CDungeon

1.0

Generato da Doxygen 1.16.1

Chapter 1

Gerarchia delle directory

1.1 Directory

Chapter 2

Indice delle strutture dati

2.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

Chapter 3

Indice dei file

3.1 Elenco dei file

Questo è un elenco di tutti i file con una loro breve descrizione:

Chapter 4

Documentazione delle directory

4.1 Riferimenti per la directory src

File

- file combattimento.c

  Implementazione delle logiche di combattimento.
- file combattimento.h

  Gestione del combattimento e delle interazioni ostili.
- file dungeon.c

  Implementazione della logica del dungeon e delle missioni.
- file dungeon.h

  Gestione della generazione e del flusso del dungeon.
- file giocatore.c

  Implementazione delle funzioni relative al giocatore.
- file giocatore.h

  Definizione della struttura Giocatore e funzioni correlate.
- file main.c

  Entry point del gioco CDungeon.
- file menu.c

  Implementazione dei menu e dell'interfaccia utente.
- file menu.h

  Gestione dei menu e dell'interfaccia utente.
- file salvataggio.c

  Implementazione del sistema di persistenza dei dati.
- file salvataggio.h

  Gestione del sistema di salvataggio e caricamento.
- file stanza.h

  Definizione della struttura Stanza e dei tipi di stanza.
- file utilita.c

  Implementazione delle funzioni di utilità.
- file utilita.h

  Funzioni di utilità generale per il gioco.

Chapter 5

Documentazione delle classi

5.1 Riferimenti per la struct Giocatore

Struttura che rappresenta lo stato del giocatore.

`#include <giocatore.h>`

Campi

- int punti_vita
- int max_punti_vita
- int monete
- int numero_oggetti
- int missione_palude
- int missione_magione
- int missione_grotta
- int ha_spada
- int ha_armatura
- int ha_spada_eroe
- int ha_chiave_castello

5.1.1 Descrizione dettagliata

Struttura che rappresenta lo stato del giocatore.

Definizione alla linea 12 del file giocatore.h.

5.1.2 Documentazione dei campi

5.1.2.1 ha_armatura

`int Giocatore::ha_armatura`

1 se possiede l'armatura, 0 altrimenti

Definizione alla linea 26 del file giocatore.h.

Referenziato da applica_danno_trappola(), inizia_combattimento(), inizializza_giocatore(), mostra_negozio(), e stampa_statistiche_giocatore().

5.1.2.2 ha_chiave_castello

```
int Giocatore::ha_chiave_castello
```

1 se possiede la chiave del castello, 0 altrimenti

Definizione alla linea 28 del file giocatore.h.

Referenziato da esegui_missione(), inizializza_giocatore(), e stampa_statistiche_giocatore().

5.1.2.3 ha_spada

```
int Giocatore::ha_spada
```

1 se possiede la spada base, 0 altrimenti

Definizione alla linea 25 del file giocatore.h.

Referenziato da inizia_combattimento(), inizializza_giocatore(), mostra_negozio(), e stampa_statistiche_giocatore().

5.1.2.4 ha_spada_eroe

```
int Giocatore::ha_spada_eroe
```

1 se possiede la spada dell'eroe, 0 altrimenti

Definizione alla linea 27 del file giocatore.h.

Referenziato da esegui_missione(), inizia_combattimento(), inizializza_giocatore(), e stampa_statistiche_giocatore().

5.1.2.5 max_punti_vita

```
int Giocatore::max_punti_vita
```

Punti vita massimi del giocatore

Definizione alla linea 15 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_villaggio(), mostra_negozio(), e stampa_statistiche_giocatore().

5.1.2.6 missione_grotta

```
int Giocatore::missione_grotta
```

1 se la missione grotta è completata, 0 altrimenti

Definizione alla linea 22 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_missione(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.7 missione_magione

```
int Giocatore::missione_magione
```

1 se la missione magione è completata, 0 altrimenti

Definizione alla linea 21 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_missione(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.8 missione_palude

```
int Giocatore::missione_palude
```

1 se la missione palude è completata, 0 altrimenti

Definizione alla linea 20 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_missione(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.9 monete

```
int Giocatore::monete
```

Numero di monete possedute

Definizione alla linea 16 del file giocatore.h.

Referenziato da esegui_missione(), gestisci_trucchi(), inizia_combattimento(), inizializza_giocatore(), mostra_menu_villaggio(), mostra_negozio(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.10 numero_oggetti

```
int Giocatore::numero_oggetti
```

Conteggio generico oggetti (non usato per inventario specifico)

Definizione alla linea 17 del file giocatore.h.

Referenziato da esegui_missione(), inizializza_giocatore(), mostra_negozio(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.11 punti_vita

```
int Giocatore::punti_vita
```

Punti vita attuali del giocatore

Definizione alla linea 14 del file giocatore.h.

Referenziato da applica_danno_trappola(), esegui_missione(), gestisci_trucchi(), inizia_combattimento(), inizializza_giocatore(), mostra_menu_missione(), mostra_menu_villaggio(), mostra_negozio(), stampa_salvataggi(), e stampa_statistiche_giocatore().

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/giocatore.h

5.2 Riferimenti per la struct NodoSalvataggio

Nodo di una lista concatenata per memorizzare i salvataggi.

`#include <salvataggio.h>`

Campi

- int id
- char timestamp [32]
- Giocatore dati_giocatore
- struct NodoSalvataggio * prossimo

5.2.1 Descrizione dettagliata

Nodo di una lista concatenata per memorizzare i salvataggi.

Definizione alla linea 14 del file salvataggio.h.

5.2.2 Documentazione dei campi

5.2.2.1 dati_giocatore

```
Giocatore NodoSalvataggio::dati_giocatore
```

Copia dello stato del giocatore

Definizione alla linea 18 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), gestisci_trucchi(), mostra_menu_salvataggi(), salva_tutto_su_file(), e stampa_salvataggi().

5.2.2.2 id

```
int NodoSalvataggio::id
```

Identificativo univoco del salvataggio

Definizione alla linea 16 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), carica_salvataggio(), elimina_salvataggio(), salva_tutto_su_file(), e stampa_salvataggi().

5.2.2.3 prossimo

```
struct NodoSalvataggio* NodoSalvataggio::prossimo
```

Puntatore al prossimo nodo della lista

Definizione alla linea 19 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), carica_salvataggio(), elimina_salvataggio(), libera_salvataggi(), salva_tutto_su_file(), e stampa_salvataggi().

5.2.2.4 timestamp

```
char NodoSalvataggio::timestamp[32]
```

Data e ora di creazione del salvataggio

Definizione alla linea 17 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), salva_tutto_su_file(), e stampa_salvataggi().

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/salvataggio.h

5.3 Riferimenti per la struct Stanza

Struttura che rappresenta una stanza del dungeon.

```
#include <stanza.h>
```

Campi

- char nome [32]
- TipoStanza tipo
- int danno
- int colpo_fatale
- int ricompensa_monete
- int is_boss

5.3.1 Descrizione dettagliata

Struttura che rappresenta una stanza del dungeon.

Definizione alla linea 23 del file stanza.h.

5.3.2 Documentazione dei campi

5.3.2.1 colpo_fatale

```
int Stanza::colpo_fatale
```

Soglia per il colpo critico/fatale del nemico

Definizione alla linea 28 del file stanza.h.

Referenziato da genera_stanza_missione(), e inizia_combattimento().

5.3.2.2 danno

```
int Stanza::danno
```

Danno potenziale della trappola o attacco nemico

Definizione alla linea 27 del file stanza.h.

Referenziato da applica_danno_trappola(), esegui_missione(), genera_stanza_missione(), e inizia_combattimento().

5.3.2.3 is_boss

```
int Stanza::is_boss
```

1 se la stanza contiene un boss, 0 altrimenti

Definizione alla linea 30 del file stanza.h.

Referenziato da esegui_missione(), e genera_stanza_missione().

5.3.2.4 nome

```
char Stanza::nome[32]
```

Nome descrittivo della stanza

Definizione alla linea 25 del file stanza.h.

Referenziato da applica_danno_trappola(), esegui_missione(), genera_stanza_missione(), e inizia_combattimento().

5.3.2.5 ricompensa_monete

```
int Stanza::ricompensa_monete
```

Monete ottenibili completando la stanza

Definizione alla linea 29 del file stanza.h.

Referenziato da esegui_missione(), genera_stanza_missione(), e inizia_combattimento().

5.3.2.6 tipo

```
TipoStanza Stanza::tipo
```

Tipo di stanza (nemico, trappola, ecc.)

Definizione alla linea 26 del file stanza.h.

Referenziato da esegui_missione(), e genera_stanza_missione().

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/stanza.h

Chapter 6

Documentazione dei file

6.1 Riferimenti per il file src/combattimento.c

Implementazione delle logiche di combattimento.

```
#include "combattimento.h"
#include "utilita.h"
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
```

Funzioni

- int inizia_combattimento (Giocatore *g, Stanza *e)

  Inizia un combattimento tra il giocatore e un nemico.
- void applica_danno_trappola (Giocatore *g, Stanza *trappola)

  Applica i danni di una trappola al giocatore.
- int combattimento_boss_finale (Giocatore *g)

  Gestisce il boss fight finale contro il Signore Oscuro.

6.1.1 Descrizione dettagliata

Implementazione delle logiche di combattimento.

Definizione nel file combattimento.c.

6.1.2 Documentazione delle funzioni

6.1.2.1 applica_danno_trappola()

```
void applica_danno_trappola (
            Giocatore * g,
            Stanza * trappola)
```


Applica i danni di una trappola al giocatore.

Parametri

g	Puntatore al giocatore.
trappola	Puntatore alla stanza contenente la trappola.

Definizione alla linea 88 del file combattimento.c.

Referenzia Stanza::danno, Giocatore::ha_armatura, Stanza::nome, e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.1.2.2 combattimento_boss_finale()

```
int combattimento_boss_finale (
            Giocatore * g)
```

Gestisce il boss fight finale contro il Signore Oscuro.

Implementa una versione a turni della Morra Cinese (Sasso-Carta-Forbice) con regole personalizzate. Il giocatore deve vincere 3 round su 5.

Parametri

g	Puntatore al giocatore.

Restituisce

1 se il giocatore vince, 0 se muore.

Definizione alla linea 106 del file combattimento.c.

Referenzia lancia_dado(), leggi_intero(), e valuta_vittoria_morra().

Referenziato da mostra_menu_missione().

6.1.2.3 inizia_combattimento()

```
int inizia_combattimento (
            Giocatore * g,
            Stanza * nemico)
```

Inizia un combattimento tra il giocatore e un nemico.

Gestisce i turni di attacco e difesa, calcola i danni e determina il vincitore del combattimento.

Parametri

g	Puntatore al giocatore.
nemico	Puntatore alla stanza contenente il nemico.

Restituisce

1 se il giocatore vince, 0 se perde (Game Over).

Definizione alla linea 13 del file combattimento.c.

Referenzia Stanza::colpo_fatale, controlla_padovan(), Stanza::danno, Giocatore::ha_armatura, Giocatore::ha_spada, Giocatore::ha_spada_eroe, lancia_dado(), Giocatore::monete, Stanza::nome, Giocatore::punti_vita, e Stanza::ricompensa_monete.

Referenziato da esegui_missione().

6.2 combattimento.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "combattimento.h"
00007 #include "utilita.h"
00008 #include <stdio.h>
00009 #include <string.h>
00010 #include <ctype.h>
00011 #include <stdlib.h>
00012
00013 int inizia_combattimento(Giocatore *g, Stanza *e)
00014 {
00015   printf("\n--- COMBATTIMENTO: %s ---\n", e->nome);
00016
00017   while (g->punti_vita > 0)
00018   {
00019     printf("\nTu: %d HP | Nemico: %s\n", g->punti_vita, e->nome);
00020     printf("Premi Invio per attaccare...");
00021     getchar();
00022
00023     int danno_nemico = e->danno;
00024
00025     // Controllo colpo fatale Generale Orco
00026     if (strcmp(e->nome, "Generale Orco") == 0 && g->ha_spada_eroe)
00027     {
00028       e->colpo_fatale = 5;
00029     }
00030
00031     // Controllo indovinello Drago
00032     if (strcmp(e->nome, "Drago Antico") == 0)
00033     {
00034       int n = (rand() % 500) + 1;
00035       printf("Drago: Il numero %d e' di Padovan? [s/n]: ", n);
00036       char r;
00037       scanf(" %c", &r);
00038       getchar();
00039       int vero = controlla_padovan(n);
00040       if ((tolower(r) == 's' && vero) || (tolower(r) == 'n' && !vero))
00041       {
00042         printf("Giusto! Il Drago non attacca.\n");
00043         danno_nemico = 0;
00044       }
00045       else
00046         printf("Sbagliato! Grrr!\n");
00047     }
00048
00049     // Turno Giocatore
00050     int dado = lancia_dado(6);
00051     int attacco = dado;
00052
00053     if (g->ha_spada_eroe)
00054       attacco += 2;
00055     else if (g->ha_spada)
00056       attacco += 1;
00057
00058     printf("Tiro: %d (Totale: %d)\n", dado, attacco);
00059
00060     if (attacco > e->colpo_fatale)
00061     {
00062       printf("\nColpo fatale! %s sconfitto!\n", e->nome);
00063       g->monete += e->ricompensa_monete;
00064       printf("Ottieni %d monete.\n", e->ricompensa_monete);
00065       return 1;
00066     }
00067     else
00068     {
00069       printf("Mancato (Serviva > %d).\n", e->colpo_fatale);
00070
00071       // Turno Nemico
00072       if (g->ha_armatura)
00073       {
00074         danno_nemico--;
00075         printf("Armatura assorbe 1 danno.\n");
00076       }
00077       g->punti_vita -= danno_nemico;
00078       printf("%s infligge %d danni!\n", e->nome, danno_nemico);
00079     }
00080   }
00081
00082   printf("\nSEI STATO SCONFITTO!\n");
00083   g->punti_vita = 0;
00084   getchar();
00085   return 0;
```

```
00086 }
00087
00088 void applica_danno_trappola(Giocatore *g, Stanza *trappola)
00089 {
00090   printf("\n!!! TRAPPOLA: %s !!!\n", trappola->nome);
00091
00092   int danno = trappola->danno;
00093
00094   if (g->ha_armatura)
00095   {
00096     danno--;
00097     if (danno < 0)
00098       danno = 0;
00099     printf("Armatura assorbe 1 danno.\n");
00100   }
00101
00102   g->punti_vita -= danno;
00103   printf("Subisci %d danni! (HP: %d)\n", danno, g->punti_vita);
00104 }
00105
00106 int combattimento_boss_finale(Giocatore *g)
00107 {
00108   printf("\n--- SCONTRO FINALE: SIGNORE OSCURO ---\n");
00109   printf("Regole: Scudo > Spada > Magia > Scudo\n");
00110   printf("Vinci 3 round su 5.\n");
00111
00112   int vittorie_eroe = 0;
00113   int vittorie_boss = 0;
00114   int round = 1;
00115   const char *mosse[] = {"", "Scudo", "Magia", "Spada"};
00116
00117   while (vittorie_eroe < 3 && vittorie_boss < 3 && round <= 5)
00118   {
00119     printf("\n--- Round %d (Eroe %d - Boss %d) ---\n", round, vittorie_eroe,
00120            vittorie_boss);
00121     printf("1. Scudo\n2. Magia\n3. Spada\nScelta: ");
00122
00123     int mossa_eroe = leggi_intero();
00124     if (mossa_eroe < 1 || mossa_eroe > 3)
00125       continue;
00126
00127     int mossa_boss = lancia_dado(3);
00128     printf("Tu: %s | Boss: %s\n", mosse[mossa_eroe], mosse[mossa_boss]);
00129
00130     int esito = valuta_vittoria_morra(mossa_eroe, mossa_boss);
00131
00132     if (esito == 0)
00133     {
00134       printf("Pareggio!\n");
00135     }
00136     else if (esito == 1)
00137     {
00138       printf("Vinci il round!\n");
00139       vittorie_eroe++;
00140       round++;
00141     }
00142     else
00143     {
00144       printf("Il Boss vince il round!\n");
00145       vittorie_boss++;
00146       round++;
00147     }
00148   }
00149
00150   return (vittorie_eroe >= 3);
00151 }
```

6.3 Riferimenti per il file src/combattimento.h

Gestione del combattimento e delle interazioni ostili.

```
#include "stanza.h"
#include "giocatore.h"
```

Funzioni

- int inizia_combattimento (Giocatore *g, Stanza *nemico)

  Inizia un combattimento tra il giocatore e un nemico.
- void applica_danno_trappola (Giocatore *g, Stanza *trappola)

  Applica i danni di una trappola al giocatore.
- int combattimento_boss_finale (Giocatore *g)

  Gestisce il boss fight finale contro il Signore Oscuro.

6.3.1 Descrizione dettagliata

Gestione del combattimento e delle interazioni ostili.

Definizione nel file combattimento.h.

6.3.2 Documentazione delle funzioni

6.3.2.1 applica_danno_trappola()

```
void applica_danno_trappola (
            Giocatore * g,
            Stanza * trappola)
```

Applica i danni di una trappola al giocatore.

Calcola i danni subiti considerando eventuali riduzioni dovute all'equipaggiamento (es. armatura).

Parametri

g	Puntatore al giocatore.
trappola	Puntatore alla stanza contenente la trappola.

Definizione alla linea 88 del file combattimento.c.

Referenzia Stanza::danno, Giocatore::ha_armatura, Stanza::nome, e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.3.2.2 combattimento_boss_finale()

```
int combattimento_boss_finale (
            Giocatore * g)
```

Gestisce il boss fight finale contro il Signore Oscuro.

Implementa una versione a turni della Morra Cinese (Sasso-Carta-Forbice) con regole personalizzate. Il giocatore deve vincere 3 round su 5.

Parametri

g	Puntatore al giocatore.

Restituisce

1 se il giocatore vince, 0 se muore.

Definizione alla linea 106 del file combattimento.c.

Referenzia lancia_dado(), leggi_intero(), e valuta_vittoria_morra().

Referenziato da mostra_menu_missione().

6.3.2.3 inizia_combattimento()

```
int inizia_combattimento (
            Giocatore * g,
            Stanza * nemico)
```

Inizia un combattimento tra il giocatore e un nemico.

Gestisce i turni di attacco e difesa, calcola i danni e determina il vincitore del combattimento.

Parametri

g	Puntatore al giocatore.
nemico	Puntatore alla stanza contenente il nemico.

Restituisce

1 se il giocatore vince, 0 se perde (Game Over).

Definizione alla linea 13 del file combattimento.c.

Referenzia Stanza::colpo_fatale, controlla_padovan(), Stanza::danno, Giocatore::ha_armatura, Giocatore::ha_spada, Giocatore::ha_spada_eroe, lancia_dado(), Giocatore::monete, Stanza::nome, Giocatore::punti_vita, e Stanza::ricompensa_monete.

Referenziato da esegui_missione().

6.4 combattimento.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef COMBATTIMENTO_H
00007 #define COMBATTIMENTO_H
00008
00009 #include "stanza.h"
00010 #include "giocatore.h"
00011
00022 int inizia_combattimento(Giocatore *g, Stanza *nemico);
00023
00033 void applica_danno_trappola(Giocatore *g, Stanza *trappola);
00034
00044 int combattimento_boss_finale(Giocatore *g);
00045
00046 #endif
```

6.5 Riferimenti per il file src/dungeon.c

Implementazione della logica del dungeon e delle missioni.

```
#include "dungeon.h"
#include "combattimento.h"
#include "menu.h"
#include "utilita.h"
#include <stdio.h>
#include <string.h>
```

Funzioni

- void mostra_negozio (Giocatore *g)
- Stanza genera_stanza_missione (int tipo_missione, int dado)

  Genera una stanza casuale in base alla missione e al lancio del dado.
- int esegui_missione (Giocatore *g, int tipo_missione, const char *nome_missione)

  Gestisce il loop principale di esecuzione di una missione.

6.5.1 Descrizione dettagliata

Implementazione della logica del dungeon e delle missioni.

Definizione nel file dungeon.c.

6.5.2 Documentazione delle funzioni

6.5.2.1 esegui_missione()

```
int esegui_missione (
            Giocatore * g,
            int tipo_missione,
            const char * nome_missione)
```

Gestisce il loop principale di esecuzione di una missione.

Include la navigazione tra le stanze, il combattimento, il negozio e la gestione degli obiettivi della missione.

Parametri

g	Puntatore al giocatore.
tipo_missione	ID della missione da eseguire.
nome_missione	Nome visualizzato della missione.

Restituisce

1 se la missione è completata con successo, 0 in caso di fallimento o rinuncia.

Definizione alla linea 174 del file dungeon.c.

Referenzia applica_danno_trappola(), Stanza::danno, genera_stanza_missione(), Giocatore::ha_chiave_castello, Giocatore::ha_spada_eroe, inizia_combattimento(), Stanza::is_boss, lancia_dado(), leggi_intero(), Giocatore::monete, mostra_negozio(), Stanza::nome, Giocatore::numero_oggetti, pulisci_schermo(), Giocatore::punti_vita, Stanza::ricompensa_monete, stampa_statistiche_giocatore(), STANZA_NEMICO, STANZA_TRAPPOLA, e Stanza::tipo.

Referenziato da mostra_menu_missione().

6.5.2.2 genera_stanza_missione()

```
Stanza genera_stanza_missione (
            int tipo_missione,
            int dado)
```

Genera una stanza casuale in base alla missione e al lancio del dado.

Parametri

tipo_missione	ID della missione corrente (1, 2 o 3).
dado	Risultato del lancio del dado per determinare il contenuto.

Restituisce

La struttura Stanza generata.

Definizione alla linea 14 del file dungeon.c.

Referenzia Stanza::colpo_fatale, Stanza::danno, Stanza::is_boss, lancia_dado(), Stanza::nome, Stanza::ricompensa_monete, STANZA_NEMICO, STANZA_NESSUNA, STANZA_TRAPPOLA, STANZA_VUOTA, e Stanza::tipo.

Referenziato da esegui_missione().

6.5.2.3 mostra_negozio()

```
void mostra_negozio (
            Giocatore * g)
```

Definizione alla linea 250 del file menu.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_spada, lancia_dado(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, Giocatore::numero_oggetti, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.6 dungeon.c

Vai alla documentazione di questo file.

```
00001
00005 #include "dungeon.h"
00006 #include "combattimento.h"
00007 #include "menu.h"
00008 #include "utilita.h"
00009 #include <stdio.h>
00010 #include <string.h>
00011
00012 void mostra_negozio(Giocatore *g); // Extern
00013
00014 Stanza genera_stanza_missione(int tipo_missione, int dado)
00015 {
00016   Stanza s;
00017   s.tipo = STANZA_NESSUNA;
00018   sprintf(s.nome, "Nulla");
00019   s.danno = 0;
00020   s.colpo_fatale = 0;
00021   s.ricompensa_monete = 0;
```

```
00022   s.is_boss = 0;
00023
00024   // Configurazione entita in base a missione e dado
00025
00026   // Missione 1: Palude
00027   if (tipo_missione == 1)
00028   {
00029     if (dado == 1)
00030     {
00031       sprintf(s.nome, "Cane Selvaggio");
00032       s.tipo = STANZA_NEMICO;
00033       s.colpo_fatale = 2;
00034       s.danno = 1;
00035     }
00036     else if (dado == 2)
00037     {
00038       sprintf(s.nome, "Goblin");
00039       s.tipo = STANZA_NEMICO;
00040       s.colpo_fatale = 3;
00041       s.danno = 2;
00042       s.ricompensa_monete = 2;
00043     }
00044     else if (dado == 3)
00045     {
00046       sprintf(s.nome, "Scheletro");
00047       s.tipo = STANZA_NEMICO;
00048       s.colpo_fatale = 4;
00049       s.danno = 2;
00050       s.ricompensa_monete = 4;
00051     }
00052     else if (dado == 4)
00053     {
00054       sprintf(s.nome, "Orco");
00055       s.tipo = STANZA_NEMICO;
00056       s.colpo_fatale = 3;
00057       s.danno = 4;
00058       s.ricompensa_monete = 6;
00059     }
00060     else if (dado == 5)
00061     {
00062       sprintf(s.nome, "Acquitrino Velenoso");
00063       s.tipo = STANZA_TRAPPOLA;
00064       s.danno = lancia_dado(6);
00065     }
00066     else if (dado == 6)
00067     {
00068       sprintf(s.nome, "Generale Orco");
00069       s.tipo = STANZA_NEMICO;
00070       s.colpo_fatale = 6;
00071       s.danno = 3;
00072       s.ricompensa_monete = 12;
00073       s.is_boss = 1;
00074     }
00075   }
00076   // Missione 2: Magione
00077   else if (tipo_missione == 2)
00078   {
00079     if (dado == 1)
00080     {
00081       sprintf(s.nome, "Botola Buia");
00082       s.tipo = STANZA_TRAPPOLA;
00083       s.danno = 3;
00084     }
00085     else if (dado == 2)
00086     {
00087       sprintf(s.nome, "Pipistrello");
00088       s.tipo = STANZA_NEMICO;
00089       s.colpo_fatale = 2;
00090       s.danno = 2;
00091       s.ricompensa_monete = 1;
00092     }
00093     else if (dado == 3)
00094     {
00095       sprintf(s.nome, "Zombie");
00096       s.tipo = STANZA_NEMICO;
00097       s.colpo_fatale = 3;
00098       s.danno = 2;
00099       s.ricompensa_monete = 2;
00100     }
00101     else if (dado == 4)
00102     {
00103       sprintf(s.nome, "Fantasma");
00104       s.tipo = STANZA_NEMICO;
00105       s.colpo_fatale = 5;
00106       s.danno = 2;
00107       s.ricompensa_monete = 4;
00108     }
```

```
00109     else if (dado == 5)
00110     {
00111       sprintf(s.nome, "Vampiro Superiore");
00112       s.tipo = STANZA_NEMICO;
00113       s.colpo_fatale = 4;
00114       s.danno = 4;
00115       s.ricompensa_monete = 7;
00116       s.is_boss = 1;
00117     }
00118     else if (dado == 6)
00119     {
00120       sprintf(s.nome, "Demone Custode");
00121       s.tipo = STANZA_NEMICO;
00122       s.colpo_fatale = 4;
00123       s.danno = 6;
00124       s.ricompensa_monete = 10;
00125       s.is_boss = 1;
00126     }
00127   }
00128   // Missione 3: Grotta
00129   else if (tipo_missione == 3)
00130   {
00131     if (dado == 1)
00132     {
00133       sprintf(s.nome, "Stanza Vuota");
00134       s.tipo = STANZA_VUOTA;
00135     }
00136     else if (dado == 2)
00137     {
00138       sprintf(s.nome, "Cristalli Cadenti");
00139       s.tipo = STANZA_TRAPPOLA;
00140       s.danno = 2;
00141     }
00142     else if (dado == 3)
00143     {
00144       sprintf(s.nome, "Ponte Pericolante");
00145       s.tipo = STANZA_TRAPPOLA;
00146       s.danno = 0;
00147     }
00148     else if (dado == 4)
00149     {
00150       sprintf(s.nome, "Forziere Misterioso");
00151       s.tipo = STANZA_TRAPPOLA;
00152       s.danno = 2;
00153       s.ricompensa_monete = 10;
00154     }
00155     else if (dado == 5)
00156     {
00157       sprintf(s.nome, "Rupe scoscesa");
00158       s.tipo = STANZA_TRAPPOLA;
00159       s.danno = lancia_dado(6);
00160     }
00161     else if (dado == 6)
00162     {
00163       sprintf(s.nome, "Drago Antico");
00164       s.tipo = STANZA_NEMICO;
00165       s.colpo_fatale = 5;
00166       s.danno = 10;
00167       s.ricompensa_monete = 12;
00168       s.is_boss = 1;
00169     }
00170   }
00171   return s;
00172 }
00173
00174 int esegui_missione(Giocatore *g, int tipo_missione, const char *nome_missione)
00175 {
00176   int target = (tipo_missione == 1) ? 3 : 1;
00177   int progress = 0;
00178   int n_stanze = 0;
00179   int completata = 0;
00180
00181   do
00182   {
00183     pulisci_schermo();
00184     printf("=== %s ===\n", nome_missione);
00185
00186     // Obiettivi missione 'Palude'
00187     if (tipo_missione == 1)
00188       printf("Obiettivo: Sconfiggi %d Generali Orco (%d/%d)\n", target, progress, target);
00189
00190     // Obiettivi missione 'Magione'
00191     else if (tipo_missione == 2)
00192       printf("Obiettivo: Sconfiggi Vampiro Superiore (%d/%d) e prendi Chiave (%d/1)\n", progress,
      target, g->ha_chiave_castello);
00193
00194     // Obiettivi missione 'Grotta'
```

```
00195     else if (tipo_missione == 3)
00196       printf("Obiettivo: Sconfiggi Drago (%d/1) e prendi Spada (%d/1)\n", progress, g->ha_spada_eroe);
00197
00198     // Stanze esplorate
00199     printf("Stanze Esplorate: %d/10\n\n", n_stanze);
00200
00201     // Menu azioni
00202     if (n_stanze < 10)
00203       printf("1. Esplora\n");
00204     printf("2. Negozio\n3. Inventario\n4. Torna al Villaggio (Completa Obbiettivo oppure Paga 50
      monete)\n\n");
00205     printf("Seleziona una delle opzioni del menu [1-4]: ");
00206
00207     // Gestione scelta utente
00208     int scelta = leggi_intero();
00209     if (scelta == 1 && n_stanze < 10)
00210     {
00211       n_stanze++;
00212       int dado = lancia_dado(6);
00213
00214       // Forza spawn boss se fine dungeon
00215       // Palude: Generale Orco
00216       if (n_stanze >= 8 && !completata && tipo_missione == 1)
00217         dado = 6;
00218
00219       // Magione: Demone Custode + Vampiro
00220       if (n_stanze >= 9 && !completata && tipo_missione == 2)
00221       {
00222         if (g->ha_chiave_castello == 0)
00223           dado = 5;
00224         else if (g->ha_chiave_castello == 1)
00225           dado = 6;
00226       }
00227       else if (completata && tipo_missione == 2)
00228         dado = lancia_dado(4); // evita di far spawnare di nuovo i boss
00229
00230       // Grotta: Drago Antico
00231       if (n_stanze >= 10 && !completata && tipo_missione == 3)
00232         dado = 6;
00233       else if (completata && tipo_missione == 3)
00234         dado = lancia_dado(5); // evita di far spawnare di nuovo il drago
00235
00236       Stanza s = genera_stanza_missione(tipo_missione, dado);
00237       printf("Incontri: %s\n", s.nome);
00238
00239       if (s.tipo == STANZA_NEMICO)
00240       {
00241         if (inizia_combattimento(g, &s))
00242         {
00243           // Update progressi missione 'Palude'
00244           if (tipo_missione == 1 && s.is_boss)
00245             progress++;
00246
00247           // Update progressi missione 'Magione'
00248           else if (tipo_missione == 2 && s.is_boss)
00249           {
00250             if (strcmp(s.nome, "Vampiro Superiore") == 0)
00251             {
00252               progress++;
00253             }
00254             else if (strcmp(s.nome, "Demone Custode") == 0)
00255             {
00256               printf("Trovata Chiave del Castello!\n");
00257               g->ha_chiave_castello = 1;
00258               g->numero_oggetti++;
00259             }
00260           }
00261           // Update progressi missione 'Grotta'
00262           else if (tipo_missione == 3 && s.is_boss)
00263           {
00264             printf("Trovata Spada Eroe!\n");
00265             g->ha_spada_eroe = 1;
00266             g->numero_oggetti++;
00267             progress++;
00268           }
00269         }
00270         else
00271           return 0; // Game Over
00272       }
00273       else if (s.tipo == STANZA_TRAPPOLA)
00274       {
00275         if (strcmp(s.nome, "Forziere Misterioso") == 0)
00276         {
00277           int chance = lancia_dado(2);
00278           printf("Apri il forziere... ");
00279           if (chance == 1)
00280           {
```

```
00281             printf("Trappola!\n");
00282             applica_danno_trappola(g, &s);
00283           }
00284           else
00285           {
00286             printf("Monete! (+%d)\n", s.ricompensa_monete);
00287             g->monete += s.ricompensa_monete;
00288           }
00289         }
00290         else if (strcmp(s.nome, "Ponte Pericolante") == 0)
00291         {
00292           g->monete -= 3;
00293           if (g->monete < 0)
00294             g->monete = 0;
00295           printf("Caduto dal ponte! Perdi 3 monete. (Monete: %d)\n", g->monete);
00296         }
00297         else if (s.danno > 0)
00298           applica_danno_trappola(g, &s);
00299       }
00300
00301       // Check completamento missione 'Palude'
00302       if (tipo_missione == 1 && progress >= target)
00303         completata = 1;
00304       // Check completamento missione 'Magione'
00305       if (tipo_missione == 2 && progress >= target && g->ha_chiave_castello == 1)
00306         completata = 1;
00307       // Check completamento missione 'Grotta'
00308       if (tipo_missione == 3 && progress >= target && g->ha_spada_eroe == 1)
00309         completata = 1;
00310
00311       if (g->punti_vita <= 0)
00312         return 0;
00313
00314       printf("Premi Invio...");
00315       getchar();
00316     }
00317
00318     // Scelta negozio
00319     else if (scelta == 2)
00320       mostra_negozio(g);
00321
00322     // Scelta inventario
00323     else if (scelta == 3)
00324     {
00325       stampa_statistiche_giocatore(g);
00326       getchar();
00327     }
00328
00329     // Scelta fuga
00330     else if (scelta == 4)
00331     {
00332       if (completata)
00333       {
00334         printf("\nMISSIONE COMPIUTA!\n");
00335         getchar();
00336         return 1;
00337       }
00338       if (g->monete >= 50)
00339       {
00340         g->monete -= 50;
00341         return 0;
00342       }
00343       printf("Non hai 50 monete!\n");
00344       getchar();
00345     }
00346   } while (1);
00347 }
```

6.7 Riferimenti per il file src/dungeon.h

Gestione della generazione e del flusso del dungeon.

```
#include "stanza.h"
#include "giocatore.h"
```

Funzioni

- Stanza genera_stanza_missione (int tipo_missione, int dado)

  Genera una stanza casuale in base alla missione e al lancio del dado.
- int esegui_missione (Giocatore *g, int tipo_missione, const char *nome_missione)

  Gestisce il loop principale di esecuzione di una missione.

6.7.1 Descrizione dettagliata

Gestione della generazione e del flusso del dungeon.

Definizione nel file dungeon.h.

6.7.2 Documentazione delle funzioni

6.7.2.1 esegui_missione()

```
int esegui_missione (
            Giocatore * g,
            int tipo_missione,
            const char * nome_missione)
```

Gestisce il loop principale di esecuzione di una missione.

Include la navigazione tra le stanze, il combattimento, il negozio e la gestione degli obiettivi della missione.

Parametri

g	Puntatore al giocatore.
tipo_missione	ID della missione da eseguire.
nome_missione	Nome visualizzato della missione.

Restituisce

1 se la missione è completata con successo, 0 in caso di fallimento o rinuncia.

Definizione alla linea 174 del file dungeon.c.

Referenzia applica_danno_trappola(), Stanza::danno, genera_stanza_missione(), Giocatore::ha_chiave_castello, Giocatore::ha_spada_eroe, inizia_combattimento(), Stanza::is_boss, lancia_dado(), leggi_intero(), Giocatore::monete, mostra_negozio(), Stanza::nome, Giocatore::numero_oggetti, pulisci_schermo(), Giocatore::punti_vita, Stanza::ricompensa_monete, stampa_statistiche_giocatore(), STANZA_NEMICO, STANZA_TRAPPOLA, e Stanza::tipo.

Referenziato da mostra_menu_missione().

6.7.2.2 genera_stanza_missione()

```
Stanza genera_stanza_missione (
            int tipo_missione,
            int dado)
```


Genera una stanza casuale in base alla missione e al lancio del dado.

Parametri

tipo_missione	ID della missione corrente (1, 2 o 3).
dado	Risultato del lancio del dado per determinare il contenuto.

Restituisce

La struttura Stanza generata.

Definizione alla linea 14 del file dungeon.c.

Referenzia Stanza::colpo_fatale, Stanza::danno, Stanza::is_boss, lancia_dado(), Stanza::nome, Stanza::ricompensa_monete, STANZA_NEMICO, STANZA_NESSUNA, STANZA_TRAPPOLA, STANZA_VUOTA, e Stanza::tipo.

Referenziato da esegui_missione().

6.8 dungeon.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef DUNGEON_H
00007 #define DUNGEON_H
00008
00009 #include "stanza.h"
00010 #include "giocatore.h"
00011
00019 Stanza genera_stanza_missione(int tipo_missione, int dado);
00020
00032 int esegui_missione(Giocatore *g, int tipo_missione, const char *nome_missione);
00033
00034 #endif
```

6.9 Riferimenti per il file src/giocatore.c

Implementazione delle funzioni relative al giocatore.

```
#include "giocatore.h"
#include <stdio.h>
```

Funzioni

- void inizializza_giocatore (Giocatore *g)

  Inizializza un nuovo giocatore con valori di default.
- void stampa_statistiche_giocatore (const Giocatore *g)

  Stampa a video le statistiche e l'inventario del giocatore.

6.9.1 Descrizione dettagliata

Implementazione delle funzioni relative al giocatore.

Definizione nel file giocatore.c.

6.9.2 Documentazione delle funzioni

6.9.2.1 inizializza_giocatore()

```
void inizializza_giocatore (
            Giocatore * g)
```

Inizializza un nuovo giocatore con valori di default.

Imposta HP a 20, monete a 0 e resetta missioni e inventario.

Parametri

g	Puntatore alla struttura Giocatore da inizializzare.

Definizione alla linea 9 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da mostra_menu_principale().

6.9.2.2 stampa_statistiche_giocatore()

```
void stampa_statistiche_giocatore (
            const Giocatore * g)
```

Stampa a video le statistiche e l'inventario del giocatore.

Mostra HP, monete, missioni completate ed equipaggiamento attuale.

Parametri

g	Puntatore alla struttura Giocatore costante.

Definizione alla linea 24 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da esegui_missione(), e mostra_menu_villaggio().

6.10 giocatore.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "giocatore.h"
00007 #include <stdio.h>
00008
00009 void inizializza_giocatore(Giocatore *g)
00010 {
00011   g->punti_vita = 20;
00012   g->max_punti_vita = 20;
00013   g->monete = 0;
00014   g->numero_oggetti = 0;
00015   g->missione_palude = 0;
00016   g->missione_magione = 0;
00017   g->missione_grotta = 0;
00018   g->ha_spada = 0;
00019   g->ha_armatura = 0;
00020   g->ha_spada_eroe = 0;
00021   g->ha_chiave_castello = 0;
00022 }
00023
00024 void stampa_statistiche_giocatore(const Giocatore *g)
00025 {
00026   printf("\n=== Inventario Giocatore ===\n");
00027
00028   printf("Punti Vita: %d / %d\n", g->punti_vita, g->max_punti_vita);
00029   printf("Monete: %d\n", g->monete);
00030   printf("Oggetti: %d\n", g->numero_oggetti);
00031   printf("Missioni Completate: %d\n", g->missione_palude + g->missione_magione + g->missione_grotta);
00032   printf("Equipaggiamento:\n");
00033   if (g->ha_spada || g->ha_spada_eroe)
00034     printf("- Spada: %s (+%d attacco)\n",
00035            g->ha_spada_eroe ? "Spada dell'Eroe" : "Spada Ferro",
00036            g->ha_spada_eroe ? 2 : 1);
00037   else
00038     printf("- Nessuna arma\n");
00039   if (g->ha_armatura)
00040     printf("- Armatura: Si (-1 danno subito)\n");
00041   else
00042     printf("- Nessuna armatura\n");
00043   if (g->ha_chiave_castello)
00044     printf("- Oggetti Chiave: Chiave del Castello\n");
00045
00046   printf("============================\n");
00047 }
```

6.11 Riferimenti per il file src/giocatore.h

Definizione della struttura Giocatore e funzioni correlate.

Strutture dati

- struct Giocatore

  Struttura che rappresenta lo stato del giocatore.

Funzioni

- void inizializza_giocatore (Giocatore *g)

  Inizializza un nuovo giocatore con valori di default.
- void stampa_statistiche_giocatore (const Giocatore *g)

  Stampa a video le statistiche e l'inventario del giocatore.

6.11.1 Descrizione dettagliata

Definizione della struttura Giocatore e funzioni correlate.

Definizione nel file giocatore.h.

6.11.2 Documentazione delle funzioni

6.11.2.1 inizializza_giocatore()

```
void inizializza_giocatore (
            Giocatore * g)
```

Inizializza un nuovo giocatore con valori di default.

Imposta HP a 20, monete a 0 e resetta missioni e inventario.

Parametri

g	Puntatore alla struttura Giocatore da inizializzare.

Definizione alla linea 9 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da mostra_menu_principale().

6.11.2.2 stampa_statistiche_giocatore()

```
void stampa_statistiche_giocatore (
            const Giocatore * g)
```

Stampa a video le statistiche e l'inventario del giocatore.

Mostra HP, monete, missioni completate ed equipaggiamento attuale.

Parametri

g	Puntatore alla struttura Giocatore costante.

Definizione alla linea 24 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da esegui_missione(), e mostra_menu_villaggio().

6.12 giocatore.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef GIOCATORE_H
00007 #define GIOCATORE_H
00008
00012 typedef struct
00013 {
00014   int punti_vita;
00015   int max_punti_vita;
00016   int monete;
00017   int numero_oggetti;
00018
00019   // Flag per missioni completate
00020   int missione_palude;
00021   int missione_magione;
00022   int missione_grotta;
00023
00024   // Flag per oggetti chiave
00025   int ha_spada;
00026   int ha_armatura;
00027   int ha_spada_eroe;
00028   int ha_chiave_castello;
00029 } Giocatore;
00030
00038 void inizializza_giocatore(Giocatore *g);
00039
00047 void stampa_statistiche_giocatore(const Giocatore *g);
00048
00049 #endif
```

6.13 Riferimenti per il file src/main.c

Entry point del gioco CDungeon.

```
#include "giocatore.h"
#include "menu.h"
#include "salvataggio.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Funzioni

- int main ()

  Funzione principale del programma.

6.13.1 Descrizione dettagliata

Entry point del gioco CDungeon.

Definizione nel file main.c.

6.13.2 Documentazione delle funzioni

6.13.2.1 main()

```
int main ()
```

Funzione principale del programma.

Inizializza il generatore di numeri casuali, carica i salvataggi e avvia il loop principale del menu.

Restituisce

0 se il programma termina correttamente.

Definizione alla linea 20 del file main.c.

Referenzia carica_salvataggi_da_file(), libera_salvataggi(), e mostra_menu_principale().

6.14 main.c

Vai alla documentazione di questo file.

```
00001
00005 #include "giocatore.h"
00006 #include "menu.h"
00007 #include "salvataggio.h"
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include <time.h>
00011
00020 int main(){
00021   NodoSalvataggio *lista_salvataggi = NULL;
00022   Giocatore giocatore_corrente;
00023
00024   // Carica i salvataggi esistenti da file
00025   carica_salvataggi_da_file(&lista_salvataggi);
00026
00027   // Inizializza seed random
00028   srand(time(NULL));
00029
00030   printf("Benvenuto in CDungeon!\n");
00031   getchar();
00032   mostra_menu_principale(&lista_salvataggi, &giocatore_corrente);
00033
00034   // Pulizia della memoria alla chiusura
00035   libera_salvataggi(lista_salvataggi);
00036
00037   return 0;
00038 }
```

6.15 Riferimenti per il file src/menu.c

Implementazione dei menu e dell'interfaccia utente.

```
#include "menu.h"
#include "combattimento.h"
#include "dungeon.h"
#include "giocatore.h"
#include "salvataggio.h"
#include "utilita.h"
#include <ctype.h>
#include <stdio.h>
#include <string.h>
```

Funzioni

- void gestisci_trucchi (Giocatore *g, NodoSalvataggio **lista)
- void mostra_menu_salvataggi (Giocatore *g, NodoSalvataggio **lista)
- void mostra_menu_principale (NodoSalvataggio **lista, Giocatore *g)

  Mostra il menu principale del gioco.
- void mostra_menu_villaggio (Giocatore *g, NodoSalvataggio **lista)

  Mostra il menu del villaggio.
- void mostra_menu_missione (Giocatore *g)

  Mostra il menu di selezione delle missioni.
- void mostra_negozio (Giocatore *g)

  Mostra il negozio del villaggio.

6.15.1 Descrizione dettagliata

Implementazione dei menu e dell'interfaccia utente.

Definizione nel file menu.c.

6.15.2 Documentazione delle funzioni

6.15.2.1 gestisci_trucchi()

```
void gestisci_trucchi (
            Giocatore * g,
            NodoSalvataggio ** lista)
```

Definizione alla linea 308 del file menu.c.

Referenzia carica_salvataggio(), NodoSalvataggio::dati_giocatore, leggi_intero(), Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, pulisci_schermo(), Giocatore::punti_vita, salva_tutto_su_file(), e stampa_salvataggi().

Referenziato da mostra_menu_principale().

6.15.2.2 mostra_menu_missione()

```
void mostra_menu_missione (
            Giocatore * g)
```

Mostra il menu di selezione delle missioni.

Elenca le missioni disponibili e avviabili dal giocatore.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 177 del file menu.c.

Referenzia combattimento_boss_finale(), esegui_missione(), leggi_intero(), Giocatore::missione_grotta, Giocatore::missione_magion
Giocatore::missione_palude, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da mostra_menu_villaggio().

6.15.2.3 mostra_menu_principale()

```
void mostra_menu_principale (
            NodoSalvataggio ** lista_salvataggi,
            Giocatore * giocatore_corrente)
```

Mostra il menu principale del gioco.

Permette di iniziare una nuova partita, caricare un salvataggio o uscire. Contiene anche la gestione dei trucchi tramite codice segreto.

Parametri

lista_salvataggi	Puntatore alla lista dei salvataggi.
giocatore_corrente	Puntatore alla struttura dati del giocatore corrente.

Definizione alla linea 20 del file menu.c.

Referenzia gestisci_trucchi(), inizializza_giocatore(), leggi_input_char(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e pulisci_schermo().

Referenziato da main(), e mostra_menu_villaggio().

6.15.2.4 mostra_menu_salvataggi()

```
void mostra_menu_salvataggi (
            Giocatore * g,
            NodoSalvataggio ** lista)
```

Definizione alla linea 82 del file menu.c.

Referenzia carica_salvataggio(), NodoSalvataggio::dati_giocatore, elimina_salvataggio(), leggi_input_char(), leggi_intero(), mostra_menu_villaggio(), e stampa_salvataggi().

Referenziato da mostra_menu_principale().

6.15.2.5 mostra_menu_villaggio()

```
void mostra_menu_villaggio (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Mostra il menu del villaggio.

Hub principale per il giocatore tra le missioni. Permette di curarsi, gestire l'inventario, salvare o intraprendere nuove missioni.

Parametri

g	Puntatore al giocatore.

lista_salvataggi	Puntatore alla lista dei salvataggi per il salvataggio manuale.

Definizione alla linea 130 del file menu.c.

Referenzia aggiungi_salvataggio(), leggi_input_char(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, mostra_menu_missione(), mostra_menu_principale(), pulisci_schermo(), Giocatore::punti_vita, e stampa_statistiche_giocatore().

Referenziato da mostra_menu_principale(), e mostra_menu_salvataggi().

6.15.2.6 mostra_negozio()

```
void mostra_negozio (
            Giocatore * g)
```

Mostra il negozio del villaggio.

Permette al giocatore di acquistare oggetti e cure in cambio di monete.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 250 del file menu.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_spada, lancia_dado(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, Giocatore::numero_oggetti, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.16 menu.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "menu.h"
00007 #include "combattimento.h"
00008 #include "dungeon.h"
00009 #include "giocatore.h"
00010 #include "salvataggio.h"
00011 #include "utilita.h"
00012 #include <ctype.h>
00013 #include <stdio.h>
00014 #include <string.h>
00015
00016 // Forward declarations
00017 void gestisci_trucchi(Giocatore *g, NodoSalvataggio **lista);
00018 void mostra_menu_salvataggi(Giocatore *g, NodoSalvataggio **lista);
00019
00020 void mostra_menu_principale(NodoSalvataggio **lista, Giocatore *g)
00021 {
00022   char in;
00023   const char *k_code = "wwssadadba ";
00024   int k_idx = 0;
00025   int cheat = 0;
00026
00027   do
00028   {
00029     pulisci_schermo();
00030     printf("=== CDungeon ===\n\n");
00031
```

```
00032     printf("1. Nuova Partita\n2. Carica Salvataggio\n");
00033     if (cheat)
00034       printf("3. Trucchi\n4. Esci\n");
00035     else
00036       printf("3. Esci\n\n");
00037
00038     printf("Seleziona una delle opzioni del menu [1-%d]: ", cheat ? 4 : 3);
00039
00040     in = leggi_input_char();
00041     if (isdigit(in))
00042     {
00043       // Converto il carattere in un intero
00044       int scelta = in - '0';
00045
00046       // Gestione delle opzioni del menu
00047       if (scelta == 1)
00048       {
00049         inizializza_giocatore(g);
00050         mostra_menu_villaggio(g, lista);
00051       }
00052       else if (scelta == 2)
00053       {
00054         mostra_menu_salvataggi(g, lista);
00055       }
00056       else if (cheat && scelta == 3)
00057         gestisci_trucchi(g, lista);
00058       else
00059         return;
00060     }
00061     else
00062     {
00063       if (in == k_code[k_idx])
00064       {
00065         k_idx++;
00066         if (k_idx == strlen(k_code))
00067         {
00068           cheat = 1;
00069           k_idx = 0;
00070           printf("Trucchi Attivi!\n");
00071           getchar();
00072         }
00073       }
00074       else
00075       {
00076         k_idx = (in == 'w') ? 1 : 0;
00077       }
00078     }
00079   } while (1);
00080 }
00081
00082 void mostra_menu_salvataggi(Giocatore *g, NodoSalvataggio **lista)
00083 {
00084   stampa_salvataggi(*lista);
00085
00086   if (!*lista)
00087   {
00088     printf("Premi Invio per tornare al menu principale...");
00089     getchar();
00090     return;
00091   }
00092
00093   printf("ID: ");
00094   int id = leggi_intero();
00095   if (id)
00096   {
00097     NodoSalvataggio *salv = carica_salvataggio(*lista, id);
00098     if (salv)
00099     {
00100       printf("\nScegli operazione:\n");
00101       printf("1. Carica\n2. Elimina\n0. Annulla\nScelta: ");
00102       int op = leggi_intero();
00103       if (op == 1)
00104       {
00105         *g = salv->dati_giocatore;
00106         printf("Salvataggio caricato.\n");
00107         getchar();
00108         mostra_menu_villaggio(g, lista);
00109       }
00110       else if (op == 2)
00111       {
00112         printf("Sei sicuro di voler eliminare il salvataggio? (s/n): ");
00113         char conferma = leggi_input_char();
00114         if (conferma == 's')
00115         {
00116           elimina_salvataggio(lista, id);
00117         }
00118         getchar(); // Attendi invio per leggere messaggio
```

```
00119       }
00120     }
00121     else
00122     {
00123       printf("Salvataggio non trovato.\n");
00124       getchar();
00125       // mostra_menu_salvataggi(g, lista);
00126     }
00127   }
00128 }
00129
00130 void mostra_menu_villaggio(Giocatore *g, NodoSalvataggio **lista)
00131 {
00132   do
00133   {
00134     if (g->punti_vita <= 0)
00135     {
00136       mostra_menu_principale(lista, g);
00137       return;
00138     }
00139
00140     pulisci_schermo();
00141     printf("=== Menù Villaggio ===\nHP: %d/%d | Monete: %d\n", g->punti_vita, g->max_punti_vita,
    g->monete);
00142
00143     printf("\n1. Intraprendi una missione\n2. Riposati\n3. Inventario\n4. Salva la partita\n5.
    Esci\n\n");
00144
00145     printf("Seleziona una delle opzioni del menù [1-5]: ");
00146
00147     // Gestione delle opzioni del menu
00148     int s = leggi_intero();
00149     switch (s)
00150     {
00151     case 1:
00152       mostra_menu_missione(g);
00153       break;
00154     case 2:
00155       g->punti_vita = g->max_punti_vita;
00156       printf("Riposato.\n");
00157       getchar();
00158       break;
00159     case 3:
00160       stampa_statistiche_giocatore(g);
00161       getchar();
00162       break;
00163     case 4:
00164       aggiungi_salvataggio(lista, g);
00165       getchar();
00166       break;
00167     case 5:
00168       printf("Stai uscendo dal gioco, ricodati di salvare la partita per non perdere i progressi!. Sei
    sicuro di voler uscire? (s/n): ");
00169       char conferma = leggi_input_char();
00170       if (conferma == 's')
00171         return;
00172       break;
00173     }
00174   } while (1);
00175 }
00176
00177 void mostra_menu_missione(Giocatore *g)
00178 {
00179   do
00180   {
00181     pulisci_schermo();
00182     printf("=== Menù di Selezione Missioni ===\n\n");
00183
00184     // Flag di controllo di quali missioni sono completate
00185     int f1 = g->missione_palude;
00186     int f2 = g->missione_magione;
00187     int f3 = g->missione_grotta;
00188
00189     if (!f1)
00190       printf("1. Palude Putrescente\n");
00191     if (!f2)
00192       printf("2. Magione Infestata\n");
00193     if (!f3)
00194       printf("3. Grotta di Cristallo\n");
00195     if (f1 && f2 && f3)
00196       printf("4. Castello del Signore Oscuro (Boss Finale)\n");
00197
00198     printf("0. Indietro\n\n");
00199
00200     if (!f1 || !f2 || !f3)
00201       printf("Seleziona una delle opzioni del menu [1-3]: ");
00202     else if (f1 && f2 && f3)
```

```
00203       printf("Seleziona una delle opzioni del menu [4]: ");
00204
00205     int s = leggi_intero();
00206     if (s == 0)
00207       return;
00208
00209     // Esegui la missione selezionata
00210     if (s == 1 && !f1)
00211     {
00212       if (esegui_missione(g, 1, "Palude"))
00213         g->missione_palude = 1;
00214       else if (g->punti_vita <= 0)
00215         return;
00216     }
00217     else if (s == 2 && !f2)
00218     {
00219       if (esegui_missione(g, 2, "Magione"))
00220         g->missione_magione = 1;
00221       else if (g->punti_vita <= 0)
00222         return;
00223     }
00224     else if (s == 3 && !f3)
00225     {
00226       if (esegui_missione(g, 3, "Grotta"))
00227         g->missione_grotta = 1;
00228       else if (g->punti_vita <= 0)
00229         return;
00230     }
00231     else if (s == 4 && f1 && f2 && f3)
00232     {
00233       if (combattimento_boss_finale(g))
00234       {
00235         printf("\nHAI VINTO IL GIOCO!\n");
00236         getchar();
00237         return;
00238       }
00239       else
00240       {
00241         printf("\nSEI STATO SCONFITTO!\n");
00242         g->punti_vita = 0;
00243         getchar();
00244         return;
00245       }
00246     }
00247   } while (1);
00248 }
00249
00250 void mostra_negozio(Giocatore *g)
00251 {
00252   do
00253   {
00254     pulisci_schermo();
00255     printf("=== Negozio (Monete: %d) ===\n", g->monete);
00256     printf("Oggetto \t| Descrizione \t\t\t| Costo\n");
00257     printf("----------------|-------------------------------|-------\n");
00258     printf("1. Cura \t| Ripristina punti vita (1-6) \t| 4 monete\n");
00259     if (!g->ha_spada)
00260       printf("2. Spada \t| Aumenta attacco (+1) \t\t| 5 monete\n");
00261     if (!g->ha_armatura)
00262       printf("3. Armatura \t| Riduce danno subito (-1) \t| 10 monete\n");
00263     printf("----------------|-------------------------------|-------\n");
00264     printf("0. Esci\n\n");
00265     printf("Seleziona un oggetto da acquistare: ");
00266
00267     // Gestione delle opzioni del menu
00268     int s = leggi_intero();
00269     if (s == 0)
00270       return;
00271
00272     if (s == 1 && g->monete >= 4)
00273     {
00274       g->monete -= 4;
00275
00276       int punti_ripristinati = lancia_dado(6);
00277       g->punti_vita += punti_ripristinati;
00278       if (g->punti_vita > g->max_punti_vita)
00279         g->punti_vita = g->max_punti_vita;
00280       printf("Ripristinati %d punti vita.\n", punti_ripristinati);
00281     }
00282     else if (s == 2 && g->monete >= 5)
00283     {
00284       if (!g->ha_spada)
00285       {
00286         g->monete -= 5;
00287         g->ha_spada = 1;
00288         g->numero_oggetti++;
00289         printf("Spada acquistata.\n");
```

```
00290       }
00291     }
00292     else if (s == 3 && g->monete >= 10)
00293     {
00294       if (!g->ha_armatura)
00295       {
00296         g->monete -= 10;
00297         g->ha_armatura = 1;
00298         g->numero_oggetti++;
00299         printf("Armatura acquistata.\n");
00300       }
00301     }
00302     else
00303       printf("Non hai abbastanza monete.\n");
00304     getchar();
00305   } while (1);
00306 }
00307
00308 void gestisci_trucchi(Giocatore *g, NodoSalvataggio **lista)
00309 {
00310   do
00311   {
00312     pulisci_schermo();
00313     stampa_salvataggi(*lista);
00314     printf("Inserisci ID salvataggio (0 per uscire): ");
00315     int id = leggi_intero();
00316     if (id == 0)
00317       return;
00318
00319     NodoSalvataggio *salv = carica_salvataggio(*lista, id);
00320     if (!salv)
00321     {
00322       printf("Salvataggio non trovato.\n");
00323       getchar();
00324       continue;
00325     }
00326
00327     // Copia i dati dal salvataggio per lavorarci
00328     *g = salv->dati_giocatore;
00329
00330     printf("\n--- TRUCCHI (Salvataggio ID: %d) ---\n", id);
00331     printf("1. Max Monete (999)\n");
00332     printf("2. Max HP (999)\n");
00333     printf("3. Sblocca Tutte le Missioni\n");
00334     printf("4. Torna indietro\n");
00335     printf("Scelta: ");
00336
00337     int s = leggi_intero();
00338     int modificato = 0;
00339
00340     if (s == 1)
00341     {
00342       salv->dati_giocatore.monete = 999;
00343       modificato = 1;
00344       printf("Monete impostate a 999!\n");
00345     }
00346     else if (s == 2)
00347     {
00348       salv->dati_giocatore.max_punti_vita = 999;
00349       salv->dati_giocatore.punti_vita = 999;
00350       modificato = 1;
00351       printf("HP impostati a 999!\n");
00352     }
00353     else if (s == 3)
00354     {
00355       salv->dati_giocatore.missione_palude = 1;
00356       salv->dati_giocatore.missione_magione = 1;
00357       salv->dati_giocatore.missione_grotta = 1;
00358       modificato = 1;
00359       printf("Tutte le missioni sbloccate!\n");
00360     }
00361     else if (s == 4)
00362     {
00363       continue;
00364     }
00365
00366     if (modificato)
00367     {
00368       salva_tutto_su_file(*lista);
00369       printf("Salvataggio aggiornato su file.\n");
00370       // Aggiorna anche g per riflettere le modifiche se l'utente carica subito
00371       *g = salv->dati_giocatore;
00372     }
00373     getchar();
00374
00375   } while (1);
00376 }
```

6.17 Riferimenti per il file src/menu.h

Gestione dei menu e dell'interfaccia utente.

```
#include "giocatore.h"
#include "salvataggio.h"
```

Funzioni

- void mostra_menu_principale (NodoSalvataggio **lista_salvataggi, Giocatore *giocatore_corrente)
  Mostra il menu principale del gioco.
- void mostra_menu_villaggio (Giocatore *g, NodoSalvataggio **lista_salvataggi)
  Mostra il menu del villaggio.
- void mostra_menu_missione (Giocatore *g)
  Mostra il menu di selezione delle missioni.
- void mostra_negozio (Giocatore *g)
  Mostra il negozio del villaggio.

6.17.1 Descrizione dettagliata

Gestione dei menu e dell'interfaccia utente.

Definizione nel file menu.h.

6.17.2 Documentazione delle funzioni

6.17.2.1 mostra_menu_missione()

```
void mostra_menu_missione (
            Giocatore * g)
```

Mostra il menu di selezione delle missioni.

Elenca le missioni disponibili e avviabili dal giocatore.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 177 del file menu.c.

Referenzia combattimento_boss_finale(), esegui_missione(), leggi_intero(), Giocatore::missione_grotta, Giocatore::missione_magion
Giocatore::missione_palude, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da mostra_menu_villaggio().

6.17.2.2 mostra_menu_principale()


```
void mostra_menu_principale (
            NodoSalvataggio ** lista_salvataggi,
            Giocatore * giocatore_corrente)
```

Parametri

lista_salvataggi	Puntatore alla lista dei salvataggi.
giocatore_corrente	Puntatore alla struttura dati del giocatore corrente.

Definizione alla linea 20 del file menu.c.

Referenzia gestisci_trucchi(), inizializza_giocatore(), leggi_input_char(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e pulisci_schermo().

Referenziato da main(), e mostra_menu_villaggio().

6.17.2.3 mostra_menu_villaggio()

```
void mostra_menu_villaggio (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Mostra il menu del villaggio.

Hub principale per il giocatore tra le missioni. Permette di curarsi, gestire l'inventario, salvare o intraprendere nuove missioni.

Parametri

g	Puntatore al giocatore.
lista_salvataggi	Puntatore alla lista dei salvataggi per il salvataggio manuale.

Definizione alla linea 130 del file menu.c.

Referenzia aggiungi_salvataggio(), leggi_input_char(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, mostra_menu_missione(), mostra_menu_principale(), pulisci_schermo(), Giocatore::punti_vita, e stampa_statistiche_giocatore().

Referenziato da mostra_menu_principale(), e mostra_menu_salvataggi().

6.17.2.4 mostra_negozio()

```
void mostra_negozio (
            Giocatore * g)
```

Mostra il negozio del villaggio.

Permette al giocatore di acquistare oggetti e cure in cambio di monete.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 250 del file menu.c.

6.18 menu.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef MENU_H
00007 #define MENU_H
00008
00009 #include "giocatore.h"
00010 #include "salvataggio.h"
00011
00021 void mostra_menu_principale(NodoSalvataggio **lista_salvataggi, Giocatore *giocatore_corrente);
00022
00032 void mostra_menu_villaggio(Giocatore *g, NodoSalvataggio **lista_salvataggi);
00033
00041 void mostra_menu_missione(Giocatore *g);
00042
00050 void mostra_negozio(Giocatore *g);
00051
00052 #endif
```

6.19 Riferimenti per il file src/salvataggio.c

Implementazione del sistema di persistenza dei dati.

```
#include "salvataggio.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

Funzioni

- void ottieni_data_corrente (char *buffer, size_t size)
- void salva_tutto_su_file (NodoSalvataggio *testa)

  Salva l'intera lista dei salvataggi su file.
- void aggiungi_salvataggio (NodoSalvataggio **testa, Giocatore *g)

  Aggiunge un nuovo salvataggio alla lista e al file.
- NodoSalvataggio * carica_salvataggio (NodoSalvataggio *testa, int id)

  Cerca un salvataggio nella lista tramite ID.
- void elimina_salvataggio (NodoSalvataggio **testa, int id)

  Elimina un salvataggio dalla lista e dal file.
- void stampa_salvataggi (NodoSalvataggio *testa)

  Stampa a video l'elenco dei salvataggi disponibili.
- void libera_salvataggi (NodoSalvataggio *testa)

  Libera la memoria allocata per la lista dei salvataggi.
- void carica_salvataggi_da_file (NodoSalvataggio **testa)

  Carica tutti i salvataggi dal file binario all'avvio.

6.19.1 Descrizione dettagliata

Implementazione del sistema di persistenza dei dati.

Definizione nel file salvataggio.c.

6.19.2 Documentazione delle funzioni

6.19.2.1 aggiungi_salvataggio()

```
void aggiungi_salvataggio (
            NodoSalvataggio ** testa,
            Giocatore * g)
```

Aggiunge un nuovo salvataggio alla lista e al file.

Crea un nuovo nodo con i dati attuali del giocatore, genera un nuovo ID, lo aggiunge in testa alla lista e aggiorna il file di salvataggio.

Parametri

testa	Puntatore doppio alla testa della lista.
g	Puntatore ai dati del giocatore da salvare.

Definizione alla linea 44 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, ottieni_data_corrente(), NodoSalvataggio::prossimo, salva_tutto_su_file(), e NodoSalvataggio::timestamp.

Referenziato da mostra_menu_villaggio().

6.19.2.2 carica_salvataggi_da_file()

```
void carica_salvataggi_da_file (
            NodoSalvataggio ** testa)
```

Carica tutti i salvataggi dal file binario all'avvio.

Legge il file "salvataggi.bin" e ricostruisce la lista concatenata in memoria.

Parametri

testa	Puntatore doppio alla testa della lista (verrà inizializzata).

Definizione alla linea 157 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, libera_salvataggi(), NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da main().

6.19.2.3 carica_salvataggio()

```
NodoSalvataggio * carica_salvataggio (
            NodoSalvataggio * testa,
            int id)
```

Cerca un salvataggio nella lista tramite ID.

Parametri

testa	Testa della lista dei salvataggi.
id	ID del salvataggio da cercare.

Restituisce

Puntatore al nodo trovato, o NULL se non esiste.

Definizione alla linea 79 del file salvataggio.c.

Referenzia NodoSalvataggio::id, e NodoSalvataggio::prossimo.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.19.2.4 elimina_salvataggio()

```
void elimina_salvataggio (
            NodoSalvataggio ** testa,
            int id)
```

Elimina un salvataggio dalla lista e dal file.

Rimuove il nodo corrispondente all'ID specificato e riscrive il file binario.

Parametri

testa	Puntatore doppio alla testa della lista.
id	ID del salvataggio da eliminare.

Definizione alla linea 93 del file salvataggio.c.

Referenzia NodoSalvataggio::id, NodoSalvataggio::prossimo, e salva_tutto_su_file().

Referenziato da mostra_menu_salvataggi().

6.19.2.5 libera_salvataggi()

```
void libera_salvataggi (
            NodoSalvataggio * testa)
```

Libera la memoria allocata per la lista dei salvataggi.

Da chiamare alla chiusura del programma.

Parametri

testa	Testa della lista da deallocare.

Definizione alla linea 145 del file salvataggio.c.

Referenzia NodoSalvataggio::prossimo.

Referenziato da carica_salvataggi_da_file(), e main().

6.19.2.6 ottieni_data_corrente()

```
void ottieni_data_corrente (
            char * buffer,
            size_t size)
```

Definizione alla linea 13 del file salvataggio.c.

Referenziato da aggiungi_salvataggio().

6.19.2.7 salva_tutto_su_file()

```
void salva_tutto_su_file (
            NodoSalvataggio * testa)
```

Salva l'intera lista dei salvataggi su file.

Sovrascrive il file "salvataggi.bin" con il contenuto attuale della lista.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 22 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da aggiungi_salvataggio(), elimina_salvataggio(), e gestisci_trucchi().

6.19.2.8 stampa_salvataggi()

```
void stampa_salvataggi (
            NodoSalvataggio * testa)
```

Stampa a video l'elenco dei salvataggi disponibili.

Mostra ID, timestamp e riepilogo delle statistiche per ogni salvataggio.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 124 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, NodoSalvataggio::prossimo, Giocatore::punti_vita, e NodoSalvataggio::timestamp.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.20 salvataggio.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "salvataggio.h"
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <string.h>
00010 #include <time.h>
00011
00012 // Funzione helper per ottenere la data corrente come stringa
00013 void ottieni_data_corrente(char *buffer, size_t size)
00014 {
00015   time_t t = time(NULL);
00016   struct tm tm = *localtime(&t);
00017   // Formato: GG-MM-AAAA HH:MM:SS
00018   snprintf(buffer, size, "%02d-%02d-%04d %02d:%02d:%02d", tm.tm_mday, tm.tm_mon + 1, tm.tm_year +
      1900, tm.tm_hour, tm.tm_min, tm.tm_sec);
00019 }
00020
00021 // Funzione helper per scrivere l'intera lista su file
00022 void salva_tutto_su_file(NodoSalvataggio *testa)
00023 {
00024   FILE *f = fopen("salvataggi.bin", "wb");
00025   if (!f)
00026   {
00027     perror("Errore apertura file salvataggi");
00028     return;
00029   }
00030
00031   NodoSalvataggio *curr = testa;
00032   while (curr)
00033   {
00034     // Scriviamo ID, timestamp e dati giocatore
00035     // Nota: Non scriviamo il puntatore 'prossimo'
00036     fwrite(&curr->id, sizeof(int), 1, f);
00037     fwrite(curr->timestamp, sizeof(curr->timestamp), 1, f);
00038     fwrite(&curr->dati_giocatore, sizeof(Giocatore), 1, f);
00039     curr = curr->prossimo;
00040   }
00041   fclose(f);
00042 }
00043
00044 void aggiungi_salvataggio(NodoSalvataggio **testa, Giocatore *g)
00045 {
00046   NodoSalvataggio *nuovo = (NodoSalvataggio *)malloc(sizeof(NodoSalvataggio));
00047   if (!nuovo)
00048   {
00049     printf("Errore di allocazione memoria per il salvataggio.\n");
00050     return;
00051   }
00052
00053   // Copia i dati del giocatore
00054   nuovo->dati_giocatore = *g;
00055   ottieni_data_corrente(nuovo->timestamp, sizeof(nuovo->timestamp));
00056   nuovo->prossimo = NULL;
00057
00058   // Calcolo ID: Massimo attuale + 1
00059   int max_id = 0;
00060   NodoSalvataggio *curr = *testa;
00061   while (curr)
00062   {
00063     if (curr->id > max_id)
00064       max_id = curr->id;
00065     curr = curr->prossimo;
00066   }
00067   nuovo->id = max_id + 1;
00068
00069   // Inserimento in testa alla lista
00070   nuovo->prossimo = *testa;
00071   *testa = nuovo;
00072
00073   // Persistenza su file
00074   salva_tutto_su_file(*testa);
00075
00076   printf("Partita salvata con successo! (ID: %d)\n", nuovo->id);
00077 }
00078
00079 NodoSalvataggio *carica_salvataggio(NodoSalvataggio *testa, int id)
00080 {
00081   NodoSalvataggio *curr = testa;
00082   while (curr)
00083   {
00084     if (curr->id == id)
```

```
00085     {
00086       return curr;
00087     }
00088     curr = curr->prossimo;
00089   }
00090   return NULL;
00091 }
00092
00093 void elimina_salvataggio(NodoSalvataggio **testa, int id)
00094 {
00095   NodoSalvataggio *curr = *testa;
00096   NodoSalvataggio *prev = NULL;
00097
00098   while (curr)
00099   {
00100     if (curr->id == id)
00101     {
00102       if (prev)
00103       {
00104         prev->prossimo = curr->prossimo;
00105       }
00106       else
00107       {
00108         *testa = curr->prossimo;
00109       }
00110       free(curr);
00111
00112       // Aggiorniamo il file dopo la modifica
00113       salva_tutto_su_file(*testa);
00114
00115       printf("Salvataggio %d eliminato.\n", id);
00116       return;
00117     }
00118     prev = curr;
00119     curr = curr->prossimo;
00120   }
00121   printf("Salvataggio con ID %d non trovato.\n", id);
00122 }
00123
00124 void stampa_salvataggi(NodoSalvataggio *testa)
00125 {
00126   if (!testa)
00127   {
00128     printf("Nessun salvataggio disponibile.\n");
00129     return;
00130   }
00131
00132   NodoSalvataggio *curr = testa;
00133   while (curr)
00134   {
00135     // Formato richiesto: ID. DATA , P. VITA , MONETE , OGGETTI , MISSIONI COMPLETATE
00136     printf("%d. %s , %d P. VITA , %d MONETE , %d OGGETTI , %d MISSIONI COMPLETATE\n",
00137            curr->id, curr->timestamp, curr->dati_giocatore.punti_vita,
00138            curr->dati_giocatore.monete, curr->dati_giocatore.numero_oggetti,
00139            curr->dati_giocatore.missione_palude + curr->dati_giocatore.missione_magione +
    curr->dati_giocatore.missione_grotta);
00140
00141     curr = curr->prossimo;
00142   }
00143 }
00144
00145 void libera_salvataggi(NodoSalvataggio *testa)
00146 {
00147   NodoSalvataggio *curr = testa;
00148   while (curr)
00149   {
00150     NodoSalvataggio *temp = curr;
00151     curr = curr->prossimo;
00152     free(temp);
00153   }
00154 }
00155
00156 // Nuova funzione per caricare dal file all'avvio
00157 void carica_salvataggi_da_file(NodoSalvataggio **testa)
00158 {
00159   FILE *f = fopen("salvataggi.bin", "rb");
00160   if (!f)
00161     return; // File non esistente, nessun salvataggio
00162
00163   // Svuota lista attuale se ce ne fosse bisogno (ma all'avvio è NULL)
00164   libera_salvataggi(*testa);
00165   *testa = NULL;
00166
00167   while (1)
00168   {
00169     NodoSalvataggio *nuovo = (NodoSalvataggio *)malloc(sizeof(NodoSalvataggio));
00170     if (!nuovo)
```

```
00171       break;
00172
00173     // Leggi i 3 campi (ID, timestamp, dati_giocatore)
00174     if (fread(&nuovo->id, sizeof(int), 1, f) != 1)
00175     {
00176       free(nuovo); // Dealloca se fallisce la lettura
00177       break;
00178     }
00179
00180     fread(nuovo->timestamp, sizeof(nuovo->timestamp), 1, f);
00181     fread(&nuovo->dati_giocatore, sizeof(Giocatore), 1, f);
00182
00183     nuovo->prossimo = NULL;
00184     if (*testa == NULL)
00185     {
00186       *testa = nuovo;
00187     }
00188     else
00189     {
00190       NodoSalvataggio *temp = *testa;
00191       while (temp->prossimo)
00192         temp = temp->prossimo;
00193       temp->prossimo = nuovo;
00194     }
00195   }
00196   fclose(f);
00197 }
```

6.21 Riferimenti per il file src/salvataggio.h

Gestione del sistema di salvataggio e caricamento.

```
#include "giocatore.h"
```

Strutture dati

- struct NodoSalvataggio

  Nodo di una lista concatenata per memorizzare i salvataggi.

Ridefinizioni di tipo (typedef)

- typedef struct NodoSalvataggio NodoSalvataggio

  Nodo di una lista concatenata per memorizzare i salvataggi.

Funzioni

- void aggiungi_salvataggio (NodoSalvataggio **testa, Giocatore *g)

  Aggiunge un nuovo salvataggio alla lista e al file.
- NodoSalvataggio * carica_salvataggio (NodoSalvataggio *testa, int id)

  Cerca un salvataggio nella lista tramite ID.
- void elimina_salvataggio (NodoSalvataggio **testa, int id)

  Elimina un salvataggio dalla lista e dal file.
- void stampa_salvataggi (NodoSalvataggio *testa)

  Stampa a video l'elenco dei salvataggi disponibili.
- void carica_salvataggi_da_file (NodoSalvataggio **testa)

  Carica tutti i salvataggi dal file binario all'avvio.
- void salva_tutto_su_file (NodoSalvataggio *testa)

  Salva l'intera lista dei salvataggi su file.
- void libera_salvataggi (NodoSalvataggio *testa)

  Libera la memoria allocata per la lista dei salvataggi.

6.21.1 Descrizione dettagliata

Gestione del sistema di salvataggio e caricamento.

Definizione nel file salvataggio.h.

6.21.2 Documentazione delle ridefinizioni di tipo (typedef)

6.21.2.1 NodoSalvataggio

```
typedef struct NodoSalvataggio NodoSalvataggio
```

Nodo di una lista concatenata per memorizzare i salvataggi.

6.21.3 Documentazione delle funzioni

6.21.3.1 aggiungi_salvataggio()

```
void aggiungi_salvataggio (
            NodoSalvataggio ** testa,
            Giocatore * g)
```

Aggiunge un nuovo salvataggio alla lista e al file.

Crea un nuovo nodo con i dati attuali del giocatore, genera un nuovo ID, lo aggiunge in testa alla lista e aggiorna il file di salvataggio.

Parametri

testa	Puntatore doppio alla testa della lista.
g	Puntatore ai dati del giocatore da salvare.

Definizione alla linea 44 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, ottieni_data_corrente(), NodoSalvataggio::prossimo, salva_tutto_su_file(), e NodoSalvataggio::timestamp.

Referenziato da mostra_menu_villaggio().

6.21.3.2 carica_salvataggi_da_file()

```
void carica_salvataggi_da_file (
            NodoSalvataggio ** testa)
```

Carica tutti i salvataggi dal file binario all'avvio.

Legge il file "salvataggi.bin" e ricostruisce la lista concatenata in memoria.

Parametri

testa	Puntatore doppio alla testa della lista (verrà inizializzata).

Definizione alla linea 157 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, libera_salvataggi(), NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da main().

6.21.3.3 carica_salvataggio()

```
NodoSalvataggio * carica_salvataggio (
            NodoSalvataggio * testa,
            int id)
```

Cerca un salvataggio nella lista tramite ID.

Parametri

testa	Testa della lista dei salvataggi.
id	ID del salvataggio da cercare.

Restituisce

Puntatore al nodo trovato, o NULL se non esiste.

Definizione alla linea 79 del file salvataggio.c.

Referenzia NodoSalvataggio::id, e NodoSalvataggio::prossimo.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.21.3.4 elimina_salvataggio()

```
void elimina_salvataggio (
            NodoSalvataggio ** testa,
            int id)
```

Elimina un salvataggio dalla lista e dal file.

Rimuove il nodo corrispondente all'ID specificato e riscrive il file binario.

Parametri

testa	Puntatore doppio alla testa della lista.
id	ID del salvataggio da eliminare.

Definizione alla linea 93 del file salvataggio.c.

Referenzia NodoSalvataggio::id, NodoSalvataggio::prossimo, e salva_tutto_su_file().

Referenziato da mostra_menu_salvataggi().

6.21.3.5 libera_salvataggi()

```
void libera_salvataggi (
            NodoSalvataggio * testa)
```

Libera la memoria allocata per la lista dei salvataggi.

Da chiamare alla chiusura del programma.

Parametri

testa	Testa della lista da deallocare.

Definizione alla linea 145 del file salvataggio.c.

Referenzia NodoSalvataggio::prossimo.

Referenziato da carica_salvataggi_da_file(), e main().

6.21.3.6 salva_tutto_su_file()

```
void salva_tutto_su_file (
            NodoSalvataggio * testa)
```

Salva l'intera lista dei salvataggi su file.

Sovrascrive il file "salvataggi.bin" con il contenuto attuale della lista.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 22 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da aggiungi_salvataggio(), elimina_salvataggio(), e gestisci_trucchi().

6.21.3.7 stampa_salvataggi()

```
void stampa_salvataggi (
            NodoSalvataggio * testa)
```

Stampa a video l'elenco dei salvataggi disponibili.

Mostra ID, timestamp e riepilogo delle statistiche per ogni salvataggio.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 124 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, NodoSalvataggio::prossimo, Giocatore::punti_vita, e NodoSalvataggio::timestamp.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.22 salvataggio.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef SALVATAGGIO_H
00007 #define SALVATAGGIO_H
00008
00009 #include "giocatore.h"
00010
00014 typedef struct NodoSalvataggio
00015 {
00016   int id;
00017   char timestamp[32];
00018   Giocatore dati_giocatore;
00019   struct NodoSalvataggio *prossimo;
00020 } NodoSalvataggio;
00021
00031 void aggiungi_salvataggio(NodoSalvataggio **testa, Giocatore *g);
00032
00040 NodoSalvataggio *carica_salvataggio(NodoSalvataggio *testa, int id);
00041
00050 void elimina_salvataggio(NodoSalvataggio **testa, int id);
00051
00059 void stampa_salvataggi(NodoSalvataggio *testa);
00060
00068 void carica_salvataggi_da_file(NodoSalvataggio **testa);
00069
00077 void salva_tutto_su_file(NodoSalvataggio *testa);
00078
00086 void libera_salvataggi(NodoSalvataggio *testa);
00087
00088 #endif
```

6.23 Riferimenti per il file src/stanza.h

Definizione della struttura Stanza e dei tipi di stanza.

Strutture dati

- struct Stanza

  Struttura che rappresenta una stanza del dungeon.

Tipi enumerati (enum)

- enum TipoStanza { STANZA_NESSUNA , STANZA_NEMICO , STANZA_TRAPPOLA , STANZA_VUOTA }

  Enumerazione per i tipi di stanza possibili nel dungeon.

6.23.1 Descrizione dettagliata

Definizione della struttura Stanza e dei tipi di stanza.

Definizione nel file stanza.h.

6.23.2 Documentazione dei tipi enumerati

6.23.2.1 TipoStanza

```
enum TipoStanza
```

Enumerazione per i tipi di stanza possibili nel dungeon.

Valori del tipo enumerato

STANZA_NESSUNA	Stanza non inizializzata o non valida
STANZA_NEMICO	Stanza contenente un nemico
STANZA_TRAPPOLA	Stanza contenente una trappola
STANZA_VUOTA	Stanza vuota sicura

Definizione alla linea 12 del file stanza.h.

6.24 stanza.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef STANZA_H
00007 #define STANZA_H
00008
00012 typedef enum
00013 {
00014   STANZA_NESSUNA,
00015   STANZA_NEMICO,
00016   STANZA_TRAPPOLA,
00017   STANZA_VUOTA
00018 } TipoStanza;
00019
00023 typedef struct
00024 {
00025   char nome[32];
00026   TipoStanza tipo;
00027   int danno;
00028   int colpo_fatale;
00029   int ricompensa_monete;
00030   int is_boss;
00031 } Stanza;
00032
00033 #endif
```

6.25 Riferimenti per il file src/utilita.c

Implementazione delle funzioni di utilità.

```
#include "utilita.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
```

Funzioni

- int lancia_dado (int facce)

  Lancia un dado con un specificato numero di facce.
- int padovan (int n)

  Calcola l'n-esimo numero della sequenza di Padovan.
- int controlla_padovan (int num)

  Verifica se un numero appartiene alla sequenza di Padovan.
- void pulisci_schermo ()

  Pulisce lo schermo del terminale.
- char leggi_input_char ()

  Legge un singolo carattere da input.
- int leggi_intero ()

  Legge un numero intero da input.
- int valuta_vittoria_morra (int m1, int m2)

  Valuta il vincitore nella Morra Cinese semplificata.

6.25.1 Descrizione dettagliata

Implementazione delle funzioni di utilità.

Definizione nel file utilita.c.

6.25.2 Documentazione delle funzioni

6.25.2.1 controlla_padovan()

```
int controlla_padovan (
            int num)
```

Verifica se un numero appartiene alla sequenza di Padovan.

Controlla i primi 25 numeri della sequenza.

Parametri

num	Il numero da verificare.

Restituisce

1 se il numero appartiene alla sequenza, 0 altrimenti.

Definizione alla linea 25 del file utilita.c.

Referenzia padovan().

Referenziato da inizia_combattimento().

6.25.2.2 lancia_dado()

```
int lancia_dado (
            int facce)
```

Parametri

facce	Il numero di facce del dado.

Restituisce

Un numero casuale tra 1 e facce.

Definizione alla linea 12 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), genera_stanza_missione(), inizia_combattimento(), e mostra_negozio().

6.25.2.3 leggi_input_char()

```
char leggi_input_char ()
```

Legge un singolo carattere da input.

Pulisce il buffer e gestisce l'input dell'utente, restituendo il carattere in minuscolo.

Restituisce

Il carattere letto, oppure 0 in caso di input vuoto o errore.

Definizione alla linea 50 del file utilita.c.

Referenziato da mostra_menu_principale(), mostra_menu_salvataggi(), e mostra_menu_villaggio().

6.25.2.4 leggi_intero()

```
int leggi_intero ()
```

Legge un numero intero da input.

Gestisce la validazione dell'input richiedendo l'inserimento finché non viene fornito un intero valido.

Restituisce

Il numero intero letto.

Definizione alla linea 76 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e mostra_negozio().

6.25.2.5 padovan()

```
int padovan (
            int n)
```


Calcola l'n-esimo numero della sequenza di Padovan.

Parametri

n	L'indice della sequenza.

Restituisce

L'n-esimo numero di Padovan.

Definizione alla linea 16 del file utilita.c.

Referenzia padovan().

Referenziato da controlla_padovan(), e padovan().

6.25.2.6 pulisci_schermo()

```
void pulisci_schermo ()
```

Pulisce lo schermo del terminale.

Supporta sia sistemi Windows che UNIX-like.

Definizione alla linea 40 del file utilita.c.

Referenziato da esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_principale(), mostra_menu_villaggio(), e mostra_negozio().

6.25.2.7 valuta_vittoria_morra()

```
int valuta_vittoria_morra (
            int mossa1,
            int mossa2)
```

Valuta il vincitore nella Morra Cinese semplificata.

Le regole sono:

- Scudo (1) batte Spada (3)
- Magia (2) batte Scudo (1)
- Spada (3) batte Magia (2)

Parametri

mossa1	La mossa del primo giocatore.
mossa2	La mossa del secondo giocatore.

Restituisce

0 per pareggio, 1 se vince mossa1, 2 se vince mossa2.

Definizione alla linea 91 del file utilita.c.

Referenziato da combattimento_boss_finale().

6.26 utilita.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "utilita.h"
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <ctype.h>
00010 #include <string.h>
00011
00012 int lancia_dado(int facce) { return (rand() % facce) + 1; }
00013
00014 // P(n) = P(n-2) + P(n-3)
00015 // P(0)=1, P(1)=1, P(2)=1
00016 int padovan(int n)
00017 {
00018   if (n < 0)
00019     return 0;
00020   if (n <= 2)
00021     return 1;
00022   return padovan(n - 2) + padovan(n - 3);
00023 }
00024
00025 int controlla_padovan(int num)
00026 {
00027   if (num < 1)
00028     return 0;
00029   for (int i = 0; i < 25; i++)
00030   {
00031     int p = padovan(i);
00032     if (p == num)
00033       return 1;
00034     if (p > num)
00035       return 0;
00036   }
00037   return 0;
00038 }
00039
00040 void pulisci_schermo()
00041 {
00042 #ifdef _WIN32
00043   system("cls");
00044 #else
00045   system("clear");
00046 #endif
00047 }
00048
00049 // Legge un singolo carattere dall'input dell'utente e lo restituisce in minuscolo
00050 char leggi_input_char()
00051 {
00052   char b[100]; // Buffer per memorizzare l'input (100 caratteri max)
00053
00054   // Legge una linea intera da stdin in modo sicuro
00055   if (fgets(b, sizeof(b), stdin))
00056   {
00057     // Rimuove il carattere di newline inserito automaticamente da fgets
00058     b[strcspn(b, "\n")] = 0;
00059
00060     // Se l'utente ha premuto solo invio (input vuoto), restituisce 0
00061     if (strlen(b) == 0)
00062       return 0;
00063
00064     // Se l'utente ha inserito solo uno spazio, lo restituisce
00065     if (strcmp(b, " ") == 0)
00066       return ' ';
00067
00068     // Restituisce il primo carattere convertito a minuscolo
00069     return tolower(b[0]);
00070   }
00071
00072   // Se fgets fallisce, restituisce 0
00073   return 0;
00074 }
00075
00076 int leggi_intero()
00077 {
00078   int valore;
00079   while (scanf("%d", &valore) != 1)
00080   {
00081     while (getchar() != '\n')
00082       ; // Pulisci buffer
00083     printf("Input non valido. Inserisci un numero: ");
00084   }
00085   while (getchar() != '\n')
```

```
00086     ; // Pulisci fine riga
00087   return valore;
00088 }
00089
00090 // 1: Scudo, 2: Magia, 3: Spada
00091 int valuta_vittoria_morra(int m1, int m2)
00092 {
00093   if (m1 == m2)
00094     return 0;
00095
00096   // Scudo(1) > Spada(3)
00097   // Magia(2) > Scudo(1)
00098   // Spada(3) > Magia(2)
00099   if ((m1 == 1 && m2 == 3) || (m1 == 2 && m2 == 1) || (m1 == 3 && m2 == 2))
00100   {
00101     return 1;
00102   }
00103   return 2;
00104 }
```

6.27 Riferimenti per il file src/utilita.h

Funzioni di utilità generale per il gioco.

Funzioni

- int lancia_dado (int facce)

  Lancia un dado con un specificato numero di facce.
- int padovan (int n)

  Calcola l'n-esimo numero della sequenza di Padovan.
- int controlla_padovan (int num)

  Verifica se un numero appartiene alla sequenza di Padovan.
- void pulisci_schermo ()

  Pulisce lo schermo del terminale.
- char leggi_input_char ()

  Legge un singolo carattere da input.
- int leggi_intero ()

  Legge un numero intero da input.
- int valuta_vittoria_morra (int mossa1, int mossa2)

  Valuta il vincitore nella Morra Cinese semplificata.

6.27.1 Descrizione dettagliata

Funzioni di utilità generale per il gioco.

Contiene funzioni per la generazione di numeri casuali, calcoli matematici (sequenza di Padovan), gestione input/output e logica di base per la Morra Cinese.

Definizione nel file utilita.h.

6.27.2 Documentazione delle funzioni

6.27.2.1 controlla_padovan()

```
int controlla_padovan (
            int num)
```

Verifica se un numero appartiene alla sequenza di Padovan.

Parametri

num	Il numero da verificare.

Restituisce

1 se il numero appartiene alla sequenza, 0 altrimenti.

Definizione alla linea 25 del file utilita.c.

Referenzia padovan().

Referenziato da inizia_combattimento().

6.27.2.2 lancia_dado()

```
int lancia_dado (
            int facce)
```

Lancia un dado con un specificato numero di facce.

Parametri

facce	Il numero di facce del dado.

Restituisce

Un numero casuale tra 1 e facce.

Definizione alla linea 12 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), genera_stanza_missione(), inizia_combattimento(), e mostra_negozio().

6.27.2.3 leggi_input_char()

```
char leggi_input_char ()
```

Legge un singolo carattere da input.

Pulisce il buffer e gestisce l'input dell'utente, restituendo il carattere in minuscolo.

Restituisce

Il carattere letto, oppure 0 in caso di input vuoto o errore.

Definizione alla linea 50 del file utilita.c.

Referenziato da mostra_menu_principale(), mostra_menu_salvataggi(), e mostra_menu_villaggio().

6.27.2.4 leggi_intero()

```
int leggi_intero ()
```

Legge un numero intero da input.

Gestisce la validazione dell'input richiedendo l'inserimento finché non viene fornito un intero valido.

Restituisce

Il numero intero letto.

Definizione alla linea 76 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e mostra_negozio().

6.27.2.5 padovan()

```
int padovan (
            int n)
```

Calcola l'n-esimo numero della sequenza di Padovan.

La sequenza è definita come P(n) = P(n-2) + P(n-3) con P(0)=1, P(1)=1, P(2)=1.

Parametri

n	L'indice della sequenza.

Restituisce

L'n-esimo numero di Padovan.

Definizione alla linea 16 del file utilita.c.

Referenzia padovan().

Referenziato da controlla_padovan(), e padovan().

6.27.2.6 pulisci_schermo()

```
void pulisci_schermo ()
```

Pulisce lo schermo del terminale.

Supporta sia sistemi Windows che UNIX-like.

Definizione alla linea 40 del file utilita.c.

Referenziato da esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_principale(), mostra_menu_villaggio(), e mostra_negozio().

6.27.2.7 valuta_vittoria_morra()

```
int valuta_vittoria_morra (
            int mossa1,
            int mossa2)
```

Valuta il vincitore nella Morra Cinese semplificata.

Le regole sono:

- Scudo (1) batte Spada (3)
- Magia (2) batte Scudo (1)
- Spada (3) batte Magia (2)

Parametri

mossa1	La mossa del primo giocatore.
mossa2	La mossa del secondo giocatore.

Restituisce

0 per pareggio, 1 se vince mossa1, 2 se vince mossa2.

Definizione alla linea 91 del file utilita.c.

Referenziato da combattimento_boss_finale().

6.28 utilita.h

Vai alla documentazione di questo file.

```
00001
00010
00011 #ifndef UTILITA_H
00012 #define UTILITA_H
00013
00019 int lancia_dado(int facce);
00020
00021 // Funzioni matematiche / logiche
00022
00032 int padovan(int n);
00033
00042 int controlla_padovan(int num);
00043
00044 // Funzioni input/output
00045
00051 void pulisci_schermo();
00052
00061 char leggi_input_char();
00062
00071 int leggi_intero();
00072
00073 // Logica Morra Cinese (1: Scudo, 2: Magia, 3: Spada)
00074
00087 int valuta_vittoria_morra(int mossa1, int mossa2);
00088
00089 #endif
```