Pseudocodice funzioni e comandi avventura testuale **Gruppo 11**

INDICE

Funzione main

Nome	Descrizione	Tipo	Vincoli
scelta_menu	Scelta dell'utente	intero	compresa tra 1 e 5

Nome	Descrizione	Tipo	Vincoli
scelta_menu	Scelta dell'utente	intero	compreso tra 1 e 5

```
MENTRE(leggere_nome_giocatore(nome_giocatore) = falso)
FINE

scelta_menu := -1
MENTRE(scelta_menu < 1 OR scelta_menu > 5)
      mostrare_menu_gioco()
      scelta_menu := leggere_intero()

      SE(scelta_menu = 1)
          ALLORA
          start_game(vero)
      ALTRIMENTI SE(scelta_menu = 2)
          ALLORA
          start_game(falso)
      ALTRIMENTI SE(scelta_menu = 3)
          ALLORA
          mostrare_guida_gioco()
      ALTRIMENTI SE(scelta_menu = 4)
          ALLORA
          mostrare_informazioni_gioco()
FINE
```

Funzione exit_game

Nome	Descrizione	Tipo	Vincoli
num_vite_corrente	Numero vite corrente del giocatore	intero	compreso tra 0 e 3
esito_controllo	Flag che se posto a falso indica che si è verificato un errore con i file	booleano	vero=no errori file falso=errori file
game_win	Flag che indica se il giocatore ha preso la corona (e quindi vinto)	booleano	vero=vinto il gioco falso=no vinto il gioco

Nome	Descrizione	Tipo	Vincoli
exit_game	Flag che indica se interrompere il gioco	booleano	vero=interrompere falso=no interrompere

```
exit_game := falso
SE (num_vite_corrente < 0)
    ALLORA
    exit_game := vero
ALTRIMENTI SE(esito_controllo = falso)
    ALLORA
    exit_game := vero
ALTRIMENTI SE(game_win = vero)
     ALLORA
     exit_game := vero

FINE
```

Funzione start_game

Nome	Descrizione	Tipo	Vincoli
nuova_partita	Flag che se posto a vero indica di iniziare una nuova partita	booleano	vero=nuova partita falso=bisogna caricare la partita dell'utente
NUM_VITE_PROTAGONISTA_INIZIALE	Numero di vite iniziale	intero	=3
ID_STANZA_PROTAGONISTA_INIZIALE	Id della stanza in cui in cui si trova il protagonista a inizio gioco	intero	=18

Nome	Descrizione	Tipo	Vincoli
mappa	Mappa del gioco caricata e riempita	mappa	

Nome	Descrizione	Tipo	Vincoli
num_vite_corrente	Numero vite corrente del giocatore	intero	compreso tra 0 e 3
esito_controllo	Flag che se posto a falso indica che si è verificato un errore con i file	booleano	vero=no errori file falso=errori file
game_win	Flag che indica se il giocatore ha preso la corona (e quindi vinto)	booleano	vero=vinto il gioco falso=no vinto il gioco
id_ultimo_oggetto_ispezionato	Id dell'ultimo oggetto ispezionato	intero	=-1 se non si è ispezionato alcun ogetto

```
// Queste quattro variabili sono globali, cioè utilizzabili in ogni modulo
game_win := falso
esito_controllo := vero
id_ultimo_oggetto_ispezionato := -1
num_vite_corrente := NUM_VITE_PROTAGONISTA_INIZIALE

init_dati(mappa_gioco, lista_comandi)
SE(nuova_partita = vero)
    ALLORA
    creare_protagonista(protagonista, inventario)
    aggiornare_id_stanza_corrente(mappa_gioco, ID_STANZA_PROTAGONISTA_INIZIALE)
    nascondere_oggetti_rilasciati(mappa_gioco)

    ALTRIMENTI
    caricare_partita(protagonista, inventario, mappa_gioco)
FINE
```

**   // Dopo aver creato/caricato le strutture dati (della mappa, personaggio) controllo che non si sia verificato un errore con i file.**

```
SE(exit_game(num_vite_corrente, esito_controllo, game_win) = falso)
```

**   // Inizio col gioco vero e proprio**

```
    eseguire_guarda(mappa_gioco, riga_stanza_corrente, col_stanza_corrente)
    id_stanza_precedente := -1

    MENTRE(exit_game(num_vite_corrente, esito_controllo, game_win) = falso)
        Leggere_stringa_tastiera(comando)
        SE(eseguire_scanner(comando) = vero)
           ALLORA
           stampa("nel comando ci sono simboli non ammessi")
        FINE
    FINE
FINE
```

Funzione init_dati

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa che deve essere creata e caricata	mappa	
lista_comandi	Lista comandi	array 1 dimensione comandi	dimensione=16

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa del gioco in cui sono stati caricati tutti gli oggetti	mappa	

Nome	Descrizione	Tipo	Vincoli
i_op	Contatore	intero	>=0

```
i_op:=0
MENTRE(i_op <= 5 AND exit_game() = falso)
     SE(i=0)
         ALLORA
         creare_mappa("dimensione_mappa.txt", mappa_gioco)
     ALTRIMENTI SE(i_op=1)
         ALLORA
         aggiungere_stanze_in_mappa(mappa_gioco)
     ALTRIMENTI SE(i_op=2)
         ALLORA
         aggiungere_porte_in_mappa(“lista_oggetti_porte.txt",mappa_gioco)
     ALTRIMENTI SE(i_op=3)
         ALLORA
         aggiungere_cibi_in_mappa ("lista_oggetti_cibo.txt" ,mappa_gioco)
     ALTRIMENTI SE(i_op=4)
         ALLORA
         aggiungere_armi_in_mappa("lista_oggetti_armi.txt" , mappa_gioco)
     ALTRIMENTI SE(i_op=5)
         ALLORA
         aggiungere_personaggi_in_mappa("lista_oggetti_personaggi.txt", mappa_gioco)
     FINE
     i_op:=i_op+1
FINE
```

Funzione leggere_nome_giocatore

Nome	Descrizione	Tipo	Vincoli
nome_giocatore	Stringa che contiene il nome del giocatore	array 1 dimensione caratteri (statico)	dim=10
NUM_VITE_PROTAGONISTA_INIZIALE	Numero di vite iniziale	intero	=3
ID_STANZA_PROTAGONISTA_INIZIALE	Id della stanza in cui in cui si trova il protagonista a inizio gioco	intero	=18

Nome	Descrizione	Tipo	Vincoli
esito_lettura	esito della lettura del nome	booleano	vero = nome corretto falso= nome non corretto

Nome	Descrizione	Tipo	Vincoli
num_vite_corrente	Numero vite corrente del giocatore	intero	compreso tra 0 e 3
esito_controllo	Flag che se posto a falso indica che si è verificato un errore con i file	booleano	vero=no errori file falso=errori file
game_win	Flag che indica se il giocatore ha preso la corona (e quindi vinto)	booleano	vero=vinto il gioco falso=no vinto il gioco
id_ultimo_oggetto_ispezionato	Id dell'ultimo oggetto ispezionato	intero	=-1 se non si è ispezionato alcun ogetto

```
Leggere_stringa(nome_giocatore)
SE(leggere_lunghezza(nome_giocatore) > 10+1)
   ALLORA
   esito_lettura:=falso
FINE

//Controllo che il nome utente non sia vuoto
Nome_giocatore[leggere_lunghezza(nome_giocatore)] := '\0'
SE(nome_giocatore[0] = '\0')
    ALLORA
    Esito_lettura := falso
FINE

//Applicazione dell'espressione regolare
regex_match := applicare_regex(nome_giocatore, "[^a-zA-Z0-9][^a-zA-Z0-9]*")
SE(regex_match = vero)
    ALLORA
    Esito_lettura := falso

    ALTRIMENTI
    Esito_lettura := vero
FINE
```

Funzione eseguire_scanner

Nome	Descrizione	Tipo	Vincoli
comando_utente	Comando immesso dall'utente	array 1 dimensione di caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
regex_match	flag che indica se una o più parole del linguaggio descritto dall'espressione regolare e' sottostringa del comando	booleano	vero=c'è un match falso= non c'è un match

Nome	Descrizione	Tipo	Vincoli
regex_testo	Espressione regolare applicata al comando	array 1 dimensione caratteri (statico)	

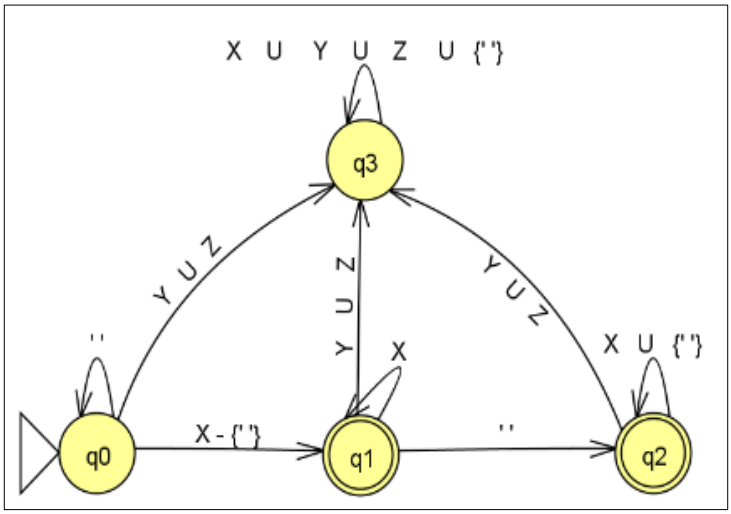
```
Regex_testo := [^a-zA-Z ][^a-zA-Z ]* //descrive il linguaggio formato da tutte le parole che non sono formate ne' da simboli alfabetici ne' da spazi.

//ottimizzazione stringa di input
togliere_maiuscole_da_stringa(comando_utente)
regex_match := applicare_espressione_regolare(regex_testo, comando_utente)
SE(comando_utente = "" OR comando_utente = " ")
   ALLORA
   regex_match:= vero
FINE
SE(regex_match = falso)
   ALLORA

   //creazione di due token
   verbo_comando     := prendere_fino_a_carattere(comando_utente, " ")
   argomento_comando := prendere_fino_a_carattere(comando_utente, "\0")

   eseguire_parser(verbo_comando, argomento_comando)
FINE
```

La funzione applicare_espressione_regolare e' una funzione per la quale esiste una libreria di gestione delle espressioni regolari. L'algoritmo consiste sostanzialmente nell'applicare una espressione regolare e verificare se almeno una parola del linguaggio descritto da tale espressione regolare compare nella stringa di input.



A destra è rappresentato l'automa che riconosce il linguaggio accettato dal gioco.
X = {A,B,C,…,Z,a,b,c,…z}
Y={0,1,2,…,9}
Z={!,",$,…}
q0 stato iniziale,
q1 è lo stato in cui l'automa si porta leggendo la prima parola del comando (che sarebbe il verbo)
q2 è lo stato in cui l'automa di porta leggendo le parole restanti del comando (che sarebbero l'argomento del comando)
q3 è uno stato pozza. Lo si raggiunge se nella parola compaiono simboli non alfabetici, né numerici.

Funzione togliere_maiuscole_da_stringa

Nome	Descrizione	Tipo	Vincoli
stringa_da_elaborare	stringa in cui vogliamo togliere le maiuscole	array 1 dimensione di caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
stringa_da_elaborare	stringa in cui tutte le maiuscole sono rese minuscole	array 1 dimensione caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
i	contatore	intero	>=0

```
i:=0
MENTRE(i<lunghezza(stringa_da_elaborare))
   stringa_da_elaborare[i] = minuscolo(stringa_da_elaborare[i])
   i:=i+1
FINE
```

Funzione ricercare_attributo

Nome	Descrizione	Tipo	Vincoli
attributi_oggetto	array dove sono memorizzati gli attributi di un oggetto	array 1 dimensione di caratteri (statico)	dimensione=11
attributo_da_ricercare	carattere da ricercare nell'elenco degli attributi	carattere	

Nome	Descrizione	Tipo	Vincoli
esito_ricerca	numero che indica la posizione dell'attributo da ricercare in attributi_oggetto	intero	>=0 se trovato, -1 se non trovato

Nome	Descrizione	Tipo	Vincoli
fine_ricerca	flag che indica se l'elemento è stato trovato, dunque si deve interrompere la ricerca	booleano	
i	contatore	intero	>=0

```
esito_ricerca := -1
fine_ricerca := falso
i:=0
MENTRE(i<11 AND fine_ricerca = falso)
      SE(attributi_oggetto[i] = attributo_da_ricercare)
            ALLORA
            fine_ricerca := vero
            esito_ricerca := i
      FINE
      i:=i+1
FINE
```

Salvare_partita

Nome	Descrizione	Tipo	Vincoli
nome_giocatore	Nome del giocatore a cui si devono salvare i progressi	array 1 dimensione caratteri (statico)	dimensione=20
protagonista	Protagonista del gioco	oggetto_personaggio	
inventario	oggetti presi dal protagonista	array 1 dimensione oggetti (statico)	dimensione=80
num_vite_corrente	Numero di vite corrente	intero	>0

Nome	Descrizione	Tipo	Vincoli
file_salvataggio	File di salvataggio	file binario	

Nome	Descrizione	Tipo	Vincoli
stanza_da_scrivere	Stanza da salvare	stanza	

```
SE( aprire_file(nome_giocatore+”_save”) = vero)
  ALLORA

Scrivere_su_file(num_vite_corrente, file_salvataggio)
Scrivere_su_file (id_stanza_corrente_protagonista, file_salvataggio)
Scrivere_su_file(protagonista, file_salvataggio)
Scrivere_su_file(inventario, file_salvataggio)
Scrivere_su_file(mappa_gioco, file_salvataggio)
i:=0
MENTRE(i<leggere_righe(mappa_gioco))
    j:=0
    MENTRE(j<leggere_col(mappa_gioco))
        stanza_da_scrivere := leggere_stanza_da_mappa(mappa_gioco,i,j)
        Scrivere_su_file(stanza_da_scrivere, file_salvataggio )
        J:=j+1
    FINE
    i:=i+1
FINE

I:=0
MENTRE(i<NUM_PERSONAGGI_MAPPA_MAX)

    personaggio_corrente := leggere_personaggio_da_mappa(mappa_gioco, i)
    Scrivere_su_file(personaggio_corrente, file_salvataggio )

        i:=i+1

FINE

FINE
chiudere_file(file_salvataggio)
```

Caricare_partita

Nome	Descrizione	Tipo	Vincoli
nome_giocatore	Nome del giocatore a cui si devono caricare i progressi	array 1 dimensione caratteri (statico)	dimensione=20
protagonista	Protagonista del gioco	oggetto_personaggio	
inventario	oggetti presi dal protagonista	array 1 dimensione oggetti (statico)	dimensione=80
num_vite_corrente	Numero di vite corrente	intero	>0

Nome	Descrizione	Tipo	Vincoli
file_caricamento	File di caricamento	file binario	

Nome	Descrizione	Tipo	Vincoli
stanza_da_scrivere	Stanza da caricare nella mappa	stanza	

```
SE( aprire_file(nome_giocatore+”_save”) = vero)
  ALLORA

leggere_da_file(num_vite_corrente, file_caricamento)
leggere_da_file (id_stanza_corrente_protagonista, file_ caricamento)
leggere_da_file (protagonista, file_ caricamento)
leggere_da_file (inventario, file_ caricamento)
leggere_da_file (mappa_gioco, file_ caricamento)
i:=0
MENTRE(i<leggere_righe(mappa_gioco))
    j:=0
    MENTRE(j<leggere_col(mappa_gioco))
        leggere_da_file (stanza_da_scrivere, file_ caricamento)
         scrivere_stanza_in_mappa(mappa_gioco,i,j, stanza_da_scrivere)
        J:=j+1
    FINE
    i:=i+1
FINE

I:=0
MENTRE(i<NUM_PERSONAGGI_MAPPA_MAX)
    leggere_da_file(personaggio_corrente, file_ caricamento)
    personaggio_corrente := leggere_personaggio_da_mappa(mappa_gioco, i)
    i:=i+1
FINE

FINE
chiudere_file(file_caricamento)
```

creare_protagonista

Nome	Descrizione	Tipo	Vincoli
protagonista	Protagonista i cui attributi devono essere inizializzati	oggetto_personaggio	
SALUTE_PROTAGONISTA_INIZIALE	Salute del protagonista a inizio partita	intero	=20
FORZA_PROTAGONISTA_INIZIALE	Forza del protagonista a inizio partita	intero	=5

Nome	Descrizione	Tipo	Vincoli
protagonista	Protagonista con gli attributi inizializzati	oggetto_personaggio	valori degli attributi: salute=20, forza=5

```
scrivere_salute_personaggio(protagonista, SALUTE_PROTAGONISTA_INIZIALE, vero)
scrivere_forza_personaggio (protagonista, FORZA_PROTAGONISTA_INIZIALE  )
init_inventario(inventario)
```

init_inventario

Nome	Descrizione	Tipo	Vincoli
inventario	Inventario senza oggetti	array 1 dimensione oggetti (statico)	dim=80
NUM_OGGETTI INVENTARIO_MAX	Numero massimo degli oggetti dell'inventario	intero	=80

Nome	Descrizione	Tipo	Vincoli
inventario	Inventario inizializzato, cioè avente 80 oggetti non validi	array 1 dimensione oggetti (statico)	dim=80

Nome	Descrizione	Tipo	Vincoli
i	contatore	intero	>=0
temp_nome_oggetto	Nome fittizio dato all'oggetto non valido	array 1 dimensione caratteri (statico)	dim=1

```
Temp_nome_oggetto := "n"
i:=0
MENTRE(i<NUM_OGGETTI_INVENTARIO_MAX)
    scrivere_id_oggetto(inventario[i], -1)
    scrivere_id_oggetto_genitore(inventario[i], 1)
    scrivere_nome_oggetto (inventario[i], temp_nome_oggetto)
    scrivere_visibilita_oggetto (inventario[i], falso)
    i:=i+1
FINE
```

Funzione eseguire_parser

Nome	Descrizione	Tipo	Vincoli
comando_utente	Comando immesso dall'utente	array 1 dimensione di caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
regex_match	flag che indica se il comando immesso dall'utente è corretto sintatticamente	booleano	vero=il comando supera i controlli sintattici falso= il comando non supera i controlli sintattici

Nome	Descrizione	Tipo	Vincoli
temp_id	Id del comando che bisogna eseguire	intero	>=0, 0=guarda, 1=sali, 2=scendi, 3=guarda, 4=ispeziona, 5=prendi, 6=apri, 7=usa, 8=mangia, 9=equipaggia, 10=leggi, 11=uccidi, 12=salva, 13=carica , 14=salta
lista_comandi	Lista di comandi, per ciascuno dei quali sono memorizzate informazioni relative a quale deve essere il verbo, l'argomento ecc…	array 1 dimensione comandi (statico)	dimensione=15
mappa_gioco	Mappa del gioco	mappa	
inventario	oggetti presi dal protagonista	array 1 dimensione oggetti (statico)	dimensione=80
oggetto_interazione	Oggetto della stanza su cui eseguire l'azione	oggetto	
protagonista	Protagonista del gioco	oggetto_personaggio	

```
//1- verifica esistenza comando
regex_match := falso
i:=0
MENTRE(i<15 AND regex_match = falso)
    temp_id := leggere_id_comando(lista_comandi[i])
    regex_match := applicare_regex(verbo_comando, leggere_regex_verbo(lista_comandi[i]))
    comando_trovato := regex_match
    i:=i+1
FINE
SE(comando_trovato = falso)
    ALLORA
    stampa("Non capisco cosa vuoi fare")
FINE

//2- Controllo se l'argomento del comando immesso dall'utente è "inventario"
SE(confronta_stringhe(argomento_comando, "inventario") = 0)
    ALLORA
    argomento_inventario := vero
    ALTRIMENTI
    argomento_inventario := falso
FINE

//3- Verifica se l'argomento del comando deve essere un oggetto o meno
SE(leggere_argomento_oggetto(lista_comandi[temp_id]) = falso)
     ALLORA
     // In questo caso l'argomento del comando NON e' un oggetto ma una locuzione.
```

```
 // Controllo che la locuzione sia valida, cioe' che l'espressione regolare che descrive
 // l'argomento di tale comando sia rispettata e che quindi ci sia un match.
 // Se il match non avviene, il comando non e' valido
 SE(argomento_comando != “”)
    ALLORA
    regex_match := applicare_regex(argomento_comando,
                                   leggere_regex_argomento(lista_comandi[temp_id]))
    argomento_corretto := regex_match
    ALTRIMENTI

    //Se il comando e' del tipo "guarda"/"salva"/"carica" allora e' comunque corretto anche
      se non ha argomento
    SE( (temp_id = 3 OR
         temp_id = 12 OR
         temp_id = 13 OR))
        ALLORA
        argomento_corretto := vero
        ALTRIMENTI
        argomento_corretto := falso
    FINE

    SE(argomento_corretto = falso)
        ALLORA
        SE(argomento_comando = “”)
            ALLORA
            stampa("Specifica un oggetto")
            ALTRIMENTI
            stampa("Non so a quale oggetto ti stai riferendo")
        FINE
        ALTRIMENTI
           SE(temp_id = 0)
              ALLORA
              eseguire_vai(mappa_gioco, argomento_comando, inventario)
           ALTRIMENTI SE(temp_id = 3)
              ALLORA
              eseguire_guarda(mappa_gioco, riga_stanza_corrente, col_stanza_corrente)
           ALTRIMENTI SE(temp_id = 12)
              Caricare_partita(protagonista, inventario, mappa_gioco)
           ALTRIMENTI SE(temp_id = 13)
              Salvare_partita(protagonista, inventario, mappa_gioco)
           ALTRIMENTI SE(temp_id = 14)
              eseguire_salta(mappa_gioco)
      FINE
  FINE
ALTRIMENTI
//L’argomento del comando è un oggetto
    ricerca_in_stanza := leggere_argomento_oggetto_in_stanza(lista_comandi[temp_id])
    ricerca_in_inventario:= leggere_argomento_oggetto_in_inventario(lista_comandi[temp_id])

    SE(ricerca_in_stanza = vero)
       ALLORA
       oggetto_interazione := null
       oggetto_interazione:=ricercare_nome_oggetto_in_stanza(stanza_corrente,
                                                             argomento_comando)
    FINE

    SE(oggetto_interazione = null AND ricerca_in_inventario = vero)
       ALLORA
```

```
          oggetto_interazione := ricercare_nome_oggetto_in_inventario(inventario,
                                                                      argomento_comando)
        FINE
        SE(oggetto_interazione = null)
            ALLORA
            stampa("Non vedo nessun ", argomento_comando, “qui”)

            ALTRIMENTI
            Attributo_da_ricercare := leggere_regex_argomento(lista_comandi[temp_id])[0]
            pos_attributo_da_ricercare :=
ricercare_attributo(leggere_attributi_oggetto(oggetto_interazione), attributo_da_ricercare)
            SE(pos_attributo_da_ricercare = -1 OR argomento_inventario = vero)
               ALLORA
               stampa(“non posso fare questa azione”)
               ALTRIMENTI

                   SE     (temp_id = 1)
                          ALLORA
                          eseguire_sali(oggetto_interazione)
                   ALTRIMENTI SE(temp_id = 2)
                          ALLORA
                          eseguire_scendi(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 4)
                          ALLORA
                          eseguire_ispeziona(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 5)
                          ALLORA
                          eseguire_prendi(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 6)
                          ALLORA
                          Eseguire_apri(mappa, inventario, oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 7)
                          ALLORA
                          eseguire_usa(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 8)
                          ALLORA
                          eseguire_mangia(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 9)
                          ALLORA
                          eseguire_equipaggia(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 10)
                          ALLORA
                          eseguire_leggi(oggetto_interazione)
                   ALTRIMENTI SE (temp_id = 11)
                          ALLORA
                          Eseguire_uccidi(oggetto_interazione,protagonista)
            FINE

        FINE
    FINE
```

Funzione ricercare_nome_oggetto_in_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_corrente	Stanza dove si trova il protagonista attualmente	stanza	
nome_oggetto	Nome oggetto da ricercare nella stanza (e ovviamente anche negli oggetti contenuti nella stanza)	array 1 dimensione caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
oggetto_trovato	Oggetto che ha lo stesso nome del dato in input alla funzione	oggetto	

Nome	Descrizione	Tipo	Vincoli
i	contatore	intero	>=0
fine_ricerca	flag che indica se interrompere la ricerca	booleano	vero=oggetto trovato,falso altrimenti
temp_nome oggetto	Nome dell'oggetto corrente della stanza	array 1 dimensione caratteri (statico)	dimensione=80
oggetto_corrente	Oggetto corrente della stanza	oggetto	

```
fine_ricerca := falso
i :=0
MENTRE(i<10 AND fine_ricerca = falso)
    oggetto_corrente := leggere_oggetto_stanza(stanza_corrente, i)
    SE(leggere_id_oggetto(oggetto_corrente)!= -1 AND
       leggere_visibilita_oggetto(oggetto_corrente) = vero)
       ALLORA
       temp_nome_oggetto := leggere_nome_oggetto(oggetto_corrente)
       togliere_maiuscole_da_stringa(temp_nome_oggetto)
       SE(confronta(temp_nome_oggetto, nome_oggetto) = 0)
          ALLORA
          oggetto_trovato := oggetto_corrente
          fine_ricerca := vero
       FINE
    FINE
    i:=i+1
FINE
SE(fine_ricerca = falso)
    ALLORA
    i=0
    MENTRE(i<5 AND fine_ricerca = falso)
    oggetto_corrente := leggere_oggetto_contenuto(stanza_corrente, i)
    SE(leggere_id_oggetto(oggetto_corrente)!= -1 AND
       leggere_visibilita_oggetto(oggetto_corrente) = vero)
       ALLORA
       temp_nome_oggetto := leggere_nome_oggetto(oggetto_corrente)
       togliere_maiuscole_da_stringa(temp_nome_oggetto)
       SE(confronta(temp_nome_oggetto, nome_oggetto) = 0)
          ALLORA
          oggetto_trovato := oggetto_corrente
          fine_ricerca := vero
       FINE
    FINE
    i:=i+1
FINE
```

Funzione ricercare_nome_oggetto_in_inventario

Nome	Descrizione	Tipo	Vincoli
inventario	Array di oggetti che il protagonista ha preso	array 1 dimensione oggetti (statico)	
nome_oggetto	Nome oggetto da ricercare nell'inventario	array 1 dimensione caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
oggetto_trovato	Oggetto che ha lo stesso nome del dato in input alla funzione	oggetto	

Nome	Descrizione	Tipo	Vincoli
i	contatore	intero	>=0
fine_ricerca	flag che indica se interrompere la ricerca	booleano	vero=oggetto trovato,falso altrimenti
temp_nome oggetto	Nome dell'oggetto corrente dell'inventario	array 1 dimensione caratteri (statico)	dimensione=80
oggetto_corrente	oggetto corrente dell'inventario	oggetto	

```
fine_ricerca := falso
i :=0
MENTRE(i<80 AND fine_ricerca = falso)
    oggetto_corrente := inventario[i]
    SE(leggere_id_oggetto(oggetto_corrente)!= -1)
       ALLORA
       temp_nome_oggetto := leggere_nome_oggetto(oggetto_corrente)
       togliere_maiuscole_da_stringa(temp_nome_oggetto)
       SE(confronta(temp_nome_oggetto, nome_oggetto) = 0)
          ALLORA
          oggetto_trovato := oggetto_corrente
          fine_ricerca := vero
       FINE
    FINE
    i:=i+1
FINE
```

Eseguire_vai

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa che comprende la lunghezza e la larghezza, utili per calcolare la stanza di arrivo del protagonista	mappa	
direzione	direzione in cui il protagonista vorrebbe recarsi	array 1 dimensione caratteri (statico)	valori accettati: nord/est/sud/ovest/porta
inventario	inventario in cui sono presenti eventuali chiavi utili per spostarsi da una stanza all'altra, nel caso ci siano porte	array 1 dimensione oggetti (statico)	dimensione=80
id_stanza_ arrivo_ protagonista	ID della stanza dove il protagonista si sposterebbe	intero	>0
riga_stanza_ corrente	Riga della stanza dove si trova il protagonista attualmente	intero	>=0
col_stanza_ corrente	Colonna della stanza dove si trova il protagonista attualmente	intero	>=0

Nome	Descrizione	Tipo	Vincoli
id_stanza_arrivo protagonista	ID della stanza in cui il protagonista si trova dopo essersi spostato	intero	

Nome	Descrizione	Tipo	Vincoli
esito_ricerca_porta_ in_stanza	Numero che indica se esiste una porta che, se varcata, porta il protagonista nella stanza in cui vorrebbe andare tramite il comando vai	intero	>=0 se una porta esiste -1 se una porta non esiste -2 se la porta esiste ma il protagonista non ha le chiavi per varcarla
temp_id_nord	Id della stanza collegata a nord con la stanza corrente	intero	=-1 se la stanza non è collegata a nord
temp_id_est	Id della stanza collegata a est con la stanza corrente	intero	=-1 se la stanza non è collegata a est
temp_id_sud	Id della stanza collegata a sud con la stanza corrente	intero	=-1 se la stanza non è collegata a sud
temp_id_ovest	Id della stanza collegata a ovest con la stanza corrente	intero	=-1 se la stanza non è collegata a ovest

```
//Calcola l'id della posizione dove il protagonista si sposterebbe

SE(confronta(direzione, “porta”)=0)
      ALLORA
      stanza_corrente := leggere_stanza_da_mappa(mappa_gioco, riga_stanza_corrente,
col_stanza_corrente)
      temp_id_nord  := leggere_collegamento_id_stanza(stanza_corrente, 0)
      temp_id_est   := leggere_collegamento_id_stanza(stanza_corrente, 1)
      temp_id_sud   := leggere_collegamento_id_stanza(stanza_corrente, 2)
      temp_id_ovest := leggere_collegamento_id_stanza(stanza_corrente, 3)

      SE(id_stanza_arrivo_protagonista = temp_id_nord)
           ALLORA
            eseguire_vai(mappa_gioco, "nord", inventario, id_stanza_arrivo_protagonista)
      ALTRIMENTI SE(id_stanza_arrivo_protagonista = temp_id_est)
           ALLORA
            eseguire_vai(mappa_gioco, "est", inventario, id_stanza_arrivo_protagonista)
      ALTRIMENTI SE(id_stanza_arrivo_protagonista = temp_id_sud)
           ALLORA
            eseguire_vai(mappa_gioco, "sud", inventario, id_stanza_arrivo_protagonista)
      ALTRIMENTI SE(id_stanza_arrivo_protagonista = temp_id_ovest)
           ALLORA
            eseguire_vai(mappa_gioco, "ovest", inventario, id_stanza_arrivo_protagonista)
      FINE


//Controlla che la stanza è collegata
SE(ricercare_collegamento_stanza(mappa_gioco, id_stanza_arrivo_protagonista, direzione) >= 0)
       ALLORA
       esito_ricerca_porta_in_stanza := ricercare_porta_in_stanza(mappa_gioco,
id_stanza_corrente_protagonista, id_stanza_arrivo_protagonista, inventario)

       //Controlla se esiste una porta
       SE(esito_ricerca_porta_in_stanza >= 0)
             ALLORA
             aggiornare_id_stanza_corrente(mappa_gioco, id_stanza_arrivo_protagonista)
       ALTRIMENTI SE(esito_ricerca_porta_in_stanza = -1)
             ALLORA
             aggiornare_id_stanza_corrente(mappa_gioco, id_stanza_arrivo_protagonista)
       ALTRIMENTI SE(esito_ricerca_porta_in_stanza = -2)
             ALLORA
             stampa("non hai la chiave per varcare la porta")
       FINE


       ALTRIMENTI
       stampa("Non puoi andare a "+direzione)
FINE
```

Funzione ricercare_collegamento_stanza

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui poter accedere alla stanza di cui si vuol fare il controllo	mappa	
id_stanza_arrivo protagonista	ID della stanza in cui il protagonista si trova dopo essersi spostato	intero	
direzione	direzione in cui il protagonista vorrebbe recarsi	array 1 dimensione caratteri (statico)	valori accettati: nord/est/sud/ovest

Nome	Descrizione	Tipo	Vincoli
pos_collegamento ricercato	Indice della posizione di id_stanza_arrivo_protagonista nell'array id_stanze_collegate relativo a stanza_corrente	intero	>=0 se id_stanza_arrivo è collegata alla stanza corrente, -1 altrimenti

Nome	Descrizione	Tipo	Vincoli
stanza_corrente	Stanza dove si trova il protagonista al momento dell'esecuzione del comando vai	stanza	
riga_stanza_corrente	Riga nella mappa della stanza corrente	intero	>=0
col_stanza_corrente	Colonna nella mappa della stanza corrente	intero	>=0

```
stanza_corrente := leggere_stanza_da_mappa(mappa_gioco, riga_stanza_corrente,
col_stanza_corrente)
pos_collegamento_ricercato := -1
SE(confronta(direzione,"nord") =0)
       ALLORA
       pos_Stanza :=0
ALTRIMENTI SE(confronta(direzione,"est") =0)
       ALLORA
       pos_stanza :=1
ALTRIMENTI SE(confronta(direzione,"sud") =0)
       ALLORA
       pos_stanza :=2
ALTRIMENTI SE(confronta(direzione,"ovest") =0)
       ALLORA
       pos_stanza :=3
FINE

SE(id_stanza_arrivo_protagonista = leggere_collegamento_id_stanza(stanza_corrente, pos_stanza))
       ALLORA
       pos_collegamento_ricercato := pos_stanza

       ALTRIMENTI
      pos_collegamento_ricercato := -1
FINE
```

Funzione ricercare_porta_in_stanza

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa da cui leggere la stanza corrente del protagonista. In questa stanza ci saranno tutti gli oggetti, tra cui anche le porte	mappa	
id_stanza_corrente protagonista	ID della stanza in cui il protagonista si trova	intero	>0
id_stanza_arrivo protagonista	ID della stanza in cui il protagonista si trova dopo essersi spostato	intero	>0
inventario	Inventario del protagonista dove sono presenti eventuali chiavi per varcare la porta	array 1 dimensione oggetti (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
pos_porta_ricercata	Indice della posizione della porta che, se varcata, porta il protagonista nella stanza desiderata	intero	>=0 se una porta esiste -1 se una porta non esiste -2 se la porta esiste ma il protagonista non ha le chiavi per varcarla

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	Oggetto della stanza che si deve controllare per vedere se è una porta o no	oggetto	
porta_corrente	Se oggetto_corrente è una porta (controllo fatto mediante leggere_tipo_oggetto(oggetto_corrente), questo viene convertito in un oggetto_porta	oggetto_porta	
chiavi_trovate	Numero che indica se in inventario sono presenti tutti gli oggetti di leggere_id_chiavi(porta_corrente)	intero	>=0 : le chiavi sono state trovate -1 : le chiavi non sono state trovate

```
stanza_corrente := leggere_stanza_da_mappa(mappa_gioco, riga_stanza_corrente,
                   col_stanza_corrente)
pos_porta_ricercata := -1
i :=0
MENTRE(i<NUM_OGGETTI_STANZA_MAX)
    oggetto_corrente := leggere_oggetto_stanza(stanza_corrente, i)
    SE(leggere_visibilita_oggetto(oggetto_corrente) = VERO)
             ALLORA
             SE(leggere_tipo_oggetto(oggetto_corrente) = 0)
                    ALLORA
                    porta_corrente := ottenere_oggetto_specifico_da_oggetto(mappa_gioco,
                                      oggetto_corrente, 0)

                     SE(leggere_id_stanza_partenza_porta(porta_corrente) =
                         id_stanza_corrente_protagonista AND
                         leggere_id_stanza_arrivo_porta(porta_corrente) =
                         id_stanza_arrivo_protagonista)
                          ALLORA

                          //Esiste una porta per la direzione del protagonista
                          pos_porta_ricercata := i

                          //Controllo che il protagonista abbia tutte gli oggetti richiesti
                            per varcare la porta
                          chiavi_trovate := ricercare_id_in_inventario(inventario,
                          leggere_id_chiavi_porta(porta_corrente), NUM_CHIAVI_PORTA_MAX)
                          SE(chiavi_trovate = -1)
                              ALLORA
                              pos_porta_ricercata = -2
                          FINE
                      FINE
             FINE
    FINE
    i:=i+1
FINE
```

Eseguire_sali

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	oggetto su cui il protagonista vorrebbe salire	oggetto	deve avere attributo ‘S’

Nome	Descrizione	Tipo	Vincoli
id_stanza_corrente	ID della stanza dove si trova il protagonista	intero	>0

Nome	Descrizione	Tipo	Vincoli
pos	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	>=0 se trovato, -1 se non trovato

```
pos := ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), 'S')
SE(pos = -1)
    ALLORA
    stampare("non puoi salire su "+leggere_nome_oggetto(oggetto_corrente))

    ALTRIMENTI
    id_stanza_corrente := leggere_id_stanza_arrivo(oggetto_corrente)
FINE
```

Eseguire_scendi

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	oggetto da cui il protagonista vorrebbe scendere	oggetto	deve avere attributo ‘s’

Nome	Descrizione	Tipo	Vincoli
id_stanza_corrente	ID della stanza dove si trova il protagonista dopo aver fatto l’azione	intero	>0

Nome	Descrizione	Tipo	Vincoli
pos	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	>=0 se trovato, -1 se non trovato

```
pos := ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), 's')
SE(pos = -1)
    ALLORA
    stampare("non puoi scendere da "+leggere_nome_oggetto(oggetto_corrente))

    ALTRIMENTI
    id_stanza_corrente := leggere_id_stanza_arrivo(oggetto_corrente)
FINE
```

Eseguire_guarda

Nome	Descrizione	Tipo	Vincoli
mappa	Mappa da cui estrapolare la stanza in cui si trova il protagonista (e quindi tutti gli oggetti della stanza)	mappa	
val_riga	Riga della stanza dove si trova attualmente il protagonista	intero	>=0
val_col	Colonna della stanza dove si trova attualmente il protagonista	intero	>=0

Nome	Descrizione	Tipo	Vincoli
nome_oggetto_visibile	Nome dell'oggetto con cui il protagonista interagisce	array 1 dimensione caratteri (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
i	Contatore	intero	>=0

```
i:=0
MENTRE(i<NUM_OGGETTI_STANZA_MAX)
    oggetto_corrente := leggere_oggetto_stanza(stanza_corrente, i)
    SE(leggere_visibilita_oggetto(oggetto_corrente) = vero)
        ALLORA
        nome_oggetto_visibile := leggere_nome_oggetto(oggetto_corrente)
        Stampa(nome_oggetto_visibile)
    FINE
    i:=i+1
FINE
//Stampa gli oggetti contenuti in cassetti
i:=0
MENTRE(i<NUM_OGGETTI_CONTENUTI_PER_STANZA_MAX)
    oggetto_corrente := leggere_oggetto_contenuto(stanza_corrente, i)
    SE(leggere_visibilita_oggetto(oggetto_corrente) = vero)
        ALLORA
        nome_oggetto_visibile := leggere_nome_oggetto(oggetto_corrente)
        Stampa(nome_oggetto_visibile)
    FINE
    i:=i+1
FINE
```

Eseguire_ispeziona

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	Oggetto che il protagonista vuole ispezionare	oggetto	deve avere tra gli attributi 'i'

Nome	Descrizione	Tipo	Vincoli
descrizione_ oggetto	Descrizione dell'oggetto che il protagonista ha ispezionato	array 1 dimensione caratteri (statico)	dimensione=200

Nome	Descrizione	Tipo	Vincoli
id_ultimo_oggetto_ ispezionato	Id dell'ultimo oggetto che il protagonista ha ispezionato. Tale valore verrà poi usato dalla funzione eseguire_usa	intero	>0

```
descrizione_oggetto := leggere_descrizione_oggetto(oggetto_corrente)
Stampa(descrizione_oggetto)
id_ultimo_oggetto_ispezionato := leggere_id_oggetto(oggetto_corrente)
```

Eseguire_prendi

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	Oggetto che il protagonista vuole prendere	oggetto	deve avere attributo ‘p’
NUM_OGGETTI_INVENTARIO_MAX	Numero massimo di oggetti presenti nell’inventario	intero	=80

Nome	Descrizione	Tipo	Vincoli
inventario	Inventario aggiornato del protagonista. E’ stato aggiunto oggetto_corrente all’inventario, ed è stato tolto oggetto_corrente dagli oggetti della stanza	array 1 dimensione di oggetti (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
pos	valore restituito dalla funzione ricercare_attributo	intero	>=0
fine_ricerca	Flag che indica di terminare la ricerca per uno spazio nell’inventario	booleano	vero=terminare la ricerca falso=proseguire la ricerca

```
pos:= ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), ‘p’)
SE(pos = -1)

   ALLORA
   //cioè se ‘p’ non compare tra gli attributi di oggetto_corrente
   stampa(“questo oggetto non è prendibile”)

   ALTRIMENTI

   fine_ricerca = falso
   i:=0
   MENTRE(i<NUM_OGGETTI_INVENTARIO_MAX AND fine_ricerca == false)
        oggetto_corrente_inventario = inventario[i]
        // Appena trovo un oggetto non valido nell'inventario, lo sostituisco con l'oggetto
        // preso
        SE(leggere_id_oggetto(oggetto_corrente_inventario) == -1 OR
leggere_visibilita_oggetto(oggetto_corrente_inventario) == falso )
            ALLORA
            oggetto_corrente_inventario := oggetto_corrente
            scrivere_visibilita_oggetto(oggetto_corrente_inventario, true)
            fine_ricerca := true
        FINE
        i:=i+1
    FINE
    SE(fine_ricerca = vero)
        ALLORA
        //L'oggetto nella stanza non esiste piu', e' stato spostato nell'inventario
        Scrivere_id_oggetto(oggetto_corrente,-1)
    FINE
FINE
```

Eseguire_apri

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui e' presente la stanza corrente del protagonista.	mappa	
inventario	array di elementi presi dal protagonista nel corso del gioco. In questa funzione, l'inventario serve perche' diventerà l'input della funzione eseguire_vai	array 1 dimensione oggetti (statico)	dimensione=80
oggetto_corrente	Oggetto che il protagonista vuole aprire	oggetto	deve avere attributi 'a' oppure 'A'

Nome	Descrizione	Tipo	Vincoli
oggetto_contenuto_corrente	Oggetto che si trova nell'oggetto che il protagonista ha aperto	oggetto	

Nome	Descrizione	Tipo	Vincoli
porta_corrente	Porta che il protagonista vuole aprire. Mediante l'accesso al metodo leggere_id_stanza_arrivo, leggiamo dove la porta reindirizza il protagonista ed eseguiamo il comando vai. Il comando "apri" si "trasforma" quindi in comando "vai"	oggetto_porta	
pos_attributo_ricercato	valore restituito dalla funzione ricercare_attributo	intero	>=0, -1 se non trovato
stanza_corrente	Stanza dove si trova al momento il protagonista. Serve per poter accedere agli oggetti contenuti	stanza	inter

```
pos_attributo_ricercato :=ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), ‘a’)
SE(pos_attributo_ricercato = -1)
    ALLORA
       pos_attributo_ricercato=ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente),
                                                                                    ‘A’)

        SE(pos_attributo_ricercato = -1)
            ALLORA
            stampa("Non posso aprire leggere_nome_oggetto(oggetto_corrente))

            ALTRIMENTI

        porta_corrente := ottenere_oggetto_specifico_da_oggetto(mappa_gioco,oggetto_corrente,0)
        id_stanza_arrivo_porta := leggere_id_stanza_arrivo_porta(porta_corrente)

        //convertiamo la destinazione della porta in una direzione
        SE(id_stanza_arrivo_porta = id_stanza_corrente_protagonista - leggere_col(mappa_gioco))
            ALLORA
            eseguire_vai(mappa_gioco, "nord", inventario)
        ALTRIMENTI SE(id_stanza_arrivo_porta = id_stanza_corrente_protagonista + 1)
            eseguire_vai(mappa_gioco, "est", inventario)
        ALTRIMENTI SE(id_stanza_arrivo_porta =
                     id_stanza_corrente_protagonista+leggere_col(mappa_gioco))
            eseguire_vai(mappa_gioco, "sud", inventario)
        ALTRIMENTI SE(id_stanza_arrivo_porta = id_stanza_corrente_protagonista - 1)
            eseguire_vai(mappa_gioco, "ovest", inventario)
        FINE

    ALTRIMENTI

        stanza_corrente := leggere_stanza_da_mappa(mappa_gioco, riga_stanza_corrente,
                                                              col_stanza_corrente)
        i:=0
        MENTRE(i<NUM_OGGETTI_CONTENUTI_PER_STANZA_MAX)
            //mostriamo tutti gli oggetti contenuti nell’oggetto. Se ad esempio l’oggetto che
              il protagonista vuole aprire e’ “cassetto” e ha come ID 94, gli oggetti che
            dovremmo mostrare saranno quelli che hanno come ID 940, 941, 942, 943,. . . 949

            oggetto_contenuto_corrente := leggere_oggetto_contenuto(stanza_corrente, i)

            SE( (leggere_id_oggetto(oggetto_contenuto_corrente)/10) =
                 leggere_id_oggetto(oggetto_corrente))
                ALLORA
                scrivere_visibilita_oggetto(oggetto_contenuto_corrente, vero)
            FINE
            i:=i+1
         FINE
 FINE
```

Eseguire_usa

Nome	Descrizione	Tipo	Vincoli
id_ultimo_oggetto_ispezionato	Id dell'ultimo oggetto ispezionato	intero	=-1 se il protagonista non ha ispezionato nessun oggetto nella stanza
oggetto_corrente	Oggetto che il protagonista vuole usare	oggetto	deve avere attributi 'u'

Nome	Descrizione	Tipo	Vincoli
messaggio	Messaggio che l'utente legge dopo aver usato l'oggetto	array 1 dimensione caratteri (statico)	dimensione=64

Nome	Descrizione	Tipo	Vincoli
riga_stanza_corrente	Riga della stanza corrente del protagonista	intero	>=0
col_stanza_corrente	Colonna della stanza corrente del protagonista	intero	>=0

```
SE(id_ultimo_oggetto_ispezionato = 132 AND id_oggetto_utilizzato = 1203)
    ALLORA
        // (vincolo 13)
       messaggio := "hai usato la fune per creare un passaggio"
scrivere_visibilita_oggetto(leggere_oggetto_stanza(leggere_stanza_da_mappa(mappa_gioco,riga_sta
nza_corrente, col_stanza_corrente),3), vero)

        rendere_oggetto_non_valido(oggetto_corrente)

ALTRIMENTI SE(id_ultimo_oggetto_ispezionato = 62 AND id_oggetto_utilizzato = 21 )
    ALLORA
       // (vincolo 2)
       messaggio := "hai creato una torcia"
      scrivere_id_oggetto(oggetto_corrente, 921)

ALTRIMENTI SE(id_ultimo_oggetto_ispezionato = 106)
    ALLORA
        // (vincolo 10a/10b)
        SE(id_oggetto_utilizzato != 840)
           ALLORA
             messaggio := "hai perso una vita usando quest'oggetto"
             scrivere_salute_personaggio(protagonista,
leggere_salute_personaggio(protagonista)*0, vero)

           ALTRIMENTI
           messaggio := "e' comparso un portale"
scrivere_visibilita_oggetto(leggere_oggetto_stanza(leggere_stanza_da_mappa(mappa_gioco,riga_sta
nza_corrente, col_stanza_corrente),7), vero)
           rendere_oggetto_non_valido(oggetto_corrente)

ALTRIMENTI SE(id_ultimo_oggetto_ispezionato = -1)
    ALLORA

        Messaggio := "devi ispezionare qualcosa prima"

ALTRIMENTI

        Messaggio := "Non succede niente"
FINE
stampa(messaggio)
```

Eseguire_salta

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
id_stanza_ arrivo	Stanza dove si trova il protagonista dopo aver saltato	intero	=120 (stanza scuderia)

```
id_stanza_arrivo := 120
SE(id_stanza_corrente_protagonista = 170)
   ALLORA
   aggiornare_id_stanza_corrente(mappa_gioco, id_stanza_arrivo)
   ALTRIMENTI
   stampa("non puoi saltare qui”)
FINE
```

Eseguire_mangia

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	oggetto che il protagonista vorrebbe mangiare	oggetto	deve avere attributo ‘m’

Nome	Descrizione	Tipo	Vincoli
protagonista	Protagonista con punti salute incrementati pari al bonus salute del cibo mangiato	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
pos	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	>=0 se trovato, -1 se non trovato

```
pos := ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), 'm')
SE(pos = -1)
    ALLORA
    stampare("non puoi mangiare "+leggere_nome_oggetto(oggetto_corrente))

    ALTRIMENTI
    scrivere_salute_personaggio(protagonista, leggere_salute_personaggio(protagonista) +
                                              leggere_bonus_salute_cibo(oggetto_corrente))

FINE
```

Eseguire_equipaggia

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	oggetto che il protagonista vorrebbe mangiare	oggetto	deve avere attributo ‘m’

Nome	Descrizione	Tipo	Vincoli
protagonista	Personaggio del gioco in cui il numero di punti salute e forza sono incrementati dopo aver equipaggiato l’arma	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
pos	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	>=0 se trovato, -1 se non trovato

```
pos := ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), 'e')
SE(pos = -1)
    ALLORA
    stampare("non puoi equipaggiare "+leggere_nome_oggetto(oggetto_corrente))

    ALTRIMENTI
    arma_da_equipaggiare := ottenere_oggetto_specifico_da_oggetto(mappa_gioco,
                                                    oggetto_corrente, 2)
    SE(leggere_stato_equipaggiato_arma(arma_da_equipaggiare) = vero)
        ALLORA
        stampare(“l’arma e' gia' equipaggiata”)

        ALTRIMENTI

        //disequipaggiare l’arma che è già euqipaggiata
        arma_gia_equipaggiata := ricercare_arma_equipaggiata(mappa_gioco, inventario)
        SE(arma_gia_equipaggiata != NULL)
            ALLORA
            // Se esiste un'arma gia' equipaggiata
            scrivere_salute_personaggio(protagonista, leggere_salute_personaggio(protagonista)
                                        - leggere_bonus_salute_arma(arma_gia_equipaggiata))
            scrivere_forza_personaggio(protagonista, leggere_forza_personaggio(protagonista) -
                                        leggere_bonus_forza_arma(arma_gia_equipaggiata))
            scrivere_stato_equipaggiato_arma(arma_gia_equipaggiata, falso)
        FINE

        scrivere_salute_personaggio(protagonista, leggere_salute_personaggio(protagonista) +
                                        leggere_bonus_salute_arma(arma_da_equipaggiare))
        scrivere_forza_personaggio(protagonista, leggere_forza_personaggio(protagonista) +
                                        leggere_bonus_forza_arma(arma_da_equipaggiare))
        scrivere_stato_equipaggiato_arma(arma_da_equipaggiare, vero)
    FINE
FINE
```

Funzione ricercare_arma_equipaggiata

Nome	Descrizione	Tipo	Vincoli
inventario	Array di oggetti presi dal protagonista nel corso del gioco	array 1 dimensione di oggetti (statico)	dimensione=80

Nome	Descrizione	Tipo	Vincoli
arma_corrente_ inventario	Arma già equipaggiata dal protagonista al momento della chiamata della funzione eseguire_equipaggia	oggetto_arma	NULL se non esiste un'arma dell'inventario già equipaggiata.

Nome	Descrizione	Tipo	Vincoli
fine_ricerca	flag che indica se l'elemento è stato trovato, dunque si deve interrompere la ricerca	booleano	
i	contatore	intero	>=0

```
i:=0
MENTRE(i< 80 AND fine_ricerca = falso)
    oggetto_corrente_inventario = inventario[i]
    SE(leggere_tipo_oggetto(oggetto_corrente_inventario) = ID_TIPO_OGGETTO_ARMA AND
       leggere_visibilita_oggetto(oggetto_corrente_inventario) = vero )
        ALLORA
        arma_corrente_inventario = ottenere_oggetto_specifico_da_oggetto(mappa_gioco,
                                   oggetto_corrente_inventario, 2)

        SE(leggere_stato_equipaggiato_arma(arma_corrente_inventario) = vero)
             ALLORA
             fine_ricerca := vero
        FINE
     FINE
     i:=i+1
 FINE
 SE(fine_ricerca = falso)
     ALLORA
     arma_corrente_inventario := NULL
FINE
```

Eseguire_leggi

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	oggetto che il protagonista vorrebbe leggere	oggetto	deve avere attributo ‘l’

Nome	Descrizione	Tipo	Vincoli
messaggio	Testo letto dal protagonista	array 1 dimensione caratteri (statico)	dimensione=200

Nome	Descrizione	Tipo	Vincoli
pos	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	>=0 se trovato, -1 se non trovato
path_lista_messaggi	Percorso del file dove sono memorizzati tutti i messaggi	array 1 dimensione caratteri (statico)	
file_lista_messaggi	File dove sono memorizzati tutti i messaggi	file di testo	

```
pos := ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), 'l')
SE(pos = -1)
    ALLORA
    stampare("non c’e’ scritto nulla su "+leggere_nome_oggetto(oggetto_corrente))

    ALTRIMENTI

    SE(esiste(PATH_LISTA_MESSAGGI) = falso)
    ALLORA
        stampare(“errore”)

    ALTRIMENTI
    file_lista_messaggi := aprire_file(PATH_LISTA_MESSAGGI)
    fine_ricerca := falso
    MENTRE(file_non_termina(file_lista_messaggi) AND fine_ricerca = falso)

        //Preleva l'id dell'oggetto su cui c'e' scritto qualcosa
        Temp_id := leggere_intero_da_file(file_lista_messaggi)
        //Preleva il messaggio dell'oggetto fino al carattere di fine campo (&)
        c := leggere_carattere_da_file(file_lista_messaggi)
        i:=0
        MENTRE(c != '&' AND c!= EOF)
            c := leggere_carattere_da_file(file_lista_messaggi)
            SE(c != '&' AND c!=EOF)
                ALLORA
                temp_messaggio[i]:=c
                i:=i+1
            FINE
        FINE
        SE(temp_id = leggere_id_oggetto(oggetto_corrente))
            ALLORA
            Stampare(temp_messaggio)
            fine_ricerca := vero
        FINE
    FINE
    Chiudere_file(txt_file_lista_messaggi)
```

Eseguire_uccidi

Nome	Descrizione	Tipo	Vincoli
oggetto_corrente	Oggetto della stanza che il protagonista vuole uccidere	oggetto	deve avere tra gli attributi ‘w’
stanza_corrente	Stanza corrente dove si trovano il protagonista e il nemico	stanza	
protagonista	Protagonista del gioco che combatte con il nemico	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
protagonista	Protagonista con i punti salute variati dopo lo scontro con il nemico	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
pos	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	>=0 se trovato, -1 se non trovato
danno_protagonista	Danno che il protagonista riceve dopo essersi scontrato con il nemico	intero	>=0
nemico	Nemico del protagonista	oggetto_personaggio	

```
pos := ricercare_attributo(leggere_attributi_oggetto(oggetto_corrente), 'w')
SE(pos = -1)
    ALLORA
    stampare(“non puoi uccidere ”, leggere_nome_oggetto(oggetto_corrente)

    ALTRIMENTI
    nemico := ottenere_oggetto_specifico_da_oggetto(mappa_gioco,
                                      oggetto_corrente, 3)
    danno := avviare_sistema_combattimento(protagonista, nemico)
    scrivere_salute_personaggio(protagonista, leggere_salute_personaggio(protagonista)-danno)
    SE(leggere_salute(protagonista) >0)
        ALLORA
        i:=0
        MENTRE(i<5)
            //rendi visibile ciascuno degli oggetti rilasciati dal nemico
            j:=0
            MENTRE(j<10)
                 SE(leggere_oggetto_stanza(stanza_corrente)=
                    leggere_oggetto_rilasciato(nemico,i))
                     ALLORA
                     scrivere_visibilita_oggetto(leggere_oggetto_stanza(stanza_corrente),vero)
                     j:=10
                 FINE
                 j:=j+1
            FINE
            i:=i+1
        FINE
 FINE
```

Avviare_sistema_combattimento

Nome	Descrizione	Tipo	Vincoli
protagonista	Primo personaggio del combattimento	oggetto_personaggio	
nemico	Secondo personaggio del combattimento	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
danno	Numero di punti salute che il protagonista perde affrontando il nemico	intero	>=0. Se danno>=salute del protagonista, il protagonista perde una vita

Nome	Descrizione	Tipo	Vincoli
dado	Indice della posizione dell'attributo ricercato nella lista degli attributi dell'oggetto	intero	compreso tra 1 e 15
salute_protagonista/nemico	Salute del protagonista/nemico	intero	>=0
forza_protagonista/nemico	Forza del protagonista/nemico	intero	>=0
danno_differenza_salute	Danno per differenza salute	intero	compreso tra 0 e (forza_protagonista/3)

```
Salute_protagonista := leggere_salute_personaggio(protagonista)
forza_protagonista := leggere_forza_personaggio(protagonista)
salute_nemico := leggere_salute_personaggio(nemico)
forza_nemico := leggere_forza_personaggio(nemico)
danno:=0
MENTRE(salute_protagonista > 0 AND salute_nemico > 0)
    dado = generare_numero_casuale(1, 15)
    SE(dado <= forza_protagonista)
        ALLORA
        SE(dado = forza_protagonista/2)
            ALLORA
            salute_nemico:=salute_nemico-(forza_protagonista+2)

            ALTRIMENTI
            salute_nemico:=salute_nemico-forza_protagonista
        FINE
        differenza_salute := salute_protagonista-salute_nemico
        SE(differenza_salute <0)
            ALLORA
            danno_differenza_salute = generare_numero_casuale(0, (forza_protagonista/3) )
            salute_protagonista:= salute_protagonista-danno_differenza_salute
            danno:=danno+danno_differenza_salute
        FINE
     ALTRIMENTI SE(dado > forza_protagonista AND dado <= forza_nemico)
            ALLORA
            salute_protagonista:=salute_protagonista-dado
            danno:=danno+dado
     ALTRIMENTI
            SE(dado <= forza_nemico+2)
            ALLORA
                salute_protagonista:=salute_protagonista-(forza_nemico/2)
                danno:=danno+(forza_nemico/2)
            ALTRIMENTI
                salute_protagonista:=salute_protagonista-(dado/2)
                danno:=danno+(dado/2)
                salute_nemico:=salute_nemico-(dado/2)
            FINE
    FINE
FINE
```

Creare_mappa

Nome	Descrizione	Tipo	Vincoli
path_dimensione mappa	Percorso del file in cui c'è la dimensione della mappa	array 1 dimensione caratteri (statico)	dimensione=80
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa allocata in memoria	mappa	

Nome	Descrizione	Tipo	Vincoli
file_dimensione_ mappa	File che contiene le dimensioni della mappa	file di testo	1^riga: numero colonne 2^riga: numero righe
in_righe	Numero di righe della mappa recuperato da file	intero	>0
in_col	Numero di colonne della mappa recuperato da file	intero	>0

```
SE(esiste(PATH_DIMENSIONE_MAPPA) = falso)
    ALLORA
        stampare(“errore”)

    ALTRIMENTI
    file_dimensione_mappa := aprire_file(PATH_DIMENSIONE_MAPPA)

    in_righe := leggere_intero_da_file(file_dimensione_mappa)
    in_col := leggere_intero_da_file(file_dimensione_mappa)
    chiudere_file(PATH_DIMENSIONE_MAPPA)

    scrivere_righe(mappa_gioco, in_righe)
    scrivere_col(mappa_gioco, in_col)
    SE(in_righe > 0 AND in_col > 0)
        ALLORA
        Allocare_spazio_mappa(mappa_gioco)
    FINE
FINE
```

Aggiungere_stanze_in_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui tutte le stanze hanno un ID, una lista di oggetti, ed altre informazioni	mappa	

Nome	Descrizione	Tipo	Vincoli
stanza_da_aggiungere	Stanza da aggiungere alla mappa in posizione che dipende dai due indici i e j. Tutte le informazioni della stanza vengono ad essa assegnate con il metodo creare_stanza_da_file	stanza	l'attributo "ID_stanza" di stanza_da_aggiungere ha valore -1 se c'è stato un errore di lettura del file in cui sono memorizzati i dati della stanza
in_righe	Numero di righe della mappa	intero	>0
in_col	Numero di colonne della mappa	intero	>0

```
in_righe := leggere_righe(mappa_gioco)
in_col := leggere_col(mappa_gioco)
i:=0
MENTRE(i<in_righe)
    j:=0
    MENTRE(j<in_col)
        creare_stanza_da_file(stanza_da_aggiungere, i, j, in_righe, in_col)
        SE(leggere_id_stanza(stanza_da_aggiungere) = -1)
            ALLORA
            Stampare("errore")
            j:=in_col       //esci da entrambi i cicli
            i:=in_righe-1


            ALTRIMENTI
            scrivere_stanza_in_mappa(mappa_gioco, i, j, stanza_da_aggiungere)
         FINE
         j:=j+1
     FINE
     i:=i+1
FINE
```

Aggiungere_porte_in_mappa

Nome	Descrizione	Tipo	Vincoli
PATH_LISTA_PORTE	Percorso del file in cui c'è la lista delle porte ed altre informazioni	array 1 dimensione caratteri (statico)	dim=80
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono state aggiunte tutte le porte	mappa	

Nome	Descrizione	Tipo	Vincoli
file_lista_porte	File in cui c'è la lista delle porte ed altre informazioni	file di testo	
porta_da_aggiungere	porta da aggiungere alla mappa, più precisamente nell'array lista_porte di mappa gioco	oggetto_porta	
i,j	contatori	intero	>=0
temp_id	Generico ID letto da file	intero	

```
SE(esiste(PATH_LISTA_PORTE) = falso)
    ALLORA
        stampare("errore")
    ALTRIMENTI
    file_lista_porte := aprire_file(PATH_LISTA_PORTE)
    i:=0
    MENTRE(file_non_termina(file_lista_porte)
        // creazione da file della porta da aggiungere
        temp_id:=leggere_intero_da_file(file_lista_porte)
        scrivere_id_porta(porta_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_porte)
        scrivere_id_stanza_partenza_porta(porta_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_porte)
        scrivere_id_stanza_arrivo_porta(porta_da_aggiungere, temp_id)

        j:=0
        temp_id:=leggere_intero_da_file(file_lista_porte)
        SE(temp_id != -1)
            ALLORA
            //La porta necessita di almeno una chiave
            MENTRE(temp_id != -1)
                scrivere_id_chiave_porta(porta_da_aggiungere, j, temp_id)
                temp_id:=leggere_intero_da_file(file_lista_porte)
                j:=j+1
            FINE
        FINE
        MENTRE(j<NUM_CHIAVI_PORTA_MAX)
            scrivere_id_chiave_porta(porta_da_aggiungere, j, temp_id)
            j:=j+1
        FINE

        // scrittura in mappa della porta da aggiungere
        scrivere_porta_in_mappa(mappa_gioco, i, porta_da_aggiungere)
        i:=i+1
    FINE
    Chiudere_file(file_lista_porte)
FINE
```

Aggiungere_cibi_in_mappa

Nome	Descrizione	Tipo	Vincoli
PATH_LISTA_CIBI	Percorso del file in cui c'è la lista dei cibi ed altre informazioni	array 1 dimensione caratteri (statico)	dim=80
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono stati aggiunti tutti i cibi	mappa	

Nome	Descrizione	Tipo	Vincoli
file_lista_cibi	File in cui c'è la lista dei cibi ed altre informazioni	file di testo	
cibo_da_aggiungere	cibo da aggiungere alla mappa, più precisamente nell'array lista_cibi di mappa gioco	oggetto_cibo	
i	contatore	intero	>=0
temp_id	Generico ID letto da file	intero	

```
SE(esiste(PATH_LISTA_CIBI) = falso)
    ALLORA
        stampare("errore")
    ALTRIMENTI
    file_lista_porte := aprire_file(PATH_LISTA_CIBI)
    i:=0
    MENTRE(file_non_termina(file_lista_cibi)
        // creazione da file della porta da aggiungere
        temp_id:=leggere_intero_da_file(file_lista_cibi)
        scrivere_id_cibo(cibo_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_cibi)
        scrivere_bonus_salute_cibo(cibo_da_aggiungere, temp_id)

        // scrittura in mappa del cibo da aggiungere
        scrivere_cibo_in_mappa(mappa_gioco, i, cibo_da_aggiungere)
        i:=i+1
    FINE
    Chiudere_file(file_lista_cibi)
FINE
```

Aggiungere_armi_in_mappa

Nome	Descrizione	Tipo	Vincoli
PATH_LISTA_ARMI	Percorso del file in cui c'è la lista delle armi ed altre informazioni	array 1 dimensione caratteri (statico)	dim=80
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono stati aggiunti tutte le armi	mappa	

Nome	Descrizione	Tipo	Vincoli
file_lista_armi	File in cui c'è la lista delle armi ed altre informazioni	file di testo	
arma_da_aggiungere	arma da aggiungere alla mappa, più precisamente nell'array lista_armi di mappa gioco	oggetto_arma	
i	contatore	intero	>=0
temp_id	Generico ID letto da file	intero	

```
SE(esiste(PATH_LISTA_ARMI) = falso)
    ALLORA
        stampare("errore")
    ALTRIMENTI
    file_lista_porte := aprire_file(PATH_LISTA_ARMI)
    i:=0
    MENTRE(file_non_termina(file_lista_armi)
        // creazione da file della porta da aggiungere
        temp_id:=leggere_intero_da_file(file_lista_armi)
        scrivere_id_arma(arma_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_armi)
        scrivere_bonus_salute_arma(arma_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_armi)
        scrivere_bonus_forza_arma(arma_da_aggiungere, temp_id)


        // scrittura in mappa dell'arma da aggiungere
        scrivere_arma_in_mappa(mappa_gioco, i, arma_da_aggiungere)
        i:=i+1
    FINE
    Chiudere_file(file_lista_armi)
FINE
```

Aggiungere_personaggi_in_mappa

Nome	Descrizione	Tipo	Vincoli
PATH_LISTA_PERSONAGGI	Percorso del file in cui c'è la lista dei personaggi ed altre informazioni	array 1 dimensione caratteri (statico)	dim=80
mappa_gioco	Mappa del gioco	mappa	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono stati aggiunti tutti i personagi	mappa	

Nome	Descrizione	Tipo	Vincoli
file_lista_personaggi	File in cui c'è la lista delle armi ed altre informazioni	file di testo	
arma_da_aggiungere	Personaggio da aggiungere alla mappa, più precisamente nell'array lista_personaggi di mappa gioco	oggetto_personaggio	
i,j	contatori	intero	>=0
temp_id	Generico ID letto da file	intero	

```
SE(esiste(PATH_LISTA_PERSONAGGI) = falso)
    ALLORA
        stampare("errore")
    ALTRIMENTI
    file_lista_personaggi := aprire_file(PATH_LISTA_PERSONAGGI)
    i:=0
    MENTRE(file_non_termina(file_lista_personaggi)
        // creazione da file del personaggio da aggiungere
        temp_id:=leggere_intero_da_file(file_lista_personaggi)
        scrivere_id_personaggio(personaggio_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_personaggi)
        scrivere_salute_personaggio(personaggio_da_aggiungere, temp_id)
        temp_id:=leggere_intero_da_file(file_lista_personaggi)
        scrivere_forza_personaggio(personaggio_da_aggiungere, temp_id)

        temp_id := leggere_intero_da_file(file_lista_personaggi)

        j:=0
        SE(temp_id != -1)
            ALLORA
            //Il personaggio rilascia almeno un oggetto
            MENTRE(temp_id != -1)
                scrivere_id_oggetto_rilasciato(personaggio_da_aggiungere, j, temp_id)
                temp_id := leggere_intero_da_file(file_lista_personaggi)
                j:=j+1
            FINE
        FINE
        //Riempio gli id dei restanti oggetti rilasciati con -1
        MENTRE(j<NUM_OGGETTI_RILASCIATI_MAX)
            scrivere_id_oggetto_rilasciato(personaggio_da_aggiungere, j, temp_id)
            j:=j+1
        FINE

        scrivere_personaggio_in_mappa(mappa_gioco, i, personaggio_da_aggiungere)
        i:=I+1
    FINE
    Chiudere_file(file_lista_porte)
FINE
```

Funzione ottenere_oggetto_specifico_da_oggetto

Questa funzione serve principalmente a “convertire” un oggetto generico (avente tipo oggetto) in un oggetto specifico (oggetto_cibo, oggetto_arma, oggetto_personaggio, oggetto_porta)

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui possiamo accedere ai 4 vettori dove sono memorizzati tutti gli oggetti specifici (classificati in 4 categorie elencate sopra)	mappa	
oggetto_gioco	Oggetto da “convertire” in un oggetto specifico	oggetto	
tipo_oggetto_da restituire	Qual e’ il tipo dell’oggetto specifico da fornire in output?	intero	0 -> l’output è di tipo oggetto_porta 1 -> l’output è di tipo oggetto_cibo 2 - > l’output è di tipo oggetto_arma 3 -> l’output è di tipo oggetto_personaggio

Nome	Descrizione	Tipo	Vincoli
oggetto_specifico	Oggetto specifico	dipende dal valore di tipo_oggetto_da_restituire	il suo tipo puo’ essere uno tra oggetto_porta, oggetto_cibo, oggetto_arma, oggetto_personaggio

Nome	Descrizione	Tipo	Vincoli
id_oggetto_gioco	id dell’oggetto generico. Verrà ricercato nelle in una dei 4 array	intero	>0
id_oggetto_specifico	id dell’oggetto specifico corrente che si va ad analizzare	intero	>0
oggetto_specifico_ trovato	flag che indica se l’oggetto specifico e’ stato trovato. Consente di terminare la ricerca	booleano	
i,j	contatori	intero	>=0

```
id_oggetto_gioco := leggere_id_oggetto(oggetto_gioco)
oggetto_specifico_trovato := falso
i:=0
//32 è la dimensione di tutti e 4 gli array che contengono gli oggetti specifici
MENTRE(i<32 AND oggetto_specifico_trovato = falso)

   //dove faccio la ricerca? Dipende dal tipo_oggetto_restituire
   SE            (tipo_oggetto_da_restituire = 0)
       ALLORA
       id_oggetto_specifico := leggere_id_porta(leggere_porta_da_mappa(mappa_gioco, i))
   ALTRIMENTI SE (tipo_oggetto_da_restituire = 1)
       ALLORA
       id_oggetto_specifico := leggere_id_cibo(leggere_cibo_da_mappa(mappa_gioco, i))
   ALTRIMENTI SE (tipo_oggetto_da_restituire = 2)
       ALLORA
       id_oggetto_specifico = leggere_id_arma(leggere_arma_da_mappa(mappa_gioco, i))
   ALTRIMENTI SE(tipo_oggetto_da_restituire = 3)
       ALLORA
      id_oggetto_specifico=leggere_id_personaggio(leggere_personaggio_da_mappa(mappa_gioco, i))
   FINE

   SE(id_oggetto_specifico = id_oggetto_gioco)
       ALLORA
       j:=i
       oggetto_specifico_trovato := vero
   FINE
    i:=i+1
FINE


SE(oggetto_specifico_trovato = VERO)
    ALLORA
    SE           (tipo_oggetto_da_restituire = 0)
        ALLORA
        Oggetto_specifico := leggere_porta_da_mappa(mappa_gioco, j)
    ALTRIMENTI SE(tipo_oggetto_da_restituire = 1)
        ALLORA
        Oggetto_specifico := leggere_cibo_da_mappa(mappa_gioco, j)
    ALTRIMENTI SE(tipo_oggetto_da_restituire = 2)
        ALLORA
        Oggetto_specifico := leggere_arma_da_mappa(mappa_gioco, j)
    ALTRIMENTI SE(tipo_oggetto_da_restituire = 3)
        ALLORA
        Oggetto_specifico := leggere_personaggio_da_mappa(mappa_gioco, j)
 FINE
```

Creare_stanza_da_file

Nome	Descrizione	Tipo	Vincoli
stanza_da_aggiungere	stanza che si deve aggiungere alla mappa. E' vuota, cioè ad essa non è stato ancora assegnato un ID e nemmeno le altre informazioni del tipo "stanza".	stanza	
i_riga	Indice della riga dove si deve aggiungere la stanza nella mappa	intero	>0
j_col	Indice della colonna dove si deve aggiungere la stanza nella mappa	intero	>0
val_righe_mappa	numero di righe della mappa	intero	>0
val_col_mappa	numero di colonne della mappa	intero	>0
NUM_STANZE_COLLEGATE_MAX	Numero massimo di stanze collegate a una stanza	intero	=4
LUNG_NOME_MAX	Lunghezza massima del nome di un oggetto	intero	=80
LUNG_ATTRIBUTI_MAX	Lunghezza massima della stringa con tutti gli attributi di un oggetto	intero	=20
NUM_OGGETTI_STANZA_MAX	Numero massimo di oggetti in una stanza	intero	=10
LUNG_DESCRIZIONE_MAX	Lunghezza massima della descrizione di un oggetto/stanza	intero	=350

Nome	Descrizione	Tipo	Vincoli
stanza_da_aggiungere	Stanza da aggiungere alla mappa con tutte le informazioni	stanza	

Nome	Descrizione	Tipo	Vincoli
path_lista_collegamenti stanze	Percorso del file in cui sono memorizzati i collegamenti delle stanze	array 1 dimensione caratteri (s)	dimensione=80
file_lista_collegamenti stanze	File in cui ci sono i collegamenti delle stanze	file binario	
fine	flag che, se posto a vero, indica di interrompere un'operazione di ricerca	booleano	
id_stanza_assegnato	id della stanza assegnato a stanza_da_agigungere	intero	>0
path_lista_oggetti_stanze	Percorso dove è memorizzato il file con gli oggetti delle stanze	array 1 dimensione caratteri (s)	dimensione=80
file_lista_oggetti stanze	File dove sono salvati gli oggetti delle stanze	file binario	
path_lista_descrizioni_stanze	Percorso dove è salvato il file con le descrizioni delle stanze	array 1 dimensione caratteri (s)	dimensione=80
file_lista_descrizioni_stanze	File dove sono salvate le descrizioni delle stanze	file binario	

```
// ====================== CALCOLO ID STANZA DA AGGIUNGERE ======================
SE(esiste(PATH_LISTA_COLLEGAMENTI_STANZE) = falso)
    ALLORA
        scrivere_id_stanza(stanza_da_aggiungere, -1)
        stampare("errore")

    ALTRIMENTI
    file_collegamenti_stanze = aprire_file("lista_id_collegamenti_stanze.txt")

    SE(i_riga != val_righe_mappa-1 )
        ALLORA
        //Se si sta creando una stanza NON isolata della mappa (cioe che NON compare all'ultima
        //riga come le stanze 50, 120, 130 ecc..) si calcola l'id facilmente data la riga e la
colonna)
        id_stanza_assegnato := val_col_mappa*(i_riga)+(j_col+1)
        scrivere_id_stanza(stanza_da_aggiungere, id_stanza_assegnato)

         posizionare_su_file(file_collegamenti_stanze, dimensione(intero) *
                                                        (NUM_STANZE_COLLEGATE_MAX+1) *
                                                      leggere_id_stanza(stanza_da_aggiungere)-1),
                                                inizio_file)

        ALTRIMENTI
        //Altrimenti il calcolo dell'id si rimanda a dopo e si mette temporaneamente id = -1
        scrivere_id_stanza(stanza_da_aggiungere, -1)

        posizionare_su_file(file_collegamenti_stanze, dimensione(intero) *
                                                      (NUM_STANZE_COLLEGATE_MAX+1) *
                                                       ( val_col_mappa * (val_righe_mappa -1) ) +
                                                       (   dimensione(intero) *
                                                       (NUM_STANZE_COLLEGATE_MAX+1) * j_col),
                                                    inizio_file
    FINE
    // ================ LETTURA ID STANZE COLLEGATE A STANZA DA AGGIUNGERE ==============
    Fine:=falso
    MENTRE(file_non_termina(file_collegamenti_stanze) AND fine=falso)
        // I campi di ogni record del file lista_collegamenti_stanze sono 5
        //(l'id della stanza corrente + 4 id stanze collegate)
        i:=0
        MENTRE(i<NUM_STANZE_COLLEGATE_MAX+1)
            SE(i=0)
                ALLORA
                // Il primo campo di ogni record del file lista_collegamenti_stanze e'
                // l'id della stanza corrente.
                // Questo viene assegnato alla stanza_da_aggiungere se questa e' una
                //  stanza isolata

                Id_stanza_assegnato:=leggere_intero_da_file(file_collegamenti_stanze)
                SE(i_riga = val_righe_mappa-1 )
                    scrivere_id_stanza(stanza_da_aggiungere, id_stanza_assegnato)

                ALTRIMENTI
                //Gli altri campi del record vengono assegnati al vettore
                // collegamenti_id_stanza mediante metodo d'accesso

                Temp_id_stanza_collegata:=leggere_intero_da_file(file_collegamenti_stanze)
                scrivere_collegamento_id_stanza(stanza_da_aggiungere, i-1,
                                                temp_id_stanza_collegata)
            FINE
```

```
        i:=i+1
    FINE
    Fine:=vero
Chiudere_file(file_collegamenti_stanze)

// ====================== LETTURA OGGETTI STANZA ======================
SE(esiste("lista_oggetti_stanze.bin") = falso)
    ALLORA
    scrivere_id_stanza(stanza_da_aggiungere, -1)
    stampare("errore")

    ALTRIMENTI
    file_lista_oggetti_stanze := aprire_file("lista_oggetti_stanze.bin")
// Se si devono recuperare gli oggetti di una stanza NON isolata utilizza l'id della stanza
// (1-20) per posizionarti nel file binario. Altrimenti posizionati nel file binario a fine
// oggetti della stanza 20 e utilizza l'indice di colonna per posizionarti

    SE(i_riga != val_righe_mappa-1 )
        ALLORA
        posizionare_su_file(file_lista_oggetti_stanze, ( dimensione(intero) * 3 +
                                                         LUNG_NOME_MAX +
                                                         LUNG_ATTRIBUTI_MAX
                                                       ) * NUM_OGGETTI_STANZA_MAX *
                                                   (leggere_id_stanza(stanza_da_aggiungere)-1),
                                              inizio_file)
        ALTRIMENTI
        posizionare_su_file(file_lista_oggetti_stanze,( dimensione(intero) * 3 +
                                    LUNG_NOME_MAX +
                                    LUNG_ATTRIBUTI_MAX
                                    ) * NUM_OGGETTI_STANZA_MAX *
                                    ( val_col_mappa * (val_righe_mappa -1) ) +
                                    ( dimensione(intero) * 3 +
                                    LUNG_NOME_MAX +
                                    LUNG_ATTRIBUTI_MAX
                                    ) * NUM_OGGETTI_STANZA_MAX *
                                    j_col, inizio_file)
     FINE
     i:=0
     MENTRE(file_non_termina(file_lista_oggetti_stanze) AND i<NUM_OGGETTI_STANZA_MAX)

         temp_id_stanza_oggetto := leggere_intero_da_file(file_lista_oggetti_stanze)
         Temp_id_ogg_genitore := leggere_intero_da_file(file_lista_oggetti_stanze)
         Temp_id_oggetto := leggere_intero_da_file(file_lista_oggetti_stanze)
         temp_nome_oggetto := leggere_stringa_da_file(file_lista_oggetti_stanze)
         Temp_attributi_oggetto := leggere_stringa_da_file(file_lista_oggetti_stanze)

       scrivere_id_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere, i),temp_id_oggetto)
          scrivere_id_oggetto_genitore(leggere_oggetto_stanza(stanza_da_aggiungere, i),
                                  temp_id_ogg_genitore)
          scrivere_nome_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere, i),
                           temp_nome_oggetto)
          scrivere_attributi_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere, i),
                           temp_attributi_oggetto )

    //Se l'oggetto letto e' un oggetto vuoto o non valido, questo non e' visibile
          SE(temp_id_oggetto = -1)
              ALLORA
        scrivere_visibilita_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere, i), falso)
        ALTRIMENTI
```

```
          scrivere_descrizione_oggetto_da_file(leggere_oggetto_stanza(stanza_da_aggiungere, i))
            scrivere_visibilita_oggetto (leggere_oggetto_stanza(stanza_da_aggiungere, i), vero)
             FINE
             i:=i+1
         FINE

    // ================== LETTURA OGGETTI CONTENUTI (OGGETTI FIGLI) ==================
    //Scansiona tutti gli oggetti visibili della stanza corrente aventi attributo 'a'
    // Ricordiamo che l'array di oggetti lista_oggetti_contenuti contiene tutti gli oggetti che
sono contenuti
    // in qualsiasi oggetto apribile della stanza. Es. Nella stanza x ci sono due oggetti
apribili:
    // - teca:     contiene oggetto1
    // - cassetto: contiene oggetto2, oggetto 5
    // Allora l'array lista_oggetti_contenuti (relativo a stanza x) contiene oggetto1,
oggetto2, oggetto5

    j:=0
    i:=0
    MENTRE(i<NUM_OGGETTI_STANZA_MAX)
        SE(leggere_tipo_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere, i)) =
           ID_TIPO_OGGETTO_GENITORE AND
           leggere_visibilita_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere,i)) = vero)

           ALLORA
            //Posizionati nel file lista_oggetti_stanze a partire da dove iniziano gli oggetti
            //contenuti, cioe' dopo gli oggetti di tutte le stanze

            Posizionamento_file(file_lista_oggetti_stanze,( dimensione(intero) * 3 +
                                    LUNG_NOME_MAX +
                                    LUNG_ATTRIBUTI_MAX
                                    ) * NUM_OGGETTI_STANZA_MAX *
                                    ( val_col_mappa * val_righe_mappa ), inizio_file)

            MENTRE(file_non_termina(file_lista_oggetti_stanze) AND
                   j<NUM_OGGETTI_CONTENUTI_PER_STANZA_MAX)
                temp_id_stanza_oggetto := leggere_intero_da_file(file_lista_oggetti_stanze)
                Temp_id_ogg_genitore := leggere_intero_da_file(file_lista_oggetti_stanze)
                Temp_id_oggetto := leggere_intero_da_file(file_lista_oggetti_stanze)
                temp_nome_oggetto := leggere_stringa_da_file(file_lista_oggetti_stanze)
                Temp_attributi_oggetto := leggere_stringa_da_file(file_lista_oggetti_stanze)

                scrivere_id_oggetto(temp_oggetto_contenuto, temp_id_oggetto)
                scrivere_id_oggetto_genitore(temp_oggetto_contenuto, temp_id_ogg_genitore)
                scrivere_nome_oggetto       (temp_oggetto_contenuto, temp_nome_oggetto)
                scrivere_descrizione_oggetto_da_file(temp_oggetto_contenuto)
                scrivere_attributi_oggetto  (temp_oggetto_contenuto, temp_attributi_oggetto)
                scrivere_visibilita_oggetto (temp_oggetto_contenuto, vero)

                SE(leggere_id_oggetto(leggere_oggetto_stanza(stanza_da_aggiungere,i)) =
                                        temp_id_ogg_genitore)
                    ALLORA
                   scrivere_oggetto_contenuto(stanza_da_aggiungere, j, temp_oggetto_contenuto)
                    j:=j+1
                FINE
             FINE
        FINE
        i:=i+1
    FINE
```

```
    //Riempi gli elementi restanti dell'array con oggetti vuoti
    MENTRE(j<NUM_OGGETTI_CONTENUTI_PER_STANZA_MAX)
        scrivere_id_oggetto         (temp_oggetto_contenuto, -1)
        scrivere_id_oggetto_genitore(temp_oggetto_contenuto, -1)
        scrivere_nome_oggetto       (temp_oggetto_contenuto, “n”)
        //scrivere_attributi_oggetto  (temp_oggetto_contenuto, “n”)
        scrivere_visibilita_oggetto (temp_oggetto_contenuto, falso)

        scrivere_oggetto_contenuto(stanza_da_aggiungere, j, temp_oggetto_contenuto)
        j:=j+1
    FINE
    Chiudere_file(file_lista_oggetti_stanze)

    // ================== LETTURA DESCRIZIONE STANZA ==================
    SE(esiste(PATH_LISTA_DESCRIZIONI_STANZE) = falso)
        ALLORA
        scrivere_id_stanza(stanza_da_aggiungere, -1)
        stampare(“errore”)

        ALTRIMENTI
        file_lista_descrizioni_stanze := aprire_file(PATH_LISTA_DESCRIZIONI_STANZE)

        SE(i_riga != val_righe_mappa-1 )
            ALLORA
             Posizionamento_file(file_lista_descrizioni_stanze,( dimensione(intero) +
                                                                 LUNG_DESCRIZIONE_MAX
                                                ) * leggere_id_stanza(stanza_da_aggiungere)-1),
                                                 inizio_file)
            ALTRIMENTI
            Posizionamento_file(file_lista_descrizioni_stanze,( dimensione(intero) +
                                        LUNG_DESCRIZIONE_MAX
                                        ) *
                                        ( val_col_mappa * (val_righe_mappa -1) ) +
                                        ( dimensione(intero) +
                                        LUNG_DESCRIZIONE_MAX
                                        ) * j_col, inizio_file)
         FINE
         temp_id := leggere_intero_da_file(file_lista_descrizioni_stanze)
         temp_descrizione_stanza := leggere_stringa_da_file(file_lista_descrizioni_stanze)

          scrivere_descrizione_stanza(stanza_da_aggiungere, temp_descrizione_stanza)

         chiudere_file(file_lista_descrizioni_stanze)
     FINE
FINE
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI mappa - scrivere_righe

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui impostiamo il numero di righe	mappa	
val_righe	Numero di righe da impostare nella mappa	intero	>0

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui è stato scritto il numero di righe	mappa	

```
mappa_gioco.righe := val_righe
```

leggere_righe

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui leggiamo il numero di righe	mappa	

Nome	Descrizione	Tipo	Vincoli
val_righe	Numero di righe letto dalla mappa	intero	>0

```
val_righe := mappa_gioco.righe
```

scrivere_col

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui impostiamo il numero di colonne	mappa	
val_col	Numero di colonne da impostare nella mappa	intero	>0

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui è stato scritto il numero di colonne	mappa	

```
mappa_gioco.col := val_col
```

leggere_col

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui leggiamo il numero di colonne	mappa	

Nome	Descrizione	Tipo	Vincoli
val_col	Numero di colonne letto dalla mappa	intero	>0

```
val_col := mappa_gioco.col
```

scrivere_porta_in_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui aggiungiamo una porta	mappa	
val_pos	Posizione della porta nell'array elenco_porte della mappa	intero	>=0
porta_da_aggiungere	Porta da aggiungere nell'array elenco_porte	oggetto_porta	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui è stata aggiunta la porta	mappa	

```
mappa_gioco.elenco_porte[val_pos] := porta_da_aggiungere
```

leggere_porta_da_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono presenti tutte le porte dalle quali vogliamo leggerne una	mappa	
val_pos	Posizione della porta da leggere nell'array elenco_porte della mappa	intero	>=0

Nome	Descrizione	Tipo	Vincoli
porta_letta	Porta letta dalla mappa	oggetto_porta	

```
Porta_letta := mappa_gioco.elenco_porte[val_pos]
```

scrivere_cibo_in_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui aggiungiamo un cibo	mappa	
val_pos	Posizione del cibo nell'array elenco_cibi della mappa	intero	>=0
cibo_da_aggiungere	Cibo da aggiungere nell'array elenco_cibi	oggetto_cibo	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui è stata aggiunta il cibo	mappa	

```
mappa_gioco.elenco_cibi[val_pos] := cibo_da_aggiungere
```

leggere_cibo_da_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono presenti tutti i cibi dalle quali vogliamo leggerne uno	mappa	
val_pos	Posizione del cibo da leggere nell'array elenco_cibi della mappa	intero	>=0

Nome	Descrizione	Tipo	Vincoli
cibo_letto	Cibo letto dalla mappa	oggetto_cibo	

```
cibo_letto := mappa_gioco.elenco_cibi[val_pos]
```

scrivere_arma_in_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui aggiungiamo un'arma	mappa	
val_pos	Posizione dell'arma nell'array elenco_armi della mappa	intero	>=0
arma_da_aggiungere	Arma da aggiungere nell'array elenco_armi	oggetto_arma	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui è stata aggiunta l'arma	mappa	

```
mappa_gioco.elenco_armi[val_pos] := arma_da_aggiungere
```

leggere_arma_da_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono presenti tutte le armi dalle quali vogliamo leggerne una	mappa	
val_pos	Posizione dell'arma da leggere nell'array elenco_armi della mappa	intero	>=0

Nome	Descrizione	Tipo	Vincoli
arma_letto	arma letta dalla mappa	oggetto_arma	

```
arma_letta := mappa_gioco.elenco_armi[val_pos]
```

scrivere_personaggio_in_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui aggiungiamo un personaggio	mappa	
val_pos	Posizione del personaggio nell'array elenco_personaggi della mappa	intero	>=0
personaggio_ da_aggiungere	Personaggio da aggiungere nell'array elenco_personaggi	oggetto_ personaggio	

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui è stato aggiunto il personaggio	mappa	

```
mappa_gioco.elenco_personaggi[val_pos] := personaggio_da_aggiungere
```

leggere_personaggio_da_mappa

Nome	Descrizione	Tipo	Vincoli
mappa_gioco	Mappa in cui sono presenti tutti i personaggi dai quali vogliamo leggerne uno	mappa	
val_pos	Posizione del personaggio da leggere nell'array elenco_personaggi della mappa	intero	>=0

Nome	Descrizione	Tipo	Vincoli
personaggio_letto	personaggio letta dalla mappa	oggetto_ personaggio	

```
personaggio_letto := mappa_gioco.elenco_personaggi[val_pos]
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI oggetto_porta - scrivere_id_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale vogliamo assegnare l'id	oggetto_porta	
val_id	ID assegnato alla porta	intero	>=0

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale è stato assegnato l'id	oggetto_porta	

```
porta.id_porta := val_id
```

leggere_id_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta dalla quale vogliamo prelevare l'id	oggetto_porta	

Nome	Descrizione	Tipo	Vincoli
val_id	id della porta di input	intero	>=0

```
Val_id := porta.id_porta
```

scrivere_id_stanza_partenza_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale vogliamo assegnare l'id della stanza di partenza	oggetto_porta	
val_id_stanza_partenza_porta	ID della stanza di partenza della porta	intero	>0

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale è stato assegnato l'id della stanza di partenza	oggetto_porta	

```
porta.id_stanza_partenza_porta := val_id_stanza_partenza_porta
```

leggere_id_stanza_partenza_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta dalla quale vogliamo prelevare l'id della stanza di partenza	oggetto_porta	

Nome	Descrizione	Tipo	Vincoli
val_id_stanza_partenza_porta	id della stanza di partenza porta di input	intero	>0

```
Val_id_stanza_partenza_porta := porta.id_stanza_partenza_porta
```

scrivere_id_stanza_arrivo_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale vogliamo assegnare l'id della stanza di arrivo	oggetto_porta	
val_id_stanza_arrivo_porta	ID della stanza di arrivo della porta	intero	>0

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale è stato assegnato l'id della stanza di arrivo	oggetto_porta	

```
porta.id_stanza_arrivo_porta := val_id_stanza_arrivo_porta
```

leggere_id_stanza_arrivo_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta dalla quale vogliamo prelevare l'id della stanza di arrivo	oggetto_porta	

Nome	Descrizione	Tipo	Vincoli
val_id_stanza_arrivo_porta	id della stanza di arrivo della porta di input	intero	>0

```
Val_id_stanza_arrivo_porta := porta.id_stanza_arrivo_porta
```

scrivere_id_chiave_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale vogliamo assegnare una chiave che permette di varcarla	oggetto_porta	
val_indice_chiave	Indice della chiave nell'array lista_id_chiavi_porta di porta	intero	>=0
val_id_chiave_porta	ID della chiave che permette di varcare porta	intero	>0

Nome	Descrizione	Tipo	Vincoli
porta	Porta alla quale è stata assegnata una chiave che la varca	oggetto_porta	

```
Porta.lista_id_chiavi_porta[val_indice_chiave] := val_id_chiave_porta
```

leggere_id_chiave_porta

Nome	Descrizione	Tipo	Vincoli
porta	Porta dalla quale vogliamo prelevare l'id della val_indice_chiave-esima chiave che la varca	oggetto_porta	
val_indice_chiave	Indice della chiave nell'array lista_id_chiavi_porta di porta	intero	>=0

Nome	Descrizione	Tipo	Vincoli
val_id_chiave_porta	id della val_indice_chiave-esima chiave che varca porta	intero	>0

```
Val_id_chiave_porta := lista_id_chiavi_porta[val_indice_chiave]
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI oggetto_personaggio - scrivere_id_personaggio

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale vogliamo assegnare l'id	oggetto_personaggio	
val_id	ID assegnato al personaggio	intero	>=0

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale è stato assegnato l'id	oggetto_personaggio	

```
personaggio.id_personaggio := val_id
```

leggere_id_personaggio

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio dal quale vogliamo prelevare l'id	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
val_id	id personaggio di input	intero	>=0

```
Val_id := personaggio.id_personaggio
```

scrivere_salute_personaggio

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale vogliamo cambiare la salute	oggetto_personaggio	
val_salute	valore della salute da assegnare a personaggio	intero	>=0
flag_protagonista	valore booleano che indica se stiamo modificando la salute del protagonista	booleano	vero=protagonista falso= no protag.
SALUTE_PROTAGONISTA_INIZIALE	salute che ha il protagonista a inizio gioco o quando perde una vita	intero	=20
num_vite_corrente	Numero vite corrente del protagonista	intero	>=0 se vale la vita zero, >0 se non vale

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale è stata modificata la salute	oggetto_personaggio	

```
SE(val_salute <= 0)
   ALLORA
   SE(flag_protagonista = vero)
       ALLORA
       num_vite_corrente:=num_vite_corrente-1
       personaggio.salute_personaggio := SALUTE_PROTAGONISTA_INIZIALE

       ALTRIMENTI
       personaggio.salute_personaggio := 0
   FINE
   ALTRIMENTI
   personaggio.salute_personaggio := val_salute
FINE
```

leggere_salute_personaggio

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio dal quale vogliamo prelevare la salute	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
val_salute	salute personaggio di input	intero	>=0

```
Val_salute := personaggio.salute_personaggio
```

scrivere_forza_personaggio

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale vogliamo cambiare la forza	oggetto_personaggio	
val_forza	valore della forza da assegnare a personaggio	intero	>=0
FORZA_PERSONAGGIO_MAX	Forza massima che ha il personaggio	intero	=15

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale è stata modificata la forza	oggetto_personaggio	

```
SE(val_forza >= FORZA_PERSONAGGIO_MAX)
   ALLORA
   personaggio.forza_personaggio := FORZA_PERSONAGGIO_MAX

   ALTRIMENTI
   personaggio.forza_personaggio := val_forza
FINE
```

leggere_forza_personaggio

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio dal quale vogliamo prelevare la salute	oggetto_personaggio	

Nome	Descrizione	Tipo	Vincoli
val_forza	forza personaggio di input	intero	>=0, <= 15

```
Val_forza := personaggio.forza_personaggio
```

scrivere_id_oggetto_rilasciato

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale vogliamo assegnare l'id di un oggetto che rilascia	oggetto_personaggio	
val_indice_oggetto	indice dell'oggetto che personaggio rilascia	intero	>=0
val_id_oggetto_rilasciato	ID dell'oggetto che personaggio rilascia	intero	

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio al quale è stato aggiunto un oggetto rilasciato	oggetto_personaggio	

```
personaggio.lista_id_oggetti_rilasciati[val_indice_oggetto] := val_id_oggetto_rilasciato
```

leggere_id_oggetto_rilasciato

Nome	Descrizione	Tipo	Vincoli
personaggio	Personaggio dal quale prelevo l'id di un oggetto che rilascia	oggetto_personaggio	
val_indice_oggetto	indice dell'oggetto che personaggio rilascia	intero	>=0

Nome	Descrizione	Tipo	Vincoli
val_id_oggetto_rilasciato	ID dell'oggetto che personaggio rilascia	intero	

```
Val_id_oggetto_rilasciato := personaggio.lista_id_oggetti_rilasciati[val_indice_oggetto]
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI oggetto_cibo - scrivere_id_cibo

Nome	Descrizione	Tipo	Vincoli
cibo	Cibo al quale vogliamo assegnare l'id	oggetto_cibo	
val_id_cibo	ID assegnato al cibo	intero	>0

Nome	Descrizione	Tipo	Vincoli
cibo	cibo al quale è stato assegnato l'id	oggetto_cibo	

```
cibo.id_cibo := val_id_cibo
```

leggere_id_cibo

Nome	Descrizione	Tipo	Vincoli
cibo	Cibo dal quale vogliamo prelevare l'id	oggetto_cibo	

Nome	Descrizione	Tipo	Vincoli
val_id_cibo	id del cibo di input	intero	>0

```
Val_id_cibo := cibo.id_cibo
```

scrivere_bonus_salute_cibo

Nome	Descrizione	Tipo	Vincoli
cibo	Cibo al quale vogliamo assegnare il suo bonus salute	oggetto_cibo	
val_bonus_salute_cibo	Valore del bonus che si vuole assegnare al cibo	intero	>0

Nome	Descrizione	Tipo	Vincoli
cibo	Cibo al quale è stato assegnato il suo bonus salute	oggetto_cibo	

```
cibo.bonus_salute_cibo:= val_bonus_salute_cibo
```

leggere_bonus_salute_cibo

Nome	Descrizione	Tipo	Vincoli
cibo	Cibo dal quale vogliamo prelevare il suo bonus salute	oggetto_cibo	

Nome	Descrizione	Tipo	Vincoli
val_bonus_salute_cibo	Bonus salute del cibo di input	intero	>0

```
val_bonus_salute_cibo:= cibo.bonus_salute_cibo
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI oggetto_arma - scrivere_id_arma

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale vogliamo assegnare l'id	oggetto_arma	
val_id_arma	ID assegnato all'arma	intero	>0

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale è stato assegnato l'id	oggetto_arma	

```
arma.id_arma := val_id_arma
```

leggere_id_arma

Nome	Descrizione	Tipo	Vincoli
cibo	Arma dal quale vogliamo prelevare l'id	oggetto_arma	

Nome	Descrizione	Tipo	Vincoli
val_id_arma	id dell'arma di input	intero	>0

```
Val_id_arma := arma.id_arma
```

scrivere_bonus_salute_arma

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale vogliamo assegnare il suo bonus salute	oggetto_arma	
val_bonus_salute_arma	Valore del bonus salute che si vuole assegnare all'arma	intero	>0

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale è stato assegnato il suo bonus salute	oggetto_arma	

```
arma.bonus_salute_arma:= val_bonus_salute_arma
```

leggere_bonus_salute_arma

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale vogliamo prelevare il suo bonus salute	oggetto_arma	

Nome	Descrizione	Tipo	Vincoli
val_bonus_salute_arma	Bonus salute dell'arma di input	intero	>0

```
val_bonus_salute_arma:= arma.bonus_salute_arma
```

scrivere_bonus_salute_forza

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale vogliamo assegnare il suo bonus forza	oggetto_arma	
val_bonus_forza_arma	Valore del bonus forza che si vuole assegnare all'arma	intero	>0

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale è stato assegnato il suo bonus forza	oggetto_arma	

```
arma.bonus_salute_arma:= val_bonus_salute_arma
```

leggere_bonus_salute_arma

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale vogliamo prelevare il suo bonus forza	oggetto_arma	

Nome	Descrizione	Tipo	Vincoli
val_bonus_forza_arma	Bonus forza dell'arma di input	intero	>0

```
val_bonus_forza_arma:= arma.bonus_forza_arma
```

scrivere_stato_equipaggiato_arma

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale vogliamo assegnare il suo stato equipaggiato	oggetto_arma	
val_stato_equipaggiato_arma	Valore dello stato equipaggiato che si vuole assegnare all'arma	booleano	vero = arma equipaggiato, falso = altrimenti

Nome	Descrizione	Tipo	Vincoli
arma	Arma alla quale è stato assegnato il suo stato equipaggiato	oggetto_arma	

```
arma.stato_equipaggiato_arma:= val_stato_equipaggiato_arma
```

leggere_stato_equipaggiato_arma

Nome	Descrizione	Tipo	Vincoli
arma	Arma dalla quale vogliamo prelevare il suo stato equipaggiato	oggetto_arma	

Nome	Descrizione	Tipo	Vincoli
val_stato_equipaggiato_arma	Valore dello stato equipaggiato che si è prelevato dall'arma	booleano	vero = arma equipaggiato, falso = altrimenti

```
val_stato_equipaggiato_arma := arma.stato_equipaggiato_arma
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI oggetto - scrivere_id_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui assegniamo l'id	oggetto	
val_id	ID assegnato all'oggetto	intero	>0

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui è assegnato l'id	oggetto	

```
Oggetto_gioco.id_oggetto := val_id
```

leggere_id_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo prelevare l'id	oggetto	

Nome	Descrizione	Tipo	Vincoli
val_id	ID assegnato all'oggetto	intero	>0

```
val_id := oggetto_gioco.id_oggetto
```

scrivere_id_oggetto_genitore

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui assegniamo l'id dell'oggetto genitore	oggetto	
val_id_oggetto_genitore	ID dell'oggetto genitore assegnato a oggetto_gioco	intero	>0

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui è assegnato l'id dell'oggetto genitore	oggetto	

```
Oggetto_gioco.id_oggetto_genitore := val_id_oggetto_genitore
```

leggere_id_oggetto_genitore

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo prelevare l'id dell'oggetto genitore	oggetto	

Nome	Descrizione	Tipo	Vincoli
val_id_oggetto_genitore	ID dell'oggetto genitore assegnato a oggetto_gioco	intero	>0

```
val_id_oggetto_genitore := oggetto_gioco.id_oggetto_genitore
```

scrivere_nome_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui assegniamo il nome	oggetto	
val_nome_oggetto	nome da assegnare a oggetto_gioco	array 1 dimensione caratteri (statico)	dim=80

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui è assegnato il nome	oggetto	

```
Copiare_stringa(val_nome_oggetto, oggetto_gioco.nome_oggetto)
```

leggere_nome_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo prelevare il suo nome	oggetto	

Nome	Descrizione	Tipo	Vincoli
val_nome_oggetto	Nome dell'oggetto assegnato a oggetto_gioco	array 1 dimensione caratteri (statico)	dim=80

```
Copiare_stringa(oggetto_gioco.nome_oggetto ,val_nome_oggetto)
```

scrivere_descrizione_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui assegniamo la descrizione	oggetto	
val_descrizione_oggetto	Descrizione da assegnare a oggetto_gioco	array 1 dimensione caratteri (statico)	dim=350

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui è assegnato la descrizione	oggetto	

```
Copiare_stringa(val_descrizione_oggetto , oggetto_gioco.descrizione_oggetto)
```

leggere_descrizione_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo prelevare la sua descrizione	oggetto	

Nome	Descrizione	Tipo	Vincoli
val_descrizione_oggetto	Descrizione dell'oggetto assegnato a oggetto_gioco	array 1 dimensione caratteri (statico)	dim=350

```
Copiare_stringa(oggetto_gioco.descrizione_oggetto , val_descrizione_oggetto)
```

scrivere_visibilita_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui assegniamo la visibilità	oggetto	
val_visibilita_oggetto	Flag che indica se quell'oggetto deve essere visibile	booleano	vero=visibile, falso=no visibile

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto gioco a cui è stata assegnata la sua visibilità	array 1 dimensione caratteri (statico)	

```
oggetto_gioco.visibilita_oggetto := val_visibilita_oggetto
```

leggere_visibilita_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo prelevare la sua visibilità	oggetto	

Nome	Descrizione	Tipo	Vincoli
val_visibilita_oggetto	Flag che indica se quell'oggetto è visibile	booleano	vero=visibile, falso=no visibile

```
val_visibilita_oggetto := oggetto_gioco.visibilita_oggetto
```

scrivere_attributi_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui assegniamo la descrizione	oggetto	
val_attributi_oggetto	Stringa che contiene gli attributi dell'oggetto, separati dalla virgola	array 1 dimensione caratteri (statico)	dim=LUNG_ATTR_MAX
NUM_ATTRIBUTI_OGGETTO_MAX	Numero di attributi massimo che un oggetto può avere	intero	=11
LUNG_ATTRIBUTI_MAX	Lunghezza massima di val_attributi_oggetto	intero	

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto a cui sono stati assegnati gli attributi partendo dalla stringa ed eliminando le virgole	oggetto	

Nome	Descrizione	Tipo	Vincoli
i, c	contatore	intero	>=0

```
i:=0
MENTRE(i<NUM_ATTRIBUTI_OGGETTO_MAX)
    oggetto_gioco.attributi_oggetto[i] := 'n'
    i:=i+1
FINE
c:=0
i:=0
MENTRE(i<LUNG_ATTRIBUTI_MAX)
    SE(val_attributi_oggetto[i] != ',' AND val_attributi_oggetto[i] != 'n')
       ALLORA
       oggetto_gioco.attributi_oggetto[c] := val_attributi_oggetto[i]
       c:=c+1

       ALTRIMENTI
          SE(val_attributi_oggetto[i] = 'n')
             ALLORA
             i:=LUNG_ATTRIBUTI_MAX
          FINE
       i:=i+1
    FINE
FINE
```

leggere_attributi_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo prelevare i suoi attributi	oggetto	
NUM_ATTRIBUTI_OGGETTO_MAX	Numero di attributi massimo che un oggetto può avere	intero	=11

Nome	Descrizione	Tipo	Vincoli
val_attributi_oggetto	Attributi dell'oggetto	array 1 dimensione caratteri (statico)	dim=NUM_ATTRIBUTI_MAX

```
val_attributi_oggetto := oggetto_gioco.attributi_oggetto
```

leggere_tipo_oggetto
tipo_oggetto non e' un attributo "definito" del tipo di dato oggetto ma è un attributo "calcolato" utilizzando l'attributo attributi_oggetto

Nome	Descrizione	Tipo	Vincoli
oggetto_gioco	Oggetto da cui vogliamo calcolare il suo tipo	oggetto	
NUM_TIPI_OGGETTO_ MAX	Gli oggetti possono essere classificati in NUM_TIPI_OGGETTO_MAX tipi	intero	=5
ID_TIPO_OGGETTO_PORTA	Numero che indica che un oggetto e' una porta	intero	=0
ID_TIPO_OGGETTO_CIBO	Numero che indica che un oggetto e' un cibo	intero	=1
ID_TIPO_OGGETTO_ARMA	Numero che indica che un oggetto e' un'arma.	intero	=2
ID_TIPO_OGGETTO_PERSONAGGIO	Numero che indica che un oggetto e' un person.	intero	=3
ID_TIPO_OGGETTO_GENITORE	Numero che indica che un oggetto e' un oggetto che contiene altri oggetti	intero	=4

Nome	Descrizione	Tipo	Vincoli
classificazione_oggetto	Classificazione dell'oggetto	intero	compreso tra 0 e 4

Nome	Descrizione	Tipo	Vincoli
i	contatore	intero	>=0
attributo_da_trovare	Attributo da trovare nell'oggetto	carattere	
attributo_trovato	Flag che indica se l'attributo è stato trovato	intero	>=0 =trovato -1 = non trovato

```
classificazione_oggetto := -1
i:=0
MENTRE(i<NUM_TIPI_OGGETTO_MAX+2 AND  classificazione_oggetto = -1)
    SE     (i=ID_TIPO_OGGETTO_PORTA)
        ALLORA
        attributo_da_trovare := 'A'
    ALTRIMENTI SE(i=ID_TIPO_OGGETTO_CIBO)
        ALLORA
        attributo_da_trovare := 'm'
    ALTRIMENTI SE(i=ID_TIPO_OGGETTO_ARMA)
        ALLORA
        attributo_da_trovare := 'e'
    ALTRIMENTI SE(i=ID_TIPO_OGGETTO_PERSONAGGIO)
        ALLORA
        attributo_da_trovare := 'w'
    ALTRIMENTI SE(i=ID_TIPO_OGGETTO_GENITORE)
        ALLORA
        attributo_da_trovare := 'a'
    ALTRIMENTI SE(i=NUM_TIPI_OGGETTO_MAX)
        ALLORA
        attributo_da_trovare := 's'
    ALTRIMENTI
        attributo_da_trovare := 'S'

        attributo_trovato := ricercare_attributo(leggere_attributi_oggetto(oggetto_gioco),
attributo_da_trovare)
        SE(carattere_trovato != -1)
            ALLORA
            classificazione_oggetto := i
```

```
            SE(classificazione_oggetto = NUM_TIPI_OGGETTO_MAX OR classificazione_oggetto =
NUM_TIPI_OGGETTO_MAX+1)
                ALLORA
                classificazione_oggetto := ID_TIPO_OGGETTO_PORTA
            FINE
        FINE
        i:=i+1
    FINE
FINE
```

FUNZIONI DI ACCESSO AGLI ATTRIBUTI DI stanza - scrivere_id_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza a cui assegniamo l'id	stanza	
val_id	ID da assegnare a stanza_mappa	intero	>0

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza a cui è assegnato l'ID	stanza	

```
Stanza_mappa.id_stanza := val_id
```

leggere_id_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza da cui preleviamo l'id	stanza	

Nome	Descrizione	Tipo	Vincoli
val_id	ID prelevato da stanza_mappa	intero	>0

```
val_id := stanza_mappa.id_stanza
```

scrivere_collegamento_id_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza a cui assegniamo l'id della stanza collegata	stanza	
pos_stanza	Indice dell'id della stanza collegata	intero	>=0
id_stanza_collegata	ID della stanza collegata a stanza mappa	intero	>0

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza a cui è assegnato l'ID della stanza collegata	stanza	

```
stanza_mappa.collegamenti_id_stanza[pos_stanza] := id_stanza_collegata
```

leggere_collegamento_id_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza da cui preleviamo l'id della stanza collegata	stanza	
pos_stanza	Indice dell'id della stanza collegata	intero	>=0

Nome	Descrizione	Tipo	Vincoli
id_stanza_collegata	ID della stanza collegata a stanza mappa	intero	>0

```
id_stanza_collegata := stanza_mappa.collegamenti_id_stanza[pos_stanza]
```

scrivere_oggetto_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza in cui aggiungiamo l'oggetto	stanza	
pos_oggetto	Indice dell'oggetto della stanza	intero	>=0
val_oggetto	Oggetto da inserire nella stanza	oggetto	

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza in cui è stato aggiunto l'oggetto	stanza	

```
stanza_mappa.lista_oggetti_stanza[pos_oggetto] := val_oggetto
```

leggere_oggetto_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza da cui preleviamo l'oggetto in base alla sua posizione	stanza	
pos_oggetto	Indice dell'oggetto da prelevare nella stanza	intero	>=0

Nome	Descrizione	Tipo	Vincoli
val_oggetto	Oggetto prelevato dalla stanza	oggetto	

```
Val_oggetto := stanza_mappa.lista_oggetti_stanza[pos_oggetto]
```

scrivere_oggetto_contenuto

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza in cui aggiungiamo l'oggetto contenuto	stanza	
pos_oggetto	Indice dell'oggetto contenuto della stanza	intero	>=0
val_oggetto	Oggetto contenuto da inserire nella stanza	oggetto	

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza in cui è stato aggiunto l'oggetto contenuto	stanza	

```
stanza_mappa.lista_oggetti_contenuti_stanza[pos_oggetto] := val_oggetto
```

leggere_oggetto_contenuto

Nome	Descrizione	Tipo	Vincoli
stanza_mappa	Stanza da cui preleviamo l'oggetto contenuto in base alla sua posizione	stanza	
pos_oggetto	Indice dell'oggetto contenuto da prelevare nella stanza	intero	>=0

Nome	Descrizione	Tipo	Vincoli
val_oggetto	Oggetto contenuto prelevato dalla stanza	oggetto	

```
Val_oggetto := stanza_mappa.lista_oggetti_contenuti[pos_oggetto]
```

scrivere_descrizione_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_gioco	Stanza in cui assegniamo la descrizione	stanza	
val_descrizione_stanza	Descrizione da assegnare a oggetto_gioco	array 1 dimensione caratteri (statico)	dim=350

Nome	Descrizione	Tipo	Vincoli
stanza_gioco	Stanza in cui è stata assegnata la descrizione	stanza	

```
Copiare_stringa(val_descrizione_stanza , stanza.descrizione_oggetto)
```

leggere_descrizione_stanza

Nome	Descrizione	Tipo	Vincoli
stanza_gioco	Stanza in cui preleviamo la descrizione	stanza	

Nome	Descrizione	Tipo	Vincoli
val_descrizione_stanza	Descrizione dell'oggetto assegnato a oggetto_gioco	array 1 dimensione caratteri (statico)	dim=350

```
copiare_stringa(stanza.descrizione_oggetto ,val_descrizione_stanza )
```

scrivere_id_comando

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui assegniamo un id	comando	
val_id_comando	ID che assegniamo il comando	intero	

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui è stato assegnato l'id	comando	

```
comando_gioco.id_comando := val_id_comando
```

leggere_id_comando

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando da cui preleviamo l'id	comando	

Nome	Descrizione	Tipo	Vincoli
val_id_comando	ID del comando	intero	

```
Val_id_comando := comando_gioco.id_comando
```

scrivere_regex_verbo

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui assegniamo l'espressione regolare che descrive il verbo (e i sinonimi)	comando	
val_regex_verbo	Espressione regolare che assegniamo al comando	array 1 dimensione caratteri (statico)	dim:350

Nome	Descrizione	Tipo	Vincoli

comando_gioco	Comando a cui è stato assegnato l'espressione regolare del verbo	comando	

```
Copiare_stringa(val_regex_verbo, comando_gioco.regex_verbo)
```

leggere_regex_verbo

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando da cui preleviamo l'espressione regolare del verbo	comando	

Nome	Descrizione	Tipo	Vincoli
val_regex_verbo	Espressione regolare del verbo relativa al comando	intero	

```
Copiare_stringa(val_regex_verbo , comando_gioco.regex_verbo)
```

scrivere_argomento_oggetto

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui assegniamo il flag che indica se l'argomento del comando deve essere un oggetto	comando	
val_flag	flag che indica se l'argomento del comando deve essere un oggetto	booleano	vero=oggetto falso=locuzione

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui è stato assegnato il flag	comando	

```
comando_gioco.argomento_oggetto := val_flag
```

leggere_argomento_oggetto

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando da cui preleviamo il flag che indica se l'argomento del comando deve essere un oggetto	comando	

Nome	Descrizione	Tipo	Vincoli
val_flag	flag che indica se l'argomento del comando deve essere un oggetto	booleano	vero=oggetto falso=locuzione

```
val_flag := comando_gioco.argomento_oggetto
```

scrivere_regex_argomento

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui assegniamo l'espressione regolare che descrive l'argomento	comando	
val_regex_argomento	Espressione regolare dell'argomento che assegniamo al comando	array 1 dimensione caratteri (statico)	dim:350 = 'x' se il comando deve avere come argomento il nome di un oggetto che ha attributo 'x' = locuzione se il comando non deve avere come argomento il nome di un oggetto

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui è stato assegnato l'espressione regolare dell'argomento	comando	

```
Copiare_stringa(val_regex_argomento, comando_gioco.regex_argomento)
```

leggere_regex_argomento

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando da cui preleviamo l'espressione regolare del argomento	comando	

Nome	Descrizione	Tipo	Vincoli
val_regex_argomento	Espressione regolare dell'argomento relativa al comando	intero	

```
Copiare_stringa(val_regex_argomento , comando_gioco.regex_argomento)
```

scrivere_argomento_oggetto_in_stanza

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui assegniamo il flag che indica se l'argomento del comando deve essere un oggetto nella stanza	comando	
val_flag	flag che indica se l'argomento del comando deve essere un oggetto nella stanza	booleano	vero=oggetto in stanza falso=oggetto non in stanza

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui è stato assegnato il flag	comando	

```
comando_gioco.argomento_oggetto_in_stanza := val_flag
```

leggere_argomento_oggetto_in_stanza

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando da cui preleviamo il flag che indica se l'argomento del comando deve essere un oggetto	comando	

Nome	Descrizione	Tipo	Vincoli
val_flag	flag che indica se l'argomento del comando deve essere un oggetto nella stanza	booleano	vero=oggetto in stanza falso=oggetto non in stanza

```
val_flag := comando_gioco.argomento_oggetto_in_stanza
```

scrivere_argomento_oggetto_in_inventario

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui assegniamo il flag che indica se l'argomento del comando deve essere un oggetto nell' inventario	comando	
val_flag	flag che indica se l'argomento del comando deve essere un oggetto nell' inventario	booleano	vero=oggetto in inventario falso=oggetto non in inventario

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando a cui è stato assegnato il flag	comando	

```
comando_gioco.argomento_oggetto_in_inventario := val_flag
```

leggere_argomento_oggetto_in_inventario

Nome	Descrizione	Tipo	Vincoli
comando_gioco	Comando da cui preleviamo il flag che indica se l'argomento del comando deve essere un oggetto nell'inventario	comando	

Nome	Descrizione	Tipo	Vincoli
val_flag	flag che indica se l'argomento del comando deve essere un oggetto nell' inventario	booleano	vero=oggetto in inventario falso=oggetto non in invent.

```
val_flag := comando_gioco.argomento_oggetto_in_inventario
```