare\_pedina\_allontanamento()

**Dati input**

p (di tipo “partita”)

cordpt (di tipo “check”)

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

pedg (intero, indica il giocatore che sta effettuando la mossa)

colonne (intero uguale a 9, indica le colonne del campo da gioco) (dato globale)

**Dati output**

p (di tipo “partita”)

**Dati locali**

i (intero, contatore)

j (intero, contatore)

stop (di tipo logico, assume il valore VERO quando si incontra una pedina non avversaria)

ctrl (di tipo “pedina”, i campi assumono diversi valori in base alle direzioni di spostamento della pedina)

**ALGORITMO**

**INIZIO**

SE (il campo **up** di “cordpt” è uguale a VERO e il campo **sx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx+j** di “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy+j** di “dap”

MENTRE (il campo **cordx** di “ctrl” è minore o uguale a (righe -1) e il campo **cordy** di “ctrl” è minore o uguale a (colonne-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore del campo **cordx** di “ctrl” di 1

Incrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **up** di “cordpt” è uguale a VERO e il campo **dx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx+j** di “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy-j** di “dap”

MENTRE (il campo **cordx** di “ctrl” è minore o uguale a (righe-1) e il campo **cordy** di “ctrl” è magggiore o uguale di 0 e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore del campo **cordx** di “ctrl” di 1

Decrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **down** di “cordpt” è uguale a VERO e il campo **sx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx-j** di “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy+j** di “dap”

MENTRE (il campo **cordx** di “ctrl” è maggiore o uguale a 0 e il campo **cordy** di “ctrl” è minore o uguale a (colonne-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore del campo **cordy** di “ctrl” di 1

Decrementare il valore del campo **cordx** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **down** di “cordpt” è uguale a VERO e il campo **dx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx-j** di “dap”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy-j** di “dap”

MENTRE (il campo **cordx** di “ctrl” è maggiore o uguale a 0 e il campo **cordy** di “ctrl” è maggiore o uguale a 0 e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Decrementare il valore del campo **cordx** di “ctrl” di 1

Decrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **up** di “cordpt” è uguale a FALSO e il campo **down** di “cordpt” è uguale a FALSO)

ALLORA

SE (il campo sx di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare a **i** il campo cordy di “dap”

MENTRE (i è minore (colonne -1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “dap” \*colonne+cordy di “dap”)+j) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “dap” \*colonne+cordy di “dap”)+j) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

SE (il campo dx di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **i** il valore 0

Assegnare a **j** il valore 1

MENTRE (i è minore del campo cordy di “dap” e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “dap” \*colonne+cordy di “dap”)-j) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “dap” \*colonne+cordy di “dap”)-j) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

ALTRIMENTI

SE (il campo **sx** di “cordpt” è uguale a FALSO e il campo **dx** di “cordpt” è uguale a FALSO)

ALLORA

SE (il campo up di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare a **i** il campo cordx di “dap”

MENTRE (i è minore di (righe-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “dap” \*colonne+cordy di “dap”)+(j\*colonne)) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “dap” \*colonne+cordy di “dap”)+(j\*colonne)) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

SE (il campo down di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **i** il valore 0

Assegnare a **j** il valore 1

MENTRE (i è minore di (righe-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “dap” \*colonne+cordy di “dap”)-(j\*colonne)) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “dap” \*colonne+cordy di “dap”)-(j\*colonne) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

FINE

FINE

FINE

FINE

FINE

**FINE**

# mangiare\_pedina\_avvicinamento()

**Dati input**

p (di tipo “partita”)

cordpt (di tipo “check”)

dap (di tipo “pedina”, indica le coordinate di partenza della pedina)

versop (di tipo “pedina”, indica le coordinate di arrivo della pedina)

pedg (intero, indica il giocatore che sta effettuando la mossa)

colonne (intero uguale a 9, indica le colonne del campo da gioco) (dato globale)

**Dati output**

p (di tipo “partita”)

**Dati locali**

i (intero, contatore)

j (intero, contatore)

stop (di tipo logico, assume il valore VERO quando si incontra una pedina non avversaria)

ctrl (di tipo “pedina”, i campi assumono diversi valori in base alle direzioni di spostamento della pedina)

**ALGORITMO**

**INIZIO**

SE (il campo **up** di “cordpt” è uguale a VERO e il campo **sx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx-j** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy-j** di “versop”

MENTRE (il campo **cordx** di “ctrl” è maggiore o uguale a 0 e il campo **cordy** di “ctrl” è maggiore o uguale a 0 e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Decrementare il valore del campo **cordx** di “ctrl” di 1

Decrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **up** di “cordpt” è uguale a VERO e il campo **dx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx-j** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy+j** di “versop”

MENTRE (il campo **cordx** di “ctrl” è maggiore o uguale a 0 e il campo **cordy** di “ctrl” è minore o uguale di (colonne -1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Decrementare il valore del campo **cordx** di “ctrl” di 1

Incrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **down** di “cordpt” è uguale a VERO e il campo **sx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx+j** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy-j** di “versop”

MENTRE (il campo **cordx** di “ctrl” è minore o uguale di (righe-1) e il campo **cordy** di “ctrl” è maggiore o uguale a 0 e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore del campo **cordx** di “ctrl” di 1

Decrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **down** di “cordpt” è uguale a VERO e il campo **dx** di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare al campo **cordx** di “ctrl” il valore del campo **cordx+j** di “versop”

Assegnare al campo **cordy** di “ctrl” il valore del campo **cordy+j** di “versop”

MENTRE (il campo **cordx** di “ctrl” è minore o uguale di (righe-1) e il campo **cordy** di “ctrl” è minore o uguale di (colonne-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione (cordx di “ctrl” \*colonne+cordy di “ctrl”) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore del campo **cordx** di “ctrl” di 1

Incrementare il valore del campo **cordy** di “ctrl” di 1

FINE

ALTRIMENTI

SE (il campo **up** di “cordpt” è uguale a FALSO e il campo **down** di “cordpt” è uguale a FALSO)

ALLORA

SE (il campo sx di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare a **i** il valore 0

MENTRE (i è minore del campo **cordy** di “versop” e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “versop” \*colonne+cordy di “versop”)-j) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “versop” \*colonne+cordy di “versop”)-j) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

SE (il campo dx di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **i** il campo cordy di “versop”

Assegnare a **j** il valore 1

MENTRE (i è minore di (colonne-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “versop” \*colonne+cordy di “versop”)+j) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “versop” \*colonne+cordy di “versop”)+j) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

ALTRIMENTI

SE (il campo **sx** di “cordpt” è uguale a FALSO e il campo **dx** di “cordpt” è uguale a FALSO)

ALLORA

SE (il campo up di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **j** il valore 1

Assegnare a **i** il valore 0

MENTRE (i è minore del campo **cordx** di “versop” e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “versop” \*colonne+cordy di “versop”)-(j\*colonne)) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “versop” \*colonne+cordy di “versop”)-(j\*colonne)) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

SE (il campo down di “cordpt” è uguale a VERO)

ALLORA

Assegnare a **stop** il valore FALSO

Assegnare a **i** il campo cordx di “versop”

Assegnare a **j** il valore 1

MENTRE (i è minore di (righe-1) e **stop** è diverso da VERO)

SE (il campo **matrice\_pedine** di “p” nella posizione ((cordx di “versop” \*colonne+cordy di “versop”)+(j\*colonne)) è uguale a **pedg**)

ALLORA

Assegnare alla **matrice\_pedine** di “p” in posizione ((cordx di “versop” \*colonne+cordy di “versop”)+(j\*colonne) il valore 0

ALTRIMENTI

Assegnare a **stop** il valore VERO

FINE

Incrementare il valore di j di 1

Incrementare il valore di i di 1

FINE

FINE

FINE

FINE

FINE

FINE

FINE

**FINE**

# leggere\_file\_topten()

**Dati input**

txt (di tipo FILE)

**Dati output**

**Dati locali**

tt (di tipo “rec”, contiene una struttura dati letta dal file)

**ALGORITMO**

**INIZIO**

Aprire il file

MENTRE (non è stata raggiunta la fine del file)

Leggere dal file un archivio “tt”

Stampare a video il valore del campo **ora** dell’archivio “tt”

Stampare a video il valore del campo **minuti** dell’archivio “tt”

Stampare a video il valore del campo **secondi** dell’archivio “tt”

Stampare a video il valore del campo **nomerecord** dell’archivio “tt”

Incrementare i di 1

FINE

Chiudere il file

**FINE**

# ordinamento\_file\_topten()

**Dati input**

p (di tipo “partita”)

maxtt (intero uguale a 11, indica la dimensione massima del vettore di schede di tipo “rec”)

**Dati output**

**Dati locali**

txt (di tipo FILE)

i (intero, contatore)

j (intero, contatore)

dim (intero, inizializzato a 0, indica quanti tempi Record sono salvati sul file)

c (carattere, legge un carattere dal file)

tt (di tipo “rec”, riceve un tempo Record letto da file)

vet\_tt (sequenza di “maxtt” di tipo “rec”, contiene tutti i tempi Record contenuti nel file)

w (di tipo “rec”, variabile di scambio per l’ordinamento)

**ALGORITMO**

**INIZIO**

Aprire il file

Leggere un carattere dal file e assegnarlo a c

SE (c è diverso dalla fine del file)

ALLORA

Scrivere sul file l’archivio “p”

Chiudere il file

ALTRIMENTI

MENTRE (ci sono archivi “tt” sul file)

SE (dim è minore di (maxtt-1))

ALLORA

Assegnare al vettore vet\_tt nella posizione dim l’archivio “tt”

Incrementare dim di 1

FINE

FINE

Chiudere il file

Assegnare al vettore vet\_tt nella posizione dim il campo recordp dell’archivio ”p”

Incrementare dim

Assegnare a j il valore 1

MENTRE (j è minore di dim)

Assegnare ad i il valore 0

MENTRE (i è minore di (dim-1))

SE (il campo diff\_t del vettore vet\_tt in posizione i è maggiore del campo campo diff\_t del vettore vet\_tt in posizione (i+1))

ALLORA

Assegnare all’archivio w il valore del vettore vet\_tt in posizione i

Assegnare al vettore vet\_tt in posizione i il valore del vettore vet\_tt in posizione (i+1)

Assegnare al vettore vet\_tt in posizione (i+1) il valore dell’archivio w

FINE

Incrementare i di 1

FINE

Incrementare j di 1

FINE

Creare un nuovo file con il vettore ordinato

Assegnare a j il valore 0

SE (dim è uguale a maxtt)

ALLORA

Decrementare dim di 1

FINE

MENTRE (j è minore di dim)

Scrivere sul file il contenuto del vettore vet\_tt in posizione j

Incrementare j di 1

FINE

Chiudere il file

FINE

**FINE**

# controllare\_pedine\_campo()

**Dati input**

p (di tipo “partita”)

pedg (intero, indica il giocatore che sta effettuando la mossa)

righe(intero uguale a 5, indica le righe della “matrice\_pedine” linearizzata) (dato globale)

colonne (intero uguale a 9, indica le colonne della “matrice\_pedine” linearizzata) (dato globale)

**Dati output**

win (di tipo logico, inizializzato a VERO, cambia il suo valore in FALSO se sono ancora presenti sul campo pedine del giocatore avversario)

**Dati locali**

r (intero, indica le righe di matrice\_pedine)

c (intero, indica le colonne di matrice\_pedine)

**ALGORITMO**

**INIZIO**

Inizializzare win a VERO

MENTRE (r è minore di righe e win è diverso da FALSO)

Assegnare a c il valore 0

MENTRE (c è minore di colonne e win è diverso da FALSO)

SE (il campo matrice\_pedine di “p” in posizione (r\*colonne+c) è uguale pedg)

ALLORA

Assegnare a win il valore FALSO

FINE

Incrementare c di 1

FINE

Incrementare r di 1

FINE

**FINE**

# controllare\_nuovo\_record()

**Dati input**

p (di tipo “partita”)

maxtt (intero uguale a 11, indica la dimensione massima del vettore di schede di tipo “rec”) (dato globale)

**Dati output**

stop (di tipo logico inizializzato a FALSO, cambia il suo valore in VERO se c’è stato un nuovo tempo record)

**Dati locali**

i (intero, contatore)

ftxt (di tipo FILE)

tt (di tipo “rec”, riceve un tempo Record letto da file)

dim (interi, inizializzato a 0, indica quanti record sono già presenti su file)

c (carattere, riceve un carattere letto da file)

**ALGORITMO**

**INIZIO**

Aprire il file

Leggere un carattere dal file e associarlo a c

SE (c è uguale alla fine del file)

ALLORA

Assegnare a stop il valore VERO

ALTRIMENTI

MENTRE (ci sono tempi Record sul file)

Incrementare dim di 1

SE (dim è minore di (maxtt-1))

ALLORA

Assegnare a stop il valore VERO

ALTRIMENTI

MENTRE (ci sono tempi record sul file)

SE (il campo diff\_t dell’archivio “recordp” contenuto nell’archivio “p” è minore del campo diff\_t dell’archivio “tt”)

ALLORA

Assegnare a stop il valore VERO

FINE

FINE

FINE

FINE

FINE

Chiudere il file

**FINE**