COSTANTI GLOBALI

NOME	DESCRIZIONE	VALORE
MAX_SLOTS	Numero di slot massimi per ciascun elenco di oggetti	10
MAX_HEALTH	Numero intero di punti vita massimi per ciascuna entità (nemico o personaggio)	100
MAX_DIREZIONI	Numero di direzioni massime possibili per ciascuna stanza	6
MAX_PAROLE	Numero di parole massime per ciascuna tabella dei simboli	50
MAX_CARATT	Numero intero di caratteri massimo per ciascuna sequenza di caratteri	30

VARIABILI GLOBALI

NOME	DESCRIZIONE	TIPO	VINCOLO
keywords	elenco delle parole chiave dell'automa	tabella_simboli	MAX_PAROLE
ident	tabella dei simboli della classificazione dell'input dell'utente	tabella_simboli	MAX_KEYWORD
james	personaggio principale	personaggio	
stanza_attuale	stanza in cui si trova il personaggio	ambiente	

Per le funzioni principali vai a pagina 25

FUNZIONI DI ACCESSO TIPO DI DATO: arma

set_danno_arma

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma di cui si vuol impostare il danno che infligge	arma	
danno	danno da assegnare all'arma	intero	danno > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma con danno aggiornato	arma	

ALGORITMO

weapon.danno := danno

get_danno_arma

INPUT			
nome	**descrizione**	**tipo**	**vincolo**

weapon	arma di cui si vuol conoscere il danno	arma	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
danno	danno che infligge l'arma	intero	

ALGORITMO

danno := weapon.danno

set_durezza_arma

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma di cui si vuol impostare la durezza	arma	
durezza	durezza da assegnare all'arma	intero	durezza > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma con durezza aggiornata	arma	

ALGORITMO

weapon.durezza := durezza

get_durezza_arma

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma di cui si vuol conoscere la durezza	arma	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
durezza	durezza dell'arma	intero	

ALGORITMO

durezza := weapon.durezza

FUNZIONI DI ACCESSO TIPO DI DATO: cibo

set_punti_vita_cibo

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
food	cibo di cui si vogliono impostare i punti vita	cibo	
hp	punti vita da assegnare al cibo	intero	hp > 0

OUTPUT

nome	descrizione	tipo	vincolo
food	cibo con punti vita aggiornati	cibo	

ALGORITMO

food.punti_vita := hp

get_punti_vita_cibo

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
food	cibo di cui si vuol conoscere i punti vita	cibo	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
hp	punti vita del cibo	intero	hp > 0

ALGORITMO

hp := food.punti_vita

FUNZIONI DI ACCESSO TIPO DI DATO: oggetto_apribile

set_apertura_apribile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto apribile di cui si vuole impostare l'apertura	oggetto_apribile	
apertura	apertura da assegnare alla porta; ha valore VERO se l'oggetto è aperto FALSO altrimenti	booleano	VERO o FALSO

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto apribile con apertura aggiornata	oggetto_apribile	

ALGORITMO

obj.apertura := apertura

get_apertura_apribile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto apribile di cui si vuol sapere l'apertura	oggetto_apribile	

OUTPUT

nome	descrizione	tipo	vincolo
apertura	apertura dell'oggetto apribile; VERO se aperto FALSO altrimenti	booleano	VERO o FALSO

ALGORITMO

apertura := obj.apertura

set_id_chiave_da_usare

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto apribile in cui si vuol aggiornare l'id della chiave da usare	oggetto_apribile	
id_key	id della chiave da assegnare all'oggetto apribile	intero	id_key > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto apribile con id chiave aggiornato	oggetto_apribile	

ALGORITMO

obj.id_chiave := id_key

get_id_chiave_da_usare

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto apribile di cui si vuol sapere l'id della chiave da usare	oggetto_apribile	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
id_key	id della chiave da usare	intero	id_key > 0

ALGORITMO

id_key := obj.id_chiave

FUNZIONI DI ACCESSO TIPO DI DATO: oggetto

set_id_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui impostare l'id	oggetto	

number	numero dell'id da assegnare all'oggetto	intero	number > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto con campo id aggiornato	oggetto	

ALGORITMO

obj.id := number

get_id_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuol sapere l'id	oggetto	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
id	id dell'oggetto	intero	id > 0

ALGORITMO

id := obj.id

set_nome_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuole impostare il nome	oggetto	
nome	nome da assegnare all'oggetto	array di caratteri	MAX_CARATT

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto con campo nome aggiornato	oggetto	

ALGORITMO

obj.nome := nome

get_nome_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuol sapere il nome	oggetto	

OUTPUT

nome	descrizione	tipo	vincolo
nome	nome dell'oggetto	array di caratteri	MAX_CARATT

ALGORITMO

nome := obj.nome

set_afferrabile_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuole impostare l'afferrabilità	oggetto	
aff	valore di afferrabilità dell'oggetto; VERO se l'oggetto è afferrabile FALSO altrimenti	booleano	VERO o FALSO
OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto con variabile afferrabile aggiornata	oggetto	

ALGORITMO

obj.afferrabile := aff

get_afferrabile_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuol sapere l'afferrabilità	oggetto	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
aff	afferrabilità dell'oggetto; VERO se l'oggetto è afferrabile FALSO altrimenti	booleano	VERO o FALSO

ALGORITMO

aff := obj.afferrabile

set_tipo_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuole impostare il tipo	oggetto	
type	lettera da assegnare all'oggetto	carattere	'c' = cibo 'a' = arma 'k' = oggetto_apribile

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**

obj	oggetto con campo tipo aggiornato	oggetto	

ALGORITMO

obj.tipo := type

get_tipo_oggetto

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto di cui si vuol sapere il tipo	oggetto	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
type	tipo dell'oggetto	carattere	

ALGORITMO

type := obj.tipo

FUNZIONI DI ACCESSO TIPO DI DATO: oggetto_guardabile

set_nome_oggetto_guardabile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile di cui impostare il nome	oggetto_guardabile	
element	nome da assegnare all'oggetto guardabile	array di caratteri	MAX_CARATT

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile con nome aggiornato	oggetto_guardabile	

ALGORITMO

obj_look.nome := element

get_nome_oggetto_guardabile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile di cui si vuol sapere il nome	oggetto_guardabile	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
element	nome dell'oggetto guardabile	array di caratteri	MAX_CARATT

ALGORITMO

element := obj_look.nome

set_id_oggetto_guardabile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile di cui si vuole impostare l'id	oggetto_guardabile	
pos	posizione nell'elenco degli id dell'oggetto guardabile	intero	pos ≥ 0
element	id dell'oggetto guardabile	intero	element > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile con id aggiornato	oggetto_guardabile	

ALGORITMO

obj_look.elenco_id[pos] := element

get_id_oggetto_guardabile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile di cui si vuol sapere l'id	oggetto_guardabile	
pos	posizione nell'elenco degli id degli oggetti guardabili	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
element	id dell'oggetto guardabile	oggetto_guardabile	

ALGORITMO

element := obj_look.elenco_id[pos]

set_numero_id

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile di cui si vuole impostare il numero di oggetti che è possibile trovare guardandolo	oggetto_guardabile	

number	numero degli oggetti che è possibile trovare	intero	number > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile con numero id aggiornati	oggetto_guardabile	

ALGORITMO

obj_look.number_id := number

get_numero_id

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile di cui si vuol sapere il numero di oggetti che è possibile trovare guardandolo	oggetto_guardabile	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
number	numero degli oggetti che è possibile trovare guardando l'oggetto in input	intero	number > 0

ALGORITMO

number := obj_look.number_id

FUNZIONI DI ACCESSO TIPO DI DATO: direzione

set_id_destinazione

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuole impostare l'id destinazione	direzione	
dest	id destinazione da assegnare	intero	dest > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**

dir	direzione con id destinazione aggiornato	direzione	

ALGORITMO

dir.id_destinazione := dest

get_id_destinazione

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuol sapere l'id destinazione	direzione	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
dest	id destinazione	intero	dest > 0

ALGORITMO

dest := dir.id_destinazione

set_punto_cardinale

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuol impostare il punto cardinale	direzione	
pc	punto cardinale da assegnare	carattere	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione con punto cardinale aggiornato	direzione	

ALGORITMO

dir.punto_cardinale := pc

get_punto_cardinale

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuol sapere il punto cardinale	direzione	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**

pc	punto cardinale della direzione	carattere	

ALGORITMO

pc := dir.punto_cardinale

set_apertura_direzione

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuole aggiornare l'apertura	direzione	
apertura	valore da assegnare alla direzione; VERO se è possibile andare FALSO altrimenti	booleano	VERO o FALSO

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione con apertura aggiornata	direzione	

ALGORITMO

dir.apertura := apertura

get_apertura_direzione

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuole sapere l'apertura	direzione	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
apertura	apertura della direzione; VERO se è possibile andare FALSO altrimenti	booleano	VERO o FALSO

ALGORITMO

apertura := dir.apertura

set_id_chiave_direzione

INPUT			
nome	**descrizione**	**tipo**	**vincolo**

dir	direzione di cui si vuole aggiornare l'id chiave	direzione	
chiave	id chiave da assegnare	intero	chiave > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione con id chiave aggiornato	direzione	

ALGORITMO

dir.id_chiave := chiave

get_id_chiave_direzione

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione di cui si vuole sapere l'id chiave	direzione	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
chiave	id chiave della direzione	intero	chiave > 0

ALGORITMO

chiave := dir.id_chiave

FUNZIONI DI ACCESSO TIPO DI DATO: ambiente

set_id_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole impostare l'id	ambiente	
id	id da assegnare alla stanza	intero	id > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con id aggiornato	ambiente	

ALGORITMO

stanza.id := id

get_id_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole sapere l'id	ambiente	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
id	id della stanza	intero	id > 0

ALGORITMO

id := stanza.id

set_descrizione_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole impostare la descrizione	ambiente	
desc	descrizione da assegnare alla stanza	array dinamico di caratteri	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con descrizione aggiornata	ambiente	

ALGORITMO

stanza.descrizione := desc

get_descrizione_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole sapere la descrizione	ambiente	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
desc	descrizione dell'ambiente	array dinamico di caratteri	

ALGORITMO

desc := stanza.descrizione

set_count_obj_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole impostare lil numero di oggetti presenti	ambiente	
number	intero che indica il numero di oggetti presenti in quella stanza	intero	number ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con count_obj aggiornato	ambiente	

ALGORITMO

stanza.count_obj := number

get_count_obj_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole sapere il numero di oggetti presenti	ambiente	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
number	intero che indica il numero di oggetti presenti	intero	number ≥ 0

ALGORITMO

number := stanza.count_obj

set_oggetto_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza dn cui si vuole aggiornare l'elenco oggetti	ambiente	
obj	oggetto da inserire nella stanza	oggetto	
pos	posizione nell'elenco oggetti della stanza	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con elenco oggetti aggiornato	ambiente	

ALGORITMO

stanza.elenco_oggetti[pos] := obj

get_oggetto_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuol sapere un oggetto nell'elenco degli oggetti	ambiente	
pos	posizione nell'elenco degli oggetti della stanza	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto nella stanza	oggetto	

ALGORITMO

obj := stanza.elenco_oggetti[pos]

set_count_dir_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole impostare il numero di direzioni possibili	ambiente	
number	intero che indica il numero di direzioni possibili	intero	number > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con count_dir aggiornato	ambiente	

ALGORITMO

stanza.count_dir := number

get_count_dir_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole sapere il numero di direzioni possibili	ambiente	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
number	intero che indica il numero di direzioni possibili	intero	number > 0

ALGORITMO

number := stanza.count_dir

set_direzione_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole aggiornare l'elenco delle direzioni possibili	ambiente	
pos	posizione nell'elenco delle direzioni della stanza	intero	pos ≥ 0
dir	direzione possibile da inserire nell'elenco direzioni	direzione	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con elenco direzioni aggiornato	ambiente	

ALGORITMO

stanza.direzioni_possibili[pos] := dir

get_direzione_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole sapere una direzione possibile	ambiente	
pos	posizione nell'elenco delle direzioni possibili	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
dir	direzione in cui è possibile andare	direzione	

ALGORITMO

dir := stanza.direzioni_possibili[pos]

set_count_look_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole impostare il numero di oggetti guardabili presenti	ambiente	
number	intero che indica il numero di oggetti guardabili presenti	intero	number ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza con count_look aggiornato	ambiente	

ALGORITMO

stanza.count_look := number

get_count_look_ambiente

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole sapere il numero di oggetti guardabili presenti	ambiente	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
number	intero che indica il numero di oggetti guardabili presenti nella stanza	intero	number ≥ 0

ALGORITMO

number := stanza.count_look

set_oggetto_guardabile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuole aggiornare l'elenco degli oggetti guardabili presenti nella stanza	ambiente	
pos	posizione nell'elenco degli oggetti guardabili	intero	pos ≥ 0
obj_look	oggetto guardabile da inserire nell'elenco degli oggetti guardabili della stanza	oggetto_guardabile	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**

stanza	stanza con elenco degli oggetti guardabili aggiornato	ambiente	

ALGORITMO

stanza.looked[pos] := obj_look

get_oggetto_guardabile

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
stanza	stanza di cui si vuol sapere un oggetto guardabile	ambiente	
pos	posizione nell'elenco degli oggetti guardabili	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj_look	oggetto guardabile nella stanza	oggetto_guardabile	

ALGORITMO

obj_look := stanza.looked[pos]

FUNZIONI DI ACCESSO TIPO DI DATO: personaggio

set_posizione_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio di cui si vuole aggiornare la posizione	personaggio	
element	posizione nella mappa da assegnare al personaggio	intero	element > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio con posizione aggiornata	personaggio	

ALGORITMO

pg.posizione := element

get_posizione_personaggio

INPUT

nome	descrizione	tipo	vincolo
pg	personaggio di cui si vuol sapere la posizione nella mappa	personaggio	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
element	posizione nella mappa del personaggio	intero	pos > 0

ALGORITMO

element := pg.posizione

set_punti_vita_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio di cui si vogliono aggiornare i punti vita	personaggio	
hp	punti vita da assegnare al personaggio	intero	hp ≤ MAX_HEALTH

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio con punti vita aggiornati	personaggio	

ALGORITMO

pg.punti_vita := hp

get_punti_vita_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio di cui si vuol sapere i punti vita	personaggio	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
hp	punti vita del personaggio	intero	hp ≤ MAX_HEALTH

ALGORITMO

hp := pg.punti_vita

set_slot_inventario_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**

pg	personaggio di cui si vuole aggiornare l'inventario	personaggio	
pos	posizione nell'inventario	intero	pos ≥ 0
obj	oggetto da inserire nell'inventario	oggetto	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio con inventario aggiornato	personaggio	

ALGORITMO

pg.inventario[pos] := obj

get_slot_inventario_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio di cui si vuol sapere un oggetto dell'inventario	personaggio	
pos	posizione nell'inventario	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
obj	oggetto contenuto nell'inventario	oggetto	

ALGORITMO

obj := pg.inventario[pos]

set_weapon_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio di cui si vuole impostare l'arma che impugna	personaggio	
weapon	arma da assegnare al personaggio	oggetto	tipo_oggetto = 'a'

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio con arma aggiornata	personaggio	

ALGORITMO

pg.arma := weapon

get_weapon_personaggio

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
pg	personaggio di cui si vuol sapere l'arma che impugna	personaggio	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma che impugna il personaggio	oggetto	

ALGORITMO

weapon := pg.arma

FUNZIONI DI ACCESSO TIPO DI DATO: nemico

set_punti_vita_nemico

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
enemy	nemico di cui si vogliono aggiornare i punti vita	nemico	
hp	punti vita da assegnare al nemico	intero	hp ≤ MAX_HEALTH

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
enemy	nemico con punti vita aggiornati	nemico	

ALGORITMO

enemy.punti_vita := hp

get_punti_vita_nemico

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
enemy	nemico di cui si vuol sapere i punti vita	nemico	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
hp	punti vita del nemico	intero	hp ≤ MAX_HEALTH

ALGORITMO

hp := enemy.punti_vita

set_arma_nemico

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
enemy	nemico di cui si vuol impostare l'arma che impugna	nemico	
weapon	arma da assegnare al nemico	oggetto	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
enemy	nemico con arma aggiornata	nemico	

ALGORITMO

enemy.weapon := weapon

get_arma_nemico

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
enemy	nemico di cui si vuol sapere l'arma	nemico	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
weapon	arma del nemico	oggetto	

ALGORITMO

weapon := enemy.weapon

FUNZIONI DI ACCESSO TIPO DI DATO: tabella_simboli

set_parola_tab_simboli

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella dei simboli in cui si vuole inserire la parola	tabella_simboli	
pos	posizione nella tabella simboli	intero	pos ≥ 0
element	parola da inserire nella tabella simboli	array di caratteri	lunghezza massima = 20

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella simboli con parola inserita	tabella_simboli	

ALGORITMO

tab[pos].parola := element

get_parola_tab_simboli

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella dei simboli di cui si vuole sapere la parola	tabella_simboli	
pos	posizione nella tabella simboli	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
element	parola della tabella simboli	array di caratteri	lunghezza massima = 20

ALGORITMO

element := tab[pos].parola

set_simbolo_tab_simboli

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella dei simboli in cui si vuole inserire il simbolo	tabella_simboli	
pos	posizione nella tabella simboli	intero	pos ≥ 0
element	simbolo da inserire nella tabella simboli	carattere	lunghezza massima = 4

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella simboli con simbolo inserito	tabella_simboli	

ALGORITMO

tab[pos].simbolo := element

get_simbolo_tab_simboli

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella dei simboli di cui si vuole sapere il simbolo	tabella_simboli	
pos	posizione nella tabella simboli	intero	pos ≥ 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
element	simbolo nella tabella simboli	carattere	lunghezza massima = 4

ALGORITMO

element := tab[pos].simbolo

set_number_tab_simboli

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella dei simboli di cui si vuole inserire il numero di parole	tabella_simboli	
element	numero di parole da inserire nella tabella simboli	intero	element > 0

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella simboli con numero di parole aggiornato	tabella_simboli	

ALGORITMO

tab.number := element

get_number_tab_simboli

INPUT			
nome	**descrizione**	**tipo**	**vincolo**
tab	tabella dei simboli di cui si vuole sapere il numero di parole	tabella_simboli	

OUTPUT			
nome	**descrizione**	**tipo**	**vincolo**
element	numero di parole nella tabella simboli	intero	element > 0

ALGORITMO

element := tab.number

FUNZIONE principale

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
input	prossima mossa inserita in input dal giocatore	vettore dinamico di caratteri	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
result	variabile intera che indica se l'analisi del comando è andata a buon fine (0)	intero	result < 0

ALGORITMO

```
inizializzazione_parole_chiave()
inizializzazione_file()
inizializzazione_personaggio(james)
descrivere_ambiente(1)
MENTRE (get_punti_vita_personaggio(james) > 0 AND get_posizione_personaggio(james) != 16)
    leggere_in_input(input)
    ident := analizzare_comando(input)
    pg := eseguire_comando(ident, james)
    gestione_errore(result)
FINE
```

FUNZIONE muovere_personaggio

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stanza_attuale	stanza attuale in cui è presente il pg	ambiente	
direzione	direzione che prende il personaggio	carattere	
pg	personaggio da muovere	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stanza_attuale	stanza in cui si trova il giocare, aggiornata se la direzione era corretta e accessibile, altrimenti il personaggio rimane nella stanza in cui si trovava	ambiente	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	count parte da 0
esito	variabile intera che indica se il movimento del personaggio è andato a buon fine (0)	numero intero	

ALGORITMO

```
count := 0
esito := -3
MENTRE(count < get_count_dir_ambiente(stanza_attuale))
    SE(direzione = get_punto_cardinale_direzione(get_direzione_ambiente(stanza_attuale, count)))
        ALLORA
        SE(get_apertura_direzione(get_direzione_ambiente(stanza_attuale, count)) = TRUE)
            ALLORA
            esito := 0;
            set_posizione_personaggio(pg, get_id_destinazione(get_direzione_ambiente(stanza_attuale,
count)))
            ALTRIMENTI
            esito := -11;
        FINE
    FINE
    count := count + 1;
FINE
SE(esito = 0)
    ALLORA
        stanza_attuale := descrivere_ambiente(get_posizione_personaggio(pg));
    ALTRIMENTI
        gestione_errore(esito);
FINE
```

FUNZIONE descrivere_ambiente

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
room_number	numero di stanza di cui prelevare tutte le informazioni necessarie	numero intero	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
trovato	variabile che indica che la stanza cercata è stata trovata	booleano	
pathname	pathname del file in cui è presente la storia	sequenza di caratteri	lunghezza massima MAX_CARATT
token	variabile in cui viene prelevata una parola per volta della riga prelevata dal file	sequenza di caratteri	lunghezza massima MAX_CARATT

prelievo	variabile di supporto in cui viene depositata una riga per volta del file	sequenza di caratteri	lunghezza massima MAX_CARATT
weapon_enemy	arma che deve impugnare il nemico contro cui combattere	arma	
temp_dir	variabile temporanea che contiene tutte le informazioni della direzione che stiamo prelevando dal file	direzione	
temp_obj	variabile temporanea che contiene tutte le informazioni dell'oggetto che stiamo prelevando	oggetto	
temp_look	variabile temporanea che contiene tutte le informazioni dell'oggetto guardabile che stiamo prelevando	oggetto_guardabile	
count	contatore	numero intero	parte da 0
hp_flag	variabile boolena che indica se abbiamo prelevato o meno i punti vita del nemico	booleana	
danno_flag	variabile boolena che indica se abbiamo prelevato o meno il danno dell'arma	booleana	
durezza_flag	variabile boolena che indica se abbiamo prelevato o meno la durezza dell'arma	booleana	
nome_flag	variabile boolena che indica se abbiamo prelevato o meno il nome dell'oggetto	booleana	
id_flag	variabile boolena che indica se abbiamo prelevato o meno l'id dell'oggetto	booleana	
aff_flag	variabile boolena che indica se abbiamo prelevato o meno l'afferrabilità dell'oggetto	booleana	
tipo_flag	variabile boolena che indica se abbiamo prelevato o meno il tipo dell'oggetto	booleana	
par1_flag	variabile boolena che indica se abbiamo prelevato o meno il primo parametro dell'oggetto	booleana	
par2_flag	variabile boolena che indica se abbiamo prelevato o meno il secondo parametro dell'oggetto	booleana	
dir_flag	variabile boolena che indica se abbiamo prelevato o meno il carattere di direzione	booelana	
apertura_flag	variabile boolena che indica se abbiamo prelevato o meno se la direzione è aperta o meno	booleana	
id_chiave_flag	variabile boolena che indica se abbiamo prelevato o meno l'id della	boolena	

	chiave per aprire la porta della direzione desiderata		
lenght	variabile necessaria per calcolare l'offset del prelievo	numero intero	
file_storia	file in cui è presente la storia del gioco	FILE	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stanza_attuale	stanza in cui è presente il pg	ambiente	

ALGORITMO

```
trovato := false
pathname := “storia.txt”
SE((file_storia = aprire_file(pathname)) = TRUE)
    ALLORA
        MENTRE(fine_file(file_storia) = FALSE)
            prelievo:= copiare_stringhe(prelievo, prendere_riga_file(file_storia))
            SE(prelievo[0] = '^')
                ALLORA
                    token = scan(prelievo, 1)
                    SE(convert_sequence_char_to_int(token) == room_number)
                        ALLORA
                            stanza_attuale := set_id_ambiente(stanza_attuale, room_number);
                            trovato = true;
                    FINE
            FINE
            SE(trovato = true)
                ALLORA
                    stanza_attuale := inizializzazione_oggetti_ambienti(stanza_attuale);
                    stanza_attuale := set_count_obj_ambiente(stanza_attuale, 0);
                    stanza_attuale := set_count_dir_ambiente(stanza_attuale, 0);
                    stanza_attuale := set_count_look_ambiente(stanza_attuale, 0);
                    MENTRE(prelievo[0] != '_')
                        prelievo:= copiare_stringhe(prelievo, prendere_riga_file(file_storia))
                        SE(prelievo[0] != '/' AND prelievo[0] != '&' AND prelievo[0] != '$' AND prelievo[0] !=
'*' AND prelievo[0] != '%' AND prelievo[0] != '_')
                            ALLORA stanza_attuale := set_descrizione_ambiente(stanza_attuale, prelievo)
```

```
                    ALTRIMENTI
                        SE(prelievo[0] = '&')
                            ALLORA
                                count := 1
                                hp_flag := false
                                danno_flag := false
                                durezza_flag := false
                                MENTRE(prelievo[count] != '\0')
                                    token = scan(prelievo, count)
                                    number = convert_sequence_char_to_int(token)
                                    SE(hp_flag = false)
                                        ALLORA
                                            set_punti_vita_nemico(enemy, number)
                                            hp_flag := true;
                                        ALTRIMENTI
                                            SE(get_punti_vita_nemico(enemy) > 0)
                                                ALLORA
                                                    SE(danno_flag = false)
                                                        ALLORA
                                                            weapon_enemy :=
set_danno_arma(weapon_enemy, number)
                                                            danno_flag := true
                                                        ALTRIMENTI
                                                            SE(durezza_flag = false)
                                                                ALLORA
                                                                weapon_enemy :=
set_durezza_arma(weapon_enemy, number);
                                                                durezza_flag := true
                                                            FINE
                                                    FINE
                                                    SE(danno_flag = true AND durezza_flag = true)
                                                        enemy := set_arma_nemico(enemy, weapon_enemy)
                                                        prelievo:= copiare_stringhe(prelievo,
prendere_riga_file(file_storia))
                                                        SE(prelievo[0] = '$')
                                                            ALLORA
```

```
                                                    stanza_attuale :=
set_descrizione_ambiente(stanza_attuale, prelievo)
                                                    combattimento(jack, enemy);
                                                FINE
                                            FINE
                                        FINE
                                        lenght := 0 + lunghezza_stringa(token)
                                        count := count + 1 + lenght
                                    FINE
                                FINE
                            ALTRIMENTI
                                SE(prelievo[0] = '/')
                                    ALLORA
                                        count := 1
                                        temp_look := set_numero_id_guardabili(temp_look, 0)
                                        nome_flag := false
                                        MENTRE(prelievo[count] != '\0')
                                            token := scan(prelievo, count)
                                            SE(nome_flag = false)
                                                ALLORA
                                                    temp_look := set_nome_oggetto_guardabile(temp_look,
token)
                                                    nome_flag := true
                                                ALTRIMENTI
                                                    temp_look := set_id_oggetto_guardabile(temp_look,
get_numero_id_guardabili(temp_look), convert_sequence_char_to_int(token));
                                                    temp_look := set_numero_id_guardabili(temp_look,
get_numero_id_guardabili(temp_look) + 1);
                                            FINE
                                            lenght := 0 + lunghezza_stringa(token);
                                            count := count + 1 + lenght;
                                        FINE
                                        stanza_attuale := set_oggetto_guardabile(stanza_attuale,
get_count_look_ambiente(stanza_attuale), temp_look)
                                        stanza_attuale := set_count_look_ambiente(stanza_attuale,
get_count_look_ambiente(stanza_attuale), 1)
                                    ALTRIMENTI
```

```
SE(prelievo[0] = '*')
    ALLORA
        id_flag := false
        nome_flag := false
        aff_flag := false
        tipo_flag := false
        par1_flag := false
        par2_flag := false
        count := 1
        MENTRE(prelievo[count] != '\0')
            token := scan(prelievo, count)
            SE(id_flag = false)
                ALLORA
                    temp_obj := set_id_oggetto(temp_obj,
convert_sequence_char_to_int(tokne))
                    id_flag := true
                ALTRIMENTI
                    SE(nome_flag = false)
                        ALLORA
                            temp_obj := set_nome_oggetto(temp_obj,
token)
                            nome_flag := true
                        ALTRIMENTI
                            SE(aff_flag = true)
                                ALLORA
                                    SE(confronto_stringhe(token, "true") =
true)
                                        ALLORA
                                            temp_obj :=
set_affettabile_oggetto(temp_obj, true)
                                        ALTRIMENTI
                                            temp_obj :=
set_affettabile_oggetto(temp_obj, false)
                                    FINE
                                    aff_flag := true
                                ALTRIMENTI
                                    SE(tipo_flag = false)
```

ALLORA

SE(confronto_stringhe(token, "a") = true)

ALLORA

temp_obj := set_tipo_oggetto(temp_obj, 'a')

ALTRIMENTI

SE(confronto_stringhe(token, "c") = true)

ALLORA

temp_obj := set_tipo_oggetto(temp_obj, 'c')

ALTRIMENTI

SE(confronto_stringhe(token, "k") = true)

ALLORA temp_obj := set_tipo_oggetto(temp_obj, 'k')

FINE

FINE

FINE

tipo_flag := true

ALTRIMENTI

SE(par1_flag = false)

ALLORA

SE(get_tipo_oggetto(temp_obj) = 'a')

ALLORA

temp_obj.oggetto_speciale.weapon := set_danno_arma(temp_obj.oggetto_speciale.weapon, convert_sequence_char_to_int(token))

ALTRIMENTI

SE(get_tipo_oggetto(temp_obj) = 'c')

ALLORA

temp_obj.oggetto_speciale.food := set_punti_vita_cibo(temp_obj.oggetto_speciale.food, convert_sequence_char_to_int(token))

ALTRIMENTI

SE(get_tipo_oggetto(temp_obj) = 'k')

ALLORA

SE(confronto_stringhe(token, "true") = true)

ALLORA

temp_obj.oggetto.speciale.key := set_apertura_apribile(temp_obj.oggetto.speciale.key, true)

ALTRIMENTI

temp_obj.oggetto.speciale.key := set_apertura_apribile(temp_obj.oggetto.speciale.key, false)

FINE

FINE

FINE

FINE

par1_flag := true

ALTRIMENTI

SE(par2_flag = false)

ALLORA

SE(get_tipo_oggetto(temp_obj) = 'a')

ALLORA

temp_obj.oggetto_speciale.weapon := set_durezza_arma(temp_obj.oggetto_speciale.weapon, convert_sequence_char_to_int(token))

ALTRIMENTI

SE(get_tipo_oggettp(temp_obj) = 'k')

ALLORA

temp_obj.oggetto_speciale.key := set_id_chiave_da_usare(temp_obj.oggetto_speciale.key, convert_sequence_char_to_int(token))

FINE

FINE

par2_flag := true

ALTRIMENTI

FINE

lenght := 0 + lunghezza_stringa(token)

count := count + 1 + lenght

```
                                                                FINE
                                                                stanza_attuale :=
set_oggetto_ambiente(stanza_attuale, get_count_obj_ambiente(stanza_attuale), temp_obj)
                                                                stanza_attuale :=
set_count_obj_ambiente(stanza_attuale, get_count_obj_ambiente(stanza_attuale) + 1)
                                                            FINE
                                                        FINE
                                                    FINE
                                                FINE
                                            FINE
                                        ALTRIMENTI
                                            SE(prelievo[0] = '%')
                                                ALLORA
                                                    id_flag := false
                                                    dir_flag := false
                                                    apertura_flag := false
                                                    id_chiave_flag := false
                                                    count := 1
                                                    MENTRE(prelievo[count] != '\0')
                                                        token := scan(prelievo, count)
                                                        SE(if_flag = false)
                                                            ALLORA
                                                                temp_dir := set_id_destinazione(temp_dir,
convert_sequence_char_to_int, (token))
                                                                id_flag := true
                                                            ALTRIMENTI
                                                                SE(dir_flag = false)
                                                                    ALLORA
                                                                        temp_dir := set_punto_carinale(temp_dir,
token);
                                                                        dir_flag := true
                                                                    ALTRIMENTI
                                                                        SE(apertura_flag = false)
                                                                            ALLORA
                                                                                SE(confronto_stringhe(token,
"true") = true)
                                                                                    ALLORA
```

temp_dir := set_apertura_direzione(temp_dir, true);

ALTRIMENTI

temp_dir := set_apertura_direzione(temp_dir, false);

FINE

apertura_flag := true

ALTRIMENTI

SE(id_chiave_flag = false)

ALLORA

temp_dir := set_id_chiave_direzione(temp, convert_sequence_char_to_int(token))

id_chiave_flag = true

FINE

FINE

FINE

FINE

FINE

lenght := 0 + lunghezza_stringa(token)

count := count + 1 + lenght

FINE

stanza_attuale := set_direzione_ambiente(stanza_attuale, get_count_dir_ambiente(stanza_attuale), temp_dir)

stanza_attuale := set_count_dir_ambiente(stanza_attuale, get_count_dir_ambiente(stanza_attuale) + 1)

FINE

FINE

FINE

FINE

FINE

FINE

FINE

chiudi_file(file_storia)

FINE

FUNZIONE inizializzazione_oggetti_ambiente

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stanza	stanza di cui inizializzare tutti gli oggetti presenti, cioè rendere tuota la stanza	ambiente	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stanza	stanza priva di oggetti	ambiente	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	count deve partire da 0
empty	oggetto vuoto necessario per impostare ogni slot della stanza a vuoto	oggetto	

ALGORITMO

empty := set_id_oggetto(empty, 0)

empty := set_nome_oggetto(empty, "empty")

empty := set_afferrabile_oggetto(empty, false)

count := 0

stanza := set_count_obj_ambiente(stanza, 0)

MENTRE(count < MAX_SLOTS)

    stanza := set_oggetto_ambiente(stanza, count, empty)

    count := count + 1

FINE

FUNZIONE scan

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
frase	sequenza di caratteri da cui prelevare la prima parola a partire da offset	sequenza di caratteri	- lunghezza massima MAX_CARATT - le parole sono separate tra loro con ‘ ‘ (carattere blank)
offset	numero del carattere in frase da cui partire per definire la parola da prelevare	numero intero	offset < calcolare_lunghe

			zza_stringa(frase)

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
token	parola prelevata da frase	sequenza di caratteri	lunghezza massima MAX_CARATT

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	

ALGORITMO

```
count := 0
MENTRE (frase[count] != ‘ ’ AND frase[count] != ‘\n’ AND count <
            calcolare_lunghezza_stringa(frase))
    token[count] := input[offset]
    offset := offset + 1
    count := count + 1
FINE
```

FUNZIONE check_parola_chiave

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
token	parola di cui verificare se è una parola chiave o meno	sequenza di caratteri	lunghezza massima MAX_CARATT
keywords	tabella della parole chiavi	tabella_simboli	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
symbol	simbolo corrispondente alla parola chiave	sequenza di caratteri	lunghezza di MAX_SYM

DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	
keywords	tabella contenente le parole chiave	tabella_simboli	

ALGORITMO

```
symbol := “EMP”
count := 0
MENTRE(count < get_number_parole_chiavi(keywords))
    SE(verificare_uguaglianza_stringhe(token, get_parola_tab_simboli(keywords, count)) = true)
        ALLORA symbol := copiare_strighe(symbol, get_simbolo_tab_simboli(keywords, count))
    FINE
    count := count +1
FINE
```

FUNZIONE analizzare_comando

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
input	sequenza di caratteri data in input dall’utente da analizzare	sequenza di caratteri	lunghezza massima stringa MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
ident	tabella in cui sono presenti i comandi inseriti con i relativi simbolo	tabella_simboli	

DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	
token	variabile di supporto della parola prelevata sequenzialmente da input	sequenza di caratteri	lunghezz massima MAX_CARATT
stato_corrente	attuale stato per la verifica della correttezza della parola	numero intero	
stato_pozza	stato in cui si entra nel caso in cui la parola inserita sia una parola sbagliata	numero intero	stato_pozza := 2
count_1	variabile contatore di token	numero intero	

ALGORITMO

```
count := 0;
```

```
ident := set_number_tab_simboli(ident, 0)
MENTRE(input[count] != '\0')
    token := scan(input, count)
    count_1 := 0
    stato_corrente := 0
    stato_pozza := 2
    MENTRE(count_1 < lunghezza_stringa(token) AND stato_corrente != stato_pozza)
        SE(token[count_1] >= 'a' AND token[count_1] <= 'z')
            ALLORA
                SE(stato_corrente = 0 OR stato_corrente = 1)
                    ALLORA
                        stato_corrente := 1
                FINE
            ALTRIMENTI stato_corente := stato_pozza
        FINE
        count_1 := count_1 + 1
    FINE
    SE(stato_corrente != stato_pozza)
        ALLORA
            ident := set_parola_tab_simboli(ident, get_number_tab_simboli(ident), token)
            SE(confronto_stringhe(check_parola_chiave(token), "EMP") = false)
                ALLORA
                    ident := set_simbolo_tab_simboli(ident, get_number_tab_simboli(ident),
check_parola_chiave(token))
                ALTRIMENTI
                    ident := set_simbolo_tab_simboli(ident, get_number_tab_simboli(ident), "OBJ")
            FINE
        ALTRIMENTI
            ident := set_parola_tab_simboli(ident, get_number_tab_simboli(ident), "0")
            ident := set_simbolo_tab_simboli(ident, get_number_tab_simboli(ident), "0")
    FINE
    ident := set_number_tab_simboli(idend, get_number_tab_simboli(ident) + 1)
FINE
```

FUNZIONE prendere_oggetto

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore che prende un oggetto	personaggio	
name_object	nome dell'oggetto da prendere	array di caratteri	lunghezza massima MAX_CARATT
stanza	stanza da cui prendere l'oggetto (se è presente ed è possibile prenderlo)	ambiente	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore con inventario aggiornato	personaggio	

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pos	posizione dell'inventario in cui inserire l'oggetto	intero	pos ≥-1
afferrabile	indica se l'oggetto in input è afferrabile o meno	booleano	VERO o FALSO
trovato	indica se l'oggetto è stato trovato o meno	booleano	VERO o FALSO
i	contatore del ciclo e indice	intero	i ≥ 0
object	variabile di supporto che contiene l'oggetto da prendere	oggetto	
esito	variabile che indica se ci sono stati errori	numero intero	

ALGORITMO

```
count := 0
esito := 0
afferrabile := false
trovato := false

dir := set_id_destinazione(dir, 16)
dir := set_punto_cardinale(dir, 'b')

MENTRE(count < get_count_obj_ambiente(stanza) AND trovato = true)
    SE(confronto_stringhe(name_object, get_nome_oggetto(get_oggetto_ambiente(stanza, count)))
= true)
        ALLORA
        object := get_oggetto_ambiente(stanza, count)
        afferrabile := get_afferrabile_oggetto(object)
        trovato := true
    FINE
    count := count + 1
FINE

pos := controllare_inventario(pg)
SE(pos != -1 AND afferrabile = true)
    ALLORA
        pg := set_slot_inventario(pg, pos, object)
        remove_object_from_file(get_id_oggetto(object))

        SE(confronto_stringhe(name_object, "libro") = true AND get_posizione_personaggio(pg) = 15)
            ALLORA
                open_door_from_file(dir, get_posizione_personaggio(pg))
```

```
        FINE
    ALTRIMENTI
        SE(pos = -1)
            ALLORA esito = -8
            ALTRIMENTI esito = -7
        FINE
FINE
gestione_errore(esito)
```

FUNZIONE controllare_inventario

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore di cui controllare se l'inventario è pieno	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pos	prima posizione vuota dell'inventario	intero	pos ≥ -1

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
i	contatore del ciclo e indice dell'inventario	intero	i ≥ 0

ALGORITMO

```
i := 0
pos := -1
MENTRE (i < MAX_SLOTS AND pos = -1)
    SE (verificare_uguaglianza_stringhe(get_nome_oggetto(pg.inventario[i]), “vuoto”) = VERO)
        ALLORA
            pos := i
    FINE
    i := i+1
FINE
```

FUNZIONE verificare_uguaglianza_stringhe

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stringa1	prima stringa da confrontare	vettore di caratteri	lunghezza massima MAX_CARATT

stringa2	seconda stringa da confrontare	vettore di caratteri	lunghezza massima MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
esito	indica se le due stringhe confrontate sono uguali tra loro	booleano	VERO o FALSO

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
i	contatore del ciclo e indice del vettore	intero	i ≥ 0

ALGORITMO

```
i := 0
lunghezza := calcolare_lunghezza_stringa(stringa1)
esito := VERO
MENTRE (i < lunghezza AND esito = VERO)
    SE (stringa1[i] != stringa2[i])
        ALLORA
            esito := FALSO
    FINE
FINE
```

FUNZIONE calcolare_lunghezza_stringa

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stringa	stringa di cui calcolare la lunghezza	vettore di caratteri	lunghezza massima MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
i	lunghezza della stringa	intero	i ≥ 0

ALGORITMO

```
i:=0
MENTRE (stringa[i] != '\0')
        i := i+1
FINE
```

FUNZIONE lasciare_oggetto

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore	personaggio	
nome_oggetto	nome dell'oggetto da lasciare	vettore di caratteri	lunghezza massima MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore con inventario aggiornato	personaggio	

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
i	contatore del ciclo e indice dell'inventario	intero	i ≥ 0
trovato	indica se l'oggetto cercato è stato trovato	booleano	VERO o FALSO
empty	oggetto vuoto	oggetto	
esito	variabile che indica se ci sonos stati errori	numero intero	

ALGORITMO

```
i := 0
esito := 0
trovato := false
empty := set_id_oggetto(empty, 0)
empty := set_nome_oggetto(empty, "vuoto")
empty := set_afferrabile_oggetto(empty, false)

MENTRE(i < MAX_SLOTS AND trovato = false)
    SE(confronto_stringhe(get_nome_oggetto(get_slot_inventario_personaggio(pg, i)),
nome_oggetto) = true)
        ALLORA
            insert_object_into_file(get_slot_inventario_personaggio(pg, i),
get_posizione_oggetto_personaggio(pg))
            pg := set_slot_inventario_personaggio(pg, i, empty)
            pg := ordinare_inventario(pg)
            trovato := true
    FINE
    i := i + 1
FINE
SE(trovato = false)
    ALLORA esito := -6
FINE
gestione_errore(esito)
```

FUNZIONE guardare_oggetto

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
stanza	stanza da cui prelevare gli oggetti che sono stati trovati	ambiente	
nome_object	nome dell'oggetto che si vuole osservare	sequenza di caratteri	lunghezza massima MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
//vengono stampati i nomi degli oggetti che vengono trovati			

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	
count_1	variabile contatore	numero intero	
count_2	variabile contatore	numero intero	
esito	variabile che indica se ci sono stati errori	numero intero	
id_oggetto	varibile di supporto che contiene l'id dell'oggetto trovato	numero intero	

ALGORITMO

```
esito := -7
count := 0
SE(get_count_look_ambiente(stanza) > 0)
    ALLORA
      MENTRE(count < get_count_look_ambiente(stanza))
         SE(confronto_stringhe(nome_object,
get_nome_oggetto_guardabile(get_oggetto_guardabile(stanza, count))) = true)
             ALLORA
                esito := 0
                count_1 := 0
                MENTRE(count_1 < get_id_number_guardabile(get_oggetto_guardabile(stanza,
count)))
                   id_oggetto := get_id_oggetto_guardabile(get_oggetto_guardabile(stanza, count),
count_1)

                   count_2 := 0
                   MENTRE(count_2 < get_count_obj_ambiente(stanza))
                      SE(id_oggetto = get_id_oggetto(get_oggetto_ambiente(stanza, count_2)))
                         ALLORA stampa_video(get_nome_oggetto(get_oggetto_ambiente(stanza,
count_2))
                      FINE
```

```
                count_2 := count_2 + 1
            FINE
            count_1 := count_1 + 1
        FINE
    FINE
    count := count + 1
  FINE
 ALTRIMENTI esito := -6
FINE
gestione_errore(esito)
```

FUNZIONE ordinare_inventario

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio di cui ordinare l'inventario, cioè senza "buchi" di oggetti vuoti in mezzo	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio con inventario ordinato	personaggio	

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	variabile contatore	numero intero	
count_1	variabile conatore	numero intero	
empty	oggetto vuoto da sostituire nell'inventario del personaggio	oggetto	

ALGORITMO

```
empty := set_id_oggetto(empty, 0)
empty := set_nome_oggetto(empty, "vuoto")
empty := set_afferrabile_oggetto(empty, false)
count := 0
MENTRE(count < MAX_SLOTS)
 SE(get_id_oggetto(get_slot_inventario_personaggio(pg, count)) != 0)
  ALLORA
   count_1 := count + 1
   MENTRE(count_1 < MAX_SLOTS)
    SE(get_id_oggetto(get_slot_inventario_personaggio(pg, count_1)) != 0)
     ALLORA
      pg := set_slot_inventario_personaggio(pg, count,
get_slot_inventario_personaggio(pg, count_1))
      pg := set_slot_inventario_personaggio(pg, count_1, empty)
      count_1 := MAX_SLOTS
```

```
            FINE
            count_1 := count_1 + 1
        FINE
    count := count + 1
FINE
FINE
```

FUNZIONE aprire_oggetto

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore di cui verificare se possiede nell'inventario la chiave necessaria per aprire l'oggetto	personaggio	
name_object	nome dell'oggetto da aprire	vettore di caratteri	lunghezza massima MAX_CARATT
stanza	stanza in cui è presente l'oggetto da aprire	ambiente	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore con inventario aggiornato	personaggio	

DATI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
temp	variabile temporanea in cui è presente l'oggetto da aprire	oggetto	
trovato	indica se l'oggetto è stato trovato e se è aperto o meno	booleano	VERO o FALSO
i	contatore del ciclo e indice dell'inventario	intero	i ≥ 0
j	contatore del ciclo e indice dell'inventario	intero	i ≥ 0
empty	oggetto vuoto da sostituire nell'inventario del personaggio	oggetto	
esito	variabile che indica se ci sono stati errori	numero intero	

ALGORITMO

```
trovato := false
i := 0
esito := 0
empty := set_id_oggetto(empty, 0)
empty := set_nome_oggetto(empty, "vuoto")
```

```
empty := set_afferrabile_oggetto(empty, false)
MENTRE(i < get_count_obj_ambiente(stanza) AND trovato = false)
    SE(confronto_stringhe(get_nome_oggetto(get_oggetto_ambiente(stanza, i)), name_object) =
true)
        ALLORA
        temp := get_oggetto_ambiente(stanza, i)
        SE(get_tipo_oggetto(temp) = 'k')
            ALLORA
                SE(get_apertura_apribile(temp.oggetto_speciale.key) = false)
                    ALLORA
                        j := 0
                        MENTRE(j < MAX_SLOTS AND trovato = false)
                            SE(get_id_oggetto(get_slot_inventario_personaggio(pg, j)) =
get_id_chiave_da_usare(temp.oggetto_speciale.key))
                                ALLORA
                                    temp := set_apertura_apribile(temp.oggetto_speciale.key, true)
                                    pg := set_slot_inventario(pg, j, empty)
                                    trovato := true
                                FINE
                                j := j + 1
                            FINE
                    ALTRIMENTI esito := -13
                    SE(trovato = false)
                        ALLORA esito := -12
                    FINE
                FINE
            ALTRIMENTI esito := -14
        FINE
        i := i + 1
    FINE
FINE
gestione_errore(esito)
```

FUNZIONE aprire_porta

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore che vuole aprire la porta,di cui controllare se ha la chiave necessaria	personaggio	
direction	punto cardinale della porta che si vuole aprire	carattere	lunghezza massima MAX_CARATT
stanza	stanza in cui è presente la porta da aprire	ambiente	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore con inventario aggiornato	personaggio	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
esito	variabile che indica se ci sono stati errori o meno	numero intero	
i	contatore	numero intero	
j	contatore	numero intero	
trovato_1	variabile booleana che indica se la direzione desiderata è consentita	booleana	
trovato_2	variabile che indica se abbiamo l'oggetto necessario (se necessario) per aprire la porta desiderata	booleana	

ALGORITMO

```
esito := 0
i := 0
trovato_1 := false
trovato_2 := false

MENTRE(i < get_count_dir_ambiente(stanza) AND trovato_1 = false)
    dir := get_direzione_ambiente(stanza, i)

    SE(get_punto_carinale(dir) = point)
        ALLORA
            trovato_1 := true
            SE(get_apertura_direzione(dir) = false)
                ALLORA
                    SE(get_id_chiave_direzione(dir) = 0)
                        ALLORA
                            dir := set_apertura_direzione(dir, true)
                            open_door_from_file(dir, get_posisizione_personaggio(pg))
                            trovato_2 := true
                        ALTRIMENTI
                            j := 0
                            MENTRE(get_id_oggetto(get_slot_inventario_personaggio(pg, j)) != 0 AND
trovato_2 = false)
                                SE(get_id_oggetto(get_slot_inventario_personaggio(pg, i)) =
get_id_chiave_direzione(dir))
                                    ALLORA
                                        dir := set_apertura_direzione(dir, true)
                                        open_door_from_file(dir, get_posizione_personaggio(pg))
                                        trovato_2 := true
                                FINE
                                j := j + 1
```

```
                    FINE
                FINE
            ALTRIMENTI esito := -13
        FINE
    FINE
    stanza := set_direzione_ambiente(stanza, i, dir)
    i := i + 1
FINE
SE(esito = 0)
    ALLORA
        SE(trovato_1 = false)
            ALLORA esito := -3
            ALTRIMENTI
                SE(trovato_2 = false)
                    ALLORA esito := -12
                FINE
        FINE
FINE
gestione_errore(esito)
```

FUNZIONE impugnare_arma

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio che vuole impugnare un'arma presente nel suo inventario	personaggio	
nome_arma	nome dell'arma presente nell'inventario che il personaggio vuole impugnare	sequenza di caratteri	lunghezza MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio che impugna l'arma desiderata	personaggio	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
i	variabile contatore	numero intero	
esito	variabile che indica se ci sono stati errori o meno	numero intero	
trovato	variabile che indica se l'arma è presente nell'inventario	booleano	
empty	oggetto vuoto da inserire nello slot in cui era presente l'arma impugnata	oggetto	

ALGORITMO

```
i := 0
esito := 0
trovato := false
empty := set_id_oggetto(empty, 0)
empty := set_nome_oggetto(empty, "vuoto")
empty := set_afferrabile_oggetto(empty, false)

MENTRE(get_id_oggetto(get_slot_inventario_personaggio(pg, i)) != 0 AND trovato = false)
    SE(confronto_stringhe(get_nome_oggetto(get_slot_inventario_personaggio(pg, i)), nome_arma)
= true)
        ALLORA
            SE(get_tipo_oggetto(geto_slot_inventario_personaggio(pg, i)) = 'a')
                ALLORA
                    pg := set_weapon_personaggio(pg, get_slot_inventario_personaggio(pg, i))
                    pg := set_slot_inventario_personaggio(pg, i, empty)
                ALTRIMENTI esito := -10
            FINE
    FINE
    i := i + 1
FINE
SE(trovato = false)
    ALLORA esito := -6
FINE
gestione_errore(esito)
```

FUNZIONE combattimento

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio che combatte	personaggio	
enemy	nemico che combatte	nemico	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
esito	variabile che indica se è molto il personaggio o il nemico	numero intero	0 se è morto il nemico (cioè i suoi punti vita sono al di sotto dello zero) -1 se è morto il pg

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO

precisione	variabile che indica se attaccando si colpisce l'avversario	intero	-precisione >= 25 attacco con successo -precisione < 25 non si colpisce il bersaglio
risposta	risposta del personaggio	vettore di caratteri	lunghezza massima MAX_CARATT
arma_pg	arma usata dal personaggio	arma	
arma_enemy	arma usata dal nemico	arma	
casuale	numero casuale che indica la risposta del nemico (se attaccare o difendere)	numero intero	casuale ≥ 0 AND casuale ≤ 100

ALGORITMO

```
arma_pg := get_weapon_personaggio(pg).oggetto_speciale.weapon
arma_enemy := get_arma_nemico(enemy).oggetto_speciale.weapon
ESEGUI
    stampare_in_output("Mossa attacca/difendi")
    leggere_in_input(risposta)
    casuale := generare_numero_random(0, 100)
    SE (verificare_uguaglianza_stringhe(risposta, "attacca") = VERO)
        ALLORA
            SE (casuale ≥ 50)
                ALLORA
                    precisione := generare_numero_random(0, 100)
                    SE (precisione ≥ 25)
                        ALLORA
                            attaccare_nemico(enemy, arma_pg)
                    FINE
                    precisione := generare_numero_random(0, 100)
                    SE (precisione ≥ 25)
                        ALLORA
                            attaccare_personaggio(pg, arma_enemy)
                    FINE
                ALTRIMENTI
                    precisione := generare_numero_random(0, 100)
                    SE (precisione ≥ 25)
                        ALLORA
                            SE (get_durezza_arma(get_arma_nemico(enemy).oggetto_speciale.weapon) > 0)
                                ALLORA
                                    difendere_nemico(enemy)
                                ALTRIMENTI
                                    attaccare_nemico(enemy, arma_pg, punti_attuali)
                            FINE
                    FINE
            FINE
        ALTRIMENTI
```

```
            SE (verificare_uguaglianza_stringhe(risposta, “difendi”) = VERO)
                ALLORA
                    SE (casuale ≥ 50)
                        ALLORA
                            precisione := generare_numero_random(0, 100)
                            SE (precisione ≥ 25)
                                ALLORA
                                    SE (get_durezza_arma(get_weapon_personaggio(pg).oggetto_speciale.weapon)
                                        >0)
                                        ALLORA
                                            difendere_personaggio(pg)
                                        ALTRIMENTI
                                            attaccare_personaggio(pg, arma_enemy, punti_attuali)
                                    FINE
                            FINE
                    FINE
                ALTRIMENTI
                    stampare_in_output(“Comando non riconosciuto“)
            FINE
        FINE
FINCHÈ (get_punti_vita_personaggio(pg) > 0 AND (get_punti_vita_nemico(enemy) > 0)
SE (get_punti_vita_personaggio(pg) ≤ 0)
    ALLORA
        esito := -1
    ALTRIMENTI
        SE (get_punti_vita_nemico(enemy) ≤ 0)
            ALLORA
                esito := 0
        FINE
FINE
```

FUNZIONE attaccare_nemico

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
enemy	nemico che viene attaccato	nemico	
arma_pg	arma utilizzata dal personaggio	arma	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
enemy	nemico con punti vita aggiornati	nemico	

DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO

punti_attuali	variabile di supporto che contiene i nuovi punti del bersaglio dopo che è stato colpito	intero	<= 100

ALGORITMO

punti_attuali := get_punti_vita_nemico(enemy) – get_danno_arma(arma_pg)
enemy := set_punti_vita_nemico(enemy, punti_attuali)

FUNZIONE difendere_nemico

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
enemy	nemico che si difende	nemico	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
enemy	nemico con durezza dell'arma aggiornata	nemico	

ALGORITMO

enemy.weapon.oggetto_speciale.weapon.durezza :=
enemy.weapon.oggetto_speciale.weapon.durezza -1

FUNZIONE attaccare_personaggio

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
personaggio	personaggio che viene attaccato	personaggio	
arma_enemy	arma utilizzata dal nemico	arma	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	pg con punti vita aggiornati	personaggio	

DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
punti_attuali	variabile di supporto che contiene i nuovi punti del bersaglio dopo che è stato colpito	numero intero	<= 100

ALGORITMO

punti_attuali := get_punti_vita_personaggio(pg) – get_danno_arma(arma_enemy)
personaggio := set_punti_vita_personaggio(personaggio, punti_attuali)

FUNZIONE difendere_personaggio

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
personaggio	personaggio che si difende	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
personaggio	personaggio con durezza dell'arma aggiornata	personaggio	

ALGORITMO

personaggio.weapon.oggetto_speciale.weapon.durezza :=
personaggio.weapon.oggetto_speciale.weapon.durezza -1

FUNZIONE mangiare_cibo

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore di cui si vuole aumentare il numero di punti_vita	personaggio	
nome_cibo	nome del cibo che si vuole mangiare	vettore di caratteri	lunghezza massima MAX_CARATT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore con punti vita aumentati ed inventario aggiornato	personaggio	

DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
i	contatore del ciclo e indice dell'inventario	intero	$i \geq 0$
hp	punti vita attuali del personaggio	intero	$hp \leq 100$
trovato	indica se il nome del cibo cercato è stato trovato	booleano	VERO o FALSO
empty	oggetto vuoto da inserire nello slot in cui era presente l'arma impugnata	oggetto	
esito	variabile che indica se ci sono stati errori o meno	numero intero	

ALGORITMO

hp := get_punti_vita_personaggio(pg)
esito := 0
trovato := false

```
empty := set_id_oggetto(empty, 0)
empty := set_nome_oggetto(empty, "vuoto")
empty := set_afferrabile_oggetto(empty, false)
SE(hp = MAX_HEALTH)
    ALLORA
        esito := -4
    ALTRIMENTI
        count := 0
        MENTRE(count < MAX_SLOTS AND trovato = false)
            SE(confronto_stringhe(nome_cibo, get_nome_oggetto(get_slot_inventario_personaggio(pg,
count))) = true)
                ALLORA
                    SE(get_tipo_oggetto(get_slot_inventario_personaggio(pg)) = 'c')
                        ALLORA
                            pg := set_punti_vita_personaggio(pg, get_punti_vita_personaggio(pg) +
get_punti_vita_cibo(get_slot_inventario(pg, count)))
                            pg := set_slot_inventario_personaggio(pg, count, empty)
                            SE(get_punti_vita_personaggio(pg) > MAX_HEALTS)
                                ALLORA set_punti_vita_personaggio(pg, MAX_HEALT)
                            FINE
                        ALTRIMENTI esito := -5

                    FINE
                    trovato := true
            FINE
            count := count + 1
        FINE
        SE(trovato = false)
            ALLORA esito := -6
        FINE
FINE
gestione_errore(esito)
```

FUNZIONE visualizzare_salute

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore di cui si vuole sapere quanti punti vita possiede	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
punti_vita	indica il numero di punti vita ancora a disposizione del personaggio	intero	punti_vita ≤ 100

ALGORITMO

punti_vita := get_punti_vita(pg)

FUNZIONE visualizzare_inventario

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	giocatore di cui si vuole conoscere il contenuto dell'inventario	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
inventario	contenuto dell'inventario del giocatore	vettore di oggetti	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
i	contatore del ciclo e indice dell'inventario	intero	i ≥ 0

ALGORITMO

```
i := 0
MENTRE (i < MAX_SLOTS)
    SE (verificare_uguaglianza_stringhe(get_nome_oggetto(pg.inventario[i]), “vuoto”) != VERO)
        ALLORA
            inventario := pg.inventario[i]
    FINE
    i := i+1
FINE
stampare_in_output(inventario)
```

FUNZIONE inizializzazione_personaggio

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio da inizializzare	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio inizializzato	personaggio	

ALGORITMO

```
pg := set_posizione_personaggio(pg, 1)
pg := set_punti_vita_personaggio(pg, MAX_HEALTH)
pg := inizializzazione_inventario(pg)
```

FUNZIONE inizializzazione_inventario

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio di cui inizializzare l'inventario	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio con inventario inizializzato	personaggio	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
empty	oggetto vuoto da inserire nell'inventario del personaggio per inizializzarlo	oggetto	
count	contatore	numero intero	

ALGORITMO

```
count := 0
empty := set_id_oggetto(empty, 0)
empty := set_nome_oggetto(empty, "vuoto")
empty := set_afferrabile_oggetto(empty, false)
MENTRE(count < MAX_SLOTS)
    pg := set_slot_inventario_personaggio(pg, count, empty)
    count := count + 1
FINE
```

FUNZIONE inizializzazione_parole_chiave

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
keyword	tabella in cui salvare tutte le parole chiave con i rispettivi simboli	tabella_simboli	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
keyword	tabella in cui sono presenti tutte le parole chiave con i rispettivi simboli	tabella_simboli	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	contatore	numero intero	

lenght	variabile necessaria per calcolare l'offset del prelievo	numero intero	
support	variabile in cui è presente una riga del file per volta	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname	nome del file in cui sono presenti le parole chiave	sequenza di caratteri	lunghezza massima MAX_CARATT
count_1	contatore	numero intero	
file_chiave	file in cui sono presenti le parole chiave	FILE	
check	variabile di controllo che ci indica come alternare l'inserimento tra parola e simbolo nella tabella	numero intero	
token	varibile in cui è presente una parola della riga prelevata dal file	sequenza di caratteri	lunghezza massima MAX_CARATT

ALGORITMO

```
pathname := "parole_chiave.txt"
count_1 := 0
SE(file_chiave = aprire_file(pathname) = true)
    ALLORA
        MENTRE(fine_file(file_chiave) = false)
            support := copiare_stringhe(support, prendere_riga_file(file_chiave))
            check := 0
            count := 0
            MENTRE(count < lunghezza_stringa(support))
                token := scan(support, count)
                SE(check = 0)
                    ALLORA
                        keywords := set_parola_tab_simboli(keywords, count_1, token)
                        check := 1
                    ALTRIMENTI
                        SE(check = 1)
                            ALLORA
                                keywords := set_simbolo_tab_simboli(keywords, count_1, token)
                                keywords := set_number_tab_simboli(keywords,
get_number_tab_simboli(keywords) + 1)

                                count_1 := count_1 + 1
                                check := 0
                        FINE
                FINE
```

```
            lenght := 0 + lunghezza_stringa(token)
            count := count + 1 + lenght
        FINE
    FINE
FINE
```

FUNZIONE convert_sequence_char_to_int

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
sequence	sequenza inserita in input che deve essere convertita in intero	vettore dinamico di caratteri	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
number	numero intero ottenuto dalla conversione della sequenza in intero	intero	number > 0

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
count	contatore del ciclo	intero	
moltiplicator	moltiplicatore	intero	

ALGORITMO

```
count := calcolare_lunghezza_stringa(sequence)-1
number := 0
moltiplicator := 1
MENTRE (count ≥ 0)
    number := ((sequence[count] – ‘0’) * moltiplicator) + number
    moltiplicator := moltiplicator * 10
    count := count+1
FINE
```

FUNZIONE remove_object_from_file

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
id_oggetto	id dell’oggetto da rimuovere dal file della storia	numero intero	id_oggetto > 0

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_write	file in cui abbiamo riscritto la storia privo dell’oggetto con id dato in input	FILE	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pathname_write	nome del file in cui riscrivere la storia senza l'oggetto con id dato in input	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname_read	nome del file da cui leggere la sotria	sequenza di caratteri	massima_lunghezza MAX_CARATT
prelievo	variabile di supporto in cui inserire una riga per volta del file letto	sequenza di caratteri	massima lunghezza MAX CARATT
token	variabile che contiene una parola per volta della riga prelevata dal file (le parole sono limitate dagli spazi)	sequenza di caratteri	massima lunghezza MAX CARATT
file_read	file da cui leggere la storia	FILE	

ALGORITMO

```
pathname_write := "storia_support.txt"
pathname_read := "storia.txt"
SE(file_read = aprire_file(pathname_read) = true AND file_write = aprire_file(pathname_write) = true)
    ALLORA
        MENTRE(fine_file(file_read) = false)
            prelievo:= copiare_stringhe(prelievo, prendere_riga_file(file_read))
            SE(prelievo[0] = '*')
                ALLORA
                    token := scan(prelievo, 1)
                    SE(convert_sequence_char_to_int(token) != id_object)
                        ALLORA file_write := scrivere_riga_file(prelievo)
                    FINE
                ALTRIMENTI file_write := scrivere_riga_file(prelievo)
            FINE
        FINE
    ALTRIMENTI
FINE
file_write := chiudere_file(file_write)
file_read := chiudere_file(file_read)
rimuovi_file(pathname_read)
file_write := rinomina_file(pathname_write, pathname_read)
```

FUNZIONE inizializzazione_file

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_read	file originale da copiare nel file in uso durante il gioco ad inizio partita	FILE	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_write	file contenente la storia che possiamo utilizzare per giocare (modificare e spostare elementi)	FILE	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pathname_write	nome del file in cui scrivere la storia originale	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname_read	nome del file da cui leggere la storia originale	sequenza di caratteri	massima_lunghezza MAX_CARATT
prelievo	variabile di supporto in cui inserire una riga per volta del file letto	sequenza di caratteri	massima lunghezza MAX CARATT

ALGORITMO

pathname_write := “storia.txt”

pathname_read := “storia_originale.txt”

SE(file_read = aprire_file(pathname_read) = true AND file_write = aprire_file(pathname_write) = true)

    MENTRE(fine_file(file_read) = false)

        prelievo := prendere_riga_file(file_read)

        file_write := scrivere_riga_file(prelievo)

    FINE

FINE

FUNZIONE insert_object_into_file

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
obj	oggetto da inserire all’interno del file	oggetto	
id_number	id della stanza in cui inserire l’oggetto	numero intero	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_storia	file in cui è stato inserito l’oggetto	FILE	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pathname_storia	nome del file da cui leggere la storia	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname_support	nome del file di supporto in cui inserire la storia compreso di oggetto da voler inserire	sequenza di caratteri	massima_lunghezza MAX_CARATT
prelievo	variabile di supporto in cui inserire una riga per volta del file letto	sequenza di caratteri	massima lunghezza MAX CARATT
token	variabile che contiene una parola per volta della riga prelevata dal file (le parole sono limitate dagli spazi)	sequenza di caratteri	massima lunghezza MAX CARATT
file_support	file in cui inserire la storia compreso di oggetto da voler inserire	FILE	

```
pathname_storia = "storia.txt"
pathname_support = "support_storia.txt"
SE(file_storia = aprire_file(pathname_storia) = true AND file_support = aprire_file(pathname_support) = true)
    ALLORA
        MENTRE(file_file(file_storia) = false)
            support := prendere_riga_file(file_storia)
            SE(support[0] = '^')
                ALLORA
                    token := scan(support, 1)
                    SE(convert_sequence_char_to_int(token) = id_number)
                        ALLORA
                            MENTRE(support[0] != '*' AND support[0] != '%')
                                file_support := scrivere_sequenza_file(support)
                                support := copiare_stringhe(support, prendere_riga_file(file_storia))
                            FINE
                            file_support := scrivere_sequenza_file(get_id_oggetto(obj), get_nome_oggetto(obj))

                            SE(get_afferrabile_oggetto(obj) = true)
                                ALLORA
                                    file_support := scrivere_sequenza_file("true")
                                ALTRIMENTI
                                    file_support := scrivere_sequenza_file("false")
                            FINE
                            SE(get_tipo_oggetto(obj) = true)
                                ALLORA
                                    SE(get_tipo_oggetto(obj) = 'a')
```

```
                                ALLORA
                                    file_support := scrivere_sequenza_file("a",
get_danno_arma(obj.oggetto_speciale.weapon), get_durezza_arma(obj.oggetto_speciale.weapon))
                                ALTRIMENTI
                                    SE(get_tipo_oggetto(obj) = 'c')
                                        ALLORA
                                            file_support := scrivere_sequenza_file("c",
get_punti_vita_cibo(obj.oggetto_speciale.food))
                                        ALTRIMENTI
                                            SE(get_tipo_oggetto(obj) = 'k')
                                                ALLORA
                                                    SE(get_apertuta_apribile(obj.oggetto_speciale.key) = true)
                                                        ALLORA file_support := scrivere_sequenza_file(" k false")
                                                        ALTRIMENTI file_support := scivere_sequenza_file(" k
true")
                                                    FINE
                                                    file_support :=
scrivere_sequenza_file(get_id_chiave_da_usare(obj.oggetto_speciale.keu))
                                            FINE
                                    FINE
                            FINE
                        FINE
                        file_support := scrivere_sequenza_file("\n", support)
                    ALTRIMENTI
                        file_support := scrivere_sequenza_file(support)
                FINE
            ALTRIMENTI
                file_support := scrivere_sequenza_file(support)
        FINE
    FINE

FINE
file_storia := chiudere_file(file_storia)
file_support := chiudere_file(file_support)

rimuovi_file(pathname_storia)
file_support := rinomina_file(pathname_support, pathname_storia)
```

FUNZIONE remove_enemy_from_file

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pos	posizione del personaggio da cui eliminare il nemico abbattuto	numero intero	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_storia	file da cui è stato eliminato il nemico	FILE	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pathname_storia	nome del file da cui leggere la storia	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname_support	nome del file di supporto in cui inserire la storia escluso il nemico	sequenza di caratteri	massima_lunghezza MAX_CARATT
prelievo	variabile di supporto in cui inserire una riga per volta del file letto	sequenza di caratteri	massima lunghezza MAX CARATT
token	variabile che contiene una parola per volta della riga prelevata dal file (le parole sono limitate dagli spazi)	sequenza di caratteri	massima lunghezza MAX CARATT
file_support	file in cui inserire la storia escluso il nemico	FILE	

```
pathname_storia := "storia.txt"
pathname_support := "storia_support.txt"
SE(file_storia = aprire_file(pathname_storia) = true AND file_support = aprire_file(pathname_support) = true)
    ALLORA
        MENTRE(fine_file(file_storia) = false)
            support := copiare_stringhe(support, prendere_riga_file(file_storia))
            SE(support[0] = '^')
                ALLORA
                    token := scan(support, 1)
                    file_support := scrivere_sequenza_file(support)

                    SE(convert_sequence_char_to_int(token) = pos)
                        ALLORA
                            MENTRE(support[0] != '_')
                                support := copiare_stringhe(support, prendere_riga_file(file_storia))

                                SE(support[0] != '$' AND support[0] != '&')
                                    file_support := scrivere_sequenza_file(support)
```

FINE

FINE

FINE

ALTRIMENTI file_support := scrivere_sequenza_file(support)

FINE

FINE

FINE

file_storia := chiudere_file(file_storia)

file_support := chiudere_file(file_support)

rimuovi_file(pathname_storia)

file_support := rinomina_file(pathname_support, pathname_storia)

FUNZIONE open_door_from_file

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pos	posizione del personaggio	numero intero	
dir	direzione di cui si vuole aprire la porta	direzione	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_storia	file da cui è stata aperta la porta (precedentemente chiusa)	FILE	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pathname_storia	nome del file da cui leggere la storia	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname_support	nome del file di supporto in cui aprire la porta	sequenza di caratteri	massima_lunghezza MAX_CARATT
prelievo	variabile di supporto in cui inserire una riga per volta del file letto	sequenza di caratteri	massima lunghezza MAX CARATT
token	variabile che contiene una parola per volta della riga prelevata dal file (le parole sono limitate dagli spazi)	sequenza di caratteri	massima lunghezza MAX CARATT
file_support	file in cui inserire la storia in cui è aperta la porta desiderata	FILE	

ALGORITMO

pathname_storia := "storia.txt"

```
pathname_support := "storia_support.txt"
SE(file_storia = aprire_file(pathname_storia) = true AND file_support = aprire_file(pathname_support) = true)
    ALLORA
        MENTRE(fine_file(file_storia) = false)
            prelievo := copiare_stringheprelievoprendere_riga_file(file_storia))
            SE(support[0] = '^')
                ALLORA
                    token := scan(support, 1)
                    file_support := scrivere_sequenza_file(support)

                    SE(convert_sequence_char_to_int(token) = pos)
                        ALLORA
                            SE(prelievo[0] = '%')
                                ALLORA
                                    token = scan(prelievo, 1)
                                    SE(convert_sequence_char_to_int(token) = get_id_destinazione(dir))
                                        ALLORA file_support := scrivere_sequenza_file(prelievo[0], get_id_destinazione(dir), get_punto_cardinale_dir, "true 0\n")
                                        ALTRIMENTI file_support := scrivere_sequenza_file(support)
                                    FINE
                                ALTRIMENTI file_support := scrivere_sequenza_file(support)
                            FINE
                    FINE
                ALTRIMENTI file_support := scrivere_sequenza_file(support)
            FINE
        FINE
FINE
file_storia := chiudere_file(file_storia)
file_support := chiudere_file(file_support)

rimuovi_file(pathname_storia)
file_support := rinomina_file(pathname_support, pathname_storia)
```

FUNZIONE open_door_from_file

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
sequence	sequenza di caratteri da convertire in un numero intero	sequenza di caratteri	lunghezza massima MAX_LENGHT

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
number	numero intero corrispondente	numero intero	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
moltiplicator	moltiplicatore necessario per convertire da carattere ad intero e per definire unità, decine, centinaia ecc...	numero intero	
count	variabile contatore	numero intero	

ALGORITMO

```
count := lunghezza_stringa(sequence) - 1
number := 0
moltiplicator := 1

MENTRE(count >= 0)
    number := ((sequence[count] - '0') * moltiplicator) + number
    moltiplicator := moltiplicator * 10
    count := count - 1
FINE
```

FUNZIONE eseguire_comando

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
ident	tabella che contiene l'input dell'utente con i rispettivi simboli identificatori	tabella_simboli	
pg	personaggio che deve eseguire l'azione	personaggio	
stanza_attuale	stanza in cui si trova il personaggio	ambiente	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio che ha eseguito l'azione	personaggio	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
esito	variabile che indica se ci sono stati errori	numero intero	

direction	carattere che indica la direzione voluta (diminutivo di nord, sud, est, ovest)	carattere	

ALGORITMO

```
esito := 0
SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "TKE") = true)
   ALLORA
      pg := prendere_oggetto(pg, get_parola_tab_simboli(tab, 1))
   ALTRIMENTI
      SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "LVE") = true)
         ALLORA
            pg := lasciare_oggetto(pg, get_parola_tab_simboli(tab, 1))
         ALTRIMENTI
            SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "VAI") = true)
               ALLORA
                  SE(confronto_stringhe(get_parola_tab_simboli(tab, 1), "nord") = true)
                     ALLORA
                        direction := 'n'
                     ALTRIMENTI
                        SE(confronto_stringhe(get_parola_tab_simboli(tab, 1), "est") = true)
                           ALLORA
                              direction := 'e'
                           ALTRIMENTI
                              SE(confronto_stringhe(get_parola_tab_simboli(tab, 1), "sud") = true)
                                 ALLORA
                                    direction := 's'
                                 ALTRIMENTI
                                    SE(confronto_stringhe(get_parola_tab_simboli(tab, 1), "ovest") =
true)
                                       ALLORA
                                          direction := 'a'
                                       ALTRIMENTI
                                          SE(confronto_stringhe(get_parola_tab_simboli(tab, 1), "su") =
true)
                                             ALLORA
                                                direction := 'b'
                                             ALTRIMENTI
                                                SE(confronto_stringhe(get_parola_tab_simboli(tab, 1),
"ovest") = true)
                                                   ALLORA
                                                      direction := 'o'
```

```
                                        ALTRIMENTI esito := -2
                                    FINE
                                FINE
                            FINE
                        FINE
                    FINE
                SE(esito = 0)
                    ALLORA
                        pg := muovere_personaggio(pg, stanza, direction)
                FINE
            ALTRIMENTI
                SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "LOK") = true)
                    ALLORA
                        SE(confronto_stringhe(get_parola_tab_simboli(tab, 1), "arma") = true)
                            ALLORA
                                SE(get_arma_personaggio(pg) = true)
                                    ALLORA stampa_video("stai impugnando: ",
get_nome_oggetto(get_arma_personaggio(pg)), "danno: "
get_danno_arma(get_arma_personaggio(pg)), "durezza: ",
get_durezza_arma(get_arma_personaggio(pg)))
                                    ALTRIMENTI esito = -10
                                FINE
                            ALTRIMENTI
                                SE(confronto_stringe(get_parola_tab_simboli(tab, 1), "uomo") = true AND
get_posizione_personaggio(pg) = 14)
                                    ALLORA indovino_indovinello()
                                    ALTRIMENTI guarda_oggetto(stanza_attuale,
get_parola_tab_simboli(tab, 1))
                                FINE
                        FINE
                    ALTRIMENTI
                        SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "OPN") = true)
                            ALLORA
                                SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 1), "porta"))
                                    ALLORA
                                        SE(confronto_stringhe(get_parola_tab_simboli(tab, 2), "nord") = true)
                                            ALLORA
                                                direction := 'n'
                                            ALTRIMENTI
                                                SE(confronto_stringhe(get_parola_tab_simboli(tab, 2), "est") =
true)
```

```
                                        ALLORA
                                            direction := 'e'
                                        ALTRIMENTI
                                            SE(confronto_stringhe(get_parola_tab_simboli(tab, 2),
"sud") = true)
                                                ALLORA
                                                    direction := 's'
                                                ALTRIMENTI
                                                    SE(confronto_stringhe(get_parola_tab_simboli(tab,
2), "ovest") = true)
                                                        ALLORA direction := 'o'
                                                        ALTRIMENTI esito := -2
                                                    FINE
                                            FINE
                                        FINE
                                    FINE
                                    SE(esito != -2)
                                        ALLORA aprire_porta(stanza_attuale, pg, direction)
                                    FINE
                                ALTRIMENTI aprire_oggetto(pg, get_parola_tab_simboli(tab, 1))
                            FINE
                        ALTRIMENTI
                            SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "EAT") = true)
                                ALLORA
                                    mangiare_cibo(pg, get_parola_tab_simboli(tab, 1))
                                ALTRIMENTI
                                    SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "INV") =
true)
                                        ALLORA stampare_inventario(pg)
                                        ALTRIMENTI
                                            SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "SLT")
= true)
                                                ALLORA
                                                    stampare_video(get_punti_vita_personaggio(pg))
                                                ALTRIMENTI
                                                    SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0),
"HLP") = true)
                                                        ALLORA aiuto()
                                                        ALTRIMENTI

    SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "SLV") = true)
                                                            ALLORA
```

```
                                                    salvataggio_partita(stanza_attuale, pg)
                                                ALTRIMENTI

    SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "LOA") = true)
                                                        ALLORA
                                                            pg := caricare_partita(pg)
                                                        ALTRIMENTI

    SE(confronto_stringhe(get_simbolo_tab_simboli(tab, 0), "IMP") = true)
                                                                ALLORA
                                                                    pg := impugnare_arma(pg,
get_parola_tab_simboli(tab, 1))
                                                                ALTRIMENTI
                                                                    esito := -1
                                                            FINE
                                                        FINE
                                                    FINE
                                                FINE
                                            FINE
                                        FINE
                                    FINE
                                FINE
                            FINE
                        FINE
                    FINE
                FINE
            FINE
        FINE
FINE
gestione_errore(esito)
```

FUNZIONE gestione_errore

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
esito	numero di errore verificato da una delle funzioni principali	numero intero	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
//messaggi di avvertimento di errori			

ALGORITMO

```
SE(esito = -1)
    ALLORA stampa_video("non ho capito")
FINE
```

```
SE(esito = -2)
    ALLORA stampa_video("non ho capito la direzione")
FINE
SE(esito = -3)
    ALLORA stampa_video("direzione non consentita")
FINE
SE(esito = -4)
    ALLORA stampa_video("punti vita massimi")
FINE
SE(esito = -5)
    ALLORA stampa_video("non penso che si possa mangiare")
FINE
SE(esito = -6)
    ALLORA stampa_video("oggetto non presente in inventario")
FINE
SE(esito = -7)
    ALLORA stampa_video("non vedi nulla del genere")
FINE
SE(esito = -8)
    ALLORA stampa_video("inventario pieno")
FINE
SE(esito = -9)
    ALLORA stampa_video("non trovi nulla di interessante")
FINE
SE(esito = -10)
    ALLORA stampa_video("non stai impugnando nulla")
FINE
SE(esito = -11)
    ALLORA stampa_video("la porta è chiusa")
FINE
SE(esito = -12)
    ALLORA stampa_video("non hai la chiave giusta")
FINE
SE(esito = -13)
    ALLORA stampa_video("è già aperto")
FINE
```

FUNZIONE salvataggio_partita

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
personaggio	personaggio da salvare nel file	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
file_current	file che contiene tutte le informazioni del personaggio	FILE	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
pathname_storia_corrente	nome del file in cui trovare la storia che il personaggio sta salvando	sequenza di caratteri	lunghezza massima MAX_CARATT
pathname_storia_salvataggio	nome del file in cui salvare la storia che il personaggio sta giocando	sequenza di caratteri	lunghezza massima MAX_CARATT
file_salvataggio	file in cui stiamo salvando le informazioni	FILE	
prelievo	variabile di supporto in viene inserita una riga per volta del file che stiamo leggendo	sequenza di caratteri	lunchezza massima MAX_CARATT
count	contatore	numero intero	deve partire da 0

ALGORITMO

pathname_storia_corrente := "storia.txt"

pathname_storia_salvataggio := "storia_salvataggio.txt"

SE(file_current := aprire_file(pathname_storia_corrente) = true AND file_salvataggio := aprire_file(pathname_storia_salvataggio) = true)

ALLORA

MENTRE(file_file(file_current) = false)

prelievo := copiare_stringhe(prelievo, prendere_riga_file(file_current))

file_salvataggio := scrivere_sequenza_file(prelievo)

FINE

FINE

file_current := chiudere_file(file_current)

file_salvataggio := chiudere_file(file_salvataggio)

pathname_personaggio_salvataggio := "personaggio_salvataggio.txt"

count := 0

SE(file_current := aprire_file(pathname_personaggio_salvataggio) = true)

ALLORA

```
file_current := scrivere_sequenza_file(get_posizione_personaggio(pg),
get_punti_vita_personaggio(pg))
SE(get_arma_personaggio(pg) = true)
    ALLORA
        file_current := scrivere_sequenza_file(get_id_oggetto(get_arma_personaggio(pg)),
get_nome_oggetto(get_arma_personaggio(pg)), get_danno_arma(get_arma_personaggio(pg)),
get_durezza_arma(get_arma_personaggio(pg)))
FINE
MENTRE(get_id_oggetto(get_slot_inventario_personaggio(pg)) != 0)
    file_current := scrivere_sequenza_file(get_id_oggetto(get_slot_inventario_personaggio(pg,
count)), get_nome_oggetto(get_slot_inventario_personaggio(pg, count)))
    SE(get_afferrabilita_oggetto(get_slot_inventario_personaggio(pg, count)))
        ALLORA
            file_current := scrivere_sequenza_file("true")
        ALTRIMENTI
            file_current := scrivere_sequenza_file("false")
    FINE
    SE(get_tipo_oggetto(get_slot_inventario_personaggio(pg, count)) = true)
        ALLORA
            SE(get_tipo_oggetto(get_slot_inventario_personaggio(pg, count)) = 'a')
                ALLORA
                    file_current := scrivere_sequenza_file("a",
get_danno_arma(get_slot_inventario_personaggio(pg, count).oggetto_speciale.weapon),
get_durezza_arma(get_slot_inventario_personaggio(pg, count).oggetto_speciale.weapon))
                ALTRIMENTI
                    SE(get_tipo_oggetto(get_slot_inventario_personaggio(pg, count)) = 'c')
                        ALLORA
                            file_current := scrivere_sequenza_file("c",
get_punti_vita_cibo(get_slot_inventario_personaggio(pg, count).oggetto_speciale.cibo))
                        ALTRIMENTI
                            SE(get_tipo_oggetto(get_slot_inventario_personaggio(pg, count)) = 'k')
                                ALLORA
                                    SE(get_apertura_apribile(get_slot_inventario_personaggio(pg, count)), =
true)
                                        ALLORA file_current := scrivere_sequenza_file("k true")
                                        ALTRIMENTI file_current := scrivere_sequenza_file("k false")
                                    FINE
                                    file_current :=
scrivere_sequenza_file(get_id_chiave_da_usare(get_slot_inventario_personaggio(pg, count)))
                            FINE
                    FINE
            FINE
    FINE
```

```
            count := count + 1
        FINE
FINE
file_current := chiudere_file(file_current)
```

FUNZIONE caricare_partita

INPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
personaggio	personaggio di cui caricare tutte le informazioni prelevate dal file	personaggio	

OUTPUT

NOME	DESCRIZIONE	TIPO	VINCOLO
pg	personaggio con le nuove informazioni caricate	personaggio	

VARIABILI DI LAVORO

NOME	DESCRIZIONE	TIPO	VINCOLO
file_storia_salvataggio	nome del file in cui sono salvate le informazioni riguardo la storia	sequenza di caratteri	lunghezza massima MAX_CARATT
file_storia_caricare	nome del file in cui è presente l'attuale storia che il pg sta giocando	sequenza di caratteri	lunghezza massima MAX_CARATT
file_salvataggio	file in cui è presente il salvataggio	FILE	
support	variabile di supporto in viene inserita una riga per volta del file che stiamo leggendo	sequenza di caratteri	lunchezza massima MAX_CARATT
count	contatore	numero intero	deve partire da 0
file_caricare	file in cui è presente la storia che il pg sta giocando	FILE	
temp_obj	variabile temporanea che contiene tutte le informazioni dell'oggetto che stiamo prelevando	oggetto	
danno_flag	variabile boolena che indica se abbiamo prelevato o meno il danno dell'arma	booleana	
durezza_flag	variabile boolena che indica se abbiamo prelevato o meno la durezza dell'arma	booleana	

nome_flag	variabile boolena che indica se abbiamo prelevato o meno il nome dell'oggetto	booleana	
id_flag	variabile boolena che indica se abbiamo prelevato o meno l'id dell'oggetto	booleana	
aff_flag	variabile boolena che indica se abbiamo prelevato o meno l'afferrabilità dell'oggetto	booleana	
tipo_flag	variabile boolena che indica se abbiamo prelevato o meno il tipo dell'oggetto	booleana	
par1_flag	variabile boolena che indica se abbiamo prelevato o meno il primo parametro dell'oggetto	booleana	
par2_flag	variabile boolena che indica se abbiamo prelevato o meno il secondo parametro dell'oggetto	booleana	
temp	variabile di supporto per il caricamento dell'arma del personaggio	oggetto	

ALGORITMO

```
file_storia_salvataggio := "storia_salvataggio.txt";
file_storia_caricare := "storia.txt";

SE((file_salvataggio := aprire_file(file_storia_salvataggio)) = true AND (file_caricare =
aprire_file(file_storia_caricare)) = true)
    ALLORA
        MENTRE(fine_file(file_salvataggio) = false)
            support := copiare_stringhe(support, prendere_riga_file(file_salvataggio))
            file_caricare := scrivere_sequenza_file(support);
        FINE
FINE
file_caricare := chiudere_file(file_caricare)
file_salvataggio := chiudere_file(file_salvataggio)

inizializzazione_inventario(pg)

file_dati_pg := "personaggio_salvataggio.txt"
SE(file_salvataggio := aprire_file(file_dati_pg) = true)
    ALLORA
        count_inv := 0
```

```
        support := copiare_stringhe(support, prendere_riga_file(file_salvataggio))
        token := scan(support, 0)
        pg := set_posizione_personaggio(pg, convert_sequence_char_to_int(token))
        support := copiare_stringhe(support, prendere_riga_file(file_salvataggio))
        token := scan(support, 0)
        pg := set_punti_vita_personaggio(pg, convert_sequence_char_to_int(token))
        MENTRE(fine_file(file_salvataggio) = false)
            count := 1
            support := copiare_stringhe(support, prendere_riga_file(file_salvataggio))
            SE(support[0] = '&')
                ALLORA
                    id_flag := false
                    nome_flag := false
                    danno_flag := false
                    durezza_flag := false
                    temp := set_afferrabilita_oggetto(temp, true)
                    temp := set_tipo_oggetto(temp, 'a')
                    MENTRE(support[count] != '\0')
                        token := scan(support, count)
                        SE(id_flag = false)
                            ALLORA
                                temp := set_id_oggetto(temp, convert_sequence_char_to_int)
                                id_flag := true
                            ALTRIMENTI
                                SE(nome_flag = false)
                                    ALLORA
                                        temp := set_nome_oggetto(temp, token)
                                        nome_flag := true
                                    ALTRIMENTI
                                        SE(danno_flag = true)
                                            ALLORA
                                                temp := set_danno_arma(temp.oggetto_speciale.weapon,
convert_sequence_char_to_int(token))
                                                danno_flag := true
                                            ALTRIMENTI
                                                SE(durezza_flag = false)
                                                    ALLORA
                                                        temp := set_durezza_arma(temp.oggetto_speciale.weapon,
convert_sequence_char_to_int(token))
                                                        durezza_flag := true
```

```
                                FINE
                            FINE
                        FINE
                    FINE
                    SE(durezza_flag = true)
                        ALLORA pg := set_weapon_personaggio(pg, temp)
                    FINE
                    lenght := 0 + lunghezza_stringe(token)
                    count := count + 1 + lenght
                FINE
            ALTRIMENTI
                SE(support[0] = '*')
                    ALLORA
                        id_flag := false
                        nome_flag := false
                        aff_flag := false
                        tipo_flag := false
                        par1_flag := false
                        par2_flag := false
                        MENTRE(prelievo[count] != '\0')
                            token := scan(prelievo, count)
                            SE(id_flag = false)
                                ALLORA
                                    temp_obj := set_id_oggetto(temp_obj,
convert_sequence_char_to_int(tokne))
                                    id_flag := true
                                ALTRIMENTI
                                    SE(nome_flag = false)
                                        ALLORA
                                            temp_obj := set_nome_oggetto(temp_obj, token)
                                            nome_flag := true
                                        ALTRIMENTI
                                            SE(aff_flag = true)
                                                ALLORA
                                                    SE(confronto_stringhe(token, "true") = true)
                                                        ALLORA
                                                            temp_obj := set_affettabile_oggetto(temp_obj, true)
                                                        ALTRIMENTI
                                                            temp_obj := set_affettabile_oggetto(temp_obj, false)
                                                    FINE
```

```
                                    aff_flag := true
                                    ALTRIMENTI
                                        SE(tipo_flag = false)
                                            ALLORA
                                                SE(confronto_stringhe(token, "a") = true)
                                                    ALLORA
                                                        temp_obj := set_tipo_oggetto(temp_obj, 'a')
                                                    ALTRIMENTI
                                                        SE(confronto_stringhe(token, "c") = true)
                                                            ALLORA
                                                                temp_obj := set_tipo_oggetto(temp_obj, 'c')
                                                            ALTRIMENTI
                                                                SE(confronto_stringhe(token, "k") = true)
                                                                    ALLORA temp_obj :=
set_tipo_oggetto(temp_obj, 'k')
                                                                FINE
                                                        FINE
                                                FINE
                                                tipo_flag := true
                                            ALTRIMENTI
                                                SE(par1_flag = false)
                                                    ALLORA
                                                        SE(get_tipo_oggetto(temp_obj) = 'a')
                                                            ALLORA
                                                                temp_obj.oggetto_speciale.weapon :=
set_danno_arma(temp_obj.oggetto_speciale.weapon, convert_sequence_char_to_int(token))
                                                            ALTRIMENTI
                                                                SE(get_tipo_oggetto(temp_obj) = 'c')
                                                                    ALLORA
                                                                        temp_obj.oggetto_speciale.food :=
set_punti_vita_cibo(temp_obj.oggetto_speciale.food, convert_sequence_char_to_int(token))
                                                                    ALTRIMENTI
                                                                        SE(get_tipo_oggetto(temp_obj) = 'k')
                                                                            ALLORA
                                                                                SE(confronto_stringhe(token,
"true") = true)
                                                                                    ALLORA
    temp_obj.oggetto.speciale.key := set_apertura_apribile(temp_obj.oggetto.speciale.key, true)
                                                                                    ALTRIMENTI
    temp_obj.oggetto.speciale.key := set_apertura_apribile(temp_obj.oggetto.speciale.key, false)
```

```
                                                                FINE
                                                            FINE
                                                        FINE
                                                    FINE
                                                    par1_flag := true
                                                ALTRIMENTI
                                                    SE(par2_flag = false)
                                                        ALLORA
                                                            SE(get_tipo_oggetto(temp_obj) = 'a')
                                                                ALLORA
                                                                    temp_obj.oggetto_speciale.weapon := set_durezza_arma(temp_obj.oggetto_speciale.weapon, convert_sequence_char_to_int(token))
                                                                ALTRIMENTI
                                                                    SE(get_tipo_oggettp(temp_obj) = 'k')
                                                                        ALLORA
                                                                            temp_obj.oggetto_speciale.key := set_id_chiave_da_usare(temp_obj.oggetto_speciale.key, convert_sequence_char_to_int(token))
                                                                    FINE
                                                            FINE
                                                            par2_flag := true
                                                        ALTRIMENTI
                                                    FINE
                                                FINE
                                            FINE
                                        FINE
                                    FINE
                                FINE
                            FINE
                            SE(aff_flag = true)
                                ALLORA pg := set_slot_inventario_personaggio(pg, count_inv, temp)
                            FINE
                            lenght := 0 + lunghezza_stringa(token)
                            count := count + 1 + lenght
                        FINE
                    count_inv := count_inv + 1
                FINE
            FINE
        FINE
    FINE
FINE
```